University of Ljubljana
Faculty of Computer and Information Science

Lan Žagar

Ranking by Multitask Learning

doctoral dissertation

Advisor: Prof. Dr. Blaž Zupan

Co-advisor: Prof. Dr. John Shawe-Taylor

Ljubljana, 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Lan Žagar

Rangiranje s hkratnim učenjem več nalog

doktorska disertacija

Mentor: Prof. Dr. Blaž Zupan

Somentor: Prof. Dr. John Shawe-Taylor

Ljubljana, 2014

Abstract

Instance ranking is a subfield of supervised machine learning and is concerned with inferring predictive models that can rank a set of data instances. We focus on multipartite ranking, where instances belong to one of a limited set of rank classes, study different approaches on synthetic and real data sets, and propose a ranking-specific evaluation framework and a new learning approach that combines multitask learning and binary decomposition.

The thesis starts with an analysis of existing ranking approaches. These are used in a practical application of ranking within the domain of molecular biology. In particular, we study embryonic stem cell differentiation posed as a multipartite ranking problem. We critically evaluate several ranking approaches and demonstrate how they can be used to construct accurate predictive models that can rank samples based on their stage of differentiation. For testing, we introduce a framework for evaluation of ranking methods including a generalization of the popular performance measure AUC (area under the ROC curve) that can be used for multipartite problems.

We proceed by analysing the family of methods based on binary decomposition, which reduces a ranking problem to a set of binary classification tasks. To improve the process of learning models for these tasks, we suggest to combine it with a multitask learning technique. Specifically, we propose a new ranking method, which we name BDMT, that combines one-against-one decomposition and multitask feature learning. The decomposition allows us to model nonlinear patterns and to simplify the learning domain to problems that are suitable for classical machine learning approaches. Multitask learning allows us to generalize across the decomposed tasks, making them interdependent through a joint regularized optimization. Our experiments show that the addition of multitask learning can greatly improve the performance of one-against-one decomposition and even succeed in outperforming

state-of-the-art ranking approaches in certain settings. Learning the models of decomposed tasks simultaneously, increases the stability of model estimation and reduces the sensitivity to perturbations of the training data set. Compared with other ranking methods that are also able to model complex patterns, BDMT remains efficient and can achieve low training times with the use of fast linear base learners. We also show how the method and the patterns it learns can be interpreted. New features learned during the training of BDMT can reveal important hidden factors and hence map the problem into a low-dimensional subspace spanned by a set of novel features. Individual models of decomposed tasks and the similarities between them can be studied to further elucidate the distribution of rank classes in this subspace.

Keywords

ranking, instance ranking, multipartite ranking, binary decomposition, multitask learning, one-against-one, feature learning, machine learning, data mining, artificial intelligence

Povzetek

Rangiranje primerov je disciplina nadzorovanega strojnega učenja, ki se ukvarja z gradnjo napovednih modelov, zmožnih rangiranja novih množic učnih primerov. V doktorski disertaciji se posvetimo večdelnemu rangiranju, kjer vsak učni primer pripada enemu izmed urejenih razredov rangiranja. Preučimo različne pristope rangiranja s pomočjo umetnih in stvarnih podatkovnih naborov ter metodologije za objektivno vrednotenje, prirejene za večdelno rangiranje. Predlagamo tudi nov učni pristop, ki združuje tehnike binarnega razcepa ter hkratnega učenja več nalog.

Disertacijo začnemo z analizo obstoječih pristopov za rangiranje. Te uporabimo tudi na primeru razvoja zarodnih celic, zastavljenega kot problem večdelnega rangiranja. Različne metode kritično ovrednotimo in pokažemo, da se je z njimi možno naučiti natančnih napovednih modelov, sposobnih rangiranja primerov glede na razvojno stopnjo. Za primerjavo metod uporabimo razvito metodologijo vrednotenja, ki med drugim vključuje posplošitev ploščine pod krivuljo (AUC) za uporabo v večdelnem rangiranju.

V nadaljevanju se posvetimo družini metod za rangiranje, ki uporablja binarni razcep, s katerim prevede problem rangiranja na množico problemov uvrščanja v dva razreda. Da bi izboljšali učenje nastale množice problemov, predlagamo nadgradnjo s tehnikami hkratnega učenja več nalog (ang. multitask learning). Pregledamo lastnosti različnih tehnik binarnega razcepa in hkratnega učenja z namenom iskanja kombinacije, kjer bi ti komponenti med sabo čim bolje sodelovali. To nas pripelje do predloga nove metode za rangiranje, poimenovane BDMT, ki povezuje metodi razcepa eden-proti-enemu in hkratnega učenja značilk.

Sledi eksperimentalna analiza, s katero najprej potrdimo vpliv lastnosti množice nalog, dobljenih z razcepom, na možne izboljšave hkratnega učenja. Pokažemo, da pristopi, osnovani na binarnem razcepu, kljub preprosto-

sti omogočajo modeliranje nelinearnih zakonitosti v podatkih. Analiziramo vpliv števila učnih primerov, značilk ter rangov na rezultate, ki jih doseže BDMT. Napredek metode BDMT v primerjavi z osnovno verzijo razcepa eden-proti-enemu eksperimentalno potrdimo in s tem upravičimo predlagani dodatek hkratnega učenja. Poleg tega se izkaže, da je z metodo BDMT možno doseči in preseči tudi druge uveljavljene pristope s področja rangiranja.

Analizo metode BDMT zaključimo s pregledom ostalih prednosti, ki niso povezane z izboljšavo točnosti modelov. Uporaba hkratnega učenja izboljša stabilnost, ki sicer pri metodah razcepa pogosto trpi zaradi učenja nalog z manj podatki. Prav tako je v primerjavi z ostalimi metodami, ki so sposobne odkriti zapletene podatkovne vzorce, hitrost učenja z metodo BDMT boljša, še posebej kadar kot osnovo uporabimo hitre linearne metode. Pokažemo tudi, kako je možno modele, naučene z metodo BDMT, razlagati. Pri tem so nam v pomoč značilke odkrite s hkratnim učenjem in opazovanje ter primerjava posameznih modelov nalog, dobljenih z razcepom.

Ključne besede

rangiranje, rangiranje primerov, večdelno rangiranje, binarni razcep, hkratno učenje več nalog, eden-proti-enemu, učenje značilk, strojno učenje, odkrivanje znanj iz podatkov, umetna inteligenca

Izjava o avtorstvu doktorske disertacije

Spodaj podpisani **Lan Žagar**, z vpisno številko **63040306**, sem avtor doktorske disertacije z naslovom:

Rangiranje s hkratnim učenjem več nalog

S svojim podpisom zagotavljam, da:

- sem doktorsko disertacijo izdelal samostojno pod vodstvom mentorja prof. dr. Blaža Zupana in somentorstvom prof. dr. Johna Shawe-Taylorja,
- so elektronska oblika doktorske disertacije, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko doktorske disertacije,
- soglašam z javno objavo elektronske oblike doktorske disertacije v zbirki "Dela FRI".

V Ljubljani, dne 2. april 2014 Podpis avtorja:

Acknowledgements

First and foremost I would like to thank my advisor, professor Dr. Blaž Zupan, for his guidance and help over the past five years. He taught me what questions to ask and how to answer them. And how to find new questions after that.

I would also like to thank my co-advisor, professor Dr. John Shawe-Taylor, who helped make my research visit to UCL possible. I discovered many things included in this dissertation there.

For all the scientific and technical help, but most of all for making the Bioinformatics laboratory a group of friends instead of just a group of co-workers, I extend my thanks to all current and former lab members. To our more or less temporary visitors from all over the world, especially Francesca Mulas, who made the research of some infernal data sets more heavenly. And to the former lab colleagues from the AI Lab and former classmates scattered throughout the faculty and other institutions.

To all my friends I would like to apologize for recent neglect; I will do my best to make up for it. Last but not least, I wish to thank my family and especially my parents, who have always been supportive and provided me with everything I truly needed. Thank you all, for the food, but even more, for all the food for thought.

Contents

Chapter 1

Introduction

We deal with ranking problems every day. For example, we rank a list of films to decide which are very good (we buy the DVD), worth seeing (rent the DVD or watch it on TV), and which are not worth seeing. Often we get a ranking from somebody else, like a friend's film suggestions or Google's ranking of web pages relevant to our search query. Afterwards, we can evaluate if the suggestions were good — similar to how we would rank the items — and decide whether to use the source again in the future or not.

Machine learning can help us solve problems when the amount of data is too large, the tasks repetitive or even to obtain better results then we could by ourselves. Solving ranking problems is no exception and the number of methods and researchers developing them is rapidly increasing. Research of ranking and related problems goes back for several decades, but was, at the beginning, mostly limited to two types of problems. One was considered as a part of information retrieval and concerned itself with the ranking of documents with respect to their relatedness to a given query. The other came in the form of an alternative performance measure for binary classification. Instead of predicting one of two labels and computing the classification accuracy, the algorithm can predict probabilities, which can be used to rank the data instances. One of the most widely used measures to evaluate such rankings is the area under the ROC (receiver operating characteristic) curve also known as AUC, which corresponds to the probability that a positive instance will be ranked higher than a negative one. When our goal is to achieve a high AUC score, the task is transformed from a classification to a

ranking problem.

Recently, however, many more new and challenging real world tasks are being formulated as ranking problems. This has resulted in the emergence of additional, slightly modified, definitions of ranking problems and sparked a stronger interest in ranking as its own research area. Diverse types of problems and goals offer the opportunity for different approaches to excel and take advantage of their unique qualities. This makes it even more interesting as a subject of research, since it favours new innovative ideas as opposed to using established off-the-shelf solutions.

One approach that has been used before, but we will try to improve upon, is using binary decomposition to transform a ranking problem into a set of classification problems. That way, it is possible to build upon the extensive work on classification, and use it to build ranking models. In its basic form, however, the obtained set of classification problems was treated just as many individual learning problems to be solved.

Meanwhile, specialized methods are being developed that can take advantage of known or even unknown structure and relations in the input data. The availability of richer data sets and multiple sources of data, which can be related to one another, is leading to more effort being put into integration and transfer of knowledge. One area that shows promise and provides some of the benefits that we seek is multitask learning. This is an area dealing with the problem of learning multiple (related) tasks. The hypothesis of multitask methods is that there are some similarities between tasks which can lead to better models when learning considers all tasks simultaneously and can share the extracted knowledge between them.

