Leren rangschikken: een ROC-gebaseerde grafentheoretische benadering

Learning to rank: a ROC-based graph-theoretic approach

Willem Waegeman

Promotoren: prof. dr. ir. Luc Boullart en prof. dr. Bernard De Baets
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektrische Energie, Systemen en Automatisering
Voorzitter: prof. dr. ir. J. Melkebeek
Faculteit Ingenieurswetenschappen
Academiejaar 2008-2009

ISBN
NUR
Wettelijk depot

Contents

1	Summary / Introduction	9
2	Ranking	13
2.1	Supervised learning	13
2.1.1	Different learning settings	15
2.1.2	Linear models for regression	17
2.1.3	Loss functions and robust estimation	19
2.1.4	Regularization	20
2.1.5	Linear models for classification	22
2.1.6	Multi-class classification	26
2.2	Kernel methods	28
2.2.1	Constructing kernels	28
2.2.2	Support Vector Machines	30
2.3	The foundations of preference modelling	35
2.3.1	From decision making to machine learning	35
2.3.2	Inference from holistic judgements	36
2.3.3	Learning methods versus elicitation procedures	40
2.4	Learning to rank	44
2.4.1	The ordinal regression setting	44
2.4.2	Pairwise preference models	51
2.5	A case study	55
2.5.1	The problem setting	55
2.5.2	Descriptive modelling	58

2.5.3	Predictive modelling	60
2.6	A roadmap through this dissertation	64
3	ROC analysis in ordinal regression	67
3.1	ROC graphs	68
3.1.1	The binary case	68
3.1.2	The area under the ROC curve	70
3.2	Multi-class ROC analysis	72
3.2.1	One-versus-all ensembles	73
3.2.2	Pairwise ensembles	73
3.3	ROC analysis in ordinal regression	74
3.3.1	A natural generalization of the binary case	74
3.3.2	Relationship with other performance measures	79
3.4	Experiments	82
3.4.1	Simulation 1	82
3.4.2	Simulation 2	84
3.4.3	Multi-objective optimization	86
3.5	Conclusion	88
4	Scalability of multi-class ordered ROC analysis	91
4.1	Variance and covariance estimators of VUS	92
4.2	Graph representation and intelligent computation of U-statistics	94
4.2.1	Motivation	94
4.2.2	Notations	94
4.2.3	Algorithm for computing VUS	95
4.2.4	Algorithm for computing the covariance estimator	101
4.2.5	Algorithm for computing the variance estimator	103
4.3	Experiments	112
4.3.1	General setup	112
4.3.2	Results for VUS	114
4.3.3	Results for the variance and covariance estimators	116
4.4	Conclusion	118
5	Ranking learning with VUS as a loss function	121
5.1	Criticism on previous approaches	121
5.2	A convex optimization framework for $\hat{U}(f, D)$	124
5.2.1	VUS as a loss function	125
5.2.2	Reformulation as a structured Support Vector Machine	126
5.2.3	Finding the maximum violating constraint	129
5.2.4	Duality aspects	134

5.2.5	Complexity of the algorithm	138
5.3	Experiments	140
5.3.1	Toy problems	141
5.3.2	Benchmark data sets	145
5.3.3	Scaling experiments	149
5.4	Conclusion	151
6	Ranking representability	153
6.1	Cycle-transitivity	154
6.1.1	Notations	154
6.1.2	Product-transitivity	155
6.1.3	Definition of cycle-transitivity	156
6.1.4	The triangle inequality	158
6.1.5	Fuzzy transitivity	158
6.1.6	Stochastic transitivity	161
6.1.7	Self-dual upper bound functions	164
6.1.8	The probabilistic sum	165
6.2	Links between different types of transitivity	166
6.3	Ranking representability of preference models	170
6.4	Ranking representability of one-versus-one ensembles	174
6.4.1	Strict ranking representability	175
6.4.2	AUC ranking representability	179
6.4.3	Dice transitivity	184
6.4.4	Sufficient conditions for AUC ranking representability	187
6.4.5	Properties of C_{jkl}	197
6.4.6	The cyclic evaluation of pairwise expected ranking accuracy	200
6.4.7	Practical considerations	209
6.5	Conclusion	214
7	Conclusions and further perspectives	217
	Bibliography	223

List of Figures

2.1	An overview of common loss functions for metric regression.	20
2.2	4-fold cross-validation illustrated. The available data is subdivided into 4 bins. One of the bins is held out in each run to test the model, while the remaining bins are used to train it. The held out bins are indicated in red. Reprint from (Bishop 2006).	22
2.3	The hyperplane (thick black line) derived from a linear discriminant function (thin black line) for a two-dimensional input space. Points lying on the wrong side of the hyperplane are incorrectly classified. The magnitude of the corresponding residuals is given by the red lines.	23
2.4	An overview of common loss functions for binary classification.	26
2.5	Three examples of non-linear basis functions and their corresponding kernel functions: polynomials (left), Gaussians (center) and sigmoids (right). The kernels $K(x, x')$ in the bottom figures are plotted as a function of the one-dimensional input x when $x' = 0$. Reprint from (Bishop 2006).	31
2.6	Illustration of the linear support vector machine. The red lines denote slack variables that are different from zero and the data points with circles denote support vectors. The area in between the two thin vertical lines is the margin and the horizontal line represents f	32

2.7	Graphical illustration of the difference between one-versus-one multi-class classification and ordinal regression models. A three-class hypothetical data set is considered.	46
2.8	An example of a proportional odds model with a one-dimensional input and four categories.	48
2.9	An example to illustrate the all-threshold loss function of the support vector ordinal regression method. The red lines denote active slack variables ξ . As shown in the figure, the misclassified red data object gives rise to two active slack variables, since two hyperplanes are situated in between its predicted category and its real category. On the other hand, only one slack variable becomes active for the misclassified green and blue data objects.	49
2.10	A series of box plots giving an overview of the fitted (continuous) outcomes returned by the proportional odds models after backward feature selection. Each box plot shows the variability of the fitted outcomes for an individual class. . . .	59
3.1	The ROC-curve visualized for a hypothetical data set and scoring classifier.	71
3.2	The ROC surface visualized for a three class ordinal regression data set.	78
3.3	Relationship between $\hat{U}(f, D)$ and $\hat{U}_{cons}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.	83
3.4	Relationship between $\hat{U}(f, D)$ and $\hat{U}_{ovo}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.	84
3.5	Relationship between $\hat{U}_{cons}(f, D)$ and $\hat{U}_{ovo}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.	85
3.6	An overview of the 1000 ranking functions randomly drawn from all possible orderings of a balanced data set of 200 objects with 4 categories. The histograms and pairwise scatter plots of the calculated values for $\hat{U}(f, D)$, $\hat{U}_{cons}(f, D)$, $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{pairs}(f, D)$ are shown.	86
3.7	The set of non-dominated solutions aggregated from 20 runs of the MOPSO-algorithm. The six-dimensional Pareto front is plotted as a matrix of two-dimensional scatter plots showing the trade-off for each pair of objectives.	88

4.1	The U -graph for the hypothetical data set and ranking function of Table 4.1.	97
4.2	An example to illustrate q_c -graphs. Here the labels of a hypothetical data set and the ranking imposed by two ranking functions f_1 and f_2 is shown. The data set is sorted according to f_1	105
4.3	The concept of q_c -graphs illustrated on the hypothetical rankings of Figure 4.2. The q_c -graphs $G_{q_c}(f_1, f_2, D_2, \mathcal{Z}_1, \mathcal{Z}_2)$ for all splits of $\mathcal{Y}$ into two disjoint sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are shown. . .	106
4.4	An illustration of the difference between a non-compressed (a) and a compressed (b) q_v -graph. In this example the data set and ranking function of Table 4.2 are used and the split of $\mathcal{Y}$ is as follows: $\mathcal{Z}_1 = \{\bar{y}_1, \bar{y}_4\}$, $\mathcal{Z}_2 = \{\bar{y}_2, \bar{y}_3, \bar{y}_5\}$. In the compressed q_v -graph, edges with weight zero are discarded.	113
4.5	The running time of an exhaustive algorithm for computing $\hat{U}(f, D)$ with an increasing data set size and a different number of classes. The size varied from 60 to 600 instances, adding 60 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.	115
4.6	The running time of Algorithm 1 with an increasing data set size and a different number of classes. The size varied from 60,000 to 600,000 instances, adding 60000 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.	115
4.7	The running time of an exhaustive algorithm for computing $\hat{\sigma}_U^2(f, D)$ with an increasing data set size and a different number of classes. The size varied from 5 to 50 instances, adding at most 5 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.	116
4.8	The running time of Algorithm 2 with an increasing data set size and a different number of classes. The size varied from 5 to 50 instances, adding at most 5 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.	117

4.9	The running time of Algorithm 3 with an increasing data set size and a different number of classes. The size varied from 500 to 5000 instances, adding 500 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.	117
5.1	One of the data sets generated for the toy problem. Each category contains 100 instances in this case.	142
5.2	An overview of the number of iterations required to find the optimal solution for different data set sizes and different values of r	149
5.3	An overview of the average time required to solve the quadratic program (left side) and to find the maximum violating constraint (right side) in each iteration of the algorithm. In both cases the average running time is computed by dividing total time by the number of iterations.	150
6.1	An overview of the links between different types of transitivity. Arrows denote implications from one type to another. The types given in green are specific types of fuzzy transitivity.	169
6.2	An example to illustrate strict ranking representability. In (a) a hypothetical data set is given. In (b) a strictly ranking representable one-versus-one model on that data set is given. The resulting global ranking is given in (c). Finally, in (d) a one-versus-one ensemble $(f'_{12}, f'_{23}, f'_{13})$ that is not strictly ranking representable is obtained by permuting only two elements from the pairwise rankings in (b).	180
6.3	The acyclic graph $G_{strict}(\overline{\mathcal{F}}, D)$ visualized for the pairwise rankings of Figure 6.2(b).	181
6.4	The cyclic graph $G_{strict}(\overline{\mathcal{F}}, D)$ visualized for the pairwise rankings of Figure 6.2(d). The edges in blue form a cycle. . .	181
6.5	An example to illustrate that the ranking obtained from a set of AUC ranking representable ranking functions is not unique. Here the labels of a hypothetical data set and the rankings imposed by three ranking functions f_1, f_2 and f_3 are shown. For these rankings the same pairwise AUC is obtained, i.e. $\hat{A}_{kl}(f_1, D) = \hat{A}_{kl}(f_2, D) = \hat{A}_{kl}(f_3, D)$ with $1 \leq k < l \leq 3$. Two rankings resulting from permutations of two objects of the same category are considered as identical.	183
6.6	The family of functions C_{jkl} visualized.	193

6.7	C_{jkl} compared to T_L with $n_j = n_k = n_l = 10$	201
6.8	C_{jkl} compared to T_P with $n_j = n_k = n_l = 10$	202
6.9	C_{jkl} compared to T_M with $n_j = n_k = n_l = 10$	203
6.10	A graphical illustration of AUC ranking representability generalized to distributions. The distributions of output scores obtained with a one-versus-one model are given on top, those obtained with a single ranking model are given at the bottom. Both models generate the same triplets of pairwise ranking accuracies in this example.	204
6.11	A visualization of the unit square for (β, γ) and the regions for which C_{P0} -transitivity and dice transitivity result in active constraints. In the figure on the left only the shaded region can cause a violation. The figure on the right illustrates that in this region no violation is observed when $n_j = n_k = n_l = 10$	209
6.12	A plot of the first two principal components of the iris data set.	211
6.13	A graphical illustration of the effect of minimizing a magnitude-based loss function with the SVORIM algorithm on the DNA test set. The first two principal components are shown with the real labels on the left side and the predicted labels on the right side. As a result of the transitivity analysis of the pairwise AUCs, the labels of categories $\bar{y}_2$ and $\bar{y}_3$ have been swapped.	214
7.1	A plot of the first two principal components of the <i>satimage</i> data set as an example of a multi-class problem where fitting a full one-versus-one model to the data would not be advisable.	221

List of Tables

2.1	An overview of the main differences between MCDM and machine learning assumptions and their consequences. . . .	41
2.2	The features retained in the final model after backward feature elimination. P -values result from the Wald-statistic. . .	58
2.3	Averages and standard deviations of the performance of the methods under study for ten runs of (nested) cross-validation. Both accuracy and C-index are computed in two ways: with and without considering errors of half a point.	62
2.4	The p -values of pairwise McNemar statistics averaged over 10 runs of nested cross-validation. The asterisks indicate statistically significant differences in accuracy at the 0.05 significance level with a Bonferroni correction for multiple comparisons.	64
3.1	Confusion matrix for a two class classification problem of size n . The rows indicate the true labels and the columns the predicted labels. The 4 table cells represent the 4 situations that can occur in binary classification. Each cell counts the number of times the situation in questions occurs. For example, FP counts the number of negative objects that are predicted positive.	69
4.1	A hypothetical data set of 17 instances. We assume that the data set is sorted according to the ranking function.	96

4.2	Algorithm 1 for the computation of $\hat{U}(f, D)$ demonstrated on a hypothetical data set D_1 and fictive ranking function f_1 . For this data set the unknown ranking function results in 78 of the 432 different r -tuples that are correctly ranked, giving a volume under the ROC surface of 0.181.	100
5.1	A hypothetical example of a data set containing 18 objects with three possible labels to illustrate the inconsistency that can occur when the ranking function is evaluated with pair-wise comparisons. The ordering of the objects in the table represents the obtained ranking, increasing from left to right (example taken from (De Schuymer <i>et al.</i> 2005)).	124
5.2	An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on a series of toy problems of size 400 with different category distributions. P-values originate from paired Wilcoxon rank-sum tests over 50 runs of both algorithms.	144
5.3	An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on seven benchmark data sets with five categories in a balanced setting (20%,20%,20%,20%,20%). P-values originate from paired Wilcoxon rank-sum tests over 10 runs for both algorithms with independent training and test sets and no mutual overlap between training sets.	147
5.4	An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on seven benchmark data sets with five categories in an unbalanced setting (40%,30%,20%,5%,5%). P-values originate from paired Wilcoxon rank-sum tests over 10 runs for both algorithms with independent training and test sets and no mutual overlap between training sets.	148
6.1	A comparison of the performance obtained on the iris data set by ten-fold cross-validation with various multi-class methods and a simple ranking model. The results of the multi-class methods are duplicated from (Hsu & Lin 2002).	211
6.2	A comparison of the performance obtained on the DNA data set (independent test set) with various multi-class methods and a simple ranking model. The results of (Hsu & Lin 2002) are given in the first row and our results in the second row. .	214

List of Symbols

Defined in Chapter 2:

$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathcal{X}$	object space
$\mathbf{x}$	an element of $\mathcal{X}$
$\mathcal{D}$	distribution defined over $\mathcal{X}$
$\mathcal{Y} = \{\bar{y}_1, \dots, \bar{y}_r\}$	ordered or unordered set of labels
r	the number of categories
$\mathcal{D}_k$	distribution of data objects of category $\bar{y}_k$
$D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$	data set of size n
n_k	the number of elements in D that belong to category $\bar{y}_k$
i, j, k, l	indices
m	the dimensionality of $\mathcal{X}$ (in case of a vector space)
m^*	the number of basis functions
ϕ	basis transformation
δ_i	residual of data object $\mathbf{x}_i$
$f : \mathcal{X} \rightarrow \mathbb{R}$	discriminant or ranking function
$h : \mathcal{X} \rightarrow \mathcal{Y}$	decision rule
$\mathbf{w}$	vector of parameters
b, b_k	bias terms or thresholds

List of Symbols

$\ \cdot\ $	L_2 -norm of a vector
C, λ	regularization parameters
$K : \mathcal{X}^2 \rightarrow \mathbb{R}$	kernel function
γ	width of the Gaussian RBF kernel
α, β	Lagrange multipliers
$Q : \mathcal{X}^2 \rightarrow [0, 1]$	reciprocal preference relation
$I : \{\text{false}, \text{true}\} \rightarrow \{0, 1\}$	indicator function
MSE	mean squared error
MAE	mean absolute error

Defined in Chapter 3:

TP	number of true positives
TN	number of true negatives
FP	number of false positives
FN	number of false negative
TPR	true positive rate
FPR	false positive rate
TNR	true negative rate
TPR_k	fraction of correctly classified object of class $\bar{y}_k$
Pr	probability measure
$\hat{a}$	estimator of a
X	random vector / random data object
$f(X)$	random variable
$\mathbf{E}[X]$	expected value
$\bar{F}$	cumulative distribution function
$\bar{f}$	probability density function
$A(f)$	expected ranking accuracy (binary case)
$U(f)$	expected ranking accuracy (general r -category case)
$\hat{A}(f, D)$	area under the ROC curve
$\hat{U}(f, D)$	volume under the ROC surface
$\hat{U}_{pairs}(f, D)$	C-index (1 – pairwise error)
$\hat{U}_{ovo}(f, D)$	multi-class AUC
$\hat{U}_{cons}(f, D)$	averaged AUC on consecutive categories
f_{kl}	bipartite ranking for $\bar{y}_k$ and $\bar{y}_l$
$\bar{\mathcal{F}}$	a set of pairwise rankings f_{kl}

Defined in Chapter 4:

$\sigma_U^2(f)$	variance of VUS
$Cov_U(f_1, f_2)$	covariance of volumes obtained for two ranking functions
$G = (V, E)$	graph with vertex set V and edge set E
$V_k \subset V$	nodes associated with category $\bar{y}_k$
$v \in V$	node / vertex
$e \in E$	edge / arc
v_s, v_e	start node and end node

Defined in Chapter 5:

$\mathbf{j} = (j_1, \dots, j_r)$	r -dimensional vector of indices
$\mathcal{S}$	set of all possible $\mathbf{j}$ in a data set D

Defined in Chapter 6:

α, β, γ	defined for cycle transitivity
$\mathcal{U}$	upper bound function
$\mathcal{L}$	dual lower bound function
Δ	domain of triplets (α, β, γ)
C	conjunctive
$\mathcal{C}$	copula
T	t-norm
T_L	Łukasiewicz t-norm
T_P	product
T_M	minimum
C_{P0}	a specific conjunctive
$\lceil \cdot \rceil$	ceiling function
$\lfloor \cdot \rfloor$	floor function

Samenvatting

In vele situaties vergelijken mensen verschillende objecten om een passend object te selecteren voor een bepaald doel. Denk daarbij bijvoorbeeld aan het kopen van kledij, het beluisteren van muziek, het bestellen van een maaltijd in een restaurant, enz. Constant moeten verschillende objecten tegen elkaar afgewogen worden op basis van criteria zoals geschiktheid, schoonheid of correctheid, enz. In disciplines zoals besluitvorming (Eng. decision making), preferentiemodellering, vage modellering, statistiek en machinaal leren (Eng. machine learning) hebben wetenschappers verschillende manieren bedacht om deze keuzen die door mensen gemaakt worden wiskundig te modelleren. Dit werk kan gesitueerd worden op de rand tussen deze domeinen met bijzondere aandacht voor machinaal leren. In hoofdzaak kunnen twee probleemstellingen onderscheiden worden met betrekking tot het modelleren van menselijke preferentie-informatie: ordinale regressie en paarsgewijze preferentiemodellering. In beide gevallen is een cruciale rol weggelegd voor het concept van een rangschikkingsfunctie. We zullen in dit proefschrift voornamelijk aandacht besteden aan ordinale regressie, waar klassen dikwijls overeenstemmen met quoteringen of linguïstische termen — bijvoorbeeld variërend van “zeer slecht” tot “zeer goed” — die een verschil uitdrukken in correctheid, kwaliteit, schoonheid of een andere eigenschap van de bestudeerde objecten.

ROC-analyse (Eng. receiver operating characteristics analysis) biedt de mogelijkheid om bij binaire classificatieproblemen een optimaal model te selecteren zonder de kost- of klassedistributie vooraf te kennen. Het opper-

vlak onder de ROC-curve (Eng. AUC) kent een grafentheoretische interpretatie en stemt overeen met de Wilcoxon-Mann-Whitney teststatistiek. Deze maat wordt heden ten dage courant gebruikt om binaire classificatiesystemen te beoordelen. Het ligt daarom voor de hand om na te denken over gelijkaardige technieken voor ordinale regressie door analyse van de onderliggende rangschikkingsfunctie. Gegeven deze algemene situering, worden in dit proefschrift de volgende vier doelstellingen geformuleerd:

1. Bestaande prestatiematen voor ordinale regressiemodellen vertonen een aantal ernstige tekortkomingen. Een eerste onderzoeksdoel bestaat daarom uit het ontwerp van alternatieve prestatiematen op basis van ROC-analyse en uit een vergelijking van deze maten met de bestaande prestatiematen.
2. In een tweede onderzoeksdoel willen we de schaalbaarheid van nieuwe prestatiematen analyseren. Aangezien naïeve algoritmen om ROC-schatters te berekenen een hoge graad van complexiteit kennen, loont het de moeite om uit te kijken naar alternatieve implementaties waarbij de schatters efficiënter berekend worden.
3. De kennis die verworven wordt bij de realisatie van de eerste twee doelen kan vervolgens aangewend worden om bestaande leeralgoritmen voor ordinale regressie en rangschikken te verbeteren. Het derde onderzoeksdoel bestaat bijgevolg uit de ontwikkeling van nieuwe leer-algoritmen op basis van ROC-technieken.
4. De onderliggende rangschikkingsfunctie is een essentieel bestanddeel van ordinale regressiemodellen, paarsgewijze preferentiemodellen en classificatiemodellen voor meer dan twee niet-geordende klassen. Als laatste onderzoeksdoel analyseren we de connecties tussen deze modellen en bekijken we de voorwaarden waarvoor ze samenvallen.

Hoofdstukken 3 tot 6 geven elk een gedetailleerde weergave van de realisatie van één van deze vier objectieven. Na een algemene inleiding tot rangschikken in Hoofdstuk 2, start Hoofdstuk 3 met een uitbreiding en vergelijking van ROC-gebaseerde prestatiematen voor ordinale regressie. Zoals voor classificatieproblemen met meer dan twee niet-geordende klassen, breiden we het oppervlak onder de ROC curve uit tot het volume onder een r -dimensionaal oppervlak (VUS) voor r geordende klassen. Door het gebruik van VUS wordt niet gekeken naar het aantal verkeerd geclassificeerde objecten, maar in plaats daarvan wordt de rangschikking beoordeeld die

bekomen wordt door een ordinaal regressiemodel. Een dergelijke aanpak biedt vooral voordelen bij niet-uniforme kost- of klassedistributies. We geven een theoretische en experimentele bespreking van de voordelen en het afwijkend gedrag van VUS ten opzichte van accuraatheid, de gemiddelde absolute fout (Eng. MAE) en andere prestatiematen die gebaseerd zijn op de verkregen rangschikking. De resultaten tonen aan dat het maximaliseren van de accuraatheid of het minimaliseren van MAE niet noodzakelijk ordinale regressiemodellen met een goede bijhorende rangschikking tot stand brengt. Omdat de berekening van VUS en de bijhorende schatters van variantie en covariantie een zware computationele kost vergen, stellen we vervolgens in Hoofdstuk 4 nieuwe algoritmen voor om deze schatters efficiënt berekenen. We tonen aan dat VUS snel kan geëvalueerd worden met een algoritme dat gebaseerd is op dynamisch programmeren en gedomineerd wordt door een sorteerbewerking op de dataset. Daarnaast herleiden we de schatters voor variantie en covariantie tot een reeks van recurrente functies die gedefinieerd zijn over gelaagde (data)grafan en vervolgens worden deze functies snel berekend met behulp van dynamisch programmeren. Simulaties tonen aan dat de voorgestelde algoritmen een goede schaling vertonen in functie van de grootte van de dataset en het aantal klassen. Het volume onder het ROC oppervlak kon bijvoorbeeld berekend worden voor datasets met meer dan 500,000 objecten, terwijl een naïeve implementatie meer tijd vergde voor datasets van slechts 1000 objecten.

Steunend op de grafentheoretische concepten die geïntroduceerd werden in het vorige hoofdstuk, beschrijven we in Hoofdstuk 5 hoe VUS aangewend kan worden als een verliesfunctie voor een rangschikkingsalgoritme. Dit leidt tot een nieuwe kernelmethode die gebaseerd is gestructureerde support vector machines. We maximaliseren in dit geval het aantal correct gerangschikte sequenties van één object van elke klasse in plaats van te kijken naar correct gerangschikte paren. In het corresponderend kwadratisch programma duikt een groot aantal beperkingen op, maar de optimale oplossing kan voor datasets van grootte n gevonden worden met tijdscomplexiteit $\mathcal{O}(n^3)$ door middel van een cutting plane algoritme en grafentheoretische technieken. Deze aanpak kan voordelen bieden in diverse toepassingsdomeinen. Op gebalanceerde en ongebalanceerde referentiedatasets werden betere prestaties geobserveerd ten opzichte van de traditionele aanpak. Daarnaast bevestigden tijdsmetingen de computationele complexiteit die theoretisch afgeleid werd.

Tenslotte leggen we in Hoofdstuk 6 de link tussen rangschikkingsmodellen, paarsgewijze preferentiemodellen en één-vs-één classificatiemodellen voor

meer dan twee klassen. In deze analyse is een cruciale rol weggelegd voor het concept cykeltransitiviteit als voorwaarde om een paarsgewijs preferentiemodel of een één-vs-één classificatiemodel voor te stellen door middel van een rangschikking. Voor paarsgewijze preferentiemodellen vormen de verschillende soorten van stochastische transitiviteit de essentiële bouwstenen die de verwantschap uitdrukken en aanleiding geven tot de concepten zwakke, gemiddelde en sterke representeerbaarheid door middel van een rangschikking. Voor classificatiemodellen met meerdere klasse is de analyse iets complexer en wordt er teruggегреpen naar ROC-maten en hun grafentheoretische interpretaties om nodige en voldoende voorwaarden af te leiden voor de representeerbaarheid van deze modellen met behulp van één enkele rangschikking. Voor drie klassen geeft dit aanleiding tot een nieuw type van cykeltransitiviteit door analyse van de paarsgewijze AUC's. Deze vorm van cykeltransitiviteit wordt geverifieerd door het oplossen van een kwadratisch programma met gehele beperkingen (Eng. integer quadratic program). Dit kan echter vermeden worden omdat de optimale oplossing van dit kwadratisch programma in de limiet (d.w.z. voor datasets van oneindige grootte) convergeert naar een eenvoudige voorstelling. Bijgevolg is de afwijking die bekomen wordt door deze limietvoorstelling te gebruiken begrensd. Bijkomend bespreken we enkele empirische resultaten die de toepassingsmogelijkheden aantonen van deze theoretische observaties.

Ter afsluiting vormen we een algemeen besluit in Hoofdstuk 7 en geven we enkele ideeën voor verder onderzoek.

Summary / Introduction

In many situations, humans are used to compare items or objects in order to choose the appropriate item for a specific goal. Think for example of buying clothes, listening to music, the dish one orders in a restaurant, etc.. Continually, we evaluate objects on criteria such as appropriateness, beauty, correctness, etc. In research areas like decision making, preference modelling, fuzzy modelling, statistics and machine learning, scientists have proposed various ways to characterize this human behavior with mathematical models. This work can be located on the edge between those domains with strong emphasis on the machine learning aspects. Mainly two learning settings can be distinguished for modelling preference information: ordinal regression models and pairwise preference models. Both of them strongly depend on the concept of an underlying ranking function. We primarily concentrate on the ordinal regression setting, where categories typically correspond to quotations or linguistic terms — varying from “very bad” to “brilliant” for example — that express a difference in correctness, quality, beauty or any other characteristic of the analyzed objects.

Receiver operating characteristics (ROC) curves provide a way to select possibly optimal models for discriminating two kinds of objects without the need of specifying the cost or class distribution. The area under the ROC curve (AUC) has a graph-theoretic interpretation and corresponds to the Wilcoxon-Mann-Whitney test statistic. It is nowadays commonly used as performance measure for evaluating binary classifiers. As a consequence, it is straightforward to think of introducing similar techniques for ordinal regression prob-

lems as well, by investigating the underlying ranking function. Given this general scope, the following four research objectives are outlined:

1. In the field it is known that existing measures for evaluating ordinal regression models suffer from a number of important shortcomings. Because of that, a first research objective is the development of alternative measures by extending existing ROC measures for classification to ordinal regression problems and by investigating whether these measures can compete with the existing ones.
2. A second research objective concerns the analysis of the scalability of ROC-based measures for ordinal regression. Since naive algorithms for computing ROC estimators become computationally intractable for large samples, it is worthwhile to seek for alternative algorithms that overcome this computational bottleneck.
3. Subsequently, the knowledge that is gained to establish these two objectives can be further utilized to enhance existing machine learning algorithms for ordinal regression and ranking. Therefore, the third research objective concerns the development of new learning algorithms that incorporate ROC-techniques.
4. From the underlying ranking function follows that ordinal regression is very related to multi-class classification and pairwise preference learning. For the last objective, we analyze the relationship between these three problem settings. In particular, we aim to discover conditions for which they coincide.

Chapters 3 to 6 each tackle one of these four research objectives. After a general introduction to ranking in Chapter 2, we start in Chapter 3 with an extension and comparison of ROC-based performance measures for ordinal regression. Similar to multi-class classification problems, we generalize the AUC and its underlying probabilistic interpretation to ordinal regression problems such that it now corresponds to the volume under an r -dimensional surface (VUS) for r ordered categories. VUS rather evaluates the ranking returned by an ordinal regression model instead of measuring the error rate. This is a way of thinking which is especially advantageous in case of skew class or cost distributions. We give theoretical and experimental evidence of the advantages and different behavior of VUS compared to error rate, mean absolute error and other ranking-based performance measures for ordinal regression. The results demonstrate that the models produced by ordinal re-

gression algorithms minimizing the error rate or a preference learning based error function not necessarily impose a good ranking on the data.

The computation of VUS as well as the U-statistics estimators of its variance and covariance for two models is believed to be complex. That's why we analyze its scalability in Chapter 4, where new algorithms to compute VUS and its (co)variance estimator are presented. In particular, the volume under the ROC surface can be found very efficiently with a simple dynamic program dominated by a single sorting operation on the data set. For the variance and covariance, the respective estimators are reformulated as a series of recurrent functions over layered data graphs, and subsequently, these functions are rapidly evaluated with a dynamic program. Simulation experiments confirm that the presented algorithms scale well with respect to the size of the data set and the number of categories. For example, the volume under the ROC surface could be rapidly computed on very large data sets of more than 500,000 instances, while a naive implementation spent much more time on data sets of size less than 1000.

In Chapter 5 we discuss how VUS can be embedded as a loss function in a learning algorithm, unlike the conventional approach of minimizing the pairwise error. This leads to a new type of kernel method based on structured support vector machines. This method tries to optimize the fraction of correctly ranked r -tuples. A large number of constraints appears in the resulting quadratic program, but the optimal solution can be computed in $\mathcal{O}(n^3)$ time for samples of size n with a cutting plane algorithm and graph-based techniques. Our approach can offer benefits for applications in various domains. On various synthetic and benchmark data sets it outperforms the pairwise approach for balanced as well as unbalanced problems. In addition, scaling experiments confirm the theoretically derived time complexity.

Finally, in the more theoretical Chapter 6, we go one step further and investigate how reciprocal pairwise preference models and one-versus-one multi-class ensembles can be represented as ranking models. Not surprisingly, transitivity turns out to be the crucial concept for ranking representability of both types of models. We start by introducing the general framework of cycle transitivity and reciting the various types of transitivity it covers. Then, we establish the links between ranking representability and transitivity for pairwise preference models by unravelling the relationships between the concepts of weak, moderate and strong representability and their corresponding forms of stochastic transitivity. In the second part of Chapter 6, we study the related and more complex problem of ranking representability of multi-class classification problems. To this end, the ROC concepts and graph formulations

of Chapter 4 are extended in order to derive necessary and sufficient conditions for ranking representability of one-versus-one multi-class ensembles. For the three class case, this results in a new type of cycle transitivity for pairwise AUCs that can be verified by solving an integer quadratic program. Moreover, solving this integer quadratic program can be avoided, since its solution converges for an infinite data sample to a simple form, resulting in a deviation bound that becomes tighter with increasing sample size. We also present empirical results that demonstrate the usefulness of these theoretical observations.

In Chapter 7 we summarize the general conclusions and present some ideas for future research.

This chapter introduces some basic concepts that will be necessary to understand further chapters. We start in Section 2.1 with a general introduction to supervised machine learning, including an overview of different learning settings and a discussion of common techniques for regression and classification. Subsequently, kernel methods are described more elaborately in Section 2.2 with particular emphasis on support vector machines. Then, in Section 2.3 a general introduction is given to preference modelling, in which we outline the main differences between machine learning and decision making approaches for representing human preferences. Since this thesis will mainly consider preference modelling from a machine learning perspective, we pay more attention to the learning aspects in Section 2.4 and make a subdivision between ordinal regression and pairwise preference models. We also clarify the important role of the concept of a ranking function that underlies these two models. To give the reader more insight into the practical aspects of preference learning, an introductory case study is presented in Section 2.5, followed by an overview of this dissertation in Section 2.6.

2.1 Supervised learning

Balancing on the edge between statistics, data mining and artificial intelligence, machine learning is a relatively broad field concerned with the analysis and development of algorithms to detect trends in data. Unlike statistics, the focus in machine learning is mainly on a predictive analysis of the data, which means that we aim to fit a model with a good predictive power to the

data, i.e. we want to construct a model capable of learning trends from the data. A large number of applications can be included in this basic problem setting. Some examples are:

- Computer vision: automatic detection of human beings in images for surveillance tasks.
- Bioinformatics: localization of genes in DNA-sequences that affect certain characteristics of living beings.
- Medical decision making: estimating the probability of survival of patients suffering from a certain disease.
- Weather forecasting: prediction of the temperature, rainfall and other weather parameters in the short- and long-term.
- Fraud detection: the automatic generation of warning signals for potentially fraudulent money transfers.
- Market intelligence: estimation of the expected sales of a newly developed product that will be brought on the market.

In all of these examples, we aim to represent the available data in a statistical way such that the resulting model gives satisfactory predictions. In order to construct such a predictive model, data can be sampled in different ways. Depending on the structure of the incoming data, machine learning problems are usually first subdivided into two main categories: *supervised* and *unsupervised* learning. In supervised learning, the incoming data contains feedback information or labelled examples for which we already know the prediction. Referring to the problem settings listed above, this feedback information could be:

- Images that have been screened by a person who marked them if a person was part of the image.
- A list of genes that have been detected during previous research.
- A database with information of persons who died or recovered from the disease under study.
- The complete weather survey of the last ten years.
- A list of fraudulent money transfers that have been detected in the past.
- Feedback data about related products that have been launched recently.

Conversely, in unsupervised learning, this kind of feedback is not available and a model must be constructed solely on the basis of unlabelled data.

2.1.1 Different learning settings

The different learning settings that we will consider in this dissertation are all supervised learning settings. Before we make a further subdivision of supervised learning problems, let us first introduce some basic mathematical notations. In general, $\mathcal{X}$ will denote the object space. It can be a finite or infinite space and for its elements the notation $\mathbf{x}$ will be used. These elements can be vectors of features or more complex structures like DNA-sequences, images, trees, strings, etc. In the case of an m -dimensional vector space, we will use the superscript notation

$$\mathbf{x} = (x^{(1)}, \dots, x^{(m)}), \quad (2.1)$$

to denote the individual elements. For supervised learning, we also consider a second space in addition to $\mathcal{X}$. This space $\mathcal{Y}$ represents the set of possible feedback information. The precise structure of $\mathcal{Y}$ gives rise to a further subdivision of supervised learning problems:

- **Binary classification:** two different labels can be associated with any data object. In the computer vision example mentioned above, there are two possible choices: a human being is present or absent in a certain image. To this end, we introduce the notation $\mathcal{Y} = \{\bar{y}_-, \bar{y}_+\}$. The two categories of objects are referred to as the negative and positive class.
- **Multi-class classification:** a direct extension of the binary case in which now in general r categories are observed. To this end, we extend the notation of the binary classification to $\mathcal{Y} = \{\bar{y}_1, \dots, \bar{y}_r\}$ with $\mathcal{Y}$ an unordered set.
- **(Metric) regression:** the variable to predict is a continuous value, i.e. $\mathcal{Y} = \mathbb{R}$. In the weather forecast example, prediction of the rainfall is done in milliliter per day.
- **Ordinal regression:** this setting can be located somewhere in between metric regression and multi-class classification, since a discrete ordered set of labels is considered. In the weather forecast example, wind strength is measured on the Beaufort scale, a scale divided into a series of 13 values, from 0 for calm winds to 12 for hurricanes. Like multi-class classification, we use the notation $\mathcal{Y} = \{\bar{y}_1, \dots, \bar{y}_r\}$, but now there is a linear order relation $<_{\mathcal{Y}}$ defined on $\mathcal{Y}$ such that

$$\bar{y}_1 <_{\mathcal{Y}} \dots <_{\mathcal{Y}} \bar{y}_r.$$

- **Pairwise preference learning:** given a couple $(\mathbf{x}_1, \mathbf{x}_2)$ of data objects, the learning system needs to decide whether $\mathbf{x}_1$ is preferred to $\mathbf{x}_2$ or not.

In the market intelligence example mentioned above, producers could be interested to know whether their products are able to compete with the products of contending producers.

- Ranking: given a set $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ of data objects, the goal consists of learning a linear or partial order relation for the set. As discussed in Section 2.4, ranking bears a strong resemblance with ordinal regression and pairwise preference learning.
- Multi-label learning: each instance can be associated with a set of labels.
- Structured learning: $\mathcal{Y}$ is an even more complex space. In this case, we aim to predict structures, like a tree, a graph, a string, etc.

In general, one usually assumes in machine learning that the observed data is sampled according to an unknown joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. We will use the notation

$$D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}, \quad (2.2)$$

to denote a data sample of size n . In the course of this dissertation, we will always assume that the observed data is identically and independently distributed (i.i.d.), which is a weak assumption (e.g. much weaker than the conditions imposed on the data by many statistical models). Nevertheless, in some occasions, the i.i.d. assumption might be too restrictive. In the weather forecast application, this is for example the case, since the observed data on day t largely depend on the data of the foregoing days. Specific machine learning algorithms like Markov models and dynamical systems exist for handling this kind of problems. Yet, we will only consider learning problems where the i.i.d. assumption is not violated. Unless stated clearly in the text, no further restrictions are put on the data.

In the remainder of Section 2.1, we briefly explain some basic techniques for supervised learning. To this end, we start with a summary of linear and non-linear models for binary classification and metric regression, by far the most studied supervised learning settings. We also discuss some fundamental concepts of learning, like regularization, the bias-variance trade-off and robust estimation. At the end of the section, we extend the binary case to multi-class classification. Many of the concepts summarized here are more thoroughly described in the books (Hastie *et al.* 2001, Bishop 2006). We refer to these works for a further discussion.

2.1.2 Linear models for regression

Historically, the linear model has played a crucial role in statistical modelling, and still it remains an important tool in any statistical toolbox. Formally, we will represent a linear model as

$$f(\mathbf{x}) = \sum_{k=1}^m w_k x^{(k)} + b = \mathbf{w} \cdot \mathbf{x} + b, \quad (2.3)$$

with $\mathbf{w}$ a vector of parameters and b a bias term, such that $f : \mathcal{X} \rightarrow \mathbb{R}$. By including the constant 1 as last component into the vector $\mathbf{x}$, we can simplify this model to

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}.$$

In the case of regression, the function $f : \mathcal{X} \rightarrow \mathbb{R}$ can be used to predict the outcome of any data object from $\mathcal{X}$. Yet, in order to have realistic predictions, we will need to infer the model based on *training data*, i.e. a sample like (2.2) containing feedback information. Using this training sample, the least squares method is a simple but often fairly accurate way to find good values for the parameter vector $\mathbf{w}$ by minimizing the mean squared error:

$$\text{MSE}(f, D) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2.$$

Since the mean squared error forms a convex objective function, the minimum can be found efficiently, in an analytic way by matrix inversion. In general, $\hat{\mathbf{w}}$ or $\hat{f}$ will represent the solution and $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ will denote the predicted value for object $\mathbf{x}_i$.

Although the linear regression model presented in (2.3) can be interpreted easily, it has important limitations for prediction since it only considers linear functions of the input variables. A straightforward extension can be obtained by taking linear combinations of non-linear functions of the input variables, leading to a model of the form

$$f(\mathbf{x}) = \sum_{k=1}^{m^*} w_k \phi_k(\mathbf{x}) + b = \mathbf{w} \cdot \phi(\mathbf{x}) + b, \quad (2.4)$$

The non-linear functions $\phi_k : \mathcal{X} \rightarrow \mathbb{R}$ are usually called basis functions. In general, m^* basis functions are considered (with usually $m^* > m$) such that the vector $\mathbf{w}$ contains m^* free parameters that have to be inferred on the

basis of training data. In addition, the basis functions can also contain free parameters. In (2.4), we also introduced $\phi : \mathcal{X} \rightarrow \mathbb{R}^{m^*}$ as a mapping of data objects to an m^* -dimensional vector space. This space will be further called the feature space and the values of the individual basis functions will be called features.

When non-linear basis functions are used, the model becomes a non-linear function of the input vector $\mathbf{x}$. Despite this non-linearization, the good properties of the linear regression model like interpretability and ease of optimization are preserved. For a one-dimensional input variable x , a well-known example of a non-linear basis function model is the polynomial regression model for which $\phi_k(x) = x^k$. In this way, all polynomials up to degree m^* are included into the model. Alternative representations for the basis functions are for example Gaussian basis functions, i.e.

$$\phi_k(x) = \exp\left(-\frac{(x - \mu_k)^2}{2s^2}\right),$$

or sigmoidal basis functions, i.e.

$$\phi_k(x) = \sigma\left(\frac{x - \mu_k}{s}\right),$$

with σ the logistic sigmoid function and μ_k and s free parameters. All three types of basis functions are visualized in Figure 2.5 and will be further examined in the context of kernel methods. When more than one input variable is measured, basis functions that put together different input variables are also commonly utilized. In this way interactions between different input variables can be investigated.

In complex pattern recognition applications, the composition of basis functions is often carried out in a pre-processing phase prior to model inference. Finding relevant features for a given task often requires a lot of background knowledge about the application domain and is therefore better completed by someone familiar with this domain. Areas like image analysis, bioinformatics, signal processing, financial forecasting, etc, all have specific methods to derive appropriate basis functions. Undeniably, for many applications, the predictive power of the machine learning model obtained in the end heavily depends on the relevance of the features derived during the pre-processing phase.

2.1.3 Loss functions and robust estimation

In statistics it is well established that least squares minimization can be interpreted as maximum likelihood estimation when the error of the model follows a normal distribution. More specifically, it is then required that the errors or residuals δ_i defined by

$$\delta_i = y_i - f(\mathbf{x}_i),$$

are normally distributed. Unfortunately, this assumption is far too restrictive for real-world machine learning applications. Often much longer tails in the error distribution are observed due to uncontrollable factors like noise, irrelevant features and incomplete information. It has been observed that least squares estimation can break down completely in the presence of noise, since training errors on noisy instances are too much penalized. A more robust estimation can be obtained by minimizing other types of loss functions. Let us introduce the notation

$$L(f, D) = \frac{1}{n} \sum_{i=1}^n \rho(f(\mathbf{x}_i), y_i), \quad (2.5)$$

to denote a general instance-wise decomposable loss function, then examples of more robust loss functions are the mean absolute error (MAE) defined by

$$\rho(f(\mathbf{x}), y) = |y - f(\mathbf{x})|,$$

the Huber robust loss defined by

$$\rho(f(\mathbf{x}), y) = \begin{cases} \frac{1}{2}(y - f(\mathbf{x}))^2, & \text{if } |y - f(\mathbf{x})| \leq \epsilon, \\ \epsilon|y - f(\mathbf{x})| - \frac{1}{2}\epsilon^2, & \text{otherwise,} \end{cases}$$

and the ϵ -insensitive loss (Vapnik 1995) defined by

$$\rho(f(\mathbf{x}), y) = \begin{cases} 0, & \text{if } |y - f(\mathbf{x})| \leq \epsilon, \\ |y - f(\mathbf{x})| - \epsilon, & \text{otherwise,} \end{cases}$$

with ϵ an additional parameter. These three robust loss functions give rise to much lower penalties on outlying residuals than the MSE, since the loss increases linearly with the magnitude of the absolute residuals (as illustrated in Figure 2.1).

Furthermore, it must be emphasized that the loss function minimized within a machine learning algorithm and the loss or error function employed to evaluate the algorithm in the end are usually not the same. For robustness and

Figure 2.1: An overview of common loss functions for metric regression.

for computational advantages, one can argue to minimize a loss other than the least squares in a regression task, but the algorithm is afterwards still evaluated by means of the MSE. For example, for the ϵ -support vector regression (ϵ -SVR) method, Vapnik's ϵ -insensitive loss is used internally, since it accomplishes the sparsity, robustness and generalization capabilities of the inferred model. Nonetheless, it does not make sense to apply this loss function to evaluate the model on test data, as residuals lying in the tube $[-\epsilon, \epsilon]$ are not penalized at all. In general, the procedure of utilizing a different loss function to fit and to evaluate a machine learning algorithm is also applied to other supervised learning tasks.

2.1.4 Regularization

The more complex a class of models is, the lower the final error on training becomes when the training error is minimized. In the extreme case, when the model contains more free parameters than training instances, the training error reduces to zero and the regression function becoming an interpolating curve. Yet, does the resulting model have a satisfying predictive power? Of course not, since it will be a very unsmooth model that generalizes badly to points outside the training data set. This phenomenon is called overfitting.

In machine learning, regularization refers to the set of methods for preventing overfitting. Tikhonov regularization is the most commonly used type, in

which one aims to solve the following general minimization problem

$$\min_{f \in \mathcal{F}} L(f, D) + \lambda J(f). \quad (2.6)$$

$L(f, D)$ denotes a general loss or error function, of which the mean squared error is an example, and J represents a penalty functional that quantifies the complexity of the function f . So, we aim to find the function f that minimizes a weighted sum of two objectives over a certain hypothesis space $\mathcal{F}$ of functions. These objectives are usually conflicting since a lower error can be obtained with a complex model. The parameter λ that has to be fine-tuned defines the trade-off between both objectives. A well-known example of regularization in statistics is ridge regression, in which the mean squared error is minimized and the L_2 -norm of $\mathbf{w}$ is used as regularizer, i.e.

$$\min_{f \in \mathcal{F}} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

Many machine learning algorithms like kernel methods and neural networks employ this kind of regularizer to prevent overfitting. In statistics, regularized models are often referred to as shrinkage methods, while regularization is known as weight decay in the neural network literature.

Within a regularized model, the trade-off parameter λ controls the effective complexity of a model. From a frequentist viewpoint, the functioning of it can be statistically explained by means of the bias-variance trade-off. For a low value of λ , minimization of the training error becomes the most decisive objective in the bi-objective optimization problem. It can lead to overfitting on the training data, as the model can be fairly complex in proportion to the number of training instances. So, the sample size is in that case relatively small compared to the expected complexity of the model, resulting in a high variance of the estimated parameters when different training sets would be sampled from the underlying unknown distribution. At the same time, the bias of the estimated parameters is low since the model is complex enough to catch less obvious trends in the data. On the other hand, for a high value of λ , the opposite conclusions can be drawn. The regularizer then becomes the most decisive objective, potentially causing underfitting on the training sample. If the model is too simple to catch less obvious trends in the data, then it will also manifest a bad generalization performance (i.e. good predictions for data objects not occurring in the training set). When the model is too simple, one speaks of underfitting.

In essence, the best trade-off between bias and variance must be found to achieve a good generalization performance. Since λ defines this trade-off,

Figure 2.2: 4-fold cross-validation illustrated. The available data is subdivided into 4 bins. One of the bins is held out in each run to test the model, while the remaining bins are used to train it. The held out bins are indicated in red. Reprint from (Bishop 2006).

a procedure must be set up to detect its optimal value¹. Cross-validation is such a procedure to fine-tune λ by means of a line search. In a nutshell, cross-validation subdivides the available data into a number of bins, let's say k for k -fold cross-validation. Repeatedly, $k - 1$ bins serve as input to train the model and the remaining bin is left out to test it. By averaging the performance on the test set over the k different models by leaving out the k -th bin, the validation error is computed for a certain value of λ . By repeating this procedure for different values of λ , we can find the optimal value within the investigated range. This value is then selected to train the final model on the entire data set. Cross-validation is graphically illustrated in Figure 2.2. It is usually preferred over similar techniques like single train-test splitting or bootstrapping.

We briefly remark that many other methods have been proposed that attempt to prevent overfitting by adding a penalty term, like Akaike's information criterion (AIC), the minimum descriptor length (MDL) and the Bayesian information criterion (BIC). Since these approaches will not pop up further within this dissertation, we do not discuss them and refer the interested reader to the introductory works mentioned above.

2.1.5 Linear models for classification

Things are slightly different in the case of classification. Contrary to metric regression, the goal in classification is to assign data objects to one of a fixed number of categories (or classes). Many famous machine learning methods for classification are restricted to binary classification, where only

¹In more complex methods, additional parameters can also control the complexity of the model and these parameters must be fine-tuned as well.

Figure 2.3: The hyperplane (thick black line) derived from a linear discriminant function (thin black line) for a two-dimensional input space. Points lying on the wrong side of the hyperplane are incorrectly classified. The magnitude of the corresponding residuals is given by the red lines.

two categories are considered. This section gives an overview of some basic algorithms for binary classification. The extension to multiple categories will be discussed in the following section.

So, we have two possible categories for which the notation $\bar{y}_-$ and $\bar{y}_+$ is chosen. We will call them respectively the negative and the positive class. Let us try to adopt the linear regression model for binary classification tasks, which means that we now have to infer a function of the form $h : \mathcal{X} \rightarrow \{\bar{y}_-, \bar{y}_+\}$. By using the linear regression model, this can be realized as follows:

$$h(\mathbf{x}) = \begin{cases} \bar{y}_-, & \text{if } f(\mathbf{x}) < 0, \\ \bar{y}_+, & \text{if } f(\mathbf{x}) \geq 0, \end{cases}$$

in which $f : \mathcal{X} \rightarrow \mathbb{R}$ can be a function of the form (2.3) or (2.4), similar to the regression setting. In the context of classification, f is often referred to as a discriminant function, since it defines the separation rule to classify a data object into one of the two categories. The decision boundary between the two categories is defined by the equation

$$f(\mathbf{x}) = 0. \quad (2.7)$$

For models of the form (2.3) or (2.4), this leads to a linear decision boundary in respectively the input space or the feature space spanned by basis functions. Geometrically, the decision boundary takes the form of a hyperplane, perpendicular to the direction defined by f . For a two-dimensional input space, this is visualized in Figure 2.3.

In many applications of binary classification, the goal simply consists of inferring a model that makes as few misclassification errors as possible. Adopting the notation of (2.5) automatically leads to the following loss function:

$$\rho(h(\mathbf{x}), y) = I_{h(\mathbf{x}) \neq y},$$

with I the indicator function returning 1 when its argument (written as subscript) is true and zero otherwise. This error function is called the zero-one error function. Equivalently to minimizing the zero-one error, one can think of maximizing accuracy or the number of correctly classified training objects:

$$\text{accuracy} = \sum_{i=1}^n I_{h(\mathbf{x}_i)=y_i}. \quad (2.8)$$

The latter way of thinking is often more convenient for classification. Regrettably, accuracy suffers from serious shortcomings when used internally within a supervised learning algorithm. Maximizing accuracy leads to a combinatorial optimization problem, since the corresponding objective function is discrete and not differentiable. Unlike least-squares estimation for regression, no numerical algorithm exists that finds the exact optimal solution within short amount of time for reasonably sized data sets. Instead of employing heuristic algorithms, one predominantly optimizes a more smooth loss function that reflects the zero-one loss in some way. In order to get a continuous loss function, one can take the distance to the decision boundary into account, leading to residuals as defined by the red lines in Figure 1.3. Let us introduce the notation

$$y'_i = \begin{cases} +1, & \text{if } y_i = \bar{y}_+, \\ -1, & \text{if } y_i = \bar{y}_-, \end{cases}$$

then we can define the residuals for classification by

$$\delta_i = \begin{cases} y'_i f(\mathbf{x}_i), & \text{if } y_i \neq h(\mathbf{x}_i), \\ 0, & \text{otherwise.} \end{cases} \quad (2.9)$$

Basic algorithms for classification typically optimize a smooth loss function defined over these residuals. Fisher discriminant analysis (FDA) is such an approach, in which the decision boundary is derived from a linear model of type (2.3), constructed on the direction defined by the respective class means

$$\mathbf{m}_- = \frac{1}{n_-} \sum_{i:y_i=\bar{y}_-} \mathbf{x}_i, \quad \mathbf{m}_+ = \frac{1}{n_+} \sum_{i:y_i=\bar{y}_+} \mathbf{x}_i,$$

with n_- and n_+ the number of data points sampled from the negative and the positive class. This implies that we maximize

$$\mathbf{w} \cdot (\mathbf{m}_+ - \mathbf{m}_-).$$

FDA can be converted into a least-squares estimation problem with the residuals defined by (2.9) and as a result it suffers from the same lack of robustness as its regression variants.

Notwithstanding its name, another well-known statistical method for classification is logistic regression. It is a probabilistic approach, in which not only the predicted category for a given data objects is returned, but also a probability estimate of the correctness of the prediction. To this end, one defines the following probability measures:

$$\begin{aligned} P_{i+} &= \Pr\{y_i = \bar{y}_+ \mid \mathbf{x}_i\}, \\ P_{i-} &= \Pr\{y_i = \bar{y}_- \mid \mathbf{x}_i\}, \end{aligned}$$

such that for all $i \in \{1, \dots, n\} : P_{i+} + P_{i-} = 1$. This can be realized by representing the log-odds or logit function of these probabilities as a linear model of the input variables, i.e.

$$\log\left(\frac{P_{i+}}{1 - P_{i+}}\right) = \mathbf{w} \cdot \mathbf{x}_i + b,$$

resulting again in a discriminant function and corresponding hyperplane. As expected, points at the decision boundary satisfy the double equality

$$P_{i+} = P_{i-} = 1/2.$$

In a more generalized form the model becomes

$$\log\left(\frac{P_{i+}}{1 - P_{i+}}\right) = f(\mathbf{x}_i),$$

in which f can now be a non-linear function of the input variables.

The error function minimized within a logistic regression model is usually called the logistic loss, i.e.

$$\rho(f(\mathbf{x}), y) = \ln(1 + \exp(-y'f(\mathbf{x}))).$$

Unlike the quadratic loss, this loss does not put a penalty on correctly classified objects that heavily deviate from 1. Moreover, noisy patterns that end up completely wrong (within the cluster formed by the other class) are less

Figure 2.4: An overview of common loss functions for binary classification.

heavily penalized, resulting in more robust predictions. Another example of a robust loss function for classification is the hinge loss (Vapnik 1995) defined by

$$\rho(y, f(\mathbf{x})) = [1 - y'f(\mathbf{x})]_+, \quad (2.10)$$

where $[\cdot]_+$ denotes the positive part. As discussed later, this loss is minimized by a support vector machine. The difference between various loss functions is illustrated in Figure 2.4. In essence, the logistic loss and hinge loss can be both efficiently minimized, respectively with a numerical Newton-Raphson procedure and with quadratic programming. Even more than for metric regression, the observation holds that the loss adopted internally to minimize the training error is distinct from the error function employed to evaluate the model on independent test data. Thus, accuracy still remains an interesting measure to evaluate a classification model, although an approximating loss function will be optimized internally.

