Studies in Learning Monotonic Models from Data

SIKS Dissertation Series No. 2014-01

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research reported in this thesis was supported by the Netherlands Organisation for Scientific Research (NWO grant no. 612.066.621).

Printed by Ipskamp Drukkers, the Netherlands.

ISBN: 978-90-393-7093-3

Studies in Learning Monotonic Models from Data

Studies in het Leren van Monotone Modellen van Data

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op maandag 10 februari 2014 des middags te 12.45 uur

door

Nicola Barile

geboren op 21 juli 1971 te Bari, Italië

Promotor: Prof.dr. A.P.J.M. Siebes
Co-promotor: Dr. A.J. Feelders

Contents

Acknowledgements

First and foremost I would like to thank my supervisors Prof. Arno Siebes and Dr. Ad Feelders for giving me the opportunity to fulfil my lifelong ambition of pursuing a PhD. Dr. Feelders in particular has taught me how good research in Data Mining is done. The joy and enthusiasm he has for his research were contagious and motivational, and, through his thorough supervision, he has made my experience as a PhD student especially productive and stimulating.

My heartfelt thanks also go out to the current and past members of the Algorithmic Data Analysis research group I had the pleasure to meet and work with, for they have contributed immensely to my experience at Utrecht University both personally and professionally. The group has been a source of friendships as well as of good advice and collaboration.

I would also like to thank the members of the reading committee, that is Prof. Eyke Hüllermeier, Prof. Peter Flach, Prof. Linda van der Gaag, Prof. Donato Malerba, and Prof. Hennie Daniels for the favourable opinions they have expressed on this thesis and on the research of which it is the end result.

Finally, I would like to thank to my family and to my friends from around the world, in particular to Dr. Shari Lawrence Pfleeger and to Dr. Charles P. Pfleeger, for all their love and encouragement. But most of all I would love to thank my loving, supportive, encouraging, and patient girlfriend Sonia for the unconditional love and faithful support she has given me through all the stages of my doctoral studies.

Abstract

This thesis describes a number of new data mining algorithms which were the result of our research into the enforcement of monotony restrictions when learning (mostly non-parametric) models from data.

Not only can judicious use of domain knowledge improve the predictive accuracy of data mining algorithms but also, crucially, models that are consistent with the knowledge of domain experts will be accepted and adopted much earlier than models that are not. Unfortunately, domain knowledge that is most of times available is often informal and poorly structured, which makes its use in practice fraught with difficulty.

The knowledge of the existence an ascending or descending relationship between predictor variables and the variable to predict represents a notable exception. Moreover, in many applications domain experts can specify such monotonic relationships with relative ease and reliability based on their knowledge and experience. It is known, for instance, that smoking and being overweight increase the risk of cardiovascular disease (an increasing relationship); on the other hand, it is likely that a higher income reduces the probability of default on a loan (a decreasing relationship).

The experiments described in this thesis show that the predictive power of our new data mining algorithms is comparable to, or sometimes even better than, that of their non-monotonic counterparts. This is obtained at a limited additional computational cost. All in all, it is possible to conclude that enforcing monotony restrictions when applicable is practically achievable and has an advantageous effect on the quality of the models produced.

Samenvatting

Dit proefschrift beschrijft een aantal nieuwe data mining algoritmen die zijn voortgekomen uit ons onderzoek naar het afdwingen van monotonie-restricties bij het – veelal niet-parametrisch – leren van modellen uit data.

Het oordeelkundig gebruik van domeinkennis kan de voorspelkracht van data mining algoritmen verbeteren, maar ten minste zo belangrijk is dat modellen die in overeenstemming zijn met de kennis van domeindeskundigen veel eerder geaccepteerd en gebruikt zullen worden dan modellen die hiermee in tegenspraak zijn. Helaas is beschikbare domeinkennis vaak informeel en slecht gestructureerd waardoor zij moeilijk gebruikt kan worden.

Kennis van een stijgend of dalend verband tussen voorspellende variabelen en de te voorspellen variabele vormt hierop een positieve uitzondering. Bovendien geldt voor veel toepassingsgebieden dat domeindeskundigen op grond van hun kennis en ervaring dergelijke monotone verbanden relatief eenvoudig en betrouwbaar kunnen specificeren. Zo is het bijvoorbeeld bekend dat roken en overgewicht de kans op hart- en vaatziekten vergroot (een stijgend verband) terwijl een hoger beschikbaar inkomen de kans op wanbetaling bij een lening zal verkleinen (een dalend verband).

De experimenten die in dit proefschrift worden beschreven tonen aan dat de voorspelkracht van onze nieuwe data mining algoritmen vergelijkbaar is met, of soms zelfs beter is dan de voorspelkracht van hun niet-monotone tegenhangers. Dit gaat slechts ten koste van een beperkte hoeveelheid extra rekenwerk. Al met al mogen we concluderen dat het afdwingen van voorhanden monotonie-restricties praktisch haalbaar is, en een gunstig effect heeft op de kwaliteit van de geproduceerde modellen.

Chapter 1

Introduction

The goal of *predictive data mining* is to determine the value of a target output for a given population. One way to tackle this kind of problem is represented by *supervised pattern recognition*.

Assuming the existence of a functional relationship between the domain of the predictors (the *feature space* $\mathcal{X}$) and the domain of the target attribute (the *target space* $\mathcal{Y}$), a *supervised pattern recognition* algorithm observes the training data set (the *training phase*) in order to learn from the examples it comprises a model which approximates the unknown functional relation also for individuals not included in the training data set (the training data set usually does not cover the whole population). The learning process can be seen as that of choosing a function (a *hypothesis*)

$$h : \mathcal{X} \to \mathcal{Y}$$

from a fixed class of functions $\mathcal{H}$ (the *hypothesis space*) which best fits the training data set according to a given criterion.

Without assumptions about the nature of the functional relation being approximated, an inductive learning task cannot be solved exactly because unseen individuals might have an arbitrary output value. Such assumptions represent the algorithm's *inductive bias* [83] and are based on the available prior *domain knowledge*, that is the auxiliary information about the learning task which can be used to guide the learning process [110]. Not only does domain knowledge provide inductive bias, but it can also facilitate other

aspects of an inductive learning task such as the preprocessing of training examples, the initiation of the hypothesis space, the initiation of the start point of the convergence, the direction of convergence etc. To sum it up, prior domain knowledge is valuable to be incorporated into a predictive algorithm. Unfortunately, though, domain knowledge is primarily informal and ill structured, which usually makes its exploitation in inductive learning systems challenging.

A common form of prior knowledge which is frequently available in many application areas is represented by the *monotonicity*, either increasing or decreasing, of the relation between the target variable and the predictor variables. This kind of prior knowledge is usually easy to obtain in a reliable way and is based on the intuition and experience of domain experts. It is known, for instance, that smoking and being overweight increase the risk of cardiovascular disease (a *monotonically increasing* relationship); on the other hand, it is likely that a higher income reduces the probability of default on a loan (a *monotonically decreasing* relationship); this a-priori knowledge about the qualitative form of the model being learned should be built into the estimation technique. Such qualitative features do not necessarily lead to better rates of convergence but help in the interpretation of the obtained curves [57]. Another example of monotonicity constraints in medicine can be found in [95].

1.1 The Need for Monotonicity

The need to integrate monotonicity in the results produced by a data mining algorithm can arise for many different reasons.

1.1.1 Monotonicity as a Property of the Target Function

In many cases, monotonicity represents a property of the target function being learned which is a consequence of intrinsic characteristics of the modelled phenomenon or whose presence can be assumed based on common sense or experience.

In economic and financial applications involving demand, supply, and pricing, monotonic functions are very common. Economic theory states

that, caeteris paribus, people tend to buy less of a product if its price increases; as a consequence, price elasticity of demand should be negative. The strength of this relationship and the precise functional form are however rarely dictated by economic theory. The usual assumption that such relationships are linear are imposed mostly for reasons of mathematical convenience. Other well-known examples from economics are the positive dependence of labour wages on age and on education [85] and the so-called *hedonic price models*, where the price of a consumer good depends on a set of characteristics for which a valuation exists [58]. In house pricing, for instance, the price of a house increases with the house area and decreases with the distance to the city centre. Gamarnik [52] gives an example in option pricing where the price of an American call option is an increasing function of the duration and the price of the underlying asset and a decreasing function of the strike price.

Hellerstein [61] describes a statistical approach to diagnosing performance problems of computer systems which exploits knowledge of monotonic relationships such as the fact that paging delays increase with the number of logged-on users.

Karpf [70] makes a strong case for the incorporation of monotonicity constraints in inductive modelling of legal systems for the development of example-based expert systems for administrative judgements. He analysed four legal decision support systems in the area of administrative law which were constructed by learning from formalised (parts of) cases from the practice of administration. The inductive systems discussed however lacked the possibility to enforce monotonicity constraints, and this was identified as a serious legal and technical shortcoming since most of the factors in the analysed systems are, in fact, monotonic.

An example of monotonicity constraints in operations research can be found in [80].

We end this section with an example from computer science. *Record matching* is the task of identifying records that match the same real-world entity. Each pair of records can be represented as a vector of similarity measures between them (e.g. each measuring the similarity between the two records on their corresponding fields); it stands to reason that if a pair of records (r_1, s_1) matches and another pair (r_2, s_2) scores at least

as high on all similarity measures, then that pair should match as well. Chaudhuri et al. [25] empirically observed that a monotonicity property is indeed satisfied in most record matching scenarios. The same argument can be used to require that scoring functions for top-k selection queries in data bases should be monotonic [26].

1.1.2 Monotonicity as a Requirement for the Learned Model

In many cases, monotonicity is not just an intrinsic property of the target concept being learned but a property that the predictive algorithm is required to enforce. Predictions which do not respect the expected monotonicity condition are likely to be rejected by a human expert evaluating them.

For instance, when the results of a predictive algorithm are used to make critical choices such as acceptance/rejection decisions, it may be required that the algorithm return monotonic predictions for reasons of fairness or liability [37]. An example would be a model used by a university to support its admission process by ranking applicants. It would be strange if, given two applicants a and b, a scored at least as well as b on all admission criteria but a were ranked less favourably than b and, consequently, had their application rejected. Such a non-monotonic admission rule would clearly be unacceptable. Similar considerations also apply to most other selection procedures, such as job selection and credit loan approval.

Another reason that might lead to the rejection of the results of a predictive algorithm is its lack of understandability or justifiability, for human decision makers require in general that a model be easy to understand and will not accept predictions returned by black box models such as neural networks or very complex decision trees [29]. Monotonicity constraints often play a big role in the acceptability of automatically-generated predictions.

For instance, Pazzani et al. report in [88] on the evaluation of the potential for monotonicity constraints to bias machine learning algorithms in order to learn rules which are both accurate and meaningful. They employ the FOCL algorithm, which is an extension of the purely inductive FOIL algorithm. While the latter generates rules exclusively from the training data,

the former also uses domain knowledge to generate additional specifications but then selects one hypothesis among all of these candidate specifications based on performance over the data. Two data sets from the problems of screening for dementia and assessing the risk of mental retardation were collected and a rule learning system with and without monotonicity constraints was run on each. An example of monotonicity constraint given by the authors is that the probability that a patient is mentally impaired depends positively on the number of errors made in recalling their address; the threshold which best distinguishes positive examples from negative examples according to the information gain criterion is selected. Rules learned with monotonicity constraints were at least as accurate as those learned without such constraints. More crucially, the authors also show that rules learned with monotonicity constraints were significantly more acceptable to medical experts than rules learned without such restrictions. They argue that one factor influencing the acceptability of learned knowledge is consistency with existing medical knowledge; they also show that most existing knowledge discovery algorithms typically violate such knowledge, resulting in concepts which are not coherent when taken in the larger context of other related problems.

Finally, also in risk analysis it is often required that predictions be monotonic with respect to the decision variables involved.

1.1.3 Monotonicity as Prior Knowledge for Probability Estimation

Quantifying a Bayesian network, namely assessing the probability distributions for each of the network's variables conditional on their direct predecessors in the network's directed graph, can be easy in domains where large data collections are available. On the other hand, when the amount of data available is not very large, the subsets from which probabilities are estimated can be empty or too small to allow for meaningful values, so the probabilities estimates obtained are deemed as unreliable. Therefore, when this happens, it is commonplace not to use the data available at all but, instead, to rely on the knowledge and experience of domain experts as the source of probabilistic estimation information.

Human experts tend to feel uncomfortable expressing their knowledge and experience in terms of probabilities and tend to provide imperfectly calibrated assessments. On the other hand, they are typically able to express probabilistic information of a semi-numerical or qualitative nature with relative conviction and clarity and with less cognitive effort; for example, they often can easily indicate which of two probabilities is smaller [32]. In addition to requiring less cognitive effort and time to gather, such relative judgements tend to be more reliable and more properly calibrated than direct numerical assessments [82]. Based the above observations, Helsper et al. designed in [41, 62] a method for obtaining the probabilities required for a Bayesian network which combined the qualitative information about probabilities obtained from domain experts with the numerical information available.

Feelders et al. demonstrate in [43] that expert knowledge about the signs of the influences between the variables in a Bayesian network can be used to improve the probability estimates computed from small samples of data from every-day problem solving in the domain. They also show how these signs impose order constraints on the probabilities required for the network. Druzdzel et al. propose in [32] a method for the elicitation of probabilities from a domain expert which is non-invasive and which accommodates whatever probabilistic information the expert is willing or capable to state. In their approach, all available information, whether qualitative or quantitative in nature, is expressed in a canonical form consisting of inequalities expressing monotonicity constraints on the hyperspace of possible joint probability distributions. This information is then used to derive second-order probability distributions over the desired probabilities.

1.2 Research Motivation and Goals

Besides being easier to understand and more consistent with the prior knowledge of domain experts, monotonic models ofter outperform better their non-monotonic counterparts when monotonicity is a valid assumption for the populations to which they are applied. For instance, Velikova et al. show in [29] that monotonic decision trees derived from the relabelled data perform better compared to those derived from the raw data. This is mainly

due to the fact that monotonic models have a tendency to suppress overfitting [29]. Moreover, enforcing monotonicity removes noise and resolves inconsistencies, which also may result in better predictive accuracy [84, 98].

The frequency with which monotonicity constraints occur in problems addressed with data mining and machine learning and the ease and reliability with which such constraints can be elicited have prompted the development of learning algorithms capable of enforcing such constraints in a justified manner. Examples can be found in the fields of instance-based classification [21, 24, 34, 35, 72, 76, 77, 96], rough-set-based classification [17, 53, 55, 56, 71], classification trees [14, 29, 37, 40, 90, 91, 105], neural networks [4, 98, 99, 108], Bayesian networks [2, 41–44, 62, 106], and rule learning [30, 73, 84].

The goal of our research was studying the incorporation of monotonicity constraints into new algorithms aimed at performing various types of predictive data mining tasks and resulted in the algorithms illustrated in this thesis, which are all based on monotonicity-preserving models derived from labelled data in a non-parametric fashion. Parametric models assume a particular functional form beforehand and are completely specified by a set of parameters; the objective of training in this case is to find appropriate values for the parameters by optimising a loss function on the training data; *non-parametric* models, on the other hands, do not assume a fixed functional form. Either approach has a drawback to it: the price to pay for parametric fitting is the chance of severe misspecification resulting in too high a model bias, while non-parametric models may result in more variable estimates especially for small sample sizes. We have opted for the non-parametric approach for the two reasons. Firstly, parametric models might be too restricted or too low-dimensional to fit unexpected features; on the contrary, by not projecting the observed data into a Procrustean bed of a fixed parametrisation (e.g. by fitting a linear model to the data), non-parametric models offer greater flexibility. Secondly, we wanted to avoid making assumptions about the function being learned other than monotonicity as they might be wrong or hard to justify and might reduce the support of problems that our research would be applicable to.

The main criterion used to evaluate our work was the quality of the models produced by our algorithms; on the other hand, it was important

to determine the quality of the algorithms developed in terms of efficiency. Therefore, the algorithms were applied to both ad-hoc generated artificial data sets and real-world data sets (see appendix A for details on the latter) in order to compare the out-of-sample predictive performance to that of other relevant data mining algorithms, either monotonicity preserving or not. The goal was to ensure we were not obtaining monotonic models at the expense of predictive accuracy. If a monotonic model is really required, then a small increase of the error might be acceptable, but clearly this should be within reasonable limits. On the other hand, if the problem is really monotonic, then we might even expect an improvement predictive performance.

1.3 Thesis Outline

Although it revolves around three algorithms which were the result of our research and which have been the subject of peer-reviewed publications [10–13], this thesis does not merely consist of a collection of separate publications; instead, it includes three main chapters, each devoted to one of the algorithms. Each of these main chapters begins with an introduction which establishes the main ideas behind the algorithm it discusses; the chapter then proceeds with a presentation of related earlier work, followed by a detailed description of the algorithm; finally, the chapter ends with an empirical evaluation of the algorithm, followed by conclusions and ideas for future work.

The following is a list of the chapters comprising this thesis together with a summary of their contents:

- In Chapter 2 common notation and some basic concepts and definitions, especially about monotonicity and probabilities, are introduced. We also introduce the statistical problem of the *isotonic regression* and the algorithm we have actually employed in our research, for the isotonic regression lies at the heart of the algorithms presented in this thesis.

- In Chapter 3 the MOCA monotonic classification algorithm is introduced, which attempts to minimise the mean absolute prediction er-

ror for classification problems with ordered class labels. The algorithm first learns a classifier with minimum mean absolute error (see equation 2.3.7) on the training sample; it then uses an interpolation scheme to predict the class label for attribute vectors not present in the training data. We compare MOCA to the related OSDL algorithm, both on artificial and real-world data sets, and show that MOCA often outperforms OSDL with respect to mean absolute prediction error.

- In Chapter 4 we present an algorithm for instance ranking called MIRA, which learns a monotonic ranking function based on the posterior class probabilities of a set of labelled training examples; monotonicity is enforced by applying the isotonic regression to the basic probability estimates, and these new estimates are combined with logistic regression in an attempt to remove unwanted rank equalities. An interpolation scheme is then used to rank new data points. Through experiments we show that MIRA produces ranking functions having predictive performance comparable to that of a state-of-the-art instance ranking algorithm, which makes MIRA a valuable alternative when monotonicity is desired or mandatory.

- In Chapter 5 we propose two algorithms which exploit monotonicity constraints for active learning in ordinal classification in two different settings. The basis of our methods is the observation that if the class label of an object is given, then monotonicity constraints may allow us to infer the labels of other objects; for instance, from knowing that loan applicant a is rejected, it can be inferred that all applicants that score worse than a on all criteria should be rejected as well. We also propose two heuristics to determine good query points; these heuristics make a selection based on the potential of a point to allow the labels of other points to be inferred. The introduced algorithms, each implemented with the proposed heuristics, evaluated on artificial and real-world data sets to study their performance. Empirical results show that exploitation of monotonicity constraints can be beneficial to active learning.

- Finally, in Chapter 6 we draw conclusions on our research, summarise

the contributions it has made, and discuss possible future developments in the work that we have done.

Chapter 2

Preliminaries

In this chapter we introduce some common notation and a few basic concepts and definitions which are used throughout this thesis. Moreover, we introduce the statistical problem of the *isotonic regression* and describe the algorithm we employed to solve it, for the isotonic regression lies at the heart of some of the algorithms presented in this thesis.

2.1 Definitions and Notation

2.1.1 Topology

Definition 1 (Partial Order)**.** A binary relation $\leq$ defined on a non-empty set P is called a *partial order* if the following three conditions are satisfied for all $a, b, c \in P$:

1) $a \leq a$ (*reflexivity*)
2) if $a \leq b$ and $b \leq a$, then $a = b$ (*antisymmetry*)
3) if $a \leq b$ and $b \leq c$, then $a \leq c$ (*transitivity*)

A *partially-ordered set* (or, briefly, a *poset*) consists of a set P together with a *partial order* defined on it. Formally, a partially-ordered set is defined as an ordered pair

$$X = (P, \leq),$$

where P is called the *ground set* of X and $\leq$ is the partial order of X.

Notation. For the sake of convenience, in this thesis a poset $X = (P, \leq)$ shall simply be referred to by specifying its ground set P when its ordering relation needs not be specified; moreover, especially in order to distinguish it from the partial order of another poset, we shall sometimes denote the order relation of the poset as $\leq_P$.

Definition 2 (Inverse Order)**.** The *inverse* or *dual* of a partial order relation $\leq$ on a non-empty set P is the binary relation $\geq$ on P defined as

$$a \geq b \iff a \leq b.$$

The inverse of a partial order relation is reflexive, transitive, and anti-symmetric; therefore, it is also a partial order relation.

Definition 3 (Dual Poset)**.** The *(order) inverse* or *(order) dual* of a partially-ordered set $X = (P, \leq)$ is the poset having P as its ground set and the inverse of $\leq$ as its order relation, namely

$$X' = (P, \geq).$$

Definition 4 (Strict Partial Order)**.** A binary relation $<$ defined on a non-empty set P is called a *strict partial order* if the following three conditions are satisfied for all $a, b, c \in P$:

1') $\neg(a < a)$ (*irreflexivity*)
2') if $a < b$, then $\neg(b < a)$ (*asymmetry*)
3') if $a < b$ and $b < c$, then $a < c$ (*transitivity*)

A *strictly-ordered set* consists of a set P together with a *strict order* defined on it and is denoted as $(P, <)$.

There is a 1-to-1 correspondence between non-strict and strict partial orders. In fact, if $\leq$ is a partial order, then the corresponding strict partial order $<$ is the *reflexive reduction* $\leq - \{(x, x) \mid x \in P\}$, namely

$$(a < b) \Leftrightarrow (a \leq b \text{ and } a \neq b)\,.$$

Vice versa, given a strict partial order $<$, the corresponding non-strict partial order $\leq$ is the *reflexive closure* $< \cup \{(x, x) \mid x \in P\}$, namely

$$(a \leq b) \Leftrightarrow (a < b \text{ or } a = b)\,.$$

Definition 5 (Comparable Pair)**.** Given a poset $(P, \leq)$, $(a, b) \in P \times P$ is called a *comparable pair* if

$$a \leq b \text{ or } b \leq a.$$

Otherwise, (a, b) is called an *incomparable pair*.

Definition 6 (Transitive Closure)**.** The *transitive closure* of a binary relation R on a set X is the transitive relation R^+ on X such that R^+ contains R and R^+ is minimal with respect to the inclusion relationship.

The transitive closure of a binary relation R always exists; it is R itself if it is transitive, otherwise it is a different relation.

Definition 7 (Transitive Reduction)**.** The *transitive reduction* of a binary relation R on a set X is a minimal relation R' on X such that the transitive closure of R' coincides with the transitive closure of R.

If the transitive closure of R is antisymmetric and finite, then R' exists and is unique. On the other hand, existence or uniqueness are not guaranteed in general.

Definition 8 (Covering Relation)**.** Given a poset $(P, \leq)$ and the reflexive reduction $<$ of $\leq$, the *covering relation* of P is the binary relation $\lessdot$ defined as:

$$a \lessdot b \quad \overset{\text{DEF}}{\Longleftrightarrow} \quad a < b \wedge \nexists\, c \in P : a < c < b.$$

The covering relation of a partially-ordered set $(P, \leq)$ is therefore the binary relation holding between comparable pairs that are immediate neighbours. Moreover, if P is finite, then it can be proved that the covering relation coincides with the transitive reduction of $\leq$. Covering relations form the basis for a useful graphical representation of the partial order they originate from.

Definition 9 (Order Graph)**.** The *order graph* of a poset $(P, \leq)$ is the directed graph $D = (V, A)$ where

1) $V = P$
2) $a \in A \iff a = (x, y) \wedge x \lessdot y$

Fig. 2.1 depicts the order graph for a poset $(P, \leq)$, where $P = \{a, b, c, d, e\}$ and $\lessdot = \{(a, b), (a, c), (a, d), (c, e)\}$.

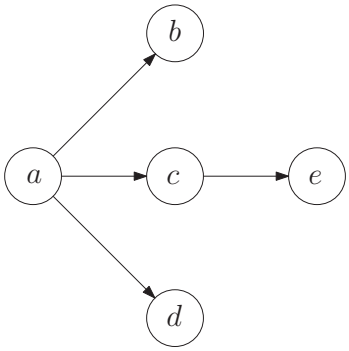

Figure 2.1: Order graph for a poset $(P, \leq)$, where $P = \{a, b, c, d, e\}$ and $\lessdot = \{(a, b), (a, c), (a, d), (c, e)\}$.

Definition 10 (Total Order)**.** A binary relation defined on a non-empty set P is called a *total order* and usually denoted as $\leq$ if the following three conditions are satisfied for all $a, b, c \in P$:

1) $a \leq b$ or $b \leq a$ (*totality*)
2) if $a \leq b$ and $b \leq a$, then $a = b$ (*antisymmetry*)
3) if $a \leq b$ and $b \leq c$, then $a \leq c$ (*transitivity*)

A set paired with a total order is called a *totally*, a *linearly ordered set* or a *chain*. At the opposite extreme, a poset in which no two distinct elements are comparable is called an *antichain*.

Totality, which indicates the fact that any pair of elements in P is under the relation, implies reflexivity; therefore, a total order is also a partial order. As a consequence, all definitions and notations concerning partially-ordered sets also apply to total orders and to totally-ordered sets. For instance, the definition of a *strict total order* is analogous to Definition 4.

Definition 11 (Upper Set)**.** An *upper set* of a poset $(P, \leq)$ is a set $U \subseteq P$ with the property that, for every $x, y \in P$, if $x \in U$ and $x \leq y$ then $y \in U$. The definition of a *lower set* is given dually.

Notation. We shall denote the collections of all upper sets and of all lower sets of a poset $(P, \leq)$ as $\mathcal{U}_P$ and $\mathcal{L}_P$ respectively.

Definition 12 (Upset)**.** Given a poset $(P, \leq)$ and a subset $A \subset P$, the *upset* of A is the upper set $\uparrow A = \{x \in P \mid \exists a \in A : a \leq x\}$. The definition of the *downset* $\downarrow A$ of A is given dually.

Notation. If $A = \{a\}$, then we denote the *upset* of a as

$$\uparrow a = \{x \in P \mid a \leq x\}.$$

The notation for the *downset* $\downarrow a$ of a is given dually. Moreover, we shall write $u(a)$ to denote the cardinality of $\uparrow a$ and $d(a)$ to denote the cardinality of $\downarrow a$.

Proposition 1 (Properties of Upper and Lower Sets)**.** *Given a poset $(P, \leq)$, the following properties are true for all $x, x' \in P$, $A, B \subseteq P$:*

1. *P is an upper set (resp. a lower set) of itself.*
2. *The complement of an upper set of P is a lower set, and vice versa.*
3. *The intersection and the union of upper sets (resp. lower sets) of P is an upper set (resp. a lower set).*
4. *$\uparrow A$ and $\downarrow A$ are, respectively the smallest upper set and lower set of P containing A.*
5. *If $A = \uparrow A$, then A is an upper set of P, while if $A = \downarrow A$, then A is a lower set of P.*
6. *If $A \subseteq B$, then $\downarrow A \subseteq \downarrow B$, and $\uparrow A \subseteq \uparrow B$.*
7. *If $x \leq x'$, then $\downarrow x \subseteq \downarrow x'$, $\uparrow x' \subseteq \uparrow x$.*

Definition 13 (Isotonic Function)**.** A function f between two partially-ordered sets $(P, \leq)$ and $(Q, \leq)$ is *isotonic* if

$$a \leq_P b \Longrightarrow f(a) \leq_Q f(b). \tag{2.1.1}$$

Definition 14 (Antitonic Function)**.** A function f between two partially-ordered sets $(P, \leq)$ and $(Q, \leq)$ is *antitonic* if

$$a \leq_P b \Longrightarrow f(a) \geq_Q f(b). \tag{2.1.2}$$

Definition 15 (Monotonic Function)**.** An isotonic or antitonic function is called *monotonic*.

2.2 Probability Theory and Statistics

Notation. Given a random variable X, its probability distribution shall be denoted as

$$P_X, \tag{2.2.1}$$

and the values of its cumulative distribution function (cdf) as

$$F_X(x), \quad \forall x \in \mathbb{R}. \tag{2.2.2}$$

Notation. Given a discrete random variable Y, the values of its probability mass function (pmf) we shall denoted as

$$p_Y(y), \quad \forall y \in \mathbb{R}. \tag{2.2.3}$$

Proposition 2. *Given a discrete random variable Y with support $R_Y = \{y_i\}$, then for all $x \in \mathbb{R}$*

$$F_Y(x) = \sum_{y_i \leq x} p_Y(y_i). \tag{2.2.4}$$

In particular, $F_Y(x) = 0$ if $x < y_1$, and $F_Y(x) = 1$ if $card(R_Y) = q$ and $y_q \leq y$.
On the other hand,

$$P_Y(y_i) = F_Y(y_i) - F_Y(y_i^-) = \begin{cases} F_Y(y_i), & \text{if } i = 1; \\ F_Y(y_i) - F_Y(y_{i-1}) & \text{otherwise.} \end{cases} \tag{2.2.5}$$

where $F_Y(y_i^-) = \lim_{y \to y_i^-} F_Y(y)$.

Proposition 2 describes the one-to-one correspondence between the probability mass function and the cumulative distribution function of a discrete random variable Y: the knowledge of one allows to determine the values of the other. Moreover, equation (2.2.4) shows that the cumulative distribution function of a simple random variable Y with support $R_Y = \{y_i\}_{i=1}^k$ is a step function comprising $k+1$ steps delimited by the points comprising R_Y always taking value 0 on the first step and value 1 on the last step.

Definition 16. Given a random variable X, $m \in \mathbb{R}$ is a *median* of the probability distribution P_X if

$$P(X \geq m) \geq \frac{1}{2}, \;\; P(X \leq m) \geq \frac{1}{2}. \tag{2.2.6}$$

Notation. Any (discrete) probability distribution has at least one median, but there may be more than one median. Nonetheless, we shall informally refer to '*the median*' and denote one of the medians as

$$Med_X.$$

A simple way to characterise all the medians of a distribution is provided by the following proposition. [77]

Proposition 3. *Given a random variable X with probability distribution P, $m \in \mathbb{R}$, then*

$$m \text{ is a median of } P \iff m \in [m_\ell, m_u]. \tag{2.2.7}$$

where $m_\ell = \arg\min_m P(X \leq m) \geq \frac{1}{2}$, and $m_u = \arg\max_m P(X \geq m) \geq \frac{1}{2}$.

In the case of a discrete random variable X, the interval in (2.2.7) reduces to the (ordered) set $\{m_\ell, m_\ell + 1, \ldots, m_u\}$.

Proposition 4. *Given a discrete random variable Y, $m \in \mathbb{R}$ is a* median *of the probability distribution P_Y if*

$$F_Y(m-1) \leq \frac{1}{2}, \;\; F_Y(m) \geq \frac{1}{2}. \tag{2.2.8}$$

Definition 17. Given a discrete random variable Y, $m \in \mathbb{R}$ is a *mode* of the discrete probability distribution P_Y if

$$m = \arg\max_{x \in \mathbb{R}} p_Y(x).$$

Notation. The mode of a discrete probability function is not necessarily unique as it may take the same maximum value at several points. Nonetheless, we shall informally refer to '*the mode*' and denote one of the modes as

$$Mod_Y.$$

Notation. Given a discrete random variable Y with support $R_Y = \{y_i\}$, a random vector $\mathbf{X}$, the values of the conditional probability distribution (also called the *posterior distribution*) $P_{Y|\mathbf{X}=\mathbf{x}}$ shall also be denoted as

$$P_i(\mathbf{x}), \quad \forall i \in \{1, \ldots, k\}. \tag{2.2.9}$$

The values of the steps 2 to k of the cdf $F_{Y|\mathbf{X}=\mathbf{x}}$ shall be denoted as

$$F_i(\mathbf{x}). \tag{2.2.10}$$

The conditional expectation $E(Y|\mathbf{X})$ of Y given the event $\mathbf{X} = \mathbf{x}$ shall be denoted as

$$E_Y(\mathbf{x}). \tag{2.2.11}$$

Finally, given $m \in \mathbb{R}$, the notations

$$Med_{Y|\mathbf{X}} \tag{2.2.12}$$

and

$$Mod_{Y|\mathbf{X}} \tag{2.2.13}$$

shall indicate the a median and the mode of the conditional probability distribution of Y given the event $\mathbf{X} = \mathbf{x}$ respectively.

2.3 Statistical Decision Theory

Given a set $\mathcal{A}$ of possible actions, *statistical decision theory* [16, 87] is concerned with deciding which action $a \in \mathcal{A}$ to take in the presence of statistical knowledge which allows to reduce the uncertainties involved in making this decision. The consequences of the decision taken depend on an unknown *state of nature*

$$\theta \in \Theta.$$

A first source of information used by statistical decision theory θ is represented by how probable the various values of θ are deemed to be; these *prior* probabilities are usually based on past experience, and their density, which can be either discrete or continuous, shall be denoted as

$$\pi(\theta).$$

Another source of information on θ is represented by the outcome of the statistical investigation performed on a population, which is described by the random vector $\mathbf{X}$ taking values on the feature space $\mathcal{X}$, in order to estimate the conditional density

$$f(x \,|\, \theta).$$

Moreover, statistical decision theory also takes into account knowledge of the possible consequences of the decision to be taken. This information is quantified in terms of the loss that would be incurred for each possible decision and for the various possible values of θ: if action a_l is taken and θ_l is the true state of nature, then a loss

$$L(\theta_l, a_l)$$

is incurred. Hence, it is assumed that a *loss function* $L(\theta, a)$ is defined for all tuples $(\theta, a) \in \Theta \times \mathcal{A}$.

Due to the presence of uncertainty, the actual loss is not entirely certain at the time when the decision is made. Therefore, the action taken is the action a which is "optimal" with respect to the "expected" loss of making a

decision is considered. As a first step towards formally defining this expected loss, it is necessary to establish a way to map an observation $\mathbf{x}$ of $\mathbf{X}$ to an action a which is appropriate for it.

Definition 18 (Decision Rule)**.** A *decision rule* δ is a function

$$\delta : \mathcal{X} \to A. \tag{2.3.1}$$

If $\mathbf{x}$ is the observed value of the sample information, then $\delta(\mathbf{x})$ represents the action that is taken.

We seek to evaluate, for a given $\mathbf{x}$, how much we would "expect" to lose by performing action $\delta(\mathbf{x})$ for each of the possible unknown states of nature θ.

Definition 19 (Conditional Risk Function)**.** The *conditional risk function* for a decision rule δ is the real-valued function $R^{\delta}(\mathbf{x}) : \Theta \to \mathbb{R}$ defined by

$$R^{\delta}(\mathbf{x}) = E_{\theta|\mathbf{X}}(L[\theta, \delta(\mathbf{x})]). \tag{2.3.2}$$

For all $\mathbf{x}$, the value $R^{\delta}(\mathbf{x})$ is called the *risk* for $\delta(\mathbf{x})$.

Notation. When the decision rule a risk function refers to is clear from the context, the superscript δ shall be omitted.

It is desirable to use a decision rule δ for which the risk is small. This choice, though, has to be made individually for each observation $\mathbf{x}$ that the risk is computed for.

In problems in which the state of nature θ consists of a real number dependent on $\mathbf{X}$ and the output of the decision function δ is an estimate $\hat{\theta}$ of θ, a popular choice for the loss function is represented by the *quadratic* or *squared loss function*

$$L_2(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2, \tag{2.3.3}$$

and the risk for choosing $\hat{\theta}$ is the *mean squared error*

$$MSE = E_{\theta|\mathbf{X}}([\theta - \hat{\theta}]^2). \tag{2.3.4}$$

The mean squared error is minimised by the conditional expectation $E(\theta|\mathbf{X})$ [23].

A generalisation of the squared-error loss which is of interest in estimation problems is the *weighted quadratic* or *squared loss function*

$$L_2^w(\theta, \hat{\theta}) = w(\theta)(\theta - \hat{\theta})^2, \tag{2.3.5}$$

This loss has the attractive feature of allowing the squared error, $(\theta - \hat{\theta})^2$, to be weighted by a positive-valued function of θ. This reflects the fact that a given error in an estimation problem often varies in severity according to what the real value of θ happens to be.

Another loss function which can adopted to solve estimation problems is represented by the L_1 *loss function* or *absolute error*, which is defined as

$$L_1(\theta, \hat{\theta}) = |\theta - \hat{\theta}|. \tag{2.3.6}$$

In this case, the risk of $\hat{\theta}$ is the *mean absolute error*

$$MAE = E_{\theta|\mathbf{X}}(|\theta - \hat{\theta}|), \tag{2.3.7}$$

which is minimised by any conditional median $m \in Med_{\theta|\mathbf{X}}$ [23].

Finally, another loss function mainly used when θ is discrete is represented by the *zero-one loss function*:

$$L_{0-1}(\theta, \hat{\theta}) = I(\theta, \hat{\theta}), \tag{2.3.8}$$

where $I(x, y)$ is the indicator function. This loss function assigns a loss of 0 for correct estimates and a loss of 1 to any kind of estimation error regardless of the value predicted, for all errors are deemed equally costly.

Notation (Loss Matrix)**.** If the state of nature takes k distinct values θ_i, the loss function used can be most conveniently represented as a *loss matrix*

$$\Lambda = \|\lambda_{i,j}\|, \tag{2.3.9}$$

where $\lambda_{i,j} = L(\theta_i, \theta_j)$, and the loss for choosing θ_j for an observation $\mathbf{x}$ is

$$R^j(\mathbf{x}) = \sum_{i=1}^{k} \lambda_{i,j} P_i(\mathbf{x}). \tag{2.3.10}$$

For instance, using the notation above, the loss matrix for the zero-one loss is

$$\lambda_{i,j} = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{otherwise.} \end{cases} \quad i, j = 1, \ldots, k, \tag{2.3.11}$$

and, applying (2.3.11) in (2.3.10), the risk associated with choosing θ_j for $\mathbf{x}$ is

$$R^j_{0-1}(\mathbf{x}) = \sum_{i \neq j} P_i(\mathbf{x}) = 1 - P_j(\mathbf{x}). \tag{2.3.12}$$

R_{0-1} is minimised by choosing for $\mathbf{x}$ the value θ_j for which $1 - P_j(\mathbf{x})$ is minimal, namely the value for which $P_j(\mathbf{x})$ is maximal. In other words, the optimal choice is represented by the conditional mode $Mod_{\Theta|\mathbf{X}}$.

A way to choose a decision rule which does not depend on the decision made for a given observation $\mathbf{x}$ is represented by considering the expected loss with respect to the prior distribution on $\mathbf{X}$.

Definition 20. The *Bayes risk* of a decision rule δ is defined as

$$R^\delta = E_{\mathbf{X}}(R^\delta(\mathbf{x})). \tag{2.3.13}$$

In case of a loss function expressed as a *loss matrix* $\Lambda = \|\lambda_{i,j}\|$, the Bayes risk becomes

$$R^\Lambda = E_{\mathbf{X}}(R^j(\mathbf{x})) = \int_{\mathcal{X}} \sum_{i=1}^{k} \lambda_{i,j} P_i(\mathbf{x}) P_{\mathbf{X}}(\mathbf{x}) \mathbf{dx}. \tag{2.3.14}$$

Since the Bayes risk is a number, one can simply seek a decision rule which minimises it.

Definition 21 (The Minimal Bayes Risk Principle)**.** Given two decision rules δ_1 and δ_2, δ_1 is preferred to δ_2 if

$$R^{\delta_1} < R^{\delta_2}.$$

A decision rule δ^* is called a *minimal risk Bayes decision rule* if

$$\delta^* = \arg\min_{\delta} R^{\delta}.$$

The quantity R^* is called the *Bayes risk*.

In case of a loss function expressed as a *loss matrix* $\Lambda = \|\lambda_{i,j}\|$, the minimal risk Bayes allocation rule δ^* corresponds to choosing for an object $\mathbf{x}$ the state of nature θ_i for which the risk $R^c(\mathbf{x}) = \sum_{i=1}^{k} \lambda_{i,j} P_i(\mathbf{x})$ is minimal.

2.4 Pattern Recognition

In machine learning, *pattern recognition* is the problem of assigning a label to an input value. In the following we introduce the generic type of pattern recognition problem addressed by the algorithms described in this thesis.

2.4.1 Supervised Pattern Recognition

A *supervised pattern recognition* algorithm assumes the availability of a set of labelled data (the *training data set* S), which in this thesis shall consist of a number of labelled examples $(\mathbf{x}, y)$, where

- $\mathbf{x}$ is a vector of *descriptive features*, *independent attributes*, *explanatory variables* or *predictors* —quantifiable properties which together constitute all known characteristics describing an instance— taking values in a p-dimensional *feature* or *input space* $\mathcal{X} = \prod_{j=1}^{p} \mathcal{X}_j$.
- y is the *label* of $\mathbf{x}$, also called the *target*, *response*, or *dependent attribute*, taking values in the *target* or *output space* $\mathcal{Y}$.

Assuming the existence of a functional relationship between the feature space and the target space, a supervised pattern recognition algorithm observes the training data set (the *training phase*) in order to learn from the examples it comprises a model which approximates the unknown functional relation also for individuals not included in the training data set (the training data set usually does not cover the whole population). The learning process can be seen as one of choosing a function (a *hypothesis*)

$$h : \mathcal{X} \to \mathcal{Y} \tag{2.4.1}$$

from a fixed class of functions $\mathcal{H}$ (the *hypothesis space*) which best fits the training data set according to a given criterion.

The training data set that the algorithms described in this thesis learn from is assumed to be a finite multiset

$$S = (S_D, m_S).$$

comprising N training examples, whose underlying set of elements is the set of *distinct* training examples observed in S

$$S_D = \{(d_i)\}_{i=1}^{m} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m},$$

and

$$m_S : S_D \to \mathbb{N}^*$$

is the multiplicity function of the elements of S.