Connecting the two ideas of binary decomposition and multitask learning appears to offer great promise and can be used to improve the construction of ranking methods. Generating binary classification subproblems using a specific procedure from a bigger (ranking) data set inevitably leads to them being related and sharing some properties and structure. Multitask methods could therefore offer a clear advantage by considering the set of subproblems together, exploiting these properties and structure.

1.1 Thesis overview

This thesis will give an overview of existing approaches to ranking, analyse some of the key properties of different methods and evaluate methods using a relevant ranking problem from the field of bioinformatics. We will also propose and evaluate a new method for ranking combining binary decomposition with multitask learning.

The second chapter provides the context for research topics covered in this thesis and gives an overview of existing related work. The main research areas, where ranking-related topics appear, are described.

The third chapter describes ranking in greater detail, defines the types of ranking problems and describes which of them will be the focus of our research. It contains an overview and categorization of existing approaches. Experiments on synthetic data are used to analyse properties of ranking problems and how to handle them. A comprehensive study of a realistic problem from bioinformatics shows the use of the developed ranking methodology.

The fourth chapter covers binary decomposition with its uses for ranking and multitask learning. We propose how these two areas can be combined to improve learning of separate models for the decomposed subproblems. A new method based on this combination is proposed and analysed.

In the fifth chapter we summarize the findings of the thesis and give our conclusions. We also mention some possibilities for future work.

The Appendix contains a more comprehensive set of results for the case study on embryonic stem cell differentiation (section 3.6), and an extended abstract in Slovene.

1.2 Contributions

The main contributions of the dissertation are:

- a comparative analysis of existing approaches for ranking,
- proposal of a new method for ranking based on binary decomposition and multitask learning.

The first contribution consists of selecting and critically evaluating methods from different branches of the machine learning field that can be used to rank data instances. The evaluation is done using methodology adapted for the problem of ranking.

The second contribution is in showing the benefits of using binary decomposition for ranking and how learning can be improved further by taking advantage of the structure of the decomposed subproblems. A new method combining binary decomposition and multitask learning is proposed and evaluated, demonstrating improved accuracy and other benefits such as better interpretability.

Chapter 2

Related work

Ranking has been present in different forms for some time and, as a consequence, work related to it can be found in many areas of machine learning, information retrieval and statistical modelling. We review some of this work and the areas where it originated, concentrating on subjects most related to this thesis.

At first, ranking problems have been researched mostly in the scope of information retrieval [1, 2, 3, 4, 5, 6]. There, the main use of ranking is to order the retrieved results (*e.g.* documents or web pages) based on their relevance to the search query. This application has remained an active topic of interest and the research of it has continued to the present day. Several ranking methods have been developed for solving this problem, many of which have since transcended their use in information retrieval and have become widely used in other applications as well. For example, the ranking support vector machine (SVM) method described by Joachims [7] was first used for a meta-search engine, where it learned a retrieval function from clickthrough data. Now, it is adopted as a general purpose ranking method and is one of the baselines for comparison in many ranking studies [8, 9, 10, 11], including this dissertation.

Recently, the use of ranking in many other applications is increasing, leading to new formulations of ranking problems being proposed and studied. It could be said, that ranking is evolving into a distinct area of machine learning [12], which places more emphasis on the general study of various ranking problems than on a specific application. In this respect, it is ap-

proaching other areas of supervised machine learning, such as classification and regression.

Other names have been used to refer to the same or very similarly defined problems as ranking. Learning to rank [13] and learning to order [14] have been used, especially in the area of information retrieval, to emphasize the difference from plain ranking (sorting) of documents based on a given attribute. With the proposal to automatically learn the predictive ranking model from data, the explicit reference to learning was used to distinguish the proposed approach from other work that is not based on learning from data. Recently, research on these topics is more often reported in the scope of machine learning, where, due to the context, the explicit reference to learning becomes obsolete.

Preference learning [15] is an umbrella term that encompasses various types of ranking problems in addition to other problems from information retrieval and recommender systems. While these related approaches are established as research topics of their own, reasoning under their broader scope and cross-discipline similarities enables preference learning to be a sort of unifying field, bringing closer topics that used to be scattered between various research areas. Although relatively new, the field of preference learning is quickly gaining recognition. This is in large part due to the efforts of Johannes Fürnkranz and Eyke Hüllermeier, whose contributions include the organization of Preference learning workshops at the ECML/PKDD conferences (2008–2010) and acting as editors for the Preference learning book [15] and a recent Special Issue on Preference Learning and ranking in the *Machine learning* journal [12]. Several new research studies of ranking have first appeared in one of these events and publications.

In this dissertation we will limit our studies to the family of ranking problems known as *instance ranking* [15], where the inferred models need to rank data instances. Another, very different but perhaps more popular task is *label ranking* [16]. There the goal is to rank a set of labels that can be associated with a single data instance. While we consider only instance ranking problems in our analyses, some ideas from other ranking problems can be generalized and also used in our work. For example, an analysis of binary decomposition used for label ranking [17] is very similar to our use of binary decomposition in instance ranking, which allows some of its

conclusions to be transferred to our problems.

We will focus on multipartite ranking problems (defined in the next chapter), where several data instances can belong to the same rank class [15]. Multipartite ranking has only recently been established as a variant of instance ranking and few publications address it specifically [8, 18]. However, many general instance ranking methods can also be used for multipartite ranking, and research of equivalent problems has been performed before, under different names. The term *ordinal regression* [19, 20] can be used almost interchangeably with multipartite instance ranking. Here, ranks of instances from the training data can be considered as ordinal labels, and instance ordering can be imposed through predicting its value to the new data instances. Ordinal regression bears similarity with standard classification and regression as it is concerned with predicting a class value. Its objective is to take advantage of the fact that the values are ordinal instead of categorical (classification) or continuous (regression). Instance ranking, on the other hand, would not need to predict the values, but is only required to rank the data instances. However, to rank the data instances, instance ranking algorithms nonetheless most often predict some intermediate values, making the distinction between ranking and predicting ordinal values less pronounced. In practice, and as we report in the next chapter, a number of approaches from machine learning can be successfully used for solving both ordinal regression as well as instance ranking problems.

Binary decomposition is a technique that can be used to split a complex learning problem into easier binary classification tasks. It is one of the approaches that ranking methods can be based on and also features prominently in our research and proposals. Much related work for it is available from its more widely known use in multiclass classification, which is very well reviewed by Lorena et al. [21]. Most findings from the multiclass setting are also directly applicable to multipartite ranking, because the latter can be considered as a multiclass problem with additional ordinal structure of the classes. Several ranking-specific methods were also proposed that make use of the added ordinal structure [22, 8, 18].

Multitask learning has been utilised in the area of ranking before. In fact, Caruana, who was one of the first to use the term multitask learning, described the application of multitask learning to improve the accuracy of

RankProp, a ranking artificial neural network method [23, 24]. There, the improvement was achieved by simultaneously training a neural network to predict other outputs in addition to the target ranks. A newer approach uses multitask learning to optimize the preference learning process by using data from multiple subjects [25]. Both of these approaches use multitask learning in a straightforward way — on multiple tasks that are given prior to learning. Ranking simply replaces another supervised learning problem that the tasks might have contained.

We incorporate multitask learning into ranking in a novel way, by first decomposing a single ranking problem into several tasks and providing the tasks to the multitask learning method, which can improve the inference of task models. To the best of our knowledge, no previous work has used multitask learning to enhance a ranking approach based on binary decomposition.

Chapter 3

Ranking

Ranking is a subfield of supervised machine learning. Unlike standard prediction approaches, such as the prototypical binary classification and regression, the problem of ranking is not concerned with prediction of class values. Instead, it's goal is to infer a predictive model from ranked training data and use it to rank a new set of data instances. This task is also referred to as *instance ranking* and is only one of many types of ranking problems [15] considered by machine learning. While different versions of ranking problems share similarities in both definition of the problem and its solutions, we will focus on methods for instance ranking in the rest of this dissertation. Unless explicitly stated otherwise, we will therefore use the more general term of "ranking" to refer to the more specific problem of instance ranking.

In the following sections, we describe the different types of instance ranking, define the problem, categorize methods for inference of ranking models, and introduce a ranking-specific evaluation framework. At the end of the chapter we analyse different approaches and compare their performance on synthetic data sets and in a practical application of ranking in the domain of molecular biology.

3.1 Types of instance ranking tasks

Instance ranking problems can be divided into different categories, depending on the number of *ranks* the instances can be assigned to. The straightforward way to rank n instances would assign them ranks of 1 through n.

But due to the nature of data and problem domain, some instances could be considered as equivalent or incomparable. This could, for instance, emerge by wilful repetition in experiment design. In such circumstances, ranking can use equivalence classes, where instances from the same equivalence class should not be ranked one above the other, but rather assigned the same rank.

With respect to the number of ranks associated with the training data, and the notion that ranks should be regarded as two or multiple equivalence classes, we will consider ranking problems of three distinct types: *continuous*, *multipartite* and *bipartite*.

3.1.1 Continuous ranking

This is the standard instance ranking problem without ties. The problem does not usually appear under a special name, and is most often simply referred to as an instance ranking problem. We use the term continuous ranking to distinguish it from the other two ranking types that consider equivalence classes.

The set of n data instances, regardless whether constituting a training or a test set, should be assigned ranks of 1 through n such that for no pair of instances the rank is shared. Models that predict a class value or a utility score, from which the final ranking is obtained, typically use a large (possibly infinite) number of different values (*e.g.* integers $\{1, 2, \ldots, n\}$ or real numbers). Figure 3.1 illustrates an example of data for this problem.

Figure 3.1: Example of continuous ranking data.

3.1.2 Multipartite ranking

When ties in rankings are present in the data and allowed in the model, the problem is known as multipartite ranking [15]. Data instances are associated with one of a limited number of possible ordered labels, which define

the rank of an instance. Several instances can have the same label, so a multipartite rank can also be considered as an equivalence class of incomparable instances (see example in Figure 3.2). Due to this and the correspondence to classes in classification, we will also use the term rank class to emphasize its role as a set of instances as opposed to a consecutive number giving the position in a ranking. We will use the notation $x \in R_i$ to say that instance x belongs to the rank class R_i. The rank classes are ordered (R_1 to R_k) and disjoint. The number of ranks k in a multipartite data set and the number of instances per rank are important properties that will often influence prediction performance of learning methods.