2.1.6 Multi-class classification

During the last decade, the extension of binary classification to multiple categories has developed into a research topic in its own and still remains an open paradigm in machine learning. As indicated above, we will in general use the notation $\mathcal{Y} = \{\bar{y}_1, \dots, \bar{y}_r\}$ for an r -category multi-class classification problem. One can make a major subdivision of multi-class algorithms into single machine approaches and ensemble methods. The former type of methods

solve in general one global and usually complex optimization problem by defining a loss function over all classes simultaneously. In this group fall the approaches of Weston & Watkins (1998), Crammer & Singer (2001a), Hsu & Lin (2002), Guermeur (2007). The latter type of methods constructs an ensemble of different binary classifiers for a multi-class classification task. This type of methods has been more popular in practice due to their simplicity and applicability to different types of binary classifiers (any binary classification method can be chosen as base classifier). The number of binary classifiers present in the ensemble can vary. In the one-versus-all (Rifkin & Klautau 2004) and one-versus-one (Hastie & Tibshirani 1998, Fürnkranz 2002) ensembles, respectively r and $r(r - 1)/2$ binary classifiers are trained in a general r -class classification task, corresponding to one classifier for each category or to a pair of categories. Other approaches are constructing sets of binary classifiers by means of error-correcting output codes (Dietterich & Bakiri 1995, Allwein *et al.* 2000, Crammer & Singer 2002), directed acyclic graphs (Platt *et al.* 2000) and tree architectures (Cheong *et al.* 2004). Hereunder we give more information about the one-versus-all and one-versus-one ensembles.

For a *one-versus-all* method, we construct new data sets from the original sample as follows:

$$D_k = \{(\mathbf{x}_1, y'_1), \dots, (\mathbf{x}_n, y'_n)\},$$

for $k \in \{1, \dots, r\}$ such that the new labels are defined by:

$$y'_i = \begin{cases} \bar{y}_+, & \text{if } y_i = \bar{y}_k, \\ \bar{y}_-, & \text{else.} \end{cases}$$

Each of these data sets serves as input for a binary classification task such that r discriminant functions f_k are estimated: f_k tries to separate objects of class $\bar{y}_k$ from the other classes. If we would consider decision boundaries like (2.7), then this approach leads to regions of the input space that are ambiguously classified. To resolve this problem, a global prediction rule is derived directly from the r discriminant functions without using the decision boundaries by simply assigning a data object to the category for which the highest value of the discriminant function is found, i.e.

$$h(\mathbf{x}) = \arg \max_{\bar{y}_k \in \mathcal{Y}} f_k(\mathbf{x}), \quad (2.11)$$

Remark that the choice of bias terms for the individual discriminant functions is of crucial importance to obtain accurate predictions. For a *one-versus-one*

method, $r(r-1)/2$ data sets are constructed from D as follows:

$$D_{kl} = \{(\mathbf{x}_i, y_i) \in D \mid y_i \in \{\bar{y}_k, \bar{y}_l\}\}.$$

As a result, a function f_{kl} is estimated for each pair of classes, in which only the objects belonging to these classes are used to train the model. This ensemble also results in an ambiguously defined decision boundary. A prediction for new instances is done here by voting over all $\frac{r(r-1)}{2}$ classifiers based on the binary decision rules.

2.2 Kernel methods

This section is primarily based on (Cristianini & Shawe-Taylor 2000, Schölkopf & Smola 2002, Shawe-Taylor & Cristianini 2004). A much more detailed introduction to kernel methods can be found in these works.

2.2.1 Constructing kernels

In Section 2.1.2, the concept of basis functions was introduced as a way to make a model non-linear without sacrificing the simplicity of linear models. To that end, the input was transformed to an m^* -dimensional space in which a linear model was constructed. In order to explain kernel methods, let us further use the notation $\mathcal{H}$ to denote the resulting feature space, such that the transformation function ϕ can be written as follows:

$$\begin{aligned} \phi &: \mathcal{X} \mapsto \mathcal{H} \\ \mathbf{x} &\rightarrow \phi(\mathbf{x}). \end{aligned}$$

With the requirement that $\mathcal{H}$ is a space for which the dot-product is defined, kernels are used to compute this dot-product efficiently. To illustrate this, let us consider the case of two input variables, so $\mathbf{x} = (x^{(1)}, x^{(2)})$, and let us define the feature map as follows:

$$\phi(x^{(1)}, x^{(2)}) = ((x^{(1)})^2, (x^{(2)})^2, x^{(1)}x^{(2)}, x^{(2)}x^{(1)}).$$

Thus, we have a polynomial representation of the two input variables, in which we treated $x^{(1)}x^{(2)}$ and $x^{(2)}x^{(1)}$ differently. For this feature map, the dot-products in the associated feature space $\mathcal{H}$ can be written as follows:

$$\begin{aligned} \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) &= \phi(x_1^{(1)}, x_1^{(2)}) \cdot \phi(x_2^{(1)}, x_2^{(2)}) \\ &= (x_1^{(1)})^2(x_2^{(1)})^2 + (x_1^{(2)})^2(x_2^{(2)})^2 + 2x_1^{(1)}x_1^{(2)}x_2^{(1)}x_2^{(2)} \\ &= (\mathbf{x}_1 \cdot \mathbf{x}_2)^2. \end{aligned} \tag{2.12}$$

As a consequence, the dot-product can be computed efficiently. Such a compact representation of dot-products in a certain feature space $\mathcal{H}$ will in general be called a kernel with the notation

$$\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) = K(\mathbf{x}_1, \mathbf{x}_2).$$

However, we can also think in the opposite way. Given a function $K : \mathcal{X}^2 \rightarrow \mathbb{R}$, under which conditions does it correspond to a dot-product in a feature space $\mathcal{H}$? The answer to this question can be seen as fundamental to the understanding of kernels.

Definition 2.1. A real $n \times n$ matrix $\mathbf{K}$ is called *positive definite* if

$$\forall \eta \in \mathbb{R}^n : \eta^T \mathbf{K} \eta > 0.$$

Definition 2.2. The Gram matrix of a function $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ on $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$ is the $n \times n$ matrix $\mathbf{K}$ defined by

$$\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j), \quad \forall i, j \in \{1, \dots, n\}.$$

Definition 2.3. A function $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ is called a *positive definite kernel* or simply *kernel* if, for any $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$ and $n \in \mathbb{N}$, the corresponding Gram matrix $\mathbf{K}$ is positive definite.

Proposition 2.4. Let $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ be a positive definite kernel, then there exists a mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ to a dot product space $\mathcal{H}$ such that:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2), \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}.$$

Kernel functions resulting in positive definite Gram matrices always yield a dot-product. As a consequence, data analysis methods based on dot-products can always be rewritten in terms of kernels. Kernel versions have been proposed for classification, regression, clustering, principal component analysis, independent component analysis and many other methods. These algorithms are quite general, because the class of models considered is simply changed by replacing the kernel function. Hereunder, the best known kernel-based learning method, namely the support vector machine (SVM) will be explained.

Kernels can be interpreted as similarity measures, allowing to model similarities for complex data objects. The specific form of the kernel function is domain-dependent and is usually constructed by the data analyst. Since the introduction of kernels, similarity measures have been proposed for a large

number of complex data types, like trees, graphs, strings, text, sets, images, DNA-sequences, etc. As kernels yield an efficient way to compute basis functions, different kernels exist for vectorial input data. The most basic kernel for vectors one can think of is the one for which ϕ defines the identity mapping, i.e.

$$K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2 .$$

This similarity measure is called a linear kernel, since it defines linear models (see further). The kernel defined by (2.12) is a polynomial kernel of degree 2. More generally, if interactions up to d criteria are allowed, one gets a polynomial kernel of degree d , i.e.

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^d .$$

Radial basis function (RBF) kernels are another type of kernels that can be written as

$$K(\mathbf{x}_1, \mathbf{x}_2) = t(d(\mathbf{x}_1, \mathbf{x}_2)) ,$$

with d a metric and t a function on $\mathbb{R}_0^+$. A popular RBF kernel for non-linear modeling is the Gaussian RBF kernel

$$K(\mathbf{x}_1, \mathbf{x}_2) = e^{\gamma \|\mathbf{x}_1 - \mathbf{x}_2\|^2} ,$$

resulting in an infinite-dimensional feature map ϕ . A last example of a kernel for vectorial data is the sigmoid kernel

$$K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma(\mathbf{x}_1 \cdot \mathbf{x}_2) + c) .$$

The class of models generated by the sigmoid kernel is closely related to neural network models. Many other kernels, like spline kernels and ANOVA kernels, exist but are rarely employed in practice. In Figure 2.5, we have visualized different kernels and their corresponding set of basis functions.

2.2.2 Support Vector Machines

Since its introduction in machine learning by Cortes & Vapnik (1995), the support vector machine (SVM) has grown out to an amazingly popular classification algorithm that is nowadays found as a standard data analysis tool in numerous application domains. What caused this popularity is difficult to say, but definitely, the SVM was able to give an answer to shortcomings of state-of-the-art methods of that period, like the lack of scalability (in terms of basis functions) and non-convex optimization function of a neural network. The SVM uses a general discriminant function of type (2.4) with the

Figure 2.5: Three examples of non-linear basis functions and their corresponding kernel functions: polynomials (left), Gaussians (center) and sigmoids (right). The kernels $K(x, x')$ in the bottom figures are plotted as a function of the one-dimensional input x when $x' = 0$. Reprint from (Bishop 2006).

set of basis functions defined by a particular kernel. It is a typical example of Tikhonov regularization, in which the regularization term is the same as in ridge regression, i.e.

$$J(f) = \frac{1}{2} \|\mathbf{w}\|^2, \quad (2.13)$$

and the loss function is the hinge loss as defined by (2.10). The choice of the regularization term is guided by the fact that (2.13) can be written in terms of kernels, leading to a penalty functional on functions operating in the feature space $\mathcal{H}$. The corresponding space of functions over which the objective is minimized is called a reproducing kernel Hilbert space (RKHS).

Beside its interpretation in terms of reproducing kernel Hilbert spaces that holds for other kernel-based methods as well, the penalty functional defined by (2.13) can be also interpreted geometrically in the case of SVMs. By definition, the SVM yields a separating hyperplane in the feature space $\mathcal{H}$, perpendicular to the direction defined by $\mathbf{w}$, and as a matter of fact, the regularizer corresponds to the size of the margin defined by the interval

$$-1 \leq f(\mathbf{x}) \leq +1,$$

Figure 2.6: Illustration of the linear support vector machine. The red lines denote slack variables that are different from zero and the data points with circles denote support vectors. The area in between the two thin vertical lines is the margin and the horizontal line represents f .

In particular, the margin satisfies:

$$\text{margin} = \frac{2}{\|\mathbf{w}\|},$$

and the distance of a point $\mathbf{x}$ to the separating hyperplane is as a consequence given by

$$\frac{y' f(\mathbf{x})}{\|\mathbf{w}\|}.$$

This is illustrated in Figure 2.6. In the same figure also the hinge loss is geometrically visualized. This loss function assigns no errors to correctly classified points lying outside the margin and gives a penalty to points lying in the margin or on the wrong side of the hyperplane. This penalty is linear in the distance to the margin (red lines). By putting everything together, one

arrives at the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & \begin{cases} y'_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0. \end{cases} \end{aligned} \quad (2.14)$$

Thus, on the one hand, the margin is maximized, and on the other hand, the hinge loss represented by the slack variables ξ_i is minimized. The regularization parameter C defines the effective complexity of the model.

Optimization problem (2.14) is a quadratic program, so its exact solution can be found efficiently. Unfortunately, solving this primal formulation requires an explicit representation of ϕ . In order to circumvent this, the optimization problem is usually rewritten in its dual formulation by using Lagrange multipliers. The Lagrange primal objective function is given by:

$$\begin{aligned} L_P = \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i (y'_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) - (1 - \xi_i)) \\ & - \sum_{i=1}^n \beta_i \xi_i, \end{aligned} \quad (2.15)$$

with α_i, β_i Lagrange multipliers. L_P can be minimized w.r.t. the primal variables $\mathbf{w}$, b and ξ_i by setting the respective derivatives to zero. We get

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y'_i \phi(\mathbf{x}_i), \quad (2.16)$$

$$0 = \sum_{i=1}^n \alpha_i y'_i, \quad (2.17)$$

$$\alpha_i = C - \beta_i, \forall i, \quad (2.18)$$

with constraints $\alpha_i, \beta_i, \xi_i \geq 0$. By substituting this into (2.15), one obtains the Wolfe dual objective function:

$$\begin{aligned} L_D &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y'_i y'_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j), \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y'_i y'_j K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned}$$

which must be maximized w.r.t. the constraints

$$\begin{aligned} 0 &\leq \alpha_i \leq C, \quad i \in \{1, \dots, n\}, \\ \sum_{i=1}^n \alpha_i y'_i &= 0. \end{aligned}$$

Many algorithms for solving SVMs and other kernel-based methods implement the dual quadratic program instead of the primal, because in the dual there is no need to compute the dot-product in kernel space explicitly. They can be replaced by the kernel function, resulting in a substantially lower running time for high-dimensional feature spaces. Efficient methods for solving the dual are for example interior point algorithms and sequential minimal optimization, an iterative technique that maximizes Lagrange multipliers two-by-two in an analytic way.

From (2.16) follows that the regularization term can be rewritten as

$$\|\mathbf{w}\|^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y'_i y'_j K(\mathbf{x}_i, \mathbf{x}_j),$$

resulting in a regularizer on functions in a RKHS defined by the kernel function. This function space can be characterized since the discriminant function defined by (2.4) becomes

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y'_i K(\mathbf{x}, \mathbf{x}_i) + b.$$

The model obtained as the final solution of the dual quadratic program gives evidence of a sparse representation, as a large proportion of the n Lagrange multipliers will be fixed to zero. These Lagrange multipliers correspond to correctly classified data objects that lie outside the margin. On the other hand, the Lagrange multipliers are different from zero for data points having an active slack ξ_i in the primal formulation (i.e. for incorrectly classified data points or data points lying in the margin). These data points are marked with a surrounding circle in Figure 2.6. They are called support vectors, since they are the only data points that define the discriminant function in the final solution. Nevertheless, the other data points still have an influence during optimization. The fraction of support vectors will change for different values of C , since this trade-off parameter controls the size of the margin. For large data sets, sparsity leads to a substantial decrease of the computation time in the training and the test phase.

2.3 The foundations of preference modelling

In many situations, humans compare items or objects in order to select an appropriate one for a specific goal. Think for example of buying clothes, listening to music, the dish one orders in a restaurant, etc. Continually, we evaluate objects on criteria such as appropriateness, beauty, correctness, etc. As a consequence of the growing amount of human preference information that comes available due to the fast rise of information retrieval, e-commerce and other internet-related applications, the demand for intelligent systems capable of representing and processing this information also increases. In research areas like decision making, preference modelling, fuzzy modelling, statistics and machine learning, scientists have proposed various ways to characterize such systems.

2.3.1 From decision making to machine learning

For the sake of convenience, the human being that assigns preferences to data objects will hereunder be called the decision maker (DM). He/she can be a qualified person, who has built up the expertise during many years to evaluate a certain task, but he/she can also be a regular person with a limited expertise, like an internet user, a consumer or an interviewee in a statistical enquiry.

In multi-criteria decision making (MCDM) and (fuzzy) preference modelling, the preference assignments of the DM are typically modelled in a logical way, giving the DM insights into the model and allowing for the incorporation of domain knowledge by interaction with the DM and the data analyst. One can distinguish input-oriented preference information (such as the relative importance of criteria or known values for certain parameters of the model) and result-oriented preference information (which typically consists of preference assignments by the decision maker for a small subset of the data). The machine learning community on the other hand has mainly focussed on processing the latter type of information, since it forms the vital data in order to set up a supervised learning task.

Furthermore, learning-based preference models are rather constructed in a statistical than a logical way, in which the uncertainty of preferences is expressed in terms of probabilities and errors instead of membership degrees or degrees of relationship. Settings like ordinal regression, ranking learning and preference learning can also capture human preference behavior, but in general less domain knowledge in terms of input-oriented information is required or even allowed. Roughly speaking, in machine learning an unknown

distribution over the data is assumed and by means of inference a predictive model is fitted to the data, mainly based on result-oriented information (labelled training sample). Machine learning and artificial intelligence techniques can be seen as a tool to extract and represent preference information automatically (Boutilier *et al.* 2004, Doyle 2004, Jung *et al.* 2005).

In recent years, the decision making community has realized the need for automatic systems to support the decision making process. Inference and elicitation procedures are becoming more and more popular in this community as well, but the setup is usually slightly different. During elicitation, the DM interacts with the system and he/she can change his/her opinion about certain data objects to which labels were assigned. Moreover, he/she can partially understand the models and is able to specify values for some of the parameters. On the other hand, most elicitation procedures can only handle consistent data, i.e. noise-free data is required, and the methods scale badly in the number of training instances.

We will mainly focus on the learning aspects of preference modelling in this dissertation. Nevertheless, the resulting algorithms lean on the foundations of preferences in decision making, and therefore, we start with a general introduction to preference modelling in this section. Before going deeper into the algorithms, we discuss ranking, ordinal regression and pairwise preference modelling from a conceptual point of view. This will allow us to characterize the links between these three related settings in later chapters. We also comment upon the main differences between approaches taken in machine learning and (multi-criteria) decision making to infer the parameters of these models. Subsequently, in the next section, we will describe the ordinal regression and pairwise preference learning setting more thoroughly as supervised learning tasks, with emphasis on the underlying ranking problem.

2.3.2 Inference from holistic judgements

In multi-criteria decision making, usually a quite different terminology and notation is used. For example, data objects and features are in this field respectively called alternatives and criteria². In an attempt to stay consistent with the previous sections, we still use the notation of (2.1). Therefore, we

²To be more precise, features are in multi-criteria decision making usually subdivided in attributes and criteria. Criteria are supposed to have a direct indication of preference, while this is not the case for regular attributes. In statistics, such a subdivision is not made, although one can attach a statistical interpretation to the concept of a criterion. Since we will only talk about machine learning in later chapters, we will not go further into detail on this topic.

will assume in this subsection that each data object $\mathbf{x} \in \mathcal{X}$ is represented by its scores on m criteria or input variables, so $\mathcal{X} \equiv \mathbb{R}^m$.

2.3.2.a Ranking versus pairwise preference modelling

In order to model preferences, one can distinguish two main types of models in decision making (Öztürk *et al.* 2005):

1. *Scoring methods*: these methods typically construct a continuous function of the form $f : \mathcal{X} \rightarrow \mathbb{R}$ such that:

$$\mathbf{x}_1 \succeq \mathbf{x}_2 \Leftrightarrow f(\mathbf{x}_1) \geq f(\mathbf{x}_2),$$

which means that $\mathbf{x}_1$ is preferred to $\mathbf{x}_2$ if the highest value was assigned to $\mathbf{x}_1$. In decision making, f is usually referred to as a *utility* function, while it is often called a *ranking* function in machine learning. Remark that the discriminant function introduced for binary classification can also be interpreted as a ranking function. To that end, the term bipartite ranking is sometimes used.

2. *Pairwise preference models*: here the preferences are modeled by one (or more) relations $R : \mathcal{X}^2 \rightarrow [0, 1]$ that express whether $\mathbf{x}_1$ is preferred over $\mathbf{x}_2$. One can distinguish different kinds of relations such as crisp relations, fuzzy relations or reciprocal relations.

The former approach has been especially popular in machine learning for scalability reasons. The latter approach allows a flexible and interpretable description of preferences and has therefore been popular in MCDM and the fuzzy set community. Under certain restrictions, ranking models can be reduced to pairwise preference models. To illustrate this, let us consider a pairwise preference model that satisfies the reciprocity property.

Definition 2.5. A function $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called a *reciprocal preference relation* if it satisfies

$$Q(\mathbf{x}_1, \mathbf{x}_2) + Q(\mathbf{x}_2, \mathbf{x}_1) = 1, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}.$$

The semantics underlying reciprocal preference relations is often probabilistic: $Q(\mathbf{x}_1, \mathbf{x}_2)$ expresses the probability that object $\mathbf{x}_1$ is preferred to $\mathbf{x}_2$. Examples of reciprocal preference models are Bradley-Terry models (Bradley & Terry 1952, Agresti 2002) and Thurstone-Case5 models (Thurstone 1927).

They have been applied in a machine learning context in (Chu & Ghahramani 2005a, Herbrich *et al.* 2007, Radlinski & Joachims 2007). In the following section, we will discuss these models in more detail and in Chapter 6 we will unravel the conditions for which pairwise preference models coincide with ranking models.

Another interesting type of $[0, 1]$ -valued preference relations are outranking relations. They are defined as follows (Perny 1992).

Definition 2.6. A function $S : \mathbb{R}^m \times \mathbb{R}^m \rightarrow [0, 1]$ is called an *outranking relation* if it satisfies the following properties:

1. $\forall \mathbf{x} \in \mathbb{R}^m : S(\mathbf{x}, \mathbf{x}) = 1$.
2. S is increasing in its first argument and decreasing in its second argument.

Outranking relations are mainly used in ELECTRE and PROMETHEE methods (Roy 1991, Vincke 1992, Bouyssou *et al.* 2006). They have a different semantics than reciprocal relations: outranking relations are fuzzy preference relations, in the sense that they can be interpreted as graded versions of the relation $\succeq$. Global outranking relations $S : \mathbb{R}^m \times \mathbb{R}^m \rightarrow [0, 1]$ are typically constructed by taking a weighted sum of single-criterion outranking relations $s_k : \mathbb{R} \times \mathbb{R} \rightarrow [0, 1]$:

$$S(\mathbf{x}_1, \mathbf{x}_2) = \sum_{k=1}^m w_k s_k(x_1^{(k)}, x_2^{(k)}). \quad (2.19)$$

See for example (Gheorghe *et al.* 2004, 2005) for a detailed discussion.

Analogously, one can fuzzify the strict preference relation P , indifference relation I and incomparability relation J , leading to the concept of a (fuzzy) preference structure (De Baets *et al.* 1995). Such a triplet of $[0, 1]$ -valued relations can be generated from an outranking relation and an indifference generator (Van De Walle *et al.* 1998, De Baets & Fodor 2003). When no incomparability is assumed, i.e.

$$J(\mathbf{x}_1, \mathbf{x}_2) = 0, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X},$$

then a close relationship exists between outranking relations and reciprocal relations, although they express different concepts (De Baets & De Meyer 2005). We will not discuss this topic in detail here. We only want to emphasize the connection between the two types of preferences relations, respectively used in MCDM and machine learning.

2.3.2.b Pairwise preference modelling versus ordinal regression

In decision making one often assumes that the parameters of preference models are found by interaction with the DM, which can be a hard task. Many models contain a lot of parameters, especially when the number of criteria is rather large, and in such cases the DM often has difficulties in specifying precise values for many of the parameters. Then, inference procedures are used to infer the unknown parameters of the model from holistic judgements given by the DM. In these approaches, the information obtained from the DM can be divided into input-oriented and result-oriented information. The input-oriented information includes all domain knowledge to construct the model, such as the relative importance of criteria or the values of some of the parameters of the model. On the other hand, the result-oriented information consists of preference judgements expressed by the DM for a small subset of the data objects. The remaining parameters are then inferred with the help of an optimization algorithm (Mousseau 2005, Greco *et al.* 2008).

Expressed in terms of machine learning, $\mathcal{Y}$ will denote the set of judgements the DM can give and D will denote the collection of result-oriented information. One can subdivide the result-oriented information gathered from the DM into two main categories:

1. *Pairwise preference modelling*: the DM gives crisp preference judgements about pairs of data objects. This implies that he has three possible choices, i.e. $\mathcal{Y} = \{+1, 0, -1\}$, in which $+1$ denotes that the first data object is preferred to the second one, -1 denotes the opposite and 0 denotes indifference:

$$\begin{aligned} y_{ij} = +1 & \Leftrightarrow \mathbf{x}_i \succ \mathbf{x}_j, \\ y_{ij} = 0 & \Leftrightarrow \mathbf{x}_i \sim \mathbf{x}_j, \\ y_{ij} = -1 & \Leftrightarrow \mathbf{x}_i \prec \mathbf{x}_j. \end{aligned}$$

We call y_{ij} the label of the couple $(\mathbf{x}_i, \mathbf{x}_j)$. So, the data set $D \subset \mathcal{X}^2 \times \mathcal{Y}$ contains couples of data objects together with their labels: $((\mathbf{x}_i, \mathbf{x}_j), y_{ij})$.

2. *Ordinal regression*: the DM assigns some of the data objects to one of r ordered categories, denoted as $\mathcal{Y} := \{\bar{y}_1, \dots, \bar{y}_r\}$, with $\bar{y}_1$ the “best” objects and $\bar{y}_r$ the “worst” category. So, $D \subset \mathcal{X} \times \mathcal{Y}$ contains couples of the form $(\mathbf{x}_i, y_i)$.

Remark that the pairwise preference modelling setting is slightly more general than the ordinal regression setting, since ordinal class assignments can

be transformed to pairwise preference judgements in the following way:

$$\begin{aligned} y_{ij} = +1 & \Leftrightarrow y_i >_Y y_j, \\ y_{ij} = 0 & \Leftrightarrow y_i =_Y y_j, \\ y_{ij} = -1 & \Leftrightarrow y_i <_Y y_j. \end{aligned}$$

2.3.3 Learning methods versus elicitation procedures

Inference from holistic judgements has for example been investigated by Mousseau & Slowinski (1998), Mousseau *et al.* (2001), Dias & Mousseau (2006) in the context of ELECTRE methods, by Fan *et al.* (2002), Wang & Parkan (2005) for reciprocal preference relations and by Kojadinovic (2004), Marichal *et al.* (2005), Grabisch *et al.* (2008) for Choquet integrals. These authors in general use mathematical programming techniques like linear or quadratic programs to infer the parameters of the respective models. The holistic judgements specified by the DM impose constraints on the parameters of the model, while the objective function to be minimized represents an error function. Briefly summarized, the constraints will appear as follows for outranking relations and pairwise preference judgements:

$$S(\mathbf{x}_i, \mathbf{x}_j) \geq \nu, \quad \forall i, j : y_{ij} = +1 \quad (2.20)$$

$$S(\mathbf{x}_i, \mathbf{x}_j) < \nu, \quad \forall i, j : y_{ij} = -1 \quad (2.21)$$

$$\nu \in [0.5, 1], \quad (2.22)$$

$$\sum_{k=1}^m w_k = 1, \quad (2.23)$$

with w_i weights expressing the importance of criteria, like in (2.19). When the DM decided that a data object $\mathbf{x}_i$ is preferred to another data object $\mathbf{x}_j$, then the value of the outranking relation must be above a certain threshold ν . Conversely, when he/she had the opposite opinion, then the value of the outranking relation must be below ν for that couple of data objects.

In machine learning, similar techniques are used to achieve the same goals, but the general assumptions and the mathematical representation of preferences differ. Before we start with clarifying the main differences, we have to mention that, when we talk about machine learning in this comparison, we actually allude to a small subfield. This subfield covers the ordinal regression, ranking and preference learning settings, which substantially differ from standard classification and (metric) regression learning. Ranking and ordinal regression both refer to learning utility functions, in which the

	Property	MCDM	Machine Learning
Assumptions	input-oriented information	available	usually unavailable
	interaction with DM	yes	no
	DM	can change his mind	fixed preferences
	consistency/noise	usually consistent data assumed	noise allowed
Consequences	typical sample size	small	large
	model type	often additive	very flexible
	regularization	no	yes
	scalability	bad	good
	data type	typically vectorial	any structure

Table 2.1: An overview of the main differences between MCDM and machine learning assumptions and their consequences.

result-oriented information takes the form of ordinal class assignments. In the former case, one is only interested in imposing a complete order on $\mathcal{X}$ given the result-oriented information, similar to ranking in MCDM. In the latter case, all elements of $\mathcal{X}$ have to be classified into one of the r ordinal classes. This corresponds to sorting in MCDM. Ordinal regression models can be constructed from the utility function, either directly with an inference procedure or by applying post-processing techniques such as ROC analysis (see Chapter 2). Ordinal regression differs from multi-class (nominal) classification because there is a linear order relation defined on $\mathcal{Y}$. See for example (Doumpos & Zopounidis 2004) for a study on multi-class classification in MCDM. Thirdly, preference learning usually refers to the situation where the DM has given pairwise preference judgements instead of ordinal class assignments (Förnkrantz & Hüllermeier 2003) and in machine learning it has been considerably less studied than ranking and ordinal regression. Here the parameters of a reciprocal preference relation Q as defined in Section 2.3.2.a are inferred. A close relationship with ranking and ordinal regression models is also retained due to the transitivity of Q .

In Table 2.1 we have tried to summarize the different assumptions between inference procedures in MCDM and learning methods, together with the consequences of these different assumptions. The table might need further explanation:

1. Machine learning is a field where little *knowledge* and *interaction* of the DM is assumed. Let us for example consider the case where the DM is an internet user who is deciding to buy an item from a large collection of items of an e-company. The company could be interested in modelling the behavior of users to improve the sales and production strategy, but to this end the company mainly has result-oriented infor-

mation to its disposal, like preferences given by customers for some of the items or the number of times a particular item was sold. Input-oriented information is much more difficult to obtain, since the visitors of the e-store do not have much time and they have no knowledge of the underlying decision process. In general, when human preferences are modelled in machine learning, little interaction with the DM is required. He/she only provides result-oriented information without having further influence on the model construction process. The absence of input-oriented information and interaction in machine learning settings results in other differences in assumptions.

2. Due to the absence of interaction, the *preferences assigned are fixed* in machine learning algorithms, while in an elicitation procedure the DM can change his mind if inconsistencies are encountered. Instead of asking the DM to revise certain preferences, inconsistencies are seen as *noise*. Although the decision making community is slowly becoming aware that noise in the form of inconsistent judgements occurs frequently in many applications (see e.g. Mousseau *et al.* (2006), Lievens *et al.* (2008, to appear)), the amount of tolerated noise is still much lower than in typical machine learning settings, where it is processed in a statistical way. We want to emphasize that, to our opinion, noise can be subdivided into three categories for modelling preference behavior. There is the noise originating from inconsistent judgements by the DM, but one can consider two other types of noise that also occur in standard classification and metric regression tasks. One of these types concerns noise occurring when incomplete information is given to the data analyst. Often a part of the information needed to model the data perfectly is missing, such as unknown or unmeasurable input variables that might influence the prediction. Such a situation is encountered in many real-world applications of preference modelling. Moreover, noise can also result from corrupted data. With manually collected data, it frequently happens that a small part of the data contains errors, resulting from typing errors or inaccurate devices for example. The MCDM community does not anticipate to the two latter two types of noise.
3. Another consequence of the absence of interaction is the *type of models* that is considered. Researchers in MCDM often focus on additive models for descriptive purposes, although they realize that in many applications complex interactions between different criteria are present.

Machine learning methods on the other hand can capture interactions and other complex patterns in the data.

4. The flexibility of the models in combination with the presence of noise in a finite set of holistic judgements advocate the need for regularization to prevent overfitting. Usually, the set of holistic judgements is rather small compared to the (possibly infinite) set of data objects $\mathcal{X}$. So, in the presence of noise and a large class of models, there is a high chance that the optimal solution w.r.t. the constraints (2.20)-(2.23) leads to a bad generalization performance. Especially for more complex decision making models like Choquet integrals, there is an undeniable need for incorporating regularization mechanisms into the algorithms. Yet, inference procedures in MCDM do not include a regularization mechanism and are therefore vulnerable to all three types of noise mentioned in point 3.
5. Also the typical *sample size* differs. In MCDM the number of data objects is usually very small, while in machine learning sometimes large data sets are analyzed. This is for example the case in internet-related applications, where often huge collections of user data are collected. *Scalability* of the algorithms becomes an issue in these situations. Conversely, inference procedures in MCDM scale badly in the number of data objects.
6. The last difference concerns the *representation* of the data. In MCDM data objects are represented by a number of criteria. However, many machine learning models can also handle data represented in another way. What to think for example on building a preference model for structured data like images, text, graphs, etc. This topic deserves a discussion on its own in MCDM and will not be further discussed here.

Notwithstanding this rather extensive enumeration of differences, we still believe that machine learning and MCDM are closely connected. Both approaches are not really conflicting for most of the assumptions, but the differences rather originate from the kinds of problems the respective methods are applied to. While MCDM has primarily concentrated on preference modelling for traditional decision making problems, applications of preference learning can rather be found in more fancy domains with other characteristics (less domain knowledge, less interaction, more noise, higher-dimensional data, etc.). Nevertheless, the idea of inferring the parameters of a preference model from holistic judgements remains a common goal and a cross-

fertilization between both fields could be meaningful. In (Waegeman *et al.* 2008*d, c*) some initial ideas are presented, but this is beyond the scope of this dissertation.

2.4 Learning to rank

In this section, preference modelling from a machine learning perspective will be discussed more into depth. As indicated in the previous section, one can subdivide the different algorithms into ordinal regression and pairwise preference learning models. Depending on the experimental setup, one will arrive in one of these two settings. Both settings are strongly connected to the underlying concept of a ranking function. We already illustrated this for pairwise preference models. Yet, in this section we also establish this relationship for ordinal regression models. This implies that the ranking function provides the fundamental link between both types of models, a link that will be further explored in later chapters.

During the last decade, a lot of interesting papers on ordinal regression and pairwise preference learning have appeared in the machine learning community (Cohen *et al.* 1998, Frank & Hall 2001, Kramer *et al.* 2000, Herbrich *et al.* 2000, Potharst & Bioch 2000, Crammer & Singer 2001*b*, Har-Peled *et al.* 2002, Shashua & Levin 2003, Harrington 2003, Freund *et al.* 2003, Fürnkranz & Hüllermeier 2003, Cao-Van 2003, Tutz & Hechenbichler 2004, Rennie & Srebro 2005, Chu & Keerthi 2005, Chu & Ghahramani 2005*a, b*, Torra *et al.* 2006, Ataman *et al.* 2006, Xu *et al.* 2006, Yu *et al.* 2006, Lin & Li 2006, Chu & Keerthi 2007, Cao *et al.* 2007, Cléménçon & Vayatis 2007, Cortes *et al.* 2007, Lievens *et al.* 2008) We do not intend to state all these approaches in detail, but rather we will focuss on the common aspects.

2.4.1 The ordinal regression setting

An ordinal regression model $h : \mathcal{X} \rightarrow \mathcal{Y}$ maps a data object to one of the categories of $\mathcal{Y}$. The vast majority of existing ordinal regression models can be represented in the following general form

$$h(\mathbf{x}) = \begin{cases} \bar{y}_1, & \text{if } f(\mathbf{x}) < b_1, \\ \bar{y}_k, & \text{if } b_{k-1} < f(\mathbf{x}) \leq b_k, k = 2, \dots, r-1, \\ \bar{y}_r, & \text{if } f(\mathbf{x}) > b_{r-1}, \end{cases} \quad (2.24)$$

with $b_1, \dots, b_{r-1}$ free parameters and $f : \mathcal{X} \rightarrow \mathbb{R}$ any function that assigns a real value to a data object. It is possible to impose an ordering on a se-

quence of data objects with f and therefore it will be further referred to as a *ranking* function. In this way, consecutive quality levels are here modelled by consecutive intervals on the real line. This is a direct extension of binary classification, where only two intervals were considered. There we had a single threshold b that defined in the linear case a hyperplane perpendicular to the discriminant function. Here we will consider in general $r - 1$ hyperplanes for linear ranking functions in an r -class ordinal regression setting.

Ordinal regression models with an underlying ranking function like (2.24) have less parameters compared to their multi-class classification counterparts: with r categories, r ranking functions are inferred in a one-versus-all ensemble and even $r(r - 1)/2$ functions are inferred in a one-versus-one ensemble. Owing to this simplification, one can often interpret ordinal regression more easily and less data is required to get stable estimates of the parameters and the predictions. In Figure 2.7 we have given a graphical illustration of the reduced complexity of an ordinal regression model compared to a multi-class classification model.

Yet, can the reduction to a single ranking on the other hand result in a two simple model? The answer is no, since in any realistic application of ordinal regression the assumption is made that the ordinal labels assigned by human beings originate from a latent (thus unobserved) continuous variable. In some way, the ranking function of the ordinal regression model represents this latent variable. Moreover, by inferring a single function, ordinal regression models also reduce the possibility of making “big” errors, i.e. misclassifying objects into a much lower or higher class than the real one. In exceptional cases, when such an underlying latent variable is missing, the data might be too complex to be modelled with a single ranking function. Then, a multi-class classification scheme would be preferred. We will not consider that kind of problems in this work. On the other hand, one can also wonder whether a multi-class classification model can be simplified to an ordinal regression model in some occasions. The answer to this question will be given in Chapter 6.

The vast majority of existing methods for ordinal regression comply with the assumption of an underlying latent variable and can therefore be represented as (2.24). This holds for example for traditional statistical methods (Agresti 2002), kernel methods (Shashua & Levin 2003, Chu & Keerthi 2007), bayesian approaches (Chu & Ghahramani 2005a), ordinal decision trees (Frank & Hall 2001, Kramer *et al.* 2000), neural models (Crammer & Singer 2001b), etc. Since this work will not concentrate on clarifying the main differences between these approaches, we will not give a detailed ex-

(a) A linear one-versus-one classifier.

(b) A linear ordinal regression model.

(c) A non-linear ordinal regression model.

Figure 2.7: Graphical illustration of the difference between one-versus-one multi-class classification and ordinal regression models. A three-class hypothetical data set is considered.

planation of all of them. Hereunder, the proportional odds model and the basic kernel-based ordinal regression model are presented.

2.4.1.a The proportional odds model

In statistics, the proportional odds model (McCullagh 1980) is without doubt the best known and most applied technique to represent ordinal responses. This kind of model involves modelling cumulative logits. Given a data set $D \subset \mathcal{X} \times \mathcal{Y}$, the cumulative probability of observing an outcome greater than or equal to $\bar{y}_k$ is defined as follows:

$$P_{ik} = \Pr\{y_i \geq \bar{y}_k \mid \mathbf{x}_i\}, \quad (2.25)$$

for $i = 1, \dots, n$ and $k = 1, \dots, r$. Similar to the logistic regression model for binary responses, the proportional odds model fits a linear model to the log-odds of these cumulative probabilities, i.e.

$$\log\left(\frac{P_{ik}}{1 - P_{ik}}\right) = \mathbf{w} \cdot \mathbf{x}_i + b_k, \quad (2.26)$$

for $i = 1, \dots, n$ and $k = 1, \dots, r$. Note that this construction can be transformed to (2.24). Since a single vector $\mathbf{w}$ of parameters is used, the model has the same effect for each category. As a consequence, all response curves for individual categories have the same shape. They share exactly the same rate of increase or decrease but are horizontally displaced by the thresholds b_k . An example is given in Figure 2.8. One can see that the proportional odds model is a direct generalization of the logistic regression model by considering a threshold for each category. If we would also consider a different slope for each category, we would obtain a one-versus-one ensemble with logistic regression models as binary classifiers.

Remark that the proportional odds model can be easily generalized to non-linear ranking functions by putting

$$\log\left(\frac{P_{ik}}{1 - P_{ik}}\right) = f(\mathbf{x}_i) + b_k, \quad (2.27)$$

but much of its interpretability gets lost.

2.4.1.b Support Vector Ordinal Regression

The support vector ordinal regression algorithm has been introduced by Shashua & Levin (2003) as a direct generalization of SVMs to more than two

Figure 2.8: An example of a proportional odds model with a one-dimensional input and four categories.

ordered categories. Later, this algorithm was enhanced by Chu & Keerthi (2005, 2007) by repairing some shortcomings of the initial algorithm. In essence, these authors presented two slightly different support vector approaches for ordinal regression. We will only expound one of them, namely the version with implicit constraints on the thresholds. Similar to other kernel methods, we consider a model of the form

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}),$$

corresponding to a linear function in kernel space. This model acts as a ranking function that forms the basis of (2.24). In this way, the thresholds can be interpreted as hyperplanes in kernel space that lie perpendicular to the direction $\mathbf{w}$. According to Tikhonov regularization, the vector $\mathbf{w}$ and $r - 1$ thresholds b_k are inferred such that the weighted sum of the error on training data and the regularizer $\frac{1}{2} \|\mathbf{w}\|^2$ are minimized. Errors occur when an object of class $\bar{y}_k$ does not lie in the part of the space defined by the thresholds b_{k-1} and b_k . When more than one hyperplane is located in between the part of the space defined by the real and the predicted class, then all these errors will be taken into account separately. This is known as an all-threshold loss function (Rennie & Srebro 2005). Similar to the SVM, individual errors are denoted with slack variables ξ . Let us introduce $\mathbf{x}_i^k$ to denote the i -th object of category $\bar{y}_k$ and ξ_{ki}^j to denote the slack associated with this object on threshold

Figure 2.9: An example to illustrate the all-threshold loss function of the support vector ordinal regression method. The red lines denote active slack variables ξ . As shown in the figure, the misclassified red data object gives rise to two active slack variables, since two hyperplanes are situated in between its predicted category and its real category. On the other hand, only one slack variable becomes active for the misclassified green and blue data objects.

b_j , then we arrive at the following (primal) optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b_k} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{j=1}^{r-1} \left(\sum_{k=1}^j \sum_{i=1}^{n^k} \xi_{ki}^j + \sum_{k=1}^j \sum_{i=1}^{n^k} \xi_{ki}^{*j} \right) \\ \text{subject to} \quad & \begin{cases} \mathbf{w} \cdot \phi(\mathbf{x}_i^k) - b_j \leq -1 + \xi_{ki}^j, \\ \xi_{ki}^j \geq 0, \\ \text{for } k = 1, \dots, j \text{ and } i = 1, \dots, n^k; \\ \mathbf{w} \cdot \phi(\mathbf{x}_i^k) - b_j \geq +1 - \xi_{ki}^{*j}, \\ \xi_{ki}^{*j} \geq 0, \\ \text{for } k = j+1, \dots, r \text{ and } i = 1, \dots, n^k, \end{cases} \end{aligned} \quad (2.28)$$

where j runs over $1, \dots, r-1$. A visualization of the all-threshold loss function and the resulting active constraints is given in Figure 2.9.

Like for other kernel methods, the transformation to the basis ϕ must not be computed explicitly due to the kernel trick. Optimization problem (2.28) can be rewritten in terms of dot products by introducing Lagrange multipliers. The dual problem is optimized by a variant of the sequential minimal optimization algorithm for support vector machines. More details about the derivation of the dual problem and the implementation can be found in

(Chu & Keerthi 2005). For our experiments we used the publicly available SVORIM-package³.

2.4.1.c Performance measures

Different measures are used in the literature to evaluate the performance of an ordinal regression model. In Chapter 3, we will give a detailed comparison of these measures, but here we already start with an introduction of the most common measures.

In the statistical literature, often the C-index or concordance index is reported as a measure of the predictive power of an ordinal regression model. This measure is an estimator of the concordance probability of an ordinal regression model by counting the number of (lower-class; higher-class) object couples that are correctly ranked by the model (Gönen & Heller 2005). Compared to the mean squared error, it has the important advantage that no metric on $\mathcal{Y}$ is required. Given a model $h : \mathcal{X} \rightarrow \mathcal{Y}$ defined by (2.24), the C-index ignores the thresholds b_k and assesses the predictive power of f . The C-index is mathematically defined as follows:

$$\text{C-index} = \frac{1}{\sum_{k < l} n_k n_l} \sum_{i,j: y_i < y_j}^n I_{\{f(x_i) < f(x_j)\}} + \frac{1}{2} I_{\{f(x_i) = f(x_j)\}}$$

with I the indicator function returning one when its argument is true and zero otherwise. In the binary case, the C-index reduces to the area under the *receiver operating characteristics* (ROC) curve or, equivalently, to the *Mann-Whitney-Wilcoxon* test statistic. The C-index is also widely applied in machine learning for ordinal regression and the very related problem of ranking, but different names such as expected ranking accuracy or pairwise error are given (see for example (Herbrich *et al.* 2000, Joachims 2006, Ataman *et al.* 2006)). A more thorough discussion will be given in Chapter 3.

Beside the C-index, various authors report other performance measures. One of these is standard accuracy or the number of correct predictions as defined by (2.8). Accuracy gives a biased view of the performance of an ordinal regression model, because it does not take into account the magnitude of an error. For example, misclassifying an object of class $\bar{y}_4$ into class $\bar{y}_1$ is a more serious error than misclassifying it into class $\bar{y}_3$. Despite this shortcoming, we will report accuracy since it can be computed for multi-class classification models. The C-index cannot be computed for such models, because it uses

³The SVORIM-package can be downloaded from <http://www.gatsby.ucl.ac.uk/~chuwei/>

the function $f : \mathcal{X} \rightarrow \mathbb{R}$ of the ordinal regression model instead of the decision rule $\mathcal{X} \rightarrow \mathcal{Y}$.

Other frequently used measures which takes the magnitude between the real and the predicted class into account are the mean squared error and the *mean absolute error* (MAE), i.e.

$$\text{MAE}(h, D) = \frac{1}{n} \sum_{i=1}^n |h(x_i) - y_i|. \quad (2.29)$$

As stated above, one assumes there is a metric defined on the output space $\mathcal{Y}$ if these performance measures are used. However, no metric can be defined on an ordinal scale and the use of mean squared error and mean absolute error in ordinal regression settings should be avoided.

2.4.2 Pairwise preference models

As stated in the previous section, machine learning models for pairwise preference learning are typically reciprocal preference relations $Q : \mathcal{X} \rightarrow [0, 1]$. Similar to ordinal regression models, often an underlying ranking function appears as building block for such a reciprocal relation. One can in general construct this relation from a ranking model in the following way:

$$Q(\mathbf{x}_1, \mathbf{x}_2) = g(f(\mathbf{x}_1), f(\mathbf{x}_2)), \quad (2.30)$$

with $g : \mathbb{R}^2 \rightarrow [0, 1]$ usually increasing in its first argument and decreasing in its second argument (Switalski 2003). Under this condition, the pairwise preference model can be interpreted as a generalization of the ordinal regression model.

Another restriction that is usually made in statistical pairwise preference models is that ties cannot occur, which means that the DM can always express a preference on one the objects for a given pair of data objects. With such a restriction, the set of labels becomes $\mathcal{Y} = \{-1, +1\}$ and the problem setting can be transformed to a binary classification task by defining:

$$\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j, \quad \forall i, j.$$

The binary classifier gets in this case a training sample of the form

$$D = \{(\mathbf{x}_{ij}, y_{ij}) \mid i, j \in \mathbb{N}^+, \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}\}.$$

2.4.2.a The Thurstone Case5 and Bradley-Terry model

The Thurstone Case5 and Bradley-Terry model (Thurstone 1927, Bradley & Terry 1952) are two well-known statistical models to represent preference data in the form of pairwise judgements. Both of them can be written as (2.30) such that the function g is proportional to the difference in value of the underlying ranking function for the two objects under consideration, i.e.

$$g(a, b) = \overline{F}(a - b).$$

with $\overline{F} : \mathbb{R}^2 \rightarrow [0, 1]$ a cumulative distribution function. For the Thurstone Case5 model and the Bradley-Terry model, $\overline{F}$ corresponds to respectively the normal and the logistic cumulative distribution function. As a consequence, one can write the Bradley-Terry model as:

$$Q(\mathbf{x}_i, \mathbf{x}_j) = \frac{f(\mathbf{x}_i)}{f(\mathbf{x}_i) + f(\mathbf{x}_j)},$$

with $f : \mathcal{X} \rightarrow \mathbb{R}_0^+$.

2.4.2.b A kernel method for pairwise preference learning

To the best of our knowledge, only a few authors so far considered machine learning models for pairwise preferences (Herbrich *et al.* 2000, Fürnkranz & Hüllermeier 2003, Chu & Ghahramani 2005a, Joachims 2006, Herbrich *et al.* 2007). The main reason for favoring ordinal regression models is that they scale better in the number of cases that have to be evaluated by the DM: for an ordinal regression, the number of evaluations is of the order $\mathcal{O}(n)$, while it is of the order $\mathcal{O}(n^2)$ for pairwise preference models.

We will only summarize the approach of Herbrich *et al.* (2000) to give the reader a little more insight into this type of methods. It is a kernel method that utilizes an underlying ranking function. Again, we consider ranking functions $f : \mathcal{X} \rightarrow \mathbb{R}$ of the following general form:

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}), \tag{2.31}$$

with ϕ a possibly infinite-dimensional and in general unknown feature mapping. Given a data set

$$D = \{((\mathbf{x}_i, \mathbf{x}_j), y_{ij}) \mid i, j \in \mathbb{N}^+, \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}\},$$

consisting of pairwise preference assignments given by a DM, the parameters $\mathbf{w}$ are inferred by solving a quadratic program defined as follows:

$$\begin{aligned} \min_{\mathbf{w}, \xi_{ij}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i,j: y_{ij}=1} \xi_{ij} \\ \text{subject to} \quad & \begin{cases} \forall i, j : y_{ij} = 1 : \mathbf{w} \cdot (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \geq 1 - \xi_{ij}, \\ \xi_{ij} \geq 0. \end{cases} \end{aligned}$$

Again, the function to be optimized puts together two objectives in a weighted sum with C a trade-off parameter. This optimization problem is an immediate extension of the SVM, since we can write

$$f(\mathbf{x}_{ij}) = \mathbf{w} \cdot \mathbf{x}_{ij} = \mathbf{w} \cdot (\mathbf{x}_i - \mathbf{x}_j) = f(\mathbf{x}_i) - f(\mathbf{x}_j).$$

As a consequence, a slack variable ξ_{ij} is introduced for each pair of better-worse objects. This slack variable becomes active when

$$f(\mathbf{x}_i) - f(\mathbf{x}_j) \leq 1,$$

in which the DM decided that $\mathbf{x}_i$ should be preferred over $\mathbf{x}_j$. We are thus looking for a ranking model of type (2.31) that preserves the preferences given by the DM as good as possible, while simultaneously the complexity of the model is bounded.

The derivation of the dual is less technical for pairwise preference models than for ordinal regression models. The primal Lagrangian is given by:

$$\begin{aligned} L_P = \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i,j: y_{ij}=1} \xi_{ij} \\ & - \sum_{i,j: y_{ij}=1} \alpha_{ij} (\mathbf{w} \cdot (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) - 1 + \xi_{ij}) \\ & - \sum_{i,j: y_{ij}=1} \beta_{ij} \xi_{ij}. \end{aligned} \tag{2.32}$$

The Karush-Kuhn-Tucker conditions imply $\frac{\partial L_P}{\partial \mathbf{w}} = 0, \frac{\partial L_P}{\partial \xi_{ij}} = 0$:

$$\mathbf{w} = \sum_{i,j: y_{ij}=1} \alpha_{ij} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)), \tag{2.33}$$

$$\beta_{ij} = C - \alpha_{ij}, \quad \forall i, j : y_{ij} = 1. \tag{2.34}$$

By substituting (2.33) and (2.34) back into (2.32) we get the dual optimization problem:

$$\begin{aligned} \max_{\alpha_{ij}} \quad & \sum_{i,j:y_{ij}=1} \alpha_{ij} - \frac{1}{2} \sum_{i,j:y_{ij}=1} \sum_{k,l:y_{kl}=1} \alpha_{ij} \alpha_{kl} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) (\phi(\mathbf{x}_k) - \phi(\mathbf{x}_l)) \\ \text{subject to} \quad & 0 \leq \alpha_{ij} \leq C, \quad \forall i, j : y_{ij} = 1. \end{aligned}$$

The kernel trick implies that we can write the dual objective in terms of kernels:

$$\begin{aligned} \max_{\alpha_{ij}} \quad & \sum_{i,j:y_{ij}=1} \alpha_{ij} - \frac{1}{2} \sum_{i,j:y_{ij}=1} \sum_{k,l:y_{kl}=1} \alpha_{ij} \alpha_{kl} (K(\mathbf{x}_i, \mathbf{x}_k) - K(\mathbf{x}_i, \mathbf{x}_l) \\ & - K(\mathbf{x}_j, \mathbf{x}_k) + K(\mathbf{x}_j, \mathbf{x}_l)), \end{aligned}$$

and the model becomes

$$f(\mathbf{x}) = \sum_{i,j:y_{ij}=1} \alpha_{ij} (K(\mathbf{x}, \mathbf{x}_i) - K(\mathbf{x}, \mathbf{x}_j)). \quad (2.35)$$

Remark also that the regularizer corresponds to the second objective in the dual optimization problem since

$$\begin{aligned} \|\mathbf{w}\|^2 = \quad & \sum_{i,j:y_{ij}=1} \sum_{k,l:y_{kl}=1} \alpha_{ij} \alpha_{kl} (K(\mathbf{x}_i, \mathbf{x}_k) - K(\mathbf{x}_i, \mathbf{x}_l) \\ & - K(\mathbf{x}_j, \mathbf{x}_k) + K(\mathbf{x}_j, \mathbf{x}_l)). \end{aligned}$$

Given such a ranking model, a reciprocal preference relation can be constructed in several ways. In general it will take the form of (2.30).

2.4.2.c Performance measures

For pairwise preference models, the same performance measures as for binary classification can be used. The binary C-index is commonly reported and in this case given by:

$$\text{C-index} = \frac{1}{\sum_{k < l} n_k n_l} \sum_{i,j:y_{ij}=-1}^n I_{\{f(x_i) < f(x_j)\}} + \frac{1}{2} I_{\{f(x_i) = f(x_j)\}}.$$

Accuracy becomes

$$\text{accuracy} = \frac{2}{|D|} \sum_{i,j:y_{ij}=1} I_{\{h(\mathbf{x}_i)=1\}}.$$

2.5 A case study

In later chapters we will mainly concentrate on the fundamentals of ranking learning and experiments will be primarily set up in order to justify theoretical concepts. Nevertheless, machine learning has always been a field where an exchange of ideas between theorists and practitioners has raised the state-of-the-art to a higher level and therefore, one should not be blind to the applicability of ideas or algorithms. As an introductory example, we will briefly describe here a case study to illustrate the necessity of ranking systems.

In this dissertation, we aim to look further than the traditional information retrieval problems that are often put on the scene to substantiate newly designed ranking algorithms. Therefore, the case study concerns an application of ordinal regression models in the field of industrial quality control. Although the corresponding data was in general too limited to be used further in experiments, the application has been a source of inspiration for some of the concepts that will be presented in later chapters.

2.5.1 The problem setting

In manufacturing and engineering the quality of products or services is in many cases determined by humans, because no automated procedure capable of matching the human expertise exists. Carpets are a kind of products that are nowadays still evaluated with the help of human experts. To receive big orders (e.g. for blocks of flats, office buildings, etc.) carpet companies need a certificate which proves the superb quality of their carpet types. These certificates are handed out by authorized laboratories with the help of human experts.

The laboratories employ a standardized procedure to evaluate carpets, in which the assignment of quality labels is primarily based on the change of appearance during wear. Wear is simulated in most cases by applying an intensive mechanical friction to the carpet. Subsequently, human experts evaluate the carpet's wear behavior by visual comparison with a set of reference samples in standard conditions. At the beginning of quality control in textiles, five main quality levels were considered (ranging from class 1: maximum wear to 5: no wear). The scale was later extended with two additional sub-levels 2.5 and 3.5 in order to further distinguish the most frequently occurring classes 3 and 4. Hence, the experts nowadays determine the quality

of a carpet on a 7-value ordinal scale⁴. The ongoing evaluation procedure by human experts has important disadvantages (Sette *et al.* 1995):

- Visual assessment is not objective and requires a certain expertise. In most of the laboratories there is a lack of experienced people.
- Visual assessment is subject to discussion. Some carpet producers believe that certain laboratories are less conservative than others in their judgements.
- Human experts frequently disagree in their judgement. Labels are therefore often assigned by majority voting of at least 3 experts. Nowadays one allows deviations of 0.5 point for individual experts.
- Consequently, the procedure is also slow as at least 3 people are involved.
- A limited number of reference sample sets has been established for the evaluation of an almost infinite number of carpet qualities.

On the other hand, carpet companies insist on a more reliable and more objective evaluation procedure and, consequently, the automated evaluation of carpet wear has been a topic of study by several institutes during the last three decades. To this end, several instrumental methods have been studied (Siew *et al.* 1988, Xu 1994) including microscopy, photography, densitometry, colorimetry, photogrammetry, glass bead filling and pile mapping. However, most researchers applied image analysis techniques to classify carpet wear. Extensive research has been conducted in this field using many different image analysis algorithms with the intention to quantify tuft definition, tuft geometry, tuft placement, periodicity, texture, pile-lay orientation and roughness (see for example (Wood & Hodgson 1989, Xu 1994, Pourdeyhimi *et al.* 1994, Sette *et al.* 1995, Van Steenlandt *et al.* 1996)). Some of the algorithms which have been used successfully on a limited set of carpet samples are grey value histogram analysis, local intensity-variation filters, statistical measures, edge detection filters, template matching and classifier systems.