We have chosen this rather general representation in order to deal with the situation where there are training examples consisting of identical feature vectors but having different class labels. This phenomenon occurs frequently in real-world applications and for different reasons (e.g. clerical errors, disagreement among experts on how to label a given individual, the existence of a latent variable), and the algorithms presented in this thesis are able to deal with it.

Notation. We shall sometimes denote a training data set as

$$S = (S_D, G_S),$$

where G_S represents the graph of its multiplicity function, that is

$$\{(a, m_S(a)) : a \in S_D\}.$$

Notation. We shall denote the set of distinct feature vectors $\mathbf{x}$ occurring in a training data set S as

$$S_{\mathcal{X}} = \{\mathbf{x}_j\}_{j=1}^{n}.$$

Moreover, for each $\mathbf{x} \in S_{\mathcal{X}}$ and $y \in \mathcal{Y}$, we shall denote the number of

occurrences of $\mathbf{x}$ in S as

$$n(\mathbf{x}),$$

and the number of occurrences of $\mathbf{x}$ in S with label equal to y as

$$n(\mathbf{x}, y).$$

It be should noted that, since the pattern recognition algorithms described in this thesis operate in a supervised setting, the adjective supervised shall sometimes be omitted in the following.

2.4.2 Empirical Risk Minimization

Because of the uncertainty due for example to the presence of noise in the data, the relation between a feature vector $\mathbf{x}$ and its label y is not modelled as a deterministic function. Instead, the training data set is assumed to be a sample drawn from the random vector $(\mathbf{X}, Y)$ having joint probability distribution $P_{\mathbf{X},Y}$, and y is the value of a random variable Y on $\mathcal{Y}$ with posterior distribution $P_{Y|\mathbf{X}}$.

Once a loss function L measuring how different the prediction $\hat{y} = h(x)$ returned by a hypothesis h is from the true label y has been chosen, a supervised pattern recognition algorithm chooses the hypothesis $h^* \in \mathcal{H}$ for which the Bayesian risk (2.3.13) R^h is minimal, that is:

$$h^* = \arg\min_{h \in \mathcal{H}} R^h.$$

In general, R^h cannot be computed because the distribution $P_{\mathbf{X},Y}$ is unknown to the learning algorithm. However, an approximation called the *empirical risk* can be computed by averaging the loss function on the training data set:

$$R^h_{emp} = \frac{1}{N} \sum_{i=1}^{N} L(h(x_i), y_i \qquad (\mathbf{x}_i, y_i) \in S. \tag{2.4.2}$$

The choice of a loss function depends on many factors, including the type of label being predicted. In the case of the squared loss function, we

have the *empirical squared risk*

$$\sum_{i=1}^{N}(y_i - h(\mathbf{x}_i))^2 \qquad (\mathbf{x}_i, y_i) \in S, \tag{2.4.3}$$

and in the case of the L_1 loss function we have the *empirical absolute risk*

$$\sum_{i=1}^{N}|y_i - h(\mathbf{x}_i)| \qquad (\mathbf{x}_i, y_i) \in S. \tag{2.4.4}$$

Definition 22 (Empirical Risk Minimisation (ERM) Principle)**.** Given a hypothesis space $\mathcal{H}$ and a training sample S, choose a hypothesis $\hat{h}$ which minimises the empirical risk:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} R^h_{emp}. \tag{2.4.5}$$

Therefore, the learning strategy defined by the ERM principle consists of solving the optimization problem (2.4.5).

2.4.3 Monotonic Pattern Recognition

The pattern recognition algorithms presented in this thesis assume the existence of a partial order $\leq_{\mathcal{X}}$ on the feature space $\mathcal{X}$ and of a total order $\leq_{\mathcal{Y}}$ on the output space $\mathcal{Y}$. Moreover, they assume that the unknown functional relationship between $\mathcal{X}$ and $\mathcal{Y}$ is monotonic, so their goal is to learn a *monotonic* or *monotonicity-preserving hypothesis* $h : \mathcal{X} \to \mathcal{Y}$, namely such that

$$\mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \Rightarrow h(\mathbf{x}) \leq_{\mathcal{Y}} h(\mathbf{x}') \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \tag{2.4.6}$$

Informally speaking, the learned hypothesis returns predictions in which a lower ordered input is not allowed to have a higher label.

In many applications, the order on $\mathcal{X}$ is derived from knowledge of the signs of the influences of the features X_i on Y as follows. An attribute X_i has a *positive influence* on the class label Y if observing a higher value for X_i makes higher values for Y more likely [107]. For example, we might expect a greater risk of diabetes in persons with a higher body mass index. A *negative influence* is defined analogously: the larger the value of X_i, the

smaller the value of Y is likely to be.

Without loss of generality, we shall assume in this thesis that all influences are positive, for a negative influence from X_i on Y can be turned into a positive one simply by reversing the order on X_i.

As an example, we list in table 2.1 the signs of the influences of the descriptive features on the target attribute *Price* for the Windsor Housing data set, which is one the data sets used to test the accuracy of our algorithms (see appendix A for more information on this and on the other real-world data sets used in our experiments). The attributes of the data set are defined as follows [3]:

- Price of the house in Canadian dollars;
- DRV = 1 if the house has a driveway;
- REC = 1 if the house has a recreational room;
- FFIN = 1 if the house has a full and finished basement;
- GHW = 1 if the house uses gas for hot water heating;
- CA = 1 if there is a central air conditioning;
- GAR shows the number of garage places;
- REG = 1 if the house in located in a preferred neighbourhood of the city, that is Riverside or South Windsor;
- LOT is a continuous variable showing the lot size of the property in square feet;
- BDMS is the number of bedrooms;
- FB is the number of full bathrooms (i.e. including, at least, a toilet, sink, and bathtub);
- STY represents the number of storeys, excluding the basement.

The above considerations on the influences of each feature on the target variable typically translate into the *product order* on $\mathcal{X}$ induced by the total order on each $\mathcal{X}_i$, that is

$$\mathbf{x} \leq \mathbf{x}' \Leftrightarrow x_i \leq x'_i \qquad \forall i = 1, \ldots, p. \tag{2.4.7}$$

Anyway, it should be pointed out that the algorithms illustrated in this thesis are not at all limited in their application to this type of order; instead, they can be applied in the presence of any arbitrary partial order on $\mathcal{X}$.

Attribute	Type	Sign
Price	Real	Target
DRV	Binary	+
REC	Binary	+
FFIN	Binary	+
GHW	Binary	+
CA	Binary	+
GAR	Integer	+
REG	Binary	+
LOT	Numeric	+
BDMS	Integer	+
FB	Integer	+
STY	Integer	+

Table 2.1: Signs of the influences of the descriptive features on the target attribute *Price* for the monotonic Windsor Housing data set.

Finally, it should be noted that in the presence of a monotonic functional relationship between feature vectors and labels, it makes sense to choose a loss function which incurs a higher cost for those misclassifications that are "far" from the true label than for those that are "close". Examples of well-known loss function which are suitable candidates are the L_1 loss function and the squared loss function; on the other hand, the widely used 0/1 loss function, is not a suitable choice in this case because it does not satisfy this property.

2.4.4 Monotonicity Violations

Oftentimes, in spite of the assumption that the functional relationship between $\mathcal{X}$ and $\mathcal{Y}$ is monotonic, the training data set is not monotonic. This situation can be formalised as follows.

Definition 23 (Monotonicity Violation, Inconsistent Data Set)**.** If the functional relationship between $\mathcal{X}$ and $\mathcal{Y}$ is isotonic, given a training data set S, $(\mathbf{x}, y), (\mathbf{x}', y') \in S_D$ are a *non-monotonic pair* or a *monotonicity violation* if

$$\mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \wedge y >_{\mathcal{Y}} y' \text{ or } \mathbf{x}' \leq_{\mathcal{X}} \mathbf{x} \wedge y' >_{\mathcal{Y}} y. \tag{2.4.8}$$

S is called an *inconsistent* (*labelled*) *data set*.

As an example, let us consider a training data set consisting of five distinct observations, each described by a two-dimensional real vector $\mathbf{x}_i$ and having an integer-valued label y_i ranging from 1 to 3. The feature vectors are ordered according to the product order induced by the standard ordering of real numbers, and the labels are totally ordered according to the standard ordering of integer numbers. The order graph for the attribute vectors comprising the data set is depicted in figure 2.2, with the observed class labels given inside the nodes; it is easy to see that only five of the ordered pairs of feature vectors which can be formed are comparable.

Because the feature space is ordered according to the product order, the vectors which are comparable with a given vector are the vectors corresponding to points situated in the first and the third quadrants of a Cartesian plane centred in that point; the incomparable feature vectors, on the other hand, are those situated in the second and in the fourth quadrants. For instance, $\mathbf{x}_0$ and $\mathbf{x}_1$ are comparable because $\mathbf{x}_0^1 \leq \mathbf{x}_1^1$ and $\mathbf{x}_0^2 \leq \mathbf{x}_1^2$, while $\mathbf{x}_0$ and $\mathbf{x}_3$ are incomparable because $\mathbf{x}_0^1 \geq \mathbf{x}_1^1$ but $\mathbf{x}_0^2 \leq \mathbf{x}_1^2$.

The same visual clue can be used to determine the upset and downset of a feature vector as they correspond to the points in the first and third quadrant respectively. In our example, the upset of $\mathbf{x}_0$ is $\uparrow \mathbf{x}_0 = \{\mathbf{x}_1\}$ because the value of each coordinate of $\mathbf{x}_1$ is greater than or equal to the corresponding value for $\mathbf{x}_0$; similarly, the downset of $\mathbf{x}_0$ is $\downarrow \mathbf{x}_0 = \{\mathbf{x}_2\}$.

Finally, if the assumption is made that there exists an isotonic functional relationship between attribute vectors and labels, then this data set is inconsistent. In fact, $(\mathbf{x}_0, y_0)$, $(\mathbf{x}_1, y_1)$ constitutes a monotonicity violation because $\mathbf{x}_0 \leq \mathbf{x}_1$ but $y_0 = 2 < 1 = y_1$.

It should be noted that the algorithms presented in this thesis are capable of returning monotonicity-preserving predictions even when trained on an inconsistent data set without the need to remove beforehand the monotonicity violations it contains.

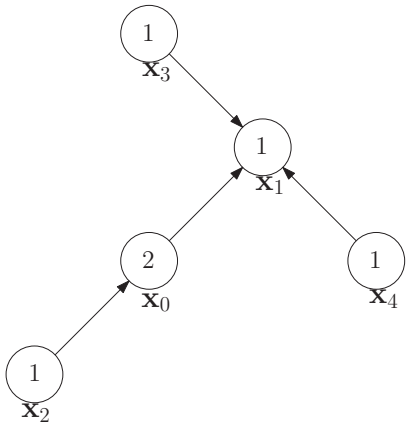

Figure 2.2: Order graph for a labelled data set where feature vectors are ordered according to the product order on the feature space. Each node represents a distinct observation; class label are given inside the nodes.

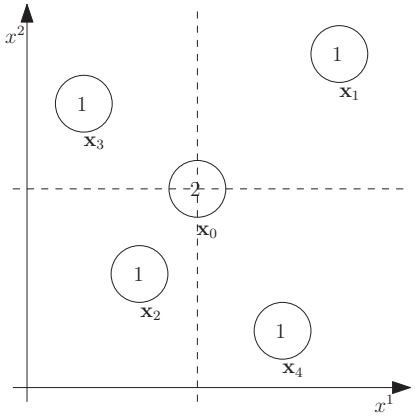

Figure 2.3: Visual clue to determine the feature vectors comparable and incomparable with and the upset and downset of a feature vector when the feature space ordered according to the product order.

2.5 The Isotonic Regression

In this section, we introduce the problem of the isotonic regression, which lies at the heart of the algorithms presented in this thesis. It should be noted that all the results and algorithms we introduce are already known and have been proved. Nonetheless, we discuss them in some detail because of their importance to our work.

Regression is an example of the more general problem of pattern recognition in which a real-valued output is assigned to a given input value. It is concerned with the fitting of curves or functions to a set of points $(\mathbf{x}, y)$ or, more generally, to the joint distribution of a random vector $(\mathbf{X}, Y)$ [94]. The function chosen to fit the points or the joint distribution is called a *regression function*; it can be used in many ways, such as for prediction, for studying the degree of association between the explanatory variable $\mathbf{X}$ and the dependent variable Y, or for providing new insights into the phenomenon which generated the data.

In many situations, one might strongly believe that the underlying regression function has a particular shape or form which can be characterised by certain order restrictions, and, consequently, it is natural to restrict the selection of the regression function to an appropriate class of functions. In the presence of a monotonic relationship between the descriptive variables and the dependent variable, the class of regression functions shall be that of isotonic functions.

The monotonic function which best fits the data can then be chosen by using some measure of quality of the fit such as least squares or least absolute deviations, which correspond to minimising the mean squared error and the mean absolute error respectively. As explained in section 2.4.3, the choice of the squared error or of the absolute error as the loss function to minimise is most appropriate in the presence of monotonicity constraints.

In the case of (restricted) least squares and of isotonic regression functions, we have the following optimisation problem.

Definition 24 (Isotonic Regression)**.** Let $Z = \{z_i\}_{i=1}^n$ be a non-empty, partially-ordered set, g be a function on Z, and $W = \{w_i\}_{1=1}^n$ a set of weights with w_i associated to z_i for all $i = 1, \ldots, n$. The *isotonic regression*

of g (*with respect to the partial order* $\leq_Z$ *and to the weights* W) is the real function g^* on Z which minimises the sum

$$\sum_{i=1}^{n} w_i \, [f(z_i) - g(z_i)]^2 \tag{2.5.1}$$

in the class of isotonic functions f on Z. The *antitonic regression* of g is defined as the function which minimises (2.5.1) within the class of antitonic functions.

Typically, for all $i = 1, \ldots, n$ the value $g(z_i)$ represents the estimate of the value of an unknown, real-valued isotonic function defined on Z and w_i a quantitative indication of the precision of this estimate.

Because an antitonic function can be easily turned into an isotonic function (e.g. by multiplying it by -1), we shall restrict our attention without loss of generality to the problem of computing the isotonic regression. Moreover, due to the following proposition, we shall speak in the following of *the* isotonic regression.

Proposition 5 (Spouge et al. [100])**.** *There always exists a unique isotonic regression* g^**.*

Computing the isotonic regression corresponds to solving the quadratic programming problem having objective function (2.5.1) and linear constraints corresponding to the comparable pairs in Z.

It should be noted that if g is already isotonic or if all of the elements of Z are incomparable, then the trivial solution

$$\forall x \in Z : g^*(x) = g(x)$$

is the unique solution to equation (2.5.1).

Various dedicated algorithms, often restricted to a particular type of order, have been proposed to solve this optimisation problem.

The *Pool Adjacent Violators* (*PAV*) [7] can be applied if Z is linearly ordered. This algorithm has a time complexity that is linear in the size of the set of constants. (See for instance [1].)

The minimum lower sets (MLS) algorithm [19] allows to compute the exact solution to the isotonic regression problem for an arbitrary partial

order. Although extremely inefficient, this algorithm is very simple and, therefore, suited to formulating small examples giving insights into the nature of the solution to the isotonic regression problem. This is the reason why we discuss this algorithm in section 2.5.1.

The algorithm which we have adopted in our research to compute the isotonic regression and which we call the *Divide-and-Conquer Algorithm* is described in section 2.5.2; the description provided is based on the work by Spouge et al. [100]. It represents a generalisation of the algorithm presented by Maxwell and Muckstadt in [81] solving a scheduling problem in economics. Spouge et al. have proved that Maxwell and Muckstadt's algorithm is incorrect but that, nonetheless, the divide-and-conquer strategy it is based on can be used to compute the isotonic regression by solving at most n maximal-flow problems to find minimum cuts instead of maximum cuts as originally suggested by Maxwell and Muckstadt. This technique was introduced by Picard in [89].

The divide-and-conquer algorithm has the best time complexity known for an exact solution to the isotonic regression problem for an arbitrary partial order, namely $O(n^4)$. Spouge et al. [100] have shown that if Z is a subset of the Euclidean plane with product ordering induced by the Euclidean distance on each dimension, then the isotonic regression problem can be solved in $O(n^3)$ time; moreover, they have proved that if the two dimensional set is restricted to a *grid* (that is, if all points have integer coordinates), the time complexity can be further reduced to $O(n^2)$.

Notation. Given a real-valued function f defined on the set Z and a real number c, we shall denote the set of points $x \in Z$ such that $f(x) = c$ as

$$[f = c].$$

Moreover, we shall denote the set of points $x \in Z$ such that $f(x) \leq c$ as

$$[f \leq c],$$

and we shall use a similar notation for the other possible order relations.

On the other hand, the notation

$$f \equiv c$$

shall denote the fact that f is a constant function with constant value c. Finally, the notation

$$f \leq c$$

shall denote the fact that for all points $x \in Z$, $f(x) \leq c$, and likewise for the other possible order relations.

Definition 25 (Level Set)**.** A subset B of Z is a *level set* if and only if there exist a lower set L and an upper set U such that $B = L \cap U$.

Proposition 6 (Robertson [94])**.** *A subset B of Z is a* level set *if and only if there exist an isotonic function f on Z and a real number c such that $B = [f = c]$.*

Definition 26 (Weighted Average)**.** Given a real-valued function f, a positive weight function w, and a subset S of the domain of f, we define the *weighted average* of S as

$$Av_f(S) = \frac{\sum_{z \in S} w(z) f(z)}{\sum_{z \in S} w(z)}. \tag{2.5.2}$$

Proposition 7 (Robertson [94])**.** *For any real number c such that the set $[g^* = c]$ is non-empty*

$$Av_g([g^* = c]) = c \tag{2.5.3}$$

Proposition 7 reduces the problem of computing g^* to finding sets on which g^* is constant, i.e its level sets.

Proposition 8 (Robertson [94])**.** *For every $z \in Z$, the isotonic regression of g is given by*

$$g^*(z) = \min_{L \in \mathcal{L}_Z : z \in L} \max_{U \in \mathcal{U}_Z : z \in U} Av_g(L \cap U). \tag{2.5.4}$$

The isotonic regression with respect to a partial order is equivalent to the *antitonic* regression with respect to the inverse order. By considering

the inverse order, lower sets and upper sets are interchanged; therefore, the antitonic regression can be characterized as follows.

Proposition 9 (Robertson [94])**.** *For every $z \in Z$, the antitonic regression of g is given by*

$$g^*(z) = \max_{L \in \mathcal{L}_Z : z \in L} \min_{U \in \mathcal{U}_Z : z \in U} Av_g(L \cap U). \tag{2.5.5}$$

2.5.1 The Minimum Lower Set Algorithm

Notation. For all non-empty, disjoint subsets A and B of Z, the notation

$$A + B$$

shall be used to denote $A \cup B$.

Definition 27. A real function M defined on the non-empty subsets of Z is a *Cauchy mean value function* if, for all non-empty subsets A, B of Z, $M(A + B)$ is between $M(A)$ and $M(B)$.

Definition 28. M is a *strict Cauchy mean value function* if it is a Cauchy mean value function and has the additional property for all non-empty subsets A, B of Z

$$M(A) < M(B) \Rightarrow M(A) < M(A + B) < M(B). \tag{2.5.6}$$

The weighted average (2.5.2) is an example of a strict Cauchy mean value function.

Proposition 10. *If M is a strict Cauchy mean value function, then for all non-empty subsets A, B of Z*

1. $M(A + B) \leq M(A) \Rightarrow M(B) \leq M(A + B)$
2. $M(A + B) \geq M(A) \Rightarrow M(B) \geq M(A + B)$

Proposition 11 (Robertson [94])**.** *The union B of all lower sets of Z of minimum average*

$$B = \bigcup \{A \in \mathcal{L}_Z \mid A = argmin_{L \in \mathcal{L}_Z} Av_g(L)\} \tag{2.5.7}$$

is the largest lower set of Z of minimum average.

Let us denote as B_1 the set defined in equation (2.5.7). This set is the level set on which g^* assumes its smallest value, and, because of proposition 7, it follows that

$$\forall x \in B_1 : g^*(x) = Av_g(B_1) = min\{Av_g(L) \mid L \in \mathcal{L}_Z\}.$$

Now let us consider the averages of level sets of the form $L \cap B_1^c$ for all $L \in \mathcal{L}$, namely the level sets consisting of lower sets of Z with B_1 subtracted. Let us select again the largest of these level sets of minimum average $B_2 = L_2 \cap B_1^c$. B_2 is the level set on which g^* assumes its smallest value:

$$\forall x \in B_2 : g^*(x) = Av_g(B_2).$$

This process is continued until Z is exhausted.

This is the idea on which the *Minimum Lower Set* (MLS) algorithm, which is illustrated in Algorithm 1, is based. The algorithm removes monotonicity violations by averaging over suitably-chosen subsets $B \subset Z$; as a consequence, it partitions the set Z into a number of blocks on which the isotonic regression is constant. At each iteration, a lower set with minimum weighted average is selected; if multiple lower sets attain the same minimum, then any of them can be considered. In Algorithm 1 the largest one is taken, which is obtained by taking the union of all minimum lower sets.

If we consider the blocks in the order in which they are selected by the MLS algorithm, then it is clear that

$$Av_g(B_i) \leq Av_g(B_{i+1}),$$

since otherwise

$$Av_g(B_i \cup B_{i+1}) < Av_g(B_i),$$

which would contradict the assumption made that B_i is a minimum lower set at step i.

The MLS algorithm can be adjusted to compute the antitonic regression by considering the collection $\mathcal{U}_Z$ of all upper sets of Z with respect to $\leq$ rather than the collection $\mathcal{L}_Z$ of all lower sets with respect to $\leq$.

What limits the applicability of this algorithm is the high computational cost of the generation of lower sets, which is exponential in the size of Z [43].

Algorithm 1 MinimumLowerSets(Z, $\leq$, $g(z)$, $w(z)$)

```
1: L ← Collection of all lower sets of (Z,≤)
2: repeat
3:    B ← ⋃{A ∈ L | Av_g(A) = min_{L∈L} Av_g(L)}
4:    for all z ∈ B do
5:       g*(z) ← Av_g(B)
6:    end for
7:    for all L ∈ L do
8:       L ← L \ B
9:    end for
10:   Z ← Z \ B
11: until Z = ∅
12: return g*
```

2.5.2 The Divide-and-Conquer Algorithm

As we mentioned at the end of the previous section, the complexity of determining the set $\mathcal{L}_Z$ of all lower sets of Z makes the adoption of the MLS algorithm impracticable in real-world situations. A more viable alternative is represented by the *divide-and-conquer algorithm*, which we introduce in this section. Our discussion in based on the description provided by Spouge et al. in [100].

Proposition 12. *If f is any isotonic function on Z, then for any real number a, $[f < a]$ is a lower set of Z, and $[f > a]$ is an upper set of Z.*

Notation. Given a real-valued function f defined on the set Z, and a subset $S \subseteq Z$, the notation

$$(f|_S)^*$$

shall denote the isotonic regression of f restricted to S, which is distinct from the restriction of the isotonic regression of f to S

$$f^*|_S.$$

Definition 29 (Complementary Pair)**.** A *complementary pair* (L, U) of Z is a partition of Z such that U is an upper set and, consequently, L is a lower set. (L, U) is *trivial* if either L or U is empty.

Definition 30 (Projection Pair)**.** A *projection pair* of Z for the function g is a non-trivial complementary pair (L, U) such that

$$\exists\, c_{LU} \in \mathbb{R} : (g|_L)^* \leq c_{LU} \leq (g|_U)^*.$$

Theorem 1. *If (L, U) is a projection pair for g, then*

$$g^*|_L = (g|_L)^*,\ g^*|_U = (g|_U)^*. \tag{2.5.8}$$

Proof. If we define

$$g^*_{LU} = \begin{cases} (g|_L)^*, & \text{if } x \in L \\ (g|_U)^*, & \text{if } x \in U \end{cases}$$

then g^*_{LU} is isotonic. In fact, if $x, x' \in L$, then

$$x \leq x' \Longrightarrow g^*_{LU}(x) = (g|_L)^*(x) \leq (g|_L)^*(x') = g^*_{LU}(x').$$

The same is true if $x, x' \in U$. On the other hand, if $x \in L$ and $x' \in U$ and, therefore, $x \leq x'$, then

$$g^*_{LU}(x) = (g|_L)^*(x) \leq c_{LU} \leq (g|_U)^*(x') = g^*_{LU}(x').$$

As a consequence,

$$\|g-g^*\|^2_Z = \|g-g^*\|^2_L + \|g-g^*\|^2_U \geq \|g-g^*_{LU}\|^2_L + \|g-g^*_{LU}\|^2_U = \|g-g^*_{LU}\|^2_Z,$$

where the inequalities are a consequence of the fact that g^*_{LU} corresponds to the isotonic regression of g restricted to L and U respectively. From the uniqueness of g^*, it then follows that $g^* = g^*_{LU}$. □

Theorem 1 shows that the problem of computing the isotonic regression g^* of a given function g on Z can be reduced to that of finding a projection pair (L, U) for g as the restriction of g^* to L and U is equal to the isotonic regression of g restricted to L and U respectively. We would like to be

able to repeat this decomposition in a recursive manner with a reduction in size and complexity at each division until the problem of computing the regression becomes trivial on the pieces. In the following we show how this is possible.

Definition 31 (Maximal Upper Set)**.** $\hat{U} \in \mathcal{U}_Z$ is a *maximal upper set* for $b \in \mathbb{R}$ if

$$\hat{U} = \arg\max_{U \in \mathcal{U}_Z} s_b(U), \tag{2.5.9}$$

where

$$s_b(A) = \sum_{x \in A} s_b(x), \quad A \subseteq Z, \tag{2.5.10}$$

and

$$s_b(x) = w(x)[g(x) - b], \quad x \in Z. \tag{2.5.11}$$

The definition of a *minimal lower set* follows dually.

Lemma 1. *For any real number c, the following implications are true:*

$$[g^* \leq c] \neq \emptyset \implies Av_g([g^* \leq c]) \leq c \tag{2.5.12}$$

$$[g^* \geq c] \neq \emptyset \implies Av_g([g^* \geq c]) \geq c \tag{2.5.13}$$

Proof. The set $[g^* \leq c]$ can be decomposed as

$$[g^* \leq c] = \cup_{y \in Y_c} [g^* = y], \tag{2.5.14}$$

with

$$Y_c = \{y \leq c \mid [g^* = y] \neq \emptyset\} = g^*(Z) \cap [-\inf, c].$$

Being a subset of the real line, the elements of Y_c can be sorted as

$$y_1 \leq y_2 \leq \ldots y_m = c.$$

Therefore, from proposition 7 and from the fact that Av_g is a strict Cauchy

mean value function, it follows that

$$\begin{aligned} Av_g([g^* \leq c]) &= Av_g(\cup_{y \in Y_{c'}}[g^* = y]) \\ &= Av_g\left([g^* = y_1] + [g^* = y_2] + \cdots + [g^* = y_{m-1}] + [g^* = y_m]\right) \\ &= Av_g\left(([g^* = y_1] + [g^* = y_2] + \cdots + [g^* = y_{m-1}]) + [g^* = y_m]\right) \\ &\leq Av_g([g^* = y_m]) = Av_g([g^* = c]) = c. \end{aligned}$$

The second implication follows by duality. □

Definition 32 (Maximal Complementary Pair)**.** A complementary pair (L, U) of Z is called *maximal* for $b \in \mathbb{R}$ if U is a maximal upper set (or, equivalently, if L is a minimal lower set) for b.

Theorem 2. *For a non-empty subset X of Z and for $b \in \mathbb{R}$, the following implications are true:*

$$X \text{ is a maximal upper set for } b \implies \forall x \in X : b \leq (g|_X)^*(x) \tag{2.5.15}$$

$$X \text{ is a minimal lower set for } b \implies \forall x \in X : (g|_X)^*(x) \leq b \tag{2.5.16}$$

Proof. Let us suppose that X is a maximal upper set

$$u = min(g\,|_X)^*.$$

Because X can be partitioned as

$$X = [(g\,|_X)^* \leq u] \cup [(g\,|_X)^* > u],$$

and because X is a maximal upper set, it follows that

$$s_b([(g\,|_X)^* > u]) \leq s_b(X) = s_b([(g\,|_X)^* \leq u]) + s_b([(g\,|_X)^* > u]),$$

and, as a consequence,

$$0 \leq s_b([(g\,|_X)^* \leq u]).$$

Therefore, from the definition of s_b, it follows that

$$b \leq Av_g([(g\,|_X)^* \leq u]).$$

From lemma 1, we finally obtain that

$$b \leq Av_g([(g\mid_X)^* \leq u]) \leq u \leq (g\mid_X)^*.$$

The second implication follows by duality. □

Corollary 1. *Every non-trivial maximal complementary pair (L, U) for $b \in \mathbb{R}$ is a projection pair for g, with $c_{LU} = b$.*

Proof. Because L and U are respectively a minimal lower set and a maximal upper set for b and are both non-empty, from theorem 2 it follows that

$$(g\mid_L)^* \leq b \leq (g\mid_U)^*.$$

Therefore, (L, U) is a projection pair for g. □

Theorem 3. *For $b = Av_g(Z)$, the following implications are true:*

$$\emptyset \text{ is a maximal upper set for } b \implies g^* \equiv b \tag{2.5.17}$$

$$\emptyset \text{ is a minimal lower set for } b \implies g^* \equiv b \tag{2.5.18}$$

Proof. First of all, it should be noted that, since $b = Av_g(Z)$, we have $s_b(Z) = 0$, as

$$s_b(Z) = \sum_{x \in Z} w(x)[g(x) - Av_g(Z)] = \tag{2.5.19}$$

$$= \sum_{x \in Z} w(x)g(x) - \frac{\sum_{x \in Z} w(x)g(x)}{\sum_{x \in Z} w(x)} \sum_{x \in Z} w(x) = 0. \tag{2.5.20}$$

Moreover,

$$s_b(\emptyset) = 0.$$

If $\emptyset$ is a maximal upper set for $Av_g(Z)$, then Z is a maximal upper set for $Av_g(Z)$ too, and, because of inequality (2.5.15),

$$b \leq g^*.$$

On the other hand, the fact that $\emptyset$ is a maximal upper set for $Av_g(Z)$ implies that Z is a minimal lower set for $Av_g(Z)$; as a consequence, because

of inequality (2.5.16),

$$g^* \leq b.$$

Therefore,

$$g^* \equiv b.$$

Implication (2.5.18) follows by duality. □

The above properties lead to the recursive procedure IR to compute the isotonic regression illustrated in Algorithm 2.

Algorithm 2 IR(Z, $\leq$, $g(\cdot)$, $w(\cdot)$)

1: $(L, U) \leftarrow$ FindMaximalComplementaryPair(Z, $\leq$, $g(\cdot)$, $w(\cdot)$)
2: **if** $L = \varnothing$ or $U = \varnothing$ **then**
3: **return** $Av_g(Z)$
4: **else**
5: **return** (IR(L, $\leq$, $g|_L(\cdot)$, $w|_L(\cdot)$) , IR(U, $\leq$, $g|_U(\cdot)$, $w|_U(\cdot)$)
6: **end if**

At each step, a maximal projection pair for $Av_g(Z)$ is computed by procedure $FindMaximalComplementaryPair$, which is illustrated in Algorithm 3.

Algorithm 3 FindMaximalComplementaryPair(Z, $\leq$, $g(\cdot)$, $w(\cdot)$)

1: Construct the following network $N = (V, A)$:
Vertices: $\forall x \in Z$ add a corresponding vertex x to V
Source and Sink: add a source vertex s and a sink vertex t to V
Edges: $\forall x, y \in Z$, if $x \leq y$ then add to A a directed edge $x \to y$ with capacity $c(x, y) = \infty$
Capacity: compute $Av_g(Z)$; $\forall x$ compute $a(x) = s_{Av_g(Z)}(x)$ (see equation 2.5.11)
Max Flow In: $\forall x \in Z : a(x) > 0$ add to A an edge $s \to x$ with capacity $c(s, x) = a(x)$
Max Flow Out: $\forall x \in Z : a(x) < 0$ add to A an edge $x \to t$ with capacity $c(x, t) = -a(x)$
2: Solve the maximum-flow problem on the network N to find the minimum $s - t$ cut $C = (S, T)$
3: **return** $(T - t, S - s)$

Algorithm 3 consists of Picard's maximal closure algorithm [89]. Picard showed that a maximal upper set can be obtained from a minimum $s - t$

cut in a network $N = (V, A)$ constructed as illustrated. The minimum $s-t$ cut required at step 2 can be computed by using a standard maximum flow algorithms such as the Ford-Fulkerson algorithm [28], which is the algorithm we adopted in the implementation we have used in our research.

Given the $O(n^3)$ time complexity of minimum $s-t$ cut algorithms, the overall complexity of the algorithm is $O(n^4)$. This upper bound on the time complexity is the best known for an exact solution to the isotonic regression problem for an arbitrary partial order. However, there are a few special cases worth mentioning:

- If Z is already isotonic, then the algorithm's time complexity is $O(n^2)$. Each time Algorithm 3 is executed, it is not possible to find a path from the source s to the sink t; both concluding there is no path and then enumerating which elements belong to L or to U can be done in $O(n)$ time.
- If each split between L and U is evenly balanced, then the algorithm's time complexity is $O(n^3 \log(n))$. As the size of the graphs constructed halves during each iteration, the sum over all constructed graphs viewed in time is $1 \cdot n^3 + 2 \cdot \left(\frac{1}{2}n\right)^3 + 4 \cdot (\frac{1}{4}n)^3 + \cdots + n \cdot (\frac{1}{n}n)^3 = \log(n)n^3$, resulting in a total running time of $O(n^3 \log(n))$.
- If Z is antitonic, then the time complexity is $O(n^3)$. It is still necessary to calculate the minimum $s-t$ cut, but there is no recursion.

All of the aforementioned cases are likely to occur during some of the algorithm's recursions in (parts of) real-world data sets; therefore, the algorithm's actual running time can be expected to be better than the $O(n^4)$ upper bound.

As an illustration of Algorithm 2, in the following section we shall apply it to computing the isotonic regression of the real-valued function g. First, though, let us introduce the following definition and notation.

Definition 33 (Path in a Directed Graph)**.** Given a directed graph $G = (V, E)$, $v, v' \in V$, a *path* of length k from vertex v to vertex v' is a sequence of vertices

$$p = \langle v_0, v_1, \ldots, v_k \rangle$$

such that $(v_{i-1}, v_i) \in E$ for $i = 1 \ldots n$.

Notation. Given a directed graph $G = (V, E)$, we shall denote the set of all paths between two vertices $v, v' \in V$ as

$$v \underset{G}{\rightsquigarrow} v'.$$

2.5.3 Example

Let us apply Algorithm 2 to computing the isotonic regression of a real-valued function g on the poset

$$Z = \{x_1, x_2, x_3, x_4, x_5\}.$$

The order graph for Z is given in Figure 2.4; in it, each node is labelled with the value of g at the point that the node represents.

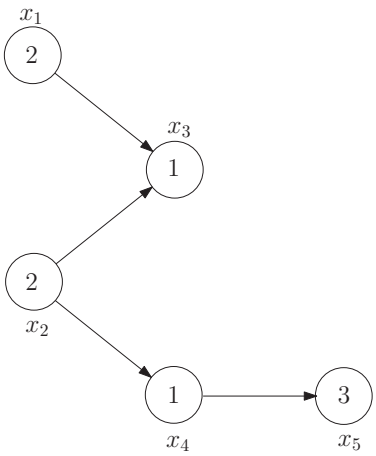

Figure 2.4: Divide-and-conquer algorithm example. Order graph for the domain $Z = \{x_1, x_2, x_3, x_4, x_5\}$ of a real-valued function g labelled with the values of the function. The monotonicity violations in the graph of g shall be resolved by the algorithm.

It should be noted that the graph of g is inconsistent because of the following three monotonicity violations (see definition 23):

1. $(x_1, 2), (x_3, 1)$
2. $(x_2, 2), (x_3, 1)$

3. $(x_2, 2), (x_4, 1)$

These violations shall be resolved by the algorithm.

We begin by determining the first maximal complementary pair. In order to do so, we compute the values required to build the flow network of step 1 of algorithm 3:

$$Av_g(Z) = \frac{2+1+2+1+3}{5} = 1.8$$

$$\begin{aligned}
a(x_1) &= w(x_1)[g(x_1) - Av_g(Z)] = 1 \cdot [2 - 1.8] = 0.2 \\
a(x_2) &= w(x_2)[g(x_2) - Av_g(Z)] = 1 \cdot [2 - 1.8] = 0.2 \\
a(x_3) &= w(x_3)[g(x_3) - Av_g(Z)] = 1 \cdot [1 - 1.8] = -0.8 \\
a(x_4) &= w(x_4)[g(x_4) - Av_g(Z)] = 1 \cdot [1 - 1.8] = -0.8 \\
a(x_5) &= w(x_5)[g(x_5) - Av_g(Z)] = 1 \cdot [3 - 1.8] = 1.2
\end{aligned}$$

Figure 2.5 shows the first iteration of the Ford-Fulkerson algorithm applied to the whole of Z. The top side represents the initial residual network R_f, with the augmenting path p_1 found in it shaded. It coincides with the input network N and all edges are labelled with their capacity because the initial flow f is equal to 0 on all of them; edges with residual capacity equal to 0 are not shown, which is a convention we shall follow in the remainder of this example. The bottom side shows the new flow $f_1 = f \uparrow f_{p_1}$ resulting from augmenting f by f_{p_1}.

Figure 2.6 shows the second and final iteration of the Ford-Fulkerson algorithm applied to Z. The top side represents the new residual network R_{f_1} with the augmenting path p_2 found in it shaded. The bottom side shows the new flow $f_2 = f_1 \uparrow f_{p_2}$. Because there are no other augmenting paths, f_2 is a maximum-flow for the network. According to the Ford-Fulkerson theorem [28], the associated $s - t$ cut (S, T) is minimum and is given by

$$\begin{aligned}
S &= \{v \in V \mid \exists p \in s \underset{R_{f_2}}{\rightsquigarrow} v\} = \{s, x_5\} \\
T &= V - S = \{x_1, x_2, x_3, x_4, t\}
\end{aligned}$$

Algorithm 3 therefore returns the cut consisting of the sets of vertices

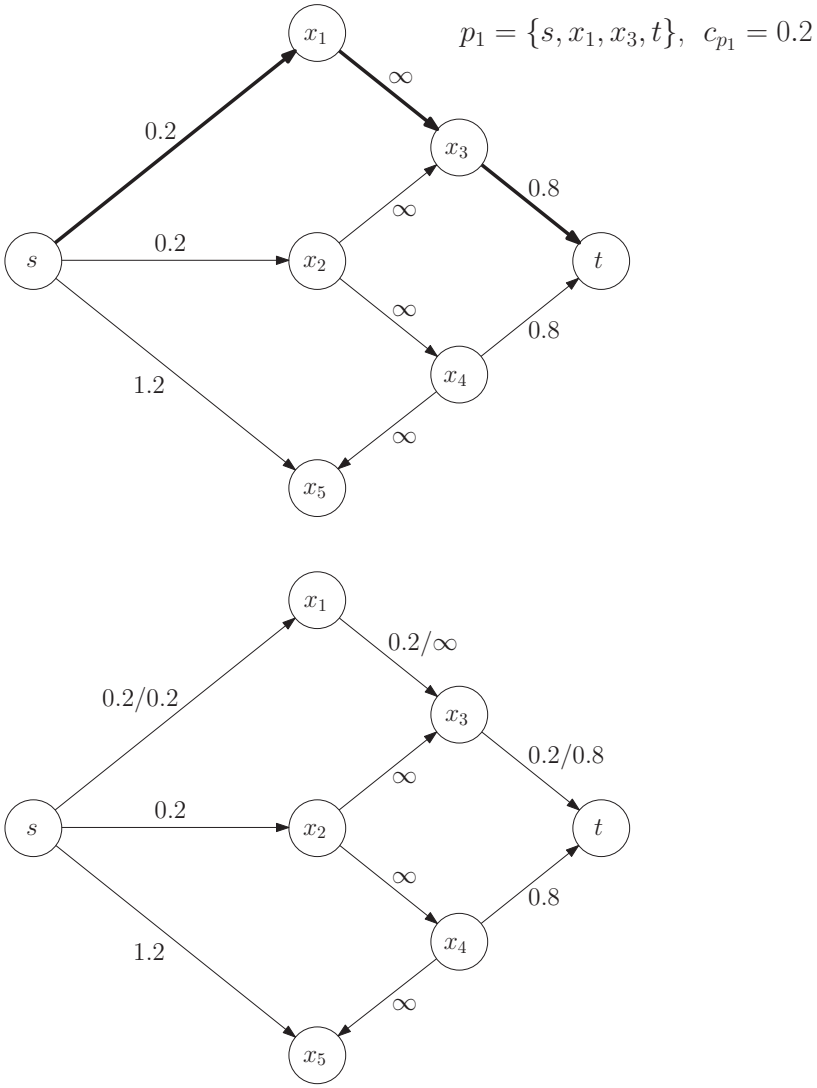

Figure 2.5: Divide-and-conquer algorithm example. First iteration of the Ford-Fulkerson algorithm applied to the set $Z = \{x_1, x_2, x_3, x_4, x_5\}$.