Figure 3.2: Example of multipartite ranking data.

Continuous ranking can be seen as an extreme form of multipartite ranking where each instance is in its own equivalence class. It is also possible to transform continuous ranking data to multipartite ranking data using discretization — for example by merging several consecutive instances into a larger equivalence class.

3.1.3 Bipartite ranking

The other extreme of multipartite ranking is when all instances belong to only two equivalence classes, usually labelled as 0 and 1 or -1 and $+1$ and called positive and negative (see example in Figure 3.3). The problem of ranking instances in such a way that the positive instances are ranked above the negative ones is called bipartite ranking. This problem has received a lot of attention by researchers from the field of classification as it is directly related to predicting probabilities for a standard binary classification task [26, 27].

While the problem is very close to binary classification, most off-the-shelf classification algorithms focus on other goals, such as high classification accuracy, and might not produce the best ranking. Obviously, if perfect classification (correct prediction of one of the two labels) is possible, then the

Figure 3.3: Example of bipartite ranking data.

inferred model would also produce a perfect ranking. But in the absence of perfect prediction, as is usually the case, the results can be quite different. Since good ranking is often more important than the accuracy of classification, there have been many proposals that try to optimize the area under the ROC curve (AUC) — a ranking performance measure [28, 29].

3.2 Problem definition

We will focus on multipartite ranking, which allows ranking ties between data instances and provides a general enough framework to be useful for many real-world instance ranking problems. Some methods that we will introduce can be applied to continuous and bipartite ranking settings as well. While it is also possible to develop specialized continuous and bipartite ranking methods, we will limit our studies to the multipartite case.

The problem of multipartite instance ranking can be defined in several ways. Existing ranking methods proposed thus far (*e.g.*, [8, 7, 18, 11, 13]) have assumed slightly different formats of the data and output. They often also have different priorities with respect to the goals they try to achieve. However, available data can often be transformed to fit the assumptions of methods that require another variation of data format or problem definition. A trivial example would be transforming the training data from a ranked list of instances to `(instance, label)` pairs, where ranks from the original data are used as labels. For this reason, we do not wish to limit ourselves to only one standard definition of the problem. Instead we will describe the most common variations along with an overview of their differences and use cases. We will also give our preference, which will be used in this work when the circumstances allow it.

3.2.1 Input

The most direct way to represent the input data for an instance ranking problem is in the form of a *ranked list* of data instances. This is different than the standard formats used in other fields like classification and regression, where `(instance, label)` pairs are the norm. This data presentation also has some drawbacks:

- Care has to be taken to preserve the order of instances. This is especially important during preprocessing operations which often consider a random sample of the instances, or shuffle the instances arbitrarily.

- Finding out the rank of a chosen instance requires determining its position in the complete list of (possibly many) instances.

- Special rules are required to represent ties or somehow mark the rank classes in multipartite data.

An alternative that avoids these drawbacks is to specify the ranks explicitly. This is also one of the most common representations used in practice, and the one we will prefer to use when possible. The data instances are represented as vectors in feature space $x \in \mathcal{X}$ (often $\mathbb{R}^n$) and the ranking is given using *ordinal labels*. Compared to a ranked list, using this representation allows us, for example, to more easily select a random sample of the data. For ease of use, the labels are often allowed to take on arbitrary real values, instead of integers 1 through k (representing ranks). The real values however are usually not used directly, but are only used to define the ranked list of instances and compute the integer ranks (*e.g.* as in Joachims [7]).

Instead of associating labels with each instance, another approach also uses vectors to describe data instances, but gives the preferences in the form of ordered *instance pairs*. These indicate that the first instance of the pair should be preferred (*i.e.* ranked higher) over the second. This format is especially well suited to provide a partial ranking or an inconsistent one (where transitivity does not hold). It could be used to provide a complete ranking as well, but the data format with rank labels can achieve that in linear space complexity instead of a quadratic one necessary to list all pairs.

3.2.2 Output

After learning from training data we infer a predictive function (model) to rank new data instances. Most ranking methods do not provide a function that directly maps a set of data instances to an ordered list of ranked instances. Instead, the prediction models can implement a function to predict an intermediary result for the given set of instances, from which the final ranking can be obtained. The two main approaches for such indirect ranking are:

Utility score function. The predictive model encodes a utility score function, which predicts a numerical *utility* value for a given data instance. Assigned utility scores can then easily be used to rank a set of instances, where the rank is lower with the lower value of utility.

Binary preference function. Another option is to use a binary preference function, which considers two data instances and predicts which one of them should be ranked higher (preferred). While this makes it easier to model some patterns in the data, it also requires more elaborate subsequent processing to obtain a ranked list of instances. The final ranking may be, for instance, derived from preferences elicited for all possible data instance pairs. Such aggregation is not trivial and is usually an important part of the proposed ranking algorithm. Another drawback of this approach is its quadratic time complexity, or possibly lower quality in case not all predictions are computed. Sometimes a complete ranking is not needed, and only two instances need to be compared at a time. A binary preference function is then very well suited for such tasks.

There is a trivial way to transform a utility function into a binary one by comparing the utilities of two given instances. However the added benefits and increased expressive power of binary functions are lost in this case. For example, it is not possible to model an intransitive function.

Converting a binary function into a utility function can be done in different ways and the procedure is typically not trivial. An example is counting the number of pairs where an instance is preferred over the alternative. More generally, any type of aggregation proposed with a ranking algorithm can be

used to rank data instances based on the outputs of the binary preference function. After that, the ranks can be considered as the utility scores.

For this reason selecting the type of function for a method is an important design decision. It has a major influence on the characteristics of the whole method. That is why it is also often used as the first criterion for grouping different approaches into families (see section 3.3).

3.2.3 The goal

There are also differences in what ranking methods try to achieve. A simple description of the goal is to learn a model that can rank new instances. But there are many details that can fundamentally change the problem and influence the appropriate choice of methods for solving them.

The first distinction is whether we seek to obtain a *complete* or *partial* ranking. A complete ranking ranks all data instances, and enables the comparison of any pair of instances, of which one must be ranked higher. A partial ranking [30] can provide preferences only for some pairs of instances while some are incomparable.

Another goal choice depends on the performance measure for evaluating rankings. Among many different measures, their most distinctive difference is if they consider *uniform* or *localized* importance of ranking. The former seeks to obtain an overall good ranking across the entire test data set, with as few mistakes as possible at all ranking positions. The latter places greater emphasis on a specific group of instances, and would typically favour the ranking accuracy within the higher-ranked data instances. The measures themselves will be described in greater detail in section 3.4.2.

3.3 Inference of ranking models

We consider several different approaches to design ranking inference methods, and will split these approaches into families exposing three design principles: *utility score*, *pairwise ranking*, *binary decomposition*.

3.3.1 Utility score

The first group of methods is based on learning a function for predicting a numerical utility score for each individual data instance. This can be done directly by predicting the rank of an instance. More frequently, we are not concerned with making integer predictions and instead allow arbitrary real numbers for utility scores. In a subsequent step of the algorithm, the instances are ranked according to their predicted utilities.

As the utility score approach transforms the ranking problem into one of predicting numerical values, there are several existing options of achieving this. New methods designed specifically for ranking have been proposed as well.

Regression

There already exist a multitude of methods for learning a function that can predict a dependent numerical variable from independent variables. One option is to use one of the established approaches from regression learning according to the properties of the data set (*e.g.* linear or nonlinear regression, sparse regression learners, and other). Additional processing that enables the use of standard regression learning methods for ranking includes preparing the target values (labels) that should be modelled. Sometimes the training data already comes in a format containing numerical labels. If not, they have to be inferred from the data, *e.g.* using the ranks of ranked instances as their labels.

Some examples of methods that can be used include: linear least squares, ridge and lasso regression [31], partial least squares (PLS) [31], support vector regression (SVR) [32], regression trees [33], etc.

Dimension reduction

Another set of potential methods that can be used for ranking are those for dimension reduction. Projecting data instances into a one-dimensional space allows them to be ranked based on their positions in this space. For this problem it would be possible to use both unsupervised and supervised methods. Unsupervised methods would alleviate the problem of overfitting, while supervised would be able to infer a better-informed model with potentially

better separation of instances of different ranks. For both, but especially for unsupervised methods, we expect that the most prominent changes in independent variables appear between higher and lower ranked instances.

The prime example of dimension reduction is principal component analysis (PCA) [34] along with its many modifications (weighted, supervised, etc.) [35]. Other more recently proposed methods include Pathrecon [36] and Minimum Curvilinear embedding [37].

Specialized ranking methods

Standard regression methods can be used to model and predict the ranks of instances, but because the methods were designed for regression, their application in ranking may be flawed. For example, many regression methods minimize the root mean square error (RMSE). In a ranking setting RMSE may not be the most relevant criterion, since the magnitudes of errors are not important. Thus optimizing it might not lead to a final ranking result that is as good as it could be if the utility scores were predicted with a *specialized* method designed for ranking.

Similar to bipartite ranking, where classification accuracy is replaced by a measure such as the AUC, specialized multipartite ranking methods also try to exchange standard regression measures like RMSE with those based on ranking performance. These are often based on minimizing the number of inversions (incorrectly ranked pairs of instances) or maximizing a rank correlation statistic (Spearman's ρ, Kendall's τ, etc.). Direct optimization of the selected measure is often intractable and replaced by optimizing an approximation.

The most well known example of a specialized method, which is also a standard benchmark for other methods, is SVM^{rank} [7, 38]. Depending on the definition, it can be considered a utility score method or a pairwise ranking method (more on this duality in the next section). Other specialized ranking approaches include methods proposed in Xu and Li [39], Pahikkala et al. [10], Herbrich et al. [40], Yu et al. [9], Raykar et al. [41].

3.3.2 Pairwise ranking

The second family of methods models a binary preference relation. Instead of constructing a function that predicts utility scores for individual instances, methods learn a function over pairs of data instances. For any two data instances the inferred function can predict which of the two instances should be ranked higher.