Almost all of this research has been conducted using a (very) limited number of carpet samples (most of the aforementioned studies concern less than 50 samples including the wear samples). The acquisition of carpet samples, applying the necessary wear and the corresponding (human) assessment is time consuming and, as a consequence, severely limits the number of samples that

⁴ Although there is no metric defined on the scale, textile engineers consider a main level and a neighboring sub-level “closer” than two neighboring main levels. For example, classifying a carpet of class 2 into class 2.5 is more tolerable than classifying the same object into class 1.

a single research institute can generate. Moreover, interest was mostly limited to one or two carpet types due to the great differences, existing, between carpet types. As a result no “global” automated carpet evaluation system has yet been developed.

Another problem in the development of such a global system is the reproducibility of the images (Wood & Hodgson 1989). Lighting arrangement, camera setup, focussing and level of digitization noise must all be considered with the greatest care to produce accurate (and reproducible) results. Even slight changes (replacement of old lamps, minor change of lighting angle, etc.) can substantially influence the resulting image.

In (Waegeman *et al.* 2008a) we presented a (partial) solution to the aforementioned problems. Our solution consisted of a three step pattern recognition approach, namely data acquisition, feature extraction and model fitting with machine learning methods. In the data acquisition phase, a laser scanner was used instead of a camera to obtain a digital copy of the carpet. This technique allows for a more direct analysis of the carpet texture with less stringent side conditions. In essence, the measured data for each carpet characterizes a three-dimensional model of the carpet surface, represented by a four-dimensional point cloud with variable size. The fourth dimension corresponds to the fraction of the laser light captured by the CCD-camera and gives an indication of the light reflection for individual points in the point cloud. The specific nature of the point cloud data (non-vectorial data) implied that standard algorithms to derive meaningful information from the data could not be used. We combined several methods like point cloud segmentation and density estimation algorithms to extract relevant features from the raw data.

In the last phase the data was modelled with different statistical and machine learning methods to explore the quality of the extracted features. For this purpose, we used recently developed kernel ordinal regression methods, which have several advantages compared to older techniques used in previous research, like neural networks and nearest neighbor classifiers (Sette *et al.* 1995, Van Steenlandt *et al.* 1996). Ordinal regression models take the ordering of the classes into account resulting in more accurate, less variable and better interpretable models, for which smaller data sets are needed. They also build up a deterministic model leading to better reproducible results.

More details about the laser scanning procedure, the characteristics of the obtained data set and the feature extraction phase can be found in (Waegeman *et al.* 2008a). Hereunder, we will only describe the third phase, in which

Feature	Coef.	S.E.	p -value
<i>Motif</i>	3.2990	0.756	0.0000
<i>Prop_q</i>	2.9623	1.704	0.0821
<i>Size</i>	-5.1686	2.444	0.0344
$\hat{b}_z^{LS}$	-52.4013	20.871	0.0120
$\hat{h}_z^{LS}$	35.7042	13.529	0.0083
$\hat{b}_z^{RANSAC}$	13.6942	6.011	0.0227
$\hat{h}_z^{RANSAC}$	51.3263	21.152	0.0152
<i>PDF_q</i>	-41.6305	14.125	0.0032
<i>LoF</i>	-20.3949	3.341	0.0000

Table 2.2: The features retained in the final model after backward feature elimination. P -values result from the Wald-statistic.

machine learning models are employed to build a prediction model for the extracted features. For this analysis 14 features are considered.

2.5.2 Descriptive modelling

We first described the data with a traditional proportional odds model to discover the most important features and to get some initial insights into the fitted model. After fitting such a model to the whole data set, not for all 14 features a significant effect was detected in this initial model. Apart from the rather small sample size, the main reason for finding insignificant parameters was multi-collinearity. This means that due to high correlations between features, high standard errors on the fitted parameters are reported. To improve the quality of the model, a backward feature elimination procedure was applied. This was done manually, because we had prior knowledge about the pairwise correlations between features. After feature elimination, 9 features were retained in the model. Table 2.2 gives an overview of these features together with the p -values computed from the Wald-statistic. In total 8 of the 9 features have a significant effect in the model (significance level 0.05). The last one was kept in the model since it should have an influence on the prediction, although it was not significant at the 0.05-level.

Of the features found during feature extraction, both features derived from the z - and q -coordinate are retained in the final model. Of the features not resulting from the point cloud data, both *LoF* and *Motif* are kept in the model

Figure 2.10: A series of box plots giving an overview of the fitted (continuous) outcomes returned by the proportional odds models after backward feature selection. Each box plot shows the variability of the fitted outcomes for an individual class.

after feature selection. These features represent additional characteristics of the carpet and were gathered in another way (not by laser scanning).

To get a graphical view of the goodness-of-fit, the estimated continuous outcomes for the different classes are shown in Figure 2.10 by means of seven boxplots. A value of 0.85 was measured for the C-index. The plot and the C-index both prove that this rather simple linear model with 9 features is able to describe the data well. In Figure 2.10 the major classes can be accurately separated (except for classes 1 and 2). Nonetheless, a lack of separability is found for the two sublevels 2.5 and 3.5. The former overlaps with class 3 and the latter with both classes 3 and 4. It can be explained by the subjectivity and sensitivity to faulty predictions of the current evaluation procedure by humans. In textile industry, individual human experts are known to mix up carpets of consecutive quality levels. A golden rule says that for individual experts a misclassification of half a point is not a serious error, while a misclassification of a full point or higher is absolutely intolerable. Consequently, one cannot expect to overcome faulty predictions of half a point with an automated system derived from a supervised (statistical or machine learning)

algorithm. Hence, for prediction we will distinguish errors of half a point and more serious errors of at least a full point.

2.5.3 Predictive modelling

Although the proportional odds model fitted the data reasonably well, it has some limitations for prediction. First of all, only linear models are considered, while in reality the best structure might be non-linear. Moreover, in the proportional odds model no regularization is involved for controlling the complexity and for handling correlated features. To give the reader an idea of the gain in predictive power that can be achieved by using machine learning methods, we analyze the data in a second way. The following five models are compared in this analysis:

1. The proportional odds model as described using all 14 features without model selection.
2. A linear kernel-based ordinal regression model as described in Section 2.4.1.b with adequate selection of the regularization parameter C . To this end, the SVORIM package is used.
3. A non-linear kernel-based ordinal regression model using the same software package as in the linear case. We select now the Gaussian RBF kernel as similarity measure, requiring model selection for the width of the kernel as well as the regularization parameter.
4. A linear one-versus-one multi-class classification model, building a linear support vector machine (SVM) for each pair of classes. To this end, the LibSVM-package is used, which implements the multi-class algorithm of Wu *et al.* (2004). Again, model selection is required for the regularization parameter.
5. A non-linear one-versus-one multi-class classification with a non-linear SVM as base learner (LibSVM-package). To compare with the non-linear kernel ordinal regression model, again a Gaussian RBF kernel serves as similarity measure and the optimal values for the width of the kernel and C are found by means of model selection.

2.5.3.a Performance measures

To evaluate the ordinal regression models, we will both consider the C-index and accuracy as performance measures. In an attempt to get an idea of the magnitude of errors and to represent the cost function inherent in this application more truthfully, we will compute them both in two different ways. In

addition to the way described in Section 1.4.1c, we compute them in a second way in which errors of half a point are not penalized:

- For accuracy we do not count misclassifications of 0.5 as an error. For example, classifying a carpet of class 3 into class 3.5 is not counted as an error, but classifying a carpet of class 2 into class 1 is still counted.
- For the C-index this means that only couples of objects differing more than 0.5 (for their real label) are examined. For example, a carpet of class 3.5 is now only coupled to all objects of the classes 5, 2.5, 2 and 1 in the data set to test whether they are correctly ranked.

As briefly discussed in the descriptive modeling part, one might expect very low values for standard accuracy due to (subjective) noise inherent in the evaluation procedure by human experts. The latter definition of accuracy therefore better reflects the actual cost function for misclassifying carpets in textile industry: errors of 0.5 can be tolerated by individual experts, full point errors or higher are not at all tolerated from an expert. In other words, it is difficult for them to distinguish a carpet of class 3 from a carpet of class 3.5 for example. As a result, they often disagree in their judgement.

2.5.3.b Sampling schemes

Similar to model selection, different sampling schemes are commonly utilized to evaluate the error rate (accuracy, AUC, mean squared error, etc.) of machine learning methods. Among the most popular are leaving out a separate test set, k -fold or leave-one-out cross-validation (CV) and bootstrapping. Demsar (2006) analyzed the papers of five recent International Conferences on Machine Learning and he noticed that cross-validation was the most frequently employed. Although train-test set splitting is still the preferred method for large data sets (mainly for computational reasons), many researchers acknowledge the benefits of cross-validation and bootstrapping for small sample sizes (Dietterich 1998). In both sampling schemes many models are fitted to the data by repeatedly sampling training and test data from the original data set.

Thus, cross-validation can be both utilized for model selection and model evaluation. However, one must be cautious with applying this technique for both purposes at the same time. In a theoretical study on microarray data, Varma & Simon (2006) conclude that the CV-error obtained for the classifier with optimal hyperparameters was particularly biased and they recommend nested cross-validation as the best alternative when dealing with small

	0.5 errors not allowed		0.5 errors allowed	
	Accuracy	C-index	Accuracy	C-index
prop. odds	0.528 ± 0.012	0.864 ± 0.003	0.83 ± 0.024	0.925 ± 0.004
linear SVM	0.717 ± 0.025	/	0.974 ± 0.005	/
non-linear SVM	0.730 ± 0.024	/	0.960 ± 0.018	/
linear KOR	0.755 ± 0.013	0.959 ± 0.007	0.974 ± 0.007	0.993 ± 0.008
non-linear KOR	0.758 ± 0.019	0.928 ± 0.015	0.973 ± 0.009	0.978 ± 0.012

Table 2.3: Averages and standard deviations of the performance of the methods under study for ten runs of (nested) cross-validation. Both accuracy and C-index are computed in two ways: with and without considering errors of half a point.

sample sizes. In this case study, we adopt such a nested cross-validation procedure to determine an unbiased estimate of the error rate and to select the optimal hyperparameters. This selection scheme is nowadays more and more used, especially in studies dealing with very small sample sizes like in bioinformatics (Statnikov *et al.* 2005, Mao *et al.* 2005). We will use 10-fold cross-validation both for the inner and outer loop. Briefly summarized, the available data is randomly subdivided into 10 subsets. Repeatedly, one subset is kept out of the model for testing, while the other 9 parts are used for training and model selection. On the remaining 90% of the data, recursively 10-fold cross-validation is carried out for model selection, so that eventually 81% of the original data will serve as input to the classifier, 9% defines the optimal hyperparameters and 10% is only known during testing.

2.5.3.c Results

To reduce the randomness inherent in cross-validation, the evaluation procedure is repeated 10 times. Since multicollinearity does not affect predictive power, all 14 features are taken into account during the predictive modelling part for the proportional odds model (as well as for the machine learning methods). The results for all models and different performance measures are given in Table 2.3.

Let us first consider regular accuracy and concordance with 0.5 errors penalized. The striking difference between the proportional odds model and the other methods for both accuracy and the C-index immediately catches the eye. Apparently, the proportional odds model does not succeed in predicting the data well. To our opinion, this is caused by this model's lack of regularization, rather than by its limitation to fitting linear models, because the other

linear models perform substantially better. Between the other four models less prominent differences in performance are encountered, but still better results are obtained for the kernel-based ordinal regression methods than for the multi-class classification algorithms. At first sight, accuracies of around 0.75 might be not convincing, but one must know that for a balanced 7-class ordinal regression problem a random classifier would only achieve an accuracy of approximately 0.14. So, contrary to the binary case, an accuracy of 0.75 indicates a good separability. For the C-index, the linear kernel ordinal regression model performed noticeably better than its non-linear counterpart. When we take only errors of a full point into account, then — as expected — both the accuracy and the C-index substantially improve, which supports the claim that wrong predictions typically result in small errors⁵. Except for the proportional odds model, less than 5% serious mistakes are made, which is very important for automatic assessment of carpet wear. Surprisingly, the multi-class classification algorithms do not behave much differently than the ordinal regression models. Nevertheless, they manifest a larger variance in error estimation (because they have more parameters to be inferred).

To verify whether the observed differences are statistically significant, statistical tests were conducted. As recommended by Salzberg (1997), Pizarro *et al.* (2002), Demsar (2006), we first employed a general test comparing the (regular) accuracies of all five classifiers simultaneously, before switching to pairwise comparisons. We preferred the non-parametric Friedman-test over repeated measures ANOVA because the assumptions for utilizing the latter might be violated. This test was, as expected, affirmative in detecting a difference among the five models (p -value smaller than 10^{-6}). Subsequently, the various models were pairwise compared. To this end, Dietterich (1998) suggests the McNemar-test for small samples and he in particular warns for the commonly used t -test since the results obtained in repeated runs are often not independent. In particular, an estimator of the variance of cross-validation is always biased (Bengio & Grandvalet 2004). On the other hand, the McNemar-statistic is computed from a two-by-two contingency table, possibly in a single run, ignoring the variability over repeated runs. The cells of the contingency table respectively represent the number of objects for which both models return the same prediction (diagonal cells for correctly and incorrectly classified objects) and the number of objects for which both models disagree (other two cells). We have computed the p -values for the average McNemar-statistic in ten runs of (nested) cross-validation. The results

⁵One could also distinguish errors of a full point and more severe errors. However, the systems under study did not generate errors of more than one point.

	prop. odds	linear SVM	non- linear SVM	linear KOR	non- linear KOR
prop. odds					
linear SVM	<0.001*				
non-linear SVM	<0.001*	0.095			
linear KOR	<0.001*	0.010	0.019		
non-linear KOR	<0.001*	0.018	0.023	0.130	

Table 2.4: The p -values of pairwise McNemar statistics averaged over 10 runs of nested cross-validation. The asterisks indicate statistically significant differences in accuracy at the 0.05 significance level with a Bonferroni correction for multiple comparisons.

are given in Table 2.4. When applying a Bonferroni-correction for multiple testing (which might be too conservative), only the proportional odds model was found significantly worse than the other four methods at the 0.05 significance level. The other p -values are all bigger than $0.05/10 = 0.005$.

2.6 A roadmap through this dissertation

Briefly summarized, the results of the case study demonstrate that laser scanner data contains valuable information regarding the degree of carpet wear. Of the extracted features, correlations up to 0.77 were measured with the labels and a proportional odds model using those features fitted the data well. For predictive modelling, we compared different methods. Except for the proportional odds model, all models had a satisfiable discriminatory power, considering that the underlying decision process with human experts itself is subject to debate. Like human experts, the analyzed methods are less accurate in predicting with a precision of half a point.

We also showed and argued why ordinal regression with a single underlying ranking function have important benefits compared to multi-class classification algorithms for this application, and in general for all types of preference modelling problems where ordinal class assignments are given by humans. They lead to more interpretable models and a better generalization performance. These observations were confirmed in the above case study, although the difference in performance was not statistically significant.

In this dissertation, we will analyze ordinal regression and pairwise preference learning with particular emphasis on the important role of the underlying ranking function. As showed in the previous sections, the ranking is a fundamental element in both the ordinal regression and the pairwise preference learning setting. In the first part, we concentrate on ordinal regression models and present ROC-based performance measures for this setting. Like the case study that we discussed in the previous section, we will propose to assess the ranking instead of the classification model by means of ROC analysis.

In Chapter 3, alternative ROC-based performance measures for ordinal regression are presented. After a general overview of the binary case, we extend ROC analysis principles to ordinal regression and derive an exact expression for the volume under the ROC surface (VUS) spanned by the true positive rates of each class. Moreover, we unravel its interpretation as the probability that a randomly drawn sequence with one object of each class is correctly ranked. Following on that, we compare VUS empirically with other performance measures for ordinal regression and illustrate that the choice of performance measure has a big impact on the learning process.

VUS might seem a very complex performance measure at first sight, being difficult to compute. That's why we investigate in more detail its scalability aspects in Chapter 4. We present graph reformulations of VUS and its (co)variance estimator, and by means of these graphs and dynamic programming techniques, exact algorithms are derived that compute the desired quantities efficiently. Again, simulations are set up to substantiate the theoretical observations.

Chapter 5 then further deals with the learning aspects of ROC-based ordinal regression learning. In particular, we present a way to use VUS as a loss function within a kernel-based learning algorithm. To this end, the graph-theoretic concepts of Chapter 4 and existing algorithms for structured learning are adjusted to optimize VUS efficiently. In addition, we give theoretical bounds on the complexity of our algorithm and illustrate the practical benefits empirically.

Finally, in the more theoretical Chapter 6, the attention is shifted a bit from ranking learning to ranking representability. This means that we aim to define the conditions for which machine learning models can be represented in terms of a ranking function. It turns out that transitivity is the defining property for ranking representability of both pairwise preference models and one-versus-one multi-class classification models. For pairwise preference models, stochastic transitivity is the vital building block. For multi-class classification models, a new type of cycle transitivity is introduced, based on

a generalization of the graph-theoretic concepts introduced for VUS in Chapter 4. It can be proved that this types of cycle transitivity yields a necessary and sufficient condition for which a multi-class classification model can be simplified to an ordinal regression model.

ROC analysis in ordinal regression

Receiver operating characteristics (ROC) analysis provides a way to select possibly optimal models for discriminating two kinds of objects without the need of specifying the cost or class distribution. It is nowadays established as a standard analysis tool in different domains, including medical decision making, pattern recognition and machine learning and the area under the ROC curve, which corresponds to the Wilcoxon-Mann-Whitney test statistic, is increasingly used as a performance measure for binary classification systems. In this chapter, we present a natural generalization of this concept for more than two ordered categories, a setting known as ordinal regression. Our extension of the Wilcoxon-Mann-Whitney statistic now corresponds to the volume under an r -dimensional surface (VUS) for r ordered categories and differs from extensions recently proposed for multi-class classification. VUS rather evaluates the ranking returned by an ordinal regression model instead of measuring the error rate, a way of thinking which has advantages with skew class or cost distributions. We give theoretical and experimental evidence of the advantages and different behavior of VUS compared to error rate, mean absolute error and other (ranking-based) performance measures for ordinal regression. The results demonstrate that the models produced by ordinal regression algorithms minimizing the error rate or a preference learning based error function, not necessarily impose a good ranking on the data.

3.1 ROC graphs

Receiver operating characteristics graphs offer the ability to visualize, organize and select classifiers based on their performance. They originate from signal detection theory and they were first used during World War 2 for the analysis of radar images. The main concept of ROC analysis says that one can consider two kinds of errors when two types of objects have to be distinguished. Since long the method is widely applied in medical decision making to separate diseased subjects from healthy ones. Quite recently, ROC analysis has been introduced in the machine learning domain as an alternative for more traditional ways of evaluating classifiers. Classifier assessment is often difficult in real-world environments because key parameters of the data are unknown. In particular, data analysts often do not know precisely the optimal cost/benefit trade-offs and prior class distributions and these parameters can change over time. Selecting the classifier with the highest accuracy might in such situations not lead to the best performance. ROC analysis overcomes these problems by considering all possible class and cost distributions and the area under the ROC curve is nowadays increasingly used as a performance measure for classification systems.

3.1.1 The binary case

We start with explaining the binary case (Provost & Fawcett 2001, Fawcett 2006). Let us consider pattern recognition methods producing continuous outputs (e.g. probability scores) as discussed in Chapter 2. For such scoring classifiers the parameter b serves as cutting point for classifying an object into the positive or the negative class. Different choices for b typically lead to different models. The predicted labels of such a model, as well as the prediction of classifiers that indicate the (discrete) class labels, can be summarized with a two-by-two contingency table or confusion matrix (shown in Table 3.1). From this table different performance measures can be computed.

Accuracy as introduced in the previous chapter simply measures the number of correctly classified examples, i.e.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN}.$$

The *true positive rate* (TPR) is obtained as the fraction of positive instances that is also positively predicted, i.e.

$$TPR = \frac{TP}{TP + FN}.$$

	$\hat{y} = -1$	$\hat{y} = 1$	
$y = -1$	<i>TN</i>	<i>FP</i>	n_-
$y = 1$	<i>FN</i>	<i>TP</i>	n_+
	<i>NP</i>	<i>PP</i>	n

Table 3.1: Confusion matrix for a two class classification problem of size n . The rows indicate the true labels and the columns the predicted labels. The 4 table cells represent the 4 situations that can occur in binary classification. Each cell counts the number of times the situation in questions occurs. For example, *FP* counts the number of negative objects that are predicted positive.

This quantity is also called *sensitivity* or sometimes *recall*. Analogously, the *false positive rate* (*FPR*) stands for the fraction of real negative examples that are positively predicted and *specificity* is defined as one minus *FPR*, i.e.

$$\begin{aligned} \text{specificity} &= TNR = \frac{TN}{TN + FP}, \\ &= 1 - FPR = 1 - \frac{FP}{TN + FP}. \end{aligned}$$

Another measure is *precision*, defined as

$$\text{precision} = \frac{TP}{TP + FP}.$$

Precision and recall define the *F-measure*, which is frequently utilized in information retrieval applications, i.e.

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

From all measures that can be computed from the contingency table, the true positive rate and false positive rate are the ones used to construct an ROC graph. An ROC curve is created by plotting the true positive rate on the y-axis versus the false positive rate on the x-axis for different classifiers. If we consider all possible settings of the threshold b , then $n + 1$ contingency tables can be composed for a scoring classifier assigning scores to a data set of n instances. Each of these $n + 1$ contingency tables uniquely identifies a sensitivity-specificity trade-off or, equivalently, a point in ROC space. Combining these $n + 1$ ROC points gives rise to an increasing curve starting in the point (0,0) and ending in (1,1). These points respectively correspond to classifying all objects in the negative or the positive category and the line

connecting them represents the behavior of a random classifier. Yet, the ROC point of any classifier claiming to perform better than random guessing must be situated above this line. We clarify these concepts with an example.

Example 3.1. In Figure 3.1(a) a hypothetical data set of 5 negative and 5 positive data objects is visualized together with the continuous output produced by a fictive binary ranking function $\hat{f}$. If $b = 0$ is chosen as threshold, then the corresponding binary classifier $\hat{h}$ predicts 80% of the data correctly. When $b = 4.5$, then the accuracy does not change and the true positive rate increases. This might be more desirable depending on the cost distribution. For this data set 11 different contingency tables can be composed by placing the threshold in between the values of two consecutive data objects. Figure 3.1(b) shows the corresponding curve for these 11 ROC-points. All points lie reasonably beyond the line corresponding to a random classifier.

3.1.2 The area under the ROC curve

Given a certain cost distribution, one can find the optimal classifier on the convex hull by drawing iso-performance lines. Unfortunately, the cost distribution is often not precisely known. Moreover, a ROC curve is a two-dimensional representation of the behavior of set of classifiers or of a single classifiers producing continuous outputs. To compare classifiers producing continuous outputs one might prefer a single scalar value instead of a graph. To this end, the *area under the ROC curve* (AUC) has been proposed as evaluation criterion by Hanley & McNeil (1982). Interestingly, one has shown Cortes & Mohri (2003), Yan *et al.* (2003) that the *AUC* is equivalent to the *Wilcoxon-Mann-Whitney* statistic. We use these results in order to give a formal definition of the area under the ROC curve.

Definition 3.2. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a ranking function and let $D \subset \mathcal{X} \times \mathcal{Y}$ be a data set of size n i.i.d. according to $\mathcal{D}$, then we define the area under the ROC curve, denoted by $\hat{A}(f, D)$, as

$$\hat{A}(f, D) = \frac{1}{n_- n_+} \sum_{y_i < y_j} I_{\{f(\mathbf{x}_i) < f(\mathbf{x}_j)\}}, \quad (3.1)$$

with I the indicator function. In the presence of ties, $A(f, D)$ becomes

$$\hat{A}(f, D) = \frac{1}{n_- n_+} \sum_{y_i < y_j} I_{\{f(\mathbf{x}_i) < f(\mathbf{x}_j)\}} + \frac{1}{2} I_{\{f(\mathbf{x}_i) = f(\mathbf{x}_j)\}}.$$

	Data set									
i	1	2	3	4	5	6	7	8	9	10
y_i	$\bar{y}_-$	$\bar{y}_-$	$\bar{y}_+$	$\bar{y}_-$	$\bar{y}_-$	$\bar{y}_+$	$\bar{y}_+$	$\bar{y}_-$	$\bar{y}_+$	$\bar{y}_+$
$\hat{f}(\mathbf{x}_i)$	-7	-5	-4	-3	-1	0.2	1.7	5	11	12

(a) Possible output of a classifier on a hypothetical data set of 10 instances.

(b) The corresponding ROC-curve and convex hull.

Figure 3.1: The ROC-curve visualized for a hypothetical data set and scoring classifier.

The area under the ROC curve offers advantages over accuracy as evaluation criterion when the class distributions are unbalanced or when different misclassification costs can be assigned to the different classes. Cortes & Mohri (2003) studied in detail the relationship between accuracy and the *AUC* and concluded that both measures will reveal different characteristics of a classifier. In particular, they derived an exact expression for the expected value and the variance of the *AUC* for a fixed error rate and showed that classifiers with the same (low) error rate can exhibit noticeable different *AUC* values.

Another difference is that the accuracy directly evaluates the performance of the decision function $h(\mathbf{x})$, while the *AUC* quantifies the ranking performance $f(\mathbf{x})$ of the classifier. Agarwal *et al.* (2005) define in this context the term *expected ranking accuracy*.

Definition 3.3. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a ranking function and let $\mathcal{D}_-$ and $\mathcal{D}_+$ respectively represent the conditional distribution over $\mathcal{X}$ for objects of the negative and the positive class, then the expected ranking accuracy, denoted by $A(f)$, is defined as

$$A(f) = \Pr_{X_- \sim \mathcal{D}_-, X_+ \sim \mathcal{D}_+} \{f(X_-) < f(X_+)\}.$$

In the presence of ties, $A(f)$ becomes

$$A(f) = \Pr_{X_- \sim \mathcal{D}_-, X_+ \sim \mathcal{D}_+} \{f(X_-) < f(X_+)\} \\ + \frac{1}{2} \Pr_{X_- \sim \mathcal{D}_-, X_+ \sim \mathcal{D}_+} \{f(X_-) = f(X_+)\}.$$

The expected ranking accuracy thus stands for the probability that the function value of an object randomly drawn from the negative class is strictly smaller than the function value of an object randomly drawn from the positive class. The choice of notation is guided by the property that $\hat{A}(f, D)$ is an unbiased nonparametric estimator of $A(f)$ on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, which can easily be seen from the formulas. Given a ranking function f and a data set D , it is $\hat{A}(f, D)$ that represents the area under the ROC curve. The expected ranking accuracy does not have a direct geometrical interpretation, but from a theoretical perspective it could be defined as the obtained area under the curve with a data set of infinite size sampled from $\mathcal{D}$.

3.2 Multi-class ROC analysis

Recently, different approaches have been proposed to extend ROC analysis for multi-class classification Hand & Till (2001), Ferri *et al.* (2003), Flach

(2004), Fieldsend & Everson (2005). In the most general case, the *volume under the ROC surface (VUS)* has to be minimized in multi-class classification. The ROC surface can be seen as a Pareto front, where each objective corresponds to one dimension. In case there are more than two classes (say r), then the number of objectives depends on the multi-class method that is used.

3.2.1 One-versus-all ensembles

In this case, r functions f_k are estimated that try to separate objects of class k from the other classes. Because of that, misclassification costs for each class are fixed and the corresponding ROC surface will have r dimensions representing the true positive rates TPR_k for each class (Flach 2004). The true positive rate of class $\bar{y}_k$ is defined as the fraction of the instances classified as class $\bar{y}_k$ that really belong to class $\bar{y}_k$, divided by the number of data objects of class $\bar{y}_k$:

$$TPR_k = \frac{1}{n_k} \sum_{y_i = \bar{y}_k} I_{\{\hat{y}_i = \bar{y}_k\}},$$

with n_k the number of training objects of class $\bar{y}_k$ as in Chapter 2. ROC points are obtained by varying the thresholds b_k in the prediction rule for one-versus-all ensembles as defined by (2.11).

3.2.2 Pairwise ensembles

In this case, a function f_{kl} is estimated for each pair of classes, which allows to specify the cost for a misclassification of an object of class $\bar{y}_k$ predicted as class $\bar{y}_l$. In the most general case, the corresponding ROC space is in this case spanned by $\frac{r(r-1)}{2}$ objectives, as considered by Ferri *et al.* (2003). To reduce the complexity of multi-class ROC analysis, usually a simplification is made to constructing two-dimensional ROC curves for each pair of categories as proposed by Hand & Till (2001). Using these pairwise ROC curves, a prediction for new instances is made by voting over all $\frac{r(r-1)}{2}$ classifiers based on the predictions of these binary classifiers. We can naturally extend the concepts of expected ranking accuracy and AUC for pairwise ROC curves.

Definition 3.4. Let $\bar{\mathcal{F}}$ be a set of $r(r-1)/2$ ranking functions $f_{kl} : \mathcal{X} \rightarrow \mathbb{R}$, let $\mathcal{Y} = \{\bar{y}_1, \dots, \bar{y}_r\}$ be an unordered set of categories, let $\mathcal{D}_k$ represent the conditional distribution over $\mathcal{X}$ for objects of category $\bar{y}_k$ with $k = 1, \dots, r$,

then the pairwise expected ranking accuracy for two categories $\bar{y}_k$ and $\bar{y}_l$ on f_{kl} , denoted by $A_{kl}(\bar{\mathcal{F}})$, is defined as

$$A_{kl}(\bar{\mathcal{F}}) = \mathbf{Pr}_{X_k \sim \mathcal{D}_k, X_l \sim \mathcal{D}_l} \{f_{kl}(X_k) < f_{kl}(X_l)\}. \quad (3.2)$$

Let $D \subset \mathcal{X} \times \mathcal{Y}$ be a data set of size n i.i.d. according to $\mathcal{D}$, then we define the area under the ROC curve $\hat{A}_{kl}(f_{kl}, D)$ between categories $\bar{y}_k$ and $\bar{y}_l$ on the ranking function f_{kl} as

$$\hat{A}_{kl}(\bar{\mathcal{F}}, D) = \frac{1}{n_k n_l} \sum_{y_i = \bar{y}_k} \sum_{y_j = \bar{y}_l} I_{f_{kl}(\mathbf{x}_i) < f_{kl}(\mathbf{x}_j)}. \quad (3.3)$$

Based on this extension of ROC analysis, Hand & Till (2001) proposed the unweighted sum of all pairwise AUCs as a performance measure for multi-class classification tasks. We will refer to this performance measure as the *multi-class AUC*.

Definition 3.5. Let $\hat{A}_{kl}$ be the pairwise AUC's resulting from a set $\bar{\mathcal{F}}$ of $r(r-1)/2$ ranking functions f_{kl} on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, then we define the multi-class AUC, denoted by $\hat{U}_{ovo}$ as

$$\hat{U}_{ovo}(\bar{\mathcal{F}}, D) = \frac{2}{r(r-1)} \sum_{k < l} \hat{A}_{kl}(\bar{\mathcal{F}}, D).$$

3.3 ROC analysis in ordinal regression

3.3.1 A natural generalization of the binary case

In ordinal regression the picture is slightly different. As pointed out in Chapter 2, the vast majority of existing ordinal regression models fit in general one function f to the data together with $r-1$ thresholds b_k for r ordered classes (see expression 2.24). As for the binary case, the thresholds will be discarded and we will further only assess the discriminative power of the ranking function. To this end, the concept of expected ranking accuracy can be intuitively extended for more than two ordered categories.

Definition 3.6. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be the ranking function of an ordinal regression model of the form (2.24), then the expected ranking accuracy, denoted by $U(f)$, is given by

$$U(f) = \mathbf{Pr}_{X_k \sim \mathcal{D}_k} \{f(X_1) < \dots < f(X_r)\}.$$

Now the expected ranking accuracy measures the probability that a random sequence of one data object of each category is correctly ranked by the ranking function f . It is estimated from a finite data set D by counting the number of sequences of r objects, one of each class, that are correctly ranked by the ranking function, i.e.

$$\hat{U}(f, D) = \frac{1}{\prod_{k=1}^r n_k} \sum_{y_{j_1} < \dots < y_{j_r}} I_{\{f(\mathbf{x}_{j_1}) < \dots < f(\mathbf{x}_{j_r})\}}. \quad (3.4)$$

We will further assume that the ranking function f is sufficiently continuous to ensure that a unique ranking is assigned to different data objects. Since this is the case for almost all ordinal regression systems, it will not restrict the generality of our analysis. Although the formulas and algorithms that will be proposed further can be easily modified to account for ties in the output, it would make the mathematical notation needlessly complex. For example, the above-mentioned estimator becomes in the presence of ties in the three class case:

$$\begin{aligned} \hat{U}(f, D) = & \frac{1}{\prod_{k=1}^r n_k} \sum_{y_{j_1} < y_{j_2} < y_{j_3}} \left(I_{f(\mathbf{x}_{j_1}) < f(\mathbf{x}_{j_2}) < f(\mathbf{x}_{j_3})} \right. \\ & + \frac{1}{2} I_{f(\mathbf{x}_{j_1}) = f(\mathbf{x}_{j_2}) < f(\mathbf{x}_{j_3})} \\ & + \frac{1}{2} I_{f(\mathbf{x}_{j_1}) < f(\mathbf{x}_{j_2}) = f(\mathbf{x}_{j_3})} \\ & \left. + \frac{1}{6} I_{f(\mathbf{x}_{j_1}) = f(\mathbf{x}_{j_2}) = f(\mathbf{x}_{j_3})} \right). \end{aligned} \quad (3.5)$$

It is easy to see that $\hat{U}(f, D)$ is an unbiased estimator of $U(f)$. Furthermore, $\hat{U}(f, D)$ has a geometrical interpretation, which can be summarized in the following theorem.

Theorem 3.7. *Given a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ that imposes a ranking over a data set $D \subset \mathcal{X} \times \mathcal{Y}$, then $\hat{U}(f, D)$ corresponds to the volume under the r -dimensional ROC surface (VUS) spanned by the true positive rates of each class.*

Proof. For a given ranking function f , the true positive rate of each class only depends on the threshold vector $\mathbf{b} = (b_0, \dots, b_r)$ as defined in (2.24) with $b_0 = -\infty$ and $b_r = +\infty$, i.e.

$$TPR_k(\mathbf{b}) = \frac{1}{n_k} \sum_{y_i = \bar{y}_k} I_{b_{k-1} < f(\mathbf{x}_i) \leq b_k}.$$

The ROC surface represents all optimal models for different cost distributions and each model corresponds to a unique vector of true positive rates. We can collect all possible values for the true positive rates by linking a particular element of the data set with each threshold, i.e.

$$b_k = f(\mathbf{x}_{j_k}),$$

with $k = 1, \dots, r - 1$ and $j_k = 1, \dots, n$. Let us therefore consider the set $\mathcal{B}$ containing all such vectors $\mathbf{b}$ that uniquely define a point on the convex hull of the ROC surface, then without loss of generality the volume under the ROC surface can be written as:

$$\text{VUS} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{b} \in \mathcal{B}} \text{TPR}_r(\mathbf{b}).$$

We will now write out this set $\mathcal{B}$ by looking at all possible vectors $\mathbf{b}$ that lead to different points on the convex hull. By definition, the cardinality of $\mathcal{B}$ is upper bounded by n^{r-1} because only $b_1, \dots, b_{r-1}$ can vary.

Nevertheless, the cardinality will be smaller because the ordering of the thresholds enforces an additional constraint on $\mathbf{b}$:

$$f(\mathbf{x}_{j_1}) \leq \dots \leq f(\mathbf{x}_{j_r}). \quad (3.6)$$

Now suppose that the data set was sorted according to f . We will count all elements of $\mathcal{B}$ starting with all thresholds at the first element of the data set: $b_1 = \dots = b_{r-1} = f(\mathbf{x}_1)$, so all instances are then classified into the last category and hence $\text{TPR}_r = 1$. This value will gradually decrease when the thresholds are shifted to the end of the ranking. We will first count all contributions while the last threshold is moved up. When the position j_{r-1} increases one step to the next element in the ordered data set, the true positive rates TPR_{r-1} and TPR_r can change. There are three possibilities:

- $y_{j_{r-1}} = \bar{y}_{r-1}$: TPR_{r-1} will increase, leading to a new element of $\mathcal{B}$ (we shift in a horizontal direction to a new point on the ROC-surface).
- $y_{j_{r-1}} = \bar{y}_r$: TPR_r will decrease, but this setting for b_{r-1} does not correspond to a new point on the surface (we shift in a vertical direction to a new point lying beneath the convex hull of all ROC points).
- $y_{j_{r-1}} \notin \{\bar{y}_{r-1}, \bar{y}_r\}$: there is no contribution to the sum because TPR_r and TPR_{r-1} remain unchanged (we stay at the same point on the ROC surface).

As a consequence, the sum over all values for b_{r-1} reduces to a sum over all elements of class $\bar{y}_{r-1}$ and recursively the same reasoning holds for each b_k . Thus, $\mathcal{B}$ contains $\prod_{k=1}^{r-1} n_k$ elements and

$$\text{VUS} = \frac{1}{\prod_{k=1}^{r-1} n_k} \sum_{\substack{b_k=f(\mathbf{x}_{j_k}) \\ y_{j_1} < \dots < y_{j_{r-1}}}} TPR_r(\mathbf{b}). \quad (3.7)$$

For all positions $(j_1, \dots, j_{r-1})$ of the thresholds the true positive rate of class $\bar{y}_r$ can be seen as:

$$TPR_r(\mathbf{b}) = \frac{1}{n_r} \sum_{y_{j_r}=\bar{y}_r} I_{f(\mathbf{x}_{j_{r-1}}) < f(\mathbf{x}_{j_r})}. \quad (3.8)$$

This value is counted over all objects of the other $r - 1$ classes. Combining (3.7) and (3.8) and moving constraint (3.6) from the sum to the indicator function leads to expression (3.4). $\square$

In statistics there has been some related work on this topic. Dreiseitl *et al.* (2000) derive formulas for the variance of $\hat{U}(f, D)$ and the covariance between two volumes in the three-class case. This work has been extended to the general r -class case by (Nakas & Yiannoutsos 2004). They conclude that bootstrapping is preferred over U-statistics¹ to compare more than two diagnostic tests because the computation of the exact variance and covariance estimators becomes intractable for large values of n and r . We focus more on the use of $\hat{U}(f, D)$ as performance measure for ordinal regression problems in this thesis.

Example 3.8. For three ordered classes, the ROC surface can be visualized. We have constructed this ROC surface for a synthetic data set. We sampled 3×100 instances from 3 bivariate Gaussian clusters representing consecutive classes. The mean of the clusters was set to (10,10), (20,10) and (20,20) respectively, σ_1 and σ_2 were set to 5 for the first two clusters and were set to 7 for the last cluster. ρ was fixed to 0. This data set is visualized in Figure 3.2(a). We used the support vector ordinal regression algorithm of Chu & Keerthi (2005) to estimate the ranking function f , without looking at the thresholds. The obtained ROC surface is shown in Figure 3.2(b).

¹A U-statistic is a class of nonparametric statistics. See for example (Lehmann 1975) for more information.

(a) A synthetic data set with three bivariate Gaussian clusters representing three ordered classes with respective means (10,10), (20,10) and (20,20). The standard deviation was set to (5,5) for the first two clusters and to (7,7) for the last cluster, while ρ was fixed to 0.

(b) The ROC surface obtained for the synthetic data given in (a) and the ranking returned by a support vector ordinal regression algorithm.

Figure 3.2: *The ROC surface visualized for a three class ordinal regression data set.*

Let us now associate a cost function $c : \mathcal{Y} \rightarrow \mathbb{R}$ with each category of $\mathcal{Y}$. In other words, $c(\bar{y}_k)$ defines the penalty of misclassifying an object of class $\bar{y}_k$. As in the binary case, the convex hull of the r -dimensional ROC surface represents the set of optimal classifiers for any particular choice of cost. With known costs and a predefined instance-wise decomposable loss function as defined by (2.5), we can search for a classifier which minimizes

$$L(f, D) = \sum_{i=1}^n c(y_i) \rho(h(\mathbf{x}_i), y_i),$$

on a training data set when the costs are provided before training. However, the costs will in many cases still vary after training and then the optimal classifier can be selected from the ROC surface.

3.3.2 Relationship with other performance measures

The volume under the ROC surface gives us the opportunity to compare the quality of two different ROC surfaces for varying costs. Because $\hat{U}(f, D)$ is a nonparametric estimator of the expected ranking accuracy, it quantifies the ranking imposed by an ordinal regression model. We will now discuss how $\hat{U}(f, D)$ relates to previous work on ordinal regression and multi-class classification.

3.3.2.a The pairwise error

Herbrich *et al.* (2000) propose a ranking-based framework for ordinal regression based on structural risk minimization. Guided by preference learning, their algorithm optimizes the number of correctly ranked object pairs, i.e.

$$\hat{U}_{pairs}(f, D) = \frac{1}{\sum_{\bar{y}_k < \bar{y}_l} n_k n_l} \sum_{y_i < y_j} I_{f(\mathbf{x}_i) < f(\mathbf{x}_j)}, \quad (3.9)$$

together with a regularization term in the form of a convex quadratic program. This loss function corresponds to the C-index defined in Chapter 2

3.3.2.b The multi-class AUC revisited

As pointed out at the beginning of this section, ROC analysis for ordinal outcomes can be seen as a simplified version of multi-class ROC analysis

since we are dealing with a single ranking function. Nevertheless, the approximation presented by Hand & Till (2001) for one-versus-one multi-class classification could easily be modified for our purpose, i.e.

$$\begin{aligned}\hat{U}_{ovo}(f, D) &= \frac{2}{r(r-1)} \sum_{k < l} \hat{A}_{kl}(f, D), \\ \hat{A}_{kl}(f, D) &= \frac{1}{n_k n_l} \sum_{y_i = \bar{y}_k} \sum_{y_j = \bar{y}_l} I_{f(\mathbf{x}_i) < f(\mathbf{x}_j)}.\end{aligned}\quad (3.10)$$

They construct a two-dimensional ROC curve for each pair of classes $\{\bar{y}_k, \bar{y}_l\}$, in which only the objects of these classes are taken into account. Instead of multi-classification, we use the same ranking function for each two-dimensional ROC curve and take the sum of the areas under these curves as a measure for the quality of the ranking. In nonparametric statistics $\hat{U}_{ovo}(f, D)$ is known as the *Jonckheere-Terpstra* test statistic, a more powerful alternative to the *Kruskal-Wallis* test statistic for simultaneously testing whether more than two ordered populations significantly differ (Higgins 2004).

3.3.2.c Consecutive categories

Another approximation $\hat{U}_{cons}(f, D)$ is directly deduced from (2.24). With a function f and $r - 1$ thresholds one could envisage threshold b_k as providing the separation between the consecutive ranks $\bar{y}_k$ and $\bar{y}_{k+1}$. Varying this threshold will change the proportion between objects predicted lower than or equal to class k and objects predicted higher than class k . This corresponds to measuring the non-weighted sum of $r - 1$ two-dimensional ROC curves representing the trade-off between consecutive classes:

$$\begin{aligned}\hat{U}_{cons}(f, D) &= \frac{1}{r-1} \sum_{k=1}^{r-1} \hat{A}_k(f, D), \\ \hat{A}_k(f, D) &= \frac{1}{\sum_{i=1}^k n_i \sum_{j=k+1}^r n_j} \sum_{y_i \leq \bar{y}_k} \sum_{y_j > \bar{y}_k} I_{f(\mathbf{x}_i) < f(\mathbf{x}_j)}.\end{aligned}$$

3.3.2.d Relationships

$\hat{U}_{pairs}(f, D)$, $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{cons}(f, D)$ all compare pairs of objects instead of sequences of r objects as in VUS. These measures can be seen as

unbiased estimators of probabilities, other than the expected ranking accuracy. This is summarized in the following theorem.

Proposition 3.9. *Let $\mathcal{D}_<$ be the joint distribution function of couples of data objects for which the second element of the couple is higher ranked than the first element, let $\mathcal{D}_k$ be as defined in Def. 3.4, let $\mathcal{D}_{k,l}$ be the conditional distribution of an object of $\mathcal{X}$ given that its label lies between categories $\bar{y}_k$ and $\bar{y}_l$ (inclusive), let $f : \mathcal{X} \rightarrow \mathbb{R}$ and let*

$$\begin{aligned} U_{pairs}(f) &= \Pr_{(X_1, X_2) \sim \mathcal{D}_<} \{f(X_1) < f(X_2)\}, \\ U_{ovo}(f) &= \frac{2}{r(r-1)} \sum_{k < l} \Pr_{X_1 \sim \mathcal{D}_k, X_2 \sim \mathcal{D}_l} \{f(X_1) < f(X_2)\}, \\ U_{cons}(f) &= \frac{1}{r-1} \sum_{k=1}^{r-1} \Pr_{X_1 \sim \mathcal{D}_{1,k}, X_2 \sim \mathcal{D}_{k+1,r}} \{f(X_1) < f(X_2)\}, \end{aligned}$$

then

$$\begin{aligned} \mathbf{E}[\widehat{U}_{pairs}(f, D)] &= U_{pairs}(f), \\ \mathbf{E}[\widehat{U}_{ovo}(f, D)] &= U_{ovo}(f), \\ \mathbf{E}[\widehat{U}_{cons}(f, D)] &= U_{cons}(f). \end{aligned}$$

The proof directly follows from the definitions. $\widehat{U}_{pairs}(f, D)$, $\widehat{U}_{ovo}(f, D)$ and $\widehat{U}_{cons}(f, D)$ assess the ranking of an ordinal regression model in another way than VUS. They can be considered as approximations of VUS because they only count object pairs instead of sequences, which is computationally more efficient. However, they also turn out to behave differently than VUS.

Proposition 3.10. *Let f_R be a random ranking function of an ordinal regression model with r classes (thus f_R randomly assigns continuous outputs to data objects), then*

$$(i) \quad U(f_R) = \frac{1}{r!}.$$

$$(ii) \quad U_{pairs}(f_R) = U_{ovo}(f_R) = U_{cons}(f_R) = \frac{1}{2}.$$

Proof. (i) There are $r!$ different ways of ordering a sequence of r objects and f_R randomly picks one of these rankings.

(ii) The other measures only compare object pairs. □

One must be careful in interpreting the values of different measures for a model. With a relatively high number of categories (say $r > 4$), a value of 0.5 for the volume under the ROC surface indicates that the ordinal regression model is able to rank the data well. Conversely, for few classes and especially in the binary classification case, a value of 0.5 for VUS alludes to absolutely no discriminative power for the ranking function. $\hat{U}_{pairs}(f, D)$, $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{cons}(f, D)$ all follow the behavior of the latter case and, hence, for poor ranking models and r relatively high, noticeably different values will be found in practice between VUS and its approximations (see also the following section). Additionally, the following relationship is observed between $\hat{U}_{pairs}(f, D)$ and $\hat{U}_{ovo}(f, D)$:

$$\hat{U}_{pairs}(f, D) = \frac{1}{\sum_{\bar{y}_k < \bar{y}_l} n_k n_l} \sum_{k < l} n_k n_l \hat{A}_{kl}(f, D). \quad (3.11)$$

As a result, both estimators are equal for balanced data sets. In the following section we will demonstrate that they can significantly differ for unbalanced data sets.

3.4 Experiments

Three kinds of experiments were set up to reveal the characteristics of $\hat{U}(f, D)$ and to make a comparison with the approximations and the standard measures accuracy and MAE. First we show by means of simulations on synthetic ranking functions that the relationship between $\hat{U}(f, D)$ and its approximations appears to be non-linear. In a second experiment we also utilize a synthetic data set to show that none of the investigated measures mutually displays a monotone relationship. The data set is kept balanced and ranking functions are randomly sampled over all permutations of the data.

In a last experiment we use real-world data to demonstrate that ranking optimization and error rate minimization are conflicting objectives for a classifier system when the data set is unbalanced. Therefore, all measures serve as fitness scores in a multi-objective stochastic search procedure concentrating on the region of the search space that represents good classifiers.

3.4.1 Simulation 1

In the first experiment we wanted to find out which values are obtained for different levels of separability and for an increasing number of classes.

Figure 3.3: Relationship between $\hat{U}(f, D)$ and $\hat{U}_{cons}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.

Therefore we assume that the function values of the model f can be represented by a distribution with cdf $F(\mathbf{x})$, in which the function values for the objects of class $\bar{y}_k$ are distributed with cdf $\bar{F}_k(\mathbf{x}) = \bar{F}(\mathbf{x} - kd)$. Furthermore, we sample from a Gaussian distribution with standard deviation $\sigma = 1$. So the function values conditioned on the labels are normally distributed with equidistant ordered means. Repeatedly, 100 data points were sampled from each class while we increased the distance d between the means of consecutive clusters. We started at $d = 0$ (random classifier) and stopped at $d = 5$ (as good as perfect separation) with step size 0.25.

The results obtained for $\hat{U}(f, D)$, $\hat{U}_{cons}(f, D)$ and $\hat{U}_{ovo}(f, D)$ are graphically compared. In this simulation all classes have the same prior of occurring, so $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{pairs}(f, D)$ will always have the same value due to (3.11). Hence, the results for $\hat{U}_{pairs}(f, D)$ are omitted. The relationship between $\hat{U}(f, D)$ and $\hat{U}_{cons}(f, D)$ on the one hand and between $\hat{U}(f, D)$ and $\hat{U}_{ovo}(f, D)$ on the other hand are respectively shown in Figures 3.3 and 3.4.

One can see that, as expected, these relationships are without doubt non-linear. As discussed at the end of the previous section, the average value of $\hat{U}(f, D)$ heavily depends on the number of classes, while this is not the case for the approximations. The approximations all take an average over a set of two-dimensional ROC-curves, so their average value is never lower than a

Figure 3.4: Relationship between $\hat{U}(f, D)$ and $\hat{U}_{ovo}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.

half, irrespective of the number of classes. Nevertheless, one can also see that $\hat{U}(f, D)$ converges rapidly to one when the distance between the subsequent means increases. In addition, $\hat{U}_{cons}(f, D)$ and $\hat{U}_{ovo}(f, D)$ behave quite similarly in this simulation. This is also shown in Figure 3.5. Their observed values become more dissimilar when the number of classes increases.

3.4.2 Simulation 2

From the previous experiment follows that $\hat{U}(f, D)$ and the other measures are not identically distributed. Next, we wanted to test whether they manifest a monotone relationship, which is as follows defined.

Definition 3.11. Let $\mathcal{F}$ be a family of functions $f : \mathcal{X} \rightarrow \mathbb{R}$, imposing a ranking over a data set $D \subset \mathcal{X} \times \mathcal{Y}$ and let $M_1, M_2 : \mathcal{F} \times (\mathcal{X} \times \mathcal{Y})^n \rightarrow \mathbb{R}$ be two performance measures on elements of $\mathcal{F}$, then there is a monotone relationship between M_1 and M_2 if and only if

$$\forall f_1, f_2 \in \mathcal{F} : M_1(f_1, D) \leq M_1(f_2, D) \Leftrightarrow M_2(f_1, D) \leq M_2(f_2, D).$$

In other words, we wanted to find out whether a ranking function which surpasses a second ranking function according to $\hat{U}(f, D)$ also surpasses that

Figure 3.5: Relationship between $\hat{U}_{cons}(f, D)$ and $\hat{U}_{ovo}(f, D)$ for $r = 1, \dots, 5$ and $d = 0, \dots, 5$ with step size 0.25. The values are averaged over 20 runs.

ranking function according to one of the approximations. The following experiment was set up to test whether this property holds for the four measures. All measures only quantify the quality of the ordering of a data set for a function f . For a data set of size n , there are $n!$ possible rankings of the objects, so evaluating them all is computationally intractable. Alternatively, we sampled randomly 1000 rankings from all possible orderings of the data set. We assumed we had 50 samples per class with four ordered classes, resulting in a sample size of 200 objects and 200! possible rankings. The results are given in Figure 3.6, which shows the distributions of all measures together with pairwise scatter plots.

All classes again have the same prior of occurring, so $U_{ovo}(f)$ and $U_{pairs}(f)$ have a perfect correlation. This is however not true for the other measures. One can clearly see that for no pair of measures the monotonic relationship holds. In general, $U(f)$, $U_{cons}(f)$ and $U_{ovo}(f)$ will have different maxima over a hypothesis space of ranking functions. So, optimizing one of the proposed approximations of $\hat{U}(f, D)$ will give rise to different classifiers.

Figure 3.6: An overview of the 1000 ranking functions randomly drawn from all possible orderings of a balanced data set of 200 objects with 4 categories. The histograms and pairwise scatter plots of the calculated values for $\hat{U}(f, D)$, $\hat{U}_{cons}(f, D)$, $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{pairs}(f, D)$ are shown.

3.4.3 Multi-objective optimization

In this experiment we wanted to find out whether optimizing the volume under the ROC surface will lead to different ordinal regression models compared to minimizing the error rate or mean absolute error. Contrary to the previous simulations real data was analyzed this time. We picked the *Boston housing* data set from the UCI Machine learning repository. This data set consists of 506 instances with 13 features and continuous labels. In previous studies on ordinal regression (Frank & Hall 2001, Chu & Keerthi 2005, Chu & Ghahramani 2005a) these continuous labels were adjusted to an ordinal value by subdividing the original data set into equal frequency bins after sorting. We could control the class frequencies in this way and precisely because we were interested in discovering the behavior of $\hat{U}(f, D)$ and other measures for unbalanced data, we chose a setting with five ordinal levels and a skew class distribution: $\pi_1 = 0.3, \pi_2 = 0.2, \pi_3 = 0.1$ and $\pi_4 = \pi_5 = 0.05$ with π_k the prior probability of observing an object of class $\bar{y}_k$. A simple linear model was considered as ranking model, i.e. $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. Together with four thresholds this resulted in a model with 17 free parameters. In order to discover the optimal values of $w_1, \dots, w_{13}$ and $b_1, \dots, b_4$ for the various per-

formance measures, a simple multi-objective stochastic algorithm, namely particle swarm optimization (MOPSO), was implemented. MOPSO (Coello & Lechunga 2002) is a relatively new multi-objective optimization technique inspired by the way large bird flocks navigate through the air and searches for a set of non-dominated solutions, the so-called Pareto front. The algorithm maintains a population of particles $\vec{\mathbf{p}} = (w_1, \dots, w_{13}, b_1, \dots, b_4)$ which will be adjusted in successive iterations according to the following update rules:

$$\begin{aligned}\vec{\mathbf{v}}_j &\leftarrow (\omega - \frac{\rho_1}{10})\vec{\mathbf{v}}_j + \rho_2(\vec{\mathbf{p}}_j^L - \vec{\mathbf{p}}_j) + \rho_3(\vec{\mathbf{p}}_j^G - \vec{\mathbf{p}}_j), \\ \vec{\mathbf{p}} &\leftarrow \vec{\mathbf{p}}_j + \text{trunc}(\vec{\mathbf{v}}_j).\end{aligned}$$

The velocities $\vec{\mathbf{v}}$ allow the particles to move to new positions in successive iterations. The function trunc truncates the components of $\vec{\mathbf{v}}$ that exceed the interval $[0, 1]$. The velocities on their part are guided toward the local best solution $\vec{\mathbf{p}}_j^L$, which is one of the non-dominated solutions found by particle $\vec{\mathbf{p}}_j$ so far, and toward the global best solution $\vec{\mathbf{p}}_j^G$, which is selected from the repository of non-dominated solutions found so far by all particles according to a selection scheme which assigns solutions lying in little explored regions higher chances of being selected. Besides, ω is an inertia weight (typically 0.4) and $\rho_1, \dots, \rho_3$ represent random numbers selected from a uniform distribution on the interval $[0, 1]$. Moreover, ρ_1 acts as a regularization term, controlling the complexity of the fitted models by pushing particles back towards the center of the search space (corresponding to a null model).