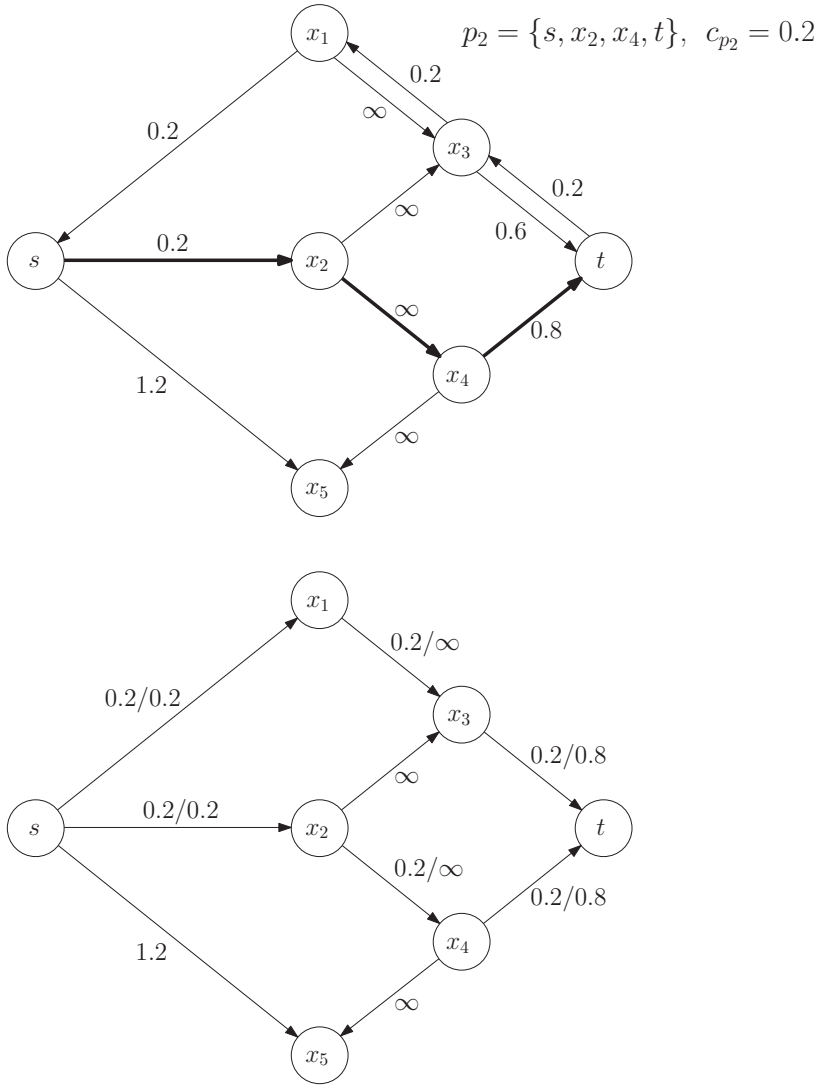

Figure 2.6: Divide-and-conquer algorithm example. Second iteration of the Ford-Fulkerson algorithm applied to the set $Z = \{x_1, x_2, x_3, x_4, x_5\}$.

$L = \{x_1, x_2, x_3, x_4\}$ and $U = \{x_5\}$. Since both L and U are non-empty, step 5 of algorithm 2 is performed; the execution of algorithm 3 on U returns the solution $Av_g(U) = 3$.

Figure 2.7 shows the first iteration of the Ford-Fulkerson algorithm applied to $Z' = L = \{x_1, x_2, x_3, x_4\}$. The top side represents the initial residual network R'_f with the augmenting path p'_1 found in it shaded. The bottom side shows the new flow $f'_1 = f' \uparrow f'_{p'_1}$.

Figure 2.8 shows the second and final iteration of the Ford-Fulkerson algorithm applied to Z'. The top side represents the new residual network $R_{f'_1}$ with the augmenting path p'_2 shaded. The bottom side shows the new flow $f'_2 = f'_1 \uparrow f_{p'_2}$ which results from augmenting f'_1 by $f_{p'_2}$.

There no other augmenting paths, and f'_2 is a maximum-flow for the network. The associated $s - t$ cut (S', T') is given by

$$S' = \{s\}$$
$$T' = V' - S' = \{x_1, x_2, x_3, x_4, t\}$$

Algorithm 3 returns the two sets $L' = \{x_1, x_2, x_3, x_4\}$ and $U' = \emptyset$. As a consequence, the execution of algorithm 2 on L returns the solution $Av_g(Z') = 1.5$.

The final solution returned by Algorithm 2 is the tuple

$$g^* = (1.5, 1.5, 1.5, 1.5, 3).$$

Figure 2.9 shows the order graph for Z labelled with the values of the isotonic regression g^*.

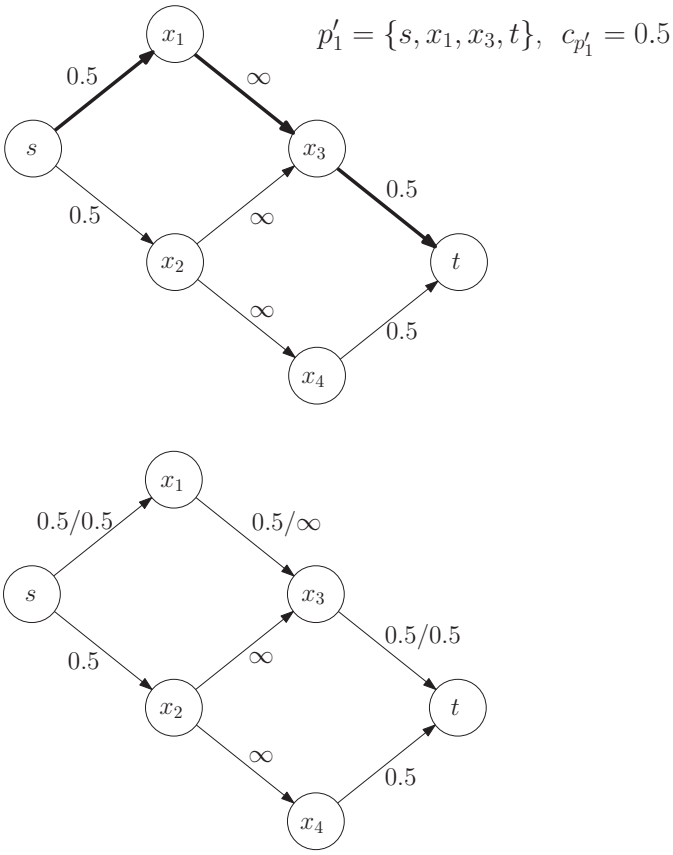

Figure 2.7: Divide-and-conquer algorithm example. First iteration of the Ford-Fulkerson algorithm applied to the set $Z' = \{x_1, x_2, x_3, x_4\}$.

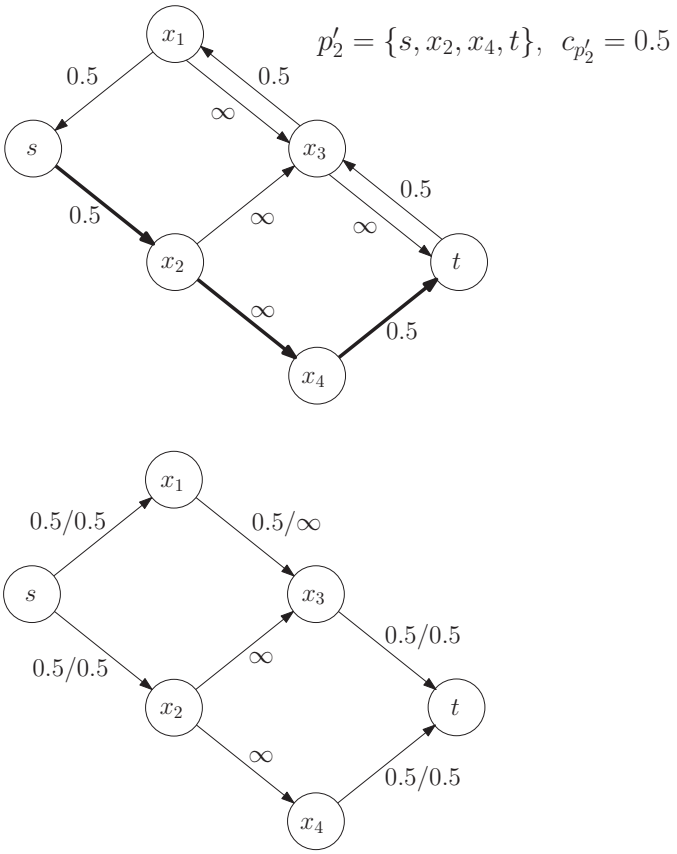

Figure 2.8: Divide-and-conquer algorithm example. Second iteration of the Ford-Fulkerson algorithm applied to the set $Z' = \{x_1, x_2, x_3, x_4\}$.

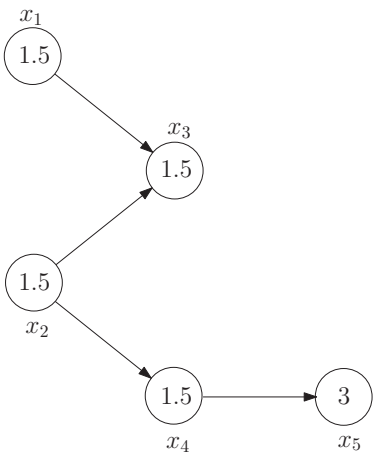

Figure 2.9: Divide-and-conquer algorithm example. Order graph made consistent after performing the isotonic regression.

Chapter 3

Monotonic Classification with MOCA

In this chapter we present the MOnotonic Classification Algorithm (MOCA), a non-parametric classification algorithm for problems in which there exists a monotonicity relationship between the descriptive attributes and the class label. During the training phase, MOCA computes monotonic estimates of the posterior class probabilities for the training sample. An interpolation scheme guaranteed to preserve the monotonicity property is used to extended the posterior distributions estimated for the training sample to new, unlabelled observation, which are then assigned to the class corresponding to the (smallest) median. MOCA's allocation rule is guaranteed to minimise the mean absolute error (see equation 2.3.7) computed on the training data.

We compared MOCA to the related OSDL algorithm [21, 77], both on artificial and real-world data sets, and showed that MOCA often outperforms OSDL with respect to mean absolute prediction error.

Because the posterior probability estimates used by OSDL and MOCA are often based on very few observations, we conjectured that both methods might be prone to overfitting. Therefore we used in both methods a smoothed version of the basic estimates which take into account observations near to where an estimate is required. We then performed a second round of experiments to verify whether this improved either classifier's performance.

This chapter is organised as follows:

- In section 3.1, we introduce the pattern recognition problem of classification and how it can be solved in a probabilistic setting.
- In section 3.2, we formalise the concept of monotonic classification. We introduce how MOCA estimates the posterior class probabilities for each training observation, formulate the MOCA allocation rule, and show that this allocation rule minimises L_1 loss on the training data; we then illustrate the interpolation scheme used to extend to new, unlabelled observations the estimates computed during the training step.
- In section 3.3, we discuss related work, focusing in particular on the OSDL algorithm by establishing similarities with and differences from MOCA.
- In section 3.4, we provide a small example to illustrate how both methods operate.
- In section 3.5, we describe the experimental comparison we performed of OSDL and MOCA on both artificial and real-world data sets.
- In section 3.6, we propose a strategy to smooth the basic estimates that are used by OSDL and MOCA.
- In section 3.7, we show the results of a second round of experiments conducted to test whether significant differences in predictive performance can be found between the original algorithms and their adapted counterparts.
- In section 3.8 we draw our conclusions on MOCA and OSDL and, in particular, on the use of smoothed class-probability estimates in them.

3.1 Classification

Classification is the predictive data mining task of identifying to which of a set of categories (sub-populations) a new observation belongs based on a training set of data comprising observations (or *instances*) whose category membership is known. One way to solve this problem is by casting it as a

pattern recognition problem (see section 2.4) in which what is predicted is a *class label* $y \in \mathcal{Y}$. A classification algorithm is also called a *classifier*.

The goal of a classification algorithm is therefore to learn how to assign a label $y \in \mathcal{Y}$ to an element $\mathbf{x} \in \mathcal{X}$. Assuming that the descriptive attributes have an influence on the target attribute, the algorithm observes a collection of training data S in order to infer the underlying and unknown *allocation* or *decision rule*

$$c : \mathcal{X} \to \mathcal{Y}$$

which maps any element $\mathbf{x} \in \mathcal{X}$ to one of the classes $y \in \mathcal{Y}$. A decision rule partitions the feature space $\mathcal{X}$ into k regions $\Omega_1, \ldots, \Omega_q$ such that if $\mathbf{x} \in \Omega_i$ then $\mathbf{x}$ is assigned to class y_i.

As shown in section 2.4, this problem can be conveniently solved by applying statistical decision theory because it allows to quantify the uncertainty associated with the possible classification decisions by using probabilities and because it allows to quantify the costs that accompany classification decisions by using an loss functions. A way to chose the optimal decision rule consists of the minimal Bayes risk principle (see definition 21).

A loss function is chosen to express the cost incurred for choosing to assign an observation $\mathbf{x}$ to class y_i when its true class is y_j; the cost shall be equal to 0 if $j = i$. This makes it possible to deal with situations in which some kinds of classification mistakes are more costly than others; for instance, in a medical diagnosis problem, it is far worse to classify a patient with severe thyroid abnormality as healthy (or having acid reflux) than the other way round.

Frequently, wrong classification decisions are assumed equally costly for all classes. One does not weigh a loss concerning misclassification of each label; rather, one is only interested in judging whether or not a classification is correct. The loss function of interest for this case is therefore the zero-one loss function (2.3.8) as it assigns a loss of 0 for correct classification and a loss of 1 to any kind of classification error regardless of the class chosen.

Using the result mentioned at the end of section 2.3, when the zero-one loss is used, the minimal risk Bayes allocation rule consists of assigning an observation $\mathbf{x}$ the class label corresponding to the conditional mode $Mod_{Y|\mathbf{X}}$. Therefore, in the case of the 0/1 loss, the minimal risk Bayes

allocation rule coincides with the *Bayes rule for minimum error*

$$c_{\text{BAYES}}(\mathbf{x}) = \arg\max_{i=1,\ldots,k} P_i(\mathbf{x}), \tag{3.1.1}$$

which minimises the probability of error [33, 109].

Estimating the posterior class probabilities $P_j(\mathbf{x})$ becomes infeasible if the number p of features is large or when there are features which can take on a large number of values. On the other hand, estimating the probabilities $P_Y(y_i)$ and $P_{\mathbf{X}|Y=y_i}(\mathbf{x})$ is a more tractable problem. Therefore, since because of the Bayes theorem we have that

$$P_i(\mathbf{x}) = \frac{P_Y(y_i)P_{\mathbf{X}|Y=y_i}(\mathbf{x})}{P_{\mathbf{X}}(\mathbf{x})},$$

the Bayes rule for minimum error actually used is

$$c'_{\text{BAYES}}(\mathbf{x}) = \arg\max_{i=1,\ldots,k} P_Y(y_i)P_{\mathbf{X}|Y=y_i}(\mathbf{x}). \tag{3.1.2}$$

An example of a classifier adopting allocation rule (3.1.2) is represented by the Naïve Bayes classifier [33, 109].

The Naïve Bayes classifier is an example of *probabilistic classifier*. Algorithms of this category use statistical inference to find the best class for a given instance, and, instead of simply returning the "best" class for a given observation, they output a probability of the instance being a member of each of the possible classes.

In particular, the Naïve Bayes classifier is an example of *generative classifier* as it is based on estimating the joint distribution of $(\mathbf{X}, Y)$. MOCA, instead, is an example of *discriminative classifier* as it is based on directly estimating the posterior class probabilities $P_i(\mathbf{x})$.

3.2 Monotonic Classification with MOCA

Assuming that the feature space $\mathcal{X}$ is partially ordered, that the set of labels $\mathcal{Y}$ is totally ordered, and that the unknown functional relationship between $\mathcal{X}$ and $\mathcal{Y}$ is monotonic, the objective of a *monotonic* or *monotonicity-preserving classifier* is to infer a *monotonic* or *monotonicity-preserving al-*

location rule $c : \mathcal{X} \to \mathcal{Y}$, with

$$\mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \Rightarrow c(\mathbf{x}) \leq_{\mathcal{Y}} c(\mathbf{x}') \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \tag{3.2.1}$$

namely such that a lower ordered input is not allowed to have a higher class label.

Notation. Without loss of generality and for the sake of convenience, we shall assume that

$$\mathcal{Y} = \{1, 2, \ldots, k\}.$$

Moreover, we shall assume that the total order on $\mathcal{Y}$ is the usual order on integers, and, therefore, we shall use the standard sign $\leq$ for inequalities between integers.

The monotonicity constraint on the allocation rule being learned represents additional information that a monotonic classifier should try to exploit. Monotonic classification is halfway between classification and regression because, as is the case with classification, the output space is finite, and because, as is the case with regression, the output space is totally ordered. Nonetheless, direct application of classification or regression methods to monotonic classification is not an appropriate choice for several reasons. First of all, traditional classification and regression algorithm shall ignore the monotonic relationship between input and output spaces. Secondly, classification algorithms shall ignore the ordering on the input and output spaces, while regression algorithms shall only exploit the ordering on the output space, unnecessarily assuming meaningful distances between output values.

Monotonic classification should not be confused with *ordinal classification*, whose only difference from general classification in that it is assumed that the set of labels is linearly ordered [45].

The crucial aspects to deal with in order to build a probabilistic classifier which is also monotonic are translating monotonicity in the population being analysed into constraints on the estimated probability distributions and incorporating this knowledge into the allocation rule used by the classifier.

The first goal can be achieved by imposing a *stochastic ordering* on the estimated posterior class probabilities $P_i(\mathbf{x})$, namely an ordering which

quantifies the concept of one (estimated) distribution being "bigger" than another. The form of stochastic ordering which is enforced in our algorithm is *weak* (*first order*) *stochastic dominance*, which can be formulated as follows:

$$P_{Y_1} \prec P_{Y_2} \Longleftrightarrow \forall x \in \mathbb{R} : F_{Y_1}(x) \geq F_{Y_2}(x), \tag{3.2.2}$$

for every two random variables Y_1 and Y_2 defined on the same probability space.

Although there exist other forms of stochastic ordering [74], we have chosen stochastic dominance because, when applied to the posterior class probabilities $P_i(\mathbf{x})$, it enforces the idea that increasing values of $\mathbf{x}$ shift the posterior class probabilities $P_i(\mathbf{x})$ upwards $\forall i = 1, \ldots, k$. This form of stochastic ordering is frequently adopted in decision theory and risk management and has been used in the context of Bayesian networks [107].

MOCA produces estimates of the posterior class probabilities $P_i(\mathbf{x})$ which satisfy the following *stochastic monotonicity constraint*:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : \mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \Rightarrow P_{Y|\mathbf{X}=\mathbf{x}} \prec P_{Y|\mathbf{X}=\mathbf{x}'}, \tag{3.2.3}$$

namely

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : \mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \Rightarrow \forall x \in \mathbb{R} : F_{Y|\mathbf{X}=\mathbf{x}}(x) \geq F_{Y|\mathbf{X}=\mathbf{x}'}(x),$$

which, due to (2.2.4), is equivalent to

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : \mathbf{x} \leq_{\mathcal{X}} \mathbf{x}' \Rightarrow \forall i = 1, \ldots, k : F_i(\mathbf{x}) \geq F_i(\mathbf{x}'). \tag{3.2.4}$$

Constraint (3.2.4) is key to achieving the goal of incorporating monotonicity in the population being analysed into an allocation rule: in the following it shall be shown that if it is enforced, then the allocation rule consisting in assigning an observation $\mathbf{x}$ to the class corresponding to a consistently-chosen median of the posterior class distribution $P_i(\mathbf{x})$ is monotonic.

3.2.1 Training Phase

In this section we introduce the monotonicity-preserving estimator which MOCA uses to estimate the posterior-class cumulative distribution functions for the distinct feature vectors $S_{\mathcal{X}}$ observed in a training data set S.

Definition 34 (Maximum Likelihood Probability Estimate)**.** For $\mathbf{x} \in S_{\mathcal{X}}$, the *unconstrained maximum likelihood estimate* of $P_i(\mathbf{x})$ for all $i = 1, \ldots, k$ is

$$\hat{P}_i(\mathbf{x}) = \frac{n(\mathbf{x}, i)}{n(\mathbf{x})}. \tag{3.2.5}$$

Definition 35 (Maximum Likelihood CDF Estimate)**.** For $\mathbf{x} \in S_{\mathcal{X}}$, the *unconstrained maximum likelihood estimate* of $F_i(\mathbf{x})$ for all $i = 1, \ldots, k$ is

$$\hat{F}_i(\mathbf{x}) = \sum_{j \leq i} \hat{P}_j(\mathbf{x}). \tag{3.2.6}$$

Equation (3.2.6) represents the proportion of observations of a feature vector $\mathbf{x}$ included in the training data whose label is less than or equal to i.

Definition 36 (MOCA Estimator)**.** For $\mathbf{x} \in S_{\mathcal{X}}$ and $\forall i = 1, \ldots, k$, the MOCA estimator of $F_i(\mathbf{x})$ is the function

$$F_i^*(\mathbf{x}) = \begin{cases} g_i^*(\mathbf{x}), & \text{if } i = 1, \ldots, k-1 \\ 1 & \text{otherwise} \end{cases}, \tag{3.2.7}$$

where $g_i^*(\mathbf{x})$ is the antitonic regression of $\hat{F}_i(\mathbf{x})$, $\forall \mathbf{x} \in S_{\mathcal{X}}$, with weights $w(\mathbf{x}) = n(\mathbf{x})$. (Only $k-1$ isotonic regressions are required, for obviously $F_k^*(\mathbf{x}) = 1$, $\forall \mathbf{x} \in S_{\mathcal{X}}$.)

Proposition 13. *$F_i^*(\mathbf{x})$, $\forall i = 1, \ldots, k$, is a cumulative distribution function for a discrete random variable.*

Proof. $F^*(\mathbf{x})$ is a distribution function for a discrete random variable because it satisfies the following properties:

1. $0 \leq F_i^*(\mathbf{x}) \leq 1$, for $i = 1, \ldots, k$;
2. $i \geq j \Rightarrow F_i^*(\mathbf{x}) \geq F_j^*(\mathbf{x})$.

As $F_i^*(\mathbf{x})$ is a weighted average of observed relative frequencies (see definition 26), the first condition follows. Moreover, since we have that $\hat{F}_i(\mathbf{x}) \geq \hat{F}_j(\mathbf{x})$ for $i \geq j$, it follows that for every set A

$$Av_{\hat{F}_i}(A) \geq Av_{\hat{F}_j}(A).$$

Hence, it follows from proposition 9 that $F_i^*(\mathbf{x}) \geq F_j^*(\mathbf{x})$. □

The MOCA estimator satisfies stochastic monotonicity constraint (3.2.4) by construction, namely $\forall \mathbf{x}, \mathbf{x}' \in S_{\mathcal{X}}$:

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow F_i^*(\mathbf{x}) \geq F_i^*(\mathbf{x}') \qquad i = 1, \ldots, k.$$

This estimator has already been used for estimation under stochastic order constraints in the past. It was proposed for linear orders already by Hogg [64], and later analysed by Barmi and Mukerjee [36]; it was also used by Feelders [38] for parameter estimation in Bayesian networks under a stochastic order constraint.

3.2.2 Prediction Phase

This section describes how the posterior-class cumulative distribution functions of new, unlabelled feature vectors are estimated by interpolation of the cumulative distribution functions estimated for the set of distinct feature vectors $S_{\mathcal{X}}$ observed labelled in the training data set $S_{\mathcal{X}}$ during the training phase.

For every $\mathbf{x}_0 \in \mathcal{X}$, we define the quantities $F_i^{\min}(\mathbf{x}_0)$ and $F_i^{\max}(\mathbf{x}_0)$ as follows:

$$F_i^{\min}(\mathbf{x}_0) = \max_{\mathbf{x} \in \uparrow \mathbf{x}_0} F_i^*(\mathbf{x}) \quad i = 1, \ldots, k, \tag{3.2.8}$$

and

$$F_i^{\max}(\mathbf{x}_0) = \min_{\mathbf{x} \in \downarrow \mathbf{x}_0} F_i^*(\mathbf{x}) \quad i = 1, \ldots, k. \tag{3.2.9}$$

If $\uparrow \mathbf{x}_0$ is empty, we then define

$$F_i^{\max}(\mathbf{x}_0) = 1 \quad i = 1, \ldots, k, \tag{3.2.10}$$

and, if $\downarrow \mathbf{x}_0$ is empty, we then define

$$F_i^{\min}(\mathbf{x}_0) = \begin{cases} 0 & i = 1, \ldots, k-1 \\ 1 & i = k \end{cases} \tag{3.2.11}$$

Theorem 4. *For every* $\mathbf{x}_0 \in \mathcal{X}$

$$F_i^{\max}(\mathbf{x}_0) \geq F_i^{\min}(\mathbf{x}_0), \quad \forall i = 1, \ldots, k.$$

Proof. Let us suppose that both $\downarrow \mathbf{x}_0$ and $\uparrow \mathbf{x}_0$ are non-empty, and let us consider two elements $\mathbf{x}' \in \downarrow \mathbf{x}_0$ and $\mathbf{x}'' \in \uparrow \mathbf{x}_0$ and $i \in \{1, \ldots, k\}$. Because $\mathbf{x}' \leq \mathbf{x}''$, it follows that

$$F_i^*(\mathbf{x}') \geq F_i^*(\mathbf{x}'').$$

Since this is true for any $\mathbf{x}' \in \downarrow \mathbf{x}_0$ and $\mathbf{x}'' \in \uparrow \mathbf{x}_0$ for any $i \in \{1, \ldots, k\}$, the theorem follows in particular.

If $\downarrow \mathbf{x}_0$ is empty, the theorem also follows in this case because

$$F_i^{\min}(\mathbf{x}_0) = 0 \leq F_i^{\max}(\mathbf{x}_0) \quad \forall i = 1, \ldots, k-1$$

and because, by the definition of cumulative distribution function,

$$F_k^{\min}(\mathbf{x}_0) = 1 = F_k^{\max}(\mathbf{x}_0).$$

Finally, if $\uparrow \mathbf{x}_0$ is empty, the theorem follows because

$$F_i^{\min}(\mathbf{x}_0) \leq 1 = F_i^{\max}(\mathbf{x}_0) \quad \forall i = 1, \ldots, k.$$

□

From theorem 4, it follows that any element of the interval

$$[F_i^{\min}, F_i^{\max}]$$

is an estimate of $F_i(\mathbf{x}_0)$ which satisfies the stochastic order constraint consistently with the values of $F_i(\mathbf{x})$ for $\mathbf{x} \in S_{\mathcal{X}}$ estimated with the MOCA estimator.

The way MOCA selects a value from the interval $[F_i^{\min}, F_i^{\max}]$ is by considering a convex combination of $F_i^{\min}$ and $F_i^{\max}$, that is

$$\tilde{F}_i(\mathbf{x}_0) = \alpha F_i^{\min}(\mathbf{x}_0) + (1-\alpha) F_i^{\max}(\mathbf{x}_0), \qquad \alpha \in [0,1]. \tag{3.2.12}$$

choosing the value of α by minimising the empirical loss on a test sample.

It should be noted that for $\mathbf{x}_0 \in S_{\mathcal{X}}$, we have that

$$\tilde{F}_i(\mathbf{x}_0) = F_i^*(\mathbf{x}_0)$$

since both $F_i^{\min}(\mathbf{x}_0)$ and $F_i^{\max}(\mathbf{x}_0)$ coincide with $F_i^*(\mathbf{x}_0)$.

Figure 3.1 illustrates an example of the interpolation scheme adopted by MOCA.

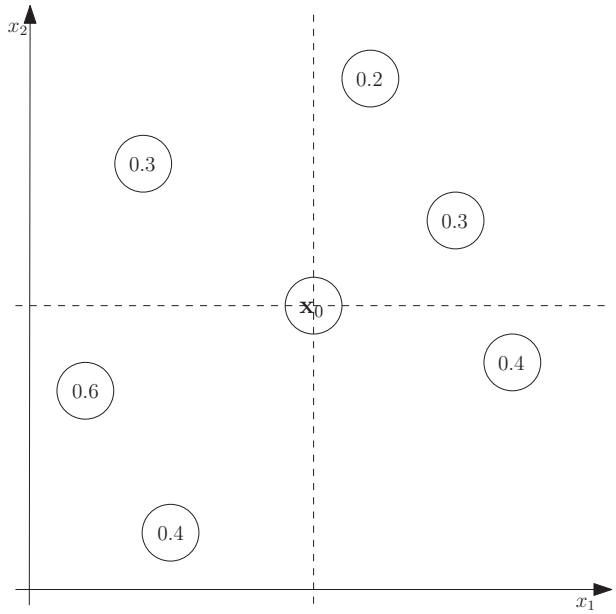

Figure 3.1: MOCA interpolation scheme example. Since $F_i^{\min}(\mathbf{x}_0) = \max_{\mathbf{x} \leq \mathbf{x}_0} F_i^*(\mathbf{x}) = \max\{0.2, 0.3\} = 0.3$ and $F_i^{\max}(\mathbf{x}_0) = \min_{\mathbf{x}_0 \leq \mathbf{x}} F_i^*(\mathbf{x}) = \min\{0.6, 0.4\} = 0.4$, any element of the interval $[0.3, 0.4]$ is an estimate of $F_i(\mathbf{x}_0)$ which satisfies the stochastic order constraint with respect to the values of $F_i(\mathbf{x})$ for $\mathbf{x} \in S_{\mathcal{X}}$ estimated with the MOCA estimator.

Besides being convenient, the interpolation scheme (3.2.12) is guaran-

teed to produce globally-monotonic estimates, as we prove in the following.

Lemma 2. *For all* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *in* $\mathcal{X}$*, if* $\mathbf{x}_1 \leq \mathbf{x}_2$ *then*

$$F_i^{\min}(\mathbf{x}_1) \geq F_i^{\min}(\mathbf{x}_2) \quad \forall i = 1, \ldots, k, \tag{3.2.13}$$

and

$$F_i^{\max}(\mathbf{x}_1) \geq F_i^{\max}(\mathbf{x}_2) \quad \forall i = 1, \ldots, k. \tag{3.2.14}$$

Proof. Since $\mathbf{x}_1 \leq \mathbf{x}_2$, we have that

$$\uparrow\mathbf{x}_2 \subseteq \uparrow\mathbf{x}_1.$$

Therefore,

$$\max_{\mathbf{x}\in\uparrow\mathbf{x}_1} F_i^*(\mathbf{x}) \geq \max_{\mathbf{x}\in\uparrow\mathbf{x}_2} F_i^*(\mathbf{x}),$$

and, as a consequence,

$$F_i^{\min}(\mathbf{x}_1) \geq F_i^{\min}(\mathbf{x}_2)$$

by definition.

Because $\mathbf{x}_1 \leq \mathbf{x}_2$, $\uparrow\mathbf{x}_1$ is not empty as it contains $\mathbf{x}_2$. On the other hand, $\mathbf{x}_1$ can be empty; if that is the case, then

$$F_i^{\min}(\mathbf{x}_1) \geq F_i^{\min}(\mathbf{x}_2)$$

because in this case $F_i^{\min}(\mathbf{x}_2)$ dominates every other distribution.

The proof of equation (3.2.14) is analogous. □

Theorem 5. *For all* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *in* $\mathcal{X}$ *we have that*

$$\mathbf{x}_1 \leq \mathbf{x}_2 \Rightarrow \tilde{F}_i(\mathbf{x}_1) \geq \tilde{F}_i(\mathbf{x}_2), \quad \forall i = 1, \ldots, k.$$

Proof. For $\alpha \in [0, 1]$, from lemma 2 it follows that

$$\begin{aligned}\tilde{F}_i(\mathbf{x}_1) &= \alpha F_i^{\min}(\mathbf{x}_1) + (1-\alpha)F_i^{\max}(\mathbf{x}_1) \geq \\ &\geq \alpha F_i^{\min}(\mathbf{x}_2) + (1-\alpha)F_i^{\max}(\mathbf{x}_2) = \tilde{F}_i(\mathbf{x}_2) \quad \forall i = 1, \ldots, k\end{aligned}$$

□

Finally, we notice that, given the two random variables X, X' and their respective interval of medians $[m_\ell, m_u]$, $[m'_\ell, m'_u]$, then [77]

$$P_X \prec P_{X'} \Rightarrow m_\ell \leq m'_\ell,\, m_u \leq m'_u.$$

As a consequence, in order to obtain a monotonicity-preserving allocation rule, it suffices to choose the label corresponding to the same position in the interval of medians (e.g. the lower end point, the upper end point, or the midpoint).

Definition 37 (MOCA allocation rule)**.** The MOCA allocation rule c_{MOCA} consists of assigning a new observation $\mathbf{x}$ the label corresponding to the smallest median of the estimates $\tilde{F}(\mathbf{x})$, namely

$$c_{\text{MOCA}}(\mathbf{x}) = \min_i : \tilde{F}_i(\mathbf{x}) \geq 0.5. \tag{3.2.15}$$

It should be noticed that the choice of the smallest median in (3.2.15) is purely arbitrary and only made for the sake of consistency.

Another possible allocation rule would be assigning a new observation $\mathbf{x}$ to the class corresponding to the conditional expectation $E_Y(\mathbf{x})$ because we have that [86]

$$P_{Y|\mathbf{X}=\mathbf{x}} \prec P_{Y|\mathbf{X}=\mathbf{x}'} \Rightarrow E_Y(\mathbf{x}) \leq E_Y(\mathbf{x}')$$

and because the expectation minimises the squared loss function (see section 2.4.3). This choice, though, would require the transformation of the ordinal scale on the set of possible labels $\mathcal{Y}$ into an interval scale [102]. The consequence would be a limitation of the possible applicability of our algorithm, whose only assumption about $\mathcal{Y}$ is that it is totally ordered.

3.2.3 Empirical L_1 Loss Minimisation

In the previous section we proved that the MOCA allocation rule preserves monotonicity. In this section we shall prove that it minimises the empirical absolute risk (2.4.4) within the class of monotonic classification functions.

Although this result is not immediate to prove, it seems plausible because, as already mentioned in section 2.3, predicting a median minimises

the L_1 loss function. This property makes the MOCA allocation rule the correspondent of the Bayes allocation rule (3.1.1) with the 0/1 loss function replaced by the L_1 loss function. The reason to prefer the L_1 loss function to the 0/1 loss function has been expressed in section 2.4.3, where it has also been stated that another appropriate choice in the presence of monotonicity is represented by the squared loss function (2.3.3). Nevertheless, the L_1 loss function was a reasonable candidate, so we focused our attention on it.

We base our proof on Dykstra et al. [35], which was one of the first publications to have addressed the problem of exact minimisation of an empirical risk in the context of monotonic classification. The authors describe two ways to derive a monotonic allocation rule via the isotonic regression which minimises the empirical squared risk (2.4.3) and the empirical absolute risk (2.4.4) respectively. To minimise the empirical squared risk, a single isotonic regression and a rounding to the nearest integer is sufficient, whereas, in order to minimise the empirical absolute risk, $k-1$ isotonic regressions and roundings are required. Therefore, in order to show that c_{MOCA} minimises the empirical absolute risk (2.4.4) within the class of monotonic integer-valued functions $c(\cdot)$, it suffices to prove that it satisfies all the requirements of Dykstra et al.'s method.

To describe their result, let us first define the relation $\preceq$ as follows:

$$B_i \preceq B_j \Leftrightarrow \exists x_i \in B_i, \exists x_j \in B_j : x_i \leq x_j, \quad \forall B_i, B_j \subset Z,$$

that is, $B_i \preceq B_j$ if block B_i contains an element which precedes an element from block B_j. Is should be noted that $\preceq$ is not transitive.

Definition 38 (Maximal Partition)**.** Let $Z = \{z_1, z_2, \ldots, z_n\}$ be a non-empty, partially-ordered, finite set of constants, each of which is associated with a real number $g(z_i)$ and with a positive real weight w_i, and let $\prec$ denote the transitive closure of $\preceq$. A *maximal partition* of Z with respect to the isotonic regression g^* of g is a partition $B_1, \ldots, B_m$ of Z such that for all $j = 1, \ldots, m$:

1. $\forall z \in B_j : g^*(z) = Av_g(B_j)$
2. $(\mathcal{B}, \prec)$ is a partially-ordered set
3. m is as large as possible

Definition 38 is slightly different from the one used in [35] because the latter does not guarantee the uniqueness of the maximal partition [101].

Intuitively, a maximal partition splits Z into the smallest possible blocks which have the property that the isotonic regression is obtained by taking the block average.

To compute the maximal partition, Dykstra et al. [35] suggest a variant of the Minimum Lower Sets algorithm (see section 2.5.1) in which in line 3 a minimum lower set of smallest size rather than the largest one is picked. This corresponds to rewriting line 3 of the algorithm as

$$B \leftarrow \arg\min_{|A|}\{A \in \mathcal{L} \mid Av_g(A) = \min_{L \in \mathcal{L}}; Av_g(L)\}. \qquad \text{(Alg. 1; 3')}$$

The problem with this approach is that the number of lower sets can grow exponentially with the size of the partial order. For example, the number of lower sets of the product order on the space spanned by just six binary attributes is approximately 8 million. As an alternative to the MLS algorithm, which is therefore of no practical use in problems of realistic size, Stegeman and Feelders propose in [101] a more efficient alternative to compute a maximal partition for g^* which is based on the divide and conquer algorithm (see section 2.5.2).

Next, we state the result of Dykstra et al. from which the optimality of the MOCA allocation rule follows.

Proposition 14 (Dykstra et al. [35])**.** *For all $\mathbf{x} \in S_{\mathcal{X}}$, let $p_i^*(\mathbf{x})$ denote the isotonic regression of $p_i(\mathbf{x}) = 1 - \hat{F}_i(\mathbf{x})$. An isotonic allocation rule $c(\mathbf{x})$ minimises the empirical absolute risk* (2.4.4) *if and only if the following three conditions are met for all $i = 1, \ldots, k-1$:*

1. *If $p_i^*(\mathbf{x}) < \frac{1}{2}$, then $c(\mathbf{x}) \leq i$*
2. *If $p_i^*(\mathbf{x}) > \frac{1}{2}$, then $c(\mathbf{x}) > i$*
3. *$c(\mathbf{x})$ is constant and equal to either of i and $i+1$ on each element of the maximal partition with respect to p_i^* which is a subset of $\{\mathbf{x} : p_i^*(\mathbf{x}) = \frac{1}{2}\}$*

Theorem 6. *For all $\mathbf{x} \in S_{\mathcal{X}}$, let $p_i^*(\mathbf{x})$ denote the isotonic regression of*

$p_i(\mathbf{x}) = 1 - \hat{F}_i(\mathbf{x})$ *and let* $F_i^*(\mathbf{x})$ *denote the* MOCA *estimator. Then*

$$p_i^*(\mathbf{x}) = 1 - F_i^*(\mathbf{x}), \quad \forall i = 1, \ldots, k-1. \tag{3.2.16}$$

Proof. It holds that

$$\begin{aligned} Av(A, p_i) &= \frac{\sum_{\mathbf{x}\in A} w(\mathbf{x})(1 - \hat{F}_i(\mathbf{x}))}{\sum_{\mathbf{x}\in A} w(\mathbf{x})} \\ &= 1 - \frac{\sum_{\mathbf{x}\in A} w(\mathbf{x})\hat{F}_i(\mathbf{x})}{\sum_{\mathbf{x}\in A} w(\mathbf{x})} \\ &= 1 - Av(A, \hat{F}_i) \end{aligned} \tag{3.2.17}$$

Therefore, from (2.5.4) and (3.2.17), it follows that

$$\begin{aligned} p_i^*(\mathbf{x}) &= \min_{L:\mathbf{x}\in L} \max_{U:\mathbf{x}\in U} Av(L \cap U, p_i) \\ &= \min_{L:\mathbf{x}\in L} \max_{U:\mathbf{x}\in U} 1 - Av(L \cap U, \hat{F}_i) \\ &= \min_{L:\mathbf{x}\in L} \left\{ 1 - \min_{U:\mathbf{x}\in U} Av(L \cap U, \hat{F}_i) \right\} \\ &= 1 - \left\{ \max_{L:\mathbf{x}\in L} \min_{U:\mathbf{x}\in U} Av(L \cap U, \hat{F}_i) \right\}, \end{aligned}$$

where, because of (2.5.5), the quantity

$$\max_{L:\mathbf{x}\in L} \min_{U:\mathbf{x}\in U} Av(L \cap U, \hat{F}_i)$$

is the antitonic regression $F_i^*(\mathbf{x})$ of $\hat{F}_i(\mathbf{x})$. □

Corollary 2. *For all* $\mathbf{x} \in S_{\mathcal{X}}$*, let* $F_i^*(\mathbf{x})$ *denote the* MOCA *estimator* (3.2.7)*. An isotonic allocation rule* $c(\mathbf{x})$ *minimises the empirical absolute risk* (2.4.4) *if and only if the following three conditions are met for all* $i = 1, \ldots, k-1$*:*

1. *If* $F_i^*(\mathbf{x}) > \frac{1}{2}$*, then* $c(\mathbf{x}) \leq i$
2. *If* $F_i^*(\mathbf{x}) < \frac{1}{2}$*, then* $c(\mathbf{x}) > i$
3. $c(\mathbf{x})$ *is constant and equal to either of* i *and* $i+1$ *on each element of the maximal partition with respect to* F_i^* *which is a subset of* $\{\mathbf{x} : F_i^*(\mathbf{x}) = \frac{1}{2}\}$

Therefore, to show that the MOCA allocation rule $c_{\text{MOCA}}(\mathbf{x})$ minimises the empirical absolute risk (2.4.4), it suffices to show that the three conditions stated in corollary 2 are satisfied.

The first two conditions follow directly from the definition of c_{MOCA} (3.2.15):

1. $F_i^*(\mathbf{x}) > \frac{1}{2} \Rightarrow c_{\text{MOCA}}(\mathbf{x}) \leq i$
2. $F_i^*(\mathbf{x}) < \frac{1}{2} \Rightarrow c_{\text{MOCA}}(\mathbf{x}) > i$

The third condition of corollary 2 is necessary because attribute vectors that belong to the same element of the maximal partition must have the same class label in an optimal solution. Hence, if there is a choice of rounding to i or $i+1$, it must be done in such a way that vectors belonging to the same element of the maximal partition are rounded to the same class label. It follows from definition 37 that for $\mathbf{x} \in S_{\mathcal{X}}$:

$$c_{\text{MOCA}}(\mathbf{x}) = \min_i : F_i^*(\mathbf{x}) \geq 0.5.$$

As a consequence, MOCA rounds all attribute vectors in $\{\mathbf{x} : F_i^*(\mathbf{x}) = \frac{1}{2}\}$ to the lower value, and, therefore, c_{MOCA} is constant on each element of the maximal partition which is a subset of $\{\mathbf{x} : F_i^*(\mathbf{x}) = \frac{1}{2}\}$. Hence, c_{MOCA} also satisfies the third condition of corollary 2.