In case the ranked training data was not given as a (sub)set of preference instance pairs, an input data transformation is required. We can form ordered instance pairs from the original ranked data instances. New binary labels denote which of the two instances from a pair was ranked higher. Constructing all possible pairs of instances with different ranks results in a data set with a quadratic size complexity (up to $n(n-1)$ pairs from n original instances). For larger problems this can significantly increase the size of training data, limiting the use of this approach.

After the training data is prepared the ranking problem is effectively reduced to a binary classification task. Similarly as for the regression problem from the previous section, it is again possible to use standard classification methods or specialized techniques. In this case the main motivation for specialized methods is not so much in a different optimization goal, as correctly predicting the order of instance pairs would also maximize ranking quality. Instead, specialized methods often try to take advantage of the structure resulting from learning on pairs of instances and arrive at a more efficient learning process than merely treating the pairs of instances as independent samples from some distribution (*e.g.* SVM^{rank}).

Possibly the simplest way to prepare the data for the inference of a pairwise ranking model is to concatenate vector descriptions of a pair of data instances x_1 and x_2 into a longer vector $x_{1,2}$ that jointly describes a pair:

$$x_{1,2} = (x_1, x_2) \tag{3.1}$$

This leaves the learning algorithm to handle any interactions between features. For example, comparing the first feature of x_1 and x_2 to see which is higher requires comparing the respective features in the concatenated representation. But the information of which features are related is lost and finding patterns among all possible pairs of features is much harder. Con-

catenation also doubles the number of features.

Another option, when all features are numeric, is to use the difference of two vectors:

$$x_{1,2} = x_1 - x_2, \quad x_1, x_2 \in \mathbb{R}^m \tag{3.2}$$

This way the number of features stays the same and comparisons between the same features are trivial. Checking if the first feature of x_1 is higher than that of x_2 is the same as checking if the first feature of $x_{1,2}$ is positive. However, as only the differences are available, finding other interactions and patterns might be harder or even impossible.

One of the biggest advantages of using differences is the fact that for linear methods a function over instance pairs can be equivalently expressed as a function over original single instances. Let $p(x_1 - x_2)$ be a decision function defined over difference vectors, predicting that x_1 should be ranked higher than x_2 if and only if $p(x_1 - x_2) > 0$. Equations 3.3 through 3.6 show that the same function can also be used as a utility score function working on original instances.

$$\begin{aligned} p(x_i - x_j) &= \langle w, x_i - x_j \rangle && (3.3)\\ &= \langle w, x_i \rangle - \langle w, x_j \rangle && (3.4)\\ &= p(x_1) - p(x_j) && (3.5) \end{aligned}$$

$$p(x_i - x_j) > 0 \iff p(x_1) > p(x_j) \tag{3.6}$$

A pairwise ranking model can therefore be expressed in the form of a utility score function, although the pairwise component is usually still present during the process of learning (*e.g.* for ranking SVM in the form of optimization constraints derived from pairwise relations). Methods from Joachims [7], Herbrich et al. [40], Yu et al. [9] can therefore be considered both pairwise ranking methods or utility score based methods.

In addition to methods based on ranking SVMs, two well known examples of pairwise ranking include the methods RankBoost [42] (based on boosting) and RankNet [43] (based on neural networks).

3.3.3 Binary decomposition

The third approach to inference of ranking models starts by transforming the multipartite ranking problem into several binary classification tasks. The obtained subproblems can then be solved with existing methods for classification, and the results combined back into a solution for the original problem. A more well known application of the same principle is used to solve multiclass classification problems [44, 21]. Note that in contrast to the previous two approaches we do not infer a single predictive model that captures global patterns in the data. Instead, the extracted knowledge is distributed among many limited models, which can be easier to fit than one complex model.

Compared with pairwise ranking approaches where reduction to classification is possible because the predictive function is defined over pairs of instances, this technique remains in the domain of individual instances. Splitting the problem into a set of binary classification problems allows us to model simpler binary relations. This means the original data set is transformed into several simpler data sets, which may overlap depending on the variation of the method used.

Methods based on this principle include an early benchmark in the field of ranking by Frank and Hall [22], as well as new proposals [8, 18, 45].

Binary decomposition will be an integral part of the enhancements proposed in the next chapter and will be described in more detail there (section 4.1).

3.4 Evaluation

3.4.1 Internal validation

To estimate the performance of different methods and enable their comparison, we can use procedures similar to those established for evaluation of classification or regression problems. These include standard techniques such as learning a model from training data and testing it on separate test data. When the amount of available data is smaller, we can use n-fold cross-validation, bagging and other similar sampling-based techniques [46].

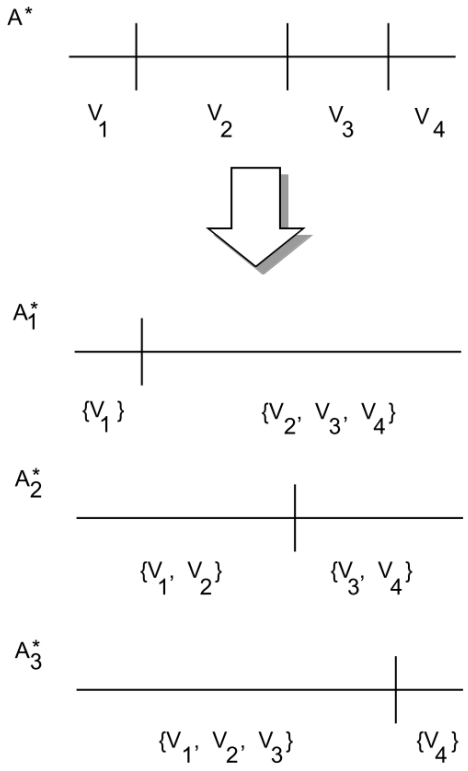


Figure 3.4: Example of binary decomposition from Frank and Hall [22]

The main difference between ranking and other problem types appears when the number of data instances is very low. For classification and regression, the leave-one-out approach uses almost all data for training by leaving only one instance aside for testing. By repeating this, we can get predictions for each instance, when it is not present in the training data, and subsequently compute a reliable prediction performance estimate.

For ranking, however, the leave-one-out approach cannot be used, as evaluating the ranking requires at least two data instances (one cannot reason on the accuracy of ranking using a single instance alone). The alternative is to use leave-pair-out (LPO) instead, and make predictions for two instances at a time [47, 48]. This procedure is compatible with performance measures based on the number of (in)correctly ordered pairs. It does require fitting the model $\binom{n}{2}$ times instead of n times for leave-one-out.

A variation of the leave-pair-out procedure that we have recently intro-

duced in experiments with data from molecular biology [49, 50] is to leave out a pair of rank classes at a time. All instances belonging to the selected two classes are set aside for testing, thus evaluating the method's ability to generalize the ranking patterns from training data to unobserved instances from new classes. This approach is more strict and can be expected to provide more conservative performance estimates, because the training instances do not cover the input space completely (no examples from selected rank classes are available). However, the increased strictness can better reflect the performance of a method when it is given training data containing certain classes and is subsequently tested on data containing instances from new unseen classes.

In our experiments the chosen procedure will depend on the amount of data available. For example when using synthetic data sets, for which we could generate a larger number of data instances, training and testing can be carried out on separate data sets. For real-world data sets, where the number of available instances could be very small, the leave-pair-out approach can provide the best performance estimations in exchange for some extra computational costs.

3.4.2 Performance measures

After a model makes a prediction and provides a ranking of a set of test instances, we wish to measure the quality of that prediction by comparing it to the true ranking. As in other fields of machine learning there are many measures that evaluate different aspects of the prediction and try to summarize the quality with a single number.

Selecting which measure to use depends on the problem and the user's priorities. In some cases, such as information retrieval, the top ranked instances need to be truly relevant, while for a different problem we might wish to minimize the number of incorrectly ordered instances in the whole ranking. Below we list some measures that have been used most often in the estimation of ranking performance.

Area under the ROC curve

Although not a multipartite ranking measure, we will start with the best known ranking measure and one of the most widely used machine learning performance measures overall. The area under the receiver operating characteristic (ROC) curve is a classification and bipartite ranking measure that has been widely used for several decades [34]. It is usually abbreviated to just area under the curve or AUC.

The first interpretation of the measure comes directly from its name. An ROC curve is a curve in two-dimensional space spanned by the false positive rate (x-axis) and the true positive rate (y-axis). Ideally a classifier should have a low false positive rate and a high true positive rate. However, most classifiers exhibit a trade-off of the two parameters — increasing one score would reduce the other. The performance of different classifiers, as characterized by the ROC curve, can not always be compared. A model can be better at lower values of the false positive rate and another model at higher values. AUC was proposed to summarize the performance with a single numerical value and obtain an overall score for predictive accuracy.

Another interpretation is that the AUC estimates the probability of a positive and negative instance being correctly ordered by the classifier. This interpretation provides a justification for why measuring the area under this specific curve has become such an important performance measure. But more importantly for the context of ranking, this interpretation also allows an intuitive generalization of the measure from two class values (bipartite) to $k > 2$ (multipartite).

Concordance index

Concordance index (also c-index) [19] is a generalization of AUC from a bipartite to a continuous or multipartite ranking measure. It measures the fraction of concordant pairs.

For continuous ranking, where all instances can be compared (there are no ties), the measure is easy to define. Looking at all possible pairs, one only needs to count the number of concordant (correctly ordered) instances and compute their fraction. Similar to the interpretation of AUC, this gives us an estimation of the probability that the classifier will correctly order a

pair of instances.

The measure can also be used for multipartite ranking, where some pairs of instances can be tied and are therefore incomparable. Taking this into account the fraction should only be computed over comparable pairs. Equation 3.7 describes the final computation.

$$C = \frac{\sum_{x \in R_i, y \in R_j, i<j} \delta(p(x) < p(y))}{\sum_{i<j} |R_i| \times |R_j|}, \tag{3.7}$$

where rank class R_i is the set of instances ranked i-th, $p(x)$ a prediction for instance x (this could be the predicted rank or more indirectly the predicted utility score), $|R_i|$ the size of R_i and $\delta(cond)$ equal to 0 or 1 if $cond$ is $False$ or $True$, respectively.

Observing how the index is computed, it can be noticed that it is possible to iterate over pairs of instances from different ranks two at a time and count how many are correctly ordered. This means using the concordance index as a performance measure and leave-pair-out cross-validation technique can be effectively combined into a complete evaluation procedure.