Six different objectives were considered, namely accuracy (or equivalently ‘1 - mean zero-one error’), ‘1 - mean absolute error’, $\hat{U}(f, D)$, $\hat{U}_{cons}(f, D)$, $\hat{U}_{ovo}(f, D)$ and $\hat{U}_{pairs}(f, D)$. The algorithm was executed 20 times for 100 iterations with a population of 500 particles and different seeds for the random generator. In all runs the non-dominated solutions found during the search were stored in a repository and afterwards the global non-dominated set of these 20 repositories was computed. This set is visualized by a matrix of two-dimensional Pareto fronts in Figure 3.7.

One can easily see that none of the six measures manifests a monotone relationship with another. Accuracy and mean absolute error on the one hand and the ranking-based measures on the other hand exhibit a relatively large trade-off, as almost all solutions lie on the two-dimensional front. The ranking-based performance measures give also rise to trade-offs, but here the monotonic association is more prominent. The multi-class approaches $\hat{U}_{cons}(f, D)$ and $\hat{U}_{ovo}(f, D)$ turn out to approximate the behavior of $\hat{U}(f, D)$ better than

Figure 3.7: The set of non-dominated solutions aggregated from 20 runs of the MOPSO-algorithm. The six-dimensional Pareto front is plotted as a matrix of two-dimensional scatter plots showing the trade-off for each pair of objectives.

simply counting all correctly ordered pairs. Apparently, for $\hat{U}_{pairs}(f, D)$ the optimal models are biased towards correctly ranking the biggest classes (due to the skew class distribution the data set contains only 640 object pairs of classes $\bar{y}_4$ and $\bar{y}_5$ compared to more than 15000 object pairs of the biggest classes $\bar{y}_1$ and $\bar{y}_2$). Methods minimizing the error rate or the number of incorrect instance pairs hence will both overfit on the biggest classes.

3.5 Conclusion

In this chapter we analyzed different performance measures for ranking-based ordinal regression models. We argued that evaluating the ranking returned by an ordinal regression model is often more appropriate than looking at ‘mean zero-one error’ or ‘mean absolute error’, especially with skew class or cost distributions. To that end, we extended the concept of expected ranking accuracy for ordinal labeled data and showed that a nonparametric unbiased estimator $\hat{U}(f, D)$ of this quantity corresponds to the volume under

the ROC surface spanned by the true positive rates of each class. Moreover, we revealed the relationship between $\hat{U}(f, D)$ and previous ranking-based performance measures, which can be considered as approximations of this statistic. The volume under the ROC surface and related measures do not manifest a monotone relationship and they also have a different distribution. Consequently, algorithms optimizing different criteria will lead to different models.

These observations were confirmed by experiments on synthetic and real data. In particular, a large trade-off was discovered between ‘mean zero-one error’ and ‘mean absolute error’ on the one hand and ranking-based measures on the other hand. The latter mutually displayed smaller trade-offs, but among them $\hat{U}_{pairs}(f, D)$ turned out to concentrate too much on the biggest classes.

We conclude that existing methods for ordinal regression, which typically minimize a loss based on error rate or the number of incorrectly ranked object pairs, might not construct appropriate models when the class or cost distributions are skew. ROC analysis offers in this case a valuable alternative allowing to pick a classifier from the surface for a specific setting of cost and the volume under the ROC surface gives a good overall indication of the quality of the model for different costs without favoring the majority classes. The results presented in this chapter have been published in (Waegeman *et al.* 2006) and (Waegeman *et al.* 2008g).

Scalability of multi-class ordered ROC analysis

In the previous chapter, the concept of a ROC curve was generalized to an r -dimensional surface for r ordered categories so that the volume under this ROC surface (VUS) measures the overall power of an ordinal regression model to classify objects of the various categories. However, the computation of this criterion as well as the U -statistics estimators of its variance and covariance for two models is believed to be complex.

New algorithms to compute VUS and its (co)variance estimator are presented in this chapter. In particular, the volume under the ROC surface can be found very efficiently with a simple dynamic program dominated by a single sorting operation on the data set. For the variance and covariance, the respective estimators are reformulated as a series of recurrent functions over layered data graphs and subsequently these functions are rapidly evaluated with a dynamic program. Simulation experiments confirm that the presented algorithms scale well with respect to the size of the data set and the number of categories. For example, the volume under the ROC surface could be rapidly computed on very large data sets of more than 500 000 instances, while a naive implementation spent much more time on data sets of size less than 1000.

In Section 4.1, we define representations for the variance and covariance of VUS together with U -statistics estimators of these quantities. In Section 4.2 the reformulation of the estimators as recurrent functions over graphs and the dynamic programming algorithms are presented. The results of time measurements on synthetic data are presented in Section 4.3.

4.1 Variance and covariance estimators of VUS

As mentioned, the generalization of ROC analysis presented in the previous chapter has been reported by Dreiseitl *et al.* (2000), Nakas & Yiannoutsos (2004) from a medical decision making perspective, in which the observed numbers are treated random variables instead of outcomes of an ordinal regression model. Based on U-statistics, these authors derive expressions for the variance of the volume under the ROC surface and the covariance of the volumes obtained for two different ranking functions f_1 and f_2 , as well as nonparametric estimators of these quantities. We briefly recapitulate their observations. To this end, let us define Υ as the set of all splits of $\mathcal{Y}$ into two (possibly empty) disjoint parts, i.e.

$$\Upsilon := \{(\mathcal{Z}_1, \mathcal{Z}_2) \mid \mathcal{Z}_1 \cup \mathcal{Z}_2 = \mathcal{Y} \wedge \mathcal{Z}_1 \cap \mathcal{Z}_2 = \emptyset\}.$$

Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a fixed ranking function and let $D \subset \mathcal{X} \times \mathcal{Y}$ be any data set of size n identically and independently distributed according to $\mathcal{D}$ with n_k instances sampled according to $\mathcal{D}_k$ as defined in Def. 3.4, then Nakas and Yiannoutsos define the variance of the volume under the ROC surface as:

$$\sigma_U^2(f) = \frac{1}{\prod_{k=1}^r n_k} \sum_{(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon} \left(\prod_{\bar{y}_l \in \mathcal{Z}_2} (n_l - 1) \right) (q_v(f, \mathcal{Z}_1, \mathcal{Z}_2) - U(f)^2).$$

In this expression $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ is still undefined. For each $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$, we introduce random variables X_i and X'_i as follows:

$$\begin{aligned} X_i &\sim \mathcal{D}_i, \quad \forall i : \bar{y}_i \in \mathcal{Y}, \\ X'_i &= X_i, \quad \forall i : \bar{y}_i \in \mathcal{Z}_1, \\ X'_i &\sim \mathcal{D}_i, \quad \forall i : \bar{y}_i \in \mathcal{Z}_2. \end{aligned}$$

Then $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ is defined as:

$$q_v(f, \mathcal{Z}_1, \mathcal{Z}_2) = \mathbf{Pr} \left\{ (f(X_1) < \dots < f(X_r)) \wedge (f(X'_1) < \dots < f(X'_r)) \right\}.$$

The measure $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ represents the probability that two r -tuples are correctly ranked by the ranking function f , in which each r -tuple is randomly sampled according to $\mathcal{D}_1 \times \dots \times \mathcal{D}_r$ with the restriction that the objects sampled from the categories $\bar{y}_k \in \mathcal{Z}_1$ are identical. Thus, we consider a single random variable for categories $\bar{y}_k \in \mathcal{Z}_1$ and two random variables for categories $\bar{y}_k \in \mathcal{Z}_2$. This expression for the variance is a natural extension of the one in the binary case (Hanley & McNeil 1982, 1983, Cortes & Mohri 2004).

The variance of the volume under the ROC surface still depends on the unknown distribution $\mathcal{D}$ of the data but can be estimated from a fixed data set $D \subset \mathcal{X} \times \mathcal{Y}$, by estimating $U(f)$ and $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ from D , i.e.

$$\hat{\sigma}_U^2(f, D) = \frac{1}{\prod_{k=1}^r n_k} \sum_{(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon} \left(\prod_{\bar{y}_l \in \mathcal{Z}_2} (n_l - 1) \right) (\hat{q}_v(f, D, \mathcal{Z}_1, \mathcal{Z}_2) - \hat{U}(f, D)^2).$$

We first introduce

$$\underline{n} := \prod_{k=1}^r n_k \prod_{\bar{y}_l \in \mathcal{Z}_2} n_l,$$

and a shorter notation for the indicator function of a given ranking function and a k -tuple, i.e.

$$I_f(j_1, \dots, j_k) := I_{\{f(\mathbf{x}_{j_1}) < \dots < f(\mathbf{x}_{j_k})\}}.$$

Using these notations, the following estimator for $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ is proposed:

$$\hat{q}_v(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{\underline{n}} \sum_{y_{j_1} < \dots < y_{j_r}} \sum_{\substack{y_{j'_1} < \dots < y_{j'_r} \\ \forall \bar{y}_k \in \mathcal{Z}_1: j_k = j'_k}} I_f(j_1, \dots, j_r) \cdot I_f(j'_1, \dots, j'_r).$$

This estimator counts the number of couples of r -tuples that are correctly ranked by f , with the restriction that the data objects sampled from $\mathcal{D}_k$ must be identical objects when $\bar{y}_k \in \mathcal{Z}_1$ (see Section 4.2 for an example).

A similar strategy can be followed to derive an expression for the covariance of the volumes obtained for two ranking functions f_1 and f_2 , i.e.

$$\text{Cov}_U(f_1, f_2) = \frac{1}{\prod_{k=1}^r n_k} \sum_{(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon} \left(\prod_{\bar{y}_l \in \mathcal{Z}_2} (n_l - 1) \right) (q_c(f, \mathcal{Z}_1, \mathcal{Z}_2) - U(f_1) \cdot U(f_2)) \quad (4.1)$$

and

$$q_c(f_1, f_2, \mathcal{Z}_1, \mathcal{Z}_2) = \mathbf{Pr} \left\{ (f_1(X_1) < \dots < f_1(X_r)) \wedge (f_2(X'_1) < \dots < f_2(X'_r)) \right\},$$

with $X_1, \dots, X_r, X'_1, \dots, X'_r$ random variables as formerly defined. We derive a nonparametric unbiased estimator in the same way as for the variance, i.e.

$$\hat{q}_c(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{\underline{n}} \sum_{y_{j_1} < \dots < y_{j_r}} \sum_{\substack{y_{j'_1} < \dots < y_{j'_r} \\ \forall \bar{y}_k \in \mathcal{Z}_1: j_k = j'_k}} I_{f_1}(j_1, \dots, j_r) \cdot I_{f_2}(j'_1, \dots, j'_r). \quad (4.2)$$

Then an unbiased estimator of the covariance $\widehat{\text{Cov}}_U(f_1, f_2, D)$ is obtained by inserting this estimator in (4.1).

4.2 Graph representation and intelligent computation of U-statistics

4.2.1 Motivation

In this section we will analyze the computational complexity of the estimators defined in the previous section. For the estimation of $U(f)$, all r -tuples are examined and hence, this number rapidly grows when the sample size or the number of categories increases. For a five class ordinal regression data set with 20 instances per class already 3,200,000 such r -tuples can be formed! In general, the number of different r -tuples to be analyzed on a data set of n elements and r categories is of the order $\mathcal{O}(n^r)$.

For the covariance (and consequently also for the variance as $\sigma_U^2(f) = \text{Cov}_U(f, f)$) all splits of $\mathcal{Y}$ are considered. The procedure for estimating $q_c(f_1, f_2, \mathcal{Z}_1, \mathcal{Z}_2)$ and $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ thus must be repeated 2^r times, but fortunately this is not a big issue since in realistic ordinal regression applications r will only occasionally exceed 10. Again, the inspection of r -tuples gives rise to the biggest bottleneck. Now couples of r -tuples with at most $2r$ different objects are investigated. As a result, the estimation of $q_c(f_1, f_2, \mathcal{Z}_1, \mathcal{Z}_2)$ and $q_v(f, \mathcal{Z}_1, \mathcal{Z}_2)$ by exhaustively examining all couples of r -tuples is here for some splits of $\mathcal{Y}$ of the order $\mathcal{O}(n^{2r})$.

Naive exhaustive implementations of the estimators for $U(f)$, its variance and covariance, in which the indicator function is verified on all r -tuples (or couples of r -tuples), will become computationally heavy for large data sets. Fortunately, much faster algorithms can be constructed by using some combinatorial tricks. To that end, the estimators are reformulated in terms of graphs. We start with recapitulating some basic material on graph theory.

4.2.2 Notations

Formally, let us define a *graph* $G = (V, E)$ as a data structure consisting of a set of *vertices* or nodes V and a set of *edges* E , where an edge $e \in E$ corresponds to a relation between two elements $v, v' \in V$, denoted as $e = (v, v')$. We will only consider *directed* graphs, which means that E contains couples (ordered pairs) of instances. For any edge (v, v') , the vertex

v is called a *direct predecessor* of v' and v' is called a direct successor of v . Furthermore, a vertex v is a predecessor of v' (and v' is a successor of v) if there is a *path* in the graph leading from v to v' . We will associate nodes with data objects of a data set D independently and identically distributed according to an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$.

In a *weighted* graph $G = (V, E, w)$ there is an additional function $w : E \rightarrow \mathbb{R}$ defined on the edges. Next, in a *bipartite* graph the set of vertices V can be split into two disjoint subsets V_1 and V_2 so that no two nodes from the same subset are connected. In a *multipartite* graph this property extends to splits of V into more than two subsets $V_1, \dots, V_k$. Finally, in a *layered* graph the set of vertices can be split into k ordered subsets (or layers) $V_1, \dots, V_k$ so that any edge connects nodes from successive layers. We will call a layered graph complete when any two nodes from successive layers are connected.

4.2.3 Algorithm for computing VUS

Definition 4.1. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a ranking function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$ and let $\mathcal{Y}$ contain r categories, then we define the U -graph $G_U(f, D) = (V, E)$ of f and D as a directed layered graph with $r + 2$ layers $V = (V_0, \dots, V_{r+1})$ in which every node of layer V_k is associated with one element in the data set of category $\bar{y}_k$, for $k = 1, \dots, r$. The first and last layers contain only a start node and an end node without a reference to any element of the data set ($V_0 = \{v_s\}, V_{r+1} = \{v_e\}$). Two nodes $v_1 = (\mathbf{x}_1, y_1)$ and $v_2 = (\mathbf{x}_2, y_2)$ from successive layers are connected if and only if $f(\mathbf{x}_1) < f(\mathbf{x}_2)$. Moreover, the start node is connected to all nodes of the layer V_1 and all nodes of the layer V_r are connected to the end node.

Lemma 4.2. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, let $G_U(f, D) = (V, E)$ be the corresponding U -graph and let $\Gamma : V \rightarrow \mathbb{R}$ be the recurrent function over this U -graph defined by:

$$\Gamma(v) = \sum_{(v', v) \in E} \Gamma(v')$$

with $\Gamma(v_s) = 1$, then

$$\hat{U}(f, D) = \frac{1}{\prod_{k=1}^r n_k} \Gamma(v_e).$$

Proof. The function $\Gamma(v)$ counts the number of distinct paths starting from v_s and ending in the node v . Each path from v_s to v_e uniquely defines a

	Data set																
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
y_i	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_1$	$\bar{y}_3$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_5$	$\bar{y}_4$	$\bar{y}_5$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_5$

Table 4.1: A hypothetical data set of 17 instances. We assume that the data set is sorted according to the ranking function.

correctly ordered r -tuple (according to f) of r elements of the data set from consecutive categories. As a result, $\Gamma(v_e)$ stands for the number of correctly ranked r -tuples. $\square$

Example 4.3. Table 4.1 gives an overview of a hypothetical data set with five categories. We assume that this data set is sorted according to a ranking function f , i.e.

$$i < j \Leftrightarrow f(\mathbf{x}_i) < f(\mathbf{x}_j).$$

In Figure 4.1 the corresponding U -graph is visualized.

Considering the U -graph it is possible to develop a $\mathcal{O}(n^2)$ algorithm for the computation of the volume under the ROC surface. However, this will not be done because it is even possible to design faster algorithms without constructing the U -graph. The U -graph will be used further on as a basic tool when we derive algorithms for the variance and the covariance.

Definition 4.4. Let $\mathcal{Z}_k = \{\bar{y}_1, \dots, \bar{y}_k\}$ contain the first k elements of $\mathcal{Y}$ and let $D \subset \mathcal{X} \times \mathcal{Y}$ be a data set sorted according to a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$, i.e.

$$\forall i, j \in \{1, \dots, n\} : i < j \Rightarrow f(\mathbf{x}_i) < f(\mathbf{x}_j),$$

then, for any $p \leq n$, we define the function Ψ by:

$$\Psi(f, D, \mathcal{Z}_k, p) = \sum_{\substack{y_{j_1} < \dots < y_{j_k} \\ \forall y_l \in \mathcal{Z}_k : j_l \leq p}} I_f(j_1, \dots, j_r).$$

$\Psi(f, D, \mathcal{Z}_k, p)$ can be interpreted as the number of correctly ordered k -tuples occurring in the first p elements of the data set D , where a k -tuple, as defined in Section 4.1, consists of a sequence of one randomly drawn element of each of the k lowest classes that is part of the first p elements of the data set D .

Figure 4.1: The U -graph for the hypothetical data set and ranking function of Table 4.1.

Proposition 4.5. *Given a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, $\widehat{U}(f, D)$ can be computed in $\mathcal{O}(n \log(n))$ time.*

Proof. We construct a fast algorithm by writing Ψ as a recurrence equation. Let the data set D be sorted according to f . From the definition of $\Psi(f, D, \mathcal{Z}, p)$ it follows that

$$\widehat{U}(f, D) = \frac{1}{\prod_{k=1}^r n_k} \Psi(f, D, \mathcal{Y}, n).$$

We can write $\Psi(f, D, \mathcal{Z}_k, p)$ as a recurrent function:

$$\Psi(f, D, \mathcal{Z}_k, p) = \begin{cases} \Psi(f, D, \mathcal{Z}_k, p-1) & , \text{if } y_p \neq \bar{y}_k, \\ \Psi(f, D, \mathcal{Z}_k, p-1) + \Psi(f, D, \mathcal{Z}_{k-1}, p-1) & , \text{if } y_p = \bar{y}_k, \end{cases} \quad (4.3)$$

for $k = 2, \dots, r$ and for $p = 1, \dots, n$. For $k = 1$ and $p = 1, \dots, n$, we have:

$$\Psi(f, D, \mathcal{Z}_1, p) = \begin{cases} \Psi(f, D, \mathcal{Z}_1, p-1) & , \text{if } y_p \neq \bar{y}_1, \\ \Psi(f, D, \mathcal{Z}_1, p-1) + 1 & , \text{if } y_p = \bar{y}_1. \end{cases}$$

In addition, the starting values ($p = 0$) are:

$$\Psi(f, D, \mathcal{Z}_k, 0) = 0,$$

for $k \in \{1, \dots, r\}$. Algorithm 1 computes $\widehat{U}(f, D)$ in $\mathcal{O}(n \log(n))$ time. Instead of implementing $\Psi(f, D, \mathcal{Y}, n)$ recursively this simple dynamic program computes the desired quantity in one iteration over the sorted data set. The algorithm is dominated by the sorting of D according to f , which takes $\mathcal{O}(n \log(n))$ time. $\square$

Algorithm 1 Computation of $\widehat{U}(f, D)$.

Input: data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, ranking function f

Sort D according to f

Initialize $\Psi(f, D, \mathcal{Z}_k) = 0$ for $k = 1, \dots, r$

for $j = 1$ to n **do**

y_j corresponds to category $\bar{y}_k$

if $k = 1$ **then**

$\Psi(f, D, \mathcal{Z}_1) \leftarrow \Psi(f, D, \mathcal{Z}_1) + 1$

end if

if $k > 1$ **then**

$\Psi(f, D, \mathcal{Z}_k) \leftarrow \Psi(f, D, \mathcal{Z}_k) + \Psi(f, D, \mathcal{Z}_{k-1})$

end if

end for

$$\widehat{U}(f, D) = \frac{1}{\prod_{k=1}^r n_k} \Psi(f, D, \mathcal{Z}_r)$$

Output: $\widehat{U}(f, D)$

Steps in the algorithm																	
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
y_i	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_1$	$\bar{y}_3$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_5$	$\bar{y}_4$	$\bar{y}_5$	$\bar{y}_3$	$\bar{y}_4$	$\bar{y}_5$
$\Psi(f_1, D, \mathcal{Z}_1)$	1	1	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3
$\Psi(f_1, D, \mathcal{Z}_2)$	0	1	1	1	3	3	3	3	6	6	6	6	6	6	6	6	6
$\Psi(f_1, D, \mathcal{Z}_3)$	0	0	0	1	1	4	4	4	4	10	10	10	10	10	16	16	16
$\Psi(f_1, D, \mathcal{Z}_4)$	0	0	0	0	0	0	4	4	4	4	14	14	24	24	24	40	40
$\Psi(f_1, D, \mathcal{Z}_5)$	0	0	0	0	0	0	0	0	0	0	0	14	14	38	38	78	78

Table 4.2: Algorithm 1 for the computation of $\widehat{U}(f, D)$ demonstrated on a hypothetical data set D_1 and fictive ranking function f_1 . For this data set the unknown ranking function results in 78 of the 432 different r -tuples that are correctly ranked, giving a volume under the ROC surface of 0.181.

Example 4.6. We demonstrate Algorithm 1 on the ranking function and hypothetical data set of Table 4.2. In this example the cardinality of $\mathcal{Y}$ is $r = 5$. The labels y_i of a fictive data set are shown and it is also assumed that the data set is sorted according to a fictive ranking function f_1 . Our algorithm computes the number of correctly ordered r -tuples in one iteration over the data. For example, the r -tuple $(1, 2, 4, 7, 12)$ is correctly ranked by f_1 and the r -tuple $(1, 5, 4, 7, 12)$ is incorrectly ranked because the object of class $\bar{y}_3$ precedes the object of class $\bar{y}_2$. An overview of the subsequent steps in the algorithm after sorting the data set is given and one can see that for this example 78 of the $3 \times 3 \times 4 \times 4 \times 3 = 432$ r -tuples are correctly ranked resulting in a volume of 0.181.

4.2.4 Algorithm for computing the covariance estimator

For the variance and covariance it is impossible to write the estimators in the form of a recurrence equation because now two rankings of the data are considered. We will first analyze the covariance because the variance can be handled as a special case for the algorithms that will be proposed further. To estimate the covariance, we introduce a graph for each $\hat{q}_c$ as defined by (4.2).

Definition 4.7. Let $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ be a split of $\mathcal{Y}$, let $f_1, f_2 : \mathcal{X} \rightarrow \mathcal{Y}$ be two functions that impose a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, then we define the q_c -graph $G_{q_c}(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2) = (V, E)$ as a directed layered graph with $r + 2$ layers $V = (V_0, \dots, V_{r+1})$. The first and last layers contain only a start node and an end node ($V_0 = \{v_s\}, V_{r+1} = \{v_e\}$). When category $\bar{y}_k$ belongs to the set $\mathcal{Z}_2$, then layer V_k contains n_k^2 nodes in which each node is associated with a pair of instances of category $\bar{y}_k$ in the data set. When category $\bar{y}_k$ belongs to the set $\mathcal{Z}_1$, then layer V_k has n_k nodes in which each node contains two references to a same instance with label $\bar{y}_k$. There is an edge leading from a node $v = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2))$ of layer V_k to a node $v' = ((\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4))$ of the consecutive layer V_{k+1} if and only if both $f_1(\mathbf{x}_1) < f_1(\mathbf{x}_3)$ and $f_2(\mathbf{x}_2) < f_2(\mathbf{x}_4)$. Moreover, the start node is connected to all nodes of layer V_1 and all nodes of layer V_r are connected to the end node.

Lemma 4.8. Let $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ be a split of $\mathcal{Y}$, let $f_1, f_2 : \mathcal{X} \rightarrow \mathcal{Y}$ be two functions that impose a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, let $G_{q_c}(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2) = (V, E)$ be the corresponding q_c -graph and let

$\Gamma : V \rightarrow \mathbb{R}$ be the recurrent function over the graph defined by:

$$\Gamma(v) = \sum_{(v',v) \in E} \Gamma(v')$$

with $\Gamma(v_s) = 1$, then

$$\widehat{q}_c(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{n} \Gamma(v_e).$$

Proof. The proof is based on the same observations as for the U-graph. Here we have to count all correctly ranked couples of r -tuples in which each r -tuple contains the same element at position k when $\bar{y}_k \in \mathcal{Z}_1$. With a correctly ranked couple we mean that the first and second r -tuple are correctly ranked, respectively by the ranking functions f_1 and f_2 . Each path leading from the start node to the end node now uniquely defines such a couple of correctly ranked r -tuples. The function $\Gamma(v)$ again counts the number of distinct paths connecting the start node and the node v and thus $\Gamma(v_e)$ computes the desired quantity. $\square$

Proposition 4.9. *Given two functions $f_1, f_2 : \mathcal{X} \rightarrow \mathbb{R}$ that impose a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, $\widehat{\text{Cov}}_U(f_1, f_2, D)$ can be computed in $\mathcal{O}(2^r n^4)$ time.*

Proof. Algorithm 2 computes the covariance based on (4.1), (4.2) and 2^r different q_c -graphs. We show that each $\widehat{q}_c$ can be computed in $\mathcal{O}(n^4)$ time. Two things have to be considered: the construction of such a q_c -graph and the evaluation of the recurrent function in its end node.

The algorithm builds up the set of vertices and edges of each graph starting at the start node. In the construction of subsequent layers most time is spent when both layers V_k and V_{k+1} are linked with a category belonging to $\mathcal{Z}_2$. In that case there are at most $n_k^2 \times n_{k+1}^2$ edges connecting the n_k^2 nodes of layer V_k to the n_{k+1}^2 nodes of layer V_{k+1} . When the layer V_k is implemented as a two-dimensional matrix indexed by two elements of D , then nodes can be retrieved in $\mathcal{O}(1)$ time. Edges are implemented as references to another node, inside the data structure of a node. Consequently, a q_c -graph can be constructed in $\mathcal{O}(n^4)$ time.

The recurrent function over the graph is implemented as a dynamic program flowing from the first layer to the last and having the benefit that the function is only once evaluated in each node. Again, most time is spent when the categories corresponding to consecutive layers are both associated with $\mathcal{Z}_2$.

For $\bar{y}_{k-1}, \bar{y}_k \in \mathcal{Z}_2$ the function is evaluated in n_k^2 nodes which are involved in at most n_{k-1}^2 edges.

Summarizing, both the graph construction and the evaluation of the recurrent function in the end node result in a worst case time complexity $\mathcal{O}(n^4)$. $\square$

Example 4.10. We illustrate the concept of q_c -graphs on a second hypothetical data set D_2 and two ranking functions f_1 and f_2 , shown in Figure 4.2. In this case the cardinality of $\mathcal{Y}$ is $r = 3$. Let us consider the computation of $\hat{q}_c(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2)$ with $\mathcal{Z}_1 = \{\bar{y}_1, \bar{y}_3\}$ and $\mathcal{Z}_2 = \{\bar{y}_2\}$. In this case we are looking for couples of sequences of three objects (one of each class) in which the first triplet is correctly ordered by f_1 and the second triplet is correctly ordered by f_2 with the restriction that the objects of class $\bar{y}_1$ and $\bar{y}_3$ are the same. The two objects of class $\bar{y}_2$ may be different objects. For example, the couple of triplets $((1, 5, 6), (1, 2, 6))$ is correctly ranked because the first triplet is correctly ranked by f_1 and the second one is correctly ranked by f_2 . On the other hand, the couple of triplets $((1, 2, 6), (1, 5, 6))$ is not correctly ranked because the 5-th and 6-th element are not correctly ranked by f_2 . Figure 4.3 displays all q_c -graphs constructed for D_2, f_1, f_2 and all splits of $\mathcal{Y}$ into $\mathcal{Z}_1$ and $\mathcal{Z}_2$. The q_c -graph for the split considered above corresponds to graph (c). One can see that the couple of triplets $((1, 5, 6), (1, 2, 6))$ defines a path connecting the start node and the end node. This is not the case for the couple of triplets $((1, 2, 6), (1, 5, 6))$ because there is no edge connecting the nodes linked with respectively the couples of data objects $(2/5)$ and $(6/6)$. In this graph one can find 4 unique paths leading from the start node to the end node. At most $\underline{n} = 2 \times 2^2 \times 2 = 16$ couples of triplets would be correctly ranked and hence for graph (c) $\hat{q}_c(f_1, f_2, D_2, \mathcal{Z}_1, \mathcal{Z}_2) = 0.25$. Similar arguments hold for other splits of $\mathcal{Y}$. By counting the number of paths in the 8 graphs one can find that $\widehat{\text{Cov}}_U(f_1, f_2, D_2) = 0$.

4.2.5 Algorithm for computing the variance estimator

Definition 4.11. Let $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ be a split of $\mathcal{Y}$ and let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, then the q_v -graph $G_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2)$ is defined as

$$G_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = G_{q_c}(f, f, D, \mathcal{Z}_1, \mathcal{Z}_2).$$

Corollary 4.12. Let $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ be a split of $\mathcal{Y}$, let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, let

Algorithm 2 Computation of $\widehat{\text{Cov}}_U(f_1, f_2, D)$.

Input: data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, ranking functions f_1, f_2
 Compute $\widehat{U}(f_1, D)$ and $\widehat{U}(f_2, D)$
 $\widehat{\text{Cov}}_U(f_1, f_2, D) = 0$
for $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ **do**
 // construct $G_{q_c}(f_1, f_2, D, \mathcal{Z}_1, \mathcal{Z}_2) = (V, E)$
 for $k = 1$ to r **do**
 if $\bar{y}_k \in \mathcal{Z}_1$ **then**
 $V = V \cup \{((\mathbf{x}_i, y_i), (\mathbf{x}_i, y_i)) \mid y_i = \bar{y}_k\}$
 end if
 if $\bar{y}_k \in \mathcal{Z}_2$ **then**
 $V = V \cup \{((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_i)) \mid y_i = \bar{y}_k \wedge y_j = \bar{y}_k\}$
 end if
 end for
 connect v_s to all nodes in V_1
 connect all nodes in V_r to v_e
 for $k = 1$ to $r - 1$ **do**
 for $v_a = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)) \in V_k$ **do**
 for $v_b = ((\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)) \in V_{k+1}$ **do**
 if $f(\mathbf{x}_1) < f(\mathbf{x}_3) \wedge f(\mathbf{x}_2) < f(\mathbf{x}_4)$ **then**
 $E = E \cup \{(v_a, v_b)\}$
 end if
 end for
 end for
 end for
 // compute the number of unique paths in the q_c -graph
 $\Gamma(v_s) = 1$
 for $k = 0$ to r **do**
 for $v_a \in V_k$ **do**
 for $v_b \in V_{k+1}$ **do**
 $\Gamma(v_b) = \Gamma(v_b) + \Gamma(v_a)$
 end for
 end for
 end for
 $\widehat{\text{Cov}}_U(f_1, f_2, D) = \widehat{\text{Cov}}_U(f_1, f_2, D) + \prod_{\bar{y}_l \in \mathcal{Z}_2} (n_l - 1) \left(\frac{\Gamma(v_e)}{n} - \widehat{U}(f_1, D) \widehat{U}(f_2, D) \right)$
end for
 $\widehat{\text{Cov}}_U(f_1, f_2, D) = \frac{1}{\sum_{k=1}^r n_k} \widehat{\text{Cov}}_U(f_1, f_2, D)$
 Output: $\widehat{\text{Cov}}_U(f_1, f_2, D)$

Data set						
i	1	2	3	4	5	6
y_i	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_1$	$\bar{y}_3$	$\bar{y}_2$	$\bar{y}_3$
(a) Hypothetical data set D_2						
i						
f_1	1	2	3	4	5	6
f_2	1	3	2	4	6	5
(b) Hypothetical rankings imposed by f_1 and f_2						

Figure 4.2: An example to illustrate q_c -graphs. Here the labels of a hypothetical data set and the ranking imposed by two ranking functions f_1 and f_2 is shown. The data set is sorted according to f_1 .

$G_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = (V, E)$ be the corresponding q_v -graph and let $\Gamma : V \rightarrow \mathbb{R}$ be the recurrent function over the graph defined by:

$$\Gamma(v) = \sum_{(v', v) \in E} \Gamma(v')$$

with $\Gamma(v_s) = 1$, then

$$\hat{q}_v(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{n} \Gamma(v_e).$$

Definition 4.13. Let $(\mathcal{Z}_1, \mathcal{Z}_2)$ be a split of $\mathcal{Y}$, let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, let us extend $\mathcal{Z}_1$ with two fictive categories $\bar{y}_0$ and $\bar{y}_{r+1}$ such that $\bar{y}_0 <_{\mathcal{Y}} \bar{y}_1$ and $\bar{y}_r <_{\mathcal{Y}} \bar{y}_{r+1}$, let us extend D with two dummy object-label pairs $(\mathbf{x}_0, y_0)$ and $(\mathbf{x}_{n+1}, y_{n+1})$ corresponding to respectively the head and the tail of the ordered data set with $y_0 = \bar{y}_0$, $y_{n+1} = \bar{y}_{r+1}$, $f(\mathbf{x}_0) = -\infty$ and $f(\mathbf{x}_{n+1}) = +\infty$, then the compressed q_v -graph $\tilde{G}_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = (\tilde{V}, \tilde{E}, w)$ is a complete weighted layered and directed graph with $|\mathcal{Z}_1|$ layers $\tilde{V}_0, \dots, \tilde{V}_{|\mathcal{Z}_1|-1}$ (all nodes of consecutive layers are connected). When category $\bar{y}_k$ corresponds to the l -th element of $\mathcal{Z}_1$, then layer $\tilde{V}_l$ has n_k nodes in which each node is associated with an instance in D with label $\bar{y}_k$. Instances in the data set for which the label is an element of $\mathcal{Z}_2$ are not linked to any node in the graph. The first and last layers contain only a start node and an end node ($\tilde{V}_0 = \{\tilde{v}_s\}$, $\tilde{V}_{|\mathcal{Z}_1|-1} = \{\tilde{v}_e\}$) and these nodes are respectively linked with the head and the tail. Given these extensions, the weight function $w : \tilde{E} \rightarrow \mathbb{R}$

Figure 4.3: The concept of q_c -graphs illustrated on the hypothetical rankings of Figure 4.2. The q_c -graphs $G_{q_c}(f_1, f_2, D_2, Z_1, Z_2)$ for all splits of $\mathcal{Y}$ into two disjoint sets Z_1 and Z_2 are shown.

for an edge $e = (v_1, v_2)$ where v_1 refers to an instance of category $\bar{y}_k$ and v_2 refers to an instance of category $\bar{y}_l$, can be formally defined as:

$$w(e) = \sum_{\substack{y_{j_{k+1}} < \dots < y_{j_{l-1}} \\ d_{\mathcal{Y}}(v_1) < y_{j_{k+1}} \\ y_{j_{l-1}} < d_{\mathcal{Y}}(v_2)}} I_f(d_{ind}(v_1), j_{k+1}, \dots, j_{l-1}, d_{ind}(v_2)). \quad (4.4)$$

The functions $d_{\mathcal{X}} : V \rightarrow \mathcal{X}$, $d_{\mathcal{Y}} : V \rightarrow \mathcal{Y}$ and $d_{ind} : V \rightarrow \mathbb{N}$ respectively return the feature vector, the label and the index of the corresponding instance in the data set.

The weight can be interpreted as the number of correctly ranked k -tuples ($k \leq r$) in the data set with the instance associated with v_1 as first element, the instance associated with v_2 as last element and the intermediate instances any element from categories $\bar{y}_{k+1}, \dots, \bar{y}_{l-1}$. Simultaneously, the weight also corresponds to the number of paths connecting v_1 and v_2 in the non-compressed graph. In the proof of the following lemma, we will need this observation.

Lemma 4.14. *Let $(\mathcal{Z}_1, \mathcal{Z}_2)$ be a split of $\mathcal{Y}$, let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, let $\tilde{G}_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = (\tilde{V}, \tilde{E}, w)$ be the corresponding compressed q_v -graph and let $\tilde{\Gamma} : \tilde{V} \rightarrow \mathbb{R}$ be the recurrent function over the graph defined by:*

$$\tilde{\Gamma}(v) = \sum_{e=(v',v) \in \tilde{E}} w^2(e) \tilde{\Gamma}(v')$$

with $\tilde{\Gamma}(v_s) = 1$, then

$$\hat{q}_v(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = \frac{1}{\underline{n}} \tilde{\Gamma}(\tilde{v}_e).$$

Proof. The proof is based on the relationship between $\tilde{G}_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2)$ and $G_{q_v}(f, D, \mathcal{Z}_1, \mathcal{Z}_2)$. By definition any node $v \in \tilde{V} \setminus \{\tilde{v}_s, \tilde{v}_e\}$ has an equivalent node in V in the sense that both nodes refer to the same data object. Such a node v will for ease of notation represent both the node in $\tilde{V}$ and V . On the other hand, we will respectively use different symbols to indicate the start node (v_s vs. $\tilde{v}_s$), end node (v_e vs. $\tilde{v}_e$) and recurrent functions (Γ vs. $\tilde{\Gamma}$) of the compressed and non-compressed q_v -graph. From Corollary 4.12 it follows that we have to prove that

$$\tilde{\Gamma}(\tilde{v}_e) = \Gamma(v_e), \quad (4.5)$$

with v_e the end node of the q_v -graph and $\tilde{v}_e$ the end node of the compressed q_v -graph. We consider two different cases.

Case 1: $\mathcal{Z}_1 = \{\bar{y}_0, \bar{y}_{r+1}\}$

In this case the compressed q_v -graph only contains a single start node $\tilde{v}_s$ and a single end node $\tilde{v}_e$. By definition, $\Gamma(v_e)$ counts the number of unique paths connecting the start node and the end node in the non-compressed graph. Each path in the non-compressed graph corresponds to a couple of correctly ranked and independently sampled r -tuples because all categories are part of $\mathcal{Z}_2$. The two (not necessarily) different r -tuples are independently sampled according to $\mathcal{D}_1 \times \dots \times \mathcal{D}_r$. The weight function, defined by (4.4), counts at the same time the number of correctly ranked r -tuples, i.e.

$$\begin{aligned} w((\tilde{v}_s, \tilde{v}_e)) &= \sum_{\substack{y_{j_1} < \dots < y_{j_r} \\ d_Y(\tilde{v}_s) < y_{j_1} \\ y_{j_r} < d_Y(\tilde{v}_e)}} I_f(d_{ind}(\tilde{v}_s), j_1, \dots, j_r, d_{ind}(\tilde{v}_e)) \\ &= \sum_{y_{j_1} < \dots < y_{j_r}} I_f(j_1, \dots, j_r), \end{aligned}$$

because the head and the tail are fixed and respectively ranked before the first instance of the data set and after the last instance of the data set. The number of correctly ranked couples of independently sampled r -tuples equals the squared number of individually correctly ranked r -tuple and, hence, the following relationship holds between the compressed and the non-compressed graph:

$$\Gamma(v_e) = \sum_{(v, v_e) \in E} \Gamma(v) = w^2((\tilde{v}_s, \tilde{v}_e)) = w^2((\tilde{v}_s, \tilde{v}_e)) \tilde{\Gamma}(\tilde{v}_s) = \tilde{\Gamma}(\tilde{v}_e).$$

Remark that in this case also

$$\Gamma(v_e) = w^2((\tilde{v}_s, \tilde{v}_e)) = \hat{U}(f, D)^2,$$

which is a necessary condition for $\hat{q}(f, D, \mathcal{Z}_1, \mathcal{Z}_2)$ to be an unbiased estimator.

Case 2: $\mathcal{Z}_1 \neq \{\bar{y}_0, \bar{y}_{r+1}\}$

In this case the compressed q_v -graph contains $|\mathcal{Z}_1| > 2$ layers $\tilde{V}_0, \dots, \tilde{V}_{|\mathcal{Z}_1|-1}$. To prove (4.5) we show by induction that for any data object with label belonging to the set $\mathcal{Z}_1$ the recurrent functions evaluated in its corresponding nodes in the compressed and the non-compressed q_v -graph, both denoted v , are equal, i.e.

$$\tilde{\Gamma}(v) = \Gamma(v). \quad (4.6)$$

Let the layer $\tilde{V}_1$ in the compressed q_v -graph correspond to category $\bar{y}_l$. Then for any $v \in \tilde{V}_1$ we find by applying (4.4):

$$w((\tilde{v}_s, v)) = \sum_{y_{j_1} < \dots < y_{j_{l-1}} < \bar{y}_l} I_f(j_1, \dots, j_{l-1}, d_{ind}(v)). \quad (4.7)$$

Because the first layer corresponds to category $\bar{y}_l$, the preceding categories $\bar{y}_1, \dots, \bar{y}_{l-1}$ are part of $\mathcal{Z}_2$ and hence they are represented by n_j^2 nodes for $\bar{y}_j$ in the non-compressed graph G_{q_v} . Each path from the start node v_s to v in the non-compressed graph now uniquely defines a couple of correctly ranked l -tuples (sequences of l data objects with one object of the first $l-1$ categories) in which the data object linked with v is the last element of both l -tuples. If the number of such unique l -tuples equals s_* , then the number of unique couples of l -tuples equals s_*^2 because all categories $\bar{y}_1, \dots, \bar{y}_{l-1}$ are part of $\mathcal{Z}_2$ and the last element is fixed in both l -tuples. Given (4.7), the definition of w for edges connecting the start node and the first layer in the compressed q_v -graph, this implies that

$$\Gamma(v) = s_*^2 = w^2((\tilde{v}_s, v)) = w^2((\tilde{v}_s, v))\tilde{\Gamma}(\tilde{v}_s) = \tilde{\Gamma}(v),$$

and thus (4.6) is fulfilled for the first layer $\tilde{V}_1$ in the compressed graph. Remark that this condition is trivially fulfilled when the first element of $\mathcal{Y}$ belongs to $\mathcal{Z}_1$ ($l=1$) because then $w((\tilde{v}_s, v)) = 1$ and thus $\Gamma(v) = \tilde{\Gamma}(v) = 1$. Suppose that (4.6) is fulfilled for a node v_1 of layer $\tilde{V}_i$ of the compressed graph. Then we show that the same condition is fulfilled for a node v_2 of the subsequent layer $\tilde{V}_{i+1}$. We assume that layer $\tilde{V}_i$ is associated with label $\bar{y}_k$ and layer $\tilde{V}_{i+1}$ with label $\bar{y}_l$. Now, the same reasoning can be applied as for V_1 and this time each path leading from v_1 to v_2 uniquely defines a couple of sequences of correctly ranked data objects of the classes $\bar{y}_k, \dots, \bar{y}_l$ with as first element the data object linked with v_1 and as last element the data object linked with the node v_2 . Analogously, the weight counts all such (individual) correctly ordered sequences. The squared weight again equals the number of couples of correctly ranked sequences because all couples of sequences are independently sampled from D according to $\mathcal{D}_{k+1} \times \dots \times \mathcal{D}_{l-1}$:

$$\begin{aligned} \Gamma(v_2) &= \sum_{v_1 \in V_k} w^2((v_1, v_2))\Gamma(v_1) \\ &= \sum_{v_1 \in V_k} w^2((v_1, v_2))\tilde{\Gamma}(v_1) = \tilde{\Gamma}(v_2). \end{aligned}$$

Hence, by induction (4.6) is satisfied for nodes of the last but one layer. To prove that (4.5) is satisfied for the end node, again the same reasoning can be

followed. Using (4.4) and (4.6) we find that

$$\begin{aligned}\Gamma(v_e) &= \sum_{v \in \tilde{V}_{|\mathcal{Z}_1|-1}} w^2((v, \tilde{v}_e)) \Gamma(v) \\ &= \sum_{v \in \tilde{V}_{|\mathcal{Z}_1|-1}} w^2((v, \tilde{v}_e)) \tilde{\Gamma}(v) = \tilde{\Gamma}(\tilde{v}_e).\end{aligned}$$

□

Proposition 4.15. *Given a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ that imposes a ranking on a data set $D \subset \mathcal{X} \times \mathcal{Y}$, $\hat{\sigma}_{\text{U}}^2(f, D)$ can be computed in $\mathcal{O}(2^r n^2)$ time.*

Proof. Algorithm 3 computes the variance of the volume under the ROC surface by constructing 2^r compressed q_v -graphs. When we consider the construction of such a graph, then contrary to the non-compressed graph, a layer associated with category $\bar{y}_k$ has not more than n_k nodes, which implies that at most $n_k \times n_l$ edges connect two layers that are respectively associated with the labels $\bar{y}_k \in \mathcal{Z}_1$ and $\bar{y}_l \in \mathcal{Z}_1$. The nodes and edges thus can be constructed in $\mathcal{O}(n^2)$ time.

The weight as well can be computed in $\mathcal{O}(n^2)$ time. A weight for any edge between two nodes in the compressed graph represents the number of unique paths connecting the corresponding nodes in the non-compressed graph. As shown in the algorithm all these weights can be computed in $\mathcal{O}(n^2)$ time on the assumption that the data was first sorted according to the ranking function (which takes $\mathcal{O}(n \log(n))$ time).

The evaluation of the recurrent function in the end node is carried out in the same way as for the U-graph and the q_c -graph leading here to $\mathcal{O}(n^2)$ operations.

As a result, the overall time complexity of Algorithm 3 is $\mathcal{O}(2^r n^2)$. □

Example 4.16. An example of the q_v -graph and compressed q_v -graph obtained on the data set and ranking function of Table 4.2 with $\mathcal{Z}_1 = \{\bar{y}_1, \bar{y}_2, \bar{y}_3\}$ and $\mathcal{Z}_2 = \{\bar{y}_4, \bar{y}_5\}$ is given in Figure 4.4. The difference in size between the non-compressed and compressed graphs catches the eye. We give an example to illustrate the computation of the weight function. Let us consider nodes 3 and 11 for example. They are respectively linked to a data object with label $\bar{y}_1$ and $\bar{y}_4$. As a result, the weight function counts the number of correctly ranked sequences of two objects of each of the classes $\bar{y}_2$ and

Algorithm 3 Computation of $\hat{\sigma}_U^2(f, D)$.

Input: data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, ranking function f
Sort D according to f
Compute $\hat{U}(f, D)$
 $\hat{\sigma}_U^2(f, D) = 0$
for $(\mathcal{Z}_1, \mathcal{Z}_2) \in \Upsilon$ **do**
 Extend $\mathcal{Z}_1$ with $\bar{y}_0$ and $\bar{y}_{r+1}$
 Extend D with a dummy head $(\mathbf{x}_0, y_0)$ and a dummy tail $(\mathbf{x}_{n+1}, y_{n+1})$
 $y_0 = \bar{y}_0, y_{n+1} = \bar{y}_{r+1}$
 // construct $\tilde{G}_{qv}(f, D, \mathcal{Z}_1, \mathcal{Z}_2) = (\tilde{V}, \tilde{E}, w)$
 $p = 0$
 for $k = 0$ to $r + 1$ **do**
 if $\bar{y}_k \in \mathcal{Z}_1$ **then**
 $\tilde{V}_p = \{(\mathbf{x}_i, y_i) \mid y_i = \bar{y}_k\}$
 $p = p + 1$
 end if
 end for
 for $p = 0$ to $|\mathcal{Z}_1| - 2$ **do**
 Connect nodes of consecutive layers V_p and V_{p+1} and set weights to zero
 $\bar{y}_k$ = category associated with layer $\tilde{V}_p$
 $\bar{y}_l$ = category associated with layer $\tilde{V}_{p+1}$
 if $l = k + 1$ **then**
 $\forall (\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) : y_i = \bar{y}_k \wedge y_j = \bar{y}_l \wedge f(\mathbf{x}_i) < f(\mathbf{x}_j) \Rightarrow w((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)) = 1$
 else
 for $y_i = \bar{y}_k$ **do**
 $w_h = 0$ for $h = 1, \dots, l - k - 1$
 for $j = i + 1$ to n **do**
 y_j corresponds to category $\bar{y}_m$
 if $m = k + 1$ **then**
 $w_1 = w_1 + 1$
 end if
 if $k + 1 < m < l$ **then**
 $w_{l-m} = w_{l-m} + w_{l-m-1}$
 end if
 if $m = l$ **then**
 $w((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)) = w_{l-1}$
 end if
 end for
 end for
 end if
 end for
 // compute the number of unique paths in the compressed qv -graph
 $\tilde{\Gamma}(\tilde{v}_s) = 1$
 for $k = 0$ to $|\mathcal{Z}_1| - 2$ **do**
 for $v_a \in \tilde{V}_k$ **do**
 for $v_b \in \tilde{V}_{k+1}$ **do**
 $\tilde{\Gamma}(v_b) = \tilde{\Gamma}(v_b) + \tilde{\Gamma}(v_a)$
 end for
 end for
 end for
 $\hat{\sigma}_U^2(f, D) = \hat{\sigma}_U^2(f, D) + \prod_{\bar{y}_l \in \mathcal{Z}_2} (n_l - 1) \left(\frac{\tilde{\Gamma}(\tilde{v}_e)}{n} - \hat{U}(f, D)^2 \right)$
end for
 $\hat{\sigma}_U^2(f, D) = \frac{1}{\prod_{k=1}^r n_k} \hat{\sigma}_U^2(f, D)$
Output: $\hat{\sigma}_U^2(f, D)$

$\bar{y}_3$ located between the 3rd and the 11th position in the sorted data set. Three sequences meet this condition: $(3, 5, 6, 11)$, $(3, 5, 10, 11)$, $(3, 9, 10, 11)$. Combining these sequences gives rise to 9 different sequence pairs and this is exactly the number of paths connecting the nodes 3/3 and 11/11 in the non-compressed graph.

4.3 Experiments

4.3.1 General setup

In the foregoing section we drew strong theoretical conclusions on the scalability of the dynamic programming algorithms for computing the volume under the ROC surface, its variance and the covariance for two ranking functions. Algorithms 1-3 as well as their exhaustive counterparts were implemented in JAVA to verify their scalability empirically. Subsequently, we carried out some simulations to investigate the running time of the exhaustive and dynamic programming algorithms.

In all experiments we adopted the following setup. Let X_k be a random variable with distribution $\mathcal{D}_k$ — the random variable X_k thus represents a random data object from class $\bar{y}_k$ — then we assume that the output value of the ranking function for that random variable follows a normal distribution, i.e.

$$f(X_k) \sim N(\mu_k, \sigma^2). \quad (4.8)$$

The variance of this normal density will be fixed to $\sigma^2 = 0.25$ for different categories in all experiments and the mean will be set to $\mu_k = k$ for each category $\bar{y}_k$. In this setting the ranking function mainly assigns positive values to data objects and the output tends to increase for the last categories. Many of the data objects will be correctly classified by a model $h : \mathcal{X} \rightarrow \mathcal{Y}$ as defined in Section 4.1, but still there is an overlap of the output values for different classes. To our opinion, this is a realistic reproduction of the behavior of a ranking function.

The scalability of all algorithms was verified by measuring their running time with increasing sample size. Such a data set was generated in the way described above in which the ranking function output was randomly sampled according to the normal densities of (4.8) without fitting an ordinal regression model to the data. This procedure was repeated for two to five ordered

(a) non-compressed q_v -graph

(b) compressed q_v -graph

Figure 4.4: An illustration of the difference between a non-compressed (a) and a compressed (b) q_v -graph. In this example the data set and ranking function of Table 4.2 are used and the split of $\mathcal{Y}$ is as follows: $\mathcal{Z}_1 = \{\bar{y}_1, \bar{y}_4\}$, $\mathcal{Z}_2 = \{\bar{y}_2, \bar{y}_3, \bar{y}_5\}$. In the compressed q_v -graph, edges with weight zero are discarded.

categories, so the binary classification case was also considered. In all experiments the fraction of data instances of each class was kept equal. The experiments were carried out on a cluster of processors and the results were averaged over twenty runs of the algorithms in which a single processor was used for each run. In this way the randomness caused by the sampling of the output values was counterbalanced. We used standard JAVA libraries to measure the running time of the different implementations.

4.3.2 Results for VUS

First we analyzed both algorithms for computing the volume under the ROC surface. In some initial experiments we immediately recorded a tremendous difference in running time between both algorithms. The difference was so striking that for relatively small data sets the running time of the dynamic program did not exceed ten milliseconds while its exhaustive counterpart already gave evidence of a considerable amount of time consumption. At a running time of only ten milliseconds time measurement becomes vulnerable to system operations which are independent of the algorithm and lead to imprecise measurements. As a result, no smooth time curves were observed for the dynamic program for these small sample sizes and therefore both algorithms were tested with different settings for the sample size.

For the exhaustive algorithm we linearly increased the size of the data set from 60 to 600 examples in ten steps. For the dynamic program much bigger data sets were considered starting at an initial size of 60,000 examples and linearly increasing to 600,000 examples in ten steps. The results observed for r varying from two to five are respectively shown in Figures 4.5 and 4.6. For the computation of $\widehat{U}(f, D)$ the dynamic program gives cause to very compact source code, which explains why it is remarkably faster for small sample sizes. More importantly, our algorithm also scales much better when the sample size grows. This can be seen easily from the figures as incrementing the sample size with 600 examples for the exhaustive implementation already leads to a faster increase in running time than incrementing the dynamic programming version with 60,000 examples¹. As expected, we observe that our algorithm is more or less independent of the number of categories. For different values of r it scales similarly. This is absolutely not the case for the other implementation: it largely depends on r .

¹Remark that the representation of running time on a log-scale might be a bit misleading when comparing different algorithms. However, the measured curves for $r = 2, \dots, 4$ would coincide with the horizontal axis in Figure 4.5 on a linear scale.

Figure 4.5: The running time of an exhaustive algorithm for computing $\hat{U}(f, D)$ with an increasing data set size and a different number of classes. The size varied from 60 to 600 instances, adding 60 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.

Figure 4.6: The running time of Algorithm 1 with an increasing data set size and a different number of classes. The size varied from 60,000 to 600,000 instances, adding 60,000 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.

Figure 4.7: The running time of an exhaustive algorithm for computing $\hat{\sigma}_U^2(f, D)$ with an increasing data set size and a different number of classes. The size varied from 5 to 50 instances, adding at most 5 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.

4.3.3 Results for the variance and covariance estimators

Subsequently, the algorithms for computing the variance and covariance were compared to the exhaustive counterpart. Since

$$\hat{\sigma}_U^2(f, D) = \widehat{\text{Cov}}_U(f, f, D),$$

we could measure the running time of the covariance estimator with a single ranking function, because we are only interested in its scalability. For the implementation with the compressed q_v -graph we sampled much bigger data sets than for the other two algorithms because again the difference was too striking. The sample size was linearly increased from 5 to 50 instances in ten steps for the exhaustive algorithm and for Algorithm 2. For Algorithm 3 we started with an initial size of 500 and ended with 5,000 examples. If necessary, the sample size was rounded down to guarantee an equal number of objects of each category, which means that for example with $r = 4$ and $n = 30$ in total 28 instances were sampled (7 for each class).

The measured running times are respectively shown in Figures 4.7, 4.8 and 4.9. The experiments basically confirm our theoretical observations. They

Figure 4.8: The running time of Algorithm 2 with an increasing data set size and a different number of classes. The size varied from 5 to 50 instances, adding at most 5 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.

Figure 4.9: The running time of Algorithm 3 with an increasing data set size and a different number of classes. The size varied from 500 to 5000 instances, adding 500 examples in each step. The running time was plotted on a log-scale and the size of the data set on a linear scale.

clearly indicate that Algorithm 3, which estimates $\hat{\sigma}_V^2(f, D)$ with a series of compressed q_v -graphs, is much faster than the other algorithms. Adding 5,000 extra instances leads for this algorithm to an even smaller increase in running time than adding 50 examples for the other algorithms, which makes Algorithm 3 applicable to very large data sets. Neither it depends on the number of categories considered, which can be seen as an important advantage.

As expected the exhaustive procedure for computing the variance performs the worst of all three methods and in particular its running time increases fast for more than two categories. Apparently, Algorithm 2, which computes the variance based on a series of non-compressed graphs, also scales rather poorly since it is of the order $\mathcal{O}(2^r n^4)$. We observed a less smooth curve for Algorithm 2 than for the other methods. This phenomenon appears because, unlike the other algorithms, the running time of Algorithm 2 depends on the ordering imposed by the ranking function or, similarly, the number of edges in the q_v -graphs.