As an illustration, let us consider the data set described in table 3.1, whose feature space is ordered according to the graph given in figure 3.2. Note that the maximal partition for both F_1^* and F_2^* is $\{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_3, \mathbf{x}_4\}\}$. According to condition (3) of Corollary 2, $c(\cdot)$ must be constant and equal to 1 or 2 on $\{\mathbf{x}_1, \mathbf{x}_2\}$, and, likewise, $c(\cdot)$ must be constant and equal to 1, 2 or 3 on $\{\mathbf{x}_3, \mathbf{x}_4\}$. Table 3.2 lists the isotonic classifications which satisfy this condition together with the respective values of the loss function. There are 5 optimal solutions corresponding to assignments 1 to 5. Assignment 6 is suboptimal because it is not constant on block $\{\mathbf{x}_3, \mathbf{x}_4\}$. Assignment 1 corresponds to c_{MOCA}.

3.3 Related work

Dykstra et al [35] propose a non-parametric monotonic classification procedure which minimises L_1 loss on the training data set subject to monotonic-

	$n(\mathbf{x}, y)$			$\hat{F}$		F^*	
	1	2	3	1	2	1	2
$\mathbf{x}_1$	4	0	6	0.4	0.4	0.5	0.7
$\mathbf{x}_2$	6	4	0	0.6	1	0.5	0.7
$\mathbf{x}_3$	4	0	6	0.4	0.4	0.5	0.5
$\mathbf{x}_4$	6	0	4	0.6	0.6	0.5	0.5

Table 3.1: Optimality of the MOCA allocation rule example. Summary of the data set.

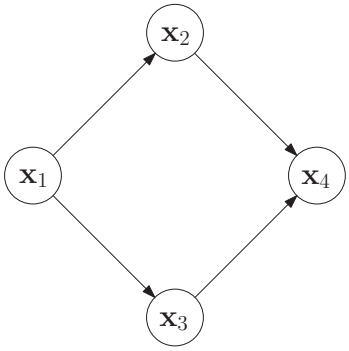

Figure 3.2: Optimality of the MOCA allocation rule example. Graph of the partial order on the distinct feature vectors.

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	absolute error
1	1	1	1	1	12+4+12+8=36
2	1	1	2	2	12+4+10+10=36
3	1	1	3	3	12+4+8+12=36
4	2	2	2	2	10+6+10+10=36
5	2	2	3	3	10+6+8+12=36
6	1	1	2	3	12+4+10+12=38 (suboptimal)

Table 3.2: Optimality of the MOCA allocation rule example. Loss of different isotonic classifications. There are 5 optimal solutions corresponding to assignments 1 to 5. Assignment 6 is suboptimal because it is not constant on block $\{\mathbf{x}_3, \mathbf{x}_4\}$. Assignment 1 corresponds to c_{MOCA}.

ity constraint (3.2.1). Their algorithm requires performing $k-1$ isotonic regressions to find an optimal solution. The authors also provide an algorithm which minimises L_2 loss on the training data set which requires performing a single isotonic regression. Dykstra et al. indicate possibilities to extend the relabelled training data to a monotonic prediction rule for the entire input space but without using any information in the training data beyond the ordering of the data points.

Kotlowski [72] shows that if a collection of probability distributions satisfies the stochastic order constraint (3.2.4), then the Bayes allocation rule c_{BAYES} (3.1.1) satisfies monotonicity constraint (3.2.1) provided the loss function is *convex*. This encompasses many reasonable loss functions but not 0/1 loss, unless the class label is binary. Kotlowski et al. [71] consider the problem of ordinal classification with monotonicity constraints in the context of rough sets; they also consider the loss-optimal relabelling problem and establish a connection with the isotonic regression. Kotlowski and Slowinski [73] use optimal relabelling as part of a boosting approach to the construction of monotonic classifiers.

Ryu et al. [24, 96] present a monotonic classification technique called isotonic separation. Also this technique involves the minimisation of empirical loss subject to the monotonicity constraint. The authors express this optimization problem as a linear program and show that the dual of the linear program is in fact a maximum-flow problem.

MOCA is related to the Ordinal Stochastic Dominance Learner (OSDL) developed by Cao-Van [21] and then generalized by Lievens et al. in [77]; therefore, we shall now give a short description of OSDL in order to point out similarities and differences between that algorithm and MOCA.

To obtain a collection of distribution functions which satisfy the stochastic monotonicity constraint, Cao-Van [21] defines for every $\mathbf{x}_0 \in \mathcal{X}$ the quantities $F_i^{\min}(\mathbf{x}_0)$ and $F_i^{\max}(\mathbf{x}_0)$ as follows:

$$F_i^{\min}(\mathbf{x}_0) = \min_{\mathbf{x} \in S_{\mathcal{X}}, \mathbf{x} \leq \mathbf{x}_0} \hat{F}_i(\mathbf{x}) \quad i = 1, \ldots, k, \tag{3.3.1}$$

and

$$F_i^{\max}(\mathbf{x}_0) = \max_{\mathbf{x} \in S_{\mathcal{X}}, \mathbf{x}_0 \leq \mathbf{x}} \hat{F}_i(\mathbf{x}) \quad i = 1, \ldots, k. \tag{3.3.2}$$

If there is no $\mathbf{x}$ in $S_{\mathcal{X}}$ such that $\mathbf{x} \leq_{\mathcal{X}} \mathbf{x}_0$, then he defines

$$F_i^{\min}(\mathbf{x}_0) = 1 \quad i = 1, \ldots, k$$

and, if there is no $\mathbf{x}$ in $S_{\mathcal{X}}$ such that $\mathbf{x}_0 \leq_{\mathcal{X}} \mathbf{x}$, then he defines

$$F_i^{\max}(\mathbf{x}_0) = \begin{cases} 0, & i = 1, \ldots, k-1 \\ 1 & i = k \end{cases}$$

It should be noted that $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$ such that $\mathbf{x} \leq_{\mathcal{X}} \mathbf{x}'$ the following inequalities are true:

$$F_i^{\min}(\mathbf{x}) \geq F_i^{\min}(\mathbf{x}') \tag{3.3.3}$$

$$F_i^{\max}(\mathbf{x}) \geq F_i^{\max}(\mathbf{x}') \tag{3.3.4}$$

Inequality (3.3.3) follows from the fact that the minimum of a partially-ordered set is never above the minimum of one of its subsets. Inequality (3.3.4) follows similarly.

In the "constant interpolation" version of OSDL (this is the version we have taken into consideration for our comparison; details on the more sophisticated "balanced" and "double balanced" interpolation schemes can be found in [77]), the cumulative distribution functions of $\mathbf{x}_0$ are estimated using the same interpolation schema used by MOCA, namely as

$$\tilde{F}_i(\mathbf{x}_0) = \alpha F_i^{\min}(\mathbf{x}_0) + (1-\alpha) F_i^{\max}(\mathbf{x}_0),$$

with $\alpha \in [0, 1]$.

These estimates satisfy the stochastic monotonicity constraint because of inequalities (3.3.3) and (3.3.4). This rule is used both for observed data points and for new data points. Similarly to MOCA, the value of the free parameter α can be selected by minimising cross validation error.

MOCA and OSDL are similar in the sense that they both make use of the same interpolation scheme. Nonetheless, an important difference between OSDL and MOCA is that, while in the former the interpolation scheme is based on the direct use of the maximum likelihood estimates $\hat{F}$, in the latter it is based on the use of the MOCA estimator F^*, namely the antitonic

regressions of the estimates $\hat{F}$. The most important consequence of this difference is that MOCA is guaranteed to minimise the empirical absolute risk (2.4.4), whereas this is not the case for OSDL. On the other hand, it should be noted that if $\hat{F}$ already satisfies the stochastic order constraint, then both methods are identical, for the isotonic regression shall not make any changes to $\hat{F}$ due to the absence of order violations.

Another important difference between OSDL and MOCA is represented by their respective allocation rules. Originally, Cao-Van [21] assigned an unlabelled individual $\mathbf{x}$ to the expected value of $\tilde{F}(\mathbf{x})$, rounded to the nearest integer. In [77] the allocation rule is changed to the class label corresponding to the median of the estimated posterior class probabilities $\hat{P}_i(\mathbf{x})$, but the choice of median is left unspecified provided that it is made so that the monotonicity constraint is satisfied. This could be interpreted as an attempt to minimise the empirical absolute risk, but this is not stated explicitly. In contrast, MOCA assigns $\mathbf{x}$ the class label corresponding to the smallest median.

3.4 Example

In this section we present an example to illustrate how MOCA and OSDL operate. Let us suppose we train both algorithms on a training data set S in which feature vectors take values in the 2-dimensional feature space

$$\mathcal{X} = \{1, 2, 3\} \times \{1, 2, 3\}$$

and have one of the class labels in the totally-ordered set

$$\mathcal{Y} = \{1, 2, 3\}.$$

The distinct features observed in the data set are

$$S_{\mathcal{X}} = \{(1, 1), (1, 2), (2, 1), (1, 3), (3, 2)\}.$$

The partial order on $S_{\mathcal{X}}$ is depicted in figure 3.3.

Table 3.3 lists, for every $\mathbf{x} \in S_{\mathcal{X}}$, the number of occurrences $n(\mathbf{x}, y)$ of $\mathbf{x}$ having label y, the total number of occurrences $n(\mathbf{x})$, and the values of

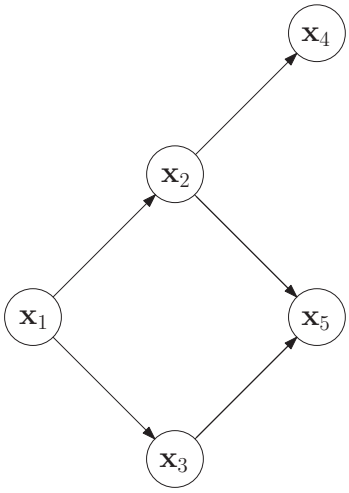

Figure 3.3: MOCA vs OSDL example. Graph of the partial order on the distinct feature vectors.

the maximum likelihood estimate $\hat{F}_i(\mathbf{x})$ of the posterior-class cumulative distribution function $F_i(\mathbf{x})$, $i = 1, 2$ ($\hat{F}_3(\mathbf{x})$ is not included because it is equal to 1.)

Table 3.4 contains the MOCA and OSDL estimates of the cumulative distribution functions for the observed attribute vectors together with their median values. In the case of OSDL, we set $\alpha = 1/2$, while for MOCA the value of α is immaterial since the estimate shall always be equal to F^* at the attribute vectors comprising the training data set.

It should be noted that the occurrences of $(2, 1)$ in S, which have class labels 2 and 3, and the occurrence of data point $(3, 2)$, which has class label 1, induce monotonicity violations in $\hat{F}_1$ and in $\hat{F}_2$ respectively. MOCA resolves the violation in in $\hat{F}_1$ by replacing the original values $\hat{F}_1(2, 1)$ and $\hat{F}_1(3, 2)$ with their weighted average:

$$\tilde{F}_1^{\text{MOCA}}(2, 1) = \tilde{F}_1^{\text{MOCA}}(3, 2) = \frac{3 \times 0 + 1 \times 1}{3 + 1} = \frac{1}{4}.$$

The violation in $\hat{F}_2$ is resolved similarly.

	(x_1, x_2)	$n(\mathbf{x}, y)$			$n(\mathbf{x})$	$\hat{F}$	
		1	2	3		1	2
$\mathbf{x}_1$	$(1,1)$	2	0	0	2	1	1
$\mathbf{x}_2$	$(1,2)$	1	2	0	3	1/3	1
$\mathbf{x}_3$	$(2,1)$	0	2	1	3	0	2/3
$\mathbf{x}_4$	$(1,3)$	0	0	1	1	0	0
$\mathbf{x}_5$	$(3,2)$	1	0	0	1	1	1

Table 3.3: MOCA vs OSDL example. Summary of the labelled data set used. For each distinct feature vector $\mathbf{x}$ observed in the data set, the second column contains the coordinates of $\mathbf{x}$, the third column the number of occurrences of $\mathbf{x}$ observed with each of the three possible labels, the fourth column contains the overall number of occurrences of $\mathbf{x}$ observed in the data set, and the fifth column contains the values of the maximum likelihood estimate $\hat{F}$ of the posterior-class cumulative distribution function $F_i(\mathbf{x})$, $i = 1, 2$. (The estimates of $\hat{F}_3(\mathbf{x})$ are omitted because they are equal to 1.)

	$\tilde{F}^{\text{MOCA}}$		Median	$\tilde{F}^{\text{OSDL}}$		Median
	1	2		1	2	
$\mathbf{x}_1$	1	1	1	1	1	1
$\mathbf{x}_2$	1/3	1	2	2/3	1	1
$\mathbf{x}_3$	1/4	3/4	2	1/2	5/6	{1,2}
$\mathbf{x}_4$	0	0	3	0	0	3
$\mathbf{x}_5$	1/4	3/4	2	1/2	5/6	{1,2}

Table 3.4: MOCA vs OSDL example. MOCA estimates $\tilde{F}^{\text{MOCA}}$ and OSDL estimates $\tilde{F}^{\text{OSDL}}$ of the posterior-class cumulative distribution function $F_i(\mathbf{x})$, $i = 1, 2$ (the estimates of $\hat{F}_3(\mathbf{x})$ are omitted because they are equal to 1) together with their median values.

To illustrate OSDL, we now show how $\tilde{F}_1^{\text{OSDL}}(1,2)$ is computed. We have

$$\begin{aligned} F_1^{\min}(1,2) &= \min\{1/3, 1\} = 1/3 \\ F_1^{\max}(1,2) &= \max\{1/3, 0, 1\} = 1 \\ \tilde{F}_1^{\text{OSDL}}(1,2) &= \tfrac{1}{2} \times \tfrac{1}{3} + \tfrac{1}{2} \times 1 = \tfrac{2}{3} \end{aligned}$$

The L_1 error of MOCA on $S_{\mathcal{X}}$ is

$$L_1^{\text{MOCA}} = 0 + 1 + 1 + 0 + 1 = 3,$$

and it is the minimum possible L_1 error on the training data for a monotonic classifier.

For OSDL we have a choice of medians for the third and fifth observation. The lowest in-sample error is attained by assign both feature vectors label 2:

$$L_1^{\text{OSDL}} = 0 + 2 + 1 + 0 + 1 = 4.$$

3.5 Experiments

As we have shown in section 3.2.3, unlike OSDL, MOCA is guaranteed to minimise the empirical absolute risk. Although this is an important property, it remained to be seen how MOCA compares to OSDL on new, unlabelled data. Therefore, we performed a series of experiments on a number of data sets in order to compare MOCA to OSDL in terms of their mean absolute error. We used both artificial and real-world data sets, and the results for each of the two categories of data sets are discussed separately in sections 3.5.1 and 3.5.2 respectively.

In all of our experiments we assumed that, like MOCA, OSDL assigns $\mathbf{x}$ the label corresponding to the smallest median according to its estimates of the posterior class probabilities $P_i(\mathbf{x})$.

3.5.1 Artificial Data Sets

To compare the performance of MOCA and OSDL in controlled circumstances, we generated artificial data from the monotonic function

$$f_1(x_1, x_2) = 1 + x_1 + \tfrac{1}{2}(x_2^2 - x_1^2), \tag{3.5.1}$$

and from the non-monotonic function

$$f_2(x_1, x_2) = 3 + \sin\left(\frac{\pi}{2}x_1\right)(2 + \sin(2\pi x_2)), \tag{3.5.2}$$

drawing the features x_1 and x_2 independently from the uniform distribution on the unit interval. Non-monotonic function f_2 was considered in order to test the robustness of both algorithms against violations of the monotonicity assumption.

We sampled 100 points for training and another 10,000 for testing in order to obtain a reliable estimate of the mean absolute prediction error. To create ordered class labels, the values of the continuous target of the overall sample were discretised into four intervals by using equal-frequency binning.

For each data set, we selected the best α value from the set

$$\{0, 0.25, 0.5, 0.75, 1\}$$

for each of MOCA and OSDL by using 10-fold stratified cross validation. We then picked the best result obtained for each method in terms of mean absolute error (2.3.7) and compared them by performing a paired-sample $t-$test.

We added a normally distributed error term with mean zero and variance σ^2 and studied the behaviour of the algorithms for different levels of noise, namely for all $\sigma^2 \in \{\frac{j}{10}M\}_{j=0}^{10}$, where M is the maximum observed value of either function. As a result of adding noise, the samples of f_1 were used may have contained non-monotonic pairs. On the other hand, the samples of f_2 contained with certainty non-monotonic pairs even at zero-noise level because f_2 was non-monotonic function.

The results of our experiments are given in tables 3.5 and 3.6. It can be

observed that MOCA has consistently lower error, except of course for the monotonic data without noise: in that case, $\hat{F}$ already satisfies the stochastic order constraint, and, hence, MOCA and OSDL give identical results.

3.5.2 Real-World Data Sets

We performed tests on the following real-world data sets (see Appendix A for details):

- Australian Credit Approval
- Auto MPG
- Boston Housing
- Car Evaluation
- Computer Hardware
- ESL
- Haberman's Survival
- Pima Indians
- Windsor Housing

The numeric target attribute of the Auto MPG, Boston Housing, Computer Hardware, and Windor Housing data sets were discretised into four labels using equal-frequency binning.

For each data set, we selected the best α value from the set

$$\{0, 0.25, 0.5, 0.75, 1\}$$

for each of MOCA and OSDL using $10-$fold stratified cross validation. We then selected the best result obtained for each method in terms of the average L_1 error and compared them by performing a paired sample t-test, the results of which are given in Table 3.7. We note that MOCA has lower error in 7 out of 9 cases, and in 4 cases this result is significant at $\alpha = 0.05$. In two cases OSDL is better, and significantly so in the case of the Car Evaluation data set.

σ^2	L_1^{MOCA}	L_1^{OSDL}	p-value
0	0.3009	0.3009	–
$(1/10)M$	0.3359	0.3688	0
$(2/10)M$	0.5039	0.5441	0
$(3/10)M$	0.6472	0.7096	0
$(4/10)M$	0.7736	0.8641	0
$(5/10)M$	0.8453	0.9616	0
$(6/10)M$	0.9623	1.1389	0
$(7/10)M$	0.9297	1.2999	0
$(8/10)M$	0.9762	1.3428	0
$(9/10)M$	1.0263	1.3558	0
M	1.0195	1.4251	0

Table 3.5: MOCA vs OSDL. Experimental results for monotonic function f_1. The first column contains the different levels of noise $\sigma^2 \in \{\frac{j}{10}M\}_{j=0}^{10}$ added to f_1, where M is the maximum observed value of f_1, the second and third columns contain the errors registered by MOCA and OSDL respectively, and the fourth column contains the p-value of a paired-sample $t-$test.

σ^2	L_1^{MOCA}	L_1^{OSDL}	p-value
0	0.621	0.6549	$1.3962e-008$
$(1/10)M$	0.7297	0.8974	0
$(2/10)M$	0.811	0.9991	0
$(3/10)M$	0.8727	1.2763	0
$(4/10)M$	1.0004	1.3331	0
$(5/10)M$	1.0207	1.4066	0
$(6/10)M$	1.0964	1.4556	0
$(7/10)M$	1.0463	1.3733	0
$(8/10)M$	1.0245	1.4614	0
$(9/10)M$	1.0944	1.4259	0
M	1.0248	1.4583	0

Table 3.6: MOCA vs OSDL. Experimental results for non-monotonic function f_2. The first column contains the different levels of noise $\sigma^2 \in \{\frac{j}{10}M\}_{j=0}^{10}$ added to f_2, where M is the maximum observed value of f_2, the second and third columns contain the errors registered by MOCA and OSDL respectively, and the fourth column contains the p-value of a paired-sample $t-$test.

Data set	L_1^{MOCA}	L_1^{OSDL}	p-value
Australian Credit	**0.1609**	0.3362	0*
Auto MPG	**0.2526**	0.2551	0.6550
Boston Housing	**0.4565**	0.5020	0.1120
Car Evaluation	0.04051	**0.0318**	0.0010*
Computer Hardware	**0.3397**	0.3828	0.0390*
ESL	**0.3340**	0.3443	0.4930
Haberman's Survival	0.2614	**0.2582**	0.8350
Pima Indians	**0.2604**	0.2656	0.4330
Windsor Housing	**0.5385**	0.5934	0.0060*

Table 3.7: MOCA vs OSDL. Experimental results on real-world data sets. The second and third columns contain the errors registered by MOCA and OSDL respectively (lower error is shown in boldface), and the fourth column contains the p-value of a paired-sample $t-$test (the superscript $*$ indicates a significant difference at $\alpha = 0.05$).

3.6 Weighted kNN probability estimation

In many applications, the training data set S comprises only one observation of most attribute vectors $\mathbf{x} \in \mathcal{X}$, especially in case of numeric attributes; as a consequence, the maximum-likelihood estimates of the posterior class probabilities $P_i(\mathbf{x})$ tend to be overfitted. In order to prevent this from happening, we developed a weight-based estimator based on the nearest neighbours principle along the lines of the one introduced by Hechenbichler and Schliep in [60], where the estimates obtained are used to perform ordinal classification (without monotonicity constraints) by predicting the median.

In the following, we first discuss kNN as a classification technique and then illustrate how to use it to estimate probabilities.

3.6.1 kNN Classification

k Nearest Neighbor (kNN) classification is an example of *instance-based* or *lazy* pattern recognition: the training data set S is stored, and a new feature vector $\mathbf{x}_0$ is assigned a label depending the k most similar elements included in S.

Similarity is measured by means of a *distance metric* or *function*, namely a real-valued function d which verifies the following properties for any feature vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$:

1. $d(\mathbf{x}, \mathbf{y}) > 0$, $d(\mathbf{x}, \mathbf{x}) = 0$
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
3. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

The distance measure that we opted for is the *Euclidean distance*:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}.$$

In order to prevent attributes which have large values from having a stronger influence than attributes measured on a smaller scale, it is important to normalise the attribute values. We opted for *$Z-$score standardisation*: for each $\mathbf{x} \in S$, the $j-$th attribute x_j is replaced with

$$\frac{x_j - \bar{x}_j}{s_j},$$

where $\bar{x}_j$ and s_j denote respectively the mean and the sample standard deviation of values of the $j-$th attribute of the feature vectors present in the training sample S.

Having selected a distance measure to determine its neighbourhood, the allocation rule adopted in kNN classification is typically represented by (*unweighted*) *majority voting*: $\mathbf{x}_0$ is assigned the label $y_0 \in \mathcal{Y}$ occurring most frequently among its k nearest neighbours $\{\mathbf{x}_i\}_{i=1}^{k}$, namely

$$y_0 = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{k} 1_y(y_i),$$

where y_i is the label of the $i-$th neighbour $\mathbf{x}_i$, and 1_y is the *indicator function* for the set $\{y\}$, $\forall y \in \mathcal{Y}$, that is

$$1_y = \begin{cases} 1 & \text{if } y_i = y \\ 0 & \text{otherwise} \end{cases}$$

3.6.2 Weighted kNN Classification

In kNN classification it is reasonable to request that neighbours that are close to $\mathbf{x}_0$ have a greater role in determining its class than those that are distant. This assumption is at the basis of *weighted $k-NN$ classification*, in which a new observation $\mathbf{x}_0$ is assigned the class label y_0 which has a *weighted majority* among its k nearest neighbours, that is

$$y_0 = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{k} \omega_i 1_y(y_i).$$

Each of the k nearest neighbours $\mathbf{x}_i$ of $\mathbf{x}_0$ is assigned a weight ω_i which is inversely proportional to its distance $d = d(\mathbf{x}_0, \mathbf{x}_i)$ from $\mathbf{x}_0$ and which is obtained by means of a *weighting function* or *kernel* $K_\lambda(\mathbf{x}_0, \mathbf{x}_i)$ [59]:

$$\omega_i = K_\lambda(\mathbf{x}_0, \mathbf{x}_i) = G\left(\frac{d(\mathbf{x}_0, \mathbf{x}_i)}{\lambda}\right). \tag{3.6.1}$$

Kernels are at the basis of the *Parzen density estimation method* [59]. In that context, the *smoothing parameter* or *bandwidth* λ dictates the width of the window considered to perform the estimation; a large value of λ implies low variance (averages are computed over more observations) but also higher bias (the true density function is assumed to be constant within the window).

Function $G(\cdot)$ in (3.6.1) can be any function which has maximum in $d = d(\mathbf{x}, \mathbf{y}) = 0$ whose values gets smaller as the value of d grows. Thus the following properties must hold [60]:

1. $G(d) \geq 0$ for all $d \in \mathbb{R}$
2. $G(d)$ gets its maximum for $d = 0$
3. $G(d)$ descends monotonically for $d \to \infty$

In the one-dimensional case, a popular kernel is obtained by using the Gaussian density function $\phi(t)$ as $G(\cdot)$, with the standard deviation playing the role of the parameter λ. In the $p-$dimensional case, with $p > 1$, its

natural generalisation is represented by

$$K_\lambda(\mathbf{x}_0, \mathbf{x}_i) = \frac{1}{\sqrt{2\pi}\lambda} \exp\left\{-\frac{1}{2}\left(\frac{\|\mathbf{x}_0 - \mathbf{x}_i\|}{\lambda}\right)^2\right\},$$

and this is the kernel that we have adopted.

Although the kernel used is in a sense a parameter of wkNN, experience has shown that its choice (with the exception of the *rectangular kernel*, which gives equal weights to all neighbours) is not crucial [59].

In equation (3.6.1) it is assumed that the value of λ remains constant over the whole of the training data set. On the other hand, it would more appropriate if the value of λ were location-dependent, with larger values for regions where the training observations are sparse and smaller values for regions of the training data sample which are densely populated. This is the idea behind *adaptive windows*: given the *width function* $h_\lambda(\mathbf{x}_0)$ (indexed by λ), which determines the width of the neighbourhood at $\mathbf{x}_0$, we have

$$K_\lambda(\mathbf{x}_0, \mathbf{x}_i) = G\left(\frac{d(\mathbf{x}_0, \mathbf{x}_i)}{h_\lambda(\mathbf{x}_0)}\right).$$

In our implementation, we set $h_\lambda(\mathbf{x}_0)$ equal to the distance $d(\mathbf{x}_0, \mathbf{x}_{k+1})$ of $\mathbf{x}_0$ from the first neighbour $\mathbf{x}_{k+1}$ which is not taken into consideration [59, 60].

3.6.3 wkNN Probability Estimates

Definition 39 (wkNN Probability Estimator)**.** Given $\mathbf{x} \in S_\mathcal{X}$, the set N of the k nearest neighbours of $\mathbf{x}$ in $S_\mathcal{X}$, the set of the indices $N_k(\mathbf{x})$ in S of all observations of the feature vectors comprising N, the *wkNN estimator* $\hat{P}_i^{wk\mathrm{NN}}(\mathbf{x})$ of the posterior class probability $P_i(\mathbf{x})$, $i = 1, \ldots, k$, is defined as follows:

$$\hat{P}_i^{wk\mathrm{NN}}(\mathbf{x}) = \frac{\sum_{j \in N_k(\mathbf{x})} \omega_j 1_{y_i}(y_j)}{\sum_{j \in N_k(\mathbf{x})} \omega_j}. \tag{3.6.2}$$

It should be noted that the set of indices in (3.6.2) $N_k(\mathbf{x})$ may comprise more than k elements in case of repeated occurrences in S of some of the feature vectors comprising N. It should also be noted that $\mathbf{x}$ is included in its own neighbourhood and that, as a consequence, its occurrences have a

relatively large weight ω_i in (3.6.2). Finally, it should be noted that if $k = 1$, then the values returned by equation (3.6.2) coincide with the maximum likelihood estimates (3.2.5).

3.7 New Experiments on Real-World Data Sets

In order to determine whether and to what extent MOCA and OSDL, both in the "constant interpolation" and in the "balanced" and "double balanced" versions (see [77]), would benefit from estimating the posterior class probabilities they require using the estimator (3.6.2) instead of the maximum-likelihood estimator, we implemented the algorithms using the new estimator. In the case of MOCA, the adoption of the wkNN probability estimator has an impact on the computation of the MOCA estimator (3.2.7) not only in terms of the probability estimates on which the antitonic regression is performed but also on the weights used, for their cardinality is now equal to that of the indices $N_k(\mathbf{x})$ (see equation 3.6.2) for each $\mathbf{x} \in S_{\mathcal{X}}$.

We then performed new tests on the following real-world data sets (see Appendix A for details about them):

- Australian Credit
- Auto MPG
- Boston Housing
- Car Evaluation
- Computer Hardware
- ERA
- ESL
- Haberman's Survival
- KC1
- KC4
- LEV
- PC3
- PC4
- Pima Indians
- SWD
- Windsor Housing

The numeric target attribute of the Auto MPG, Boston Housing, Computer Hardware, and Windor Housing data sets were discretised into four labels using equal-frequency binning.

Each of the data sets was randomly divided into two parts: the first half, consisting of approximately two thirds of the data, was used as the training data set, and the second half was used as the validation data set.

The training set was used to determine the optimal values of the neighbourhood size k and of the nuisance parameter α for each of MOCA and OSDL through 10-fold stratified cross validation. Starting with $k = 1$, the value was incremented by one until the difference of the average L_1 error between two consecutive iterations was less than or equal to 1^{-6}. For each value of k considered, the optimal α in $\{0, 0.25, 0.5, 0.75, 1\}$ was determined. Once the optimal parameter values were determined for each algorithm, they were used to train them on the complete training set and then to test them on the validation data set.

The average L_1 error rate attained on the validation sets by the algorithms was registered, and then a paired $t-$test of the L_1 errors was performed in order to determine whether observed differences were statistically significant.

Table 3.8 lists the results of the experiments involving MOCA and "constant interpolation" OSDL implemented using the MLE and the wkNN estimator of the posterior class probabilities $P_i(\mathbf{x})$. The wkNN version of OSDL performed significantly better (at $\alpha = 0.05$) than the MLE version four times, whereas it performed significantly worse once (on the SWD data set); furthermore, the former almost never registered estimated error latter than the former. On the other hand, the wkNN version of MOCA was significantly better than the MLE version only once (on the ERA data set); all other observed differences were not statistically significant.

Table 3.9 shows the results of the investigation into the effect of on the "balanced" and "double balanced" versions of OSDL (for details on these other version of OSDL, see [77]). wkNN estimation of the posterior class probabilities $P_i(\mathbf{x})$ did not have much effect on both versions of the algorithm; the only one significant improvement was registered in the case of the KC1 data set.

Furthermore, comparing table 3.8 and table 3.9, we observe that con-

Data Set	MOCA wkNN	OSDL wkNN	MOCA MLE	OSDL MLE	1 vs 3	2 vs 4
Australian Credit	0.1304	0.1130	0.1348	0.3565	0.3184	0
Auto MPG	0.2977	0.2977	0.2977	0.2977	–	–
Boston Housing	0.5030	0.4675	0.4675	0.5207	0.4929	0.2085
Car Evaluation	0.0625	0.0556	0.0625	0.0556	–	–
Computer Hardware	0.3571	0.3286	0.3571	0.3571	–	0.5310
ERA	1.2006	1.2066	1.2814	1.2814	0.0247	0.0445
ESL	0.3620	0.3742	0.3620	0.4110	1	0.2018
Haberman's Survival	0.3529	0.3431	0.3529	0.3431	–	–
KC1	0.1863	0.1977	0.1863	0.3940	1	0
KC4	0.8095	0.8095	0.8571	0.8571	0.4208	0.5336
LEV	0.4162	0.4162	0.4102	0.4102	0.8060	0.8060
PC3	0.1228	0.1228	0.1228	0.1228	–	–
PC4	0.1872	0.1872	0.1872	0.1872	–	–
Pima Indians	0.3008	0.3086	0.3008	0.3086	–	–
SWD	0.5359	0.5479	0.5060	0.4940	0.1492	0.0092
Windsor Housing	0.5604	0.5220	0.5604	0.6044	–	0.0249

Table 3.8: Results of the experiments involving MOCA and "constant interpolation" OSDL implemented using the MLE and the wkNN estimator of the posterior class probabilities $P_i(\mathbf{x})$. The first four columns contain the average L_1 errors of both versions of both algorithms on the validation part of each data set. The penultimate column contains the $p-$value resulting from the comparison of both versions of MOCA, and the final column contains the $p-$value resulting from the comparison of both versions of OSDL.

stant interpolation OSDL with smoothing tends to outperform its balanced and double balanced counterparts.

3.8 Conclusions and Further Research

We have presented MOCA, a new non-parametric monotonic classification algorithm which attempts to minimise the mean absolute prediction error for classification problems with ordered class labels. We have shown that MOCA minimises the L_1 error on the training sample subject to monotonicity constraints. Through experiments on artificial and real-world data sets, we have shown that it compares favourably to OSDL, a similar classification algorithm also intended for the monotonic classification problems.

The maximum-likelihood (ML) estimates of the posterior class probabilities used by both algorithms are typically based on very few observations, so we conjectured that they both have a tendency to overfit the training sample. As we thought that a point of possible improvement would be

Data Set	BOSDL wkNN	BOSDL MLE	DBOSDL wkNN	DBOSDL MLE	1 vs 2	3 vs 4
Australian Credit	0.3565	0.3565	0.3565	0.3565	–	–
Auto MPG	0.2977	0.2977	0.2977	0.2977	–	–
Boston Housing	0.5207	0.5207	0.5207	0.5207	–	–
Car Evaluation	0.0556	0.0556	0.0556	0.0556	–	–
ERA	1.8533	1.9760	1.8533	1.9760	0.1065	0.1065
ESL	0.5092	0.5092	0.5092	0.5092	–	–
Haberman's Survival	0.3431	0.3431	0.3431	0.3431	–	–
KC1	0.1892	0.3030	0.3883	0.3940	0	0.2061
KC4	0.7619	0.8571	0.7857	0.8571	0.1031	0.1829
LEV	0.9251	0.9251	0.9251	0.9251	–	–
Computer Hardware	0.4143	0.4143	0.4143	0.4143	–	–
PC3	0.1228	0.1228	0.1228	0.1228	–	–
PC4	0.1872	0.1872	0.1872	0.1872	–	–
Pima Indians	0.2930	0.2930	0.2930	0.2930	–	–
SWD	0.6617	0.6557	0.6617	0.6437	0.8212	0.4863
Windsor Housing	0.6044	0.6044	0.6044	0.6044	–	–

Table 3.9: Results of the experiments involving the "balanced" (BOSDL) and the "double balanced" (DBOSDL) versions of OSDL implemented using the MLE and the wkNN estimator of the posterior class probabilities $P_i(\mathbf{x})$. The first four columns contain the average L_1 errors of both versions of both algorithms on the validation part of each data set. The penultimate column contains the $p-$value resulting from the comparison of both versions of BOSDL, and the final column contains the $p-$value resulting from the comparison of both versions of DBOSDL.

to consider an alternative to the ML estimates, we introduced a weighted k nearest neighbour (wkNN) approach to estimating the probabilities the algorithms use, and then run a second set of experiments to measure the improvements brought by using the wkNN estimates. Results showed that smoothing is beneficial for OSDL, for its predictive performance was significantly better on a number of data sets and almost never worse. The adoption of "balanced" and "double balanced" versions of OSDL do not seem to lead to an improvement over the constant interpolation version on the data sets considered. As for MOCA, smoothing seems to have a much more reduced effect. This is probably a consequence of the fact that the isotonic regression already smooths the basic estimates by averaging them in case of order reversals. Hence, MOCA is already rather competitive when using wkNN probability estimates for $k = 1$, that is when using ML estimates of the posterior class probabilities.

There are two interesting problems for further research which we men-

tion in the following. We have looked at the L_1 loss function, but this is certainly not the only loss function suitable for ordinal classification problems. Hence, it would be interesting to try to derive similar results for more general classes of loss functions. Another important issue is finding ways to reduce MOCA's execution time. One possibility might be represented by using the $O(n^2)$ approximate solution of the isotonic regression problem presented by Burdakov et al. in [20].

Chapter 4

Monotonic Instance Ranking with MIRA

Preference learning is an emerging subfield of machine learning concerning the learning of preference models from observed (or extracted) preference data; a *preference* can be considered as a relaxed constraint which, if necessary, can be violated to some degree [46]. The learned models are used to predict preferences of new, unseen individuals or groups or to predict preferences in new situations in which the learner has not been before. Preference learning is an interdisciplinary research field connecting machine learning with other areas in which preferences play a key role such as operations research, social sciences, and decision theory. More generally, that of "preferences" is an important topic in current artificial intelligence research. Automatic discovery of preferences of individuals is useful in various fields in which increasing personalisation has been taking place, such as recommender systems, adaptive user interfaces, adaptive retrieval systems, autonomous agents, and gaming.

Preference learning problems arise quite naturally in many application areas. A widely studied preference learning problem occurring in information retrieval is the ranking of the results returned by a search engine: given a query q and a set of Web pages P, the goal is to find a ranking of P reflecting their relevance with respect to q. The learned ranking is based on an unknown preference relation $\succ_q$ inferred from user feedback on past rankings of retrieval results for different queries [79]. Because users

are typically unwilling to provide explicit feedback by stating how relevant a retrieved page is, research has been carried out on how to collect user feedback indirectly. For instance, Joachims et. al propose several ways to gather user feedback through clicking behaviour [68, 69], and Arens [5] employs active learning (see chapter 5 of this thesis) to automatically select the most appropriate training examples to feed into the ranking algorithm he proposes in that same paper.

In many ranking problems, common sense dictates that the rank assigned to an instance should be increasing (or decreasing) in one or more of the attributes describing it. Let us consider, for example, the problem of ranking documents as *not relevant*, *somewhat relevant*, and *relevant* with respect to a particular query; in this case, typical attributes are counts of query terms in the abstract or title of the document, so one would expect an increasing relationship between these counts and document relevance.

In this chapter we present a new instance ranking algorithm named MIRA (*Monotonic Instance Ranking Algorithm*) which learns a monotonic ranking model from a set of labelled training examples. This algorithm builds on the ideas formulated by Fürnkranz et al. in [51] and on our own work on non-parametric monotonic classification (see chapter 3 of this thesis). Monotonicity is enforced in MIRA by applying the isotonic regression to the training sample, and an interpolation scheme is used to rank new data points. The basic ranking model is then combined with logistic regression in an attempt to remove unwanted rank equalities. Through experiments we showed that MIRA produces ranking functions having predictive performance comparable to that of a state-of-the-art instance ranking algorithm; this makes MIRA a valuable alternative when monotonicity is desired or mandatory.

This chapter is organised as follows:

- In section 4.1, we describe the three main ranking tasks which are addressed in the literature on preference learning by adopting the terminology proposed by Fürnkranz and Hüllermeier in [46]. In particular, in section 4.1.3 we introduce instance ranking, which is the specific ranking problem solved by MIRA.

- In section 4.2, we introduce a general approach to preference learning

based on binary decomposition techniques which have previously been used for multi-class and ordinal classification, and in section 4.2.1 we outline the aggregation scheme that MIRA builds on, which was proposed by Fürnkranz et al. [51].

- In section 4.3, the monotonic instance ranking problem is defined and an initial formulation of MIRA is presented.
- In section 4.4, we illustrate how the initial version of MIRA compares to the best-performing algorithm presented in [51] both on artificial and on real-world data in terms of ranking accuracy.
- In section 4.5, an improved and final version of our algorithm is introduced
- In section 4.6, the results of additional experiments to evaluate the performance of the second version of MIRA are presented.
- In section 4.7, conclusions are drawn and possibilities for future work are suggested.

4.1 Preference Learning Tasks

Given a set of training instances for which preferences are known, a *preference learning* task consists of learning a function which predicts preferences for a new set of items or for the same set of items but in a different context. Preferences typically consist of a total order (called in this case a *ranking*) $\succ$ on a finite set of alternatives

$$\mathcal{A} = \{a_1, \ldots, a_r\},$$

where, for all distinct $i, j \in \{1, 2, \ldots, r\}$,

$$a_i \succ a_j$$

indicates that alternative a_i is preferred to alternative a_j, or, equivalently, that a_i has *higher rank* than a_j.

It should be noted that a ranking $\succ$ of a the set of r alternatives $\mathcal{A}$ can be identified with a permutation π of $\mathcal{A}$ (or, equivalently, of the set $\{1, \ldots, r\}$) such that $\pi(i) < \pi(j)$ whenever $a_i \succ a_j$, where $\pi(i)$ is the position of an alternative a_i in the ranking. Therefore, the higher the degree of preference for an alternative, the smaller its index in the permutation. This equivalence shall play a fundamental role in the following.

Notation. We shall denote the set of all permutations of $\{1, \ldots, r\}$ as

$$P_r$$

for any non-negative integer r.

If instances are represented as feature vectors taking values in a $p-$dimensional feature space $\mathcal{X}$, a ranking problem can be cast as the pattern recognition problem of learning a function mapping a subset of $\mathcal{X}$ to a permutation $\pi \in P_r$ of the r possible alternatives $\mathcal{A}$ for the subset. The target function is called a *ranking function*.

In the following we describe the three main ranking tasks which are addressed as pattern recognition problems in the literature on preference learning, adopting the terminology proposed by Fürnkranz and Hüllermeier in [46]. Each task is different in terms of the type of training data it makes use of, which is not required to contain complete or consistent information. Moreover, each task differs in terms of type of alternatives that the learned ranking function returns a ranking of and in terms of the size of the subset of $\mathcal{X}$ that the ranking is learned for.