Rank correlation statistics

For comparing the quality of a predicted ranking compared to the true ranking, it is also possible to use any of the rank correlation statistics. The two most well know examples are Spearman's ρ coefficient and Kendall's τ coefficient.

Spearman's ρ is computed in the same way as the standard Pearson's correlation coefficient. The only difference is the values are first replaced by ranks.

Kendall's τ is similar to the concordance index in that it is defined using the number of concordant and discordant pairs. However, as it is a correlation coefficient it is defined in the range $[-1, 1]$ (as opposed to the C score's range of $[0, 1]$). It is computed as the difference between the number of concordant and discordant pairs, normalized by the number of all possible pairs.

Localized measures

In this thesis, we will focus on the general multipartite instance ranking problem and treat all instances equally. This means the family of performance measures developed and used in information retrieval that place more emphasis on higher ranked instances will not be as relevant. For the sake of completeness, the most popular ones are listed here, as they frequently appear in research papers describing methods with applications outside the scope of information retrieval as well.

- *Precision-at-k* is defined for the bipartite case (relevant and non-relevant instances) and is computed as the fraction of relevant instances among the top k ranked instances.

- *Average precision* is the weighted average of precision values at all possible k. That is, it is the area under the precision-recall curve. It is most often used in document retrieval, where the average precision can be computed for multiple queries. The mean average precision (MAP) is defined as the mean of average precision values over all queries.

- *Discounted cumulative gain* (DCG) [6] can be used for multipartite and continuous ranking as well. In terms of information retrieval this means the documents can have different degrees of relevance instead of being considered as just relevant or not. It is computed as a weighted sum of relevance values, with the instances lower in the ranking penalized by logarithmically increasing weights.

3.5 Comparative analysis

Here we discuss differences between utility-based and pairwise ranking approaches. Binary decomposition techniques will be analysed in the next chapter. Comparing models for predicting numerical values is not in the scope of this work. Different methods have their individual advantages and disadvantages and should be selected based on the specific properties of the problem to be solved. Instead, we look at the properties of ranking problems and how they can influence the appropriate choice of method.

3.5.1 Choosing appropriate utility values

To be successful, utility score ranking methods need to learn an accurate model, which can predict utility values relevant to the ranking problem. Before the learning can start, appropriate utility values that relate to ranking need to be assigned to the training data. If these are not already provided, the construction of utility values may be a challenge.

Even if the data does not include utility values, there is always an option to use the ranks as utilities. Sometimes expert domain knowledge can help in constructing better utility values. When this is not possible, using ranks is the simplest solution and can also be very effective.

When the utility values are already part of the training data, they present an obvious choice for the dependent variable. However, these values might have been only estimates or measurements of a parameter related to the true utilities, but not exactly the same.

As an example, imagine ranking circles by their radius. If the area was measured instead, no information is lost. But if there is a linear relationship between the independent variables and the radius it will be harder to model the area since a nonlinear (quadratic) transformation would be required. Consequently a linear regression model would have a lower performance than had it used the radius for utilities.

A simple experiment on synthetic data sets shows how the choice of utility values can impact the performance. We tested the predictive accuracy of methods when modelling the ranks of instances or when the provided utility values are modelled instead. Our hypothesis was that to maximize performance the more suitable target has to be chosen depending on the data and its true underlying model. One approach can not be preferred over the other a priori.

We generated two multipartite ranking data sets. The description of data instances included $m = 100$ independent variables, with values generated randomly from a uniform distribution over $[0, 1)$. The true utility values were computed as a linear combination of independent variables. In the first data set (`linear-ind`) these true utility values were integer indices from 1 to k. In the second data set (`linear-sq`) the squares of indices (1, 4, ..., k^2) were used. Each of the training and test data sets consisted of $k = 10$

different rank classes with 10 instances per rank. Two linear ridge regression methods were evaluated: one tried to model the indices (ridge-ind) and one that modelled the squares of indices (ridge-sq). Each method was therefore predicting the correct utility values, as defined by the data generation model, for one data set, and only a related variable for the other data set.

Concordance index (C) and Spearman's ρ correlation coefficient were computed and averaged over 100 repetitions. Results are reported together with the next experiment in Table 3.1. A big difference can be seen in the performance of linear regression, when used to predict the correct utility values or only a related variable. This confirms our hypothesis and shows that in practice, when it is unknown whether provided utilities or indices better correspond to the true model, both should be considered. Using internal validation, the one that works better for the given problem can be selected.

3.5.2 Specialized ranking approaches

To continue from the previous experiment, what if the true utilities can not be used? What if the underlying model does not correspond well to either the given utility values nor to the ranks? This is a very plausible situation and just choosing the better of those two options might not be good enough. This is one motivation for using specialized ranking approaches and a possible reason for why they could outperform general regression algorithms.

Regression functions try to build a model that will accurately predict the dependent variable from the independent ones. Specialized ranking methods can take advantage of the fact that the only purpose of the model they are fitting is to rank the instances, so they can allow bigger discrepancies between the predicted and true values, as long as the values perform well for ranking the instances.

As an experiment, we tested the performance of SVM^{rank}, a benchmark specialized ranking method, to see if it can perform better when the given utility values are not the ones that generated the data. The same two data sets were used as above, but we intentionally provided the "wrong" utility values to the learning algorithm (the squared values for `linear-ind`, and the indices for `linear-sq`).

Table 3.1: Evaluation of the impact the choice of utility values being predicted has on prediction results.

	`linear-ind`		`linear-sq`	
method	C	ρ	C	ρ
ridge-ind	0.91	0.92	0.80	0.73
ridge-sq	0.81	0.77	0.98	0.98
SVM^{rank}	0.85	0.83	0.79	0.72

The results are shown in Table 3.1, together with the results of the previous experiment. SVM^{rank} did indeed perform better on `linear-ind` than linear regression with the wrong utilities. However, on `linear-sq` it could not improve the results of linear regression predicting indices.

3.5.3 Pairwise ranking

To show the advantages of optimization targeted at ranking we performed another experiment with a different model for generating data. In the previous experiments we could use the same model (linear regression) for the learning method as was used to generate the data. In real settings, the data generation model is usually more complex and unknown. Under such conditions, pairwise ranking approaches could have an advantage in building a good ranking model compared with methods that directly model the utility values.

Instead of using a linear combination for predicting utilities, a more general data generation approach was used for this experiment. Instances of each rank class were sampled from a distinct multivariate normal distribution. The complete data set thus consisted of several Gaussian clusters, randomly distributed in m-dimensional space. Instances were ranked based on their cluster membership, where the same rank was assigned to the data instances of the same cluster. Clusters were defined by randomly sampling their centres $\mu_{\mathbf{i}} \in \mathbb{R}^m$ from a multivariate normal distribution $N(\mathbf{0}, I)$, where $\mathbf{0}$ is an m-dimensional vector of zeros, and I an $m \times m$ identity matrix. Instances of cluster i were then sampled from $N(\mu_{\mathbf{i}}, I)$.

The training and test data contained $k = 10$ different ranks and 10 instances per rank. The number of independent variables was $m = 10$ for

Table 3.2: Comparison of ridge regression and SVM^{rank} on multipartite data with randomly distributed rank classes.

	`normal-10`		`normal-100`	
method	C	ρ	C	ρ
ridge-ind	0.61	0.31	0.69	0.49
SVM^{rank}	0.72	0.57	0.92	0.93

data set `normal-10`, and $m = 100$ for `normal-100`. The latter should allow higher quality predictions, since separating the instances in a higher dimensional space is easier. The randomized data generation process was repeated 100 times, and we report the average Concordance index and Spearman's ρ coefficient.

Results (Table 3.2) show that even though the data model did not correspond to either method's linear model, both methods found an approximation that worked quite well. The pairwise-based optimization of SVM^{rank} was, however, noticeably more successful. The difference was even more pronounced on `normal-100`, where a very good approximation with a linear model was possible (and found by SVM^{rank}) due to the high dimensionality of the space.

3.6 Case study: embryonic stem cell development

We here describe a practical use case of rank learning that demonstrates how a recent research problem from the field of bioinformatics can be posed in the form of multipartite ranking, making use of the methodology described in this chapter. This work has been previously published in the *Bioinformatics* journal [49] and presented at a conference [50].

3.6.1 Problem description

Embryonic stem cells (ESCs) and other pluripotent cell types are increasingly being studied for their potential therapeutic use in regenerative medicine [51]. ESCs are isolated from the inner cell mass of the blastocyst, they

replicate indefinitely, maintaining pluripotent characteristics and may differentiate in vitro to most of the somatic cell types present in the adult. While the stages of ESC differentiation into a specific cell type have been broadly identified, numerous aspects of this process remain unknown or difficult to interpret. Differentiation is a complex, multiple-step process that presents a nonlinear progression within a cell population. The stem status is not lost immediately, but it gradually decreases. This is true particularly at the very beginning when differentiation is induced (*e.g.*, either by an internal or external signal) and cells own a heterogeneous status of differentiation being a mixture of diverse developmental stages.

ESCs as well as embryonic carcinoma and induced pluripotent stem cells [52] own common and specific molecular signatures that define their pluripotent status. When differentiation is induced, this molecular signature is gradually lost in favour of one that defines a more differentiated type of cellular identity. Novershtern et al. [53] demonstrated that this cellular transition is due to a large number of transcription factors whose expression changes across different hematopoietic states. Other recent studies of various developmental processes have shown that they are governed by transcriptional programs in which genes are regulated in successive waves of transcriptions that mark the stages of differentiation [54, 55, 51, 56, 37, 57]. Thus, cell's transcriptional profiles could be used as whole-genome markers of differentiation.

We will use models that, given the transcription profile of a cell, predict its differentiation stage. Differentiation is a continuous process, and for interpretation it could be convenient if the model would map whole-genome transcription profiles to a one-dimensional projection. In this thesis, we refer to this projection as a *differentiation scale*, and evaluate it on the basis of preservation of the order of data points with respect to the staging of differentiation. Projection of differentiation landmarks on this scale may further expose the dynamics of the observed process. To this end, we investigate the utility of various state-of-the-art data transformation approaches and their predictive accuracy in a systematic evaluation on 14 publicly available cell differentiation data sets from mouse, rat and human.