The highest running times for Algorithm 2 are observed when r is low. This might be surprising but can be explained by our experimental setup. For a fixed balanced data set the number of data objects per class n_k decreases when the number of classes increases. Contrary to the exhaustive algorithm, the running time of Algorithm 2 rather depends on n_k instead of n and consequently it is not entirely correct to compare both algorithms for equal sample sizes. Nonetheless, the exhaustive implementation and Algorithm 2 complement each other: the first one is more appropriate when r is small and the second one in the other case.

4.4 Conclusion

In this chapter, we analyzed the complexity of computing the volume under the ROC surface (VUS), its variance estimator and covariance estimator for two volumes. For the computation of VUS, we proposed a $\mathcal{O}(n \log(n))$ dynamic programming algorithm dominated by a single sorting operation on the data set. The estimators of variance and covariance were reformulated in terms of recurrent functions over graphs and we showed that with dynamic programming techniques for evaluating these recurrent functions, much faster algorithms could be constructed compared to exhaustive algorithms that evaluate all r -tuples. In particular, respectively $\mathcal{O}(2^r n^2)$ and

$\mathcal{O}(2^r n^4)$ algorithms were designed for estimating the variance and the covariance.

Simulations confirmed our theoretical observations and showed that the algorithms for computing $\hat{U}(f, D)$ and $\hat{\sigma}_U^2(f, D)$ are applicable to very large data sets. Our algorithm for computing the covariance estimator resulted in a smaller time gain compared to an exhaustive algorithm, but still scaled well with increasing r .

The algorithms presented in this chapter are useful for different purposes. First of all, they make it possible to perform ROC analysis on large data sets and to estimate $U(f)$ on large ordinal regression models. They can be directly plugged into the statistical tests developed by Nakas & Yiannoutsos (2004). With these algorithms, new ranking methods can be constructed for using or optimizing the volume under the ROC surface (that is exactly what we will do in the following chapter). Such algorithms have been proposed for the binary case and in particular optimizing ranking-based measures can lead to noticeably different models compared to minimizing the error rate (Yan *et al.* 2003, Cortes & Mohri 2003, Herschtal & Raskutti 2005). Furthermore, the variance and covariance can provide important criteria to choose a particular ranking function in the model selection phase. The variance of the AUC is for example used in (Agarwal *et al.* 2005) to derive distribution-free generalization bounds for the AUC. The covariance gives an indication of the similarity between two ranking functions.

The results presented in this chapter have been published in (Waegeman *et al.* 2008f).

Ranking learning with VUS as a loss function

The relationship between ranking algorithms, graph theory and ROC analysis has been established in the two previous chapters. In this chapter, we discuss how VUS can be embedded as a loss function in a learning algorithm. Unlike the conventional approach of minimizing the pairwise error, we present a new type of kernel method based on structured support vector machines that tries to optimize the fraction of correctly ranked r -tuples. A large number of constraints appear in the resulting quadratic program, but the optimal solution can be computed in $\mathcal{O}(n^3)$ time for samples of size n with a cutting plane algorithm and graph-based techniques. Our approach can offer benefits for applications in various domains. On various synthetic and benchmark data sets, it outperforms the pairwise approach for balanced as well as unbalanced problems. In addition, scaling experiments confirm the theoretically derived time complexity.

5.1 Criticism on previous approaches

In recent years, the supervised learning problem of ranking data has been studied under a variety of settings in the machine learning community. The setting in which objects come from two categories, positive and negative, and in which the goal is to learn a (continuous) function that ranks positive instances higher than negative ones, is sometimes called the *bipartite* ranking problem (Freund *et al.* 2003, Agarwal *et al.* 2005), because the ranking can be represented as a bipartite ranking graph, in which the two subsets of vertices correspond to objects of respectively the positive and negative category and

the edges connect correctly ranked couples with one object of each category. All binary classifiers with an underlying discriminant function induce such a bipartite graph. Actually, binary scoring classifiers that optimize the AUC, such as the methods proposed in (Yan *et al.* 2003, Brefeld & Scheffer 2005, Joachims 2005, Herschtal & Raskutti 2005, Ataman *et al.* 2006) can be seen as bipartite ranking algorithms because the AUC is derived from the bipartite graph by counting the number of edges (Agarwal *et al.* 2005). Consequently, much work on ROC analysis and ranking in machine learning is very related. Similarly to previous chapters, we will discuss the more general ordinal regression setting of ranking objects sampled from in general r ordered categories, a setting we will call layered ranking due to a strong relationship with layered graphs. To our opinion, ordinal regression and layered ranking can be distinguished in a similar way as binary classification and bipartite ranking: while in classification one is primarily interested in learning a model predicting class labels to unknown objects, the goal in ranking rather consists of finding a continuous function that imposes an ordering on the data. Optimizing a loss function deduced from these two different approaches will therefore typically lead to different models. As described before, a popular way to define such a ranking-based loss function is looking at all (lower-class; higher-class) couples that are incorrectly ranked by the model. For example, the articles (Herbrich *et al.* 2000, Rennie & Srebro 2005, Rajaram & Agarwal 2005, Joachims 2006) all consider such a loss function.

Evaluating the ranking by pairwise comparisons might in many cases not be the best option. This is for example the case in information retrieval applications, where typically one is primarily interested in the top of the ranking. Measures like normal discounted cumulative gain and mean average precision better reflect the desired performance in information retrieval applications. To this end, several ranking methods that concentrate on learning the top of the ranking have been proposed recently, see e.g. (Cao *et al.* 2007, Clmenon & Vayatis 2007).

The idea of focussing on correctly predicting the top of the ranking might be advisable for other application domains as well, but correctly predicting one end of the ranking does not have to be important in general. One could be interested in correctly predicting the top and the tail simultaneously, or, in other situations, all r categories could be equally important. The first situation for example arises when the ordinal levels express a linguistic degree of uncertainty about the category of the object, which is typical for label assignments by humans.

When the goal does not consist of predicting the top end of the ranking correctly, the traditional pairwise approach might manifest important shortcomings as well. The main reason is that the pairwise approach, as well as the multi-class AUC, subdivides the performance evaluation into evaluations of pairs of classes individually. However, the observed trend might be substantially different for each pair of categories. Let us illustrate this claim with a rather extreme example. Table 5.1 gives an overview of a hypothetical ranking of a data set containing 18 instances from three ordered categories. Computing the AUC for each pair of categories yields:

$$\begin{aligned}\hat{A}_{12}(f, D) &= 0.556, \\ \hat{A}_{23}(f, D) &= 0.694,\end{aligned}\tag{5.1}$$

$$\hat{A}_{13}(f, D) = 0.419.\tag{5.2}$$

Intuitively, one would expect that

$$\hat{A}_{13}(f, D) \geq \max\{\hat{A}_{12}(f, D), \hat{A}_{23}(f, D)\}.$$

In preference modelling, this property is known as strong stochastic transitivity. However, the triplet of pairwise AUCs as defined in Chapter 3 is in the example even not weakly stochastically transitive, which would require that

$$\hat{A}_{1,3}(f, D) \geq \frac{1}{2}.$$

The pairwise AUCs can be interpreted as a generalized dice model for comparing independent random variables and they exhibit a specific type of cycle-transitivity, namely dice-transitivity (De Schuymer *et al.* 2003, 2005), which is much weaker than strong stochastic transitivity. In Chapter 6, we will discuss in detail the transitivity of pairwise AUCs and the connection with dice models in order to establish a link between ranking models and multi-class classification models. Here we only want to illustrate that both $\hat{U}_{pairs}(f, D)$ and $\hat{U}_{ovo}(f, D)$ can yield quite inconsistent performance evaluations with regard to the performance on each pair of categories individually, because they are constructed from binary AUCs. This effect was for example observed for unbalanced data sets in (Xu *et al.* 2006). When the data set is very unbalanced, models minimizing the pairwise error might overfit on the most occurring categories and these categories are often not the categories wherein one is the most interested.

	Ranking																	
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
y_i	$\bar{y}_3$	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_1$

Table 5.1: A hypothetical example of a data set containing 18 objects with three possible labels to illustrate the inconsistency that can occur when the ranking function is evaluated with pairwise comparisons. The ordering of the objects in the table represents the obtained ranking, increasing from left to right (example taken from (De Schuymer et al. 2005)).

5.2 A convex optimization framework for $\hat{U}(f, D)$

Following the ideas introduced in the previous chapters, we aim to assess the ranking in a global way by means of VUS. In order to convert VUS into a loss function that can be optimized efficiently, we need to extend the U-graph as introduced in Def. 4.1 from an unweighted to a weighted graph.

Definition 5.1. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on an i.i.d. data set D and let $\mathcal{Y}$ contain r categories, then we define the Δ -graph $G_\Delta(f, D) = (V, E)$ of f and D as a directed weighted layered graph with $r+2$ layers $V = (V_0, \dots, V_{r+1})$ in which a node of layer V_k is associated with any element in the data set of category $\bar{y}_k$, for $k = 1, \dots, r$. The first and last layers contain only a start node (source) and an end node (sink) without a reference to any element of the data set ($V_0 = \{v_s\}$, $V_{r+1} = \{v_e\}$). All nodes of successive layers are connected and the weight of the edge $e = (v_1, v_2)$ connecting the nodes $v_1 = (\mathbf{x}_1, y_1) \in V_k$ and $v_2 = (\mathbf{x}_2, y_2) \in V_{k+1}$ is defined as:

$$w(e) = \begin{cases} +\infty, & \text{if } k \in \{0, r\} \\ f(\mathbf{x}_2) - f(\mathbf{x}_1), & \text{if } k \in \{1, \dots, r-1\}. \end{cases}$$

Definition 5.2. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on an i.i.d. data set D , let $G_\Delta(f, D) = (V, E)$ be the corresponding Δ -graph, then we define a path ϖ between v_s and v_e as a sequence of $r+1$ edges $\varpi = (e_1, \dots, e_{r+1})$ connecting v_s and v_e , i.e.

$$\begin{aligned} e_1 &= (v_s, v_1), \\ e_2 &= (v_1, v_2), \\ &\vdots \\ e_r &= (v_{r-1}, v_r), \\ e_{r+1} &= (v_r, v_e). \end{aligned}$$

The set of all paths connecting v_s and v_e will be denoted Ξ .

If we replace the U -graph by the Δ -graph, then Lemma 4.2 becomes

Corollary 5.3. *Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a function that imposes a ranking on an i.i.d. data set D , let $G_\Delta(f, D) = (V, E)$ be the corresponding Δ -graph with Ξ the set of paths connecting source and sink, then*

$$\widehat{U}(f, D) = \frac{|\{\varpi \in \Xi \mid \forall e \in \varpi : w(e) > 0\}|}{\prod_{k=1}^r n_k}.$$

Given a data set D we will utilize a shorthand notation for $\mathbf{j} = (j_1, \dots, j_r)$ to denote an r -tuple corresponding to a sequence of r objects, one of each category. We introduce $\mathcal{S}$ as the set of all r -tuples in D , i.e.

$$\mathcal{S} := \{\mathbf{j} = (j_1, \dots, j_r) \mid y_{j_1} < \dots < y_{j_r}\},$$

and the symbol $s = |\mathcal{S}| = \prod_{k=1}^r n_k$ will represent the cardinality of this set.

5.2.1 VUS as a loss function

We want to develop a learning algorithm that optimizes $\widehat{U}(f, D)$ or, equivalently, that learns a layered graph with a maximal number of paths connecting source and sink. It means that we will convert $\widehat{U}(f, D)$ into a suitable loss function in order to embed it into the framework of kernel methods. We add the large margin principle to our framework by constructing a loss function from $\widehat{U}(f, D)$. Therefore, we will define a single slack variable for each r -tuple in the data set. Let us consider such an r -tuple $\mathbf{j}$ of data objects, then we define the corresponding slack variable $\xi_{\mathbf{j}}$ as

$$\xi_{\mathbf{j}} = \max \left\{ 0, \max_{k \in \{1, \dots, r-1\}} \{1 - (f(\mathbf{x}_{j_{k+1}}) - f(\mathbf{x}_{j_k}))\} \right\}.$$

This means that the slack variable $\xi_{\mathbf{j}}$ of an r -tuple $\mathbf{j}$ will differ from zero only when the difference of the function values of any couple of consecutive elements of the r -tuple is less than one. In particular, $\xi_{\mathbf{j}}$ will take 1 minus the lowest value of these $r - 1$ differences in function values for any couple of consecutive objects in the r -tuple $\mathbf{j}$, or, in terms of the Δ -graph, $\xi_{\mathbf{j}}$ equals 1 minus the weight of the edge with the lowest weight on the path from source to sink defined by the r -tuple $\mathbf{j}$.

Using the concepts introduced in Chapter 2, let us consider learning in reproducing kernel Hilbert spaces $\mathcal{F}$ induced by a kernel function $K : \mathcal{X}^2 \rightarrow \mathbb{R}$

and let us minimize a regularized risk functional of type (2.6) with $L(f, D)$ a sample-based loss which is now defined as

$$L(f, D) = \sum_{\mathbf{j} \in \mathcal{S}} \xi_{\mathbf{j}},$$

and $J : \mathcal{F} \rightarrow \mathbb{R}$ the standard penalty functional in all kernel methods, i.e.

$$J(f) = \frac{1}{2} \|\mathbf{w}\|^2.$$

Similar to the C -formulation of the support vector machine (C -SVM) or the Ranking SVM, we arrive at the following primal optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi_{\mathbf{j}}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{\mathbf{j} \in \mathcal{S}} \xi_{\mathbf{j}} \\ \text{subject to} \quad & \begin{cases} \min_{k \in \{1, \dots, r-1\}} \{\mathbf{w} \cdot \delta(\mathbf{j}, k)\} \geq 1 - \xi_{\mathbf{j}} \\ \xi_{\mathbf{j}} \geq 0 \\ \mathbf{j} \in \mathcal{S} \end{cases} \end{aligned} \quad (5.3)$$

in which we used the shorthand notation $\delta(\mathbf{j}, k) := \phi(\mathbf{x}_{j_{k+1}}) - \phi(\mathbf{x}_{j_k})$. This optimization problem can be solved by a QP-solver by deriving the dual quadratic program, and in the dual formulation, the ranking function $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})$ can be written only in terms of kernels and Lagrange multipliers. Unfortunately, the large number of constraints gives rise to an identical number of Lagrange multipliers in the dual formulation, and hence, the running time required by a QP-solver will be excessively high, especially when the number of categories increases. However, we will show that the same solution can be found within a reasonable amount of time by using an approximation algorithm that only considers a subset of the constraints.

5.2.2 Reformulation as a structured Support Vector Machine

Recently, Tsochantaridis *et al.* (2005) and Joachims (2005) proposed a general support vector algorithm for structured outputs and multivariate performance measures which basically consists of the optimization of a quadratic program with a combinatorial number of constraints. Instead of optimizing this huge quadratic program, they start optimizing the unconstrained objective function and iteratively add new constraints to the optimization problem.

We will follow a similar approach. Let us therefore consider the following optimization problem:

$$\begin{aligned} & \min_{\mathbf{w}, \xi} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \\ & \text{subject to} \quad \begin{cases} \mathbf{w} \cdot \left(\frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right) \geq \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \xi \\ \xi \geq 0 \\ \mathbf{j} \in \mathcal{S} \\ \mathbf{c} \in \{0, 1\}^s \\ \mathbf{t} \in \{1, \dots, r-1\}^s \end{cases} \end{aligned} \quad (5.4)$$

with $\mathbf{w}$, C and $\mathcal{S}$ as defined before. Remarkably, it contains only one slack variable. The following proposition reveals the connection between both optimization problems.

Proposition 5.4. *The solutions of optimization problems (5.3) and (5.4) result in identical models $\mathbf{w}$ and their errors are related: $\xi = \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} \xi_{\mathbf{j}}$.*

Proof. The proof is based on (Tsochantaridis *et al.* 2005). For a fixed $\mathbf{w}$, the optimal ξ is given by:

$$\xi = \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1,\dots,r-1\}^s}} \left\{ \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \mathbf{w} \cdot \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right\}.$$

All components of the vectors $\mathbf{c}$ and $\mathbf{t}$ can be optimized independently:

$$\xi = \sum_{\mathbf{j} \in \mathcal{S}} \max_{\substack{c_{\mathbf{j}} \in \{0,1\} \\ t_{\mathbf{j}} \in \{1,\dots,r-1\}}} \left\{ \frac{1}{s} c_{\mathbf{j}} - \frac{1}{s} c_{\mathbf{j}} (\mathbf{w} \cdot \delta(\mathbf{j}, t_{\mathbf{j}})) \right\} \quad (5.5)$$

$$= \sum_{\mathbf{j} \in \mathcal{S}} \max_{c_{\mathbf{j}} \in \{0,1\}} \left\{ \frac{1}{s} c_{\mathbf{j}} - \frac{1}{s} c_{\mathbf{j}} \min_{k \in \{1,\dots,r-1\}} \{ \mathbf{w} \cdot \delta(\mathbf{j}, k) \} \right\}. \quad (5.6)$$

Because $\xi \geq 0$ and $\xi_{\mathbf{j}} \geq 0$, it holds that

$$\begin{aligned} \xi &= \sum_{\mathbf{j} \in \mathcal{S}} \max \left\{ 0, \frac{1}{s} - \frac{1}{s} \min_{k \in \{1,\dots,r-1\}} \mathbf{w} \cdot (\phi(\mathbf{x}_{j_{k+1}}) - \phi(\mathbf{x}_{j_k})) \right\} \\ &= \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} \xi_{\mathbf{j}}. \end{aligned}$$

□

Algorithm 4 Optimization of $\widehat{U}(f, D)$.

Input: data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, C , $\mathbf{K}$, ϵ

$\mathcal{W} \leftarrow \emptyset$

repeat

$\widehat{\mathbf{w}} \leftarrow$ optimize $\mathbf{w}$ over working set $\mathcal{W}$

$\widehat{\xi} \leftarrow$ corresponding ξ for $\widehat{\mathbf{w}}$

$(\mathbf{c}, \mathbf{t}) \leftarrow \arg \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1,\dots,r-1\}^s}} \left\{ \frac{1}{s} \sum_{j \in \mathcal{S}} c_j - \widehat{\mathbf{w}} \cdot \frac{1}{s} \sum_{j \in \mathcal{S}} c_j \delta(\mathbf{j}, t_j) \right\}$

$\xi_{\max} \leftarrow \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1,\dots,r-1\}^s}} \left\{ \frac{1}{s} \sum_{j \in \mathcal{S}} c_j - \widehat{\mathbf{w}} \cdot \frac{1}{s} \sum_{j \in \mathcal{S}} c_j \delta(\mathbf{j}, t_j) \right\}$

$\mathcal{W} \leftarrow \mathcal{W} \cup \{(\mathbf{c}, \mathbf{t})\}$

until $\xi_{\max} \leq \widehat{\xi} + \epsilon$

Output: $\widehat{\mathbf{w}}$

Optimization problem (5.4) has even more constraints than the previous one. Nevertheless, it can be optimized much more efficiently with a cutting plane algorithm (Kelley 1960). Algorithm 4 gives an overview of this iterative procedure. Of crucial importance is that not all constraints are active in the quadratic program. Normally, for only one $(\mathbf{c}, \mathbf{t})$ the inequality constraint will become an equality constraint. To this end, a working set $\mathcal{W}$ of active constraints is introduced, so $\mathcal{W} \subseteq \{0, 1\}^s \times \{1, \dots, r - 1\}^s$.

Given a data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ with corresponding Gram matrix $\mathbf{K}$, regularization parameter C and stopping criterion ϵ , the algorithm starts with $\mathcal{W} = \emptyset$. In each iteration, first the optimal solution $\widehat{\mathbf{w}}$ and corresponding slack $\widehat{\xi}$ are computed. Then, the constraint with maximum violation is added to the working set (this is the constraint for which the inequality becomes an equality with regard to the current solution $\widehat{\mathbf{w}}$). In the next iteration, the objective function is again optimized over the adjusted working set of constraints. Consequently, the constraints not belonging to $\mathcal{W}$ are not necessarily observed. The algorithm repeatedly adds new constraints to the working set until no constraint is violated more than the stopping criterion ϵ . It has been proven before that such a cutting plane algorithm stops after adding a constant number of constraints, independent of the data set size. Details about the time complexity are given in Section 5.2.5. The total running time therefore strongly depends on the time required to find the maximum violating constraint. The following section addresses this topic.

5.2.3 Finding the maximum violating constraint

In this section we will present an algorithm for computing the maximum violating constraint in each iteration of Algorithm 2, i.e.

$$(\mathbf{c}, \mathbf{t}) = \arg \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1, \dots, r-1\}^s}} \left\{ \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \mathbf{w} \cdot \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right\}.$$

Intuitively, this might seem a difficult job, but we will show that this constraint can be found efficiently for any $\mathbf{w}$. First of all, for any couple $(\mathbf{c}, \mathbf{t}) \in \{0, 1\}^s \times \{1, \dots, r-1\}^s$, a corresponding $(\mathbf{c}^+, \mathbf{c}^-) \in \mathbb{N}^n \times \mathbb{N}^n$ can be found such that:

$$\begin{aligned} \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \mathbf{w} \cdot \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) = \\ \frac{1}{2s} \sum_{i=1}^n (c_i^+ + c_i^-) - \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) \mathbf{w} \cdot \phi(\mathbf{x}_i)). \end{aligned}$$

Thus, computing the maximum violating constraint already reduces to finding two vectors of size n instead of vectors of size s . This will be denoted $(\mathbf{c}, \mathbf{t}) \hookrightarrow (\mathbf{c}^+, \mathbf{c}^-)$. The following lemma attaches a graph-based interpretation to $(\mathbf{c}^+, \mathbf{c}^-)$.

Lemma 5.5. *Let $f : \mathcal{X} \rightarrow \mathbb{R}$ with $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})$ be a function that imposes a ranking on an i.i.d. data set D , let $G_{\Delta}(f, D) = (V, E)$ be the corresponding Δ -graph with Ξ the set of paths connecting source and sink, let $v = (\mathbf{x}_i, y_i) \in V_k, k \in \{1, \dots, r\}$ and let*

$$(\mathbf{c}^+, \mathbf{c}^-) \hookrightarrow (\mathbf{c}, \mathbf{t}) = \arg \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1, \dots, r-1\}^s}} \left\{ \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \mathbf{w} \cdot \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right\},$$

then:

1. c_i^+ corresponds to the number of distinct paths passing through v for which an edge e arriving in v has minimal weight among all edges along the respective path and $w(e) < 1$, i.e.

$$c_i^+ = |\{\varpi \in \Xi \mid \exists v_j \in V_{k-1} : (v_j, v_i) = \arg \min_{e \in \varpi \wedge w(e) < 1} w(e)\}|.$$

2. c_i^- corresponds to the number of distinct paths passing through v for which an edge e departing from v has minimal weight among all edges along the respective path and $w(e) < 1$, i.e.

$$c_i^- = |\{\varpi \in \Xi \mid \exists v_j \in V_{k+1} : (v_i, v_j) = \arg \min_{e \in \varpi \wedge w(e) < 1} w(e)\}|.$$

Proof. Similarly to the proof of Theorem 5.4, we find:

$$(c_j, t_j) = \arg \max_{\substack{c_j \in \{0,1\} \\ t_j \in \{1, \dots, r-1\}}} \left\{ \frac{1}{s} c_j - \frac{1}{s} c_j (\mathbf{w} \cdot \delta(\mathbf{j}, t_j)) \right\}.$$

The vector $\mathbf{t}$ can be eliminated as follows:

$$\begin{aligned} c_j &= \arg \max_{c_j \in \{0,1\}} \left\{ \frac{1}{s} c_j - \frac{1}{s} c_j \min_{k \in \{1, \dots, r-1\}} \{ \mathbf{w} \cdot \delta(\mathbf{j}, k) \} \right\} \\ &= \begin{cases} 0, & \text{if } \min_{k \in \{1, \dots, r-1\}} \{ \mathbf{w} \cdot \delta(\mathbf{j}, k) \} \geq 1, \\ 1, & \text{if } \min_{k \in \{1, \dots, r-1\}} \{ \mathbf{w} \cdot \delta(\mathbf{j}, k) \} < 1. \end{cases} \end{aligned}$$

With every r -tuple $((\mathbf{x}_{j_1}, y_{j_1}), (\mathbf{x}_{j_2}, y_{j_2}), \dots, (\mathbf{x}_{j_r}, y_{j_r}))$ a path ϖ in the Δ -graph can be associated unambiguously and

$$c_j = c_\varpi = \begin{cases} 0, & \text{if } \min_{e \in \varpi} w(e) \geq 1, \\ 1, & \text{if } \min_{e \in \varpi} w(e) < 1. \end{cases}$$

Let us now consider the following simple algorithm:

```

for  $\varpi \in \Xi$  do
   $(v_i, v_j) = \arg \min_{e \in \varpi} w(e)$ 
  if  $w((v_i, v_j)) < 1$  then
     $c_i^- \leftarrow c_i^- + 1$ 
     $c_j^+ \leftarrow c_j^+ + 1$ 
  end if
end for

```

The algorithm glances through all paths in Ξ . For each path, it determines the edge $e = (v_i, v_j)$ with minimal weight along the path in which the vertices are respectively associated with the i -th and j -th element in the data set: $v_i = (\mathbf{x}_i, y_i) \in V_k, v_j = (\mathbf{x}_j, y_j) \in V_{k+1}$ with $k \in \{1, \dots, r-1\}$. It is straightforward to see that the following equalities hold upon termination of the algorithm:

$$\frac{1}{s} \sum_{j \in S} c_j = \frac{1}{s} \sum_{\varpi \in \Xi} c_\varpi = \frac{1}{2s} \sum_{i=1}^n (c_i^+ + c_i^-)$$

and

$$\begin{aligned} \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \min_{k \in \{1, \dots, r-1\}} \{ \mathbf{w} \cdot \delta(\mathbf{j}, k) \} &= \frac{1}{s} \sum_{\varpi \in \Xi} c_{\varpi} \min_{e \in \varpi} w(e) \\ &= \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) \mathbf{w} \cdot \phi(\mathbf{x}_i)) . \end{aligned}$$

□

The algorithm described in the proof in general takes $\mathcal{O}(n^r)$ time to find the maximum violating constraint. Using the graph, we will now describe a more intelligent way to identify this constraint. If we want to compute $(\mathbf{c}^+, \mathbf{c}^-)$ on the Δ -graph, then at least all edges with weight smaller than 1 need to be investigated. So, a lower bound for the complexity of an algorithm will be $\mathcal{O}(n^2)$. The following procedure will correctly identify $(\mathbf{c}^+, \mathbf{c}^-)$:

1. Initialize $c_i^+ = c_i^- = 0$ for $i = 1, \dots, n$.
2. Find the edge $(v_i, v_j) \in E$ with minimal weight.
3. Increase c_i^- and c_j^+ with the number of paths passing through (v_i, v_j) .
4. Construct a subgraph of $G_{\Delta}(f, D) = (V, E)$ by removing the edge (v_i, v_j) from the graph, i.e. $E = E \setminus \{(v_i, v_j)\}$, and repeat steps 2,3 and 4 until no edges with weight smaller than 1 remain in E .

By removing the edge from the graph, it is guaranteed that no path will be counted twice. If E is sorted according to the weight function, then step 2 simply consists of retrieving its first element. For step 3 and step 4, one needs a compact representation of the data for which both the number of paths passing through the edge (v_i, v_j) can be found quickly and the edge can be removed from the graph in an efficient way. Remark also that the edge (v_i, v_j) under study has minimal weight of all paths passing through it due to the sorting of E . To derive the algorithm in more detail, we first introduce an additional concept.

Definition 5.6. Let $f : \mathcal{X} \rightarrow \mathbb{R}$ with $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})$ be a function that imposes a ranking on an i.i.d. data set D , let $G_{\Delta}(f, D) = (V, E)$ be the corresponding Δ -graph and let $G_{\Delta}(f, D)^* = (V, E^*)$ be a subgraph of $G_{\Delta}(f, D) = (V, E)$, i.e. $E^* \subseteq E$, then we call the ordered set $V = (v_s, v_1, \dots, v_n, v_e)$ consistent with E^* if

1. $\forall (v_i, v_j) \in E : (v_i, v_j) \in E^* \Leftrightarrow i < j$,
2. $\forall k \in \{1, \dots, r\}, \forall v_i = (\mathbf{x}_i, y_i), v_j = (\mathbf{x}_j, y_j) \in V_k : f(\mathbf{x}_i) < f(\mathbf{x}_j) \Leftrightarrow i < j$.

Lemma 5.7. *Let $G_\Delta^*(f, D) = (V, E^*)$ be a subgraph of $G_\Delta(f, D) = (V, E)$, i.e. $E^* \subseteq E$, let V be consistent with E^* , and let $e = (v_a, v_b)$ be the edge in E^* with minimal weight, then the number of paths passing through e can be found in $\mathcal{O}(n)$ time.*

Proof. Let us consider the function Λ , which is recursively defined as follows:

$$\Lambda_k(i) = \begin{cases} \Lambda_{k-1}(i-1), & \text{if } y_i = \bar{y}_k \wedge (i = a \vee i = b), \\ \Lambda_{k-1}(i-1) + \Lambda_k(i-1), & \text{if } y_i = \bar{y}_k \wedge (y_i \neq y_a \wedge y_i \neq y_b), \\ \Lambda_k(i-1), & \text{else,} \end{cases}$$

for $k = 1, \dots, r$ and $i = 1, \dots, n$ with starting values

$$\Lambda_k(0) = \begin{cases} 0, & \text{if } k \in \{1, \dots, r\}, \\ 1, & \text{if } k = 0. \end{cases}$$

$\Lambda_r(n)$ can be computed in a single iteration over the data set. We show that it corresponds to the number of distinct paths passing through (v_a, v_b) , when (v_a, v_b) has the minimal weight of all edges remaining in the graph.

First of all, $\Lambda_1(i)$ represents the number of paths connecting the source with a vertex that is part of the first i vertices in V and belongs to layer V_1 . Let the edge (v_a, v_b) connect a node of the layer V_k and a node of layer V_{k+1} . Then, for all nodes v_i belonging to a layer V_l with $l < k$, $\Lambda_l(i)$ represents the number of distinct paths from the source to an edge that is part of the first i vertices in V and belongs to layer V_l . The number of paths arriving in an edge part of the first i elements of V and belonging to layer V_l , recursively consists of the number of paths arriving in an edge part of the first $i-1$ elements of V and belonging to layer V_l plus the number of paths arriving in an edge connected to v_i and thus part of layer V_{l-1} . This last number equals $\Lambda_{l-1}(i-1)$, because V is consistent with E^* and, hence, all edges connected to v_i are found in the first $i-1$ elements of V . Summarizing, the number of paths from the source to an edge in the first i elements of V and belonging to a layer V_l is given by:

$$\Lambda_l(i) = \Lambda_{l-1}(i-1) + \Lambda_l(i-1),$$

when $l < k$.

Subsequently, in layer V_k only the number of paths arriving in the edge v_a must be considered. This equals

$$\Lambda_k(a) = \Lambda_{k-1}(a - 1).$$

The same reasoning holds for the edge v_b of the next layer. The number of paths arriving in v_b equals:

$$\Lambda_{k+1}(b) = \Lambda_k(b - 1) = \Lambda_k(a),$$

since (v_a, v_b) has minimal weight and, hence, no edges of layer V_k are found in V between v_a and v_b .

For layers V_l with $l > k + 1$, the same reasoning applies as for layers V_l with $l < k$. The number of paths arriving in a node part of the first i elements of V and belonging to layer V_l again equals:

$$\Lambda_l(i) = \Lambda_{l-1}(i - 1) + \Lambda_l(i - 1).$$

At the end, $\Lambda_r(n)$ represents the number of distinct paths through (v_a, v_b) arriving in any node of layer V_r for which (v_a, v_b) has the minimal weight of all edges along the path or, equivalently, the number of paths through (v_a, v_b) arriving in v_e for which (v_a, v_b) has minimal weight of all edges along the path. $\square$

Lemma 5.8. *Let $G_\Delta(f, D)^* = (V, E^*)$ be a subgraph of $G_\Delta(f, D) = (V, E)$, i.e. $E^* \subseteq E$, let V be consistent with E^* , let $e = (v_a, v_b)$ be the edge in E^* with minimal weight and let $E^{**} = E^* \setminus \{e\}$, then V can be made consistent with E^{**} in $\mathcal{O}(n)$ time.*

Proof. Let us assume that V is implemented as a linked list and that V is consistent with E^* . To make V consistent with E^{**} , the order of v_a and v_b must be reversed in the linked list. Unfortunately, this might cause other vertex couples to be inverted too, when other vertices are located between v_a and v_b . We need an algorithm that switches the order of v_a and v_b , while preserving the ordering of vertex couples in E (not E^* or E^{**}). The following simple procedure solves this problem:

```

 $V^{**} = \emptyset$ 
for  $i = a, \dots, b$  do
  if  $y_i < y_b$  then
     $V \leftarrow \text{remove } (x_i, y_i) \text{ from } V$ 
     $V^{**} \leftarrow \text{append } (x_i, y_i) \text{ to } V^{**}$ 

```

```

    end if
  end for
   $V \leftarrow \text{insert } V^{**} \text{ in } V \text{ after } (\mathbf{x}_b, y_b)$ 

```

The procedure examines all nodes located between v_a and v_b . If we would simply switch the elements v_a and v_b , then all nodes located between these two nodes and belonging to a lower layer than v_b would cause V to become inconsistent again. Therefore, these nodes must be kept farther in the list than v_a . Together with v_a , they are transferred to the list V^{**} . When v_b is reached, the list V^{**} is inserted in the original list after v_b . If a linked list is used as data structure, this takes only constant time. Nodes belonging to a higher layer than v_a on the other hand, are not removed from V and appended to V^{**} . They would cause conflicts with v_a , making V inconsistent with E^{**} . $\square$

These two lemmas form the main tools for the four-step procedure outlined above. Algorithm 5 gives a more detailed overview of the steps needed to find the maximum violating constraint. To this end, the algorithm constructs the set of edges of the graph and the set of nodes is simply retrieved from the data set. The weights are defined by f and E is sorted according to the weight in order to find the edge with minimal weight efficiently at any step. At the start, V must be consistent with E and this is achieved by ordering V as follows:

$$V = (v_s, V_1, \dots, V_r, v_e),$$

in which the layers V_k are sorted according to increasing f . It is easy to verify that in this case both conditions in Def. 5.6 are fulfilled. After initialization, the algorithm iterates over all edges having a weight lower than one and removes investigated edges from E . By computing the recursive function in Lemma 5.7, the number of paths passing through the edge which has currently minimal weight in E is found efficiently and, consecutively, this edge is removed and V is made consistent with the reduced set of edges by applying the trick expounded in Lemma 5.8. Lemma 5.5 ensures that the algorithm correctly finds the maximum violating constraint for any f .

5.2.4 Duality aspects

Until now, only finite-dimensional feature mappings ϕ were considered. In this section it is demonstrated that all parts of the algorithm can be written in

Algorithm 5 Computation of the maximum violating constraint.

Input: data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, f
Construct the set E and the weight function $w : E \rightarrow \mathbb{R}$ of the graph $G_\Delta(f, D)$
Sort E according to ascending w
 $\forall k \in \{1, \dots, r\}$: construct list $V_k := \{(\mathbf{x}_i, y_i) \in D \mid y_i = \bar{y}_k\}$
 $\forall k \in \{1, \dots, r\}$: sort V_k according to ascending f
 $V \leftarrow$ concatenate $V_1, \dots, V_r$
 $e \leftarrow$ pop first element $((\mathbf{x}_a, y_a), (\mathbf{x}_b, y_b))$ from E
while $w(e) < 1$ **do**
 $\forall k \in \{1, \dots, r\} : \Lambda_k \leftarrow 0$
 $V^{**} \leftarrow \emptyset$
 $\Lambda_0 \leftarrow 1$
 for $(\mathbf{x}_i, y_i) \in V$ **do**
 if $y_i = \bar{y}_k \wedge k > 0$ **then**
 if $i = a \vee i = b$ **then**
 $\Lambda_k \leftarrow \Lambda_{k-1}$
 end if
 if $y_i \neq y_a \wedge y_i \neq y_b$ **then**
 $\Lambda_k \leftarrow \Lambda_k + \Lambda_{k-1}$
 end if
 end if
 if $i \geq a \wedge i < b \wedge y_i < y_b$ **then**
 $V \leftarrow$ remove $(\mathbf{x}_i, y_i)$ from V
 $V^{**} \leftarrow$ append $(\mathbf{x}_i, y_i)$ to V^{**}
 end if
 end for
 $c_a^- \leftarrow c_a^- + \Lambda_r$
 $c_b^+ \leftarrow c_b^+ + \Lambda_r$
 $V \leftarrow$ insert V^{**} in V after $(\mathbf{x}_b, y_b)$
 $e \leftarrow$ pop first element $((\mathbf{x}_a, y_a), (\mathbf{x}_b, y_b))$ from E
end while
Output: $\mathbf{c}^+, \mathbf{c}^-$

terms of kernels without knowledge of ϕ . Let us recapitulate the quadratic program that has to be solved in each iteration of Algorithm 4.

$$\begin{aligned} & \min_{\mathbf{w}, \xi} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \\ & \text{subject to} \quad \begin{cases} \mathbf{w} \cdot \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) \phi(\mathbf{x}_i)) \geq \frac{1}{2s} \sum_{i=1}^n (c_i^+ + c_i^-) - \xi, \\ \xi \geq 0, \\ (\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}. \end{cases} \end{aligned} \quad (5.7)$$

The primal Lagrangian is given by:

$$\begin{aligned} L_P = & \frac{1}{2} \|\mathbf{w}\|^2 + C\xi - \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \left(\mathbf{w} \cdot \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) \phi(\mathbf{x}_i)) \right. \\ & \left. - \frac{1}{2s} \sum_{i=1}^n (c_i^+ + c_i^-) + \xi \right) - \beta \xi. \end{aligned} \quad (5.8)$$

The KKT-conditions imply:

$$\mathbf{w} = \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) \phi(\mathbf{x}_i)), \quad (5.9)$$

$$\beta = C - \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)}. \quad (5.10)$$

By substituting (5.9) and (5.10) back into (5.8) we get the dual optimization problem:

$$\begin{aligned} & \max_{\alpha_{(\mathbf{c}^+, \mathbf{c}^-)}} \quad \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \frac{1}{2s} \sum_{i=1}^n (c_i^+ + c_i^-) \\ & \quad - \frac{1}{2} \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \sum_{(\mathbf{d}^+, \mathbf{d}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \alpha_{(\mathbf{d}^+, \mathbf{d}^-)} Q((\mathbf{c}^+, \mathbf{c}^-), (\mathbf{d}^+, \mathbf{d}^-)) \\ & \quad \text{subject to} \quad 0 \leq \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \leq C, \end{aligned}$$

with Q a kernel defined over $\mathcal{W}^2$. The following theorem reveals that specifying a positive definite kernel K defined over $\mathcal{X}^2$ is sufficient to optimize a convex objective function.

Proposition 5.9. Let $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ be a positive definite kernel with Gram matrix $\mathbf{K}$ on $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathcal{X}$, i.e. $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, then the function $Q : \mathcal{W}^2 \rightarrow \mathbb{R}$ defined by

$$Q((\mathbf{c}^+, \mathbf{c}^-), (\mathbf{d}^+, \mathbf{d}^-)) = \frac{(\mathbf{c}^+ - \mathbf{c}^-)^T \mathbf{K} (\mathbf{d}^+ - \mathbf{d}^-)}{s^2},$$

is a positive definite kernel over $\mathcal{W}^2$.

Proof. The kernel K is positive definite if for any $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ the corresponding Gram matrix $\mathbf{K}$ is positive definite (Schölkopf & Smola 2002). By definition, the real matrix $\mathbf{K}$ is positive definite if for any $\eta \in \mathbb{R}^n$

$$\eta^T \mathbf{K} \eta > 0. \quad (5.11)$$

Equivalently, the function Q is positive definite if any $\mathcal{W}$ gives rise to a positive definite Gram matrix $\mathbf{Q}$ with

$$\mathbf{Q}_{((\mathbf{c}^+, \mathbf{c}^-), (\mathbf{d}^+, \mathbf{d}^-))} := Q((\mathbf{c}^+, \mathbf{c}^-), (\mathbf{d}^+, \mathbf{d}^-)), \quad \forall (\mathbf{c}^+, \mathbf{c}^-), (\mathbf{d}^+, \mathbf{d}^-) \in \mathcal{W}.$$

Let us define the $|\mathcal{W}| \times n$ matrix $\mathbf{M}$ as follows:

$$\mathbf{M}_{(\mathbf{c}^+, \mathbf{c}^-), i} := \frac{c_i^+ - c_i^-}{s}, \quad \forall (\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}, \forall i \in \{1, \dots, n\},$$

then, for any $\alpha \in \mathbb{R}^{|\mathcal{W}|}$, we find

$$\alpha^T \mathbf{Q} \alpha = \alpha^T \mathbf{M}^T \mathbf{K} \mathbf{M} \alpha = (\mathbf{M} \alpha)^T \mathbf{K} \mathbf{M} \alpha.$$

If (5.11) holds, then this quantity is always greater than zero since $\mathbf{M} \alpha \in \mathbb{R}^n$. $\square$

Both Algorithms 4 and 5 are implemented in terms of the dual representation. The dual in each iteration is solved with the quadratic solver of the MOSEK-package¹. This solver implements an interior point algorithm and was for example formerly used by Lanckriet *et al.* (2004), Heller & Schnörr (2006) in the context of machine learning. The prediction rule for a new object $\mathbf{x} \in \mathcal{X}$ can be written in terms of kernels by again applying (5.9), i.e.

$$f(\mathbf{x}) = \sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)} \frac{1}{s} \sum_{i=1}^n ((c_i^+ - c_i^-) K(\mathbf{x}, \mathbf{x}_i)).$$

¹The MOSEK-package can be freely downloaded for non-commercial use from www.mosek.com

This prediction rule will also serve as input for computing the maximum violating constraint during training. In this way, all parts of the optimization procedure are carried out without explicitly processing $\mathbf{w}$ and ϕ .

Sparsity in the obtained solution can be ensured in two ways:

1. For constraints that are added to $\mathcal{W}$, many elements of the n -dimensional vectors $\mathbf{c}^+$ and $\mathbf{c}^-$ will manifest zeroes. Zeroes are encountered when no ingoing and outgoing edges of the corresponding node in the graph have minimal weight for any path from source to sink. This happens frequently.
2. Once a new constraint $(\mathbf{c}^+, \mathbf{c}^-)$ is added to $\mathcal{W}$ and the dual is re-optimized, the corresponding Lagrange multiplier $\alpha_{(\mathbf{c}^+, \mathbf{c}^-)}$ will typically decrease in further iterations. Because the upper bound C on

$$\sum_{(\mathbf{c}^+, \mathbf{c}^-) \in \mathcal{W}} \alpha_{(\mathbf{c}^+, \mathbf{c}^-)}$$

will normally be reached during any iteration in the cutting plane algorithm, at least one Lagrange multiplier has to decrease when a new constraint is added to $\mathcal{W}$. Many Lagrange multipliers tend to become zero again in practice, few iterations after adding them to the quadratic program.

5.2.5 Complexity of the algorithm

Lemma 5.10. *For any $C > 0$, $\epsilon > 0$, positive definite kernel $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ and training data set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, Algorithm 4 stops after adding at most*

$$\min \left\{ \frac{2}{\epsilon}, \frac{16C \max_i \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}}{\epsilon^2} \right\}$$

constraints to the working set $\mathcal{W}$.

Proof. The proof is primarily based based on (Tsochantaridis *et al.* 2005) (see Proposition 17 and Theorem 18 therein). Because our algorithm is a special type of the algorithms mentioned there, the improvement of the objective of the dual of optimization problem (5.4) is in each iteration lower bounded by

$$\frac{\epsilon^2}{8R},$$

with R the maximal norm of the feature representation of a structured output (here a layered graph) over all possible outputs. R is in our case given by:

$$\begin{aligned}
 R &= \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1, \dots, r-1\}^s}} \left\| \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right\| \\
 &= \max_{\mathbf{c} \in \{0,1\}^s} \left\| \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \max_{k \in \{1, \dots, r-1\}} (\phi(\mathbf{x}_{j_{k+1}}) - \phi(\mathbf{x}_{j_k})) \right\| \\
 &\leq \frac{2}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \max_i \|\phi(\mathbf{x}_i)\| \\
 &\leq 2 \max_i \|\phi(\mathbf{x}_i)\| \\
 &= 2 \max_i \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}.
 \end{aligned}$$

We thus have a lower bound on the increase of the dual objective in each iteration. On the other hand, the solution of the dual optimization problem is upper bounded by C , which is a feasible value of the primal objective (when $\mathbf{w} = 0$ and $\xi = 1$). C will be reached after adding at most

$$\min \left\{ \frac{2}{\epsilon}, \frac{16C \max_i \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}}{\epsilon^2} \right\}$$

constraints to the quadratic program. □

Lemma 5.11. *For any choice of $\mathbf{w}$, the maximum violating constraint in optimization problem (5.4), i.e.*

$$(\mathbf{c}, \mathbf{t}) = \arg \max_{\substack{\mathbf{c} \in \{0,1\}^s \\ \mathbf{t} \in \{1, \dots, r-1\}^s}} \left\{ \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} - \mathbf{w} \cdot \frac{1}{s} \sum_{\mathbf{j} \in \mathcal{S}} c_{\mathbf{j}} \delta(\mathbf{j}, t_{\mathbf{j}}) \right\}$$

can be computed in $\mathcal{O}(n^3)$.

Proof. First, the set of edges E is constructed together with the weights. In general, E contains $\mathcal{O}(n^2)$ edges and sorting E will take $\mathcal{O}(n^2 \log(n^2))$ time consequently. The set V contains $n + 2$ elements and the ordering given in (5.7) is obtained in $\mathcal{O}(n \log(n))$ time.

Then, the algorithm investigates all edges in E with weight lower than one, which takes at most $\mathcal{O}(n^2)$ time. For every edge e with $w(e) < 1$, the number of paths for which the edge has minimal weight must be computed and V must be manipulated in order to become consistent with the reduced $E \setminus \{e\}$. Both actions take at most $\mathcal{O}(n)$ time, which leads to an overall worst case time complexity of $\mathcal{O}(n^3)$ for finding the maximum violating constraint. □

Combining this result with complexity bounds mentioned in the literature (Tsochantaridis *et al.* 2005), we can quantify the complexity of Algorithm 4.

Proposition 5.12. *For any $C > 0$, $\epsilon > 0$, positive definite kernel $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ and training sample $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, Algorithm 4 has a $\mathcal{O}(n^3)$ time complexity.*

Proof. The algorithm terminates after adding a constant number of constraints. In each iteration, finding the maximum violating constraint occupies the majority of the time. Solving the quadratic program takes $\mathcal{O}(|\mathcal{W}|^2)$ time and normally $|\mathcal{W}| \ll n$. $\square$

5.3 Experiments

Three types of experiments are reported in this section to demonstrate the behavior of the algorithm. In a first type of experiments toy problems are considered to illustrate some basic properties with respect to generalization performance. Secondly, the same methodology is adopted to evaluate the algorithm on benchmark data sets. In the last type of experiments, the scaling properties of the algorithm are studied.

For the evaluation of generalization performance on synthetic and benchmark problems, we were primarily interested in comparing our algorithm, which aims to rank sequences of objects correctly, versus the conventional approach in which correctly ranked pairs of objects are considered. For that reason, it is logical to compare our approach with the algorithm of Joachims (2006), which is an enhanced optimization procedure for the pairwise loss introduced by Herbrich *et al.* (2000). Beside the difference in loss function, the optimization algorithm with cutting planes is the same as in our algorithm, so differences in performance are only caused by the loss function. As discussed in the introduction, a distinction can be made between authors presenting layered ranking models and authors presenting ordinal regression models. Many of these authors do not provide publicly available implementations or their implementations were impracticable for this analysis. We restricted the comparison to the pairwise approach since we were not really interested in comparing with ordinal regression models. We only wanted to find out the effect of minimizing a sequence-based loss.

5.3.1 Toy problems

In this experiment we considered toy problems for two reasons. First of all, we wanted to gain more insight into the behavior of optimizing VUS versus optimizing the C-index by knowing the data distribution. Secondly, with synthetic data an infinite number of data objects can be sampled (in theory), and, hence, statistically significant differences in performance between various algorithms can be detected more easily.

To this end, toy problems with two-dimensional feature vectors and four categories were constructed. We assumed that objects of each category were grouped as a cluster and for all clusters a bivariate Gaussian distribution was chosen. Among the four categories the mean of the bivariate Gaussian clusters varied. In increasing order according to the category labels, respectively the points (10,10), (20,10), (20,20), (30,20) were taken as means. An equal standard deviation ($\sigma = 3$) for all four clusters and uncorrelated features ($\rho = 0$) were assumed, resulting in a relatively good separability for all four clusters. The sample size was fixed to 400. The only parameter that varied was the number of objects we sampled from each category.

In the beginning of this chapter, we argued that the pairwise error evaluates the ranking model by pairwise comparisons of categories. One of the goals of this experiment was to illustrate how this effect can harm model inference. Let us for the toy problem consider a ranking that coincides with the x-axis, which would imply that the second feature has no effect in the model. Such a model yields a relatively high C-index (or low pairwise error), although it is inappropriate. Conversely, the y-dimension provides meaningful information with respect to the ranking. The same model is very unlikely to be fitted by our algorithm because it evaluates the ranking in a global way.

In the experiment a Gaussian RBF-kernel was chosen as similarity function, because the optimal ranking function would definitely be non-linear. To tune the width γ of the kernel and the regularization parameter C , 20 training and validation sets were generated with 100 objects of each category. One of the generated data sets is shown in Figure 5.1. Then, a two-dimensional grid search was performed 20 times for C and γ . For the pairwise approach, we used the pairwise error as evaluation criterion and for our approach VUS. To select the best values for C and γ , we averaged over the optimal values found during the 20 individual runs. After model selection, we let vary the number of objects sampled from each category, while keeping the total sample size fixed to 400. In this way, we considered six different sampling schemes, ranging from the balanced setting with 100 objects of each category to very

Figure 5.1: One of the data sets generated for the toy problem. Each category contains 100 instances in this case.

unbalanced settings with only 8 objects of a particular category. All examined sampling schemes are given in the first column of Table 5.2. One can assume that in all sampling schemes the optimal values for C and γ remain unaffected, because neither the desired complexity of the model nor the similarity between data points changes by sampling more objects of a particular category, on condition that the total sample size is kept fixed.

To compare the generalization performance of both models, we take into account the fraction of correctly ranked r -tuples (VUS) as well as the fraction of correctly ranked pairs (C-index). In order to detect statistically significant differences, we repeated training and testing on a large number of independently sampled data sets (in total 50 training sets and 50 test sets). The independence assumption allows to employ a simple paired Wilcoxon rank-sum test, as recommended by Demsar (2006) for comparing two classifiers over multiple data sets. When the independence assumption would be violated, then more complicated experimental designs and tests would be required (Dietterich 1998).

Table 5.2 gives an overview of the obtained results together with the number of times each algorithm won and the p-values of the signed Wilcoxon rank-sum test. In all settings a significantly better VUS was measured for the r -tuple approach, which supports the claim that optimizing correctly ranked

r -tuples leads to a better generalization performance for VUS. The results were totally different for the C-index. Here the obtained values were very comparable. Only in one setting a significantly better C-index was measured. We conclude that optimizing VUS harms less the C-index than conversely.

Data set	Method	$\hat{U}(f, D)$	Wins	p-value	C-index	Wins	p-value
100-100-100-100	VUS-opt.	0.914 ± 0.024	40	<0.001	0.985 ± 0.004	40	<0.001
	Pairs-opt.	0.910 ± 0.023	10		0.985 ± 0.004	10	
160-160-40-40	VUS-opt.	0.913 ± 0.028	43	<0.001	0.984 ± 0.005	30	0.954
	Pairs-opt.	0.894 ± 0.037	7		0.985 ± 0.005	20	
160-120-80-40	VUS-opt.	0.901 ± 0.039	41	<0.001	0.982 ± 0.007	25	0.858
	Pairs-opt.	0.892 ± 0.046	9		0.982 ± 0.008	25	
40-160-160-40	VUS-opt.	0.912 ± 0.029	40	<0.001	0.975 ± 0.009	12	<0.001
	Pairs-opt.	0.900 ± 0.032	10		0.979 ± 0.007	38	
320-48-24-8	VUS-opt.	0.859 ± 0.071	48	<0.001	0.984 ± 0.006	23	0.472
	Pairs-opt.	0.686 ± 0.144	2		0.985 ± 0.007	27	
24-320-8-48	VUS-opt.	0.896 ± 0.050	36	0.001	0.989 ± 0.007	25	0.434
	Pairs-opt.	0.881 ± 0.051	14		0.990 ± 0.006	25	

Table 5.2: An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on a series of toy problems of size 400 with different category distributions. P-values originate from paired Wilcoxon rank-sum tests over 50 runs of both algorithms.

When we looked at the effect of imbalance, then trends were less remarkable. Unsurprisingly, the results for both methods were the most consistent in the balanced setting, but in the unbalanced settings trade-offs were less prominent than expected beforehand. In the fourth setting, for which 80% of the data was sampled from the two inner clusters, a significant trade-off is visible. Apparently, the pairwise approach here clearly overfits on the inner clusters, which leads to a good generalization performance in terms of the C-index, but a bad one in terms of VUS.

5.3.2 Benchmark data sets

In this experiment we tested both algorithms on several benchmark data sets. Since no collections of ordinal labeled data are publicly available and because we wanted to investigate both balanced and unbalanced data sets, we converted existing metric regression data sets to an ordinal scale. To this end, we used a repository² containing various data sets from different domains. In order to adopt a similar setup as for the toy problems, we selected all real-world data sets having at least 8000 cases from this repository³. The repository also contained artificial data sets, which were not used further. This yielded in total seven data sets to be transformed to an ordinal scale. The transformation was done in two ways. In the balanced setting, each data set was subdivided into five equal-frequency bins after sorting the data according to the continuous labels. In the unbalanced setting, also five categories were considered, but here respectively 40%, 30%, 20%, 5% and 5% of the data was placed in a bin.

After transformation to the ordinal scale, all data sets were randomly subdivided into 20 subsets (10 training sets and 10 test sets). In this way, the independence assumption for applying a paired Wilcoxon rank-sum test was not violated, so the same setup as in the previous experiments could be utilized. The optimal values of C and γ were also tuned as before, but now only one of the training sets was used during model selection due to the increased complexity of ranking r -tuples.

²The data sets can be downloaded from <http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html>

³For the *Bank*, *Computer Activity* and *Census* domains two version of the data were available. To be consistent, the highest dimensional version was chosen in all three cases. The domains *Delta Ailetrons*, *Elevators* and *Delta Elevators* were all related to F16 aircrafts. We only used *Delta Elevators*.

Tables 5.3 and 5.4 respectively summarize the results obtained for the balanced and the unbalanced setting. The paired Wilcoxon rank-sum test is reported for each domain individually. Also the p-value of a global test on all 70 test sets is given for completeness, although we realize that assumptions for applying a paired Wilcoxon rank-sum test might be slightly violated in that case. First of all, no sharp discrepancies are observed between the balanced and the unbalanced case. For the majority of the analyzed data sets, the winning algorithm in the balanced case also outperforms its opponent in the unbalanced case. Only for the *Computer Activity* domain conflicting results are observed (in terms of the C-index) between the balanced and unbalanced setting. For some domains, the difference in performance between both algorithms was statistically significant in one case, but not in the other case. Nonetheless, the results obtained for the balanced and unbalanced settings are very similar. In both settings, the r -tuple approach wins from the pairwise approach on the majority of the data sets with respect to VUS⁴ as performance measure. For the C-index, the r -tuple approach is only significantly better in the balanced case. In the unbalanced case, a small trade-off turns up between optimizing correctly ranked r -tuples on the one hand and correctly ranked pairs on the other hand. Similar trends were observed in the experiments on toy problems. We conclude that optimizing correctly ranked r -tuples in general improves the generalization performance on the majority of the analyzed problems.