4.1.1 Label Ranking

Given a set of possible alternatives consisting of a set of labels $\mathcal{Y} = \{y_1, \ldots, y_k\}$ and a set of training instances $(\mathbf{x}, (y, y'))$, where $\mathbf{x}$ is a feature vector and the pair of labels (y, y') indicates that alternative y is preferred to alternative y' for $\mathbf{x}$, a *label ranking* task consists of learning a ranking function (*label ranker*)

$$r : \mathcal{X} \to P_k$$

mapping any new instance $\mathbf{x} \in \mathcal{X}$ to a permutation $\pi_{\mathbf{x}}$ of $\mathcal{Y}$.

Label ranking can be seen as a generalisation of classification in which a total ordering of the class labels $\mathcal{Y}$

$$y_{\pi_{\mathbf{x}}^{-1}(1)} \succ_{\mathbf{x}} y_{\pi_{\mathbf{x}}^{-1}(2)} \succ_{\mathbf{x}} \cdots \succ_{\mathbf{x}} y_{\pi_{\mathbf{x}}^{-1}(k)}$$

is associated with an instance $\mathbf{x}$ instead of only a single class label, where $\pi_{\mathbf{x}}^{-1}(i)$ is the position of label y_i in the ranking associated with $\mathbf{x}$.

One of the main application areas of label ranking is represented by the prediction of orderings of a fixed set of elements, such as predicting the ranking of the cars sold by a retailer for a potential customer based on their demographics and income, or ranking a set of genes according to their expression level (as measured by microarray analysis) based on their phylogenetic profile features [8]. Moreover, many conventional learning problems, such as classification and multilabel classification, may be formulated in terms of label preferences [49].

To measure the predictive performance of a label ranker, a loss function is required, and its choice depends on the purpose the predicted ranking serves. The following two settings are the most covered in the literature due to their numerous practical applications [65].

In the first setting, which is the most frequent and probably most obvious label ranking scenario, what is relevant is the *complete ranking*, namely the positions assigned to all of the labels. The quality of a predicted ranking is assessed by using a distance measure quantifying the deviation of the predicted ranking from the true one, which is called *ranking error*. Any distance or correlation measure on rankings (e.g. Spearman's rank correlation coefficient or Kendall tau correlation coefficient) can be used for this purpose.

The second setting corresponds to the situation in which what is assumed is the existence of a single target label and not of a target ranking; the objective in this case is to predict a label ranking where the target item appears as high as possible. Labels in this scenario correspond for example to the causes of a technical failure, and the predicted ranking represents a search process testing the possible fault until the real one is found. In this setting, the quality of a predicted ranking is the number of labels which precede the target item in it. This type of error, namely the distance in the

predicted ranking of the target label from the top-rank, is called *position error*.

4.1.2 Object Ranking

Given a set of pairwise preferences of the form $\mathbf{x} \succ \mathbf{x}'$ indicating that object $\mathbf{x}$ should be ranked higher than object $\mathbf{x}'$, an *object ranking* task consists of learning a ranking function (called an *object ranker*)

$$r : 2^{\mathcal{X}} \to P_n$$

mapping any non-empty, finite set of objects $X \subseteq \mathcal{X}$ of cardinality n to a permutation π of X. This is typically accomplished by assigning a real-valued *score* to each object $\mathbf{x}$ and then ranking the objects according to the scores: the larger the score, the higher the rank. Therefore, what is actually learned is a *scoring function*

$$s : 2^{\mathcal{X}} \to \mathbb{R}$$

assigning a score to the elements of a subset $X \subseteq \mathcal{X}$.

This preference learning task, which is one of the first to have been studied, is also known as *learning to order things* [27]. An example is represented by the problem of learning to rank the query results returned by a search engine tackled by Joachims in [68]. The training information is provided implicitly by the user who clicks on some of the links in the query result and not on others; this information can be turned into binary preferences by assuming that the selected pages are preferred over nearby pages that are not clicked on [69].

Similarly to a label ranker, one way to measure the performance of an object ranker consists of measuring the ranking error, namely comparing the predicted ranking with the target ranking. Frequently, though, the number of items to be ordered in an object ranking task is likely to be much larger than in a label ranking task, so one is typically not interested in predicting a full ranking but only, for instance, the top-k ranks; this occurs, for instance, in the task of ranking Web search engine results. In this case, top-k measures comparing the top positions of rankings are used

instead.

Finally, in some applications one is not interested in predicting a ranking but in distinguishing between positive and negative examples or between relevant and irrelevant items (such as, which documents are relevant to a query). In this case, performance is measured by adopting measures typical of information retrieval such as precision, recall, or normalised discounted cumulative gain (NDCG) [67].

4.1.3 Instance Ranking

Given a set of possible alternatives consisting of a set of totally ordered labels $\mathcal{Y} = \{y_1, \ldots, y_k\}$ (or, without loss of generality, $\mathcal{Y} = \{1, 2, \ldots, k\}$) representing preference degrees and a collection of labelled training instances $\{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{N}$, an *instance ranking* task consists of learning a ranking function (called an *instance ranker*)

$$r : 2^{\mathcal{X}} \to P_n$$

mapping a non-empty, finite set of unlabelled individuals $X \subseteq \mathcal{X}$ of cardinality n to a permutation π of X such that, ideally, the instances with preference degree y_k precede those with preference degree y_{k-1}, which in turn precede those with preference degree y_{k-2}, etc. As in object ranking, this is typically accomplished by learning a *scoring function*

$$s : 2^{\mathcal{X}} \to \mathbb{R},$$

which is used to assign a real-valued *score* to each element of X to then rank them according to their scores, with higher ranks corresponding to larger scores.

Instance ranking is therefore similar to object ranking in terms of the type of ranking returned but proceeds from the setting of ordinal classification in terms of the training data used (see section 3.2). Moreover, the ranking predicted by an instance ranker is in a way a refinement of the order information provided by an ordinal classifier because, unlike the latter, it distinguishes between objects within the same category.

As an example, consider a bank possessing a data set describing past

loan applicants in terms of their demographic characteristics and categorised as *highly reliable*, *reliable*, and *unreliable*. The goal for the bank is ranking new loan applicants according to their risk level from the most likely reliable to the most likely unreliable and, therefore, not just partitioning the applicants into the three categories above.

For $k = 2$, this learning problem corresponds to the well-known *bipartite ranking* problem. It consists of learning a ranking function from a training set of positively and negatively labelled examples which, when applied to a set of unlabelled instances, totally orders them from most likely positive to most likely negative. For $k > 2$ this learning problem has been studied by Fürnkranz et al. in [51] and by Rajaram et al. in [93] with the name of *multi-partite* ranking and *k-partite ranking* respectively. Similarly to bipartite ranking, the goal is not to learn a good classifier but a good ranker, that is, a function which systematically ranks "high" classes ahead of "low" classes.

Different types of accuracy measures have been proposed for instance ranking. As the goal is to produce a ranking in which instances from higher classes precede those from lower classes, these measures count the *number of ranking errors*, namely the number of pairs $(\mathbf{x}, \mathbf{x}') \in X \times X$ such that $\mathbf{x}$ is ranked higher than $\mathbf{x}'$ even though the former belongs to a lower class than the latter.

In the two-class case, this corresponds to the area under the ROC curve (AUC)

$$AUC\,(s, X) = \frac{1}{|P||N|} \sum_{\mathbf{x} \in P} \sum_{\mathbf{x}' \in N} S(s(\mathbf{x}), s(\mathbf{x}')), \tag{4.1.1}$$

where $P \subset X$ and $N \subset X$ are the positive and negative instances in X, and $S(\cdot, \cdot)$ is defined as follows:

$$S(a, b) = \begin{cases} 0 & \text{if } a > b \\ 1 & \text{if } a < b \\ 1/2 & \text{if } a = b \end{cases} \tag{4.1.2}$$

Mapping (4.1.2) outputs 1 when a positive instance is ranked before the negative one and 0 in the reverse case; the value returned is $\frac{1}{2}$ when the instances are assigned the same score.

The AUC can be interpreted as the probability that a randomly chosen instance belonging to the positive class is ranked higher than a randomly chosen instance belonging to the negative class. Its generalisation to more than two classes is represented by the probability that a randomly chosen pair of observations $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ belonging to different classes is ranked consistently by the ranking function, namely

$$P(s\left(\mathbf{x}\right) < s(\mathbf{x}') \mid y < y'\right). \tag{4.1.3}$$

If the training examples are independently and identically drawn from a probability distribution on $\mathcal{X} \times \mathcal{Y}$, an unbiased estimator of (4.1.3) is represented by the *concordance index* (*C-index*) [54]

$$C\left(s, X\right) = \frac{1}{\sum_{i<j} n_i n_j} \sum_{i<j} \sum_{(\mathbf{x},\mathbf{x}') \in X_i \times X_j} S(s(\mathbf{x}), s(\mathbf{x}')), \tag{4.1.4}$$

where X_i and X_j are the subsets of instances $\mathbf{x} \in X$ having class y_i and y_j respectively, $n_i = |X_i|$, and $S(\cdot, \cdot)$ is the mapping defined in (4.1.2).

4.2 Learning Preference Relations

Preferences can be expressed in several ways, two of which are very natural and have been investigated in the literature on choice and decision theory; they form the basis for the two main approaches to tackling the preference learning tasks illustrated in the previous section.

The first approach to modelling preferences consists of evaluating individual alternatives by means of a *utility function*, which assigns an abstract *utility degree* to each alternative under consideration. This approach translates into the problem of *learning utility functions* given a training data set representing preferences. Learning utility functions can be challenging because utility degrees tend to be difficult to elicit; for example, it is difficult to ensure a consistent scale even if all utility evaluations are performed by the same user, and the situation becomes even more problematic if utility scores are elicited from different users, which may not have a uniform scale of scores [27].

A second main approach to modeling preferences consists of comparing

pairs of alternatives in terms of *preference relations*, that is learning one or more binary preference predicates comparing pairs of alternatives. This second approach is more appealing: instead of choosing one alternative from a set of options or ranking all alternatives according to their desirability directly, it is often easier to first compare the possible alternatives in a pairwise fashion and then choose the best alternative or rank all possible alternatives in a second step on the basis of these pairwise comparisons. This intuition is supported by cognitive psychology research [103] and forms the basis for many decision theory methodologies. Crucially, this approach is also attractive from a machine learning point of view as it allows a preference learning task to be carried out as a two-step procedure: first a binary comparison relation is learned, and then the pairwise comparisons the learned relation returns are combined to make predictions. The training phase is much simpler compared to that of learning utility functions because comparative preference information is used directly instead of having to translate it into a set of constraints on a latent utility function to be learned and because it translates into a set of learning problems which can typically be solved in a more accurate and efficient way. Moreover, different prediction problems may be performed using the results of the training phase simply by choosing a different aggregation method.

A third approach to learning ranking functions is based on making specific assumptions about the structure of the preference relations, e.g. that the target ranking of a set of objects can be represented as a lexicographic order.

Finally, another possibility is to use local estimation techniques such as the nearest neighbour principle [50].

4.2.1 Instance Ranking by Learning Preference Relations

Initial methods to solve the instance ranking problem were based on the idea of learning the optimal scoring function s by turning the initial training data set into a set of *order constraints* on it and then learning a function in agreement as much as possible with them. Each pair of observed examples $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ such that $y_i > y_j$ gives rise to a constraint

$$s(\mathbf{x}_i) > s(\mathbf{x}_j). \tag{4.2.1}$$

In order to use existing learning algorithms, one common approach consisted of expressing these constraints as classification examples and then learning a binary probabilistic classifier whose probability estimates are then used as a scoring function. One important example of this approach is represented by the algorithm introduced by Herbrich et al. in [63].

One drawback to these methods is that the number of constraints, and, therefore, the size of the training data derived for training the binary classifier is usually larger than the original training data set. Moreover, these methods are not specifically designed for instance ranking, although they are applicable to it; their input is a set of order constraints (4.2.1) which, in this case, are derived from inequalities between pairs of labels.

Fürnkranz et al. introduced in [51] two pairwise learning techniques specifically designed for instance ranking which make better and more explicit use of the class label information contained in the training data. Instead of transforming the original problem into a single binary classification problem, the authors suggested decomposing it into several binary problems by applying well-established decomposition techniques which had already been used successfully in multi-class classification. Several binary classifiers, therefore several scoring functions, are learned from the training instances. Given a set of individuals X to rank, both approaches submit each $\mathbf{x} \in X$ to all learned models and aggregate the corresponding predictions into an overall score. The set X is then ranked based on these aggregate scores. One key advantage of this decomposition technique is that the resulting learning problems are simpler and usually much smaller than a single binary problem, for they avoid the combinatorial explosion caused by considering all pairs of the original training examples. Roughly speaking, by exploiting class information, a large number of order constraints can be satisfied in an implicit way: a classifier which separates n_0 negative from n_1 positive instances, and which is hence trained on $n_0 + n_1$ examples, automatically satisfies $n_0 n_1$ order constraints.

Fürnkranz et al. also motivate the use of binary decomposition methods to tackle instance ranking by their successful application to similar problems, such as classification [47,48], ordinal classification [45], and label ranking [66]. Moreover, as Fürnkranz et al. point out, an ordinal classifier might indeed be used as a scoring function by interpreting the fact that an

instance $\mathbf{x}$ is assigned label y_i as $\mathbf{x}$ being given score i; the resulting scoring function, though, is not very good as it produces a large number of ties. On the other hand, a ranking function s can be turned into an ordinal classifier by using *thresholding*: given $m+1$ threshold values t_i, $i = 1, \ldots, k+1$, class y_i is predicted for an instance $\mathbf{x}$ if the score $s(\mathbf{x})$ is between t_i and t_{i+1}.

Fürnkranz et al. show that their decomposition methods are competitive, if not superior, to a state-of-the-art ranking method introduced by Joachims in [68] in terms of predictive accuracy while having much lower computational cost. Of the algorithms they introduce, we now describe the algorithms which builds on the decomposition technique aimed at ordinal classification presented by Frank and Hall in [45], for our own algorithm is based on the same aggregation scheme.

The ordinal classification algorithm presented by Frank and Hall in [45] solves a problem involving $k > 2$ classes by training $k-1$ binary classifiers. The model s_i learned by the i-th classifier, for $i = 1, \ldots, k-1$, separates the meta-classes $C_- = \{1, \ldots, i\}$ and $C_+ = \{i+1, \ldots, k\}$. Given a new instance $\mathbf{x}$, a prediction $s_i(\mathbf{x})$ is interpreted as an estimate of the probability

$$P(y > i \,|\, \mathbf{x}),$$

that is the probability that the class y of $\mathbf{x}$ is in C_+. An estimate of the posterior class probabilities $P(y|\mathbf{x})$ are then derived from the above cumulative probabilities as follows:

$$P_i(\mathbf{x}) = \begin{cases} 1 - P(y > i \,|\, \mathbf{x}), & \text{if } i = 1 \\ \max\{(P(y > i-1 \,|\, \mathbf{x}) - P(y > i \,|\, \mathbf{x})), 0\}, & \text{if } i = 2, \ldots, k-1 \\ P(y > i-1 \,|\, \mathbf{x}), & \text{if } i = k \end{cases}$$

(It should be noted that in Frank and Hall's aggregation scheme $P_i(\mathbf{x})$ for $i = 2, \ldots, k-1$ is set equal to 0 if $P(y > i-1 \,|\, \mathbf{x}) \leq P(y > i \,|\, \mathbf{x})$ because the collection of binary classifiers it is based on does not preclude that from happening.) The class predicted is the class corresponding to the highest posterior probability.

To obtain a ranking function, Fürnkranz, Hüllermeier and Vanderlooy

suggest aggregating the models s_i as

$$s_{\text{agg}}(\mathbf{x}) = \sum_{i=1}^{k-1} s_i(\mathbf{x}) = \sum_{i=1}^{k-1} P(y > i \mid \mathbf{x}) \tag{4.2.2}$$

because (4.2.2) systematically assigns higher scores to instances from higher classes [51].

Notation. When the s_i in this aggregation scheme are logistic regression models [59] as is the case in [18, 51], we shall denote this ranking function as s_{fhv} and the resulting ranking model as FHV.

It should be noted that logistic regression models produce linear class boundaries.

4.3 Monotonic Instance Ranking

Assuming that the feature space $\mathcal{X}$ is partially-ordered, that the set of preference degrees $\mathcal{Y}$ is totally ordered, and that there exists an unknown monotonic functional relationship between $\mathcal{X}$ and $\mathcal{Y}$, a *monotonic instance ranking* task is an instance ranking task in which the learned scoring function $s : \mathcal{X} \to \mathbb{R}$ is monotonic, that is

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow s(\mathbf{x}) \leq s(\mathbf{x}'), \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \tag{4.3.1}$$

In other words, a lower ordered input is not allowed to have a higher score.

It should be noted that if we have that

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow P(y > i \mid \mathbf{x}) \leq P(y > i \mid \mathbf{x}') \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \forall i = 1, \ldots, k,$$

which, as shown in section 3.2, is equivalent to

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow F_i(\mathbf{x}) \geq F_i(\mathbf{x}') \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \forall i = 1, \ldots, k, \tag{4.3.2}$$

where $F_i(\mathbf{x})$ denotes the cumulative probability $P(y \leq i \mid \mathbf{x})$, then scoring function (4.2.2) satisfies condition (4.3.1), namely

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow s_{\text{agg}}(\mathbf{x}) \leq s_{\text{agg}}(\mathbf{x}'),$$

since $P(y > i \mid \mathbf{x}) = 1 - F_i(\mathbf{x})$. Therefore, to obtain a monotonic scoring function using aggregation scheme (4.2.2), it is sufficient that stochastic order constraint (4.3.2) holds.

In order to achieve this goal, we propose adopting the MOCA estimator $\tilde{F}_i$ (see equation 3.2.7), for it satisfies the stochastic monotonicity constraint (4.3.2) by construction (see section 3.2.1).

Definition 40 (MIRA Scoring Function)**.**

$$s_{\text{mira}}(\mathbf{x}) = \sum_{i=1}^{k-1} 1 - \tilde{F}_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathcal{X}. \tag{4.3.3}$$

The MIRA scoring function is guaranteed to produce globally-monotonic estimates, as the following proposition shows.

Proposition 15. *The* MIRA *scoring function is monotonic.*

Proof. Given two feature vectors $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ such that $\mathbf{x} \leq \mathbf{x}'$, it follows from theorem 5 on page 65 that

$$1 \geq \tilde{F}_i(\mathbf{x}) \geq \tilde{F}_i(\mathbf{x}') \qquad \forall i = 1, \ldots, k-1,$$

and then

$$0 \leq 1 - \tilde{F}_i(\mathbf{x}) \leq 1 - \tilde{F}_i(\mathbf{x}') \qquad \forall i = 1, \ldots, k-1,$$

from which the proposition follows immediately. □

4.3.1 Example

To illustrate the differences between them, we apply the original scoring function (4.2.2) and the MIRA scoring function (4.3.3) to a monotonic labelled data set S, interpreting the labels as degrees of preference. We assume the former to employ maximum-likelihood estimates of the cumulative probabilities and denote it as $\hat{f}$. The examples comprising S consist of feature vectors taking values in the 2-dimensional feature space

$$\mathcal{X} = \{1, 2, 3\} \times \{1, 2, 3\}$$

and having one of the class labels in the totally-ordered set

$$\mathcal{Y} = \{1, 2, 3\}.$$

Figure 4.1 depicts the partial order on $S_{\mathcal{X}}$.

Table 4.1 lists, for every $\mathbf{x} \in S_{\mathcal{X}}$, the number of occurrences $n(\mathbf{x}, y)$ of $\mathbf{x}$ having label y, the total number of occurrences $n(\mathbf{x})$, the maximum likelihood estimate $\hat{F}_i(\mathbf{x})$ of the posterior-class cumulative distribution function $F_i(\mathbf{x})$, $i = 1, 2$, and, finally, the antitonic regression $\tilde{F}$ of $F_i(\mathbf{x})$. $\hat{F}_3(\mathbf{x})$ and $\tilde{F}_3(\mathbf{x})$ are not included because they are equal to 1; moreover, it should be recalled that $\tilde{F} = F^*$ for the elements of the training data set.

The maximum-likelihood estimates $\hat{F}$ violate the stochastic order constraint because $(2, 1) \leq (3, 2)$, but $\hat{F}_1(2, 1) < \hat{F}_1(3, 2)$; moreover, $(1, 2) \leq (3, 2)$, but $\hat{F}_1(1, 2) < \hat{F}_1(3, 2)$. The antitonic regression resolves these violations by assigning the weighted average of $\hat{F}_1(2, 1)$ and $\hat{F}_1(3, 2)$ to both attribute vectors:

$$\tilde{F}_1(2, 1) = \tilde{F}_1(3, 2) = \frac{3 \times 0 + 2 \times 1/2}{3 + 2} = \frac{1}{5}.$$

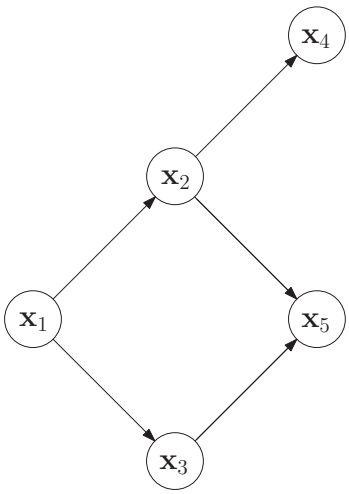

Figure 4.1: MIRA example. Graph of the partial order on the distinct feature vectors observed in the training data set.

	(x_1, x_2)	$n(\mathbf{x}, y)$			$n(\mathbf{x})$	$\hat{F}$		$\tilde{F}$	
		1	2	3		1	2	1	2
$\mathbf{x}_1$	$(1, 1)$	2	0	0	2	1	1	1	1
$\mathbf{x}_2$	$(1, 2)$	1	2	0	3	$1/3$	1	$1/3$	1
$\mathbf{x}_3$	$(2, 1)$	0	2	1	3	0	$2/3$	$1/5$	$2/3$
$\mathbf{x}_4$	$(1, 3)$	0	0	1	1	0	0	0	0
$\mathbf{x}_5$	$(3, 2)$	1	0	1	2	$1/2$	$1/2$	$1/5$	$1/2$

Table 4.1: MIRA example. Summary of the labelled data set used. For each distinct feature vector $\mathbf{x}$ observed in the data set, the second column contains the coordinates of $\mathbf{x}$, the third column the number of occurrences of $\mathbf{x}$ observed with each of the three possible labels, the fourth column contains the overall number of occurrences of $\mathbf{x}$ observed in the data set, the fifth column contains the values of the maximum likelihood estimate $\hat{F}$ of the posterior-class cumulative distribution function $F_i(\mathbf{x})$, $i = 1, 2$, and the sixth column contains the antitonic regression $\tilde{F}$ of $F_i(\mathbf{x})$. (The estimates of $\hat{F}_3(\mathbf{x})$ and of $\tilde{F}_3(\mathbf{x})$ are omitted because they are equal to 1.)

The conflict between $(1, 2)$ and $(3, 2)$ is resolved by this averaging as well.

In table 4.2, we see the effect of the estimator choice on score values. Since scoring function $\hat{s}$ is based on $\hat{F}$, that is, since we filled $\hat{F}$ into equation (4.3.3) instead of $\tilde{F}$, we have

$$\hat{s}(2, 1) > \hat{s}(3, 2),$$

which violates the monotonicity constraint. This violation is resolved in s_{mira} as $s_{\text{mira}}(2, 1) < s_{\text{mira}}(3, 2)$.

(x_1, x_2)	$\hat{s}$	s_{mira}
$(1, 1)$	0	0
$(1, 2)$	0.67	0.67
$(2, 1)$	1.33	1.13
$(1, 3)$	2	2
$(3, 2)$	1	1.3

Table 4.2: MIRA example. Scores based on $\hat{F}$ and $\tilde{F}$ respectively.

To compute $\tilde{F}$ for feature vector $\mathbf{x}_0 = (2, 2)$, we proceed as follows. As the points in the training data smaller than $\mathbf{x}_0$ are $(1, 1)$, $(1, 2)$ and $(2, 1)$, we have that

$$F_2^{\max}(\mathbf{x}_0) = \min\{F_2^*(1,1), F_2^*(1,2), F_2^*(2,1)\} = \min\{1, 1, 2/3\} = 2/3.$$

The only feature vector in the training data bigger than $\mathbf{x}_0$ is $(3, 2)$, so we have that

$$F_2^{\min}(\mathbf{x}_0) = \max\{F_2^*(3,2)\} = \max\{1/2\} = 1/2.$$

So for $\alpha = \frac{1}{2}$ we would have

$$\tilde{F}_2(\mathbf{x}_0) = \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + \left(\frac{1}{2}\right)\left(\frac{2}{3}\right) = \frac{7}{12}.$$

$\tilde{F}_1(\mathbf{x}_0)$ is computed in a similar fashion.

4.4 Experiments

In order to verify that the desired monotonicity property does not come at the cost of a decrease in predictive performance, we performed an empirical comparison of the performance of scoring function s_{mira} with that of s_{fhv} in terms of the concordance index (see equation 4.1.4). The comparison involved both artificial and real data.

4.4.1 Artificial Data

To show that MIRA can improve the performance of the FHV logistic regression model, we performed tests on data sets generated from a monotonic non-linear model. The class boundaries of this model are depicted in figure 4.2. For example, given a feature vector (x_1, x_2) if $x_1 > 0.4$ and $x_2 > 0.4$, then the vector is assigned to class 3. The two features x_1 and x_2 were drawn independently from the uniform distribution on the interval [0,1].

We considered two different training set sizes and four different noise levels. For example, a noise level of 0.1 indicates that an observation from

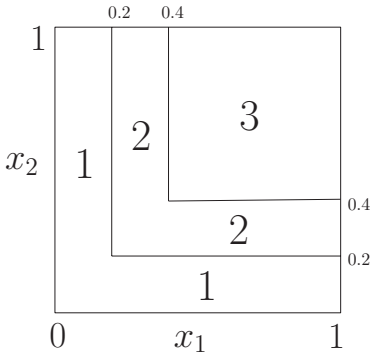

Figure 4.2: MIRA experiments on artificial data. Class boundaries according to which the data was generated. For example, given a feature vector $\mathbf{x} = (x_1, x_2)$, if $x_1 > 0.4$ and $x_2 > 0.4$, then the vector is assigned to class 3.

class 1 is flipped to class 2 with probability 0.05, and is flipped to class 3 with probability 0.05 as well. The concordance index of trained models was estimated on a test set of size 10,000.

We performed five-fold stratified cross validation to select the best value of $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

Table 4.3 gives the concordance index, averaged over five repetitions, of MIRA and FHV for two different training set sizes and four different noise levels; a noise level of ε indicates that the probability that the true label is flipped to one of the other two labels is equal to ε. The results conform to our expectations: since the class boundaries are monotonic but non-linear, MIRA has some advantage over the linear FHV model (with the odd exception of $n = 100$ and $\varepsilon = 0.2$). MIRA's advantage appears to become smaller as the noise level increases; this makes sense, for a non-parametric technique such as MIRA is more sensitive to noise.

The results also conform to the expectation that bigger training sets tend to give better results for both methods.

4.4.2 Real Data

We also performed tests on the following real-world data sets (see Appendix A for more details on them):

- Auto MPG

Size n	100		500	
Noise ε	FHV	MIRA	FHV	MIRA
0.0	0.9013	0.9856	0.9041	0.9942
0.1	0.8379	0.8957	0.8416	0.8906
0.2	0.7802	0.7431	0.7842	0.8256
0.3	0.7148	0.7484	0.7271	0.7578

Table 4.3: MIRA experiments on artificial data. Concordance index of MIRA and FHV for two different training set sizes and four different noise levels. A noise level of ε indicates that the probability that the true label is flipped to one of the other two labels is equal to ε.

- Boston Housing
- Computer Hardware
- ERA
- ESL
- Haberman's Survival
- LEV
- Ohsumed
- Pima Indians
- SWD
- Windsor Housing

The numeric target attributes of the Auto MPG, Boston Housing, Computer Hardware, and Windor Housing data sets were discretised into five labels using equal-frequency binning. Moreover, we did not use the whole of the Ohsumed data but instead only the data for query 3 and only considered the descriptive attributes 5, 6, 7, 18, 20, 21, 22, 35, 36, and 37.

For each data set, the concordance index of both algorithms was estimated as the average of the concordance indices registered during five repetitions of a 10−fold stratified cross validation. Within each fold, the best value –in terms of the concordance index– for MIRA's α in $\{0, 0.25, 0.5, 0.75, 1\}$ was chosen by performing 5−fold stratified cross validation on the training data set for the fold. The best value of α was then used to train MIRA and complete the fold.

Table 4.4 contains the concordance index attained for the best value of α by MIRA and by FHV respectively averaged over the five repetitions of a 10−fold stratified cross validation. We observe that the difference in concordance index is typically very small. Hence, we conclude that the desired property of monotonicity can be obtained without sacrificing much in terms of predictive accuracy. Even though the difference mostly occurs only in the second or third decimal place, we observe that FHV has the higher concordance index 10 out of 11 times. When performing Wilcoxon's signed-ranks test, as recommended by Demšar [31], the p−value obtained is 0.02.

We may summarise these findings by stating that FHV has a consistently slightly better predictive performance.

4.5 An Improved Monotonic Ranking Function

Scoring function (4.3.3) has the potential disadvantage that violations of monotonicity are resolved by averaging the conflicting (partial) scores, possibly leading to setting the scores equal altogether.

To illustrate this point, let us consider the simplest possible case of

Data set	MIRA	FHV
Auto MPG	0.9327	0.9494
Boston Housing	0.8773	0.9179
Computer Hardware	0.9064	0.9085
ERA	0.7326	0.7384
ESL	0.9572	0.9623
Haberman's Survival	0.7024	0.6815
LEV	0.8617	0.8656
Ohsumed	0.6346	0.6680
Pima Indians	0.7697	0.8298
SWD	0.8070	0.8144
Windsor Housing	0.8529	0.8601

Table 4.4: MIRA experiments on real data. Concordance index attained for the best value of α by MIRA and by FHV respectively averaged over five repetitions of a 10−fold stratified cross validation.

two examples $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ in a binary problem where $\mathbf{x}_1 \leq \mathbf{x}_2$ but $y_1 > y_2$. The isotonic regression resolves this monotonicity violation by averaging the probabilities, giving both examples a probability of 0.5 for the lower class; as a consequence, the two examples get the same score, giving a concordance index on the training data of 0.5. However, the monotonicity constraint expresses the belief that the higher ordered input $\mathbf{x}_2$ tends to have higher class labels in general. Therefore, it would be more appropriate to give $\mathbf{x}_2$ a higher score than $\mathbf{x}_1$. It should be noted that doing so would actually yield a lower concordance index on the training data as it would decrease to 0. If the monotonicity constraint is correct, however, then the concordance index on unseen data improves by making the inequality a strict one.

In an attempt to enforce strict inequalities, we try to break ties by combining each of the probabilities added up in equation (4.3.3) with those estimated using the logistic regression model [59]

$$\log \frac{P_i(C_+|\mathbf{x})}{P_i(C_-|\mathbf{x})} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{4.5.1}$$

where $P_i(C_+|\mathbf{x})$ and $P_i(C_-|\mathbf{x})$ are the posterior class probabilities that a given instance belongs to the metaclasses C_+ and C_- for the $i-$th initial rank respectively.

Definition 41 (MIRA$^+$ Scoring Function)**.**

$$s_{\text{MIRA}^+}(\mathbf{x}) = \sum_{i=1}^{k-1} \gamma P_i^{\text{LR}}(\mathbf{x}) + (1-\gamma)(1-\tilde{F}_i(\mathbf{x})) \quad \forall \mathbf{x} \in \mathcal{X}, \tag{4.5.2}$$

where each $P_i^{\text{LR}}(\mathbf{x}) = P_i(C_+|\mathbf{x})$ is the fitted probability of the logistic regression model for the $i-$th metaclass C_+, and $\gamma \in [0, 1]$.

The reason for using correction (4.5.1) is that if $\beta_j > 0$, then there is a strictly positive monotonic relationship between x_j and the probability of belonging to the metaclass C_+; consequently, this produces a ranking function which is strictly monotonic in x_j.

In order to only get positive coefficients in the logistic regression model, the full logistic models (namely, including all descriptive attributes) are

first estimated. If the models obtained contain negative coefficients, then all of the corresponding descriptive attributes are removed, and the logistic models are re-estimated with the remaining descriptive attributes. The re-estimation of the logistic models is repeated until only attributes with positive coefficients remain.

It should be noted that because attributes with negative coefficients are removed, we may not be able to always enforce the desired strict monotonicity. To illustrate this point, let us denote with $\mathbf{x}_\ell$ the subset of $\mathbf{x}$ included in the logistic regression model. If $\mathbf{x} \leq \mathbf{x}'$ but $\mathbf{x}_\ell = \mathbf{x}'_\ell$, then the logistic regression model shall obviously assign the same probabilities to $\mathbf{x}$ and $\mathbf{x}'$, that is $P_i(C_+|\mathbf{x}) = P_i(C_+|\mathbf{x}')$.

Finally, it should also be noted that if $\gamma = 0$ in (4.5.2), then the scoring function being used effectively is (4.3.3), and that if $\gamma = 1$, then only the logistic regression correction (4.5.1) is being used. Moreover, even if $\gamma = 1$, then s_{mira} is not identical to s_{fhv} because MIRA enforces positive coefficients in order to obtain the required monotonicity directions.

4.6 Additional Experiments

We performed new experiments to evaluate the performance of the MIRA$^+$ scoring function using the same experimental set-up used in the experiments reported in section 4.4.2. Table 4.5 reports the concordance index attained for the best value of α by MIRA$^+$ for different values of γ and by FHV respectively averaged over five repetitions of a 10−fold stratified cross validation.

To test the null-hypothesis that all methods have the same performance, we carried out a Friedman's test as recommended by Demšar [31]. The test yielded a p-value of 0.0003, leading to rejection of this hypothesis at any reasonable value of α, so we proceeded with a Nemenyi post-hoc test in which we compared MIRA$^+$ at different levels of γ with FHV, using the latter as the baseline or control ranker in this experiment. The p-values of the null-hypothesis of no difference are given in the final row of table 4.5 for each value of γ. To control for family-wise error, we performed the Bonferroni correction. Taking $\alpha = 0.1$, we divided this by the number of

comparisons performed, which was eight; hence, only p-value smaller than 0.0125 are considered to indicate a significant difference.

Summarising the results, it seems fair to say that striking a balance between the isotonic regression and the logistic regression components works best for MIRA, for the best results are obtained for $\gamma = 0.5$ and $\gamma = 0.7$. This can be inferred from the high p-values and low average ranks for these values of γ. Finally, the fact that results for $\gamma = 0$ and $\gamma = 1$ are among the worst proves that both components of MIRA's scoring function must be present in order to achieve the best performance.

4.7 Conclusion and further research

We presented MIRA, a monotonic instance ranking algorithm. MIRA extends our earlier work on non-parametric monotonic classification [10] to ranking problems and builds on the best-performing decomposition and aggregation scheme among those proposed by Fürnkranz et al. in [51].

By performing experiments on real data, we showed that MIRA's predictive accuracy measured by means of the concordance index is comparable to that of the algorithm by Fürnkranz et al. Moreover, experiments performed on an artificial data set show that MIRA can outperform the linear model the FHV algorithm is based on if class boundaries are monotonic and non-linear. More importantly, MIRA is guaranteed to produce a monotonic ranking func-

Data set	$\gamma = 0$	$\gamma = 0.05$	$\gamma = 0.1$	$\gamma = 0.2$	$\gamma = 0.5$	$\gamma = 0.7$	$\gamma = 0.9$	$\gamma = 1$	FHV
Auto MPG	0.9327	0.9389	0.9398	0.9427	0.9488	0.9504	0.9491	0.9479	0.9494
Boston Housing	0.8773	0.8817	0.8844	0.8887	0.9000	0.9020	0.9006	0.8951	0.9179
Computer Hardware	0.9064	0.9134	0.9140	0.9145	0.9194	0.9187	0.9141	0.9097	0.9085
ERA	0.7326	0.7333	0.7337	0.7348	0.7369	0.7379	0.7365	0.7356	0.7384
ESL	0.9572	0.9579	0.9583	0.9579	0.9469	0.9374	0.9366	0.8601	0.9623
Haberman's Survival	0.7024	0.7003	0.7000	0.6992	0.6951	0.6949	0.6915	0.6841	0.6815
LEV	0.8617	0.8626	0.8634	0.8647	0.8669	0.8674	0.8664	0.8656	0.8656
Ohsumed	0.6346	0.6434	0.6444	0.6478	0.6583	0.6660	0.6688	0.6700	0.6680
Pima Indians	0.7697	0.7924	0.7949	0.7989	0.8126	0.8205	0.8279	0.8288	0.8298
SWD	0.8070	0.8082	0.8091	0.8105	0.8129	0.8081	0.8008	0.7949	0.8144
Windsor Housing	0.8529	0.8570	0.8578	0.8594	0.8609	0.8616	0.8592	0.8582	0.8601
Average rank	7.7	6.6	5.7	4.9	3.5	3.3	4.5	5.6	3.1
$p-$value	0.0000	0.0005	0.0066	0.0491	0.3476	0.4481	0.0887	0.0094	–

Table 4.5: MIRA$^+$ experiments on real data. Concordance index attained for the best value of α by MIRA$^+$ for different values of γ and by FHV respectively averaged over five repetitions of a 10−fold stratified cross validation.

tion. Monotonicity is desired or even required for many applications, so this represents a valuable addition to existing ranking algorithms.

One issue for future research is determining whether MIRA's predictive accuracy can be further improved by using a different estimator capable of attaining higher concordance index values. We tried to do so by using the weighted kNN estimator introduced by Barile and Feelders in [11] and also described in this thesis without obtaining any improvements. Another way to improve the performance of the algorithm to investigate is the adoption of different aggregation schemes.

Chapter 5

Active Learning in Monotonic Classification

In order to learn accurate classification models from data, traditional supervised approaches typically require a large collection of labelled training vectors. There are many applications, however, in which data are abundant but class labels are difficult, time-consuming, or expensive to obtain. In such cases, we would like to be able to select the examples (the *query points*) whose labels are in some sense most informative about the location of the decision boundary between classes. This problem setting has been studied in the relatively new field of *active learning* [97]. At each step, an active learning algorithm can ask an *oracle* (e.g., a human annotator, a domain expert, or a device which is expensive to query such as a satellite) to label an example, which may be drawn from a set of unlabelled examples or synthesised ad hoc.

As part of our project, we investigated how to exploit prior knowledge in the form of monotonicity constraints for active learning in monotonic classification (see section 3.2). We focused on the pool-based active learning setting, in which there is an initial pool of unlabelled data points and the learner is allowed ask an *oracle* for the label of (a limited number of) data points. In this setting, the central problem is how to choose the data points to query the oracle for; ideally, the learner should submit queries whose answers allow it to draw conclusions about the labels of as many other unlabelled points as possible.

Many active learning approaches to classification use a specific type of classifier (e.g. support vector machines, classification trees, or logistic regression models) relative to which good query points are determined. In our case, instead, we have decoupled active learning from a specific classification model. Our approach is based on the observation that if the class label of an object a is given, then it is possible to infer that all objects which score worse than a on all attributes cannot have a higher class label and that, on the other hand, all those that score better than a on all attributes cannot have a lower class label. Therefore, thanks to the monotonicity constraints, we hypothesised that we might be able to infer the labels of a substantial subset of the training vectors from just a few well-chosen query points.

The most important research question was how we could use the known monotonicity of the "true" class labels to determine good query points. Is it possible to find optimal strategies to select query points? Are these strategies computationally feasible? If not, can we develop easy-to-compute heuristics for selecting good query points? As a result of our investigation, we devised heuristics which, next to a theoretical analysis, were implemented in two new active learning algorithms. We determined the quality of our approach empirically by performing tests on both artificial and real-life data sets; in each test we used the training data set returned by each algorithm to fit a classification model and then estimated the error rate of each learned model on a test sample. We do not enforce monotonicity constraints on the trained models because we are only interested in using monotonicity to infer the labels of data points as a way of augmenting the training sample. The fitted models are merely instrumental in determining the relative quality of the training samples produced using the proposed heuristics.

This chapter is organised as follows:

- In section 5.1 we introduce the active learning paradigm, providing some theoretical arguments to support it in section 5.1.1.
- In section 5.2 we discuss the basic ideas of active learning with monotonicity constraints, then we propose two heuristics to select good query points and two algorithms, each operating in a different setting, which are designed to make use of the introduced heuristic.

- In section 5.3, we discuss related work.
- In section 5.4 we provide and interpret the results of experiments performed on artificial as well as real data sets.
- Finally, in section 5.5, we draw conclusions and indicate possibilities for further research.

5.1 Active Learning

In order to learn accurate classification models from data, traditional supervised approaches typically require a large collection of labelled training vectors. There are many applications, however, in which data are abundant but class labels are difficult, time-consuming, or expensive to obtain. For instance, learning to classify documents (e.g., articles or Web pages) or any other kind of media (e.g., image, audio, and video files) requires that users label hundreds or thousands of instances with labels such as "relevant" or "not relevant". Instance labelling by manual annotation is typically unlikely to be carried out on the required scale because of its tediousness; moreover, it can lead to redundancy or contradiction. In such cases, we would like to be able to select the examples (the *query points*) whose labels are in some sense most informative about the location of the decision boundary between the classes.

This problem setting has been studied in the relatively new field of *active learning* (also called *query learning* or *optimal experimental design* in statistical literature) [97]. It is an iterative type of pattern recognition based on the idea that, if the learning algorithm is allowed to choose the data from which it learns, then it shall perform better with fewer training examples. At each step, an active learning algorithm can ask an *oracle* (e.g., a human annotator, a domain expert, or a device which is expensive to query such as a satellite) to label an informative example which may be drawn from a set of unlabelled examples or synthesised ad hoc.