3.6.2 Methods

Let us consider a data set where the genome-wide expression profile has been observed for n different samples (data instances) along differentiation. Each sample is therefore represented with expression of thousands of genes. To infer a stage prediction model, we select the most informative genes and use data projection methods that combine the selected gene expression values and project sample profiles onto a one-dimensional ruler — a differentiation scale. Labels on the scale indicate the time at which the gene expression was measured and should, in standard culturing conditions, reflect the stages of cellular differentiation. Due to experimental noise and the variability of expression, samples from the same stage are not projected to the same point on the differentiation scale. To characterize the stage rather than the individual samples, these projections are fused into a single median position on the scale. The predictive model and its associated visualization through the differentiation scale can then predict the developmental stage of a sample coming from an experiment where different culturing conditions may have perturbed the differentiation process, and where the actual stage has yet to be determined (Figure 3.5).

In the following, we describe various data mining approaches we have considered for inference of predictive models. The transcriptome data typically includes measurements of a large number of genes for a small number of samples that were observed at different developmental stages. The inference starts with selection of the most informative genes, that is, those that could be best used for the characterization of staging. Samples described with the selection of genes are then projected to the differentiation scale via unsupervised data mining methods or by additionally considering the stage information from the training data. We use a variant of leave-pair-out testing and a concordance index score to test and compare various model inference approaches.

Gene subset selection from time-sequence expression data

Our aim here is to reduce the computational costs of the inference and exclude from further analysis genes whose expression is either too constant or too irregular across the observed set of stages. A large variety of gene sub-

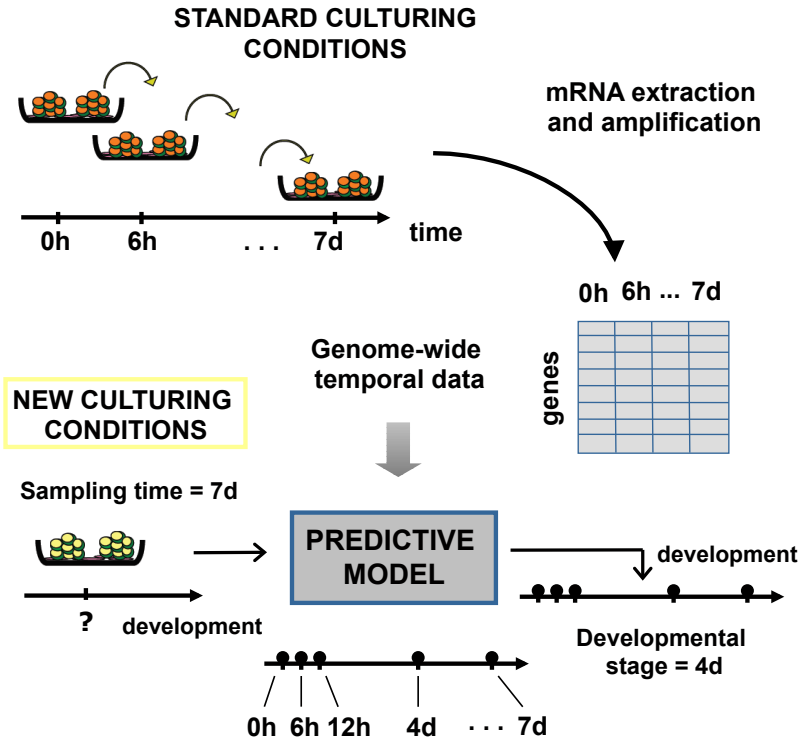


Figure 3.5: Differentiation stage prediction models and their application. A prediction model is inferred from microarray data from experiments in standard culture conditions. The model can predict the cells developmental stage when the culture conditions are changed. The illustration shows the projection of a sample that, despite having been collected after seven days of differentiation, has a molecular identity of four days along the scale of standard differentiation.

set selection methods have been proposed for case-control studies [58], but there are considerably fewer approaches for studies of expression dynamics. The distinguishing feature of such data sets is a low number of replicates observed at each of the developmental stages, which invalidates the utility of several standard statistical approaches, like those based on the analysis of variance [59, 60, 58].

We experimented with two different gene subset selection approaches, one that considers time (staging) and the other that ignores the series of events and treats stages as separate, unrelated experiments:

- AREA: As proposed by Di Camillo et al. [61], the area between the expression profile of a gene and its control profile was computed. The control profile was constant, equal to the expression at the first time point.

- FC: For each gene and each time point the fold change with respect to the control condition is computed [62, 63]. Gene expression at the initial time point was used as a control. Gene's score is defined as the number of time-points where expression change is at least two-fold.

The two scoring methods were used to select 1000 best-ranked genes whose expression was then considered for inference of predictive models. The performance of prediction methods dropped when a smaller number of informative genes was considered (see Appendix A.1).

In order to remove the systematic bias introduced by experiments, we used quantile normalization as described by Bolstad et al. [64] prior to gene subset selection.

As a knowledge-based alternative to data-driven gene subset selection, we also identified a small set of *differentiation markers*, whose transcriptional signature could be considered for the inference of predictive models. These were obtained from the Mouse Genome Informatics (MGI) repository and Gene Ontology by retrieving genes associated to stem cell differentiation (see `http://www.biolab.si/supp/stagerank`). Our experimental data sets included from 20 to 60 marker genes, and in this part of the experiments, only these genes were used for inference of stage prediction models.

Inference of prediction models

Let us assume we have a training data set represented with a $n \times m$ matrix X, where expression of m genes has been observed for n different samples. Let each sample be labelled by the development stage at which the measurements were performed. These can be placed in a column vector Y of size n. Our typical data set would contain about 5 to 12 different development stages and around 10 to 50 samples (typically, 3 or fewer samples per stage). Each sample would typically be represented with expression of 5,000 to 25,000 genes, from which we select 1,000 most informative ones using gene subset selection methods. Our goal is to project the samples to a 1-dimensional space, that is, place each sample on a line. Supervised or unsupervised methods may be used to address this problem — with or without the knowledge of development stages (Y).

Unsupervised methods reduce the dimensionality of the data without considering the sample labels (stages). *Principal component analysis* (PCA) is one of the best known methods from this category. It linearly projects samples into a low-dimensional space that explains the highest degree of variance in the original data. We used the first principal component to project the samples into a single dimension. The projection of a sample x is computed as $p(x) = xV$, where the column vector V contains the first eigenvector of the covariance matrix $X^T X$, for mean-centred data X.

As an alternative unsupervised method, we also considered Pathrecon [36]. Pathrecon starts by constructing a complete weighted graph with samples as nodes and their expression profile-based distances as edge weights. Then it finds a minimum spanning tree that connects all the nodes. The longest path in the tree is called the diameter path. Similarly to PCA's principal direction, diameter path orders the samples (nodes), but unlike PCA — and to the possible advantage of Pathrecon — the ordering is not constrained to a linear projection. Samples contained in the branches off the diameter path are assigned the same ordering index as the diameter path element to which they connect. If long off-diameter branches exist, a data structure called PQ-tree is used to summarize the uncertainties of path variations. Pathrecon traverses the PQ-tree to find candidate orderings, and ranks them by the distance of the path they describe.

Another approach we considered is Minimum Curvilinear Embedding (MCE), a nonlinear dimension reduction method proposed by Cannistraci et al. [37]. The first step of the method computes a distance matrix (MC-matrix) containing pairwise distances between data instances calculated over the minimum spanning tree (MST). The MST is constructed by the Kruskal method and can use either the Euclidean distances or correlation distances, defined as $1-corr_{Pearson}(x_i, x_j)$. The authors report that in general both distances provide comparable results. The second step performs the dimension reduction by embedding high-dimensional data points into a lower dimensional space using multidimensional scaling (MDS). The MC-matrix from the first step is used as the input for MDS.

Supervised methods use additional information on sample labels (Y). Successive labels can be represented with continuous (real) values, allowing regression methods to be used. These can map a transcription profile to a real value, in this way projecting the sample to an already defined differentiation scale. We aim to find the projection that best separates the different development stages.

Since we have many more genes than samples ($n \ll m$), it is very easy to obtain a good separation and overfit the training data. *Partial least squares* (PLS) regression is known to work well even in such situations [65]. It is also closely related to PCA and hence provides a good supervised counterpart. The particular variant of PLS we used is commonly referred to as PLS1 [66], since the outcome matrix has only one column. In short, PLS1 first obtains a low-dimensional representation of X by projecting it to a small number of latent variables. Then it models Y as a linear combination of the latent variables. Computing the prediction for a new sample is done the same way: the values of the latent variables are calculated first and their weighted sum gives us the predicted result.

We tested two choices for real-valued labels. One was to use the time (in hours) at which the samples were measured. For the other, the consecutive developmental stages were represented with indices (e.g. $0, 1, 2, \dots$).

Ranking SVM [7] was chosen as a representative of specialized ranking methods. It tries to find a ranking function that maximizes Kendall's τ or, equivalently, minimizes the number of discordant pairs. Although this is NP-hard it can be approximated with a slight modification of the optimization

problem. Introducing (non-negative) slack variables $\xi_{i,j}$ allows us to enforce all constraints and minimize $\sum \xi_{i,j}$ like in classification SVMs resulting in the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & V(w,\xi) = \frac{1}{2}\langle w, w\rangle + C\sum \xi_{i,j} \\ \text{subject to} \quad & \forall\,(x_i, x_j) \in r^* : \langle w, x_i\rangle \geq \langle w, x_j\rangle + 1 - \xi_{i,j} \\ & \forall i \forall j : \xi_{i,j} \geq 0. \end{aligned} \tag{3.8}$$

C is a parameter that allows trading-off margin size against training error and the notation $(x_i, x_j) \in r^*$ is used to give the ranking in the form of instance pairs (*i.e.* instance x_i is ranked higher than x_j). As in regular SVMs, nonlinear models can be built by using kernels instead of standard linear scalar products. We used the freely available implementation SVM^{rank} [38], which is very efficient when using a linear kernel.

Evaluation and model scoring

We have experimentally compared various techniques for construction of stage prediction models. We used a number of gene expression data sets for testing, and performed evaluation either within the same data set (internal validation), or developed a model on one and tested the predictions on a different data set (external validation).