⁴The observed values for VUS might seem extremely low at first sight. However, VUS follows a skew distribution and for a random classifier, i.e. a classifier which randomly picks one of the $n!$ permutations of the data set as ranking, the expected value for VUS is $\frac{1}{r!}$.

Data set	Method	$\bar{U}(f, D)$	Wins	p-value	C-index	Wins	p-value
Pole Telecomm	VUS-opt.	0.134 ± 0.017	5	0.770	0.819 ± 0.011	4	0.375
	Pairs-opt.	0.135 ± 0.017	5		0.820 ± 0.011	6	
Bank	VUS-opt.	0.081 ± 0.018	10	0.002	0.744 ± 0.027	10	0.002
	Pairs-opt.	0.060 ± 0.016	0		0.711 ± 0.028	0	
Computer Activity	VUS-opt.	0.587 ± 0.032	8	0.027	0.950 ± 0.006	9	0.010
	Pairs-opt.	0.567 ± 0.043	2		0.947 ± 0.005	1	
California Housing	VUS-opt.	0.258 ± 0.015	10	0.002	0.864 ± 0.005	10	0.002
	Pairs-opt.	0.206 ± 0.013	0		0.842 ± 0.006	0	
Census	VUS-opt.	0.212 ± 0.014	10	0.002	0.846 ± 0.008	9	0.004
	Pairs-opt.	0.180 ± 0.009	0		0.836 ± 0.004	1	
Kinematics	VUS-opt.	0.231 ± 0.031	4	0.322	0.861 ± 0.013	5	0.357
	Pairs-opt.	0.247 ± 0.027	6		0.865 ± 0.008	5	
Delta Elevators	VUS-opt.	0.095 ± 0.033	0	0.002	0.777 ± 0.037	0	0.002
	Pairs-opt.	0.153 ± 0.016	10		0.834 ± 0.009	10	
Total	VUS-opt.		47	0.026		47	0.028
	Pairs-opt.		23			23	

Table 5.3: An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on seven benchmark data sets with five categories in a balanced setting (20%,20%,20%,20%,20%). P -values originate from paired Wilcoxon rank-sum tests over 10 runs for both algorithms with independent training and test sets and no mutual overlap between training sets.

Data set	Method	$\hat{U}(f, D)$	Wins	p-value	C-index	Wins	p-value
Pole Telecomm.	VUS-opt.	0.151 ± 0.039	7	0.232	0.838 ± 0.011	9	0.018
	Pairs-opt.	0.137 ± 0.026	3		0.835 ± 0.009	1	
Bank	VUS-opt.	0.137 ± 0.032	10	0.002	0.778 ± 0.018	10	0.002
	Pairs-opt.	0.100 ± 0.035	0		0.748 ± 0.034	0	
Computer Activity	VUS-opt.	0.481 ± 0.066	5	0.770	0.946 ± 0.007	1	0.004
	Pairs-opt.	0.483 ± 0.052	5		0.954 ± 0.003	9	
California Housing	VUS-opt.	0.269 ± 0.035	10	0.002	0.872 ± 0.005	10	0.002
	Pairs-opt.	0.221 ± 0.028	0		0.854 ± 0.005	0	
Census	VUS-opt.	0.220 ± 0.023	10	0.002	0.868 ± 0.003	10	0.002
	Pairs-opt.	0.141 ± 0.011	0		0.854 ± 0.004	0	
Kinematics	VUS-opt.	0.295 ± 0.061	5	0.492	0.868 ± 0.014	1	0.004
	Pairs-opt.	0.289 ± 0.062	5		0.882 ± 0.011	9	
Delta Elevators	VUS-opt.	0.114 ± 0.039	1	0.020	0.814 ± 0.041	0	0.002
	Pairs-opt.	0.155 ± 0.036	9		0.852 ± 0.010	10	
Total	VUS-opt.		48	<0.001		40	0.271
	Pairs-opt.		22			29	

Table 5.4: An overview of the obtained performance of optimizing r -tuples and optimizing pairs (with Ranking SVM) on seven benchmark data sets with five categories in an unbalanced setting (40%,30%,20%,5%,5%). P -values originate from paired Wilcoxon rank-sum tests over 10 runs for both algorithms with independent training and test sets and no mutual overlap between training sets.

Figure 5.2: An overview of the number of iterations required to find the optimal solution for different data set sizes and different values of r .

5.3.3 Scaling experiments

The running time of Algorithm 4 strongly depends on the number of times the iterative procedure is repeated. Although the number of required iterations is independent of n , it still depends on C , ϵ and the separability of the problem. To have a good view on the separability of the problem, again synthetic data is used for the scaling experiments. To this end, we generated similar data sets as in Section 5.3.1, but now we let vary n as well as r . Training sets ranging from 120 to 1200 cases were sampled with an increase of 120 cases in each step. For the number of classes, we did experiments with $r \in \{2, 3, 4, 5\}$. For the different classes we again assumed the same type of bivariate Gaussian clusters with the same variances and, in increasing order, (10,10), (20,10), (20,20), (30,20), (30,30) as respective cluster means. An equal number of instances was sampled from each cluster. So, when $n = 120$, 60 cases from the first two clusters are sampled for $r = 2$ and 24 cases are sampled from all five clusters for $r = 5$. New data was sampled for every combination of n and r , and the experiments were repeated 10 times, as well with independent data.

Figure 5.2 shows the number of constraints added to the optimization problem for different n and r . For two classes, substantially less constraints are added to the quadratic program. A simple explanation is that the categories

Figure 5.3: An overview of the average time required to solve the quadratic program (left side) and to find the maximum violating constraint (right side) in each iteration of the algorithm. In both cases the average running time is computed by dividing total time by the number of iterations.

are easily distinguishable in this case and, as a result, only a small minority of the slack variables in optimization problem (5.3) becomes active. When more than two clusters are considered, incorrect rankings mainly occur for objects belonging to the intermediate clusters, because these clusters overlap more with other clusters. Since the difference in separability is less striking in these cases, the number of added constraints varies less compared to the case $r = 2$.

Figure 5.3 displays the average running time to solve the quadratic program (left side) and the average running time to compute the argmax in each iteration. First of all, the results explain why in this article no specific optimization algorithms for solving the quadratic program are discussed: because the running time is strongly dominated by computing the argmax. All four curves on the right side manifest the same, approximately cubic, scaling. Thus, only the number of iterations required can vary for different r . The time needed to find the maximum violating constraint is independent of r .

5.4 Conclusion

In this chapter we proposed a new algorithm for ranking data sampled from in general r ordered categories. Inspired by the bipartite case for two categories, we called this setting the layered ranking problem because each correctly ranked r -tuple, consisting of one object of each category, induces a path in the corresponding layered ranking graph. Contrary to conventional approaches for layered ranking in which typically the number of correctly ranked object pairs is optimized, we proposed to learn a ranking function with a maximal number of correctly ranked r -tuples. A naive implementation would require at least $\mathcal{O}(n^{2r})$ computation time, but we showed that the ϵ -approximate solution could be found in $\mathcal{O}(n^3)$ time for any r by reformulating the quadratic program as a structured learning problem.

Notwithstanding the increasing complexity of such an approach, we believe that looking at correctly ranked r -tuples instead of correctly ranked pairs has important benefits and leads to better models. The experiments reported in this chapter clearly support this claim. On synthetic data and several benchmark data sets a statistically significant improvement in performance was measured, definitely in terms of the VUS, but also in terms of the C-index. For some unbalanced problems, a small trade-off appeared between optimizing VUS on the one hand and the C-index on the other hand. We argued that the pairwise approach might focus too strongly on some of the categories instead of assessing the ranking model globally. Nevertheless, we acknowledge that the complexity of ranking r -tuples remains a bottleneck for large samples.

The results presented in this chapter have been published in (Waegeman *et al.* 2007) and (Waegeman *et al.* 2008e).

Ranking representability

In the previous chapters we have studied ranking models, which can be seen as transitive models of preference. When an object $\mathbf{x}_1$ is ranked before an object $\mathbf{x}_2$ and $\mathbf{x}_2$ is ranked before $\mathbf{x}_3$, then, of course, $\mathbf{x}_1$ is ranked before $\mathbf{x}_3$, which means that transitivity holds. In this chapter we further explore the importance of transitivity for ranking problems. We show that it provides the crucial link between ranking models, pairwise preference models and one-versus-one multi-class classification models.

We start by introducing the general framework of cycle-transitivity and reciting the various types of transitivity it covers. Then, we establish the links between ranking representability and transitivity for pairwise preference models by unravelling the relationships between the concepts of weak, moderate and strong ranking representability and their corresponding forms of stochastic transitivity.

In the last section, we study the related and more complex problem of ranking representability of multi-class classification problems. To this end, the ROC concepts and graph formulations of Chapter 4 are extended in order to derive necessary and sufficient conditions for ranking representability of one-versus-one multi-class ensembles. For the three class case, this results in a new type of cycle-transitivity for pairwise AUCs that can be verified by solving an integer quadratic program. Moreover, solving this integer quadratic program can be avoided, since its solution converges for an infinite data sample to a simple form, resulting in a deviation bound that becomes tighter with increasing sample size. In addition, we illustrate the usefulness of our approach empirically.

6.1 Cycle-transitivity

Following the notation introduced in Chapter 2, we will further use $Q : \mathcal{X}^2 \rightarrow [0, 1]$ to denote a reciprocal pairwise preference relation defined on the set of data objects. In the last section, we will also consider preference relations defined on random variables, for which an alternative notation will be employed. In this section we give a detailed introduction to the framework of cycle-transitivity (De Baets *et al.* 2006), which has been quite recently put forward as a unification of fuzzy transitivity on the one hand and stochastic transitivity (Luce & Suppes 1965, Tversky 1998) on the other hand. In (De Baets & De Meyer 2005) it was shown that cycle-transitivity covers *FG*-transitivity, a slightly older unifying framework for fuzzy and stochastic transitivity. Moreover, other types of transitivity, unlike fuzzy or stochastic transitivity, can also be elegantly expressed in terms of cycle-transitivity. We will give a brief overview of all types of cycle-transitivity that have been found so far. For more details we again refer to (De Baets *et al.* 2006). In later section we will then present two new instantiations of cycle-transitivity.

6.1.1 Notations

Let $Q : \mathcal{X}^2 \rightarrow [0, 1]$ be a reciprocal preference relation defined on a set of data objects. For any pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{X}^2$, we first introduce the shorthand notation $Q_{ij} = Q(\mathbf{x}_i, \mathbf{x}_j)$. For any triplet $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ we define

$$\begin{aligned}\alpha_{123} &= \min(Q_{12}, Q_{23}, Q_{31}), \\ \beta_{123} &= \text{median}(Q_{12}, Q_{23}, Q_{31}), \\ \gamma_{123} &= \max(Q_{12}, Q_{23}, Q_{31}).\end{aligned}$$

It now obviously holds that

$$\alpha_{123} \leq \beta_{123} \leq \gamma_{123},$$

and also

$$\begin{aligned}\alpha_{123} &= \alpha_{231} = \alpha_{312}, \\ \beta_{123} &= \beta_{231} = \beta_{312}, \\ \gamma_{123} &= \gamma_{231} = \gamma_{312}.\end{aligned}$$

Since Q satisfies the reciprocity property, we get:

$$\begin{aligned}\alpha_{321} &= 1 - \gamma_{123}, \\ \beta_{321} &= 1 - \beta_{123}, \\ \gamma_{321} &= 1 - \alpha_{123}.\end{aligned}$$

6.1.2 Product-transitivity

Product-transitivity (further denoted T_P -transitivity) can be considered as a specific case of T -transitivity, a popular concept in fuzzy set theory. Further we will give a formal definition of T -transitivity, but here we briefly mention that the product t-norm $T_P(a, b) = ab$ gives rise to a type of T -transitivity and it also forms the basis for the introduction of cycle-transitivity, as shown in (De Baets *et al.* 2006). Product-transitivity can be written as

$$\forall \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{X} : Q_{12}Q_{23} \leq Q_{13}. \quad (6.1)$$

Let us now consider a single triplet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. (6.1) gives rise to the following six conditions on the triplet Q_{12}, Q_{23}, Q_{31} of preference relations:

$$\begin{aligned}Q_{12}Q_{23} &\leq Q_{13}, & Q_{13}Q_{32} &\leq Q_{12}, \\ Q_{23}Q_{31} &\leq Q_{21}, & Q_{21}Q_{13} &\leq Q_{23}, \\ Q_{31}Q_{12} &\leq Q_{32}, & Q_{32}Q_{21} &\leq Q_{31}.\end{aligned}$$

Since Q is reciprocal, these six conditions can be expressed in terms of $\alpha_{123}, \beta_{123}, \gamma_{123}$ as follows:

$$(1 - \beta_{123})(1 - \gamma_{123}) \leq \alpha_{123}, \quad \beta_{123}\gamma_{123} \leq 1 - \alpha_{123}, \quad (6.2)$$

$$(1 - \alpha_{123})(1 - \gamma_{123}) \leq \beta_{123}, \quad \alpha_{123}\gamma_{123} \leq 1 - \beta_{123}, \quad (6.3)$$

$$(1 - \alpha_{123})(1 - \beta_{123}) \leq \gamma_{123}, \quad \alpha_{123}\beta_{123} \leq 1 - \gamma_{123}. \quad (6.4)$$

The three left-hand inequalities can be rewritten as follows:

$$\begin{aligned}\alpha_{123}\beta_{123} &\leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1, \\ \alpha_{123}\gamma_{123} &\leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1, \\ \beta_{123}\gamma_{123} &\leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1,\end{aligned}$$

while the right-hand inequalities can be expressed as:

$$\begin{aligned}\alpha_{123} + \beta_{123} + \gamma_{123} - 1 &\leq 1 - (1 - \alpha_{123})(1 - \beta_{123}), \\ \alpha_{123} + \beta_{123} + \gamma_{123} - 1 &\leq 1 - (1 - \alpha_{123})(1 - \gamma_{123}), \\ \alpha_{123} + \beta_{123} + \gamma_{123} - 1 &\leq 1 - (1 - \beta_{123})(1 - \gamma_{123}).\end{aligned}$$

Since $\alpha_{123}\beta_{123} \leq \alpha_{123}\gamma_{123} \leq \beta_{123}\gamma_{123}$ and $(1 - \alpha_{123})(1 - \beta_{123}) \geq (1 - \alpha_{123})(1 - \gamma_{123}) \geq (1 - \beta_{123})(1 - \gamma_{123})$, we can reduce these six inequalities to one double inequality:

$$\beta_{123}\gamma_{123} \leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1 \leq 1 - (1 - \alpha_{123})(1 - \beta_{123}). \quad (6.5)$$

Let us now introduce the notations $\mathcal{L}(\beta_{123}, \gamma_{123})$ and $\mathcal{U}(\alpha_{123}, \beta_{123})$ for the lower and upper bound in the above expression, then the following equality between both bounds is observed:

$$\mathcal{L}(\beta_{123}, \gamma_{123}) = 1 - \mathcal{U}(1 - \gamma_{123}, 1 - \beta_{123}).$$

Moreover, the obtained lower and upper bound are indifferent to any permutation of $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ and the double inequality holds for both directions of the loop. For example,

$$\begin{aligned} \alpha_{321} + \beta_{321} + \gamma_{321} - 1 &= 1 - (\alpha_{123} + \beta_{123} + \gamma_{123}), \\ &\geq 1 - \mathcal{U}(\alpha_{123}, \beta_{123}), \\ &= 1 - \mathcal{U}(1 - \gamma_{321}, 1 - \beta_{321}) = \mathcal{L}(\beta_{321}, \gamma_{321}). \end{aligned}$$

6.1.3 Definition of cycle-transitivity

The observation made to rewrite $T_{\mathbf{P}}$ -transitivity as the double inequality of (6.5) lays the foundation of cycle-transitivity. Within the framework of cycle-transitivity, the upper bound (and corresponding lower bound) are generalized towards other bounds than the ones given above. To this end, let us define $\Delta = \{(\alpha, \beta, \gamma) \in [0, 1]^3 \mid \alpha \leq \beta \leq \gamma\}$ and consider a function $\mathcal{U} : \Delta \rightarrow \mathbb{R}$, then, by analogy with (6.5), we can call a reciprocal preference relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ cycle-transitive w.r.t. $\mathcal{U}$ if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$

$$\begin{aligned} 1 - \mathcal{U}(1 - \gamma_{123}, 1 - \beta_{123}, 1 - \alpha_{123}) \\ \leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1 \leq \\ \mathcal{U}(\alpha_{123}, \beta_{123}, \gamma_{123}). \end{aligned} \quad (6.6)$$

In case of $T_{\mathbf{P}}$ -transitivity, the upper bound function becomes

$$\mathcal{U}_{T_{\mathbf{P}}}(\alpha, \beta, \gamma) = \alpha + \beta - \alpha\beta.$$

The case where $\mathcal{U}(\alpha, \beta, \gamma) = \beta$ turns out to be another form of fuzzy transitivity as discussed in Section 6.1.5. The double inequality (6.6) leads to two conditions: the lower bound should not exceed the upper bound and the value $\alpha + \beta + \gamma - 1$ should be located between both bounds.

Definition 6.1. A function $\mathcal{U} : \Delta \rightarrow \mathbb{R}$ is called an upper bound function if it satisfies the following properties:

1. $\mathcal{U}(0, 0, 1) \geq 0$ and $\mathcal{U}(0, 1, 1) \geq 1$,
2. for any $\alpha, \beta, \gamma \in \Delta$:

$$\mathcal{U}(\alpha, \beta, \gamma) + \mathcal{U}(1 - \gamma, 1 - \beta, 1 - \alpha) \geq 1. \quad (6.7)$$

The class of upper bound functions will be denoted $\mathfrak{U}$.

The definition of an upper bound function does not include any monotonicity condition. We define the dual lower bound function $\mathcal{L} : \Delta \rightarrow \mathbb{R}$ of a given upper bound function $\mathcal{U}$ as

$$\mathcal{L}(\alpha, \beta, \gamma) = 1 - \mathcal{U}(1 - \gamma, 1 - \beta, 1 - \alpha),$$

implying that $\mathcal{L} \leq \mathcal{U}$ when (6.7) holds. These tools give us the opportunity to define formally the concept of cycle-transitivity.

Definition 6.2. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called cycle-transitive w.r.t. an upper bound function $\mathcal{U} \in \mathfrak{U}$ if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$ it holds that

$$\mathcal{L}(\alpha_{123}, \beta_{123}, \gamma_{123}) \leq \alpha_{123} + \beta_{123} + \gamma_{123} - 1 \leq \mathcal{U}(\alpha_{123}, \beta_{123}, \gamma_{123}) \quad (6.8)$$

where $\mathcal{L}$ is the dual lower bound function of $\mathcal{U}$.

From this construction immediately follows that, as soon as the double inequality is fulfilled for a triplet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$, it is also fulfilled for any permutation of the triplet. Therefore, in practice one only needs to check (6.8) for a single permutation of $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Alternatively, due to the same duality, one can also opt to verify only the upper bound, or equivalently the lower bound, for two permutations of $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ that are not cyclic permutations of one another. This is summarized as follows.

Proposition 6.3. (De Baets et al. 2006) A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is cycle-transitive w.r.t. an upper bound function $\mathcal{U} \in \mathfrak{U}$ if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$ it holds that

$$\alpha_{123} + \beta_{123} + \gamma_{123} - 1 \leq \mathcal{U}(\alpha_{123}, \beta_{123}, \gamma_{123}), \quad (6.9)$$

The most loose upper bound one can choose is the function $\mathcal{U} = 2$, which means that there is no restriction on the values the preference relation can take. It will become clear later that the upper bound function represents a very straightforward way to link different types of transitivity and, in particular, to determine whether a particular form of transitivity follows from another form of transitivity. For example, given two types of transitivity A and B that can be casted in the framework of cycle-transitivity by means of upper bound functions $\mathcal{U}_A$ and $\mathcal{U}_B$ such that

$$\forall(\alpha, \beta, \gamma) \in \Delta : \mathcal{U}_A(\alpha, \beta, \gamma) \leq \mathcal{U}_B(\alpha, \beta, \gamma),$$

we automatically know that type- A transitivity implies type- B transitivity. It is shown in the following sections how cycle-transitivity includes different types of transitivity.

6.1.4 The triangle inequality

Remarkably, a simple and well-known example of cycle-transitivity is the triangle inequality.

Definition 6.4. *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ satisfies the triangle inequality if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$*

$$Q(\mathbf{x}_1, \mathbf{x}_3) \leq Q(\mathbf{x}_1, \mathbf{x}_2) + Q(\mathbf{x}_2, \mathbf{x}_3).$$

Proposition 6.5. *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow \mathbb{R}$ satisfies the triangle inequality if and only if it is cycle-transitive w.r.t. to the upper bound function $\mathcal{U}_\Delta = 1$.*

Proof. The lower bound directly follows from the definition, while the upper bound is found by applying the reciprocity property. $\square$

We remark that the triangle inequality has already been considered by Luce & Suppes (1965), Koppen (1995) as a property to characterize reciprocal preference relations.

6.1.5 Fuzzy transitivity

T -transitivity is an important concept in the fuzzy set literature as a desirable property of fuzzy relations. In this work we will only consider the case where

fuzzy relations are reciprocal relations, a condition that does not hold in general. The traditional definition given in terms of t-norms can be generalized to the more general class of conjunctors. As shown by De Baets *et al.* (2006), we will start with this more general case in order to establish the link with cycle-transitivity.

Definition 6.6. A binary operation $C : [0, 1]^2 \rightarrow [0, 1]$ is called a conjunctor if it satisfies the following properties:

1. Its restriction to $\{0, 1\}^2$ coincides with the boolean conjunction.
2. Monotonicity: C is increasing in both variables.

This gives us the opportunity to define C -transitivity.

Definition 6.7. A fuzzy relation $R : \mathcal{X}^2 \rightarrow [0, 1]$ is called C -transitive with C a conjunctor if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$

$$C(R(\mathbf{x}_1, \mathbf{x}_2), R(\mathbf{x}_2, \mathbf{x}_3)) \leq R(\mathbf{x}_1, \mathbf{x}_3). \quad (6.10)$$

Given the restriction to reciprocal relations, the following proposition characterizes the reformulation in terms of cycle-transitivity.

Proposition 6.8. (De Baets *et al.* 2006) Let C be a commutative conjunctor such that $C \leq T_M$ (see below). A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is C -transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_C$ defined by

$$\mathcal{U}_C(\alpha, \beta, \gamma) = \min(\alpha + \beta - C(\alpha, \beta), \alpha + \gamma - C(\alpha, \gamma), \beta + \gamma - C(\beta, \gamma)).$$

6.1.5.a The case of copulas

Copulas and quasi-copulas are specific types of conjunctors. Their corresponding upper bound functions can be further simplified.

Definition 6.9. A binary operation $\mathcal{C} : [0, 1]^2 \rightarrow [0, 1]$ is called a quasi-copula if it satisfies the following properties:

- Neutral element 1: $\forall a \in [0, 1] : \mathcal{C}(1, a) = \mathcal{C}(a, 1) = a$.
- Absorbing element 0: $\forall a \in [0, 1] : \mathcal{C}(0, a) = \mathcal{C}(a, 0) = 0$.
- Monotonicity: $\mathcal{C}$ is increasing in both variables.
- 1-Lipschitz property: $\forall (a_1, a_2, b_1, b_2) \in [0, 1]^4 :$

$$|\mathcal{C}(a_1, b_1) - \mathcal{C}(a_2, b_2)| \leq |a_1 - a_2| + |b_1 - b_2|.$$

If instead of the 1-Lipschitz property the following property is satisfied

- *Moderate growth*: $\forall (a_1, a_2, b_1, b_2) \in [0, 1]^4$:

$$a_1 \leq a_2 \wedge b_1 \leq b_2 \Rightarrow \mathcal{C}(a_1, b_1) + \mathcal{C}(a_2, b_2) \geq \mathcal{C}(a_1, b_2) + \mathcal{C}(a_2, b_1) ,$$

then it is called a copula.

Proposition 6.10. (De Baets et al. 2006) Let $\mathcal{C}$ be a commutative copula such that $g \leq T_{\mathbf{M}}$. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called $\mathcal{C}$ -transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_{\mathcal{C}}$ defined by

$$\mathcal{U}_{\mathcal{C}}(\alpha, \beta, \gamma) = \alpha + \beta - \mathcal{C}(\alpha, \beta) .$$

6.1.5.b The case of t-norms

Another type of conjunctors are t-norms. They are defined as follows.

Definition 6.11. A binary operation $T : [0, 1]^2 \rightarrow [0, 1]$ is called a t-norm if it satisfies the following properties:

- *Neutral element 1*: $\forall a \in [0, 1] : T(1, a) = T(a, 1) = a$.
- *Monotonicity*: T is increasing in both variables.
- *Commutativity*: $\forall (a, b) \in [0, 1]^2 : T(a, b) = T(b, a)$.
- *Associativity*: $\forall (a, b, c) \in [0, 1]^3 : T(T(a, b), c) = T(a, T(b, c))$.

A close relationships exists between t-norms and copulas, since t-norms having the 1-Lipschitz correspond to associative copulas. Three important t-norms that will appear further in this chapter are the minimum t-norm $T_{\mathbf{M}}(a, b) = \min(a, b)$, the product t-norm $T_{\mathbf{P}}(a, b) = ab$ and the Łukasiewicz t-norm $T_{\mathbf{L}}(a, b) = \max(a + b - 1, 0)$. They are copulas as well and they respectively define the following upper bound functions:

$$\begin{aligned} \mathcal{U}_{T_{\mathbf{M}}}(\alpha, \beta, \gamma) &= \beta , \\ \mathcal{U}_{T_{\mathbf{P}}}(\alpha, \beta, \gamma) &= \alpha + \beta - \alpha\beta , \\ \mathcal{U}_{T_{\mathbf{L}}}(\alpha, \beta, \gamma) &= 1 . \end{aligned}$$

This means that $T_{\mathbf{L}}$ -transitivity is equivalent to the triangle inequality.

6.1.6 Stochastic transitivity

We first introduced fuzzy transitivity for its straightforward reformulation in terms of cycle-transitivity. On the other hand, stochastic transitivity is a fairly different framework for characterizing preference relations. Historically, it has played a more dominant role than fuzzy transitivity for reciprocal relations. As we will show in Section 6.3, stochastic transitivity is closely connected to ranking representability of preference relations.

Definition 6.12. Let g be an increasing $[1/2, 1]^2 \rightarrow [0, 1]$ mapping. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called g -stochastic transitive if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$

$$Q(\mathbf{x}_1, \mathbf{x}_2) \geq 1/2 \wedge Q(\mathbf{x}_2, \mathbf{x}_3) \geq 1/2 \Rightarrow Q(\mathbf{x}_1, \mathbf{x}_3) \geq g(Q(\mathbf{x}_1, \mathbf{x}_2), Q(\mathbf{x}_2, \mathbf{x}_3)).$$

Many specific types of stochastic transitivity can be found in the literature:

1. Weak stochastic transitivity when $\forall(a, b) \in [0, 1]^2 : g(a, b) = 1/2$.
2. Moderate stochastic transitivity when $\forall(a, b) \in [0, 1]^2 : g(a, b) = \min(a, b)$.
3. Strong stochastic transitivity when $\forall(a, b) \in [0, 1]^2 : g(a, b) = \max(a, b)$.
4. λ -transitivity, with $\lambda \in [0, 1]$, when $\forall a, b \in [0, 1]^2 : g(a, b) = \lambda \max(a, b) + (1 - \lambda) \min(a, b)$.
5. Product stochastic transitivity when $\forall(a, b) \in [0, 1]^2 : g(a, b) = ab$. Historically, this type of transitivity has been denoted the *multiplicative rule* (Luce & Suppes 1965). Further we will only use the term product stochastic transitivity in order to avoid a possible confusion with multiplicative transitivity and product-transitivity (T_P -transitivity).

Proposition 6.13. (De Baets et al. 2006) Let g be a commutative, increasing $[1/2, 1]^2 \rightarrow [0, 1]$ mapping such that $g(1/2, x) \leq x$ for any $x \in [1/2, 1]$. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ \min(\alpha + \beta - g(\alpha, \beta), \alpha + \gamma - g(\alpha, \gamma), & \\ \quad \beta + \gamma - g(\beta, \gamma)), & \text{if } \alpha \geq 1/2, \\ 2 & \text{if } \beta < 1/2. \end{cases}$$

Proposition 6.14. (De Baets et al. 2006) *Let g be a commutative, increasing $[1/2, 1]^2 \rightarrow [1/2, 1]$ mapping such that $g(1/2, x) \leq x$ for any $x \in [1/2, 1]$. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by*

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ 1/2, & \text{if } \alpha \geq 1/2, \\ 2 & \text{if } \beta < 1/2. \end{cases}$$

Proposition 6.15. (De Baets et al. 2006) *Let g be a commutative, increasing $[1/2, 1]^2 \rightarrow [1/2, 1]$ mapping with neutral element $1/2$. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by*

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \beta \geq 1/2, \\ 2 & \text{if } \beta < 1/2. \end{cases}$$

Remark that for the case $\beta < 1/2$ the upper bound is always satisfied since in that case $\alpha + \beta + \gamma - 1 \leq \gamma$. Consequently, the upper bound function can be replaced by the following tighter bound.

Corollary 6.16. *Let g be a commutative, increasing $[1/2, 1]^2 \rightarrow [0, 1]$ mapping such that $g(1/2, x) \leq x$ for any $x \in [1/2, 1]$. A reciprocal relation is $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by*

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ \min(\alpha + \beta - g(\alpha, \beta), \alpha + \gamma - g(\alpha, \gamma), \\ \quad \beta + \gamma - g(\beta, \gamma)), & \text{if } \alpha \geq 1/2, \\ \gamma, & \text{if } \beta < 1/2. \end{cases}$$

We will only use the latter upper bound function, because implications between different types of transitivity can be better deduced from it. By putting additional restrictions on g , the upper bound function can be further simplified as follows.

Proposition 6.17. *Let g be a commutative, increasing, 1-Lipschitz $[0, 1]^2 \rightarrow [0, 1]$ mapping such that $g(1/2, x) \leq x$ for any $x \in [0, 1]$. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by*

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \alpha < 1/2, \\ \alpha + \beta - g(\alpha, \beta), & \text{if } \alpha \geq 1/2. \end{cases}$$

Proof. The simplification follows from the 1-Lipschitz property implying

$$\alpha + \beta - g(\alpha, \beta) \leq \alpha + \gamma - g(\alpha, \gamma) \leq \beta + \gamma - g(\beta, \gamma).$$

□

Proposition 6.18. *Let h be a commutative, increasing $[1/2, 1]^2 \rightarrow [1/2, 1]$ mapping with neutral element $1/2$ and let g be a commutative, increasing $[0, 1]^2 \rightarrow [0, 1]$ mapping such that $g(1/2, x) \leq x$ for any $x \in [0, 1]$ and g coincides with h on the input domain of h . A reciprocal relation $Q : \mathcal{X}^2 \rightarrow \mathbb{R}$ is g -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_g$ defined by*

$$\mathcal{U}_g(\alpha, \beta, \gamma) = \beta + \gamma - g(\beta, \gamma).$$

For particular forms of stochastic transitivity, the upper bound function can be further simplified. The following upper bound functions are obtained by applying Prop. 6.17.

Proposition 6.19. *A reciprocal relation Q is strongly stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_{SST}$ defined by*

$$\mathcal{U}_{SST}(\alpha, \beta, \gamma) = \begin{cases} \beta, & \text{if } \beta \geq 1/2, \\ \gamma, & \text{if } \beta < 1/2. \end{cases}$$

Proposition 6.20. *A reciprocal relation Q is moderately stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_{MST}$ defined by*

$$\mathcal{U}_{MST}(\alpha, \beta, \gamma) = \begin{cases} \gamma, & \text{if } \alpha < 1/2, \\ 1/2, & \text{if } \alpha \geq 1/2. \end{cases}$$

Proposition 6.21. *A reciprocal relation Q is weakly stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_{WST}$ defined by*

$$\mathcal{U}_{WST}(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - 1/2, & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ 1/2, & \text{if } \alpha \geq 1/2. \end{cases}$$

Proposition 6.22. *A reciprocal relation Q is λ -stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_\lambda$ defined by*

$$\mathcal{U}_\lambda(\alpha, \beta, \gamma) = \begin{cases} \lambda\beta + (1 - \lambda)\gamma, & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ 1/2, & \text{if } \alpha \geq 1/2. \end{cases}$$

Proposition 6.23. *A reciprocal relation Q is product stochastic transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_{PST}$ defined by*

$$\mathcal{U}_{PST}(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - \beta\gamma, & \text{if } \beta \geq 1/2 \wedge \alpha < 1/2, \\ \alpha + \beta - \alpha\beta, & \text{if } \alpha \geq 1/2. \end{cases}$$

6.1.7 Self-dual upper bound functions

De Baets *et al.* (2006) call an upper bound function self-dual if for any $(\alpha, \beta, \gamma) \in \Delta$

$$\mathcal{U}(\alpha, \beta, \gamma) + \mathcal{U}(1 - \gamma, 1 - \beta, 1 - \alpha) = 1.$$

Then, the lower and upper bound coincide and the double inequality becomes an equality constraint. Furthermore, it holds that $\mathcal{U}(0, 0, 1) = 0$ and $\mathcal{U}(0, 1, 1) = 1$. The simplest self-dual upper bound function is the median and turned out to correspond to T_M -transitivity. Some other types of transitivity from the literature can be represented by means of a self-dual upper bound function.

6.1.7.a Multiplicative transitivity

Definition 6.24. (Luce & Suppes 1965) *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called multiplicatively transitive if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$*

$$Q(\mathbf{x}_1, \mathbf{x}_2)Q(\mathbf{x}_2, \mathbf{x}_3)Q(\mathbf{x}_3, \mathbf{x}_1) = Q(\mathbf{x}_2, \mathbf{x}_1)Q(\mathbf{x}_1, \mathbf{x}_3)Q(\mathbf{x}_3, \mathbf{x}_2).$$

Proposition 6.25. (De Baets *et al.* 2006) *A reciprocal relation Q is multiplicative transitive if and only if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_E$ defined by*

$$\mathcal{U}_E(\alpha, \beta, \gamma) = \alpha\beta + \alpha\gamma + \beta\gamma - 2\alpha\beta\gamma.$$

6.1.7.b Isostochastic transitivity

Definition 6.26. *Let g be a commutative, increasing $[1/2, 1]^2 \rightarrow [1/2, 1]$ mapping. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called g -isostochastic transitive if for any $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \mathcal{X}^3$*

$$Q(\mathbf{x}_1, \mathbf{x}_2) \geq 1/2 \wedge Q(\mathbf{x}_2, \mathbf{x}_3) \geq 1/2 \Rightarrow Q(\mathbf{x}_1, \mathbf{x}_3) = g(Q(\mathbf{x}_1, \mathbf{x}_2), Q(\mathbf{x}_2, \mathbf{x}_3)).$$

Proposition 6.27. (De Baets et al. 2006) A reciprocal relation Q is g -isostochastic transitive if and only if it is cycle-transitive w.r.t. the self-dual upper bound function $\mathcal{U}_{\tilde{g}}$ defined by

$$\mathcal{U}_{\tilde{g}}(\alpha, \beta, \gamma) = \begin{cases} \beta + \gamma - g(\beta, \gamma), & \text{if } \beta \geq 1/2, \\ \alpha + \beta - 1 + g(1 - \beta, 1 - \alpha), & \text{if } \beta < 1/2. \end{cases}$$

T_M -transitivity and multiplicative transitivity both can be seen as special cases of isostochastic transitivity by defining g respectively as $g(a, b) = \max(a, b)$ and

$$g(a, b) = \frac{ab}{ab + (1 - a)(1 - b)}.$$

6.1.8 The probabilistic sum

Another class of interesting upper bound functions is inspired by the probabilistic sum, as discussed in (De Baets *et al.*, unpublished). More specifically, the following three upper bound functions can be derived from the probabilistic sum:

$$\begin{aligned} \mathcal{U}_{T_P}(\alpha, \beta, \gamma) &= \alpha + \beta - \alpha\beta, \\ \mathcal{U}_P(\alpha, \beta, \gamma) &= \alpha + \gamma - \alpha\gamma, \\ \mathcal{U}_D(\alpha, \beta, \gamma) &= \beta + \gamma - \beta\gamma. \end{aligned}$$

There exists a graph-theoretic interpretation to describe the link between these three upper bound functions. We already saw that the first one characterizes product-transitivity. The second one has recently found an application in the fields of partially ordered sets and will not be discussed further on. Finally, the third upper bound function characterizes reciprocal relations for dice games (De Schuymer *et al.* 2003) and for comparing independent random variables (De Schuymer *et al.* 2005). Because of that, this type of transitivity has been called dice transitivity, and we will need it to analyze the ranking representability of one-versus-one multi-class ensembles later in this chapter. A detailed discussion of dice models and dice transitivity is for this reason postponed to Section 6.4.3.

6.2 Links between different types of transitivity

The framework of cycle-transitivity yields an interesting tool to express links between different types of transitivity. The results hereunder all immediately follow from the respective upper bound functions.

Proposition 6.28. *The following statements hold:*

- *Strong stochastic transitivity implies λ -stochastic transitivity.*
- *λ -stochastic transitivity implies moderate stochastic transitivity.*
- *Moderate stochastic transitivity implies weak stochastic transitivity.*
- *Moderate stochastic transitivity implies product stochastic transitivity.*
- *Neither weak stochastic transitivity nor product stochastic transitivity implies the other.*
- *Product stochastic transitivity implies dice transitivity.*
- *T_M -transitivity implies T_P -transitivity.*
- *T_P -transitivity implies cycle-transitivity w.r.t. to the upper bound function U_P .*
- *Cycle-transitivity w.r.t. to the upper bound function U_P implies dice transitivity.*
- *Dice transitivity implies the triangle inequality (T_L -transitivity).*

Some of these observations have been proven in the early work of Luce & Suppes (1965), but within the framework of cycle transitivity they follow much more naturally by examining the upper bound functions. With a bit more effort, other links can be derived from the upper bound functions.

Proposition 6.29. *Multiplicative transitivity implies strong stochastic transitivity.*

Proof. To prove the proposition, it is sufficient to prove that

$$\alpha\beta + \alpha\gamma + \beta\gamma - 2\alpha\beta\gamma \leq \beta.$$

Writing the equality

$$\alpha + \beta + \gamma - 1 = \alpha\beta + \alpha\gamma + \beta\gamma - 2\alpha\beta\gamma,$$

as a function of α and β yields:

$$\gamma = \frac{\alpha + \beta - \alpha\beta - 1}{\alpha + \beta - 2\alpha\beta - 1}.$$

Substituting this back into U_E gives:

$$\begin{aligned} U_E &= \alpha\beta + \alpha\gamma + \beta\gamma - 2\alpha\beta\gamma \\ &= \alpha\beta + (\alpha + \beta - 2\alpha\beta) \frac{\alpha + \beta - \alpha\beta - 1}{\alpha + \beta - 2\alpha\beta - 1} \\ &= \alpha + \beta - 1 + \frac{\alpha + \beta - \alpha\beta - 1}{\alpha + \beta - 2\alpha\beta - 1}. \end{aligned}$$

Hence, it is sufficient to prove that:

$$\frac{\alpha + \beta - \alpha\beta - 1}{\alpha + \beta - 2\alpha\beta - 1} \leq 1 - \alpha. \quad (6.11)$$

Since

$$\begin{aligned} \alpha + \beta - 2\alpha\beta - 1 &\leq \alpha + \beta - \alpha\beta - 1 \\ &= \alpha + (1 - \alpha)\beta - 1 \\ &\leq 0, \end{aligned}$$

the condition can be rewritten as:

$$\alpha + \beta - \alpha\beta - 1 \geq (1 - \alpha)(\alpha + \beta - 2\alpha\beta - 1), \quad (6.12)$$

and is trivially fulfilled for $0 \leq \alpha \leq \beta \leq 1$. $\square$

This result can be generalized towards isostochastic transitivity.

Proposition 6.30. *(De Schuymer 2005) Isostochastic transitivity implies strong stochastic transitivity.*

Proof. The statement follows directly from the fact that in case of isostochastic transitivity $g \geq \max$. $\square$

Another question to ask is how self-dual upper bound functions compare mutually. For self-dual upper bound functions, the set of permissible values for (α, β, γ) lies on a plane within the intersection of the unit cube and the inequalities $\alpha \leq \beta \leq \gamma$. Consequently, any self-dual upper bound function does not imply another self-dual upper bound function, but usually points can be found that satisfy several self-dual upper bound functions simultaneously. The following statement describes the particular case of multiplicative transitivity and T_M -transitivity.

Proposition 6.31. *Multiplicative transitivity and T_M -transitivity coincide in the following two situations:*

1. $\alpha = 0 \wedge \gamma = 1$.
2. $\beta = 1/2$.

Proof. The self-dual upper bound functions of multiplicative transitivity and T_M -transitivity respectively result in the following two conditions on γ :

$$\gamma = \frac{\alpha + \beta - \alpha\beta - 1}{\alpha + \beta - 2\alpha\beta - 1}, \quad (6.13)$$

$$\gamma = 1 - \alpha. \quad (6.14)$$

Similar to the proof of the previous proposition, we arrive at (6.12), which now becomes an equality constraint. This yields:

$$\begin{aligned} \alpha + \beta - \alpha\beta - 1 &\geq (1 - \alpha)(\alpha + \beta - 2\alpha\beta - 1) \\ \Leftrightarrow \quad \beta &= \alpha\beta + (1 - \alpha)(\alpha + \beta - \alpha\beta) \\ \Leftrightarrow \quad \alpha^2 + 2\alpha\beta - 2\alpha^2\beta - \alpha &= 0 \\ \Leftrightarrow \quad \alpha(\alpha - 1 - 2\beta(\alpha - 1)) &= 0 \\ \Leftrightarrow \quad \alpha(\alpha - 1)(1 - 2\beta) &= 0. \end{aligned}$$

Thus, three cases can be distinguished: $\alpha = 0$, $\alpha = 1$ and $\beta = 1/2$. The case $\alpha = 1$ must be omitted since it violates $\alpha \leq \gamma$ when (6.14) holds. $\square$

Proposition 6.32. *Multiplicative transitivity implies T_P -transitivity.*

Proof. We find:

$$\begin{aligned} \mathcal{U}_E(\alpha, \beta, \gamma) &= \alpha\beta + \alpha\gamma + \beta\gamma - 2\alpha\beta\gamma, \\ &= \gamma(\alpha + \beta - \alpha\beta) + (1 - \gamma)\alpha\beta, \end{aligned}$$

then it suffices to show that

$$\alpha\beta \leq \alpha + \beta - \alpha\beta.$$

This condition can be rewritten as follows:

$$(2\beta - 1)\alpha \leq \beta,$$

and is always fulfilled when $\alpha \leq \beta$. $\square$

All links between different types of transitivity are summarized in Figure 6.1.

Figure 6.1: An overview of the links between different types of transitivity. Arrows denote implications from one type to another. The types given in green are specific types of fuzzy transitivity.

6.3 Ranking representability of preference models

In Chapter 2 we introduced ordinal regression models and pairwise preference models as two different ways to express human preference judgments in a statistical way, and we claimed that the underlying ranking function plays a crucial role in both types of models. In Chapters 3-5 this was shown in detail for ordinal regression models. Here we briefly illustrate the importance of the ranking function for pairwise preference models. More specifically, the existence of an underlying ranking function heavily depends upon the transitivity of pairwise preference models. Let us to this end introduce the concept of ranking representability.

Definition 6.33. *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called weakly ranking representable if there exists a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that:*

$$\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} : Q(\mathbf{x}_i, \mathbf{x}_j) \leq \frac{1}{2} \Leftrightarrow f(\mathbf{x}_i) \leq f(\mathbf{x}_j).$$

This concept was originally introduced in domains like decision making and social choice theory, where it is usually referred to as utility representability or numerical representability (Luce & Suppes 1965, Fishburn 1970). Given the subject of this dissertation, we will further only use ranking representability as terminology. We also remark that in recent literature ranking representability has been mainly studied for fuzzy preference relations. Billot (1995) and Fono & Andjiga (2007) establish links between the ranking representability of fuzzy preference relations and different types of fuzzy transitivity. For reciprocal relations we can derive a similar connection.

Proposition 6.34. *Let $\mathcal{X}$ be a countable infinite set, then a reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is weakly ranking representable if and only if it is weakly stochastically transitive.*

Proof. We give a simple proof by contradiction to prove that weak ranking representability implies weak stochastic transitivity. Suppose therefore that Q is weakly ranking representable and not weakly stochastically transitive. This implies that we can find $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{X}$ such that:

$$\begin{aligned} Q(\mathbf{x}_1, \mathbf{x}_2) &\geq \frac{1}{2} \\ Q(\mathbf{x}_2, \mathbf{x}_3) &\geq \frac{1}{2} \\ Q(\mathbf{x}_1, \mathbf{x}_3) &< \frac{1}{2}. \end{aligned}$$

This would also imply that:

$$\begin{aligned} f(\mathbf{x}_1) &\geq f(\mathbf{x}_2), \\ f(\mathbf{x}_2) &\geq f(\mathbf{x}_3), \\ f(\mathbf{x}_1) &< f(\mathbf{x}_3). \end{aligned}$$

To show that weak stochastic transitivity implies weak ranking representability, we give a proof by induction. Obviously, Q is ranking representable when $\mathcal{X}$ contains two elements. Let us now suppose that it is also ranking representable when $\mathcal{X}$ contains n elements. As a consequence, we can give the n elements a new index such that:

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \dots \leq f(\mathbf{x}_n). \quad (6.15)$$

If we extend $\mathcal{X}$ with a new element $\mathbf{x}_{n+1}$, then we have to consider two cases. If we assume that

$$\forall i \in \{1, \dots, n\} : Q(\mathbf{x}_{n+1}, \mathbf{x}_i) \geq \frac{1}{2}, \quad (6.16)$$

then we can simply assign to $f(\mathbf{x}_{n+1})$ the highest rank. In the other case, it follows from the weak stochastic transitivity of Q that we can find an index $k \in \{1, \dots, n\}$ such that

$$\forall i \in \{1, \dots, k-1\} : Q(\mathbf{x}_{n+1}, \mathbf{x}_i) \geq \frac{1}{2} \wedge \forall i \in \{k, \dots, n\} : Q(\mathbf{x}_{n+1}, \mathbf{x}_i) < \frac{1}{2}.$$

In this case, we place the new element between $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$ in the ranking. $\square$

As pointed out by Switalski (2003), a weakly ranking representable reciprocal relation can be characterized in terms of a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ in the following construction:

$$Q(\mathbf{x}_1, \mathbf{x}_2) = g(f(\mathbf{x}_1), f(\mathbf{x}_2)), \quad (6.17)$$

such that $g : \mathbb{R}^2 \rightarrow \mathbb{R}$ satisfies:

$$\begin{aligned} \forall a, b \in \mathbb{R} : g(a, b) &> \frac{1}{2} \Leftrightarrow a > b, \\ \forall a, b \in \mathbb{R} : g(a, b) &= \frac{1}{2} \Leftrightarrow a = b, \end{aligned}$$

By imposing additional conditions on g , one can define more restrictive forms of ranking representability.

Definition 6.35. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called moderately ranking representable if it can be written as (6.17) with g of the form

$$g(f(\mathbf{x}_1), f(\mathbf{x}_2)) = \bar{F}\left(\frac{f(\mathbf{x}_1) - f(\mathbf{x}_2)}{d(\mathbf{x}_1, \mathbf{x}_2)}\right),$$

where $\bar{F} : \mathbb{R} \rightarrow [0, 1]$ is a cumulative distribution function satisfying $\bar{F}(0) = \frac{1}{2}$ and $d : \mathcal{X}^2 \rightarrow \mathbb{R}$ is a metric.

Proposition 6.36. (Carroll et al. 1990) A moderately ranking representable reciprocal relation is moderately stochastically transitive.

Definition 6.37. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called strongly ranking representable if it can be written as (6.17) with g of the form

$$g(f(\mathbf{x}_1), f(\mathbf{x}_2)) = \bar{F}(f(\mathbf{x}_1) - f(\mathbf{x}_2)),$$

where $\bar{F} : \mathbb{R} \rightarrow [0, 1]$ is a cumulative distribution function satisfying $\bar{F}(0) = \frac{1}{2}$.

Proposition 6.38. (Luce & Suppes 1965) A strongly ranking representable reciprocal relation is strongly stochastically transitive.

Definition 6.39. A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called strictly ranking representable if it can be written as (6.17) with g of the form

$$g(f(\mathbf{x}_1), f(\mathbf{x}_2)) = \frac{f(\mathbf{x}_1)}{f(\mathbf{x}_1) + f(\mathbf{x}_2)}. \quad (6.18)$$

where $\bar{F} : \mathbb{R} \rightarrow [0, 1]$ is a cumulative distribution function satisfying $\bar{F}(0) = \frac{1}{2}$.

Proposition 6.40. (Luce & Suppes 1965) A strictly ranking representable reciprocal relation is multiplicatively transitive.

Remark that a strictly ranking representable model is exactly the Bradley-Terry model as defined in Chapter 2. We refer to Luce & Suppes (1965), Carroll *et al.* (1990), Ballinger & Wilcox (1997), Tversky (1998), Swital-ski (2003), Dhazafarov (2003), Zhang (2004) for proofs and a more detailed discussion on the topic. One can easily verify that:

- Strict ranking representability implies strong ranking representability.
- Strong ranking representability implies moderate ranking representability.

- Moderate ranking representability implies weak ranking representability.

Moreover, additional properties of reciprocal preference relations can be investigated using the framework of cycle-transitivity.

Proposition 6.41. *For a strongly ranking representable reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$, the following properties hold:*

$$\begin{aligned} \forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X} : Q(\mathbf{x}_i, \mathbf{x}_j) \geq Q(\mathbf{x}_i, \mathbf{x}_k) &\Leftrightarrow Q(\mathbf{x}_j, \mathbf{x}_k) \leq \frac{1}{2}, \\ \forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X} : Q(\mathbf{x}_i, \mathbf{x}_j) \geq Q(\mathbf{x}_k, \mathbf{x}_j) &\Leftrightarrow Q(\mathbf{x}_i, \mathbf{x}_k) \geq \frac{1}{2}. \end{aligned}$$

Proof. We find that for all $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}$:

$$\begin{aligned} Q(\mathbf{x}_i, \mathbf{x}_j) \geq Q(\mathbf{x}_i, \mathbf{x}_k) &\Leftrightarrow f(\mathbf{x}_i) - f(\mathbf{x}_j) \geq f(\mathbf{x}_i) - f(\mathbf{x}_k) \\ &\Leftrightarrow f(\mathbf{x}_j) \leq f(\mathbf{x}_k) \\ &\Leftrightarrow Q(\mathbf{x}_j, \mathbf{x}_k) \leq \frac{1}{2}. \end{aligned}$$

The proof of the second property is similar. □

Proposition 6.42. *Let $\bar{F} : \mathbb{R} \rightarrow [0, 1]$ be a cumulative distribution function with the additional properties:*

1. $\bar{F}(0) = \frac{1}{2}$,
2. $\bar{F}(\cdot)$ is convex in $] -\infty, 0]$,
3. $\bar{F}(\cdot)$ is concave in $[0, +\infty[$,

then the reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ defined by Eq. 6.18 is cycle-transitive w.r.t. to the upper bound function $\mathcal{U}_{con}$:

$$\mathcal{U}_{con}(\alpha, \beta, \gamma) = \max(\beta, 1/2).$$

Proof. The proof has been given by Switalski (2003) within the framework of FG-transitivity. He proves the following property when $\bar{F}$ fulfills the conditions stated above:

$$\begin{aligned} Q(\mathbf{x}_1, \mathbf{x}_2) \geq 1/2 \wedge Q(\mathbf{x}_2, \mathbf{x}_3) \geq 1/2 &\Rightarrow \\ Q(\mathbf{x}_1, \mathbf{x}_3) &\leq Q(\mathbf{x}_1, \mathbf{x}_2) + Q(\mathbf{x}_2, \mathbf{x}_3) - 1/2. \end{aligned}$$

If we rewrite the consequent of this implication in terms of cycle-transitivity, then we get

$$Q(\mathbf{x}_1, \mathbf{x}_3) + Q(\mathbf{x}_2, \mathbf{x}_1) + Q(\mathbf{x}_3, \mathbf{x}_2) \leq 3/2.$$

From the antecedent follows that this condition must be fulfilled when $\beta \leq 1/2$. Combining this upper bound with the upper bound of strong stochastic transitivity leads to the above form of cycle-transitivity. $\square$

One can further restrict $\overline{F}$ to a linear function, which implies that the cumulative distribution function becomes 0 on the outer left side of the real line and becomes 1 on the outer right side of the real line. In an interval $[-\epsilon, \epsilon]$ with ϵ not further specified, we get a linear increase such that $\overline{F}(a) + \overline{F}(b) = \overline{F}(a + b)$. It has been observed that the reciprocal relation satisfies in this case additive transitivity (Tanino 1990). As summarized in the following proposition, additive transitivity can be included within cycle-transitivity (when the definition of an upper bound function is slightly relaxed).

Proposition 6.43. *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow \mathbb{R}$ is called additively transitive if it is cycle-transitive w.r.t. the upper bound function $\mathcal{U}_A$ defined by*

$$\mathcal{U}_A(\alpha, \beta, \gamma) = 1/2.$$

$\mathcal{U}_A$ is a self-dual upper bound function.

6.4 Ranking representability of one-versus-one ensembles

In this section we tackle a slightly different problem compared to the previous section. However, studying the ranking representability of one-versus-one multi-class classification ensembles will exhibit some important similarities with the ranking representability of reciprocal preference relations.

Many machine learning algorithms for multi-class classification aggregate several binary classifiers to compose a decision rule (see e.g. Dietterich & Bakiri (1995), Platt *et al.* (2000), Allwein *et al.* (2000), Crammer & Singer (2002), Hsu & Lin (2002). In the popular one-versus-one ensemble (Hastie & Tibshirani 1998, Fürnkranz 2002) a classifier is trained on each pair of

categories, but do we really need such a complex model for every multi-class classification task? More specifically, a one-versus-one ensemble is a very flexible model, since it can represent the trends in the data for each pair of categories separately. Because of that, it is also a complex model, being difficult to fine-tune with a limited amount of data. On the other hand, a one-versus-all ensemble has substantially less free parameters, decreasing the chance of overfitting, but increasing the chance of underfitting the data.

One might agree that different multi-class classification schemes have a different degree of complexity, but no consensus has been reached on which one to prefer (Rifkin & Klautau 2004). In this work we go one step further and investigate whether a one-versus-one multi-class model can be simplified to a ranking model (or an ordinal regression model if we consider accuracy as performance measure). We start from the assumption that the optimal complexity of a multi-class model is problem-specific. Reducing a one-versus-one ensemble to a ranking model can be seen as a quite drastic application of the bias-variance trade-off: a one-versus-one classification scheme is a complex model, resulting in a low bias and a high variance of the performance, while an ordinal regression model is a much simpler model, manifesting a high bias, but a low variance. So, we do not claim that a one-versus-one scheme can always be reduced to a ranking model. We rather look for necessary and sufficient conditions that allow for such a reduction.

Similar to previous chapters, we will not compare both types of models in terms of accuracy. We will only evaluate the ranking function that underlies these models. As discussed more elaborately in Chapter 2, this means for a one-versus-one ensemble that we have a ranking function for each pair of categories, as for example done in one-versus-one support vector machines to generate pairwise probability estimates (Wu *et al.* 2004). For an ordinal regression model, this means that we have a single ranking function from which the class boundaries are derived by putting some thresholds as discussed in Chapter 3.