Several scenarios have been studied in the literature on active learning [97]. In our investigation, we focused our attention on the *pool-based sampling* setting because of its relevance to data mining applications. In this scenario (see figure 5.1), which was first introduced in the context of

text mining by Lewis et al. in [75], it is assumed that a small set of labelled feature vectors $\mathcal{L}$ and a large set (or *pool*) of unlabelled feature vectors $\mathcal{U}$ are available. The active learning algorithm begins with $\mathcal{L}$ as the initial training data set and then selects the most informative instance from $\mathcal{U}$ and queries the oracle for its label; this choice is typically made according to a greedy querying strategy based on an informativeness measure which is used to evaluate all feature vectors in the pool $\mathcal{U}$ or, if $\mathcal{U}$ is very large, a sample thereof. The newly labelled example is added to $\mathcal{L}$, which the algorithm then uses in a standard supervised way. The process is repeated until the maximum number of queries allowed has been performed.

Active learning research mainly focuses on the strategy to choose the data points to be queried. The most common querying strategy is *uncertainty sampling*, where the unlabelled instance chosen to be queried is the instance that the learner is the least certain how to label. Other strategies try to reduce the subset of all hypotheses that are consistent with the observed training tuples. Finally, a decision-theoretic approach may follow which estimates expected error reduction, and the selected tuples are the ones that would result in the greatest reduction in the total number of incorrect predictions such as by reducing the expected entropy over $\mathcal{U}$.

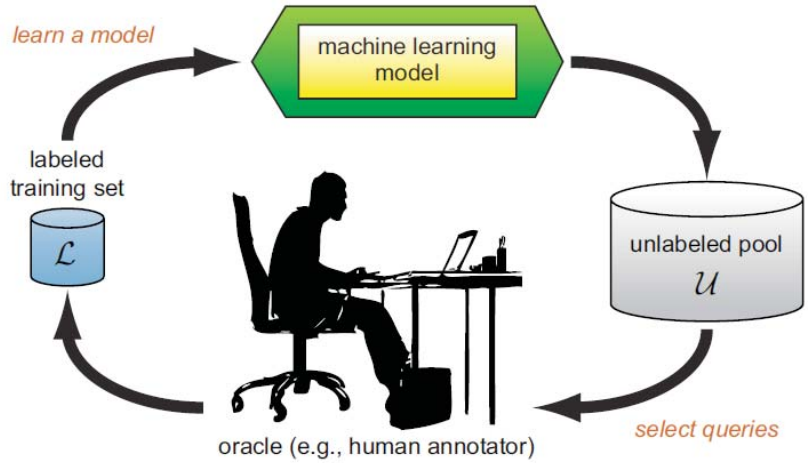

Figure 5.1: The pool-based active learning cycle [97].

5.1.1 Theoretical Justification

A theoretical argument for why and when active learning can produce better results than traditional supervised pattern recognition is an active research topic. The goal is to find, given a learning problem, a bound on the number of queries required to learn an accurate enough model and a guarantee that this number is smaller than in the traditional supervised setting.

To illustrate the kind of argument sought, let us consider the problem of learning a binary threshold function f parametrised by an unknown constant θ for points lying on a one-dimensional line:

$$f(x, \theta) = \begin{cases} 1 & \text{if } x > \theta, \\ 0 & \text{otherwise} \end{cases}$$

According to the probably approximately correct (PAC) learning framework, given a desired maximum error rate ϵ, if the underlying data distribution can be classified without error by an hypothesis from the class of hypotheses we consider (that is, the realisable case), then it is enough to draw randomly $O(1/\epsilon)$ labelled instances.

On the other hand, if this problem is solved in a pool-based active learning setting, the instances labelled by querying the oracle can be seen as an array sorted based on their labels, namely as a sequence of zeros followed by ones. The goal of the active learning algorithm can therefore be seen as discovering the level at which the transition from zero to one occurs while asking for as few labels as possible. A classification error smaller than ϵ can therefore be attained only requiring $O(\log 1/\epsilon)$ queries by conducting a binary search through the unlabelled instances.

Generalisations to higher dimensions of the example above within the active learning literature have focused on linear separators (see for example Balcan et al. [9]) rather than general monotonic classifiers as we did in our research.

5.2 Active Learning and Monotonicity

Many active-learning approaches to classification make use of a specific type of classifier (e.g. support vector machines, classification trees, or logistic re-

gression models) relative to which good query points are determined. For example, one possibility is to learn a logistic regression model on the set of examples labelled at a certain point and then apply this model to the unlabelled examples to determine the one about which the classifier is most uncertain (as measured, for example, by the entropy of the probability distribution over the classes). This would determine the next query point, and so on.

In our case, instead, we decoupled active learning from a specific classification model, for we investigated how prior knowledge about the application domain in the form of monotonicity constraints can be exploited for active learning.

The basis of our approach is the observation that if the class label of an object a is given, then it is possible to infer that all objects that score worse than a on all attributes cannot have a higher class label and that, on the other hand, all those that score better than a on all attributes cannot have a lower class label. Therefore, thanks to the monotonicity constraints, we might be able to infer the labels of a substantial subset of the training vectors from just a few well-chosen query points.

To show how monotonicity constraints can be exploited in active learning, let us consider the problem of ranking loan applicants based on their disposable income and on the number of years they have worked with their current employer. In this example, a monotonicity relation between the descriptive attributes and the target is expected to hold. In particular, it is reasonable to assume that the higher an applicant's disposable income, the lower the risk that they shall not be able to pay back the loan; likewise, experience shows that, caeteris paribus, the more years an applicant has been working for his or her current employer, the smaller their probability of defaulting on a loan. Let us then consider the collection of unlabelled examples given in table 5.1. If we know that applicant a_2 has been accepted by the loan officer, since a_3 and a_4 score not worse than a_2 on both decision criteria, it can be inferred that the loan officer shall also accept these other applicants. Hence, there's no point in querying the loan officer about them. On the other hand, learning the loan officer's opinion on a_2 does not allow to infer anything about a_1 because the two points are incomparable: a_2 scores better on income, but a_1 scores better on years of employment.

Applicant	Disposable Income	Years Employed
a_1	3000	15
a_2	3500	8
a_3	5000	10
a_4	4000	8
a_5	3000	6

Table 5.1: Monotonicity constraints exploitation in active learning example. Unlabelled loan applicant data.

In the following, we introduce two querying strategies which make explicit use of the monotonic constraints assumed to be present in the data. We then present two new active learning algorithms for monotonic classification named MAL (Monotonic Active Learner) and ND-MAL (Non-Deterministic Monotonic Active Learner) designed to make use of the introduced heuristics. Each algorithm operates in a different setting.

In order to be able to deal with the possibility of repeated occurrences of the same feature vector with the same label, the pool of unlabelled vectors $\mathcal{U}$ is assumed to be a multiset. Moreover, both the heuristics and the algorithms have knowledge of the number of occurrences of each distinct feature vector occurring in $\mathcal{U}$ and of the partial ordering on them.

Notation. We shall indicate the set of distinct feature vectors in $\mathcal{U}$ as $\mathcal{U}_{\mathcal{X}}$.

5.2.1 Querying with Monotonicity Constraints

Notation. Given a feature vector $\mathbf{x} \in \mathcal{U}_{\mathcal{X}}$, we shall denote the size of $\downarrow\mathbf{x}$ as $d(\mathbf{x})$ and the size of $\uparrow\mathbf{x}$ as $u(\mathbf{x})$.

Notation. Given a feature vector $\mathbf{x} \in \mathcal{U}_{\mathcal{X}}$, we shall denote the number of lower sets of $\mathcal{U}_{\mathcal{X}}$ which include $\mathbf{x}$ as $L(\mathbf{x})$ and the number of upper sets of $\mathcal{U}_{\mathcal{X}}$ which include $\mathbf{x}$ as $U(\mathbf{x})$.

In order to develop some intuition, let us begin by studying the special case of a chain

$$\mathbf{x}_1 \leq \mathbf{x}_2 \ldots \leq \mathbf{x}_n$$

comprising n feature vectors, each having a binary class label $y \in \{1, 2\}$. In general, if the oracle states that the label of the $i-$th element of the chain is $y_i = 1$, then it is possible to infer that $y_j = 1$ for all $j < i$; similarly, from knowing that $y_i = 2$, it can be inferred that $y_j = 2$ for all $j > i$.

Let us suppose that we query the oracle for the label of the smallest element of the chain $\mathbf{x}_1$. If the result of the query is that $y_1 = 2$, then it can be inferred that the labels of all other elements of the chain is also 2; on the other hand, if y_1 turns out to be 1, then no inference can be made about the labels of the other elements of the chain. It seems unlikely that the label of $\mathbf{x}_1$ may be 2 because there is only one monotonic classification such that $y_1 = 2$; hence, the odds that this case occurs seem very small.

Let us make the ideas expressed above more precise. First of all, it should be noted that the number of monotonic binary classifications of a chain with n points is $n + 1$, since label 1 can be assigned to any initial segment (including the empty one and the complete chain) and label 2 to the remaining upper segment. Moreover, the number of monotonic classifications in which a point $\mathbf{x}$ is assigned label 1 is equal to the size of its upset $\uparrow\mathbf{x}$, for assigning label 1 to the downset of each element of $\uparrow\mathbf{x}$ yields one such monotonic classification. Likewise, the number of monotonic classifications which assign label 2 to a point $\mathbf{x}$ is equal to the size of its downset $\downarrow\mathbf{x}$.

If we assume that the latent monotonic decision function being inferred has been drawn at random from the set of all possible monotonic functions on the chain, then

$$P(y_i = 1) = \frac{u(\mathbf{x}_i)}{d(\mathbf{x}_i) + u(\mathbf{x}_i)}, \qquad P(y_i = 2) = \frac{d(\mathbf{x}_i)}{d(\mathbf{x}_i) + u(\mathbf{x}_i)}.$$

If the label of the $i-$th element of the chain $\mathbf{x}_i$ is 1, then it can be inferred that the label of all the points comprising its downset also have label 1; therefore, the number of labels which can be inferred is equal to $d(\mathbf{x}_i)$. This makes it possible to compute the expected number of labels

$N(\mathbf{x}_i)$ that can be inferred by querying $\mathbf{x}_i$ as

$$\begin{aligned}\mathbb{E}[N(\mathbf{x}_i)] &= P(y_i = 1)d(\mathbf{x}_i) + P(y_i = 2)u(\mathbf{x}_i) \\ &= \frac{u(\mathbf{x}_i)d(\mathbf{x}_i)}{d(\mathbf{x}_i) + u(\mathbf{x}_i)} + \frac{d(\mathbf{x}_i)u(\mathbf{x}_i)}{d(\mathbf{x}_i) + u(\mathbf{x}_i)}\end{aligned} \tag{5.2.1}$$

Maximising it with respect to $u(\mathbf{x}_i)$ and $d(\mathbf{x}_i)$, we find that (5.2.1) is largest when $d(\mathbf{x}_i) = u(\mathbf{x}_i)$. As a consequence, the most informative feature vector to query is the one in the middle of the chain.

The reader must have noted the similarity with the problem of searching for an element in a sorted list: using the well-known binary search strategy, all labels can be inferred by asking the oracle for the value of $\log_2(n) + 1$ of them.

The generalisation to arbitrary partial orders is conceptually straightforward. In order to obtain it, it should be noted that there is a one-to-one correspondence between lower sets and monotonic binary classifications: assigning label 1 to all elements of the lower set and label 2 to its complement upper set yields a monotonic classification, and, for any monotonic classification, the set of points assigned label 1 is a lower set. Hence, the number of monotonic classifications in which $\mathbf{x}_i$ is assigned label 1 (respectively label 2) is equal to the number of lower sets (respectively upper sets) which include $\mathbf{x}_i$. Therefore

$$P(y_i = 1) = \frac{L(\mathbf{x}_i)}{L(\mathbf{x}_i) + U(\mathbf{x}_i)}, \quad P(y_i = 2) = \frac{U(\mathbf{x}_i)}{L(\mathbf{x}_i) + U(\mathbf{x}_i)}. \tag{5.2.2}$$

It follows that filling probabilities (5.2.2) into equation (5.2.1) would make it possible to determine the feature vector having the highest expected number $\mathbb{E}[N(\mathbf{x}_i)]$ of inferred labels.

Unfortunately, as proved by Provan and Ball in [92], counting the number of lower sets of a partial order is a #P-complete problem. A problem is *#P-complete* if and only if it is in #P, and all problems in #P are reducible to it by a polynomial-time counting reduction. This class is harder than NP-complete, and a polynomial-time solution to a #P-complete problem would imply P=NP. Therefore, the choice of the point to query can only be made heuristically. For the binary case, a reasonable heuristic in terms of

computational cost consists of maximising the number of values which can be inferred in the worst case, that is selecting the query point $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{U}_\mathcal{X}} \min\{d(\mathbf{x}), u(\mathbf{x})\}. \tag{5.2.3}$$

It should be noted that this heuristic selects the middle element of a chain.

In the non-binary case, selecting a feature vector to query becomes slightly more complex and requires some extra notation.

Notation. Given a feature vector $\mathbf{x} \in \mathcal{U}_\mathcal{X}$, we shall denote the interval of possible labels for an unlabelled instance $\mathbf{x}$ as $[\ell_\mathbf{x}, h_\mathbf{x}]$.

Assuming we start with no labelled examples, initially it is

$$[\ell_\mathbf{x}, h_\mathbf{x}] = [1, k] \quad \forall \mathbf{x} \in \mathcal{U}_\mathcal{X}.$$

As the oracle is queried, the bounds of the label intervals are adjusted based on the monotonicity constraint in the following way: if the oracle returns label y for query point $\mathbf{x}$, then it can be inferred that

1. for all $\mathbf{x}' \in{\downarrow}\mathbf{x} : h_{\mathbf{x}'} \leftarrow \min(h_{\mathbf{x}'}, y)$;
2. for all $\mathbf{x}'' \in{\uparrow}\mathbf{x} : \ell_{\mathbf{x}''} \leftarrow \max(\ell_{\mathbf{x}''}, y)$.

Let $N(\mathbf{x}, y)$ denote the overall number of candidate labels which can be scrapped for the feature vectors comprising $\mathcal{U}$ after learning from the oracle that the label of $\mathbf{x}$ is y. It clearly is

$$N(\mathbf{x}, y) = \sum_{\mathbf{x}'\in{\downarrow}\mathbf{x}} (h_{\mathbf{x}'} - y)_+ + \sum_{\mathbf{x}''\in{\uparrow}\mathbf{x}} (y - \ell_{\mathbf{x}''})_+,$$

where $z_+ = \max(0, z)$. By maximising the worst case, we obtain the following heuristic.

Definition 42 (Heuristic 1)**.** Given a pool of unlabelled observations, query the oracle for the label of

$$x^* = \arg\max_{\mathbf{x}\in\mathcal{U}} \min_{y\in[\ell_\mathbf{x}, h_\mathbf{x}]} \{N(\mathbf{x}, y)\}. \tag{5.2.4}$$

To illustrate this heuristic, let us consider an unlabelled data set S comprising observations of the four, two-dimensional feature vectors described in table 5.2. The feature space is ordered according to the product order induced by the total order on the domain of each attribute (see equation 2.4.7), which are both assumed to be the set of integers. The ordered set of possible labels is $\mathcal{Y} = \{1, 2, 3\}$

The distinct feature vectors comprising the data set are plotted in figure 5.2; by looking at the plot, it is clear that the data set contains an incomparable pair, namely $(\mathbf{x}_2, \mathbf{x}_3)$ (see definition 5).

Figure 5.3 depicts the graph of the partial order on $S_{\mathcal{X}}$; since the number of possible labels is 3, each point is initially labelled with the interval $[1, 3]$. In table 5.3 we indicate, for each feature vector, the overall number of candidate labels which can be eliminated after observing each of the three possible labels. Both $\mathbf{x}_2$ and $\mathbf{x}_3$ satisfy equation (5.2.4), so either can be chosen. Let us suppose that $\mathbf{x}_2$ is selected and that the label that the oracle returns for it is $y_2 = 2$.

Figure 5.4 shows the new intervals after processing the first query. Ta-

	x^1	x^2
$\mathbf{x}_1$	2	2
$\mathbf{x}_2$	4	6
$\mathbf{x}_3$	8	3
$\mathbf{x}_4$	12	10

Table 5.2: Heuristic 1 example. Distinct feature vectors.

	$y = 1$	$y = 2$	$y = 3$	min
$\mathbf{x}_1$	2	5	8	2
$\mathbf{x}_2$	4	4	4	4
$\mathbf{x}_3$	4	4	4	4
$\mathbf{x}_4$	8	5	2	2

Table 5.3: Heuristic 1 example. Overall number of candidate labels which can be eliminated for every feature vector after observing each of the possible labels.

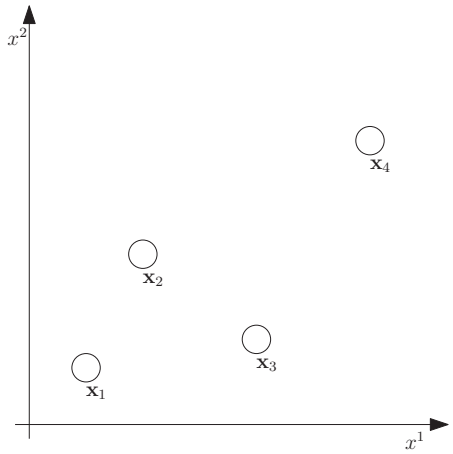

Figure 5.2: Heuristic 1 example. Plot of the distinct feature vectors.

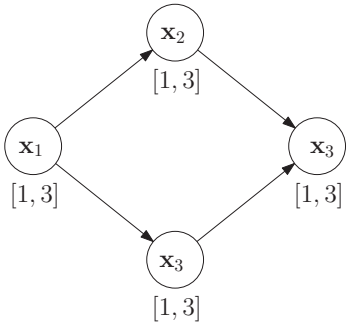

Figure 5.3: Heuristic 1 example. Order graph for the distinct feature vectors labelled with the interval of possible labels.

ble 5.4 contains the updated information concerning the number of candidate labels which can be eliminated after observing each of the possible labels; based on the information contained in the table, the second point selected to query the oracle for is $\mathbf{x}_3$. If the oracle returns label $y = 1$ for $\mathbf{x}_3$, then it is possible to infer that $y = 1$ and only the label of $\mathbf{x}_4$ remains uncertain.

Another possible criterion consists of fitting a classification model to examples which have been labelled at the current stage, either by querying the oracle or by making inferences based on mononiticity, and then using

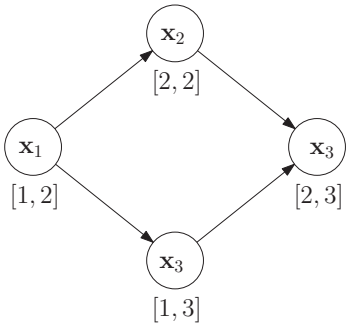

Figure 5.4: Heuristic 1 example. Order graph for the distinct feature vectors labelled with the interval of possible labels after observing $y_2 = 2$.

	$y = 1$	$y = 2$	$y = 3$	min
$\mathbf{x}_1$	1	2	-	1
$\mathbf{x}_2$	-	0	-	0
$\mathbf{x}_3$	3	2	3	2
$\mathbf{x}_4$	-	2	1	1

Table 5.4: Heuristic 1 example. Overall number of candidate labels which can be eliminated for every feature vector after observing each of the three possible labels after observing $y_2 = 2$.

the learned model to estimate the posterior class probabilities $P(y|\mathbf{x}_i)$ for the remaining unlabelled data points. The selected query point shall be the one which maximises the expected number of label inferences based on these probability estimates.

Definition 43 (Heuristic 2)**.** Given a pool of unlabelled observations, query the oracle for the label of

$$x^* = \arg\max_{\mathbf{x} \in \mathcal{U}} \sum_{y \in [\ell_{\mathbf{x}}, h_{\mathbf{x}}]} \hat{P}(y|\mathbf{x}) N(\mathbf{x}, y). \tag{5.2.5}$$

5.2.2 MAL

The MAL algorithm operates in the ideal situation in which the oracle answers a query for the label a feature vector always returning the same label

and in which the oracle is guaranteed to return labels so that monotonicity is satisfied; in other words, when queried for the labels of two feature vectors $\mathbf{x}$ and $\mathbf{x}'$, the oracle assigns them the label y and y' respectively so that

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow y \leq y'.$$

It follows that, in case of repeated occurrences of the same feature vector $\mathbf{x}$ in $\mathcal{U}$, the oracle needs only be queried once for it.

Our active learning algorithm for this setting is described in Algorithm 4: given the multiset $\mathcal{U}$ of unlabelled feature vectors, MAL returns a multiset $\mathcal{L}$ of training examples labelled by querying the oracle at most max times for the labels of points chosen according to a heuristic H. In line 3, the best query point according to H is selected from $\mathcal{U}_\mathcal{X}$. The label of the selected point is then returned by the oracle, and the new labelled example is added to the training set. In lines 7-11, the intervals of possible labels for the feature vectors comprising $\downarrow\mathbf{x}^*$ and $\uparrow\mathbf{x}^*$ are updated based on the answer provided by the oracle. Then, if the label of a point $\mathbf{x} \in \mathcal{U}_\mathcal{X}$ is uniquely determined, i.e. $\ell_\mathbf{x} = h_\mathbf{x} = y$, then $n(\mathbf{x})$ examples $(\mathbf{x}, y)$ are added to $\mathcal{L}$, and $\mathbf{x}$ is removed from $\mathcal{U}_\mathcal{X}$. The algorithm iterates until a maximum number of iterations max has been performed or $\mathcal{U}_\mathcal{X}$ is empty.

5.2.3 ND-MAL

The ND-MAL algorithm operates in the situation where the monotonic relation between the descriptive attributes and the class label is assumed to be of a probabilistic nature: for a given feature vector $\mathbf{x}$, the class label y is distributed according to a probability distribution $P(y|\mathbf{x})$, and, when queried for the label of $\mathbf{x}$, the oracle replies by producing a random draw from $P(y|\mathbf{x})$.

This second setting contemplates the possibility that the oracle may make errors or, more generally, that the class label is not uniquely determined by the chosen descriptive attributes. For example, heavy smokers have indeed a higher probability of developing lung cancer than non-smokers; nonetheless, not every smoker develops lung cancer, and some non-smokers do develop lung cancer.

Algorithm 4 $\mathcal{L} = \mathrm{MAL}(\mathcal{U}, max)$

$\mathcal{L} \leftarrow \varnothing$
while $max > 0 \wedge \mathcal{U}_{\mathcal{X}} \neq \varnothing$ **do**
  $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x} \in \mathcal{U}_{\mathcal{X}}} H(\mathbf{x})$
  $y^* \leftarrow \mathcal{O}(\mathbf{x}^*)$
  $\mathcal{L}_D \leftarrow \mathcal{L}_D \cup \{(\mathbf{x}^*, y^*)\}$
  $G_{\mathcal{L}} \leftarrow G_{\mathcal{L}} \cup \{((\mathbf{x}^*, y^*), n(\mathbf{x}^*))\}$
  $\mathcal{U}_{\mathcal{X}} \leftarrow \mathcal{U}_{\mathcal{X}} \setminus \{\mathbf{x}^*\}$
  for all $\mathbf{x} \in{\downarrow}\mathbf{x}^*$ **do**
    $h_{\mathbf{x}} \leftarrow \min(h_{\mathbf{x}}, y^*)$
  end for
  for all $\mathbf{x} \in{\uparrow}\mathbf{x}^*$ **do**
    $\ell_{\mathbf{x}} \leftarrow \max(\ell_{\mathbf{x}}, y^*)$
  end for
  for all $\mathbf{x} \in \mathcal{U}_{\mathcal{X}}$ **do**
    if $\ell_{\mathbf{x}} = h_{\mathbf{x}}$ **then**
      $y \leftarrow \ell_{\mathbf{x}}$
      $\mathcal{L}_D \leftarrow \mathcal{L}_D \cup \{(\mathbf{x}, y)\}$
      $G_{\mathcal{L}} \leftarrow G_{\mathcal{L}} \cup \{((\mathbf{x}, y), n(\mathbf{x}))\}$
      $\mathcal{U}_{\mathcal{X}} \leftarrow \mathcal{U}_{\mathcal{X}} \setminus \{\mathbf{x}\}$
    end if
  end for
  $max \leftarrow max - 1$
end while
return $\mathcal{L}$

Similarly to the MOCA algorithm, we shall express the constraint that Y is increasing in $\mathbf{X}$ in terms of the stochastic order constraint (3.2.3).

Because of the stochastic nature of the oracle, the set of examples constructed from its answers may contain monotonicity violations (see definition 23), which must be therefore repaired. In order to do so, we notice that if the prediction error is measured by means of a convex loss function (e.g. absolute error or squared error), then the Bayes allocation rule (21), namely

$$f^*(\mathbf{x}) = \arg\min_{y' \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} L(y, y') P(y|\mathbf{x})$$

is monotonic when the stochastic order constraint is satisfied [72]. Hence, it makes sense to relabel $\mathcal{L}$ by means of f^*.

We estimate f^* by using the relabelling algorithm proposed by Feelders in [39], which is based on the empirical risk minimisation principle (see definition 22) with the hypothesis space $\mathcal{H}$ equal to the class of monotonic functions. In other words, the algorithm estimates f^* as

$$\hat{f}(\mathbf{x}) = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)). \tag{5.2.6}$$

The relabelling algorithm runs in polynomial time, so this appears to be a feasible tool to be used in this second setting.

After applying the relabelling algorithm to a set of labelled examples, an interval $[\ell_{\mathbf{x}}, h_{\mathbf{x}}]$ of optimal labels is determined for each feature vector $\mathbf{x}$ occurring in the set. If $\ell_{\mathbf{x}} = h_{\mathbf{x}} = y$, then y is the label of $\mathbf{x}$ in every optimal relabelling. Hence, a very conservative inference strategy is to consider the collection of relabelled examples for which $\ell_{\mathbf{x}} = h_{\mathbf{x}}$.

Algorithm 5 outlines our algorithm for this second active learning setting: given the multiset $\mathcal{U}$ of unlabelled feature vectors, ND-MAL returns a multiset $\mathcal{L}$ of training examples labelled by querying the oracle at most *max* times for the labels of points chosen according to a heuristic H. Q denotes the collection of examples that have been labelled by querying the oracle, and $Q_{\mathcal{X}}$ the set of distinct feature vectors in Q.

For each $\mathbf{x} \in \mathcal{U}_{\mathcal{X}}$, ND-MAL expects the oracle to return $n(\mathbf{x})$ possibly distinct labels for $\mathbf{x}$. Every time the oracle is queried for the label of a feature vector $\mathbf{x}^*$, it replies with a label which is interpreted as a random draw from the labels of the labelled observations of $\mathbf{x}^*$ which have not been used yet, that is from the examples which are not in Q yet.

In line 1 the first query point $\mathbf{x}^*$ is chosen as the one which maximises H over $\mathcal{U}_{\mathcal{X}}$. After the oracle returns the label y^* of $\mathbf{x}^*$ (line 2), the labelled example $(\mathbf{x}^*, y^*)$ is added to Q (line 3), and the number of remaining unlabelled occurrences of $\mathbf{x}^*$ is decreased by one (line 4). If there are no more labelled occurrences, then $\mathbf{x}^*$ is removed from $\mathcal{U}_{\mathcal{X}}$ (line 5-7). The next query vector $\mathbf{x}^*$ chosen from Q is the vector which maximises the average of the values of the heuristic H corresponding to the minimal and the maximal optimal relabelling of Q respectively (line 10); the alternative of maximising the average over all optimal relabellings of Q is not viable

because the number of optimal relabellings can grow exponentially in the size of Q [39]. After the oracle returns the label y^* of the chosen query point $\mathbf{x}^*$, the labelled example $(\mathbf{x}^*, y^*)$ is added to Q (lines 14 and 15). In line 16, possible monotonicity violations in Q are resolved by applying the relabelling algorithm by [39]. The algorithm returns, for each feature vector $\mathbf{x}$ in Q, an interval $[\ell_{\mathbf{x}}, u_{\mathbf{x}}]$ of optimal labels; assigning each $\mathbf{x} \in Q$ the label $\ell_{\mathbf{x}}$ produces the minimal optimal monotonic relabelling of Q, and assigning $u_{\mathbf{x}}$ produces the maximal optimal monotonic relabelling of Q. The number of remaining unlabelled occurrences of $\mathbf{x}^*$ is decreased by one; if there are no more labelled occurrences of $\mathbf{x}^*$, then $\mathbf{x}^*$ is removed from $\mathcal{U}_{\mathcal{X}}$ (lines 17-20). When the maximum number of queries has been performed or $\mathcal{U}_{\mathcal{X}}$ is empty, for each queried vector $\mathbf{x} \in Q$, the respective intervals of possible labels for the vectors comprising $\downarrow\mathbf{x}$ and $\uparrow\mathbf{x}$ are updated based on monotonicity constraints (lines 23-30). The algorithm then returns $n(\mathbf{x})$ examples $(\mathbf{x}, y)$ of a feature vector $\mathbf{x}$ such that $\ell_{\mathbf{x}} = u_{\mathbf{x}} = y$ if

1. $\mathbf{x} \in Q_{\mathcal{X}}$, for $\mathbf{x}$ has label y in every optimal relabelling;
2. $\mathbf{x} \in \mathcal{U}_{\mathcal{X}}$, and its interval of possible labels has reduced to y because of the monotonicity constraint.

Hence, the labelled data returned by the algorithm consists of the set of examples which receive the same label in every optimal relabelling together with the set of the points whose labels can be uniquely inferred from the former.

5.3 Related Work

Torvik et al. [104] consider the problem of learning monotonic *Boolean* functions

$$f : \mathcal{X} \to \mathcal{Y},$$

which in our setting corresponds to the case that $\mathcal{X}_i = \{0, 1\}$ and $\mathcal{Y} = \{0, 1\}$. Loosely speaking, their objective is to determine for each $\mathbf{x} \in \mathcal{X}$ the value of $f(\mathbf{x})$ by asking as few queries as possible. They propose an algorithm which computes the minimum average number of queries, where the average is taken over all monotonic Boolean functions on $\mathcal{X}$. This algorithm is

Algorithm 5 $\mathcal{L}$ = ND-MAL($\mathcal{U}$, max)

$\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}\in\mathcal{U}_\mathcal{X}} H(\mathbf{x})$
$y^* \leftarrow \mathcal{O}(\mathbf{x}^*)$
$Q \leftarrow \{(\mathbf{x}^*, y^*)\}$
$n(\mathbf{x}^*) \leftarrow n(\mathbf{x}^*) - 1$
if $n(\mathbf{x}^*) = 0$ **then**
  $\mathcal{U}_\mathcal{X} \leftarrow \mathcal{U}_\mathcal{X} \setminus \{\mathbf{x}^*\}$
end if
while $max > 0 \wedge \mathcal{U}_\mathcal{X} \neq \varnothing$ **do**
  for all $\mathbf{x} \in \mathcal{U}_\mathcal{X}$ **do**
    $H_\ell(\mathbf{x}) \leftarrow$ heuristics value for $\mathbf{x}$ corresponding to the minimal optimal relabelling of Q
    $H_u(\mathbf{x}) \leftarrow$ heuristics value for $\mathbf{x}$ corresponding to the maximal optimal relabelling of Q
  end for
  $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}\in\mathcal{U}_\mathcal{X}} (H_\ell(\mathbf{x}) + H_u(\mathbf{x}))/2$
  $y^* \leftarrow \mathcal{O}(\mathbf{x}^*)$
  $Q \leftarrow Q \cup \{(\mathbf{x}^*, y^*)\}$
  Relabel(Q)
  $n(\mathbf{x}^*) \leftarrow n(\mathbf{x}^*) - 1$
  if $n(\mathbf{x}^*) = 0$ **then**
    $\mathcal{U}_\mathcal{X} \leftarrow \mathcal{U}_\mathcal{X} \setminus \{\mathbf{x}^*\}$
  end if
  $max \leftarrow max - 1$
end while
for all $\mathbf{x} \in Q_\mathcal{X}$ **do**
  for all $\mathbf{x}' \in{\downarrow}\mathbf{x}$ **do**
    $h_{\mathbf{x}'} \leftarrow \min(h_{\mathbf{x}'}, y)$
  end for
  for all $\mathbf{x}' \in{\uparrow}\mathbf{x}$ **do**
    $\ell_{\mathbf{x}'} \leftarrow \max(\ell_{\mathbf{x}'}, y)$
  end for
end for
$\mathcal{L} \leftarrow \varnothing$
for all $\mathbf{x} \in \mathcal{U}_\mathcal{X} \cup Q_\mathcal{X}$ **do**
  if $\ell_\mathbf{x} = h_\mathbf{x}$ **then**
    $y \leftarrow \ell_\mathbf{x}$
    $\mathcal{L}_D \leftarrow \mathcal{L}_D \cup \{(\mathbf{x}, y)\}$
    $G_\mathcal{L} \leftarrow G_\mathcal{L} \cup \{((\mathbf{x}, y), n(\mathbf{x}))\}$
  end if
end for
return $\mathcal{L}$

intractable for $p > 5$; nonetheless, based on the optimal solutions found for $p \leq 5$, the authors develop a heuristic which proves to be rather effective. Their proposed heuristic, which is based on the observation that in a chain poset, it is optimal to query the middle element, whereas in a sawtooth poset it is optimal to query an endpoint, corresponds to querying the oracle for the label of a vector $\mathbf{x}^*$ such that

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} |u(\mathbf{x}) - d(\mathbf{x})|. \tag{5.3.1}$$

The authors show that heuristic (5.3.1) produces optimal results for $p \leq 4$ and compares favourably to other heuristics for $p > 4$.

It should be noted, though, that there are several important differences between our problem setting and that addressed by Torvik et al. Firstly, we are not allowed to query the oracle for any arbitrary $\mathbf{x} \in \mathcal{X}$ but only for $\mathbf{x} \in S_{\mathcal{X}}$; moreover, $S_{\mathcal{X}}$ usually is only a small subset of $\mathcal{X}$. Secondly, the objective of Torvik et al. is to determine $f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, either by querying the oracle or by making inferences from labelled points. In our setting this is not possible; instead, we want to construct a good training sample by asking as few queries as possible. A third important difference is that we do not restrict ourselves to binary attributes and class labels. Instead, in our setting attributes can be numeric or discrete with ordered values; moreover, class labels are allowed to be non-binary as long as their values are ordered.

We pointed out in section 5.2.1 the equivalence of learning a deterministic binary monotonic function on a chain, and the problem of searching an element in a sorted list. Work has been done on searching elements in partially-ordered lists as well ([22, 78]), and it stands to reason that this problem is strongly related to learning a binary monotonic function on a partial order in the deterministic setting.

Carmo et al. [22] define an optimal search strategy as one where the longest search is as short as possible and prove that in general determining such a strategy is NP-hard. Nevertheless, they show that there are efficient approximation algorithms under the random graph model and the uniform model. It would be interesting to be able to apply this work to learning non-binary monotonic functions, for example by decomposing it into a collection

of binary problems. The following differences, though, need be taken into account:

1. Due to the nature of the problem, the work concerned with searching in partial orders only considers the deterministic setting.
2. The mapping of our problem to the problem of search only applies to the case of binary classification.
3. The work concerned with search focuses on finding algorithms with the best worst case complexity. Translated to the problem of learning a binary monotonic classification, this amounts to developing query strategies which minimise the number of queries that have to be asked in the worst case to infer the complete function. In the context of active learning, it is not so obvious that this should be a good criterion.

5.4 Experiments

To determine the quality our approach, we used the training data sets returned by the algorithms we proposed, each implemented with the proposed heuristics, to fit a classification model and then estimated the error rate of each learned model on a test sample. It should be noted that we do not enforce the monotonicity constraint on the trained models because we are interested in using monotonicity to infer the labels of data points as a way of augmenting the training sample; the fitted models are merely instrumental in determining the relative quality of the training samples inferred by using the proposed heuristics. We performed our evaluation on both artificial and real data sets.

Both algorithms were implemented employing Heuristic 1 (see definition 42) and Heuristic 2 (see definition 43). As for the latter, the posterior class probabilities $\hat{P}(y|\mathbf{x}_i)$ were estimated by using the weighted kNN probability estimation strategy discussed in section 3.6 with $k = 5$; since to compute the kNN estimates at least 5 labelled data points are needed, until that number was reached, Heuristic 1 was used instead.

Each data set, whether real or artificial, was partitioned into two parts. The first part was used as the knowledge that the oracle used to answer the queries of both algorithms. Each learned data set was then used to perform

weighted kNN classification of the second part of the original data set, with $k = 5$ (see section 3.6 for details); the classification took place only if the learned data set contained labelled instances of at least 5 distinct feature vectors. We repeated this 20 times and compared all the variants in terms of the average mean absolute error registered over the repetitions performed by the weighted kNN classifier.

The 20 experimental rounds on each data set were repeated three times, each time setting the maximum number of iterations *max* for each algorithm equal to 5, 10, and 20 per cent of the number of the individuals selected for active learning respectively.

For each algorithm, we used as baselines the average mean absolute error attained by

1. a kNN classifier based on the whole of the labelled observations set aside for active learning;
2. the algorithm itself with the heuristic choice of the next individual to query the oracle about replaced with random picking, with and without the interval of allowed labels updated based on monotonicity.

5.4.1 Artificial Data

We generated data from two monotonic non-linear models with two-dimensional feature vectors $\mathbf{x} = (x_1, x_2)$, the first having three class labels and the second having two. The class boundaries of both models are depicted in figure 5.7; for example, in the first model if $x_1 > 0.4$ and $x_2 > 0.4$, then the observation is assigned to class 3. To evaluate ND-MAL, we used the same data sets, perturbing them by adding a noise level of 0.1, which indicates that an observation from class 1 is flipped to class 2 with probability 0.05 and flipped to class 3 with probability 0.05 as well.

Moreover, it stands to reason that the impact of the monotonicity constraint depends on the shape of the partial order: if all attribute vectors are incomparable to each other, then the constraint brings no benefits because any label assignment is vacuously monotonic; on the other hand, if the training vectors form a chain, then we profit maximally from the monotonicity constraint. To verify this hypothesis, we generated different versions of the data sets each with a different degree of comparability among feature vec-

tors. We did so by sampling the two descriptive attributes from the uniform distribution on the interval $[0, 1]$ with different degrees of correlation: if the attributes are positively correlated, then there tend to be more comparable pairs than if they are not correlated, while there tend to be even fewer comparable pairs if there is a negative correlation. This phenomenon can be visualised easily: if the descriptive attributes are positively correlated, then feature vectors tend to cluster around a line with positive slope (see figure 5.5), which leads to a relatively high proportion of comparable pairs; on the other hand, in case of negative correlation, feature vectors tend to cluster around a line with negative slope (see figure 5.6), which leads to a relatively small proportion of comparable pairs.

For each of the 20 repetitions performed, from each of the two monotonic models considered we drew a random sample of 300 observations to use for active learning and another sample of size 1000 for testing.

The results for MAL (i.e. the deterministic case) are illustrated in tables 5.5 and 5.6. It is interesting to note how the correlation between the descriptive attributes and, hence, the number of comparable pairs, influences the results. If we consider the results for the data set with three class labels where we query 5% of the points, the trend is clear: higher correlation (namely, more comparable pairs) translates into better performance. For example, for $\rho = -0.9$, the lowest error was 0.2362, whereas the error using the complete training set was 0.0592. For $\rho = 0.9$, on the other hand, the best active learning result was 0.0597 versus 0.0368 when the whole training set was used. The effect of correlation becomes less relevant as the percentage of points queried becomes bigger, for the average error goes down for all active learning variants. With a few exceptions, both heuristics registered better performance than random querying with monotonicity inference on larger training samples. Random querying without monotonicity inference was clearly worse than all other options, as was expected. It should be noted that in some cases the number of points queried is smaller than the size of the part of the training set used because the algorithm simply ran out of unlabelled attribute vectors before the limit was reached.

The results for the data set with two class labels paint a similar picture, although the influence of correlation on performance is in this case rather small. On the other hand, the effect is still easily seen in the size of the

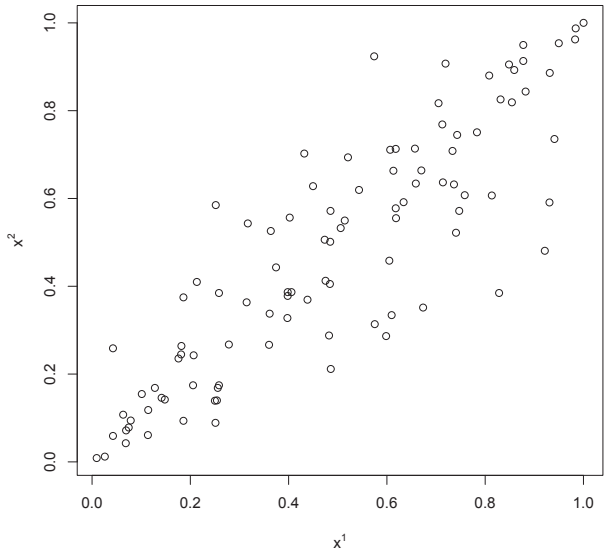

Figure 5.5: Artificial data used to test the active learning algorithms having strong positive correlation ($\rho = 0.9$). There are long chains from the bottom left to the top right corner. Of all pairs, 85% are comparable.

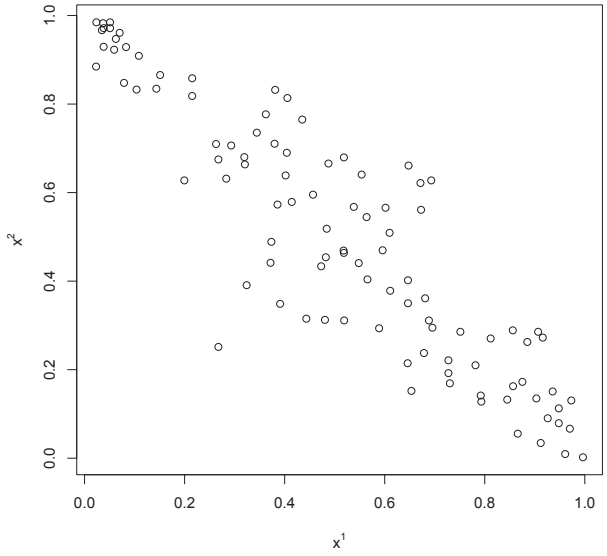

Figure 5.6: Artificial data used to test the active learning algorithms having strong positive correlation ($\rho = -0.9$). There are no long chains from the bottom left to the top right corner. Only 13% of the pairs are comparable.