For internal validation we use a variant of the *leave-pair-out* (LPO) approach described in section 3.4.1. The procedure removes all samples of two developmental stages to obtain the training data, performs gene selection on the training data and then infers a prediction model. Finally, it tests the model on the samples from the two stages that were left out. This procedure is repeated for all different stage pairs. Notice that our implementation of leave-pair-out differs from the standard one which would leave out the samples regardless of their stages. Our concern here was that while retaining several samples from the specific stage in the training data, prediction of samples from that stage in the test set would have an advantage due to the potentially high similarity of same-stage samples. Staged leave-pair-out is thus more stringent, and in this respect even pessimistic: in real applications models may be presented with samples that belong to the stage that was also described in the training data.

For external validation, the prediction model is first developed on a selected training set. The model is then used to order the samples in the second (external) test set, where the quality of predictions is scored.

Pathrecon and MCE rank the samples, but do not explicitly provide a model for ranking new samples. To enable stage prediction, we included both training and test samples in the input data, and determined the staging for the test samples from the obtained ordering.

For scoring the quality of predictions we use the concordance index (C score) following its definition in section 3.4.2. Computing the C score works well in combination with the LPO cross-validation since we only need to check results for a pair of samples at a time. We also get good, unbiased score estimates even when evaluation is done on smaller data sets [67].

Predictive accuracies are estimated on different datasets. To compare the performance of multiple methods on multiple datasets, we use the Friedman test as described in Demšar [68]. The methods are ranked for each dataset separately and their average ranks compared. We use the Nemenyi test as a post hoc test to compute the required difference in ranks of two methods for their performance to be considered significantly different. A critical difference graph [68] is used for a visual depiction of the results.

3.6.3 Experimental analysis

We have evaluated different combinations of three gene selection methods (data-driven: FC and AREA, and marker-based: Markers) and four modelling techniques (PCA, PLS, SVM^{rank}, MCE). In addition, PLS used either the actual time values (*time* in the name of the method) or indices. Pathrecon was run with authors' own implementation [36] on entire data sets. We here report MCE with Euclidean distance as it performed better than correlation-based distance. Also, only PCA is reported in combination with marker-genes. Entire set of experimental results with all possible combinations of gene selection, modelling methods, and distances for MCE can be found in Appendix A. Methods were tested on data sets from Gene Expression Omnibus (GEO, `http://www.ncbi.nlm.nih.gov/geo`).

Data

Several data sets deposited in GEO focus on complex biological processes evolving over time, such as disease progression, development and cell differentiation, and thus provide the gene expression time series which could benefit from the construction of development stage prediction models. From a larger collection of such data sets, we have considered only those with at least six time points (ranks/stages) and with at least three samples for each stage. We foresee that one of the most promising applications of our work is the prediction of developmental potency of ESCs. We were thus more interested in experiments on cell development than in studies in which the behaviour of cells under different treatments or in different disease states is analysed over time. For this reason, we did not consider data on case-control studies but have analysed only time series experiments of different organisms.

Ten data sets from different species meet these criteria and were chosen for our evaluation (GDS2666, GDS2667, GDS2668, GDS2669, GDS2671, GDS2672, GDS586, GDS587, GDS2431, GDS2688). Most of these data sets study the differentiation of mouse ESCs. In particular, the first six have been collected by Hailesellasse Sene et al. [69] to study 11 stages of differentiation into embryoid bodies for three biologically equivalent but genetically distinct mouse ESC lines. The compatibility in the type of experiment and microarray data of these six data sets allowed to carry out external validation, that is, assess the predictive models trained from one data set through the quality of predictions on another data set. Data sets GDS586 and GDS587 analyse gene expression in a 12-day time course of mouse differentiating myoblasts. The last two data sets included in our analysis contain human and rat data, respectively. In GDS2431 the authors monitor gene expression in developing human erythroid progenitors, while for data set GDS2688 they analyse the temporal response of skeletal muscles to corticosteroid exposure in rats for up to 7 days. Although the aim of the latter study was different from the other cell differentiation data sets, it also had enough time points and replicates and we decided to include it for comparison.

In a separate experiment, we used the data sets from the study by Aiba et al. [70] (GSE11523). From their collection of samples, we selected four

cell lines (N, Z, G, F) that included at least three stages of cellular differentiation. The authors have already shown that the samples from the same cell lines project nicely and consistently in three dimensional space, and that the trajectory could qualitatively indicate the developmental potency of mouse ESCs. We adopt their data in order to quantitatively and systematically assess the quality of such predictions. In their original study Aiba et al. also show that the principal component-projected trajectories diverge for different cell lines when visualized in three dimensions. We were still interested if, despite this divergence, the predictive models developed on one cell-line maintain their stage prediction quality when predicting on the data from other cell lines.

Assessment of predictive accuracy

We report the C scores for different internal and external validations. Table 3.3 summarizes results of the internal validation on the Gene Expression Omnibus data sets. For each data set, we ranked the methods according to the achieved C score, and then report the average rank. Data set GDS2688 was not included in these averages as Pathrecon's score for it could not be computed in a reasonable amount of time (one day). The score for GDS2688 when using known markers instead of gene selection is not given, since the stem cell differentiation markers are not relevant for the process studied in this data set. The statistical analysis of rank differences using the Friedman-Nemenyi test [68] is presented in Figure 3.6.

Results of external validation for six selected data sets are summarized in Table 3.4. Due to the insignificant differences in performance of different best-ranked methods we have here only used PCA for development of predictive models.

Similar analysis was also performed on data sets from Aiba et al. [70]. Again, for the internal validation, there were no significant differences in performance of various best-ranked methods considered ($p < 0.05$). For brevity, Table 3.5 compares only the scores for the six best-ranked methods from Table 3.3. Results of external validation are given in Table 3.6. As before, only the performance of the PCA-inferred model is reported.

Table 3.3: C scores of leave-pair-out internal validation on ten data sets from GEO (the top score for each data set is in bold). For each data set the methods are ranked according to the data set-specific C score. Methods' average ranks and average C scores are also reported.

	2431	2666	2667	2668	2669	2671
MCE-euclid-FC	**0.993**	0.972	**0.964**	0.897	0.895	**0.964**
PCA-FC	0.874	**0.974**	0.899	**0.931**	**0.909**	0.822
PLS-AREA	0.867	0.945	0.913	0.923	0.903	0.909
PCA-AREA	0.896	0.952	0.921	0.929	0.889	0.824
MCE-euclid-AREA	0.941	0.966	0.941	0.901	0.877	0.911
PLS-FC	0.859	0.962	0.883	0.905	**0.909**	0.911
SVM^{rank}-FC	0.844	0.915	0.907	0.889	0.883	0.893
SVM^{rank}-AREA	0.859	0.883	0.881	0.893	0.905	0.859
PLS-FC-time	0.859	0.966	0.786	0.766	0.903	0.871
Pathrecon	0.956	0.840	0.887	0.859	0.812	0.919
PLS-AREA-time	0.867	0.952	0.766	0.760	0.798	0.863
PCA-Markers	0.600	0.911	0.869	0.877	0.842	0.887

	2672	2688	586	587	$\overline{C}$	$\overline{rank}$
MCE-euclid-FC	**0.939**	**0.750**	0.853	0.825	0.922	4.000
PCA-FC	0.794	0.732	0.948	**0.942**	0.899	4.556
PLS-AREA	0.812	0.581	0.944	0.884	0.900	5.278
PCA-AREA	0.798	0.738	0.944	0.937	0.899	5.556
MCE-euclid-AREA	0.828	0.728	0.817	0.820	0.889	5.667
PLS-FC	0.764	0.588	0.948	0.857	0.889	6.000
SVM^{rank}-FC	0.897	0.551	**0.972**	0.857	0.895	6.278
SVM^{rank}-AREA	0.913	0.542	0.964	0.862	0.891	6.333
PLS-FC-time	0.782	0.423	0.960	0.815	0.856	7.778
Pathrecon	0.784	N/A	0.897	0.804	0.862	8.111
PLS-AREA-time	0.842	0.392	0.952	0.841	0.849	8.111
PCA-Markers	0.776	N/A	0.730	0.519	0.779	10.333

Table 3.4: C scores for the PCA-AREA inferred models developed on a training set (row label) and tested on an independent test set (column label). Labels in superscripts of the scores denote the relationship between the two data sets: ^{a} same cell line, different platform; ^{b} different cell line, same platform; ^{c} different cell line, different platform.

	2666	2667	2668	2669	2671	2672
GDS2666	/	0.915^{a}	0.939^{b}	0.901^{b}	0.840^{c}	0.818^{c}
GDS2667	0.947^{a}	/	0.949^{c}	0.909^{b}	0.869^{c}	0.857^{b}
GDS2668	0.980^{b}	0.891^{c}	/	0.893^{a}	0.770^{b}	0.804^{c}
GDS2669	0.941^{c}	0.954^{b}	0.941^{a}	/	0.828^{c}	0.830^{b}
GDS2671	0.958^{b}	0.921^{c}	0.954^{b}	0.875^{c}	/	0.711^{a}
GDS2672	0.941^{c}	0.960^{b}	0.935^{c}	0.909^{b}	0.840^{a}	/

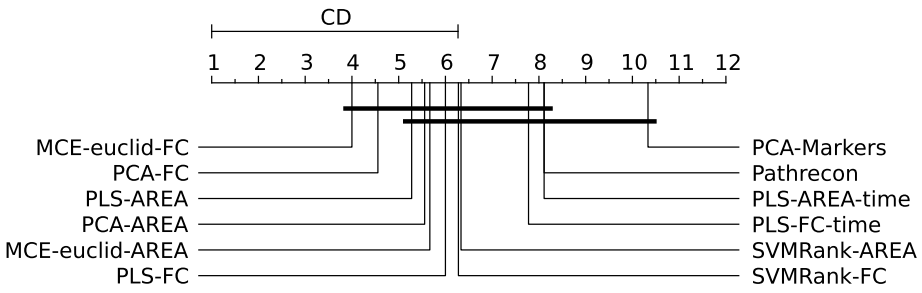


Figure 3.6: Critical difference graph for methods' average ranks (Table 3.3). Critical difference (CD) indicates the difference in ranks that would separate two significantly different approaches according to the Friedman-Nemenyi test ($p < 0.05$).