6.4.1 Strict ranking representability

So, let us pose the problem setting a bit more formally. Given the notations introduced in Chapter 3, D_{kl} will denote the subset of D containing all objects of categories $\bar{y}_k$ and $\bar{y}_l$ and $\bar{\mathcal{F}}$ will represent a one-versus-one multi-class classification model as a set $\bar{\mathcal{F}}$ of $r(r-1)/2$ pairwise ranking functions f_{kl} . When can we reduce this model to a single ranking function that gives a better performance on unknown test data and when do we need

Algorithm 6 Verification of strict ranking representability

```

construct subsets  $D_{kl}$  from  $D$  for  $1 \leq k < l \leq r$ 
sort  $D_{kl}$  according to increasing  $f_{kl}$ 
 $D' = \emptyset$ 
while  $\exists k, l : D_{kl} \neq \emptyset$  do
     $V = \emptyset$ 
    for  $1 \leq k < l \leq r$  do
        copy the last element of  $D_{kl}$  into  $V$ 
    end for
     $(\mathbf{x}_{\max}, y_{\max}) \leftarrow$  data object with the highest frequency in  $V$ 
    if  $(\mathbf{x}_{\max}, y_{\max})$  occurs  $r - 1$  times in  $V$  then
        for  $1 \leq k < l \leq r : y_{\max} \in \{\bar{y}_k, \bar{y}_l\}$  do
            remove the last element of  $D_{kl}$ 
        end for
        insert  $(\mathbf{x}_{\max}, y_{\max})$  in front of  $D'$ 
    else
        throw error
    end if
end while
output:  $D'$  as the data set sorted according to a single ranking function.

```

the more complex model? Or, equivalently, can we simplify the one-versus-one model to a single ranking model without decreasing the error on training data? Having in mind the bias-variance trade-off, it would be appropriate to prefer the single ranking model over the one-versus-one scheme if the training error does not increase. In that case, the former model is complex enough in order to fit the data well in spite of having a lower variance over different data sets. In its most strict form, we can define ranking representability of a one-versus-one classification scheme as follows.

Definition 6.44. Let $D \subset \mathcal{X} \times \mathcal{Y}$. We call a set $\bar{\mathcal{F}}$ of pairwise ranking functions strictly ranking representable on D if there exists a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that for all $1 \leq k < l \leq r$ and all $(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) \in D_{kl}$

$$f_{kl}(\mathbf{x}_i) < f_{kl}(\mathbf{x}_j) \Leftrightarrow f(\mathbf{x}_i) < f(\mathbf{x}_j). \quad (6.19)$$

Strict ranking representability of a set of pairwise ranking functions can be easily verified with a simple algorithm that is linear in the size of the data set. Let us for example consider the pseudo-code listed in Algorithm 6. The algorithm tries to find a single ranking that is consistent with all pairwise

rankings. If such a ranking exists, then this (unique) ranking is returned as output. If the set $\overline{\mathcal{F}}$ is not strictly ranking representable, then an error is thrown.

In a one-versus-one multi-class setting, only the subset D_{kl} is used to estimate the function f_{kl} . After sorting D_{kl} according to f_{kl} , strict ranking representability is checked by a single iteration over all subsets. The main observation is that for a strictly ranking representable set $\overline{\mathcal{F}}$ we can always find a data object $(\mathbf{x}_{\max}, y_{\max})$ appearing on top of exactly $r - 1$ pairwise rankings. Indeed, this is a necessary condition for strict ranking representability, since the data object appearing on top of the single ranking must appear on top in all pairwise rankings in which it occurs. Remark that any data object can occur in at most $r - 1$ pairwise rankings. Subsequently, the algorithm removes the data object $(\mathbf{x}_{\max}, y_{\max})$ from all pairwise rankings maintained in the ordered subsets D_{kl} and, recursively, the same reasoning can then be adopted to the remaining data objects: again one data object must appear on top of $r - 1$ pairwise rankings. If the algorithm succeeds in detecting such a data object in any iteration, then the single ranking of D is returned at the end. Otherwise, an error is thrown and the algorithm stops prematurely.

For purposes that will become clear further, we will now propose a graphical reformulation of strict ranking representability.

Definition 6.45. Let $D \subset \mathcal{X} \times \mathcal{Y}$ and let $\overline{\mathcal{F}}$ be a set of pairwise ranking functions. We define the graph $G_{\text{strict}}(\overline{\mathcal{F}}, D) = (V, E)$ of $\overline{\mathcal{F}}$ and D such that each node v_i in V is associated with one data object $(\mathbf{x}_i, y_i)$ in D and

$$(\exists k, l : f_{kl}(\mathbf{x}_i) < f_{kl}(\mathbf{x}_j)) \Leftrightarrow (v_i, v_j) \in E, \quad (6.20)$$

with $y_i, y_j \in \{\overline{y}_k, \overline{y}_l\}$ and $f_{lk} = -f_{kl}$ for $1 \leq k < l \leq r$.

Proposition 6.46. Let $D \subset \mathcal{X} \times \mathcal{Y}$. A set $\overline{\mathcal{F}}$ of pairwise ranking functions is strictly ranking representable on D if and only if $G_{\text{strict}}(\overline{\mathcal{F}}, D)$ is a directed acyclic graph (DAG).

Proof. Part 1: strict ranking representability $\Rightarrow$ DAG

For a strictly ranking representable set $\overline{\mathcal{F}}$, we can rewrite (6.20) as

$$f(\mathbf{x}_i) < f(\mathbf{x}_j) \Leftrightarrow (v_i, v_j) \in E.$$

From this follows that $G_{\text{strict}}(\overline{\mathcal{F}}, D)$ is acyclic. Otherwise, if the graph would contain a cycle of length s such that

$$v_{i_1} \rightarrow v_{i_2} \rightarrow \dots \rightarrow v_{i_s} \rightarrow v_{i_1},$$

Algorithm 7 Topological sorting

```

Input: DAG  $G = (V, E)$ 
 $V' = \emptyset$ 
 $V^* \leftarrow$  set of nodes from  $V$  with no incoming edges in  $E$ 
while  $V^* \neq \emptyset$  do
     $v \leftarrow$  randomly remove a node from  $V^*$ 
    append  $v$  to  $V'$ 
    for each node  $v^*$  with an edge  $e$  from  $v$  to  $v^*$  do
        remove edge  $e$  from  $E$ 
        if  $v^*$  has no other incoming edges then
            add  $v^*$  to  $V^*$ 
        end if
    end for
end while
Output:  $V'$  as the linear ordering of  $V$ 

```

then we would find

$$f(\mathbf{x}_{i_1}) < f(\mathbf{x}_{i_2}) < \dots < f(\mathbf{x}_{i_s}) < f(\mathbf{x}_{i_1}).$$

Part 2: DAG $\Rightarrow$ strict ranking representability

For a DAG the following well-known property holds (see e.g. (Schrijver 2003)[chpt. 14])

Property Let $G = (V, E)$ be a directed acyclic graph, then V contains at least one node without incoming edges.

Applying this property to $G_{strict}(\overline{\mathcal{F}}, D)$ immediately yields strict ranking representability. To this end, we will compose a single ranking function for D by ordering the nodes in V . This can be done by topological sorting, a simple technique for computing a linear ordering of the nodes of a DAG (Kahn 1962). The pseudo-code is listed in Algorithm 7. No further explanation is needed to substantiate that the obtained ranking of V (and D) satisfies (6.19). Consequently, topological sorting results in a unique linear ordering of V for the graph $G_{strict}(\overline{\mathcal{F}}, D)$. $\square$

Example 6.47. Let us illustrate strict ranking representability with an example. In Figure 6.2 (a) a data set consisting of 9 objects is given such that $n_1 = n_2 = n_3$ resulting in a three-class classification task. In Figure 6.2 (b)

a triplet of three pairwise rankings is represented, one for each pair of categories. One can easily verify that this triplet is strictly ranking representable. The global ranking of the data is given in Figure 6.2 (c) and the acyclic graph $G_{strict}(\overline{\mathcal{F}}, D)$ is given in Figure 6.3. On the other hand, the triplet of pairwise rankings given in Figure 6.2 (d) is not strictly ranking representable. The failure immediately arises in the first step of Algorithm 6, since a different object appears on top of all pairwise rankings. This results in a cycle in the corresponding graph, as visualized in Figure 6.4.

6.4.2 AUC ranking representability

It goes without saying that strict ranking representability has a very limited applicability to reduce one-versus-one multi-class schemes, since the condition is too strong to be satisfied for realistic data samples. When fitting $r(r-1)/2$ functions to the data in a multi-class setting, it is unrealistic to think that all these functions will impose a consistent ranking, i.e. a ranking satisfying (6.19). Yet, is it really necessary to require strict ranking representability in order to exchange a one-versus-one model for a single ranking model? Of course not, since we are interested in a good performance on independent test data. Therefore, demanding that a single ranking gives exactly the same result on training data as a one-versus-one scheme might be too strong as condition. An obvious relaxation could exist in requiring that a single ranking model yields the same *performance* on training instead of requiring the same results. This makes a subtle difference since it is now allowed that both models make errors on different data objects, as long as the total error of both models is similar. As claimed above, the single ranking model should attain better results on independent test data when taking the bias-variance trade-off into consideration. For the sake of convenience we forget here that the complexity of individual ranking functions might differ for f_{kl} and f in this kind of reasoning, although it can happen that a single ranking model gives rise to a higher variance on independent test data than a pairwise model. For example, if we compare a one-versus-one scheme with linear base classifiers to a non-linear ordinal regression model, then the latter one is not necessarily the least complex model (as illustrated in Figure 2.7). Consequently, we will put forward the additional assumption that the base pairwise rankings in a one-versus-one scheme and the single ranking function underlying an ordinal regression model have a similar degree of complexity. In other words, if we try to reduce a one-versus-one ensemble to a less complex ranking model, then we should have used non-linear base

Data set									
i	1	2	3	4	5	6	7	8	9
y_i	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_3$

(a) Hypothetical data set

i						
f_{12}	1	2	4	5	6	3
f_{23}	4	5	7	8	9	6
f_{13}	1	2	7	8	9	3

(b) A triplet of strictly ranking representable rankings

i									
f_1	1	2	4	5	7	8	9	6	3

(c) The ranking function f generated by (f_{12}, f_{23}, f_{13})

i						
f'_{12}	1	2	4	5	6	3
f'_{23}	4	5	7	8	9	6
f'_{13}	1	2	7	8	3	9

(d) A triplet of not strictly ranking representable rankings

Figure 6.2: An example to illustrate strict ranking representability. In (a) a hypothetical data set is given. In (b) a strictly ranking representable one-versus-one model on that data set is given. The resulting global ranking is given in (c). Finally, in (d) a one-versus-one ensemble $(f'_{12}, f'_{23}, f'_{13})$ that is not strictly ranking representable is obtained by permuting only two elements from the pairwise rankings in (b).

learners for the ensemble in the first place, since the derived ranking model will probably be non-linear as well.

The performance measure that we will consider is the pairwise AUC as defined in Chapter 3, leading to the concept of AUC ranking representability. So, we can restate the question that we aim to answer as follows: under which conditions can we represent the set of pairwise AUCs defined on $r(r-1)/2$ ranking functions as a new set of pairwise AUCs, but now defined on a single ranking? Remark that we already postulated the idea of computing pairwise AUCs on a single ranking function in Section 3.3.2.b. There we considered multi-class classification and ranking as two different supervised learning problems. Here we investigate the conditions under which both problems coincide. Given the definitions of $\hat{A}_{kl}(\overline{\mathcal{F}}, D)$ and $\hat{A}_{kl}(f, D)$ in Chapter 3, let us first introduce a more formal definition of AUC ranking representability.

Definition 6.48. *Let $D \subset \mathcal{X} \times \mathcal{Y}$. We call a set $\overline{\mathcal{F}}$ of pairwise ranking functions AUC ranking representable on D if there exists a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that for $1 \leq k < l \leq r$*

$$\hat{A}_{kl}(\overline{\mathcal{F}}, D) = \hat{A}_{kl}(f, D). \quad (6.21)$$

The following proposition naturally follows from the definitions.

Proposition 6.49. *Strict ranking representability implies AUC ranking representability.*

Unlike strict ranking representability, AUC ranking representability does not result in a unique ranking function. To illustrate this, let us consider a three-class data set with 9 data objects such that $n_1 = n_2 = n_3 = 3$ and suppose we have a one-versus-one ensemble $\overline{\mathcal{F}} = \{f_{12}, f_{23}, f_{13}\}$ such that

$$\hat{A}_{12}(\overline{\mathcal{F}}, D) = \hat{A}_{23}(\overline{\mathcal{F}}, D) = \hat{A}_{13}(\overline{\mathcal{F}}, D) = \frac{6}{9}.$$

The set of pairwise rankings is AUC ranking representable. Moreover, we can find three different ranking functions satisfying (6.21). They are given in Figure 6.5. Remark that rankings resulting from permutations of the ranks of data objects with the same label are not considered as different rankings, because all these rankings always yield the same AUC. For example, in the three rankings of Figure 6.5, we could find more rankings by exchanging the data object $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$, but the resulting rankings yield the AUC and only one of the permutations is considered.

Data set									
i	1	2	3	4	5	6	7	8	9
y_i	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_1$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_2$	$\bar{y}_3$	$\bar{y}_3$	$\bar{y}_3$

(a) Hypothetical data set.

i									
f_1	1	2	4	5	7	8	9	6	3
f_2	1	4	7	2	5	8	3	6	9
f_3	7	4	1	2	3	5	6	8	9

(b) Hypothetical rankings imposed by f_1 , f_2 and f_3 .

Figure 6.5: An example to illustrate that the ranking obtained from a set of AUC ranking representable ranking functions is not unique. Here the labels of a hypothetical data set and the rankings imposed by three ranking functions f_1 , f_2 and f_3 are shown. For these rankings the same pairwise AUC is obtained, i.e. $\hat{A}_{kl}(f_1, D) = \hat{A}_{kl}(f_2, D) = \hat{A}_{kl}(f_3, D)$ with $1 \leq k < l \leq 3$. Two rankings resulting from permutations of two objects of the same category are considered as identical.

Without careful consideration, one could think of modifying Algorithm 6 to verify AUC ranking representability of a one-versus-one scheme, leading to the pseudo-code listed in Algorithm 8. For AUC ranking representability, we now look how many times a particular category (instead of a particular data object) appears on top of the pairwise rankings. If a category appears on top of exactly $r - 1$ pairwise rankings, then we can put that category on top of a single ranking function. However, this property is only a sufficient condition and, as a consequence, the intuitive algorithm is incorrect.

Remark 6.50. Let $D \subset \mathcal{X} \times \mathcal{Y}$. Sets $\bar{\mathcal{F}}$ of AUC ranking representable pairwise rankings exist for which Algorithm 8 throws an error.

Proof. Since AUC ranking representability does not result in a unique global ranking function f , the pairwise rankings f_{kl} can be composed from different global ranking functions f . Let us illustrate this with the rankings given in Figure 6.5. Suppose we derive the pairwise rankings f_{12} and f_{23} from the global ranking function f_1 and the pairwise ranking function f_{13} from f_2 . This yields the following pairwise rankings:

Algorithm 8 (Incorrect) verification of AUC ranking representability

```

construct subsets  $D_{kl}$  from  $D$  for  $1 \leq k < l \leq r$ 
sort  $D_{kl}$  according to increasing  $f_{kl}$ 
 $D' = \emptyset$ 
while  $\exists k, l : D_{kl} \neq \emptyset$  do
     $V = \emptyset$ 
    for  $1 \leq k < l \leq r$  do
        copy the last element of  $D_{kl}$  into  $V$ 
    end for
     $\bar{y}_{\max} \leftarrow$  category with the highest frequency in  $V$ 
    if  $r - 1$  data objects in  $V$  belong to category  $\bar{y}_{\max}$  then
        for  $1 \leq k < l \leq r : \bar{y}_{\max} \in \{\bar{y}_k, \bar{y}_l\}$  do
            remove the last element of  $D_{kl}$ 
        end for
        randomly remove one data object  $(\mathbf{x}, y)$  in  $D$  for which  $y = \bar{y}_{\max}$ 
        and insert it in front of  $D'$ 
    else
        throw error
    end if
end while
Output:  $D'$  as the data set sorted according to a single ranking function.

```

	i					
f_{12}	1	2	4	5	6	3
f_{23}	4	5	7	8	9	6
f_{13}	1	7	2	8	3	9

We have shown before that this set of pairwise rankings is AUC ranking representable, but Algorithm 8 will throw an error in its first iteration. $\square$

6.4.3 Dice transitivity

Given the extension $f_{lk} = -f_{kl}$, the pairwise AUCs can be interpreted as a special type of reciprocal relation, since one can easily verify that

$$\hat{A}_{kl}(\overline{\mathcal{F}}, D) = 1 - \hat{A}_{lk}(\overline{\mathcal{F}}, D).$$

Instead of a reciprocal relation defined on data objects, we now have a reciprocal relation defined on sets of objects, in which each set contains all

elements in D with an identical label. In recent work, De Schuymer *et al.* (2003) investigated the transitivity of ranking representable AUCs within the framework of cycle-transitivity. The main results given in this section stem from that work. They are reformulated in terms of ROC analysis, since the authors present their results in a totally different context, namely the context of dice models. So, let us explain dice models in a bit more detail, while keeping the notation of this dissertation in mind.

Dice models are a specific type of reciprocal relations that originate from game theory. In an ordinary dice game between two players, one usually assumes that the players dispose of three fair dice with 6 faces. Furthermore, the faces of the dice are blank and Player 1 is asked to write numbers from $\mathbb{N}[1, \dots, 18]$ to the faces of the three dice such that each number occurs only once. For both players, the goal of the game is to pick one of the three dice such that the chance of winning is as high as possible. A person wins the bet when on top of his/her die the highest number is observed. Since Player 1 has assigned the numbers to the dice, it makes sense to let Player 2 choose a die first. Player 1 then chooses one of the two remaining dice. In such a game, one can wonder who has the highest probability of winning. Surprisingly, despite having to choose last, it can be shown that Player 1 will always win from Player 2 in the long run, when he/she assigns numbers to the dice in a clever way.

Let us put things a little bit more general to illustrate the resemblance with ROC analysis. We now have in general r dice with labels $\bar{y}_1, \dots, \bar{y}_r$ and we assume that die $\bar{y}_k$ has n_k faces, so different dice can have a different number of faces. Player 1 assigns now unique numbers from $\mathbb{N}[1, n_1 + n_2 + n_3]$ to the dice and both players choose a die. Assume that Player 1 picks die $\bar{y}_k$ and Player 2 takes die $\bar{y}_l$, then we need to compute the probability that $\bar{y}_k$ wins from $\bar{y}_l$ to know the player winning in the long run. From the numbers assigned to the dice, we retrieve a total ranking f of the faces and we can compute the probability of winning by comparing all faces of die $\bar{y}_k$ and $\bar{y}_l$ in pairs. Reformulating everything in terms of ROC analysis, the probability that die $\bar{y}_l$ wins from die $\bar{y}_k$ is precisely the pairwise AUC $\hat{A}_{kl}(f, D)$. Considering all pairs of dice, we have a set of ranking representable AUCs, since a single ranking defines the order over the faces of all dice.

Example 6.51. Let us consider the ranking given in Table 5.1, for which the pairwise AUCs were given by (5.1). Let us now associate a die with all three categories in this three-class ordinal regression problem, then we have

to write the following numbers on the three dice:

$$\begin{aligned}\text{die}_{\bar{y}_1} &= \{2, 3, 4, 15, 16, 18\}, \\ \text{die}_{\bar{y}_2} &= \{5, 6, 7, 8, 9, 17\}, \\ \text{die}_{\bar{y}_3} &= \{1, 10, 11, 12, 13, 14\}.\end{aligned}$$

One can easily verify that the probability that die $\bar{y}_l$ wins from die $\bar{y}_k$ is $\hat{A}_{kl}(f, D)$. As discussed in Chapter 5, the pairwise AUCs are not weakly stochastically transitive in this example.

In Chapter 5, we already illustrated that a collection of ranking representable AUCs is not necessarily weakly stochastically transitive. Here we reformulate the results given in (De Schuymer *et al.* 2003) in terms of ROC measures. It is shown that ranking representable AUCs possess a specific form of cycle-transitivity, namely dice transitivity.

Proposition 6.52. (De Schuymer *et al.* 2003) *Let $D \subset \mathcal{Y} \times \mathcal{Y}$. The pairwise AUCs generated by a set $\bar{\mathcal{F}}$ of AUC ranking representable rankings satisfy the triangle inequality.*

Proof. The proof has been reformulated from De Schuymer *et al.* (2003) for its interesting link with the volume under the ROC surface. We need to show that for any triplet $(\bar{y}_i, \bar{y}_j, \bar{y}_k)$ of categories and any sample D , the resulting reciprocal relations $\hat{A}_{ij}$, $\hat{A}_{jk}$ and $\hat{A}_{ki}$ satisfy the double inequality

$$0 \leq \alpha_{ijk} + \beta_{ijk} + \gamma_{ijk} - 1 \leq 1.$$

Let us define

$$\hat{U}_{ijk} = \frac{1}{n_i n_j n_k} \sum_{a,b,c: y_a=\bar{y}_i, y_b=\bar{y}_j, y_c=\bar{y}_k} I^*(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c),$$

in which we used the shorthand notation:

$$\begin{aligned}I^*(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) &= I_{f(\mathbf{x}_a) < f(\mathbf{x}_b) < f(\mathbf{x}_c)} + \frac{1}{2} I_{f(\mathbf{x}_a) < f(\mathbf{x}_b) = f(\mathbf{x}_c)} \\ &\quad + \frac{1}{2} I_{f(\mathbf{x}_a) = f(\mathbf{x}_b) < f(\mathbf{x}_c)} + \frac{1}{6} I_{f(\mathbf{x}_a) = f(\mathbf{x}_b) = f(\mathbf{x}_c)}.\end{aligned}$$

Remark that $\hat{U}_{ijk}$ represents the volume under the ROC surface if we impose the order $\bar{y}_i <_{\mathcal{Y}} \bar{y}_j <_{\mathcal{Y}} \bar{y}_k$ on $\mathcal{Y}$. It follows that:

$$\hat{U}_{ijk} + \hat{U}_{jki} + \hat{U}_{kij} + \hat{U}_{ikj} + \hat{U}_{kji} + \hat{U}_{jik} = 1.$$

Moreover, we also have that:

$$\begin{aligned}\hat{A}_{ij} &= \hat{U}_{ijk} + \hat{U}_{ikj} + \hat{U}_{kij}, \\ \hat{A}_{jk} &= \hat{U}_{ijk} + \hat{U}_{jik} + \hat{U}_{jki}, \\ \hat{A}_{ki} &= \hat{U}_{kij} + \hat{U}_{jki} + \hat{U}_{kji}.\end{aligned}$$

Consequently, we find:

$$\begin{aligned}\alpha_{ijk} + \beta_{ijk} + \gamma_{ijk} - 1 &= \hat{A}_{ij} + \hat{A}_{jk} + \hat{A}_{ki} - 1, \\ &= \hat{U}_{ijk} + \hat{U}_{jki} + \hat{U}_{kij}.\end{aligned}$$

This quantity lies in the interval $[0, 1]$. □

Thus, the pairwise AUCs have to satisfy the triangle inequality so that the corresponding one-versus-one model is AUC ranking representable. An even stronger necessary condition can be found.

Proposition 6.53. *(De Schuymer et al. 2003) Let $D \subset \mathcal{X} \times \mathcal{Y}$. The pairwise AUCs generated by a set $\overline{\mathcal{F}}$ of AUC ranking representable rankings are dice transitive.*

6.4.4 Sufficient conditions for AUC ranking representability

The pairwise AUCs generated from an AUC ranking representable one-versus-one multi-class classification model exhibit a particular form of transitivity, stronger than the triangle inequality, but weaker than $T_{\mathbf{P}}$ -transitivity. So, dice transitivity gives rise to a necessary condition for AUC ranking representability, but is it also a sufficient condition? The answer is definitely negative, since even much stronger types of transitivity not necessarily lead to AUC ranking representability. To illustrate this, we present a graph-theoretic reformulation of AUC ranking representability in the next section, but first we show that dice transitivity becomes a sufficient condition in the three class case for a relaxation of AUC ranking representability.

6.4.4.a Semi AUC ranking representability

In particular, the following proposition has been proven for dice models.

Proposition 6.54. *(De Schuymer et al. 2003) Every three-dimensional dice-transitive reciprocal relation with rational elements can be generated with a dice model.*

However, this does not immediately result in a sufficient condition for AUC ranking representability, because our definition of an AUC ranking representable one-versus-one ensemble is in some narrower than the definition of a dice model. We first present the relaxation of AUC ranking representability for which dice-transitivity becomes a necessary and sufficient condition and then illustrate with an example why this relaxation is crucial.

Definition 6.55. Let $D \subset \mathcal{X} \times \mathcal{Y}$. We call a set $\overline{\mathcal{F}}$ of pairwise ranking functions semi AUC ranking representable on D if there exists a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ and a second data set $D' \subset \mathcal{X} \times \mathcal{Y}$ such that for all $1 \leq k < l \leq r$

$$\hat{A}_{kl}(\overline{\mathcal{F}}, D) = \hat{A}_{kl}(f, D').$$

Corollary 6.56. Let $D \subset \mathcal{X} \times \mathcal{Y}$. A triplet $\overline{\mathcal{F}} = \{f_{12}, f_{23}, f_{13}\}$ of pairwise ranking functions is semi AUC ranking representable on D if and only if the corresponding triplet of AUCs is dice-transitive.

Example 6.57. Let us consider the following strongly stochastically transitive triplet of pairwise AUCs:

$$\hat{A}_{12} = \frac{5}{9}, \quad \hat{A}_{23} = \frac{5}{9}, \quad \hat{A}_{13} = \frac{9}{9}.$$

Such a triplet of AUCs can be generated by data sets of different size, but the minimal cardinality is given by $n_1 = n_2 = n_3$. Using graph-theoretic reformulations, we will easily show in the next section that the one-versus-one model is not AUC ranking representable for this minimal cardinality. However, De Schuymer *et al.* (2003) do not put any restriction on the number of faces a die can have (i.e. the cardinality of the data set). In general, they prove that a three-dimensional dice-transitive reciprocal relation can always be represented by a dice model with at least $3 \times n_1 \times n_2 \times n_3$ faces. Often fewer faces are needed, as for the above triplet of AUCs. Let us for example consider a three-class data set with 27 data objects such that $n_1 = n_2 = n_3 = 9$:

Data set									
i	1	...	9	10	...	18	19	...	27
y_i	$\bar{y}_1$	...	$\bar{y}_1$	$\bar{y}_2$	...	$\bar{y}_2$	$\bar{y}_3$	...	$\bar{y}_3$

The above triplet of AUCs is generated by the following ranking function:

i					
f	10–13	1–9	14	19–27	15–18

Unfortunately, semi AUC ranking representability is quite useless in practice, since one usually does not have a second data set a with higher cardinality at one's disposal. We have to keep the cardinality of the data set fixed, when reducing a one-versus-one model to a single ranking model. Therefore, we will present another way to find a verifiable sufficient condition by using graph-theoretic concepts.

6.4.4.b Graphical reformulations of pairwise AUCs

Definition 6.58. Let $D \subset \mathcal{X} \times \mathcal{Y}$ and let $\overline{\mathcal{F}}$ be a set of pairwise ranking functions. We define $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ as a set of complete directed graphs $G = (V, E)$ for which the following three properties hold:

1. Each node v_i in V is associated with one data object $(\mathbf{x}_i, y_i)$ in D .
2. No cycles occur in the subsets $V_k = \{v_i \in V \mid y_i = \overline{y}_k\}$.
3. For $1 \leq k < l \leq r$:

$$\hat{A}_{kl}(\overline{\mathcal{F}}, D) = \frac{|\{(v_i, v_j) \in E \mid y_i = \overline{y}_k \wedge y_j = \overline{y}_l\}|}{n_k n_l}. \quad (6.22)$$

Let us try to express this definition in a less formal way. We are in essence looking for all graphs $G = (V, E)$ in which we associate one data object from the data set with a node such that we obtain r subsets $V_1, \dots, V_r$ for r categories (similar to the graphs introduced in previous chapters). We require in addition that the nodes within each subset are ordered (which results in an acyclic subgraph for these subsets), and that the fraction of edges from subset V_k to V_l corresponds to $\hat{A}_{kl}(\overline{\mathcal{F}}, D)$. We only consider complete directed graphs in this way. Remark that a complete directed graph is a graph in which each pair of nodes is connected by exactly one (directed) edge. So, $(v, v') \in E$ implies $(v', v) \notin E$.

It follows directly from the definition that $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ cannot be empty. Its cardinality will usually be greater than 1 since different graphs satisfying (6.22) will be found for a given $\overline{\mathcal{F}}$ and D . In the following proposition, AUC ranking representability is reformulated in terms of these graphs.

Definition 6.59. We introduce $\mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$ as the subset of $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ containing only acyclic graphs.

Proposition 6.60. *Let $D \subset \mathcal{X} \times \mathcal{Y}$. A set $\overline{\mathcal{F}}$ of pairwise ranking functions is AUC ranking representable on D if and only if $\mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$ is not empty.*

Proof. Part 1: AUC ranking representability $\Rightarrow$ DAG

Let the set $\overline{\mathcal{F}}$ of pairwise ranking functions be AUC ranking representable. We need to find a DAG satisfying (6.22). We will describe a way to generate such a graph $G^* = (V^*, E^*)$. Similar to the graph $G_{strict}(\overline{\mathcal{F}}, D)$, we associate with each node v_i in V^* precisely one data object $(\mathbf{x}_i, y_i)$ from D and we define E^* as

$$f(\mathbf{x}_i) < f(\mathbf{x}_j) \Leftrightarrow (v_i, v_j) \in E^*, \quad \forall i, j \in \{1, \dots, n\}$$

Obviously, the graph G^* is acyclic.

Part 2: DAG $\Rightarrow$ AUC ranking representability

Suppose we have a DAG $G_{AUC}(\overline{\mathcal{F}}, D)$. By applying Property 6.4.1 and a topological sorting algorithm, we can find a linear ordering of the nodes and, hence, a single ranking of the data exists such that (6.22) holds. $\square$

We will first use the graph concepts to show that the AUC can take any value as a fraction of the number of data objects.

Proposition 6.61. *Let $D \subset \mathcal{X} \times \mathcal{Y}$ be a binary classification data set with n_1 objects of the negative class and n_2 objects of the positive class. For any rational AUC defined by*

$$\hat{A}_{12}(f, D) = \frac{a}{n_1 n_2},$$

with $a \in \mathbb{N}[0, \dots, n_1 \times n_2]$, there exists a corresponding ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ that generates this AUC.

Proof. We construct a bipartite graph that is consistent with Def. 6.58. This graph is bipartite since we only have two classes here, and it can always be found as we simply need to draw a edges from layer V_1 to V_2 and $n_1 n_2 - a$ edges from V_2 to V_1 . As the graph will contain no cycles, we can find a ranking with a topological sorting algorithm. $\square$

With the help of the graphical reformulation of AUC ranking representability, we can also substantiate that strong stochastic transitivity and even T_M -transitivity of the pairwise AUCs does not necessarily lead to AUC ranking representability. From this we can conclude that dice transitivity, being much weaker than those types, provides only a necessary condition for AUC ranking representability.

Remark 6.62. Not all triplets of pairwise rankings generating a triplet of strongly stochastically transitive AUCs are AUC ranking representable.

Proof. We reconsider the strongly stochastically transitive triplet of pairwise AUCs defined in Example 6.57 and let us try to construct an acyclic graph satisfying (6.22) for this triplet. We must have 4 edges departing from V_2 and arriving in V_1 and we must have 4 edge departing from V_3 and arriving in V_2 . Consequently, two V_2 -nodes have incoming V_3 -edges and two V_2 -nodes have outgoing edges to V_1 -nodes. This implies that at least one V_2 -node has both edges incoming from V_3 and outgoing edges to V_1 , which means that we have a path from at least one V_3 -node to one V_1 -node. This pair of nodes give rise to a cycle since every node from V_1 has also an outgoing edge to every V_3 -node. $\square$

Remark 6.63. Not all triplets of pairwise rankings generating a triplet of T_M -transitive pairwise AUCs are AUC ranking representable.

Proof. Let us consider the following T_M -transitive collection of pairwise AUCs:

$$\begin{aligned}\hat{A}_{12} = \alpha &= \frac{1}{12} = \frac{3}{36}, \\ \hat{A}_{23} = \beta &= \frac{7}{9} = \frac{32}{36}, \\ \hat{A}_{31} = \gamma &= \frac{11}{12} = \frac{33}{36},\end{aligned}$$

such that $n_1 = 4$ and $n_2 = n_3 = 3$. This triplet of AUCs is T_M -transitive since

$$\gamma = 1 - \alpha. \quad (6.23)$$

If we try to construct a graph for this triplet of AUCs, then we must have 11 edges departing from V_3 and arriving in V_1 and we must have 1 edge departing from V_1 and arriving in V_2 . Consequently, the V_1 -node from which this last edge departs, has at least two incoming edges from V_3 -nodes. The V_2 -node in which this edge arrives cannot be connected to the two V_3 -nodes, if we want the triplet of AUCs to be AUC ranking representable. Otherwise we would have a cycle. So, 7 is an upper bound on the number of edges departing from V_2 and arriving in V_3 , but in this example we need 8 edges. $\square$

6.4.4.c AUC transitivity

Using the graph-theoretic concepts introduced above, we have a sufficient condition for AUC ranking representability. Nevertheless, this condition cannot be verified for large datasets, since the cardinality of $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ exponentially increases with the size of D . Similar to De Schuymer *et al.* (2003), we will further examine the three-class case in order to find a sufficient condition that can be verified more easily. The reason for this restriction is that we will need cycle-transitivity (which is only defined on triplets). The results obtained for three classes can then be further extended to more classes with approximation techniques. We start with introducing a new type of transitivity.

Definition 6.64. An (a^*, s) -split $\mathbf{a}^*$ is an increasing ordered list (or vector) $\mathbf{a}^* = (a_1^*, a_2^*, \dots, a_s^*)$ of s (not necessarily strictly) positive integers summing up to a^* . An (a^*, s, t) -split is an (a^*, s) -split for which each component of $\mathbf{a}^*$ is upper bounded by t . The set of all (a^*, s, t) splits will be denoted $\mathfrak{S}(a^*, s, t)$. We define the dual $\mathbf{b}$ of an (a^*, s, t) -split as the decreasing vector $\mathbf{b}^* = (a_s^*, a_{s-1}^*, \dots, a_1^*)$. The set of all dual (a^*, s, t) -splits will be denoted $\tilde{\mathfrak{S}}(a^*, s, t)$.

Example 6.65. $\mathfrak{S}(10, 4, 3) = \{(1, 3, 3, 3), (2, 2, 3, 3)\}$.
 $\tilde{\mathfrak{S}}(11, 3, 6) = \{(6, 5, 0), (6, 4, 1), (6, 3, 2), (5, 5, 1), (5, 4, 2), (5, 3, 3)\}$.

Definition 6.66. Let $(n_1, \dots, n_r) \in \mathbb{N}^r$ and let

$$\mathcal{U}_{kl} = \{a \in [0, 1] \mid (\exists a^* \in \mathbb{N})(a = \frac{a^*}{n_k n_l})\}.$$

The family of functions $C_{jkl} : \mathcal{U}_{jk} \times \mathcal{U}_{kl} \rightarrow \mathcal{U}_{jl}$ is defined by:

$$C_{jkl}(a, b) = \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \tilde{\mathfrak{S}}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} (a_i^* - a_{i-1}^*) b_i^*,$$

for $j, k, l \in \{1, \dots, r\}$.

C_{jkl} is the solution of an integer quadratic program. To illustrate this, let us rewrite the minimization as:

$$\min_{\mathbf{a}^*, \mathbf{b}^*} \frac{1}{n_j n_l} \sum_{i=1}^{n_k} (a_i^* - a_{i-1}^*) b_i^*$$

Figure 6.6: The family of functions C_{jkl} visualized.

$$\text{subject to } \begin{cases} \sum_{i=1}^{n_k} a_i^* = a^*, \\ \sum_{i=1}^{n_k} b_i^* = b^*, \\ a_i^* \geq a_{i-1}^*, \forall i \in \{1, \dots, n_k\}, \\ b_i^* \leq b_{i-1}^*, \forall i \in \{2, \dots, n_k + 1\}, \\ 0 \leq a_i^* \leq n_j, \forall i \in \{1, \dots, n_k\}, \\ 0 \leq b_i^* \leq n_l, \forall i \in \{1, \dots, n_k\}, \\ a_i^*, b_i^* \in \mathbb{N}, \forall i \in \{1, \dots, n_k\}, \\ a_0^* = 0, b_{n_k+1}^* = 0. \end{cases} \quad (6.24)$$

In Figure 6.6 the family of functions C_{jkl} is visualized for some (small) n_j , n_k and n_l . The function values were computed by exhaustively verifying all feasible solutions of the integer quadratic program, which can be only done for small values of n_j , n_k and n_l .

Example 6.67. Let us consider the situation: $n_j = 3$, $n_k = 4$, $n_l = 3$, $a = 9/12$, $b = 13/20$. The objective is minimized over the following splits:

$$\mathfrak{S}(9, 4, 3) = \{(0, 3, 3, 3), (1, 2, 3, 3), (2, 2, 2, 3)\}.$$

$$\begin{aligned} \mathfrak{S}(13, 4, 5) = \{ & (5, 5, 3, 0), (5, 5, 2, 1), (5, 4, 4, 0), (5, 4, 3, 1), \\ & (5, 4, 2, 2), (5, 3, 3, 2), (4, 4, 4, 1), (4, 4, 3, 2), (4, 3, 3, 3)\}. \end{aligned}$$

The minimum of the objective function is obtained for the splits $\mathbf{a}^* = (2, 2, 2, 3)$ and $\mathbf{b}^* = (4, 3, 3, 3)$ such that $C_{jkl}(a, b) = 11/15$. It turns out that for this example (and many other cases) the minimum can be found without computing the objective function for all splits exhaustively. To understand this, we have to reveal the graph-theoretic interpretation from which the integer quadratic program originates.

Proposition 6.68. $C_{jkl}(\hat{A}_{jk}, \hat{A}_{kl}) =$

$$\min_{G \in \mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)} \frac{|\{(v_a, v_c) \in V_j \times V_l \mid (\exists v_b \in V_k)((v_a, v_b), (v_b, v_c) \in E)\}|}{n_j n_l}.$$

Proof. Before we give the proof, let us first explain in words what this equation means. Given a data set D and a one-versus-one model $\overline{\mathcal{F}}$, we examine all couples of nodes $(v_a, v_c) \in V_j \times V_l$ in all graphs $G \in \mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$. The proposition states that C_{jkl} equals the minimal number of such couples connected by a path passing through a node of layer V_k over all these graphs. For any graph $G \in \mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$, we have that a^* represents the number of edges departing from a node v_a of subset V_j and ending in a node v_b of subset V_k . Similarly, b^* represents the number of edges departing from a node v_b of subset V_k and arriving in a node v_c of subset V_l . Because the three subgraphs containing solely nodes from the layers V_j , V_k and V_l cannot have cycles, we can put an order on the nodes of these layers. Let us denote $v_i^{(j)}$ as the i -th member of V_j such that we obtain the following chains:

$$\begin{aligned} v_1^{(j)} &\rightarrow v_2^{(j)} \rightarrow \dots \rightarrow v_{n_j}^{(j)}, \\ v_1^{(k)} &\rightarrow v_2^{(k)} \rightarrow \dots \rightarrow v_{n_k}^{(k)}, \\ v_1^{(l)} &\rightarrow v_2^{(l)} \rightarrow \dots \rightarrow v_{n_l}^{(l)}. \end{aligned}$$

Each of the n_k nodes of layer V_k will have a number of incoming edges from layer V_j . Let a_i^* denote the number of incoming edges from V_j for node $v_i^{(k)}$, then the following equation must be satisfied:

$$\sum_{i=1}^{n_k} a_i^* = a^*,$$

which explains why it appears as a constraint in the integer quadratic program (6.24). Since we conduct a minimization over acyclic graphs, the constraint

$$a_i^* \geq a_{i-1}^*, \forall i \in \{2, \dots, n_k\}, \quad (6.25)$$

also has to be included into (6.24). Otherwise a cycle would be induced in the subgraph containing the nodes and edges from layers V_j and V_k . We additionally introduce $a_0^* = 0$ for compactness of the representation. This extension will later allow for a simpler notation and it makes that the constraint now holds for all $i \in \{1, \dots, n_k\}$.

In addition, let b_i^* denote the number of outgoing edges from $v_i^{(k)}$ to any node of V_l , then we find the following two constraints

$$\sum_{i=1}^{n_k} b_i^* = b^*,$$

$$b_i^* \leq b_{i-1}^*, \forall i \in \{2, \dots, n_k\},$$

in which the latter constraint again is required to avoid cycles. Similarly, we also add the extension $b_{n_k+1}^* = 0$ such that constraint holds for all $i \in \{2, \dots, n_k + 1\}$. The other constraints in (6.24) are obvious.

Given this graph-theoretic interpretation of the integer quadratic program, it further needs to be shown that for any graph in $\mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$

$$\sum_{i=1}^{n_k} (a_i^* - a_{i-1}^*) b_i^* \quad (6.26)$$

corresponds to the number of couples in $V_j \times V_l$ for which a path through a node of V_k exists, i.e. the number

$$|\{(v_a, v_c) \in V_j \times V_l \mid (\exists v_b \in V_k)((v_a, v_b), (v_b, v_c) \in E)\}|.$$

The easiest way to understand this resemblance is by looking at (6.26) as an iterative procedure to compute the number of connected couples (v_a, v_c) . We start at the first node $v_1^{(k)}$ of layer V_k for which we have a_1^* incoming edges and b_1^* outgoing edges. So, we have already $a_1^* b_1^*$ connected couples (v_a, v_c) . Subsequently, we examine node $v_2^{(k)}$ for which $a_2^* b_2^*$ connected couples are found, but then we have counted some couples twice. Due to the restriction to acyclic graphs, all incoming edges of $v_1^{(k)}$ are also incoming edges for $v_2^{(k)}$ (this is also implemented by constraint (6.25)). So, we have counted $a_1^* b_2^*$ couples twice. Only $(a_2^* - a_1^*) b_2^*$ couples found in step two lead to new couples of connected nodes. The situation at the other nodes of layer V_k is similar, leading to a recursive representation of (6.26). $\square$

The attentive reader will already have noticed that the graph concepts introduced here bear a strong resemblance with the VUS-graph of Chapter 4. However, there we assumed that the categories in $\mathcal{Y}$ were ordered, resulting in a layered graph in which we looked at all paths from the first layer to the last one in order to compute the volume under the ROC surface. Here we limit our analysis to three-layered graphs (for three-class classification) and we allow that the third layer has outgoing edges to the first one. Furthermore, we look for connected couples of nodes instead of paths and we consider a set of acyclic graphs $\mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$ instead of a single graph. In essence, each of these graphs gives rise to a ranking function and a corresponding VUS-graph. Based on the graph-theoretic interpretation of the family C_{jkl} , we introduce a new type of (cycle) transitivity.

Definition 6.69. A reciprocal relation of pairwise AUCs $\hat{A}_{kl}(\overline{\mathcal{F}}, D)$ is called AUC-transitive if for any $j, k, l \in \{1, \dots, r\}$ it holds that

$$C_{jkl}(\hat{A}_{jk}, \hat{A}_{kl}) \leq \hat{A}_{jl}. \quad (6.27)$$

Proposition 6.70. Let $D \subset \mathcal{X} \times \mathcal{Y}$ and let $\overline{\mathcal{F}} = \{f_{12}, f_{23}, f_{13}\}$ be a triplet of pairwise ranking functions. The corresponding triplet of pairwise AUCs is AUC-transitive if and only if $\mathfrak{H}_{AUC}(\overline{\mathcal{F}}, D)$ is not empty.

Proof. Part 1: AUC transitivity $\Rightarrow$ DAG

We need to show that in the three-class case always a DAG can be found when the pairwise AUCs are AUC-transitive. In order to construct such a DAG, we will adopt a similar strategy as discussed in the proof of the previous proposition. This means that we start with a graph for which the subsets V_j , V_k and V_l are ordered and we ascertain that no cycles can occur in the subgraphs formed by using only two of the three layers (e.g. by imposing a constraint of type (6.25)). A graph satisfying these conditions can always be found, irrespective of the fulfillment of AUC transitivity. For this graph we thus already know that the subgraphs are free of cycles and we only need to check whether cycles occur spanning all three layers. When the pairwise AUCs satisfy AUC transitivity, this property is guaranteed by construction. The easiest way to understand this is by considering the graph-theoretic interpretation as discussed in Prop. 6.68. AUC transitivity guarantees in essence that a situation as described in the proofs of Props. 6.62-6.63 cannot occur.

Part 2: DAG $\Rightarrow$ AUC transitivity

Let us suppose that $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ contains at least one DAG $G = (V, E)$ with three layers V_j, V_k and V_l . Using the same notations as above, we know

that for any choice of $(j, k, l) \in \{1, \dots, r\}^3$ (with $j \neq k \neq l$)

$$\hat{A}_{jl} = \frac{|\{(v_a, v_c) \in V_j \times V_l \mid (\exists v_b \in V_k)((v_a, v_b), (v_b, v_c) \in E)\}|}{n_j n_l}.$$

Otherwise we would have cycles. If the right-hand side is minimized over all graphs in $\mathfrak{H}_{AUC}(\bar{\mathcal{F}}, D)$, then it becomes a fortiori always smaller than the left hand side. $\square$

Corollary 6.71. *Let $D \subset \mathcal{X} \times \mathcal{Y}$. A triplet $\bar{\mathcal{F}} = \{f_{12}, f_{23}, f_{13}\}$ of pairwise ranking functions is AUC ranking representable on D if and only if the corresponding reciprocal relation of AUCs is AUC-transitive.*

Corollary 6.72. *AUC transitivity implies dice transitivity.*

Proof. We have proven the following chain of equivalences and implications:

$$\begin{aligned} & \text{AUC transitivity of pairwise AUCs} \\ & \Leftrightarrow (\text{Proposition 6.70}) \\ & \text{existence of a DAG in } \mathfrak{G}_{AUC}(\bar{\mathcal{F}}, D) \\ & \Leftrightarrow (\text{Proposition 6.60}) \\ & \text{AUC ranking representability of } \bar{\mathcal{F}} \\ & \Leftrightarrow (\text{informal proof in Section 6.4.3}) \\ & \quad \text{dice model} \\ & \Rightarrow (\text{Proposition 6.53}) \\ & \quad \text{dice transitivity.} \end{aligned}$$

$\square$

6.4.5 Properties of C_{jkl}

Proposition 6.73. *Let $(n_1, \dots, n_r) \in \mathbb{N}^r$. The family of functions $C_{jkl} : \mathcal{U}_{jk} \times \mathcal{U}_{kl} \rightarrow \mathcal{U}_{jl}$ as in Definition (6.66) has the following properties:*

1. $\forall j, k, l \in \{1, \dots, r\} : C_{jkl}$ is increasing in both variables.
2. $\forall j, k, l \in \{1, \dots, r\}, \forall (a, b) \in \mathcal{U}_{jk} \times \mathcal{U}_{kl} : C_{jkl}(a, b) = C_{ljk}(b, a)$.
3. $\forall j, k, l \in \{1, \dots, r\}, \forall a \in \mathcal{U}_{jk} : C_{jkl}(a, 0) = 0$.
4. $\forall j, k, l \in \{1, \dots, r\}, \forall a \in \mathcal{U}_{jk} : C_{jkl}(a, 1) = \frac{1}{n_j} \lceil \frac{a^*}{n_k} \rceil$.
5. $\forall j, k, l \in \{1, \dots, r\}, \forall (a, b) \in \mathcal{U}_{jk} \times \mathcal{U}_{kl} :$

$$C_{jkl}(a, b) \leq \frac{1}{n_j n_l} \lceil \frac{a^*}{n_k} \rceil \lceil \frac{b^*}{n_k} \rceil.$$

6. $\forall j, k, l \in \{1, \dots, r\}, \forall (a, b) \in \mathcal{U}_{jk} \times \mathcal{U}_{kl} :$

$$n_j a \in \mathbb{N} \wedge n_l b \in \mathbb{N} \Rightarrow C_{jkl}(a, b) \leq ab.$$

with $a = a^*/n_j n_k$, $b = b^*/n_k n_l$ and $\lceil \cdot \rceil : \mathbb{R} \rightarrow \mathbb{N}$ the ceiling function that retrieves the closest integer above a given real number.

Proof. Property 1:

The objective function of optimization problem (6.24) is an increasing function of a^* and b^* , since $a_i^* \geq a_{i-1}^*$.

Property 2:

Let us define $b_{n_k+1} = 0$. We find:

$$\begin{aligned} C_{jkl}(a, b) &= \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \tilde{\mathfrak{S}}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} a_i^* b_i^* - \sum_{i=1}^{n_k} a_{i-1}^* b_i^*, \\ &= \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \tilde{\mathfrak{S}}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} a_i^* b_i^* - \sum_{i=1}^{n_k} a_i^* b_{i+1}^*, \\ &= \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \tilde{\mathfrak{S}}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} a_i^* (b_i^* - b_{i+1}^*), \\ &= \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \tilde{\mathfrak{S}}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \mathfrak{S}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} a_i^* (b_i^* - b_{i-1}^*), \\ &= C_{lkj}(b, a). \end{aligned}$$

Property 3:

When we fill in 0 for a^* or b^* in optimization problem (6.24), then the solution of the integer quadratic program is 0.

Property 4:

Remember that we still have $a = a^*/n_j n_k$.

$$\begin{aligned}
 C_{jkl}(a, 1) &= \frac{1}{n_j n_l} \min_{\substack{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j) \\ \mathbf{b}^* \in \mathfrak{S}(b^*, n_k, n_l)}} \sum_{i=1}^{n_k} (a_i^* - a_{i-1}^*) n_l, \\
 &= \frac{1}{n_j} \min_{\mathbf{a}^* \in \mathfrak{S}(a^*, n_k, n_j)} \sum_{i=1}^{n_k} (a_i^* - a_{i-1}^*), \\
 &= \frac{1}{n_j} \min_{\mathbf{a}^* \in \mathfrak{S}(a^*, n_j, n_j)} a_{n_k}^*, \\
 &= \frac{1}{n_j} \lceil \frac{a^*}{n_k} \rceil.
 \end{aligned}$$

Property 5:

To understand this property, again the graph-theoretic interpretation of C_{jkl} as established in Proposition 6.68 is needed. Let us consider the following two strategies to draw edges from layer V_j to V_k and from V_k to V_l :

- **Strategy 1:** Assign the edges in such a way that the number of incoming edges from V_j and outgoing edges to V_l is as balanced as possible for all nodes of layer V_k ; this strategy corresponds to choosing the most balanced splits in $\mathfrak{S}(a^*, n_k, n_j)$ and $\mathfrak{S}(b^*, n_k, n_l)$.
- **Strategy 2:** Assign the edges in such a way that the number of incoming edges from V_j and outgoing edges to V_l is as imbalanced as possible for all nodes of layer V_k ; this strategy corresponds to choosing the most imbalanced splits in $\mathfrak{S}(a^*, n_k, n_j)$ and $\mathfrak{S}(b^*, n_k, n_l)$.

Here we only need Strategy 1, the other strategy will be used further on in Prop. 6.76. We show that the quantity

$$\frac{1}{n_j n_l} \lceil \frac{a^*}{n_k} \rceil \lceil \frac{b^*}{n_k} \rceil$$

acts as an upper bound for the objective function in (6.24) when Strategy 1 is followed, and a fortiori it will also be an upper bound for the minimum of the integer quadratic program. Strategy 1 corresponds to a way of drawing edges from V_j to V_k and from V_k to V_l such that the third condition in Def. 6.58 is satisfied, because then at most $\lceil \frac{a^*}{n_k} \rceil$ nodes of layer V_j have outgoing edges to V_k and at most $\lceil \frac{b^*}{n_k} \rceil$ nodes have incoming edges from V_k . So, we have at most $\lceil \frac{a^*}{n_k} \rceil \lceil \frac{b^*}{n_k} \rceil$ connected couples through a node of V_k .

Property 6:

This property immediately follows from Property 5 since in this case $\frac{a^*}{n_k} = \lceil \frac{a^*}{n_k} \rceil$ and $\frac{b^*}{n_k} = \lceil \frac{b^*}{n_k} \rceil$. $\square$

From these properties it follows that C_{jkl} is a family of discrete conjunctors (the functions are only defined over $\mathcal{U}_{jk} \times \mathcal{U}_{kl}$ instead of $[0, 1]^2$). It is interesting to look how these functions behave compared to standard t-norms. In Figure 6.7-6.9, we have compared C_{jkl} to respectively T_L , T_P and T_M for $n_j = n_k = n_l = 10$. One can see that the family of functions C_{jkl} is always greater than T_L with a peak in the upper triangle. In the region close to $(0, 0)$, it is substantially smaller than T_P , while it is similar to T_P in the region close to $(1, 1)$. Thirdly, in almost all parts of the input domain, C_{jkl} is smaller than T_M .

We briefly remark that AUC-transitivity could be translated as a form of cycle-transitivity, because C_{jkl} is a family of conjunctors (however, these conjunctors are not defined on the entire domain $[0, 1]^2$).

6.4.6 The cyclic evaluation of pairwise expected ranking accuracy

Since AUC-transitivity acts as a necessary and sufficient condition for AUC ranking representability, it is able to reveal deeper insights of one-versus-one models, but it is not of great practical value. The functions C_{jkl} are solutions of an integer quadratic program, which is an NP-complete problem, and as a result, the condition can only be exactly verified for small data sets. Instead of focussing on intelligent algorithms to solve the integer quadratic program approximately, we will present another approach to circumvent this computational bottleneck.

From a machine learning point of view, the primary concern is not to know the pairwise coupling of rankings on a finite training set. Rather, we want to find the coupling that characterizes the underlying joint distribution; given the underlying distribution that generates for each of the pairwise rankings two univariate distributions of prediction scores (for each of the two categories), we want to find out whether the distribution of the data allows a single ranking representation of prediction score distributions generated from a pairwise model. This is a direct generalization of AUC ranking representability from an observed data sample to the underlying distribution. The situation is graphically illustrated in Figure 6.10 for a three-class classification problem.

(a) T_L

(b) 3D plot of C_{jkl} . The color indicates $C_{jkl} - T_L$.

(c) Contour plot of $C_{jkl} - T_L$.

Figure 6.7: C_{jkl} compared to T_L with $n_j = n_k = n_l = 10$.

(a) $T_{\mathbf{P}}$

(b) 3D plot of C_{jkl} . The color indicates $C_{jkl} - T_{\mathbf{P}}$.

(c) Contour plot of $C_{jkl} - T_{\mathbf{P}}$.

Figure 6.8: C_{jkl} compared to $T_{\mathbf{P}}$ with $n_j = n_k = n_l = 10$.

(a) T_M

(b) 3D plot of C_{jkl} . The color indicates $C_{jkl} - T_M$.

(c) Contour plot of $C_{jkl} - T_M$.

Figure 6.9: C_{jkl} compared to T_M with $n_j = n_k = n_l = 10$.

(a) Obtained with pairwise rankings f_{12} , f_{13} and f_{23} .

(b) Obtained with a single ranking model.

Figure 6.10: A graphical illustration of AUC ranking representability generalized to distributions. The distributions of output scores obtained with a one-versus-one model are given on top, those obtained with a single ranking model are given at the bottom. Both models generate the same triplets of pairwise ranking accuracies in this example.