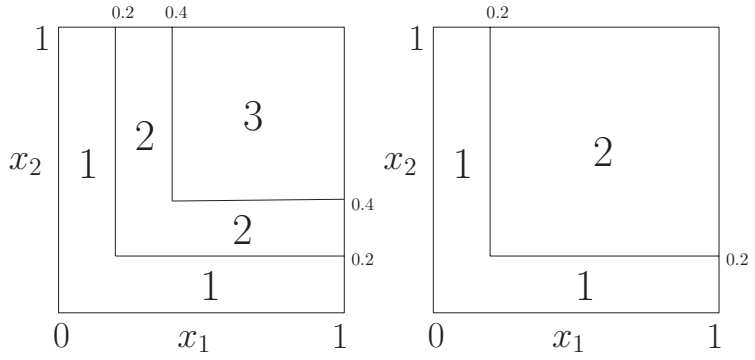

Figure 5.7: Class boundaries of the two monotonic non-linear models from which the artificial data used to test the active learning algorithms was generated. Both models have two-dimensional feature vectors $\mathbf{x} = (x_1, x_2)$; the first has three class labels, and the second has two.

training sample. For example, for $\rho = -0.9$ and 5%, Heuristic 1 allowed to obtain a training set of size 173.05 on average; for $\rho = 0.9$, the average size was 293.

For both data sets, when querying 20% of the points, the error attained is equal or slightly larger than when using the whole of the points. Surprisingly enough, in three-label data sets, for $\rho = 0$ the active learning actually leads to better performance than using the whole data set.

The results for ND-MAL (i.e. the non-deterministic case) are shown in table 5.7 and 5.8. Also in this case it is clear that higher correlation corresponds to better performance. For the data set with three class labels, for $\rho = -0.9$ and using 5% of the training data for active learning, Heuristic 2 performed best, for it attained an average error of 0.4853 against 0.3317 when the whole training set is used; on the other hand, for $\rho = 0.9$ and using 5% of the data, Heuristic 1 provided the best performance, for it allowed the algorithm to attain an average error of 0.3767 against 0.3297 when the whole training set is used. Similarly, for the data set with two class labels, for $\rho = -0.9$ and using 5% of the training data, Heuristic 2 was the best performing one as it attained an average error of 0.4114 against 0.2879 when the whole training set was used. For $\rho = 0.9$ and using 5% of the training data, random querying with monotonicity inference gives the best performance, attaining an average error of 0.2918, against 0.2618 when the

	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = −0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0592							
Random pick (no Monotonicity)	0.4099	0.2091	0.1433	15	30	60	15	30	60
Random pick (Monotonicity)	**0.2362**	0.1210	0.0709	"	"	"	95.75	159.10	243.30
Heuristic 1	0.2641	0.1047	0.0636	"	"	"	105.70	164.90	239.15
Heuristic 2	0.3160	**0.0880**	**0.0595**	"	"	59.5	100.65	214.05	285.50
Correlation = −0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0608							
Random pick (no Monotonicity)	0.4266	0.2620	0.1507	15	30	60	15	30	60
Random pick (Monotonicity)	0.2087	0.1094	0.0665	"	"	59.80	126.65	205.70	276.95
Heuristic 1	**0.1826**	**0.0910**	0.0661	"	"	60	159.75	225.75	281.80
Heuristic 2	0.2915	0.1238	**0.0630**	"	"	59.55	124.95	207.75	291.35
Correlation = 0	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0603							
Random pick (no Monotonicity)	0.3888	0.2449	0.1547	15	30	60	15	30	60
Random pick (Monotonicity)	0.2167	0.1097	***0.0601***	"	"	59.2	158.45	229.25	291.05
Heuristic 1	**0.1354**	**0.0773**	0.0628	"	"	59.8	196.35	252.90	293.70
Heuristic 2	0.2252	0.0987	*0.0602*	"	"	58.3	166.30	240.35	294.50
Correlation = 0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0503							
Random pick (no Monotonicity)	0.3392	0.2257	0.1303	15	30	60	15	30	60
Random pick (Monotonicity)	0.1677	0.0863	0.0519	"	"	57.30	183.35	243.25	295.30
Heuristic 1	**0.1216**	**0.0682**	**0.0504**	"	"	54.65	223.80	269.75	299.60
Heuristic 2	0.1628	0.0866	0.0506	"	"	57.05	193.30	256.15	298.85
Correlation = 0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0368							
Random pick (no Monotonicity)	0.2017	0.1027	0.0824	15	30	60	15	30	60
Random pick (Monotonicity)	0.0851	0.0502	0.0372	"	"	47.75	206.05	271.50	299.55
Heuristic 1	**0.0597**	**0.0447**	***0.0368***	"	"	42.85	258.55	285.85	300
Heuristic 2	0.0800	0.0464	***0.0368***	"	"	44.50	241.85	283.45	300

Table 5.5: Experimental results of MAL on the artificial data set with three labels without noise. Lowest mean absolute error is shown in boldface, and mean absolute errors for an actively-learned data set smaller than or equal to that of the whole data set are shown in italics.

whole training set is used. Unlike in the deterministic case, though, the two heuristics introduced registered worse performance than random querying with monotonicity inference on larger training samples; when using 20% of the labelled data, random querying with monotonicity inference generally has better performance than the other options, with an average error very close to that of the classifier trained on the whole training data set.

5.4.2 Real Data

We performed tests on the following real-world data sets (see Appendix A for more details on them):

- Auto MPG

	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = −0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0327							
Random pick (No Monotonicity)	0.2363	0.1121	0.0766	15	30	60	15	30	60
Random pick (Monotonicity)	0.0844	0.0391	0.0332	"	"	59.15	159.90	234.35	293.20
Heuristic 1	**0.0545**	0.0388	0.0332	"	"	60	173.05	231.65	281.30
Heuristic 2	0.1308	**0.0337**	***0.0327***	"	"	47.65	192.65	278.95	299.90
Correlation = −0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0359							
Random pick (No Monotonicity)	0.2897	0.1676	0.1051	15	30	60	15	30	60
Random pick (Monotonicity)	0.0761	0.0449	***0.0357***	"	"	46.25	204.50	269.10	299.25
Heuristic 1	**0.0590**	**0.0369**	*0.0359*	"	"	49.15	232.15	277.70	300
Heuristic 2	0.1585	0.0390	*0.0359*	"	"	44.45	189.55	279.35	300
Correlation = 0	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0344							
Random pick (No Monotonicity)	0.2508	0.1638	0.0884	15	30	60	15	30	60
Random pick (Monotonicity)	0.0710	0.0384	***0.0344***	"	29.90	42.10	238.75	284.90	300
Heuristic 1	**0.0505**	0.0358	***0.0344***	"	30	38.85	262.65	290.05	300
Heuristic 2	0.0810	**0.0351**	***0.0344***	29.95	38.80	"	238.20	289.15	300
Correlation = 0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0277							
Random pick (No Monotonicity)	0.1731	0.1208	0.0658	15	30	60	15	30	60
Random pick (Monotonicity)	0.0606	0.0310	***0.0277***	"	28.10	34.10	255.70	294.25	300
Heuristic 1	**0.0410**	**0.0281**	***0.0277***	"	28.30	28.90	282.05	299.40	300
Heuristic 2	0.0659	0.0297	***0.0277***	"	29.75	35.75	262.30	293.50	300
Correlation = 0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.0186							
Random pick (No Monotonicity)	0.1085	0.0518	0.0407	15	30	60	15	30	60
Random pick (Monotonicity)	0.0283	0.0197	***0.0186***	"	25.45	23.60	280.50	298.95	300
Heuristic 1	**0.0214**	***0.0186***	***0.0186***	"	21.65	21.65	293	300	300
Heuristic 2	0.0255	***0.0186***	***0.0186***	"	23.75	23.75	289.80	300	300

Table 5.6: Experimental results of MAL on the artificial data set with two labels without noise. Lowest mean absolute error is shown in boldface, and mean absolute errors for an actively-learned data set smaller than or equal to that of the whole data set are shown in italics.

- Computer Hardware
- ESL
- Haberman's Survival
- KC4
- Ohsumed
- Pima Indians
- Windsor Housing
- Wisconsin Breast Cancer

The numeric target attribute of the Auto MPG, Boston Housing, Computer Hardware, and Windsor Housing data sets were discretised into four labels using equal-frequency binning. Moreover, we did not use the whole of

	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = −0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3317							
Random pick (No Monotonicity)	0.6162	0.4849	0.4079	30	60	120	15	30	60
Random pick (Monotonicity)	0.5845	0.4582	**0.3678**	16	31	61	83.80	129.05	169.85
Heuristic 1	0.5697	0.5136	0.4455	"	"	"	112.50	169.15	233.35
Heuristic 2	**0.4853**	**0.4018**	0.3711	"	"	"	141.90	232.15	276
Correlation = −0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3425							
Random pick (No Monotonicity)	0.6157	0.5015	0.4207	30	60	120	15	30	60
Random pick (Monotonicity)	0.5851	0.4683	**0.3679**	16	31	61	108.95	164.60	206.75
Heuristic 1	0.5235	0.4444	0.4272	"	"	"	170.55	224.50	254
Heuristic 2	**0.4851**	**0.4149**	0.4029	"	"	"	189.05	262.45	282.05
Correlation = 0	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3559							
Random pick (No Monotonicity)	0.6427	0.5081	0.4345	30	60	120	15	30	60
Random pick (Monotonicity)	0.5376	0.4617	**0.3890**	16	31	61	108.60	167.95	219.95
Heuristic 1	**0.4622**	**0.4180**	0.3964	"	"	"	195.10	250.60	263
Heuristic 2	0.4825	0.4343	0.4094	"	"	"	198	265.35	283.85
Correlation = 0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3529							
Random pick (No Monotonicity)	0.5460	0.4750	0.4204	30	60	120	15	30	60
Random pick (Monotonicity)	0.4988	0.4368	**0.3713**	16	31	61	141.80	188	228.75
Heuristic 1	**0.4125**	**0.378**4	0.3735	"	"	"	231.05	268.90	270.75
Heuristic 2	0.4213	0.3900	0.3851	"	"	"	236.10	275.45	286.40
Correlation = 0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3297							
Random pick (No Monotonicity)	0.4672	0.4069	0.3723	30	60	120	15	30	60
Random pick (Monotonicity)	0.4107	0.3792	**0.3434**	16	31	61	180.80	223.95	246.50
Heuristic 1	**0.3767**	**0.3666**	0.3508	"	"	"	264.15	274.30	279.40
Heuristic 2	0.3799	0.3675	0.3542	"	"	"	265.30	289.80	281.55

Table 5.7: Experimental results of ND-MAL on the artificial data set having three labels with added noise. Lowest mean absolute error is shown in boldface.

the Ohsumed data but only the data for query 3 the descriptive attributes 1 and 16.

For each data set, we randomly drew 20 samples consisting of 70 per cent of the observations. Each experimental round consisted of applying our algorithms, which were implemented with Heuristic 1, Heuristic 2, random pick with and without monotonicity, to the drawn samples using 5%, 10%, and 20% of the data respectively. Each actively-learned data set and the whole of the sample were then used to perform kNN classification of the remaining 30% of the observations.

To make sure that the oracles used by the MAL algorithm be monotonic, we relabelled each original data set by using the algorithm presented by Feelders in [39] so that mean squared error was minimised. The choice of the mean squared error was purely arbitrary, for our only goal was to relabel

	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = −0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2879							
Random pick (No Monotonicity)	0.4378	0.3588	0.3233	30	60	120	15	30	60
Random pick (Monotonicity)	0.4537	0.3583	**0.3081**	16	31	61	76.90	124.90	
Heuristic 1	0.5384	0.4619	0.4160	"	"	"	83.65	151.60	231.70
Heuristic 2	**0.4114**	**0.3434**	0.3327	"	"	"	160.30	255.15	276.70
Correlation = −0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2914							
Random pick (No Monotonicity)	0.4695	0.3806	0.3480	30	60	120	15	30	60
Random pick (Monotonicity)	0.4692	**0.3942**	**0.3161**	16	31	61	114.35	157.55	216.25
Heuristic 1	0.4953	0.4469	0.4173	"	"	"	147.25	219.80	259.95
Heuristic 2	**0.4553**	0.3976	0.3908	"	"	"	209.70	275.35	284.30
	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = 0	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2944							
Random pick (No Monotonicity)	0.4443	0.3681	0.3459	30	60	120	15	30	60
Random pick (Monotonicity)	**0.3933**	**0.3547**	**0.3084**	16	31	61	134.75	170.30	233.75
Heuristic 1	0.4757	0.4443	0.4274	"	"	"	190.75	249.45	265.60
Heuristic 2	0.4727	0.4305	0.3996	"	"	"	218.40	278.40	288.55
	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = 0.5	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2874							
Random pick (No Monotonicity)	0.3793	0.3474	0.3077	30	60	120	15	30	60
Random pick (Monotonicity)	**0.3404**	**0.3204**	**0.2954**	16	31	61	152.60	219.15	248.25
Heuristic 1	0.4302	0.4058	0.3958	"	"	"	219.90	269.15	262.40
Heuristic 2	0.4205	0.4007	0.3924	"	"	"	249.75	289.75	291.35
	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Correlation = 0.9	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2618							
Random pick (No Monotonicity)	0.3416	0.2996	0.2784	30	60	120	15	30	60
Random pick (Monotonicity)	**0.2918**	**0.2790**	**0.2619**	16	31	61	225.50	250.60	265.40
Heuristic 1	0.4035	0.3723	0.3681	"	"	"	270.70	272.20	284.30
Heuristic 2	0.3937	0.3893	0.3724	"	"	"	283.75	292.70	292.85

Table 5.8: Experimental results of ND-MAL on the artificial data set having two labels with added noise. Lowest mean absolute error is shown in boldface.

the data sets in order to make them monotonic. Other loss functions such as mean absolute error might as well have been used. For example, to make the KC4 data set monotonic, 10 points were relabelled, with a total squared error of 13 (see table 5.9).

Change	-2	-1	0	1
Frequency	1	5	115	4

Table 5.9: Tests of the active learning algorithms on real data. Detailed relabelling results of the KC4 data set.

The results for MAL on the relabelled data sets are displayed in table 5.10. Firstly it should be noted that using the monotonicity constraint to infer new labels almost invariably results in a lower mean absolute error. Comparing the three implementations of the algorithm which use monotonicity inference, the implementations based on Heuristic 1 and Heuristic 2 usually lead to better results than querying points at random. The single, slight exception to this is represented by the Haberman's Survival data set when querying 20% of the data. In that case, the error attained by using the complete training set is only 0.1000, and the random pick without monotonicity inference has an error of 0.0973. Nonetheless, all other variants have slightly higher error; in particular, Heuristic 2 registered an error equal to 0.1000. In general, the models trained on the whole of each relabelled data set attained lower error, although sometimes the difference compared to using active learning is rather small, namely with Auto MPG at 20%, Haberman's Survival at 20%, and Windsor Housing at 20%. In the case of three relabelled data sets, namely Haberman's Survival, Pima Indians, and Wisconsin Breast Cancer, active learning actually lead to better performance than using the whole relabelled data set. It is hard to find a satisfactory explanation for this phenomenon. The expectation was also that on data sets with many class labels the results of active learning would be worse, since in that case it is harder to infer labels of attribute vectors. This intuition was confirmed by the results of our experiments: MAL's performance on ESL (9 labels) was relatively bad, whereas its performance on binary classification data sets was rather satisfactory. It can also noticed that on a data set with many comparable pairs such as Auto MPG MAL is able to infer plenty of labels: Heuristic 1 allowed the algorithm to infer a labelled data set with on average 171.55 data points when using 5% of the data set by querying just 14 attribute vectors. On the other hand, on data sets such as Wisconsin Breast Cancer which have very few comparable pairs, the algorithm was able to infer hardly any training points.

Finally, the results for ND-MAL on the original real data sets are shown in table 5.11. On the whole, the results are aligned with those of the other cases, but of course the results are worse than for the deterministic case. Nevertheless, using monotonicity inference almost always pays off, the Ohsumed data set being the only exception. Again, we note that

Heuristic 1 and Heuristic 2 almost invariably allow to infer a larger training set compared to random querying with monotonicity inference.

5.5 Conclusion

We introduced monotonicity constraints into active learning for supervised classification by proposing algorithms selecting instances to query the oracle which maximise the number of labels that can be inferred based on monotonicity. When the oracle is ensured to return monotonicity-preserving answers and in the presence of many comparable pairs, experiments show that active learning can drastically reduce the number queries necessary to infer a labelled data set capable of ensuring high levels of predictive accuracy; this is especially true when the number of class labels is not too large.

In order to deal with the realistic scenario in which the oracle does not always return monotonic labels, we have proposed an algorithm based on relabelling the set of queried points to make it monotonic while minimising absolute error. Only the attribute vectors whose label is uniquely determined after relabelling and vectors whose label can be inferred from them are included in the learned training sample.

Also the experiments we performed to test our second algorithm gave promising results; on the other hand, we think there is room for improvement. Firstly, as the number of class labels k increases, it becomes more and more difficult to infer unique class labels from points that the oracle has been queried for; consequently, the training set returned is likely to be relatively small. As there are expected to be several points whose set of possible labels is reduced considerably by means of monotonicity-based inference, by only considering points whose label has been uniquely determined in the training set the algorithm might end discarding potentially valuable information. Therefore, one possibility to improve the algorithm's performance could consist of exploiting this partial label information by also including into the inferred training sample examples corresponding to data points whose label has not been uniquely determined. Secondly, the model-based inference used in one of the many active learning approaches described in [97] could be used to complement the monotonicity-based inference the algorithm is based on.

	Mean Absolute Error			*Learned Data Set Size*		
Auto MPG	5%	10%	20%	5% (14)	10% (27.9)	20% (54.9)
Whole Data Set (275)		0.0889				
Random pick (No Monotonicity)	0.3235	0.1496	0.1017	14.10	28.05	55.45
Random pick (With Monotonicity)	0.1637	0.1073	0.0957	122.65	177.20	238.95
Heuristic 1	**0.1098**	0.0991	0.0932	171.55	217.30	254.95
Heuristic 2	0.1368	**0.0966**	**0.0897**	143.65	216.30	259.15
Computer Hardware	5%	10%	20%	5% (7)	10% (14)	20% (27.7)
Whole Data Set (147)		0.2718				
Random pick (No Monotonicity)	0.8048	0.6097	0.4387	7.55	15	30.15
Random pick (With Monotonicity)	0.6621	0.4573	0.3726	19.10	37.50	59.20
Heuristic 1	**0.4419**	**0.3306**	**0.3242**	31.90	45	72.50
Heuristic 2	0.4911	0.3742	0.3323	23.20	36.85	61.65
ESL	5%	10%	20%	5% (8.85)	10% (16.95)	20% (33.35)
Whole Data Set (342)		0.1640				
Random pick (No Monotonicity)	0.9493	0.6384	0.4445	17.15	34.55	69.60
Random pick (With Monotonicity)	0.9007	0.5966	0.4089	22.60	48.25	112.95
Heuristic 1	**0.5685**	0.4658	0.3349	51.55	97.65	172.95
Heuristic 2	0.5870	**0.4271**	**0.3158**	46.45	65.95	144.55
Haberman's Survival	5%	10%	20%	5% (10.9)	10% (20.9)	20% (41.20)
Whole Data Set (215)		0.1000				
Random pick (No Monotonicity)	0.1593	0.1571	0.1154	11.50	21.80	43.50
Random pick (With Monotonicity)	0.1203	0.1077	***0.0973***	119.30	162.55	205.40
Heuristic 1	0.1088	***0.0995***	0.1016	145.45	176.70	201.25
Heuristic 2	**0.1060**	0.1027	*0.1000*	147.20	181.75	210.85
KC4	5%	10%	20%	5% (5)	10% (9)	20% (17)
Whole Data Set (88)		0.4892				
Random pick (No Monotonicity)	NA	0.7297	0.6662	5.45	9.50	17.85
Random pick (With Monotonicity)	**0.7127**	0.7027	0.6297	9.65	14.35	25.80
Heuristic 1	0.7311	0.7284	0.7284	13.25	17.45	26.05
Heuristic 2	0.7824	**0.6581**	**0.5838**	13.20	20.05	30.65
Ohsumed	5%	10%	20%	5% (3)	10% (5.15)	20% (10.1)
Whole Data Set (165)		0.0450				
Random pick (No Monotonicity)	NA	0.1333	0.1650	9.45	16	34.65
Random pick (With Monotonicity)	0.1624	0.1014	0.0600	75.60	123.15	148.85
Heuristic 1	**0.0721**	**0.0643**	**0.0493**	137.25	146.90	162.35
Heuristic 2	0.1886	0.0993	0.0507	105.80	125.20	153.35
Pima Indians	5%	10%	20%	5% (27)	10% (54)	20% (108)
Whole Data Set (538)		0.2085				
Random pick (No Monotonicity)	0.2470	0.2530	0.2165	27	54	108
Random pick (With Monotonicity)	0.2343	0.2215	0.2133	123.85	186.50	283.35
Heuristic 1	0.2283	0.2209	***0.1970***	154.15	197.25	261.85
Heuristic 2	**0.2239**	**0.2185**	*0.2063*	176.95	244.35	337.80
Windsor Housing	5%	10%	20%	5% (19)	10% (38)	20% (75.45)
Whole Data Set (383)		0.3334				
Random pick (No Monotonicity)	0.6307	0.5025	0.4337	19.45	38.20	77.70
Random pick (With Monotonicity)	0.5724	0.4577	0.3969	54.95	97.85	166.60
Heuristic 1	**0.4926**	0.4037	0.3715	83.60	133.25	198
Heuristic 2	0.5000	**0.4227**	**0.3626**	57.90	104.10	176.25
Wisconsin Breast Cancer	5%	10%	20%	5% (7)	10% (14)	20% (28)
Whole Data Set (136)		0.2414				
Random pick (No Monotonicity)	**0.2621**	0.2448	***0.2397***	7	14	28
Random pick (With Monotonicity)	0.2853	0.2655	0.2422	8.45	17.55	33.7
Heuristic 1	0.2957	0.2638	***0.2397***	7	14	28
Heuristic 2	0.2957	**0.2422**	0.2509	7	24.85	42.2

Table 5.10: Experimental results of MAL on the relabelled real data sets. Lowest mean absolute error is shown in boldface, and mean absolute errors for an actively-learned data set smaller than or equal to that of the whole data set are shown in italics. The impossibility to perform kNN classification due to the lack of enough observations in the actively-learned data set is indicated as NA. The average number of queries performed is indicated in brackets.

	Mean Absolute Error			*Queries Performed*			*Learned Data Set Size*		
Auto MPG	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2355							
Random pick (No Monotonicity)	0.5145	0.3620	0.3004	14	28	55	14	28	55
Random pick (Monotonicity)	0.4303	0.4094	0.3026	"	"	"	52.60	83	119.65
Heuristic 1	**0.3962**	**0.3350**	**0.2927**	"	"	"	80.25	121.85	160.25
Heuristic 2	0.4239	0.3838	0.3274	"	"	"	71.35	88	126.95
Computer Hardware	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3798							
Random pick (No Monotonicity)	0.8798	0.6790	0.5323	7	14.05	27.80	7.60	15.35	30
Random pick (Monotonicity)	0.8285	0.6629	0.4968	"	"	"	26	38.80	58.50
Heuristic 1	**0.5702**	**0.4944**	**0.4685**	"	"	"	39.95	59.25	83.05
Heuristic 2	**0.5702**	0.4992	0.4774	"	"	"	"	56.15	81.20
ESL	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.3452							
Random pick (No Monotonicity)	1.0062	0.7342	0.6099	8.9	16.75	31.90	16.40	31.85	63.40
Random pick (Monotonicity)	1.1175	0.7161	0.6038	8.8	16.85	31.75	26.15	65.05	110.10
Heuristic 1	0.8295	**0.6024**	**0.4918**	8.9	17.05	33.35	54.40	102.65	184.55
Heuristic 2	**0.6932**	0.6257	0.5116	"	"	"	61	107.90	177.25
Haberman's Survival	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2808							
Random pick (No Monotonicity)	0.2923	0.2989	0.2885	10.8	21	40.65	11.50	21.95	43.25
Random pick (Monotonicity)	0.2918	0.2874	*0.2527*	"	"	40.75	103.40	132.50	157.15
Heuristic 1	*0.2725*	*0.2648*	*0.2648*	"	"	"	137.15	170.80	186.25
Heuristic 2	***0.2676***	***0.2484***	***0.2357***	"	"	40.80	144.55	191.95	204.20
KC4	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.7419							
Random pick (No Monotonicity)	NA	0.7716	0.7365	5	9	17	5.2	9.35	17.65
Random pick (Monotonicity)	***0.7405***	***0.7162***	***0.7284***	"	8.95	16.90	8.40	15.70	26.65
Heuristic 1	0.7959	0.7608	0.7676	"	9	17.05	14.80	17.80	20.25
Heuristic 2	0.7959	0.7608	0.7878	"	9	"	"	"	28
Ohsumed	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.6743							
Random pick (No Monotonicity)	NA	***0.6214***	**0.7014**	2.95	5.10	9.50	13	18.60	28.80
Random pick (Monotonicity)	0.8107	0.7960	0.7343	3	5.05	9.30	73.10	61.55	100
Heuristic 1	0.7979	0.8736	0.8121	"	5.20	8.90	69.40	116.80	136.65
Heuristic 2	0.7827	0.8621	0.7586	"	"	8.75	71.05	108.50	131.95
Pima Indians	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2630							
Random pick (No Monotonicity)	0.3115	0.2946	0.2822	27	54	108	27	54	108
Random pick (Monotonicity)	0.2880	**0.2778**	0.2735	"	"	"	131.30	187.15	256.45
Heuristic 1	**0.2848**	0.2793	0.2761	"	"	"	155.90	195.75	248.55
Heuristic 2	0.2989	0.2807	**0.2678**	"	"	"	250.15	323.05	361.10
Windsor Housing	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.6123							
Random pick (No Monotonicity)	0.7847	0.7003	0.6871	19	37.95	75.20	19.35	39.10	77.15
Random pick (Monotonicity)	0.7417	0.6844	**0.6227**	"	37.90	"	69.25	114.05	158.85
Heuristic 1	**0.7319**	**0.6831**	0.6635	"	37.95	75.05	138.70	182.85	240.10
Heuristic 2	0.7552	0.6874	0.6304	"	"	75.25	124.85	154.75	228.10
Wisconsin Breast Cancer	5%	10%	20%	5%	10%	20%	5%	10%	20%
Whole Data Set		0.2448							
Random pick (No Monotonicity)	0.3224	0.2759	0.2853	7	14	28	7	14	28
Random pick (Monotonicity)	*0.2466*	***0.2440***	**0.2681**	"	"	"	9.65	18.05	33.05
Heuristic 1	***0.2259***	0.2457	0.2974	"	"	"	7	14	28.75
Heuristic 2	***0.2259***	0.2560	**0.2681**	"	"	"	"	25.05	39.70

Table 5.11: Experimental results of ND-MAL on the original real data sets. Lowest mean absolute error is shown in boldface, and mean absolute errors for an actively-learned data set smaller than or equal to that of the whole data set are shown in italics. The impossibility to perform kNN classification due to the lack of enough observations in the actively-learned data set is indicated as NA. The average number of queries performed is indicated in brackets.

Chapter 6

Conclusions

In this thesis, we have presented pattern recognition algorithms tackling three different data mining problems by making use of or enforcing the monotonicity of the functional relation between the input and output spaces:

- We have presented MOCA, a new non-parametric monotonic classification algorithm which attempts to minimise the mean absolute prediction error for classification problems with ordered class labels. We have shown that MOCA minimises the L_1 error on the training sample subject to monotonicity constraints. Through experiments on artificial and real-world data sets, we have shown that it compares favourably to OSDL, a similar classification algorithm also intended for monotonic classification problems. Since the maximum-likelihood (ML) estimates of the posterior class probabilities used by MOCA and OSDL are typically based on very few observations, we conjectured that they both have a tendency to overfit on the training sample. We thought that a point of possible improvement would be obtained be considering an alternative to the ML estimates; therefore, we proposed a weighted k nearest neighbour (wkNN) approach to estimating the probabilities the algorithms use and then ran a second set of experiments to measure the improvements brought by using the wkNN estimates. Results showed that smoothing is beneficial for OSDL as its predictive performance was significantly better on a number of data sets and almost never worse; the adoption of the "balanced" and

"double balanced" versions of OSDL do not seem to lead to an improvement over the constant interpolation version on the data sets considered during the experiments. As for MOCA, smoothing seems to have a much more reduced effect; this is probably due to the fact that the isotonic regression already smooths the basic estimates by averaging them in case of order reversals. Hence, MOCA is already rather competitive when using wkNN probability estimates for $k = 1$, that is when using ML estimates of the posterior class probabilities. There are two interesting problems concerning MOCA which in our view are worth investigating in future research. We have only looked at the L_1 loss function, but this is certainly not the only appropriate loss function for ordinal classification problems; hence, it would be interesting to try to derive similar results for more general classes of loss functions. Another important issue is finding ways to reduce MOCA's execution time; one possibility might be represented by using the $O(n^2)$ approximate solution of the isotonic regression problem presented by Burdakov et al. in [20].

- We have then presented MIRA, a monotonic instance ranking algorithm. MIRA extends our work on non-parametric monotonic classification to ranking problems and builds on the best-performing decomposition and aggregation scheme among those proposed by Fürnkranz et al. in [51]. By performing experiments on real data, we showed that MIRA's predictive accuracy measured by means of the concordance index is comparable to that of the algorithm by Fürnkranz et al. Moreover, experiments performed on an artificial data set showed that MIRA can outperform the linear model the other algorithm is based on if class boundaries are monotonic and non-linear. More importantly, MIRA is guaranteed to produce a monotonic ranking function. Monotonicity is desired or even required for many applications, so this algorithm represents a valuable addition to existing ranking algorithms. One issue for future research is determining whether MIRA's predictive accuracy can be further improved by using a different estimator capable of attaining higher concordance index values. We tried to do so by using the weighted kNN estimator introduced by Barile and Feelders in [11] without obtaining any improvements. Another

way to improve the performance of the algorithm to investigate is the adoption of different aggregation schemes.

- Finally, we have introduced active learning algorithms for supervised classification which, in the presence of monotonicity constraints, select the feature vectors to query the oracle for in order to maximise the number of labels that can be inferred based on monotonicity. When the oracle is ensured to return monotonicity-preserving answers and in the presence of many comparable pairs, experiments showed that active learning can drastically reduce the number of queries necessary to infer a labelled data set capable of ensuring high levels of predictive accuracy; this is especially true when the number of class labels is not too large. In order to deal with the realistic scenario in which the oracle does not always return monotonic labels, we have proposed an algorithm based on relabelling the set of queried points to make it monotonic while minimizing absolute error. Only the attribute vectors whose label is uniquely determined after relabelling and attribute vectors whose label can be inferred from the former are included in the learned training sample. The experiments we performed to test our second algorithm also gave promising results; on the other hand, we think there is room for improvement. Firstly, as the number of class labels k increases, it becomes more and more difficult to infer unique class labels from points that the oracle has been queried for; consequently, the training set returned is likely to be relatively small. As there are expected to be several points whose set of possible labels is reduced considerably by means of monotonicity-based inference, by only considering points whose label has been uniquely determined in the training set, the algorithm might end discarding potentially valuable information. Therefore, one possibility to improve the algorithm's performance could consist of exploiting this partial label information by also including into the inferred training sample examples corresponding to data points whose label has not been uniquely determined. Secondly, the model-based inference used in one of the many active learning approaches described in [97] could be used to complement the monotonicity-based inference the algorithm is based on.

As a result of our research, it can be concluded that using or enforcing monotonicity in a predictive algorithm leads to performance comparable to or even better than that of state-of-the art, non-monotonic counterparts. Because this can be done at a computational cost which is affordable in most realistic cases, enforcing monotonicity when it is a property of the target concept or a requirement for the learned is a viable option.

Appendix A

Real-World Data Sets

To test the performance of our algorithms, we selected the following real-world data sets from different sources on the basis that the presence of an increasing (or decreasing) relation between the attributes and the response variable was a plausible assumption:

- the ERA (Employee Rejection/Acceptance), ESL (Employee Selection), LEV (Lecturers Evaluation), and SWD (Social Workers Decisions) data sets are available on the Web site of the Weka project [1] and have been donated by Dr. Arie Ben David [15];
- the Windsor Housing [3] data set is available on the Journal of Applied Econometrics Data Archive [2];
- the KC1, KC4, PC3 and PC4 data sets are included in the NASA's Metrics Data Program Data Repository [3];
- the Ohsumed data set is included the LETOR (LEarning TO Rank for information retrieval) collection of data sets [4], each of which contains a wide variety of queries with user feedback in several domains. Much of the research on preference learning is conducted on these data sets;
- the remaining data sets are included in the UCI machine learning repository [5] [6].

[1] http://www.cs.waikato.ac.nz/ml/weka/datasets.html
[2] http://econ.queensu.ca/jae/
[3] http://spinoff.nasa.gov/Spinoff2006/ct_1.html/
[4] http://research.microsoft.com/en-us/um/beijing/projects/letor/
[5] http://archive.ics.uci.edu/ml/

The chosen data sets vary widely in size and number and type of features in order to be representative of as wide range of data that may occur in practical situations as possible. Moreover, due to the lack of ordinal classification data sets, we also included a few regression data sets, discretising their target attributes using equal-frequency binning before running experiments on them. This approach, which as has often been pursued in the literature [45,48,51], is reasonable and has the advantage that, by changing the number of labels to discretise a data set into, several ordinal data sets can be obtained.

It should be emphasised that monotonicity judgements were only based on common sense. For example, the Ohsumed attributes are data retrieval metrics such as the counts of the number of times a query term appears in the title and abstract of a document respectively, and the class label indicates the relevance of the document to the query, where a higher label indicates higher relevance (the class labels are "irrelevant", "partially relevant" and "highly relevant"); as a consequence, it is reasonable to assume that the class label should be increasing in both attributes.

We did not always use the data sets in their original form. In general, when common sense suggested that there should be a decreasing rather increasing relationship between a feature and the response variable, we simply inverted the attribute values. We tested this by looking at the correlation between each attribute and the response. In case of a negative correlation between an attribute x and the response, we transformed the values of x as follows:

$$x_i = x_{max} - x_i + x_{min}, \qquad i = 1, \ldots, n$$

with $x_{max} = \max(x)$, and $x_{min} = \min(x)$.

More specifically,

- in the case of the Australian Credit data set, we only used the descriptive attributes 7, 8, 9 and 10 of the original data set.
- In the case of the Boston Housing data set, we excluded the *Charles River* dummy variable;
- we did not use the whole of the Ohsumed data set but instead only the data for query 3; in one case we only used the descriptive attributes 5, 6, 7, 18, 20, 21, 22, 35, 36, and 37, and in the other the descriptive

attributes 1 and 16;

- as for all of the NASA data sets, the attribute `ERROR_COUNT` was used as the response. All attributes that contained missing values were removed. Furthermore, the attribute `MODULE` was removed because it is a unique identifier of the module and the `ERROR_DENSITY` was removed because it is a function of the response variable. Finally, attributes with zero variance were removed.

Table A.1 lists some basic properties of the data sets. Besides the case of the Ohsumed data set, the table contains repeated entries for regression data sets whose target attributes were discretised into a different number of intervals. The first four columns of the table concern basis properties of the data sets: the first column of the table contains the overall number of labelled examples, the second column the number of distinct attribute vectors (multiple labelled occurrences of the same feature vector are possible in a data set), the third column contains the number of descriptive attributes *after* preprocessing, and the fourth column specifies the type of the target variable. The last two column of table A.1 contain some monotonicity properties of the data sets. The fifth column gives the number of comparable pairs (see definition 5) expressed as a percentage of the total number of pairs. This quantity gives and indication of the potential benefit of applying monotonicity constraints: the more the comparable pairs, the higher the potential benefit of taking monotonicity constraints into account. If all pairs are incomparable, the monotonicity constraint is satisfied vacuously. Finally, the sixth column gives the percentage of pairs which do not give rise to a monotonicity violation among the comparable ones; if this percentage is low, then the monotonicity assumption becomes questionable.

Data set	N	n	Number of Attributes	Target Variable	Comparable Pairs	Monotonicity
Australian Credit	690	360	4	Binary	71.62	95.34
Auto MPG[1]	392	392	7	Numeric	40.09	99.76
Auto MPG[2]	"	"	"	"	"	99.36
Boston Housing[1]	506	506	12	Numeric	19.10	97.17
Boston Housing[2]	"	"	"	"	"	96.99
Car Evaluation	1728	1728	6	4 Classes	14.36	99.96
Computer Hardware[1]	209	190	6	Numeric	49.53	98.49
Computer Hardware[2]	"	"	"	"	"	97.22
ERA	1000	44	4	9 Classes	16.77	85.08
ESL	488	199	4	9 Classes	70.65	98.85
Haberman's Survival	306	283	3	Binary	31.23	87.76
KC1	2107	1190	21	3 Classes	14.82	98.62
KC4	125	116	13	3 Classes	2.62	92.61
LEV	1000	92	4	5 Classes	24.08	95.73
Ohsumed[3]	235	194	10	3 Classes	52.29	81.61
Ohsumed[4]	"	55	2	"	77.99	80.79
PC3	1563	1436	36	3 Classes	0.12	99.22
PC4	1458	1343	37	4 Classes	0.12	99.08
Pima Indians	768	768	8	Binary	7.32	97.76
SWD	1,000	117	10	4 Classes	12.62	94.20
Windsor Housing[1]	546	529	11	Numeric	27.37	96.77
Windsor Housing[2]	"	"	"	"	27.37	96.37
Wisconsin Breast Cancer	194	194	32	2 Classes	0.56	96.19

Table A.1: Basic properties of the real-world data sets used to test our algorithms on. Besides the case of the Ohsumed data set, the table contains repeated entries for regression data sets whose target attributes were discretised into a different number of intervals. The first column contains the overall number of labelled examples, the second column the number of distinct attribute vectors (multiple labelled occurrences of the same feature vector are possible in a data set), the third column contains the number of descriptive attributes *after* preprocessing, and the fourth column specifies the type of the target variable, the fifth column the number of comparable pairs (see definition 5) expressed as a percentage of the total number of pairs, and the sixth column the percentage of pairs which do not give rise to a monotonicity violation (see definition 23) among the comparable ones.

[1]Discretised; Number of distinct labels: 4
[2]Discretised; Number of distinct labels: 5
[3]Query 3; attributes 5, 6, 7, 18, 20, 21, 22, 35, 36, and 37
[4]Query 3; attributes 1 and 16

Bibliography

[1] Ravindra K. Ahuja and James B. Orlin. A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Operations Research*, 49(5):784–789, 2001.

[2] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In Fahiem Bacchus and Tommi Jaakkola, editors, *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 18–25. AUAI Press, 2005.

[3] P.M. Anglin and R. Gençay. Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, 11(6):633–648, 1996.

[4] Norman P. Archer and Shi-Tong Wang. Application of the back-propagation neural network algorithm with monotonicity constraints for two-group classification problems. *Decision Sciences*, 24(1):60–75, 1993.

[5] Robert Arens. Learning SVM ranking functions from user feedback using document metadata and active learning in the biomedical domain. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 363–383. Springer-Verlag, 2010.

[6] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[7] Miriam Ayer, H.D. Brunk, George M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with in-

complete information. *Annals of Mathematical Statistics*, 26:641–647, 1955.

[8] Rajarajeswari Balasubramaniyan, Eyke Hüllermeier, Nils Weskamp, and Jörg Kämper. Clustering of gene expression data using a local shape-based similarity measure. *Bioinformatics*, 21(7):1069–1077, 2005.

[9] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *Journal of Computing and System Sciences*, 75:78–89, 2009.

[10] Nicola Barile and Ad J. Feelders. Nonparametric monotone classification with MOCA. In Fosca Giannotti, editor, *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM 2008)*, pages 731–736. IEEE Computer Society, 2008.

[11] Nicola Barile and Ad J. Feelders. Nonparametric ordinal classification with monotonicity constraints. In and Ad J. Feelders and Rob Potharst, editors, *Workshop Proceedings of MoMo 2009 at ECML PKDD 2009*, pages 47–63, 2009.

[12] Nicola Barile and Ad J. Feelders. Monotone instance ranking with mira. In *Discovery Science*, pages 31–45, 2011.

[13] Nicola Barile and Ad J. Feelders. Active learning with monotonicity constraints. In *Proceedings of the 14th international conference on Discovery Science*, 2012.

[14] Arie Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19:29–43, 1995.

[15] Arie Ben-David, Leon Sterling, and Yoh-Han Pao. Learning and classification of monotonic ordinal concepts. *Computational Intelligence*, 5:45–49, 1989.

[16] James O. Berger. *Statistical Decision Theory and Bayesian Analysis, 2nd Edition*. Springer, 1985.

[17] Jan C. Bioch and Viara Popova. Rough sets and ordinal classification. In *Proceedings of the 11th International Conference on Algorithmic Learning Theory*, ALT '00, pages 291–305, London, UK, UK, 2000. Springer-Verlag.

[18] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] H.D. Brunk. Maximum likelihood estimates of monotone parameters. *Annals of Mathematical Statistics*, 26:607–616, 1955.

[20] Oleg Burdakov, Oleg Sysoev, Anders Grimvall, and Mohamed Hussian. An algorithm for isotonic regression problems. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*, 2004.