It is important to note that we will not require that the distributions of prediction scores generated by a single ranking model have to be identical to those generated by a one-versus-one model, since that again would result in a concept similar to strict ranking representability. We will only enforce that the pairs of prediction score distributions have the same level of separability for both types of models, i.e. we require that the data can be separated similarly with a one-versus-one and a single ranking model. In Chapter 3 we introduced the expected ranking accuracy (ERA) as a measure of separability of two prediction score distributions and we generalized it to multi-class problems, in which A_{kl} was defined for categories $\bar{y}_k$ and $\bar{y}_l$. A_{kl} itself gives rise to a reciprocal relation defined on random data objects (random vectors when $\mathcal{X}$ is a vector space) that represent the different categories. We will use the symbol X_k to denote a random data object with distribution $\mathcal{D}_k$ for category $\bar{y}_k$. This random data object then generates a univariate distribution of prediction scores $f(X_k)$ for a ranking function f . For two categories $\bar{y}_k$ and $\bar{y}_l$, the expected ranking accuracy can be expressed in terms of the joint bivariate cumulative distribution function $\bar{F}_{X_k, X_l}$ of the random data objects X_k and X_l , i.e.

$$A_{kl}(f_{kl}) = \int_{f_{kl}(\mathbf{x}_i) < f_{kl}(\mathbf{x}_j)} d\bar{F}_{X_k, X_l}(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2} \int_{f_{kl}(\mathbf{x}_i) = f_{kl}(\mathbf{x}_j)} d\bar{F}_{X_k, X_l}(\mathbf{x}_i, \mathbf{x}_j).$$

Proposition 6.74. *The pairwise expected ranking accuracy as defined by (3.2) is a reciprocal relation.*

Proof. For a ranking function f_{kl} , let f_{lk} still be defined by $f_{lk} = -f_{kl}$ with $1 \leq k < l \leq r$ and let $\bar{F}_{X_k, X_l}$ denote the bivariate cumulative distribution function for the random variables X_k and X_l , then by definition we find

$$\begin{aligned} A_{kl}(f_{kl}) + A_{lk}(f_{lk}) &= \int_{f_{kl}(\mathbf{x}_i) < f_{kl}(\mathbf{x}_j)} d\bar{F}_{X_k, X_l}(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \int_{f_{kl}(\mathbf{x}_i) = f_{kl}(\mathbf{x}_j)} d\bar{F}_{X_k, X_l}(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \int_{f_{kl}(\mathbf{x}_i) > f_{kl}(\mathbf{x}_j)} d\bar{F}_{X_k, X_l}(\mathbf{x}_i, \mathbf{x}_j) \\ &= 1. \end{aligned}$$

□

Given these observations, we can generalize AUC ranking representability to ERA ranking representability.

Definition 6.75. *Let $X_1, \dots, X_r$ be independent random data objects. We call a set $\bar{\mathcal{F}}$ of pairwise ranking functions ERA ranking representable on $X_1, \dots, X_r$ if there exists a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that for $1 \leq k < l \leq r$*

$$A_{kl}(\bar{\mathcal{F}}) = A_{kl}(f). \quad (6.28)$$

Using this extended definition, we can prove the following remarkable theorem.

Proposition 6.76. *A triplet $\bar{\mathcal{F}} = \{f_{12}, f_{23}, f_{13}\}$ of pairwise ranking functions defined on three independent random data objects is ERA ranking representable if and only if the triplet of expected ranking accuracies is C -transitive w.r.t. the conjunctive $C_{\mathbf{P}_0}$ defined by*

$$C_{\mathbf{P}_0}(a, b) = \begin{cases} 0, & \text{if } a + b \leq 1, \\ ab, & \text{if } a + b > 1. \end{cases}$$

Proof. The proof is inspired by the sufficient condition derived for AUC ranking representability in the previous section. Given that expected ranking accuracy is the immediate generalization of the AUC from a finite sample to the underlying distribution, we only need to show that C_{jkl} converges in the limit to $C_{P_0} : [0, 1]^2 \rightarrow [0, 1]$. In order to examine this limit behavior, we will have to assume that the ratio of the number of data objects sampled from the respective categories remains unchanged. We will only discuss the case where $n_j = n_k = n_l$. For other ratios, a similar proof can be formulated. So, we have to show that

$$C_{P_0}(a, b) = \lim_{n_j \rightarrow \infty} C_{jjj}(a, b),$$

with $n_j \in \mathbb{N}$. We immediately find that in the limit Properties (1)–(4) in Proposition (6.73) respectively reduce to monotonicity, commutativity, 0 as absorbing element and 1 as neutral element. However, C_{P_0} will not be a t-norm since associativity does not hold. To compute the limit of C_{jjj} , we will consider three cases.

Case 1: $\lfloor \frac{a^*}{n_j} \rfloor + \lfloor \frac{b^*}{n_j} \rfloor > n_j$

We show that in this case the minimum of the integer quadratic program is found by applying Strategy 1 as described in the proof of the fifth property in Proposition 6.73. The easiest way to recognize this is by considering the graph-theoretic interpretation of Proposition 6.68. Given $\lfloor \frac{a^*}{n_j} \rfloor + \lfloor \frac{b^*}{n_j} \rfloor > n_j$, always paths will be found from V_j to V_l that pass through a node of V_k . The first thing to observe is that the splits for $\mathbf{a}^* = (a_1^*, \dots, a_{n_j}^*)$ and $\mathbf{b}^* = (b_1^*, \dots, b_{n_j}^*)$ considered in Strategy 2 lead to a value of 1 for the objective function, because a node in V_k can be found that has incoming edges from all V_j -nodes and outgoing edges to all V_l -nodes. As a consequence, we connect all couples of nodes from $V_j \times V_l$ in this way. Irrespective the split of a^* and b^* that is chosen, we will always find connected couples. The only chance to end up with as few connected couples as possible is by constructing as many paths as possible through couples that have to be connected anyway. This is exactly what is accomplished by Strategy 1, leading to vectors $\mathbf{a}^*$ and $\mathbf{b}^*$ that are constructed as follows:

$$a_i^* = \begin{cases} \lfloor \frac{a^*}{n_j} \rfloor, & \text{if } i \leq n_j - a^* \bmod n_j, \\ \lceil \frac{a^*}{n_j} \rceil, & \text{if } i > n_j - a^* \bmod n_j, \end{cases}$$

$$b_i^* = \begin{cases} \lfloor \frac{b^*}{n_j} \rfloor, & \text{if } i > b^* \bmod n_j, \\ \lceil \frac{b^*}{n_j} \rceil, & \text{if } i \leq b^* \bmod n_j. \end{cases}$$

Let us now try to derive a closed form for the objective function. By applying Strategy 1, we minimize the number of nodes from V_j with outgoing edges to V_k . At least $\lceil \frac{a^*}{n_j} \rceil$ nodes from V_j have outgoing edges to V_k . $\lfloor \frac{a^*}{n_j} \rfloor$ of these nodes have n_j outgoing edges to V_k (more precisely, to all elements of V_k). Only one node can have less than n_j edges (when a^*/n_j is not an integer). Similarly, we find that at least $\lceil \frac{b^*}{n_j} \rceil$ nodes from V_l have incoming edges from V_k . $\lfloor \frac{b^*}{n_j} \rfloor$ of these nodes have n_j incoming edges, and the remaining node can have less than n_j edges (when b^*/n_j is not an integer). Putting everything together, we find that all V_j -nodes with outgoing edges to V_k and all V_l -nodes with incoming edges from V_k have to be connected in this way, except the V_j -node and V_l -node with less than n_j outgoing (respectively incoming) edges. One can easily verify that these two nodes will also be connected when $a^* \bmod n_j + b^* \bmod n_j \geq 1$. This corresponds to the following value for the objective function:

$$\tau = \begin{cases} \frac{1}{n_j^*} (\lceil \frac{a^*}{n_j} \rceil \lceil \frac{b^*}{n_j} \rceil - 1), & \text{if } a^* \bmod n_j + b^* \bmod n_j < 1, \\ \frac{1}{n_j^*} \lceil \frac{a^*}{n_j} \rceil \lceil \frac{b^*}{n_j} \rceil, & \text{if } a^* \bmod n_j + b^* \bmod n_j \geq 1. \end{cases}$$

Remark that the -1 corresponds to the couple that is potentially not connected. τ reduces in the limit to the following simple expression

$$\lim_{n_j \rightarrow \infty} \tau = \frac{a^* b^*}{n_j^4} = ab.$$

Case 2: $\lceil \frac{a^*}{n_j} \rceil + \lceil \frac{b^*}{n_j} \rceil \leq n_j$

We show that in this case the minimum of the integer quadratic program is 0. This minimum is found by applying Strategy 2 as described in the proof of the fifth property in Proposition 6.73. The vectors $\mathbf{a}^* = (a_1^*, \dots, a_{n_j}^*)$ and $\mathbf{b}^* = (b_1^*, \dots, b_{n_j}^*)$ are now constructed as follows:

$$a_i^* = \begin{cases} 0, & \text{if } i < n_j - \lceil \frac{a^*}{n_j} \rceil, \\ a^* \bmod n_j, & \text{if } i = n_j - \lceil \frac{a^*}{n_j} \rceil, \\ n_j, & \text{if } i > n_j - \lceil \frac{a^*}{n_j} \rceil, \end{cases}$$

and

$$b_i^* = \begin{cases} n_j, & \text{if } i < \lceil \frac{b^*}{n_j} \rceil, \\ b^* \bmod n_j, & \text{if } i = \lceil \frac{b^*}{n_j} \rceil, \\ 0, & \text{if } i > \lceil \frac{b^*}{n_j} \rceil, \end{cases}$$

which of course results in a feasible solution for the integer quadratic program. Given that $(\lceil \frac{a^*}{n_j} \rceil + \lceil \frac{b^*}{n_j} \rceil) \leq n_j$, it follows that $a_i^* b_i^* = 0$ for all $i = 1, \dots, n_j$ such that the objective function becomes zero.

Besides these two cases, normally also a third case has to be distinguished, when none of the above two conditions hold. However, we will not discuss this third case for which the minimum of the objective function is more difficult to express. Fortunately, this case vanishes in the limit, since

$$\begin{aligned} \lim_{n_j \rightarrow \infty} \lceil \frac{a^*}{n_j} \rceil - \lfloor \frac{a^*}{n_j} \rfloor &= 0, \\ \lim_{n_j \rightarrow \infty} \lceil \frac{b^*}{n_j} \rceil - \lfloor \frac{b^*}{n_j} \rfloor &= 0. \end{aligned}$$

□

C_{P_0} -transitivity turns out to be a type of C -transitivity.

Proposition 6.77. *A reciprocal relation $Q : \mathcal{X}^2 \rightarrow [0, 1]$ is called C_{P_0} -transitive if and only if Q is cycle-transitive w.r.t. the upper bound function*

$$\mathcal{U}_{C_{P_0}}(\alpha, \beta, \gamma) = \min(\alpha + \beta - C_{P_0}(\alpha, \beta), \alpha + \gamma - C_{P_0}(\alpha, \gamma), \beta + \gamma - C_{P_0}(\beta, \gamma)).$$

Proposition 6.78. *C_{P_0} -transitivity implies dice transitivity.*

Proof. Let us consider a unit square for the couple (β, γ) . From the constraint $\beta \leq \gamma$ follows that only the region above the bisector (i.e. the line given by the identity function) must be considered. Dice transitivity can be expressed as $\beta\gamma \leq 1 - \alpha$. Let us try to make this constraint as tight as possible by choosing $\alpha = \beta$. This means that dice transitivity only imposes a constraint on the reciprocal relations when the following inequality holds:

$$\gamma > \frac{1 - \beta}{\beta}.$$

We have visualized the corresponding region of the unit square in Figure 6.11(a). This region is a subregion of the region $\beta + \gamma > 1$ where C_{P_0} -transitivity equals $\beta\gamma$. C_{P_0} -transitivity is therefore a stronger type of transitivity than dice transitivity. □

This proposition mainly confirms that all pieces of the puzzle fit surprisingly well. In the previous sections it was shown how AUC transitivity induces

Figure 6.11: A visualization of the unit square for (β, γ) and the regions for which C_{P0} -transitivity and dice transitivity result in active constraints. In the figure on the left only the shaded region can cause a violation. The figure on the right illustrates that in this region no violation is observed when $n_j = n_k = n_l = 10$.

a sufficient condition for AUC ranking representability, while dice transitivity could only lead to a necessary condition. From this we were able to prove indirectly that the former type of transitivity had to be stronger than the latter one, but this could not be observed directly from the upper bound functions. Since this relationship between both types of cycle-transitivity can be observed very easily in the infinite case, it gives an additional confirmation of the correctness of our analysis in the finite case. To draw the attention of the reader to the potentially tight bound between AUC-transitivity and dice transitivity, Figure 6.11(b) visualizes all considered regions for $n_j = n_k = n_l = 10$. It is shown that for this choice of n_j, n_k, n_l the regions of the square where $C_{jkl}(\beta, \gamma)$ exceeds $\beta\gamma$ do not overlap with the region where the dice constraint is active, yet both regions are located very close to each other.

Proposition 6.79. T_P -transitivity implies C_{P0} -transitivity.

Proof. Immediate since $C_{P0} \leq T_P$. □

6.4.7 Practical considerations

In the previous sections an interesting sufficient condition was derived for reducing one-versus-one ensembles to ranking models by investigating the pairwise AUCs. The sufficient condition for the three class case can be verified by solving an integer quadratic program. This class of problems can in

general not be solved exactly in polynomial time, but we believe that in this specific setting a closed form for the solution can be derived such that optimization can be skipped. However, we have not elaborated on this idea since we were able to derive a more simple expression for the infinite case where a generalization is made from a sample to the underlying distribution. As a consequence, it makes sense to verify ERA transitivity on the pairwise AUCs instead of AUC transitivity, since we are mainly interested in generalizing to out-of-sample data. So, we have obtained a verifiable condition to check whether a one-versus-one method can be reduced to a single ranking for the three-class case, but will this approach also work in practice? The answer is not unhesitatingly yes for the following two main reasons:

1. The framework of AUC ranking representability only identifies the existence of a single ranking function that yields the same error in terms of pairwise expected ranking accuracy as a one-versus-one ensemble, but nothing can be said about the complexity of this ranking model. When we minimize a loss function over a hypothesis space of ranking models, no guarantee can be given that the model that we try to find is included in this hypothesis space.
2. Ordinal regression models do not try to optimize accuracy, but a loss function that takes the magnitude of an error into account. However, the magnitude of an error has no meaning at all in a multi-class setting, even if we artificially try to impose an order on the categories. The optimization of a magnitude-based loss function can harm the performance significantly when only accuracy is taken into account as performance measure. Even for ordinal regression data sets this effect can be observed. For example, we first did some initial experiments on the benchmark problems of the previous chapter before looking at real multi-class data sets, and on these benchmark data sets one-versus-one SVMs outperformed kernel-based ordinal regression models significantly when only accuracy was taken into account as performance measure. Conversely, for the carpet wear assessment data that was considered in Chapter 2, this phenomenon did not occur, probably because that data set is very small.

These two findings imply that in practice an AUC ranking representable one-versus-one model will not always be beaten by a single ranking model. Revealing the situations where a performance gain will be obtained might not be done easily and would require an experimental validation on numerous

Figure 6.12: A plot of the first two principal components of the iris data set.

Method	OVO	DAG	OVA	W&W	C&S	Ranking
Acc.	97.333	97.333	96.000	97.333	87.333	98.000

Table 6.1: A comparison of the performance obtained on the iris data set by ten-fold cross-validation with various multi-class methods and a simple ranking model. The results of the multi-class methods are duplicated from (Hsu & Lin 2002).

data sets. Since most benchmark problems for multi-class classification consider more than three classes, first an extension of our approach to more than three categories has to be invented. An obvious generalization could be established by looking at all triplets of classes and simplifying those for which AUC ranking representability is fulfilled, but further research is required to verify whether this idea would work. Hereunder we have analyzed two three-class benchmark problems from the UCI repository to illustrate the potential benefits of AUC ranking representability. Both problems have been analyzed by Hsu & Lin (2002) in an experimental comparison of different multi-class schemes with SVMs as base classifiers. We decided to compare with their results because they describe their experimental setup in such a way that the experiments could be easily replicated.

6.4.7.a Iris data

The first data set that was analyzed is the well-known *iris* data set, which is probably one of the most frequently utilized data sets to evaluate multi-class classifiers. The first two principal components of the data are visualized in Figure 6.12. One can see that class $\bar{y}_2$ is sandwiched on the left side by class $\bar{y}_1$ and class $\bar{y}_3$ on the right side, thus theoretically we impose the order $\bar{y}_1 < \bar{y}_2 < \bar{y}_3$ on the categories if we would fit an ordinal regression model to that data set. Hsu & Lin (2002) report for this data set that a one-versus-one model outperforms all other multi-class schemes. Since the iris data does not have an accompanying test set, they draw their conclusions based on the 10-fold cross-validation error obtained for the best choice of λ and γ (RBF-kernel) found during model selection. A short overview of the results is given in Table 6.1 together with the results obtained by fitting the kernel-based ordinal regression model of Chapter 2 to the data. To this end, we first fitted a one-versus-one SVM to the data by using the values of the hyperparameters specified by Hsu & Lin (2002), resulting in the following pairwise AUCs when the whole data set is used for training¹:

$$\hat{A}_{12} = 1.0, \quad \hat{A}_{23} = 0.995, \quad \hat{A}_{13} = 1.0.$$

These pairwise AUCs definitely satisfy C_{P0} -transitivity, so a reduction to a ranking model would make sense theoretically. When we fit an ordinal regression model to the data with the SVORIM-package, then the following pairwise AUCs are measured on training data:

$$\hat{A}_{12} = 1.0, \quad \hat{A}_{23} = 0.998, \quad \hat{A}_{13} = 1.0.$$

Thus, the performance on training data increases, but more importantly, a better cross-validated performance in terms of accuracy is obtained with the ordinal regression model, using the same experimental setup as Hsu & Lin (2002). This might be surprising at first sight, but one must take into account that PCA analysis already identified an ordinal structure of the classes. This is clearly an example where a reduction to a single ranking model can boost the generalization performance.

6.4.7.b DNA data

The second data set that was analyzed is the DNA data set. Contrary to the iris data set, this data set has substantially more instances. The data is also

¹Remark that we have to train on the whole data set in order to compute pairwise AUCs, since multivariate performance measures cannot be computed by means of cross-validation.

relatively high-dimensional (180 features) such that the curse of dimensionality can play a role. Furthermore, the data has been split into a train and test set, so one can avoid the use of cross-validation here. Using the methodology of Hsu & Lin (2002), this resulted in the following pairwise AUCs on the train set:

$$\hat{A}_{12} = 1.0, \quad \hat{A}_{23} = 0.952, \quad \hat{A}_{13} = 0.909.$$

This triplet of pairwise AUCs again results in an AUC ranking representable model and suggests the order $\bar{y}_1 < \bar{y}_3 < \bar{y}_2$ on the categories. So, let us swap categories $\bar{y}_2$ and $\bar{y}_3$, then the following pairwise AUCs on training data are obtained:

$$\hat{A}_{12} = 0.909, \quad \hat{A}_{23} = 0.952, \quad \hat{A}_{13} = 1.0.$$

Subsequently, we tested the one-versus-one SVM and the SVORIM algorithm on the test set. Using the same methodology of Hsu & Lin (2002), we were not able to achieve exactly the same results for this data set, yet we obtained a similar (but slightly worse) accuracy on the test set. We do not give any plausible explanation for that since we adopted exactly the same setup. The results are summarized in Table 6.2 and give the impression that the SVORIM algorithm is not able to compete with the other multi-class approaches. However, nothing is further from the truth as the opposite conclusion can be drawn from the pairwise AUCs measured on the test set. For the one-versus-one model the following values are obtained:

$$\hat{A}_{12} = 0.808, \quad \hat{A}_{23} = 0.833, \quad \hat{A}_{13} = 0.997,$$

while the ordinal regression model yields substantially better pairwise AUCs on the test set:

$$\hat{A}_{12} = 0.906, \quad \hat{A}_{23} = 0.907, \quad \hat{A}_{13} = 0.996.$$

So, how can these surprisingly different trends between accuracy on the one hand and the pairwise AUCs on the other hand be explained? As discussed above, this is caused by the fact the SVORIM algorithm does not optimize accuracy but a magnitude-based loss function. Figure 6.13 gives a good overview of what is going on. On the left side, it shows the first two principal components of the test set with the real labels and on the right side it shows the same test set with the labels predicted by SVORIM. The ordinal regression algorithm clearly assigns too many instances to the middle class in an attempt to minimize the magnitude of errors. Apparently, that does not

Figure 6.13: A graphical illustration of the effect of minimizing a magnitude-based loss function with the SVORIM algorithm on the DNA test set. The first two principal components are shown with the real labels on the left side and the predicted labels on the right side. As a result of the transitivity analysis of the pairwise AUCs, the labels of categories $\bar{y}_2$ and $\bar{y}_3$ have been swapped.

Method	OVO	DAG	OVA	W&W	C&S	Ranking
Acc.	95.441	95.447	95.784	95.618	95.889	
Acc.	94.266					76.560

Table 6.2: A comparison of the performance obtained on the DNA data set (independent test set) with various multi-class methods and a simple ranking model. The results of (Hsu & Lin 2002) are given in the first row and our results in the second row.

affect the performance in terms of pairwise AUCs for the DNA data set and we would definitely recommend to look at this last criterion instead of accuracy in order to compare the performance of one-versus-one models and single ranking models.

6.5 Conclusion

In this chapter the connection between ordinal regression models, pairwise preference models and multi-class classification models was investigated by examining the underlying ranking function that characterizes all three types of models. We introduced the framework of cycle-transitivity for the analysis of reciprocal relations as the main tool to establish this connection, leading

to the concept of ranking representability. For pairwise preference models, different forms of ranking representability could be defined, each of them corresponding to a specific type of stochastic transitivity. For one-versus-one ensembles, we defined the concepts of strict ranking representability, AUC ranking representability and ERA ranking representability by using ROC-based concepts and their graph-theoretic reformulations. We paid specific attention to the three-class case for which the sufficient condition could be verified by solving an integer quadratic program. Moreover, an approximate solution of this integer quadratic program could be found without using any solver, by considering the generalization from a finite sample to the underlying distribution. This gave rise to the concept of ERA ranking representability and a new type of cycle-transitivity that can be interpreted as a necessary and sufficient condition for the ERA ranking representability of a triplet of random variables, each of them corresponding to one category in a three-class classification task. In addition, we presented some initial results on the practical usefulness and potential limitations of our approach, but a more thorough experimental validation might be advisable.

The results presented in this chapter have been published in (Waegeman *et al.* 2008b).

Conclusions and further perspectives

In this dissertation, we considered the modelling of preference judgments given by humans from a supervised learning perspective. Special emphasis was put on the ordinal regression and pairwise preference learning settings, in which the concept of an underlying ranking function plays an important role.

In Chapter 3, we analyzed different performance measures for ranking-based ordinal regression models. We argued that evaluating the ranking returned by an ordinal regression model is often more appropriate than looking at ‘mean zero-one error’ or ‘mean absolute error’, especially with skew class or cost distributions. To that end, we extended the concept of expected ranking accuracy for ordinal labelled data and showed that a nonparametric unbiased estimator $\hat{U}(f, D)$ of this quantity corresponds to the volume under the ROC surface (VUS) spanned by the true positive rates of each class. Moreover, we revealed the relationship between $\hat{U}(f, D)$ and well-known ranking-based performance measures, which can be considered as approximations of this statistic. The volume under the ROC surface and related measures do not manifest a monotone relationship and they also have a different distribution. These observations were confirmed by experiments on synthetic and real data. In particular, a large trade-off was discovered between ‘mean zero-one error’ and ‘mean absolute error’ on the one hand and ranking-based measures on the other hand. The latter mutually displayed smaller trade-offs, but among them $\hat{U}_{pairs}(f, D)$ turned out to concentrate too much on the biggest classes.

We conclude from Chapter 3 that existing methods for ordinal regression, which typically minimize a loss based on error rate or the number of incorrectly ranked object pairs, might not construct appropriate models when the class or cost distributions are skew. ROC analysis offers in this case a valuable alternative allowing to pick a classifier from the surface for a specific setting of cost and the volume under the ROC surface gives a good overall indication of the quality of the model for different costs without favoring the majority classes.

In Chapter 4 we analyzed the complexity of computing VUS, its variance estimator and covariance estimator for two volumes. For the computation of VUS, we proposed a $\mathcal{O}(n \log(n))$ dynamic programming algorithm dominated by a single sorting operation on the data set. The estimators of variance and covariance were reformulated in terms of recurrent functions over graphs and we showed that with dynamic programming techniques for evaluating these recurrent functions, much faster algorithms could be constructed compared to exhaustive algorithms that evaluate all r -tuples. In particular, respectively $\mathcal{O}(2^r n^2)$ and $\mathcal{O}(2^r n^4)$ algorithms were designed for estimating the variance and the covariance. Simulations confirmed our theoretical observations and showed that the algorithms for computing $\hat{U}(f, D)$ and $\hat{\sigma}_U^2(f, D)$ are applicable to very large data sets. Our algorithm for computing the covariance estimator resulted in a smaller time gain compared to an exhaustive algorithm, but still scaled well with increasing r .

The algorithms presented in Chapter 4 are useful for different purposes. First of all, they make it possible to perform ROC analysis on large data sets and to estimate $U(f)$ on large ordinal regression models. They can be directly plugged into the statistical tests developed by Nakas & Yiannoutsos (2004). With these algorithms, new ranking methods can be constructed for using or optimizing the volume under the ROC surface (that is exactly what we did in Chapter 5). Such algorithms have been proposed for the binary case and in particular optimizing ranking-based measures can lead to noticeably different models compared to minimizing the error rate (Yan *et al.* 2003, Cortes & Mohri 2003, Herschtal & Raskutti 2005). Furthermore, the variance and covariance can provide important criteria to choose a particular ranking function in the model selection phase. The variance of the AUC is for example used in (Agarwal *et al.* 2005) to derive distribution-free generalization bounds for the AUC. The covariance gives an indication of the similarity between two ranking functions.

So, in Chapter 5 we further exploited these ideas postulated in order to develop a new algorithm for ordinal regression learning. Inspired by the graph-

theoretic reformulations for VUS, we called this setting the layered ranking problem because each correctly ranked r -tuple, consisting of one object of each category, induces a path in the corresponding layered ranking graph. Contrary to conventional approaches for layered ranking in which typically the number of correctly ranked object pairs is optimized, we proposed to learn a ranking function with a maximal number of correctly ranked r -tuples. A naive implementation would require at least $\mathcal{O}(n^{2r})$ computation time, but we showed that the ϵ -approximate solution could be found in $\mathcal{O}(n^3)$ time for any r by reformulating the quadratic program as a structured learning problem.

Notwithstanding the increasing complexity of such an approach, we believe that looking at correctly ranked r -tuples instead of correctly ranked pairs has important benefits and leads to better models. The reported experiments clearly support this claim. On synthetic data and several benchmark data sets, a statistically significant improvement in performance was measured, definitely in terms of the VUS, but also in terms of the C-index. For some unbalanced problems, a small trade-off appeared between optimizing VUS on the one hand and the C-index on the other hand. We argued that the pairwise approach might focus too strongly on some of the categories instead of assessing the ranking model globally. Nevertheless, we acknowledge that the complexity of ranking r -tuples remains a bottleneck for large samples.

Finally, in the more theoretical Chapter 6, the connection between ordinal regression models, pairwise preference models and one-versus-one ensembles was investigated. It turned out that again the underlying ranking function forms the fundamental building block in this analysis. Therefore, we defined the concept of ranking representability of pairwise preference models and one-versus-one ensembles, and established its strong relationship with the framework of (cycle) transitivity. For pairwise preference models, different forms of ranking representability could be defined, each of them corresponding to a specific type of stochastic transitivity. For one-versus-one ensembles, we defined the concepts of strict ranking representability and AUC ranking representability. Unlike strict ranking representability, it is far from trivial to verify whether a set of pairwise rankings $\overline{\mathcal{F}}$ is AUC ranking representable, since examining all graphs in $\mathfrak{G}_{AUC}(\overline{\mathcal{F}}, D)$ will be computationally intractable for large training samples. We paid specific attention to the three class case for which the sufficient condition could be verified by solving an integer quadratic program. Moreover, an approximate solution of this integer quadratic program could be found without using any solver, by exploring additional properties of the functions C_{jkl} . More specifically, we were able

to compute the form that C_{jkl} takes in the limit with the sample size n going to infinity. In the limit, C_{jkl} became a conjunctive, leading to a new type of cycle transitivity. This type of cycle transitivity can be interpreted as a necessary and sufficient condition for the AUC ranking representability of a triplet of ranking functions defined on three random data objects such that each of these objects corresponding to one category in a three-class classification task. This resulted in a deviation bound on the difference between C_{jkl} and C . As a consequence, one can verify AUC ranking representability in an approximate way without any computational burden by computing this conjunctive on the observed AUCs.

The ideas presented in this thesis leave different possibilities for further research. In the first place, we think at a further generalization of AUC ranking representability and a more thorough experimental validation of its usefulness in practical situations. The approach described in Chapter 6 might at first sight be only useful for the three-class case, but the procedure can be immediately generalized to more than three classes. This can for example be realized with an algorithm that examines all triplets of pairwise rankings in an iterative way, and simplifies them to a single ranking model if the conditions are satisfied. Since the number of triplets that have to be compared rapidly increases with the number of categories, the chance of detecting triplets that can be simplified increases too. Remark also that AUC transitivity becomes a quite weak type of transitivity in the limit, so one should be able to find realistic problems for which the reduction to a ranking model makes sense. To give just one example, the *satimage* data set visualized in Figure 7.1 is an UCI multi-class problem ($r = 6$) where a reduction of a full one-versus-one model would increase the generalization performance.

Furthermore, the practical usefulness of the concept of AUC ranking representability is at this moment still unclear. While the reduction to single ranking models definitely makes sense for certain multi-class problems, a gain in classification performance was not observed for all data sets on which an AUC ranking representable multi-class model could be constructed. This is to our opinion mainly caused by the fact that ordinal regression models optimize a loss function that takes the magnitude of errors into account. A further experimental and algorithmic exploration of these findings might be advisable.

Figure 7.1: *A plot of the first two principal components of the satimage data set as an example of a multi-class problem where fitting a full one-versus-one model to the data would not be advisable.*

Bibliography

- Agarwal, S., Graepel, T., Herbrich, R., Har-Peled, S. & Roth, D.** (2005), “Generalization Bounds for the Area under the ROC Curve”, *Journal of Machine Learning Research*, 6, 393–425.
- Agresti, A.** (2002), “Categorical Data Analysis, 2nd version”, John Wiley and Sons.
- Allwein, E., Schapire, R. & Singer, Y.** (2000), “Reducing multi-class to binary: A Unifying Approach for Margin Classifiers”, *Journal of Machine Learning Research*, 1, 113–141.
- Ataman, K., Street, N. & Zhang, Y.** (2006), “Learning to Rank by Maximizing AUC with Linear Programming”, in : *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vancouver, BC, Canada, 123–129.
- Ballinger, T. & Wilcox, P.** (1997), “Decisions, Error and Heterogeneity”, *The Economic Journal of Artificial Intelligence Research*, 107, 1090–1105.
- Bengio, Y. & Grandvalet, Y.** (2004), “No Unbiased Estimator of the Variance of k -fold Cross-Validation”, *Journal of Machine Learning Research*, 5, 1089–1105.
- Billot, A.** (1995), “An Existence Theorem for Fuzzy Utility Functions: A New Elementary Proof”, *Fuzzy Sets and Systems*, 74, 271–276.

- Bishop, C.** (2006), “Pattern Recognition and Machine Learning”, Springer.
- Boutilier, C., Brafman, R., Domshlak, C., Hoos, H. & Poole, D.** (2004), “CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements”, *Journal of Artificial Intelligence Research*, 21, 135–191.
- Bouyssou, D., Marchant, T., Pirlot, M., Tsoukias, A. & Vincke, P.** (2006), “Evaluation and Decision Models with Multiple Criteria”, Springer.
- Bradley, R. & Terry, M.** (1952), “Rank analysis of incomplete block designs. I: The method of paired comparisons”, *Biometrika*, 39, 324–345.
- Brefeld, U. & Scheffer, T.** (2005), “AUC Maximizing Support Vector Learning”, in : *Proceedings of the ICML Workshop on ROC Analysis in Machine Learning*, Bonn, Germany.
- Cao, Z., Qin, T., Liu, T., Tsai, M. & Li, H.** (2007), “Learning to Rank: from Pairwise Approach to Listwise Approach”, in : *Proceedings of the International Conference on Machine Learning*, Corvallis, OR, USA, 129–136.
- Cao-Van, K.** (2003), “Supervised Ranking, from semantics to algorithms”, PhD-thesis, Ghent University, Belgium.
- Carroll, J., De Soete, G. & DeSarbo, W.** (1990), “Two Stochastic Multidimensional Choice Models for Marketing Research”, *Decision Sciences*, 21, 337–356.
- Cheong, S., Oh, S. & Lee, S.** (2004), “Support Vector Machines with Binary Tree Architecture for Multi-class Classification”, *Neural Information Processing*, 2 (3), 47–51.
- Chu, W. & Ghahramani, Z.** (2005*a*), “Gaussian Processes for Ordinal Regression”, *Journal of Machine Learning Research*, 6, 1019–1041.
- Chu, W. & Ghahramani, Z.** (2005*b*), “Preference Learning with Gaussian Processes”, in : *Proceedings of the International Conference on Machine Learning*, Bonn, Germany, 137–144.
- Chu, W. & Keerthi, S.** (2005), “New Approaches to Support Vector Ordinal Regression”, in : *Proceedings of the International Conference on Machine Learning*, Bonn, Germany, 321–328.

-
- Chu, W. & Keerthi, S.** (2007), “Support Vector Ordinal Regression”, *Neural Computation*, 19 (3), 792–815.
- Clmenon, S. & Vayatis, N.** (2007), “Ranking the Best Instances”, *Journal of Machine Learning Research*, 8, 2671–2699.
- Coello, A. & Lechunga, M.** (2002), “MOPSO: A Proposal for Multi-Objective Swarm Optimization”, *Evolutionary Computation*, 2, 1051–1056.
- Cohen, W., Schapire, R. & Singer, Y.** (1998), “Learning to Order Things”, in : *Advances in Neural Information Processing Systems* 10.
- Cortes, C. & Mohri, M.** (2003), “AUC Optimization versus Error Rate Minimization”, in : *Advances in Neural Information Processing Systems*, Vancouver, Canada, 16, MIT Press.
- Cortes, C. & Mohri, M.** (2004), “Confidence Intervals for the Area Under the ROC Curve”, in : *Advances in Neural Information Processing Systems (NIPS 2004)*, Vancouver, Canada, 17, 305–312, MIT Press.
- Cortes, C., Mohri, M. & Rastogi, A.** (2007), “Magnitude-Preserving Ranking Algorithms”, in : *Proceedings of the International Conference on Machine Learning*, Corvallis, OR, USA.
- Cortes, C. & Vapnik, V.** (1995), “Support Vector Networks”, *Machine Learning*, 20 (3), 273–297.
- Crammer, K. & Singer, Y.** (2001*a*), “On the algorithmic implementation of of multiclass kernel-based vector machines”, *Journal of Machine Learning Research*, 2, 265–292.
- Crammer, K. & Singer, Y.** (2001*b*), “Pranking with Ranking”, in : *Proceedings of the Conference on Neural Information Processing Systems*, Vancouver, Canada, 641–647.
- Crammer, K. & Singer, Y.** (2002), “On the learnability and design of output codes for multiclass problems”, *Machine Learning*, 47 (2), 201–233.
- Cristianini, N. & Shawe-Taylor, J.** (2000), “An Introduction to Support Vector Machines”, Cambridge University Press.
- De Baets, B. & De Meyer, H.** (2005), “Transitivity frameworks for reciprocal relations: cycle-transitivity versus *FG*-transitivity”, *Fuzzy Sets and Systems*, 152, 249–270.

- De Baets, B., De Meyer, H., De Schuymer, B. & Jenei, S.** (2006), “Cyclic evaluation of transitivity of reciprocal relations”, *Social Choice and Welfare*, 26, 217–238.
- De Baets, B., De Schuymer, B. & H. De Meyer** (), “A Graph-Theoretic Characterization of Cycle-Transitivity w.r.t. Quasi-Copulas”, Unpublished work.
- De Baets, B. & Fodor, J.** (2003), “Principles of Fuzzy Preference Modelling and Decision Making”, Academia Press, chapter Additive Fuzzy Preference Structures, the Next Generation, 15–27.
- De Baets, B., Van De Walle, B. & Kerre, E.** (1995), “Fuzzy Preference Structures without Incomparability”, *Fuzzy Sets and Systems*, 76, 333–348.
- De Schuymer, B.** (2005), “Comparison of Random Variables form a Game-Theoretic Perspective”, PhD-thesis, Ghent University.
- De Schuymer, B., De Meyer, H. & De Baets, B.** (2005), “Cycle-Transitive Comparison of Independent Random Variables”, *Journal of Multivariate Analysis*, 96, 352–373.
- De Schuymer, B., De Meyer, H., De Baets, B. & Jenei, S.** (2003), “On the cycle-transitivity of the dice model”, *Theory and Decision*, 54, 164–185.
- Demsar, J.** (2006), “Statistical Comparisons of Classifiers over Multiple Datasets”, *Journal of Machine Learning Research*, 7, 1–30.
- Dhzafarov, E.** (2003), “Thurstonian-type Representations for Same-Different Discriminations: Deterministic Decisions and Independent Images”, *Journal of Mathematical Psychology*, 47, 184–204.
- Dias, L. & Mousseau, V.** (2006), “Inferring ELECTRE’s Veto-Related Parameters from Outranking Examples”, *European Journal of Operational Research*, 170, 172–191.
- Dietterich, T.** (1998), “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms”, *Neural Computation*, 10, 1895–1924.
- Dietterich, T. G. & Bakiri, G.** (1995), “Solving Multiclass Learning Problems via Error-Correcting Output Codes”, *Journal of Artificial Intelligence Research*, 2, 263–286.

-
- Doumpos, M. & Zopounidis, C.** (2004), "A multicriteria classification approach based on pairwise comparisons", *European Journal of Operational Research*, 158, 378–389.
- Doyle, J.** (2004), "Prospects for Preferences", *Computational Intelligence*, 20, 111–136.
- Dreiseitl, S., Ohno-Machado, L. & Binder, M.** (2000), "Comparing Three-Class Diagnostic Tests by Three-Way ROC Analysis", *Medical Decision Making*, 20, 323–331.
- Fan, Z., Ma, J. & Zhang, Q.** (2002), "An Approach to Multiple-Attribute Decision Making Based on Fuzzy Preference Information on Alternatives", *Fuzzy Sets and Systems*, 131, 101–106.
- Fawcett, T.** (2006), "An Introduction to ROC Analysis", *Pattern Recognition Letters*, 27 (8), 861–874.
- Ferri, C., Hernandez-Orallo, J. & Salido, M.** (2003), "Volume Under ROC Surface for Multi-class Problems.", in : *Proceedings of the European Conference on Machine Learning*, Dubrovnik, Croatia, 108–120.
- Fieldsend, J. & Everson, M.** (2005), "Formulation and Comparison of Multi-Class ROC Surfaces.", in : *Proceedings of the ICML Workshop on ROC Analysis in Machine Learning*, 49–56.
- Fishburn, P.** (1970), "Utility Theory for Decision Making", Wiley.
- Flach, P.** (2004), "The Many Faces of ROC Analysis in Machine Learning", Tutorial presented at the European Conference on Machine Learning, Valencia, Spain.
- Fono, L. & Andjiga, N.** (2007), "Utility Function of Fuzzy Preferences on a Countable Set under Max- $\ast$ -Transitivity", *Social Choice and Welfare*, 28, 667–683.
- Frank, E. & Hall, M.** (2001), "A Simple Approach to Ordinal Classification", in : *Proceedings of the European Conference on Machine Learning*, Freiburg, Germany, 146–156.
- Freund, Y., Yier, R., Schapire, R. & Singer, Y.** (2003), "An Efficient Boosting Algorithm for Combining Preferences", *Journal of Machine Learning Research*, 4, 933–969.

- Fürnkranz, J.** (2002), “Round Robin Classification”, *Journal of Machine Learning Research*, 2, 723–747.
- Fürnkranz, J. & Hüllermeier, E.** (2003), “Pairwise Preference Learning and Ranking”, *Lecture Notes in Computer Science*, 2837, 145–156.
- Gheorghe, R., Bufardi, A. & Xirouchakis, P.** (2004), “Construction of Two-Parameters Fuzzy Outranking Relations from Fuzzy Evaluations”, *Fuzzy Sets and Systems*, 143, 391–412.
- Gheorghe, R., Bufardi, A. & Xirouchakis, P.** (2005), “Construction of global fuzzy preference structures from two-parameter single-criterion fuzzy outranking relations”, *Fuzzy Sets and Systems*, 153, 303–330.
- Gönen, M. & Heller, G.** (2005), “Concordance Probability and Discriminatory Power in Proportional Hazards Regression”, *Biometrika*, 92 (4), 965–970.
- Grabisch, M., Kojadinovic, I. & Meyer, P.** (2008), “A Review of Methods for Capacity Identification in Choquet Integral based Multi-Attribute Utility Theory, Applications of the Kappalab R Package”, *European Journal of Operational Research*, 186, 766–785.
- Greco, S., Mousseau, V. & Slowinski, R.** (2008), “Ordinal regression revisited: multiple criteria ranking using a set of additive value functions”, *European Journal of Operational Research*, 191, 415–435.
- Guermeur, Y.** (2007), “VC Theory of Large Margin Multi-Category Classifiers”, *Journal of Machine Learning Research*, 8, 2551–2594.
- Hand, D. & Till, R.** (2001), “A Simple Generalization of the Area under the ROC Curve for Multiple Class Problems”, *Machine Learning*, 45, 171–186.
- Hanley, J. & McNeil, B.** (1982), “The Meaning and Use of the Area Under a Receiver Operating Characteristics ROC Curve”, *Radiology*, 143, 29–36.
- Hanley, J. & McNeil, B.** (1983), “A Method of Comparing Receiver Operating Characteristics Curves Derived from the Same Class”, *Radiology*, 148, 839–843.
- Har-Peled, S., Roth, D. & Zimak, D.** (2002), “Constraint Classification: A New Approach to Multi-class Classification and Ranking”, in : *Advances in Neural Information Processing Systems 15*, pages 785–792.

-
- Harrington, E.** (2003), “Online Ranking/Collaborative Filtering using the Perceptron Algorithm”, in : Proceedings of the 20th International Conference on Machine Learning, Washington, USA, 250–257.
- Hastie, T. & Tibshirani, R.** (1998), “Classification by Pairwise Coupling”, *The Annals of Statistics*, 26 (2), 451–471.
- Hastie, T., Tibshirani, R. & Friedman, J.** (2001), “The Elements of Statistical Learning”, Springer.
- Heller, M. & Schnörr, B.** (2006), “Learning Sparse Representations by Non-Negative Matrix Factorization and Sequential Cone Programming”, *Journal of Machine Learning Research*, 7, 1385–1407.
- Herbrich, R., Graepel, T. & Obermayer, K.** (2000), “Large Margin Rank Boundaries for Ordinal Regression”, in : A. Smola, P. Bartlett, B. Schölkopf & D. Schuurmans, eds., *Advances in Large Margin Classifiers*, 115–132, MIT Press.
- Herbrich, R., Minka, T. & Graepel, T.** (2007), “TrueSkill: a Bayesian Skill Rating System”, in : B. Schölkopf, J. Platt & T. Hoffman, eds., *Advances in Neural Information Processing Systems 19*, 569–576, The MIT Press, Cambridge, MA.
- Herschtal, A. & Raskutti, B.** (2005), “Optimizing Area Under the ROC Curve using Gradient Descent”, in : Proceedings of the International Conference on Machine Learning, Bonn, Germany, 49–57.
- Higgins, J.** (2004), “Introduction to Modern Nonparametric Statistics”, Duxbury Press.
- Hsu, C. & Lin, C.** (2002), “A comparison of methods for multi-class support vector machines”, *IEEE Transactions on Neural Networks*, 13, 415–425.
- Joachims, T.** (2005), “A Support Vector Method for Multivariate Performance Measures”, in : Proceedings of the International Conference on Machine Learning, Bonn, Germany, 377–384.
- Joachims, T.** (2006), “Training Linear SVMs in Linear Time”, in : Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 217–226.

- Jung, S., Hong, J. & Kim, T.** (2005), “A Statistical Model for User Preference”, *IEEE Transactions on Knowledge and Data Engineering*, 17, 834–843.
- Kahn, A.** (1962), “Topological sorting of large networks”, *Communications of the ACM*, 5, 558–562.
- Kelley, J.** (1960), “The Cutting Plane Method for Convex Programs”, *Journal of the Society for Industrial Applied Mathematics*, 9, 703–712.
- Kojadinovic, I.** (2004), “Estimation of the Weights of Interacting Criteria from the Set of Profiles by means of Information-theoretic Functionals”, *European Journal of Operational Research*, 155, 741–751.
- Koppen, M.** (1995), “Random Utility Representation of Binary Choice Probabilities: Critical Graphs yielding Critical Necessary Conditions”, *Journal of Mathematical Psychology*, 39, 21–39.
- Kramer, S., Widmer, G., Pfahringer, B. & Degroeve, M.** (2000), “Prediction of Ordinal Classes Using Regression Trees”, *Fundamenta Informaticae*, 24, 1–15.
- Laanckriet, G., Cristianini, N., Bartlett, P., El Gaoui, L. & Jordan, M.** (2004), “Learning the Kernel Matrix with Semidefinite Programming”, *Journal of Machine Learning Research*, 5, 27–72.
- Lehmann, L.** (1975), “Nonparametrics: Statistical Methods based on Ranks”, Holden Day.
- Lievens, S., De Baets, B. & Cao-Van, K.** (2008), “A Probabilistic Framework for the Design of Instance-Based Supervised Ranking Algorithms in an Ordinal Setting”, *Annals of Operations Research*, 163, 115–142.
- Lin, H. & Li, L.** (2006), “Large-Margin Thresholded Ensembles for Ordinal Regression: Theory and Practice”, *Lecture Notes in Computer Science. Proceedings of the International Conference on Algorithmic Learning Theory*, Barcelona, Spain, 4264, 319–333.
- Luce, R. & Suppes, P.** (1965), “Handbook of Mathematical Psychology”, Wiley, chapter Preference, Utility and Subjective Probability, 249–410.
- Mao, R., Wang, X., Jr., E. S., Frelin, L., Ting, J., Kim, H. D. J., Ruczinski, I., Downey, T. & Pevsner, J.** (2005), “Primary and Secondary Transcriptional Effects in the Developing Human Down Syndrome Brain and Heart”, *Genome Biology*, 6, 117.

-
- Marichal, J., Meyer, P. & Roubens, M.** (2005), "Sorting Multi-Attribute Alternatives: The TOMASO Method", *Computers & Operations Research*, 32, 861–877.
- McCullagh, P.** (1980), "Regression Models for Ordinal Data", *Journal of the Royal Statistical Society, Series B*, 42 (2), 109–142.
- Mousseau, V.** (2005), "A General Framework for Constructive Learning Preference Elicitation in Multiple Criteria Decision Aid", Technical report, LAMSADE, Université Paris-Dauphine, France.
- Mousseau, V., Dias, L. & Figueira, J.** (2006), "Dealing with inconsistent judgments in multiple criteria sorting", *4OR*, 4, 145–158.
- Mousseau, V., Figueira, J. & Naux, J.** (2001), "Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results", *European Journal of Operational Research*, 130 (2), 263–275.
- Mousseau, V. & Slowinski, R.** (1998), "Inferring an ELECTRE TRI Model from Assignment Examples", *Journal of Global Optimization*, 12, 157–174.
- Nakas, C. & Yiannoutsos, C.** (2004), "Ordered Multiple-Class ROC Analysis with Continuous Measurements", *Statistics in Medicine*, 22, 3437–3449.
- Öztürk, M., Tsoukiàs, A. & Vincke, P.** (2005), "Preference Modelling", in : J. Figueira, S. Greco & M. Ehrgott, eds., *Multiple Criteria Decision Analysis. State of the art surveys*, 27–71, Springer-Verlag.
- Perny, P.** (1992), "Modelisation, Agrégation et Exploitation de Préférences Floues dans une Problématique de Rangement", PhD-thesis, Université Paris-Dauphine.
- Pizarro, J., Guerrero, E. & Galindo, P.** (2002), "Multiple Comparison Procedures Applied to Model Selection", *Neurocomputing*, 48, 155–173.
- Platt, J., Cristianini, N. & Shawe-Taylor, J.** (2000), "Large Margin DAGs for Multiclass Classification", *Advances in Neural Processing Systems*, 12, 547–553.
- Potharst, R. & Bioch, J.** (2000), "Decision Trees for Ordinal Classification", *Intelligent Data Processing*, 4 (2).

- Pourdeyhimi, B., Xu, B. & Nayernouri, A.** (1994), “Evaluating Carpet Appearance Loss: Pile Lay Orientation”, *Textile Research Journal*, 64, 130–135.
- Provost, F. & Fawcett, T.** (2001), “Robust Classification for Imprecise Environments”, *Machine Learning*, 42, 203–231.
- Radlinski, F. & Joachims, T.** (2007), “Active Exploration for Learning Rankings from Clickthrough Data”, in : *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, U.S.A., 570–579.
- Rajaram, S. & Agarwal, S.** (2005), “Generalisation Bounds for k -partite Ranking”, in : *Proceedings of the NIPS Workshop on Learning to Rank*, Vancouver, Canada, 18–23.
- Rennie, J. D. & Srebro, N.** (2005), “Loss Functions for Preference Levels: Regression with Discrete, Ordered Labels”, in : *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, Edinburgh, Scotland, 180–186.
- Rifkin, R. & Klautau, A.** (2004), “In Defense of One-versus-All Classification”, *Journal of Machine Learning Research*, 5, 101–143.
- Roy, B.** (1991), “The outranking approach and the foundations of ELECTRE methods”, *Theory and Decision*, 31 (1), 49–73.
- Salzberg, L.** (1997), “On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach”, *Data Mining and Knowledge Discovery*, 1, 317–328.
- Schölkopf, B. & Smola, A.** (2002), “Learning with Kernels, Support Vector Machines, Regularisation, Optimization and Beyond”, The MIT Press.
- Schrijver, A.** (2003), “Combinatorial Optimization”, Springer.
- Sette, S., Kiekens, P. & Boullart, L.** (1995), “Self-Organizing Neural Nets: A New Approach to Quality Control in Textiles”, *Textile Research Journal*, 64, 196–202.
- Shashua, A. & Levin, A.** (2003), “Ranking with Large Margin Principle: Two Approaches”, in : *Proceedings of the International Conference on Neural Information Processing Systems*, Vancouver, Canada, 937–944, The MIT Press.

-
- Shawe-Taylor, J. & Cristianini, N.** (2004), “Kernel Methods for Pattern Analysis”, Cambridge University Press.
- Siew, L., Hodgson, R. & Wood, E.** (1988), “Texture Measures for Carpet Wear Assessment”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10, 92–105.
- Statnikov, A., Tsamardinos, I., Dosbayev, Y. & Aliferis, C.** (2005), “GEMS: A System for Automated Cancer Diagnosis and Biomarker Discovery from Microarray Gene Expression Data”, *International Journal of Medical Informatics*, 74, 491–503.
- Switalski, Z.** (2003), “General Transitivity Conditions for Fuzzy Reciprocal Preference Matrices”, *Fuzzy Sets and Systems*, 137, 85–100.
- Tanino, T.** (1990), “Multiperson Decision Making Models Using Fuzzy Sets and Possibility Theory”, Kluwer Academic Publishers, Dordrecht, chapter On Group Decision Making under Fuzzy Preferences, 172–185.
- Thurstone, L.** (1927), “A law of comparative judgment”, *Psychological Review*, 79, 281–299.
- Torra, V., Ferrer, J.-D., Sanz, J. M. & Ng, M.** (2006), “Regression for Ordinal Variables without Underlying Continuous Variables”, *Information Sciences*, 176, 465–474.
- Tsochantaridis, Y., Joachims, T., Hofmann, T. & Altun, Y.** (2005), “Large Margin Methods for Structured and Independent Output Variables”, *Journal of Machine Learning Research*, 6, 1453–1484.
- Tutz, G. & Hechenbichler, K.** (2004), “Aggregating Classifiers with Ordinal Response Structure”, *Journal of Statistical Computation and Simulation*, 75 (5).
- Tversky, A.** (1998), “Preference, Belief and Similarity, selected writings”, MIT Press.
- Van De Walle, B., De Baets, B. & Kerre, E.** (1998), “Characterizable Fuzzy Preference Structures”, *Annals of Operations Research*, 80, 105–136.
- Van Steenlandt, W., Collet, D., Sette, S., Bernard, P., Luning, R., Teze, L., Bohland, H. & Schulz, H.** (1996), “Automatic Assessment of Carpet Wear Using Image Analysis and Neural Networks”, *Textile Research Journal*, 66, 555–561.

- Vapnik, V.** (1995), “The Nature of Statistical Learning Theory”, Springer-Verlag.
- Varma, S. & Simon, R.** (2006), “Bias in Error Estimation when Using Cross-Validation for Model Selection”, *Bioinformatics*, 7, 91.
- Vincke, P.** (1992), “Multicriteria Decision-Aid”, Wiley.
- Waegeman, W., Cottyn, J., Wyns, B., Boullart, L., De Baets, B., Van Langenhove, L. & Detand, J.** (2008a), “Classifying Carpets based on Laser Scanner Data”, *Engineering Applications of Artificial Intelligence*, 21, 907–918.
- Waegeman, W., De Baets, B. & Boullart, L.** (2006), “A Comparison of Different Performance Measures for Ordinal Regression”, in : *Proceedings of the ICML Workshop on ROC Analysis*, Pittsburgh, PA, USA, 63–69.
- Waegeman, W., De Baets, B. & Boullart, L.** (2007), “Learning a Layered Graph with a Maximal Number of Paths Connecting Source and Sink”, in : *Proceedings of the ICML Workshop on Constrained Optimization and Structured Output Spaces*, Corvallis, OR, USA.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008b), “A Graph-theoretic Approach for Reducing One-versus-one Multi-class Classification to Ranking”, in : *Proceedings of the International Workshop on Mining and Learning with Graphs*, Helsinki, Finland.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008c), “Integrating Expert Knowledge into Kernel-based Preference Models”, in : *Proceedings of the ECML Workshop on Preference Learning*, Antwerp, Belgium.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008d), “Kernel-based Learning Methods for Preference Aggregation”, *4OR*, in press.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008e), “Learning Layered Ranking Functions with Structured Support Vector Machines”, *Neural Networks*, in press.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008f), “On the Scalability of Ordered Multi-class ROC Analysis”, *Computational Statistics and Data Analysis*, 52, 3371–3388.
- Waegeman, W., De Baets, B. & Boullart, L.** (2008g), “ROC Analysis in Ordinal Regression Learning”, *Pattern Recognition Letters*, 29, 1–9.

-
- Wang, Y.-M. & Parkan, C.** (2005), "Multiple Attribute Decision Making Based on Fuzzy Preference Information on Alternatives: Ranking and Weighting", *Fuzzy Sets and Systems*, 153, 331–346.
- Weston, J. & Watkins, C.** (1998), "Multi-class Support Vector Machines", Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, Egham TW20 OEX, UK.
- Wood, E. & Hodgson, R.** (1989), "Carpet Texture Measurement Using Image Analysis", *Textile Research Journal*, 59, 1–12.
- Wu, F., Lin, C. & Weng, R.** (2004), "Probability Estimates for Multi-Class Support Vector Machines by Pairwise Coupling", *Journal of Machine Learning Research*, 5, 975–1005.
- Xu, B.** (1994), "Assessing Carpet Appearance Retention by Image Analysis", *Textile Research Journal*, 64, 497–509.
- Xu, J., Cao, Y., Li, H. & Huang, Y.** (2006), "Cost-sensitive learning of SVM for ranking", in : *Proceedings of the 17th European Conference on Machine Learning*, Berlin, Germany, 833–840.
- Yan, L., Dodier, R., Mozer, M. & Wolniewicz, R.** (2003), "Optimizing Classifier Performance Via an Approximation to the Wilcoxon-Mann-Whitney Statistic", in : *Proceedings of the International Conference on Machine Learning*, Washington DC., USA, 848–855.
- Yu, S., Yu, K., Tresp, V. & Kriegel, H.** (2006), "Collaborative Ordinal Regression", in : *Proceedings of the International Conference on Machine Learning*, Pittsburgh, PA, 1089–1096.
- Zhang, J.** (2004), "Binary Choice, Subset Choice, Random Utility, and Ranking: A Unified Perspective using the Permutahedron", *Journal of Mathematical Psychology*, 48, 107–134.