[21] Kim Cao-Van. *Supervised Ranking, from Semantics to Algorithms*. PhD thesis, Universiteit Gent, 2003.

[22] Renato Carmo, Jair Donadelli, Yoshiharu Kohayakawa, and Eduardo Sany Laber. Searching in random partially ordered sets. In *Proceedings of the 5th Latin American Symposium on Theoretical Informatics*, LATIN '02, pages 278–292. Springer-Verlag, 2002.

[23] George Casella and Roger L. Berger. *Statistical inference*. Duxbury Press, 2. ed edition, 2002.

[24] Ramaswamy Chandrasekaran, Young U. Ryu, Varghese S. Jacob, and Sungchul Hong. Isotonic separation. *INFORMS Journal On Computing*, 17(4):462–474, 2005.

[25] Surajit Chaudhuri, Bee-Chung Chen, Venkatesh Ganti, and Raghav Kaushik. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd International Conference on Very large Databases*, VLDB '07, pages 327–338. VLDB Endowment, 2007.

[26] Surajit Chaudhuri and Luis Gravano. Evaluating top-k selection queries. In *Proceedings of VLDB 1999*, pages 397–410, 1999.

[27] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 1999.

[28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[29] Hennie A.M. Daniels and Marina V. Velikova. Derivation of monotone decision models from non-monotone data. Discussion Paper 2003-30, Tilburg University, Center for Economic Research, 2003.

[30] Krzysztof Dembczynski, Wojciech Kotlowski, and Roman Slowinski. Ordinal classification with decision rules. In Zbigniew W. Raś, Shusaku Tsumoto, and Djamel Zighed, editors, *Proceedings of the 3rd ECML/PKDD international conference on Mining complex data*, pages 169–181, Warsaw, Poland, 2007. Springer–Verlag.

[31] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7(December 2006):1–30, 2006.

[32] Marek J. Druzdzel and Linda C. van der Gaag. Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 141–148. Morgan Kaufmann, 1995.

[33] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[34] Wouter Duivesteijn and Ad J. Feelders. Nearest neighbour classification with monotonicity constraints. In Walter Daelemans, editor, *Proceedings of ECML/PKDD 2008*, volume 5211 of *LNAI*, pages 301–316. Springer, 2008.

[35] Richard Dykstra, John Hewett, and Tim Robertson. Nonparametric, isotonic discriminant procedures. *Biometrika*, 86(2):429–438, 1999.

[36] Hammou El Barmi and Hari Mukerjee. Inferences under a stochastic ordering constraint: the k-sample sase. *Journal of the American Statistical Association*, 100(469):252–261, 2005.

[37] Ad J. Feelders. Prior knowledge in economic applications of data mining. In D.A. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of PKDD 2000*, volume 1910 of *LNAI*, pages 395–400. Springer, 2000.

[38] Ad J. Feelders. A new parameter learning method for Bayesian networks with qualitative influences. In R. Parr and L.C. van der Gaag, editors, *Proceedings of Uncertainty in Artificial Intelligence 2007 (UAI07)*, pages 117–124. AUAI Press, 2007.

[39] Ad J. Feelders. Monotone relabeling in ordinal classification. In *ICDM 2010, Proceedings of the 10th IEEE International Conference on Data Mining*, pages 803–808. IEEE Computer Society, 2010.

[40] Ad J. Feelders and Martijn Pardoel. Pruning for monotone classification trees. In Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt, editors, *Advances in Intelligent Data Analysis V*, volume 2810 of *LNCS*, pages 1–12. Springer, 2003.

[41] Ad J. Feelders and Linda C. van der Gaag. Learning Bayesian network parameters under order constraints. In Peter Lucas, editor, *Proceedings of the second European workshop on probabilistic graphical models (PGM'04)*, pages 73–80, 2004.

[42] Ad J. Feelders and Linda C. van der Gaag. Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences. In Fahiem Bacchus and Tommi Jaakkola, editors, *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 193–200. AUAI Press, 2005.

[43] Ad J. Feelders and Linda C. van der Gaag. Learning bayesian network parameters under order constraints. *International Journal of Approximate Reasoning*, 42(1–2):37–53, 2006.

[44] Ad J. Feelders and Robert van Straalen. Parameter learning for Bayesian networks with strict qualitative influences. In Michael R. Berthold, John Shawe-Taylor, and Nadia Lavrač, editors, *Advances in Intelligent Data Analysis VII*, volume 4723 of *LNCS*, pages 48–58. Springer, 2007.

[45] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In Luc De Raedt and Peter A. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML/PKDD-01)*, pages 145–156, Freiburg, Germany, 2001. Springer-Verlag.

[46] FJohannes Fürnkranz and Eyke Hüllermeier, editors. *Preference Learning*. Springer-Verlag, 2010.

[47] Johannes Fürnkranz. Round robin classification. *The Journal of Machine Learning Research*, 2:721–747, March 2002.

[48] Johannes Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–403, October 2003.

[49] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In *Proceedings of the 14th European Conference on Machine Learning*, pages 145–156. Springer-Verlag, 2003.

[50] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning: An introduction. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 1–17. Springer-Verlag, 2010.

[51] Johannes Fürnkranz, Eyke Hüllermeier, and S. Vanderlooy. Binary decomposition methods for multipartite ranking. In Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, volume Part I, pages 359–374, Bled, Slovenia, 2009. Springer-Verlag.

[52] David Gamarnik. Efficient learning of monotone concepts via quadratic optimization. In *Proceedings of the Eleventh Annual Confer-*

ence on Computational Learning Theory, pages 134–143. ACM Press, 1998.

[53] Günther Gediga and Ivo Düntsch. Approximation quality for sorting rules. *Computational Statistics & Data Analysis*, 40(3):499–526, 2002.

[54] Mithat Gönen and Glenn Heller. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970, 2005.

[55] Salvatore Greco, B. Matarazzo, and R. Slowinski. Rough approximation by dominance relations. *International Journal of Intelligent Systems*, 17:153–171, 2002.

[56] Salvatore Greco, Roman Slowiński, Jerzy Stefanowski, and Marcin Żurawski. Lecture notes in computer science. In James F. Peters, Andrzej Skowron, Didier Dubois, Jerzy W. Grzymala-Busse, and Masahiro Inuiguchi, editors, *Transactions on Rough Sets II*, chapter Incremental versus non-incremental rule induction for multicriteria classification, pages 33–53. Springer-Verlag, Berlin, Heidelberg, 2004.

[57] Wolfgang Härdle. *Applied Nonparametric Regression*. Econometric Society Monographs. Cambridge University Press, 1992.

[58] O. Harrison and D. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 53:81–102, 1978.

[59] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[60] Klaus Hechenbichler and Klaus Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. Discussion Paper 399, Collaborative Research Center (SFB) 386 Statistical Analysis of Discrete Structures - Applications in Biometrics and Econometrics, 2004.

[61] Joseph L. Hellerstein. A statistical approach to diagnosing intermittent performance-problems using monotone relationships. In *Proceedings of the 1989 ACM SIGMETRICS International Conference on*

Measurement and Modeling of Computer Systems, pages 20–28. ACM Press, 1989.

[62] Eveline M. Helsper, Linda C. van der Gaag, Ad J. Feelders, Willie L.A. Loeffen, Petra L. Geenen, and Armin R.W. Elbers. Bringing order into bayesian-network construction. In *Proceedings of the Third International Conference on Knowledge Capture*, pages 121–128. ACM Press, 2005.

[63] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large-Margin Classifiers*, pages 115–132, 2000.

[64] Robert V. Hogg. On models and hypotheses with restricted alternatives. *Journal of the American Statistical Association*, 60(312):1153–1162, 1965.

[65] Eyke Hüllermeier and Johannes Fürnkranz. Learning label preferences: ranking error versus position error. In *Proceedings of the 6th international conference on Advances in Intelligent Data Analysis*, IDA'05, pages 180–191, Berlin, Heidelberg, 2005. Springer-Verlag.

[66] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.

[67] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 2002.

[68] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[69] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[70] Jørgen Karpf. Inductive modelling in law: Example based expert systems in administrative law. In *Proceedings of the Third International Conference on Artificial Intelligence in Law*, pages 297–306. ACM Press, 1991.

[71] Wojciech Kotlowski, Krzysztof Dembczynski, Salvatore Greco, and Roman Slowinski. Stochastic dominance-based rough set model for ordinal classification. *Information Sciences*, 178:4019–4037, 2008.

[72] Wojciech Kotlowski and Roman Slowinski. Statistical approach to ordinal classification with monotonicity constraints. In Eyke Hüllermeier and Johannes Fürnkranz, editors, *ECML PKDD 2008 Workshop on Preference Learning*, 2008.

[73] Wojciech Kotlowski and Roman Slowinski. Rule learning with monotonicity constraints. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 537–544, 2009.

[74] E.L. Lehmann. Ordered families of distributions. *Annals of Mathematical Statistics*, 26:399–Ũ419, 1955.

[75] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12. Springer-Verlag, 1994.

[76] Stijn F. Lievens and Bernard De Baets. Supervised ranking in the weka environment. *Inf. Sci.*, 180:4763–4771, December 2010.

[77] Stijn F. Lievens, Bernard De Baets, and Kim Cao-Van. A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operations Research*, 163:115–142, 2008.

[78] Nathan Linial and Michael E. Saks. Searching ordered structures. *Journal of Algorithms*, 6:86–103, 1985.

[79] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, pages 225–331, 2009.

[80] Antoine Mahul and Alexandre Aussem. Learning with monotonicity requirements for optimal routing with end-to-end quality of service constraints. In *European Symposium on Artificial Neural Networks*, pages 455–460, 2006.

[81] William L. Maxwell and John A. Muckstadt. Establishing consistent and realistic reorder intervals in production-distribution systems. *Operations Research*, 33(6):1316–1341, 1985.

[82] M.A. Meyer and J.M. Booker. *Eliciting and Analyzing Expert Judgment: A Practical Guide*. Series on Statistics and Applied Probability. ASA-SIAM, 2001.

[83] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[84] Helen M. Moshkovich, Alexander I. Mechitov, and David L. Olson. Rule induction in data mining: Effect of ordinal scales. *Expert Systems with Applications Journal*, 22:303–311, 2002.

[85] Hari Mukarjee and Steven Stern. Feasible nonparametric estimation of multiargument monotone functions. *Journal of the American Statistical Association*, 89(425):77–80, 1994.

[86] Wlodzimierz Ogryczak and Andrzej Ruszczynski. Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization*, 13(1):60–78, 2002.

[87] Giovanni Parmigiani and Lurdes Inoue. *Decision theory. Principles and approaches*. John Wiley & Sons, 2009.

[88] Michael J. Pazzani, S. Mani, and William R. Shankle. Acceptance of rules generated by machine learning among medical experts. *Methods of Information in Medicine*, 40:380–385, 2001.

[89] Jean-Claude Picard. Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):1268–1272, 1976.

[90] Rob Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4(2):97–112, 2000.

[91] Rob Potharst and Ad J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, 4:1–10, 2002.

[92] J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

[93] Shyamsundar Rajaram and Shivani Agarwal. Generalization bounds for k-partite ranking. In Shivani Agarwal, Corinna Cortes, and Ralf Herbrich, editors, *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, pages 28–23, 2005.

[94] Tim Robertson, F. Wright, and Richard L. Dykstra. *Order Restricted Statistical Inference*. Wiley, 1988.

[95] Patrick Royston. A useful monotonic non-linear model with applications in medicine and epidemiology. *Statistics in Medicine*, 19(15):2053–2066, 2000.

[96] Young U. Ryu, Ramaswamy Chandrasekaran, and Varghese S. Jacob. Breast cancer prediction using the isotonic separation technique. *European Journal Of Operational Research*, 181:842–854, 2007.

[97] Burr Settles. Active learning literature survey. Computer Science Technical Report 1648, University of Wisconsin–Madison, 2009.

[98] Joseph Sill. Monotonic networks. In *Advances in Neural Information Processing Systems*, NIPS (Vol. 10), pages 661–667, 1998.

[99] Joseph Sill and Yaser S. Abu-Mostafa. Monotonicity hints. In *Advances in Neural Information Processing Systems*, NIPS (Vol. 9), pages 634–640, 1997.

[100] J. Spouge, H. Wan, and W.J. Wilbur. Least squares isotonic regression in two dimensions. *Journal Of Optimization Theory And Applications*, 117(3):585–605, 2003.

[101] Luite Stegeman and Ad J. Feelders. On generating all optimal monotone classifications. In *Proceedings of the Eleventh IEEE International Conference on Data Mining (ICDM 2011)*, pages 685–694, 2011.

[102] Stanley S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, June 1946.

[103] Louis L. Thurstone. A law of comparative judgment. *Psychological Review*, 34:273–286, 1927.

[104] Vetle I. Torvik and Evangelos Triantaphyllou. Minimizing the average query complexity of learning monotone boolean functions. *INFORMS Journal on Computing*, 14(2):144–174, 2002.

[105] Rémon van de Kamp, Ad J. Feelders, and Nicola Barile. Isotonic classification trees. In Niall M. Adams, Céline Robardet, Arno P.J.M. Siebes, and Jean-Francois Boulicaut, editors, *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, pages 405–416, Lyon, France, 2009. Springer–Verlag.

[106] Linda C. van der Gaag, H. Bodlaender, and Ad J. Feelders. Monotonicity in Bayesian networks. In M. Chickering and J. Halpern, editors, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 569–576. AUAI Press, 2004.

[107] Linda C. van der Gaag, Hans L. Bodlaender, and Ad J. Feelders. Monotonicity in bayesian networks. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 569–576, Arlington, Virginia, United States, 2004. AUAI Press.

[108] Marina V. Velikova, Hennie A.M. Daniels, and Ad J. Feelders. Mixtures of monotone networks for prediction. *International Journal of Computational Intelligence*, 3(3):204–214, 2006.

[109] Andrew R. Webb. *Statistical Pattern Recognition, 2nd Edition*. John Wiley & Sons, 2002.

[110] Ting Yu, Simeon Simoff, and Tony Jan. Vqsvm: A case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing*, 73:2614–2623, 2010.

SIKS Dissertation Series

1998	01	Johan van den Akker (CWI), DEGAS - An Active, Temporal Database of Autonomous Objects
	02	Floris Wiesman (UM), Information Retrieval by Graphically Browsing Meta-Information
	03	Ans Steuten (TUD), A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
	04	Dennis Breuker (UM), Memory versus Search in Games
	05	E.W.Oskamp (RUL), Computerondersteuning bij Straftoemeting
1999	01	Mark Sloof (VU), Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products
	02	Rob Potharst (EUR), Classification using decision trees and neural nets
	03	Don Beal (UM), The Nature of Minimax Search
	04	Jacques Penders (UM), The practical Art of Moving Physical Objects
	05	Aldo de Moor (KUB), Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
	06	Niek J.E. Wijngaards (VU), Re-design of compositional systems
	07	David Spelt (UT), Verification support for object database design
	08	Jacques H.J. Lenting (UM), Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation
2000	01	Frank Niessink (VU), Perspectives on Improving Software Maintenance
	02	Koen Holtman (TUE), Prototyping of CMS Storage Management
	03	Carolien M.T. Metselaar (UvA), Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief
	04	Geert de Haan (VU), ETAG, A Formal Model of Competence Knowledge for User Interface Design
	05	Ruud van der Pol (UM), Knowledge-based Query Formulation in Information Retrieval
	06	Rogier van Eijk (UU), Programming Languages for Agent Communication
	07	Niels Peek (UU), Decision-theoretic Planning of Clinical Patient Management
	08	Veerle CoupĆ (EUR), Sensitivity Analyis of Decision-Theoretic Networks
	09	Florian Waas (CWI), Principles of Probabilistic Query Optimization
	10	Niels Nes (CWI), Image Database Management System Design Considerations, Algorithms and Architecture
	11	Jonas Karlsson (CWI), Scalable Distributed Data Structures for Database Management
2001	01	Silja Renooij (UU), Qualitative Approaches to Quantifying Probabilistic Networks
	02	Koen Hindriks (UU), Agent Programming Languages: Programming with Mental Models
	03	Maarten van Someren (UvA), Learning as problem solving
	04	Evgueni Smirnov (UM), Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
	05	Jacco van Ossenbruggen (VU), Processing Structured Hypermedia: A Matter of Style
	06	Martijn van Welie (VU), Task-based User Interface Design
	07	Bastiaan Schonhage (VU), Diva: Architectural Perspectives on Information Visualization
	08	Pascal van Eck (VU), A Compositional Semantic Structure for Multi-Agent Systems Dynamics

09 Pieter Jan 't Hoen (RUL), Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
10 Maarten Sierhuis (UvA), Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
11 Tom M. van Engers (VUA), Knowledge Management: The Role of Mental Models in Business Systems Design

2002 01 Nico Lassing (VU), Architecture-Level Modifiability Analysis
02 Roelof van Zwol (UT), Modelling and searching web-based document collections
03 Henk Ernst Blok (UT), Database Optimization Aspects for Information Retrieval
04 Juan Roberto Castelo Valdueza (UU), The Discrete Acyclic Digraph Markov Model in Data Mining
05 Radu Serban (VU), The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
06 Laurens Mommers (UL), Applied legal epistemology; Building a knowledge-based ontology of the legal domain
07 Peter Boncz (CWI), Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
08 Jaap Gordijn (VU), Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
09 Willem-Jan van den Heuvel (KUB), Integrating Modern Business Applications with Objectified Legacy Systems
10 Brian Sheppard (UM), Towards Perfect Play of Scrabble
11 Wouter C.A. Wijngaards (VU), Agent Based Modelling of Dynamics: Biological and Organisational Applications
12 Albrecht Schmidt (UvA), Processing XML in Database Systems
13 Hongjing Wu (TUE), A Reference Architecture for Adaptive Hypermedia Applications
14 Wieke de Vries (UU), Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
15 Rik Eshuis (UT), Semantics and Verification of UML Activity Diagrams for Workflow Modelling
16 Pieter van Langen (VU), The Anatomy of Design: Foundations, Models and Applications
17 Stefan Manegold (UvA), Understanding, Modeling, and Improving Main-Memory Database Performance

2003 01 Heiner Stuckenschmidt (VU), Ontology-Based Information Sharing in Weakly Structured Environments
02 Jan Broersen (VU), Modal Action Logics for Reasoning About Reactive Systems
03 Martijn Schuemie (TUD), Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
04 Milan Petkovic (UT), Content-Based Video Retrieval Supported by Database Technology
05 Jos Lehmann (UvA), Causation in Artificial Intelligence and Law - A modelling approach
06 Boris van Schooten (UT), Development and specification of virtual environments
07 Machiel Jansen (UvA), Formal Explorations of Knowledge Intensive Tasks
08 Yongping Ran (UM), Repair Based Scheduling
09 Rens Kortmann (UM), The resolution of visually guided behaviour
10 Andreas Lincke (UvT), Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
11 Simon Keizer (UT), Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
12 Roeland Ordelman (UT), Dutch speech recognition in multimedia information retrieval
13 Jeroen Donkers (UM), Nosce Hostem - Searching with Opponent Models
14 Stijn Hoppenbrouwers (KUN), Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
15 Mathijs de Weerdt (TUD), Plan Merging in Multi-Agent Systems
16 Menzo Windhouwer (CWI), Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
17 David Jansen (UT), Extensions of Statecharts with Probability, Time, and Stochastic Timing
18 Levente Kocsis (UM), Learning Search Decisions

2004 01 Virginia Dignum (UU), A Model for Organizational Interaction: Based on Agents, Founded in Logic
02 Lai Xu (UvT), Monitoring Multi-party Contracts for E-business
03 Perry Groot (VU), A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
04 Chris van Aart (UvA), Organizational Principles for Multi-Agent Architectures
05 Viara Popova (EUR), Knowledge discovery and monotonicity
06 Bart-Jan Hommes (TUD), The Evaluation of Business Process Modeling Techniques
07 Elise Boltjes (UM), Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
08 Joop Verbeek(UM), Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiŁle gegevensuitwisseling en digitale expertise
09 Martin Caminada (VU), For the Sake of the Argument; explorations into argument-based reasoning
10 Suzanne Kabel (UvA), Knowledge-rich indexing of learning-objects
11 Michel Klein (VU), Change Management for Distributed Ontologies
12 The Duy Bui (UT), Creating emotions and facial expressions for embodied agents
13 Wojciech Jamroga (UT), Using Multiple Models of Reality: On Agents who Know how to Play
14 Paul Harrenstein (UU), Logic in Conflict. Logical Explorations in Strategic Equilibrium
15 Arno Knobbe (UU), Multi-Relational Data Mining
16 Federico Divina (VU), Hybrid Genetic Relational Search for Inductive Learning
17 Mark Winands (UM), Informed Search in Complex Games
18 Vania Bessa Machado (UvA), Supporting the Construction of Qualitative Knowledge Models
19 Thijs Westerveld (UT), Using generative probabilistic models for multimedia retrieval
20 Madelon Evers (Nyenrode), Learning from Design: facilitating multidisciplinary design teams

2005 01 Floor Verdenius (UvA), Methodological Aspects of Designing Induction-Based Applications
02 Erik van der Werf (UM), AI techniques for the game of Go
03 Franc Grootjen (RUN), A Pragmatic Approach to the Conceptualisation of Language
04 Nirvana Meratnia (UT), Towards Database Support for Moving Object data
05 Gabriel Infante-Lopez (UvA), Two-Level Probabilistic Grammars for Natural Language Parsing
06 Pieter Spronck (UM), Adaptive Game AI
07 Flavius Frasincar (TUE), Hypermedia Presentation Generation for Semantic Web Information Systems
08 Richard Vdovjak (TUE), A Model-driven Approach for Building Distributed Ontology-based Web Applications
09 Jeen Broekstra (VU), Storage, Querying and Inferencing for Semantic Web Languages
10 Anders Bouwer (UvA), Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
11 Elth Ogston (VU), Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
12 Csaba Boer (EUR), Distributed Simulation in Industry
13 Fred Hamburg (UL), Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
14 Borys Omelayenko (VU), Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
15 Tibor Bosse (VU), Analysis of the Dynamics of Cognitive Processes
16 Joris Graaumans (UU), Usability of XML Query Languages
17 Boris Shishkov (TUD), Software Specification Based on Re-usable Business Components
18 Danielle Sent (UU), Test-selection strategies for probabilistic networks
19 Michel van Dartel (UM), Situated Representation
20 Cristina Coteanu (UL), Cyber Consumer Law, State of the Art and Perspectives
21 Wijnand Derks (UT), Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006 01 Samuil Angelov (TUE), Foundations of B2B Electronic Contracting
02 Cristina Chisalita (VU), Contextual issues in the design and use of information technology in organizations

03 Noor Christoph (UvA), The role of metacognitive skills in learning to solve problems
04 Marta Sabou (VU), Building Web Service Ontologies
05 Cees Pierik (UU), Validation Techniques for Object-Oriented Proof Outlines
06 Ziv Baida (VU), Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
07 Marko Smiljanic (UT), XML schema matching - balancing efficiency and effectiveness by means of clustering
08 Eelco Herder (UT), Forward, Back and Home Again - Analyzing User Behavior on the Web
09 Mohamed Wahdan (UM), Automatic Formulation of the Auditor's Opinion
10 Ronny Siebes (VU), Semantic Routing in Peer-to-Peer Systems
11 Joeri van Ruth (UT), Flattening Queries over Nested Data Types
12 Bert Bongers (VU), Interactivation - Towards an e-cology of people, our technological environment, and the arts
13 Henk-Jan Lebbink (UU), Dialogue and Decision Games for Information Exchanging Agents
14 Johan Hoorn (VU), Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
15 Rainer Malik (UU), CONAN: Text Mining in the Biomedical Domain
16 Carsten Riggelsen (UU), Approximation Methods for Efficient Learning of Bayesian Networks
17 Stacey Nagata (UU), User Assistance for Multitasking with Interruptions on a Mobile Device
18 Valentin Zhizhkun (UvA), Graph transformation for Natural Language Processing
19 Birna van Riemsdijk (UU), Cognitive Agent Programming: A Semantic Approach
20 Marina Velikova (UvT), Monotone models for prediction in data mining
21 Bas van Gils (RUN), Aptness on the Web
22 Paul de Vrieze (RUN), Fundaments of Adaptive Personalisation
23 Ion Juvina (UU), Development of Cognitive Model for Navigating on the Web
24 Laura Hollink (VU), Semantic Annotation for Retrieval of Visual Resources
25 Madalina Drugan (UU), Conditional log-likelihood MDL and Evolutionary MCMC
26 Vojkan Mihajlovic (UT), Score Region Algebra: A Flexible Framework for Structured Information Retrieval
27 Stefano Bocconi (CWI), Vox Populi: generating video documentaries from semantically annotated media repositories
28 Borkur Sigurbjornsson (UvA), Focused Information Access using XML Element Retrieval

---

2007 01 Kees Leune (UvT), Access Control and Service-Oriented Architectures
02 Wouter Teepe (RUG), Reconciling Information Exchange and Confidentiality: A Formal Approach
03 Peter Mika (VU), Social Networks and the Semantic Web
04 Jurriaan van Diggelen (UU), Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
05 Bart Schermer (UL), Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance
06 Gilad Mishne (UvA), Applied Text Analytics for Blogs
07 Natasa Jovanovic' (UT), To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
08 Mark Hoogendoorn (VU), Modeling of Change in Multi-Agent Organizations
09 David Mobach (VU), Agent-Based Mediated Service Negotiation
10 Huib Aldewereld (UU), Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols
11 Natalia Stash (TUE), Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
12 Marcel van Gerven (RUN), Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
13 Rutger Rienks (UT), Meetings in Smart Environments; Implications of Progressing Technology
14 Niek Bergboer (UM), Context-Based Image Analysis
15 Joyca Lacroix (UM), NIM: a Situated Computational Memory Model
16 Davide Grossi (UU), Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
17 Theodore Charitos (UU), Reasoning with Dynamic Networks in Practice

18 Bart Orriens (UvT), On the development an management of adaptive business collaborations
19 David Levy (UM), Intimate relationships with artificial partners
20 Slinger Jansen (UU), Customer Configuration Updating in a Software Supply Network
21 Karianne Vermaas (UU), Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
22 Zlatko Zlatev (UT), Goal-oriented design of value and process models from patterns
23 Peter Barna (TUE), Specification of Application Logic in Web Information Systems
24 Georgina Ramírez Camps (CWI), Structural Features in XML Retrieval
25 Joost Schalken (VU), Empirical Investigations in Software Process Improvement

---

2008 01 Katalin Boer-Sorbán (EUR), Agent-Based Simulation of Financial Markets: A modular, continuous-time approach
02 Alexei Sharpanskykh (VU), On Computer-Aided Methods for Modeling and Analysis of Organizations
03 Vera Hollink (UvA), Optimizing hierarchical menus: a usage-based approach
04 Ander de Keijzer (UT), Management of Uncertain Data - towards unattended integration
05 Bela Mutschler (UT), Modeling and simulating causal dependencies on process-aware information systems from a cost perspective
06 Arjen Hommersom (RUN), On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective
07 Peter van Rosmalen (OU), Supporting the tutor in the design and support of adaptive e-learning
08 Janneke Bolt (UU), Bayesian Networks: Aspects of Approximate Inference
09 Christof van Nimwegen (UU), The paradox of the guided user: assistance can be counter-effective
10 Wauter Bosma (UT), Discourse oriented summarization
11 Vera Kartseva (VU), Designing Controls for Network Organizations: A Value-Based Approach
12 Jozsef Farkas (RUN), A Semiotically Oriented Cognitive Model of Knowledge Representation
13 Caterina Carraciolo (UvA), Topic Driven Access to Scientific Handbooks
14 Arthur van Bunningen (UT), Context-Aware Querying; Better Answers with Less Effort
15 Martijn van Otterlo (UT), The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains
16 Henriette van Vugt (VU), Embodied agents from a user's perspective
17 Martin Op 't Land (TUD), Applying Architecture and Ontology to the Splitting and Allying of Enterprises
18 Guido de Croon (UM), Adaptive Active Vision
19 Henning Rode (UT), From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search
20 Rex Arendsen (UvA), Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven
21 Krisztian Balog (UvA), People Search in the Enterprise
22 Henk Koning (UU), Communication of IT-Architecture
23 Stefan Visscher (UU), Bayesian network models for the management of ventilator-associated pneumonia
24 Zharko Aleksovski (VU), Using background knowledge in ontology matching
25 Geert Jonker (UU), Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency
26 Marijn Huijbregts (UT), Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled
27 Hubert Vogten (OU), Design and Implementation Strategies for IMS Learning Design
28 Ildiko Flesch (RUN), On the Use of Independence Relations in Bayesian Networks
29 Dennis Reidsma (UT), Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans
30 Wouter van Atteveldt (VU), Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content
31 Loes Braun (UM), Pro-Active Medical Information Retrieval

32 Trung H. Bui (UT), Toward Affective Dialogue Management using Partially Observable Markov Decision Processes
33 Frank Terpstra (UvA), Scientific Workflow Design; theoretical and practical issues
34 Jeroen de Knijf (UU), Studies in Frequent Tree Mining
35 Ben Torben Nielsen (UvT), Dendritic morphologies: function shapes structure

---

2009 01 Rasa Jurgelenaite (RUN), Symmetric Causal Independence Models
02 Willem Robert van Hage (VU), Evaluating Ontology-Alignment Techniques
03 Hans Stol (UvT), A Framework for Evidence-based Policy Making Using IT
04 Josephine Nabukenya (RUN), Improving the Quality of Organisational Policy Making using Collaboration Engineering
05 Sietse Overbeek (RUN), Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
06 Muhammad Subianto (UU), Understanding Classification
07 Ronald Poppe (UT), Discriminative Vision-Based Recovery and Recognition of Human Motion
08 Volker Nannen (VU), Evolutionary Agent-Based Policy Analysis in Dynamic Environments
09 Benjamin Kanagwa (RUN), Design, Discovery and Construction of Service-oriented Systems
10 Jan Wielemaker (UvA), Logic programming for knowledge-intensive interactive applications
11 Alexander Boer (UvA), Legal Theory, Sources of Law & the Semantic Web
12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin), Operating Guidelines for Services
13 Steven de Jong (UM), Fairness in Multi-Agent Systems
14 Maksym Korotkiy (VU), From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)
15 Rinke Hoekstra (UvA), Ontology Representation - Design Patterns and Ontologies that Make Sense
16 Fritz Reul (UvT), New Architectures in Computer Chess
17 Laurens van der Maaten (UvT), Feature Extraction from Visual Data
18 Fabian Groffen (CWI), Armada, An Evolving Database System
19 Valentin Robu (CWI), Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
20 Bob van der Vecht (UU), Adjustable Autonomy: Controling Influences on Decision Making
21 Stijn Vanderlooy (UM), Ranking and Reliable Classification
22 Pavel Serdyukov (UT), Search For Expertise: Going beyond direct evidence
23 Peter Hofgesang (VU), Modelling Web Usage in a Changing Environment
24 Annerieke Heuvelink (VUA), Cognitive Models for Training Simulations
25 Alex van Ballegooij (CWI), RAM: Array Database Management through Relational Mapping
26 Fernando Koch (UU), An Agent-Based Model for the Development of Intelligent Mobile Services
27 Christian Glahn (OU), Contextual Support of social Engagement and Reflection on the Web
28 Sander Evers (UT), Sensor Data Management with Probabilistic Models
29 Stanislav Pokraev (UT), Model-Driven Semantic Integration of Service-Oriented Applications
30 Marcin Zukowski (CWI), Balancing vectorized query execution with bandwidth-optimized storage
31 Sofiya Katrenko (UvA), A Closer Look at Learning Relations from Text
32 Rik Farenhorst (VU)and Remco de Boer (VU), Architectural Knowledge Management: Supporting Architects and Auditors
33 Khiet Truong (UT), How Does Real Affect Affect Affect Recognition In Speech?
34 Inge van de Weerd (UU), Advancing in Software Product Management: An Incremental Method Engineering Approach
35 Wouter Koelewijn (UL), Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling
36 Marco Kalz (OUN), Placement Support for Learners in Learning Networks
37 Hendrik Drachsler (OUN), Navigation Support for Learners in Informal Learning Networks
38 Riina Vuorikari (OU), Tags and self-organisation: a metadata ecology for learning resources in a multilingual context
39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin), Service Substitution - A Behavioral Approach Based on Petri Nets

40 Stephan Raaijmakers (UvT), Multinomial Language Learning: Investigations into the Geometry of Language
41 Igor Berezhnyy (UvT), Digital Analysis of Paintings
42 Toine Bogers (UvT), Recommender Systems for Social Bookmarking
43 Virginia Nunes Leal Franqueira (UT), Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients
44 Roberto Santana Tapia (UT), Assessing Business-IT Alignment in Networked Organizations
45 Jilles Vreeken (UU), Making Pattern Mining Useful
46 Loredana Afanasiev (UvA), Querying XML: Benchmarks and Recursion

---

2010 01 Matthijs van Leeuwen (UU), Patterns that Matter
02 Ingo Wassink (UT), Work flows in Life Science
03 Joost Geurts (CWI), A Document Engineering Model and Processing Framework for Multimedia documents
04 Olga Kulyk (UT), Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments
05 Claudia Hauff (UT), Predicting the Effectiveness of Queries and Retrieval Systems
06 Sander Bakkes (UvT), Rapid Adaptation of Video Game AI
07 Wim Fikkert (UT), Gesture interaction at a Distance
08 Krzysztof Siewicz (UL), Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments
09 Hugo Kielman (UL), A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging
10 Rebecca Ong (UL), Mobile Communication and Protection of Children
11 Adriaan Ter Mors (TUD), The world according to MARP: Multi-Agent Route Planning
12 Susan van den Braak (UU), Sensemaking software for crime analysis
13 Gianluigi Folino (RUN), High Performance Data Mining using Bio-inspired techniques
14 Sander van Splunter (VU), Automated Web Service Reconfiguration
15 Lianne Bodenstaff (UT), Managing Dependency Relations in Inter-Organizational Models
16 Sicco Verwer (TUD), Efficient Identification of Timed Automata, theory and practice
17 Spyros Kotoulas (VU), Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications
18 Charlotte Gerritsen (VU), Caught in the Act: Investigating Crime by Agent-Based Simulation
19 Henriette Cramer (UvA), People's Responses to Autonomous and Adaptive Systems
20 Ivo Swartjes (UT), Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative
21 Harold van Heerde (UT), Privacy-aware data management by means of data degradation
22 Michiel Hildebrand (CWI), End-user Support for Access to Heterogeneous Linked Data
23 Bas Steunebrink (UU), The Logical Structure of Emotions
24 Dmytro Tykhonov, Designing Generic and Efficient Negotiation Strategies
25 Zulfiqar Ali Memon (VU), Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective
26 Ying Zhang (CWI), XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines
27 Marten Voulon (UL), Automatisch contracteren
28 Arne Koopman (UU), Characteristic Relational Patterns
29 Stratos Idreos(CWI), Database Cracking: Towards Auto-tuning Database Kernels
30 Marieke van Erp (UvT), Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval
31 Victor de Boer (UvA), Ontology Enrichment from Heterogeneous Sources on the Web
32 Marcel Hiel (UvT), An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems
33 Robin Aly (UT), Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval
34 Teduh Dirgahayu (UT), Interaction Design in Service Compositions
35 Dolf Trieschnigg (UT), Proof of Concept: Concept-based Biomedical Information Retrieval
36 Jose Janssen (OU), Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

37 Niels Lohmann (TUE), Correctness of services and their composition
38 Dirk Fahland (TUE), From Scenarios to components
39 Ghazanfar Farooq Siddiqui (VU), Integrative modeling of emotions in virtual agents
40 Mark van Assem (VU), Converting and Integrating Vocabularies for the Semantic Web
41 Guillaume Chaslot (UM), Monte-Carlo Tree Search
42 Sybren de Kinderen (VU), Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach
43 Peter van Kranenburg (UU), A Computational Approach to Content-Based Retrieval of Folk Song Melodies
44 Pieter Bellekens (TUE), An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain
45 Vasilios Andrikopoulos (UvT), A theory and model for the evolution of software services
46 Vincent Pijpers (VU), e3alignment: Exploring Inter-Organizational Business-ICT Alignment
47 Chen Li (UT), Mining Process Model Variants: Challenges, Techniques, Examples
48 Withdrawn
49 Jahn-Takeshi Saito (UM), Solving difficult game positions
50 Bouke Huurnink (UvA), Search in Audiovisual Broadcast Archives
51 Alia Khairia Amin (CWI), Understanding and supporting information seeking tasks in multiple sources
52 Peter-Paul van Maanen (VU), Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention
53 Edgar Meij (UvA), Combining Concepts and Language Models for Information Access

2011 01 Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models
02 Nick Tinnemeier(UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language
03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems
04 Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms
05 Base van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline
06 Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage
07 Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction
08 Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues
09 Tim de Jong (OU), Contextualised Mobile Media for Learning
10 Bart Bogaert (UvT), Cloud Content Contention
11 Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective
12 Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining
13 Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling
14 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
15 Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval
16 Maarten Schadd (UM), Selective Search in Games of Different Complexity
17 Jiyin He (UvA), Exploring Topic Structure: Coherence, Diversity and Relatedness
18 Mark Ponsen (UM), Strategic Decision-Making in complex games
19 Ellen Rusman (OU), The Mind ' s Eye on Personal Profiles
20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach
21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems
22 Junte Zhang (UvA), System Evaluation of Archival Description and Access
23 Wouter Weerkamp (UvA), Finding People and their Utterances in Social Media
24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior
25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics
26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
28 Rianne Kaptein(UvA), Effective Focused Retrieval by Exploiting Query Context and Document Structure
29 Faisal Kamiran (TUE), Discrimination-aware Classification
30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions
31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality
32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science
33 Tom van der Weide (UU), Arguing to Motivate Decisions
34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization
39 Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
40 Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
41 Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
42 Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
43 Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
44 Boris Reuderink (UT), Robust Brain-Computer Interfaces
45 Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
46 Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47 Azizi Bin Ab Aziz(VU), Exploring Computational Models for Intelligent Support of Persons with Depression
48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49 Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

---

2012 01 Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
02 Muhammad Umair(VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
03 Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
04 Jurriaan Souer (UU), Development of Content Management System-based Web Applications
05 Marijn Plomp (UU), Maturing Interorganisational Information Systems
06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
08 Gerben de Vries (UvA), Kernel Methods for Vessel Trajectories
09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
13 Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
14 Evgeny Knutov(TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes
16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment

17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
18 Eltjo Poort (VU), Improving Solution Architecting Practices
19 Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval
22 Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
26 Emile de Maat (UvA), Making Sense of Legal Text
27 Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
28 Nancy Pascall (UvT), Engendering Technology Empowering Women
29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
31 Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
33 Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
35 Evert Haasdijk (VU), Never Too Old To Learn - On-line Evolution of Controllers in Swarm- and Modular Robotics
36 Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks
40 Agus Gunawan (UvT), Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU), Game Design Patterns for Learning
42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
43 Withdrawn
44 Anna Tordai (VU), On Combining Alignment Techniques
45 Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
46 Simon Carter (UvA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UvA), Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
49 Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50 Steven van Kervel (TUD), Ontologogy driven Enterprise Information Systems Engineering
51 Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

---

2013 01 Viorel Milea (EUR), News Analytics for Financial Decision Support
02 Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
03 Szymon Klarman (VU), Reasoning with Contexts in Description Logics
04 Chetan Yadati(TUD), Coordinating autonomous planning and scheduling
05 Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
06 Romulo Goncalves(CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
07 Giel van Lankveld (UvT), Quantifying Individual Player Differences
08 Robbert-Jan Merk(VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators

09 Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications
10 Jeewanie Jayasinghe Arachchige(UvT), A Unified Modeling Framework for Service Design
11 Evangelos Pournaras(TUD), Multi-level Reconfigurable Self-organization in Overlay Services
12 Marian Razavian(VU), Knowledge-driven Migration to Services
13 Mohammad Safiri(UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14 Jafar Tanha (UvA), Ensemble Approaches to Semi-Supervised Learning Learning
15 Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
16 Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation
17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid
18 Jeroen Janssens (UvT), Outlier Selection and One-Class Classification
19 Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling
20 Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval
21 Sander Wubben (UvT), Text-to-text generation by monolingual machine translation
22 Tom Claassen (RUN), Causal Discovery and Logic
23 Patricio de Alencar Silva(UvT), Value Activity Monitoring
24 Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning
25 Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
26 Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning
27 Mohammad Huq (UT), Inference-based Framework Managing Data Provenance
28 Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience
29 Iwan de Kok (UT), Listening Heads
30 Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support
31 Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications
32 Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
33 Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere
34 Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search
35 Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction Promotor: Prof. dr. L. Hardman (CWI/UVA)
36 Than Lam Hoang (TUE), Pattern Mining in Data Streams
37 Dirk Börner (OUN), Ambient Learning Displays
38 Eelco den Heijer (VU), Autonomous Evolutionary Art
39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games
41 Jochem Liem (UvA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
42 Léon Planken (TUD), Algorithms for Simple Temporal Reasoning
43 Marc Bron (UvA), Exploration and Contextualization through Interaction and Concepts

2014 01 Nicola Barile (UU), Studies in Learning Monotonic Models from Data

Curriculum Vitae

Nicola Barile was born in Bari, Italy on 21st July 1971. He earned his master's degree in Computer Science at the University of Bari in Italy in 2006 by defending his thesis titled 'Transductive Classification: A Relational Approach' under the supervision of Prof. Donato Malerba. Upon completing his undergraduate studies, he participated in a EU-funded project on Web Usage Mining carried out at a Web consultancy based in Bari, Italy under the supervision of Prof. Malerba. In 2008 he started his PhD in Data Mining at Utrecht University within the AMOC (Algorithms for Nonparametric Monotonic Classification) project under the supervision of Prof. Arno Siebes and of Dr. Ad Feelders. After completing his doctoral work and while finalising his thesis, he worked as a quantitative developer at a software house based in Amsterdam, the Netherlands. Since the end of 2013, namely a few months before defending his PhD, he has been working at ING in Amsterdam, the Netherlands as a data scientist specialising in Big Data.