Table 3.5: LPO-validation C scores and comparison of four different modelling methods on data sets from Aiba *et al.* Methods' average C score across different data sets and average rank are reported.

	F	G	N	Z	$\overline{C}$	$\overline{rank}$
PLS-FC	0.883	**1.000**	0.905	**0.983**	0.943	2.875
PCA-AREA	0.883	**1.000**	**0.917**	0.917	0.929	2.875
PLS-AREA	0.950	0.933	0.905	0.950	0.935	3.125
MCE-euclid-AREA	**0.983**	0.700	0.905	0.583	0.793	4.000
MCE-euclid-FC	**0.983**	0.733	0.905	0.533	0.789	4.000
PCA-FC	0.883	0.867	0.857	**0.983**	0.898	4.125

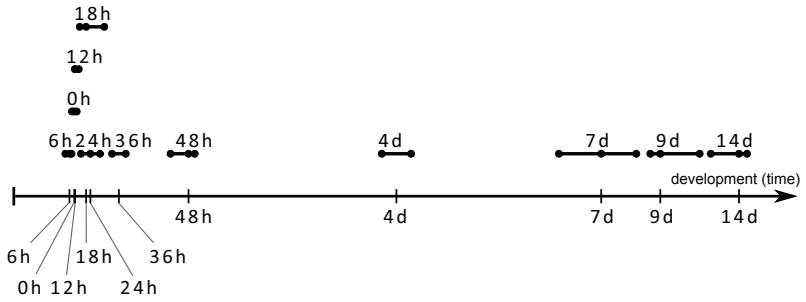
Table 3.6: C scores of external validation for PCA-AREA models inferred on data sets from Aiba *et al.* Training sets (row labels) and test sets (column labels) represent different cell lines measured with the same experimental platform.

	F	G	N	Z
F	/	1.000	0.976	1.000
G	0.933	/	1.000	0.950
N	0.850	0.883	/	0.817
Z	0.767	0.750	0.988	/

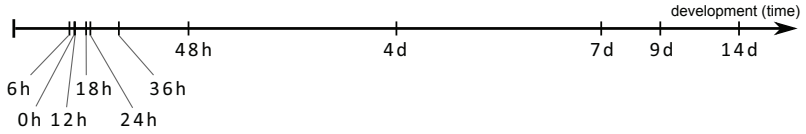
Analysis of inferred differentiation scales

From the five modelling methods considered, PCA and MCE are the only ones that truly discover the relations between cell stages from the data, that is, constructs an informative differentiation scale. PLS and SVM^{rank} are supervised and perhaps focus too much on optimizing their respective goals. For example, while expression profiles taken after 18 and 24 hours might be very similar, supervised algorithms will still try to separate the projections, because they know the samples come from different time-points. While Pathrecon is unsupervised, it only orders samples and does not provide a model for projection. MCE can be used for projection, but does not provide an explicit model for staging of new samples. That is why we here examine only PCA's differentiation scales. For all of the examined data sets, we found that the scales order the stages very well with only minor errors in the order of similar stages. For brevity we demonstrate this successful result on two selected data sets (Figures 3.7 and 3.8).

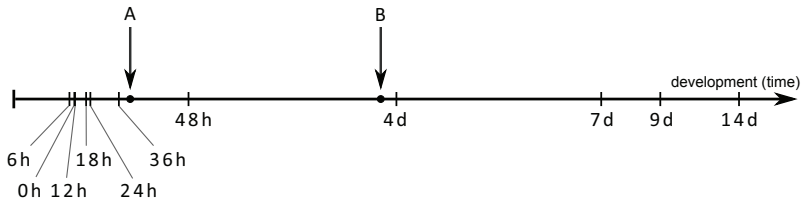
As a first example, let us illustrate the composition and utility of the differentiation scale and associated prediction model with the data from a study of the mouse R1 ESC line. The data included 11 different time points during 14 days of differentiation into embryoid bodies (EBs) [69]. At each time point the data (GDS2667) contains measurements of over 18,000 genes in three different biological replications. The predictive model was inferred from the data comprising the entire set of 33 samples from which we have excluded two samples for testing purposes. The projections in Figure 3.7 were inferred using PCA-AREA on a subset of the 1,000 most informative genes. The Gene Ontology annotation of this group of selected genes highlighted the efficacy of the selection strategy, with a significant number of genes annotated to biological functions involved in cellular differentiation, such as developmental process (25% of genes), growth (17%), and apoptosis (8%). The time-ticks in the differentiation scale in Figure 3.7(b), which indicate the developmental stages of the cell, correspond to the median position of the projections of samples taken at the same stage of development. They are ordered as expected, except for one transposition of the very similar time points at 0 and 6 hours. We can also observe a wide gap around 4 days of development, most probably reflecting the specific time resolution used



(a) Projection of the samples from the training data illustrates the construction of the predictive model and its associated differentiation scale. For visual clarity, projections are arbitrarily vertically dispersed. Samples observed at the same development stage are connected with a line.



(b) Differentiation scale for mouse embryonic stem cell differentiation.



(c) Prediction of the developmental stage for two samples (A and B).

Figure 3.7: An example of the projection of samples in a single-dimensional plot demonstrating the construction of the differentiation scale (a), the obtained differentiation scale representing a predictive model (b) and prediction of developmental stages of new samples (c).

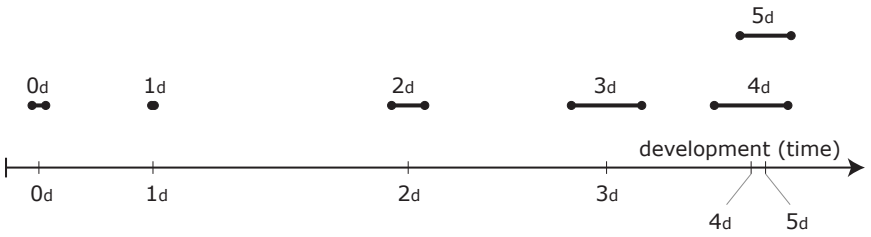


Figure 3.8: Differentiation scale for *M. musculus* embryonal carcinoma stem cells differentiating into primitive endoderm (data set F).

in the experiment, but also indicating that the cells undergo a substantial change in the time period between 48 hours and the 7th day.

To show the predictive capability of the model, we have left out two samples (A and B, Figure 3.7(c)) which were measured at 36 hours and 4 days, respectively. They were correctly projected, thus validating the model as a predictive tool.

For a second example we study a different type of pluripotent cells, the F9 embryonal carcinoma cell line during its differentiation into parietal endoderm for 5 days (Gene Expression Omnibus, GSE11523). The projection of this data set results in a perfectly monotonic scale of development. The last two stages do overlap, but this could be explained by the intrinsic variability in the speed of differentiation and in the composition of the three germ layers of each single cell line. This variation increases during differentiation and affects DNA microarray measurements.

3.6.4 Conclusions

The predictive accuracy of inferred models is very high when they are applied to data from the same cell lines as used in the training set. The reasonable range of C scores is from 0.5 (random predictor) to 1.0 (perfect prediction). The majority of C scores for the described methods are close to 0.9, a very high score indicating an excellent quality of predictions. The only notable exception is data set GDS2688, where all methods achieved lower scores. PCA and MCE were here the only two methods that obtained reasonably

good results. We can conclude that predicting the stage of development from transcriptional profiles is feasible and that the resulting prediction models can accurately predict developmental stages within a chosen cell line.

The results of external validation are also interesting. Aiba et al. observed that the trajectories obtained from cells of different cell lines diverged to a large extent. We therefore expected that the predictions of models developed on one cell line would fail when applied to data from another cell line. Results on our selection of data sets from GEO (Table 3.4) refute this expectation, and demonstrate that the tested predictive models can be applied across different cell lines. In addition, external validation on data sets from Aiba et al. [70] (Table 3.6) is also qualitatively similar, showing that prediction across different cell lines is indeed feasible and may be highly accurate. The scores we have obtained are surprisingly high, with only four below 0.80 and 24 above 0.90 (out of 42). The results for all possible pairs of data sets shown in the Supplementary data confirm the high accuracy of predictions even for data coming from different studies. Poor predictions were obtained only for training and test data from different species.

Utility of stage prediction models across different cell lines was further confirmed in IPSCs and MEF experiments. Projection of related transcription profiles on a PCA-inferred differentiation scale highlighted the difference in pluripotency between the adult cells and the reprogrammed cell line.

Among the tested methods the differences in predictive quality were not statistically significant. PCA, MCE, PLS and SVM^{rank} are all time-efficient and construct corresponding models for the data sets in our study within seconds. Pathrecon can be very slow with execution times of several hours or even days for data sets where construction of a PQ-tree and examination of all candidate orderings is required. We thus prefer principal component analysis because of its simplicity, explicit prediction model, and the added benefit of its informative differentiation scales (Figures 3.7 and 3.8). The staging is easy to interpret by biologists, and the visualization uncovers the dynamics of the changes with phenotypically different stages being placed farther apart on the differentiation scale. As its nonlinear counterpart, MCE looks very promising and should be considered along PCA in further studies of this kind.

Chapter 4

Decomposition and multitask learning

Binary decomposition was mentioned in the previous chapter as one of the approaches for solving multipartite ranking problems. In this chapter we propose an approach that upgrades and improves the binary decomposition with multitask learning.

Binary decomposition transforms an original ranking problem into several simpler binary classification subproblems. It is an elegant way to reduce the (newer) problem of ranking to well-established tasks and methods of classification. Existing ranking methods that use this approach were primarily concerned with how to decompose the problem and aggregate the results. Solving the decomposed tasks was considered secondary — a standard problem that most researchers should be familiar with, and which can be solved by one of the many readily available methods. This is true to a certain degree and the approach has been shown to work in practice [22, 8, 21, 71].

We here postulate that the approach can be improved further. Individual, decomposed subproblems are indeed simple in form and do not differ much from the definition of a standard classification problem. But the set of all subproblems does not need to be considered separately, as unrelated and independent tasks. We will show that there is additional structure in the similarities between these tasks, which can be exploited. We claim that the problem of learning a rank prediction function from decomposed ranking provides the setting that multitask learning has been designed for.

Multitask learning [24] is based on the assumption that learning of related tasks simultaneously can be done better than simply learning each task independently. The shared structure and similarities between tasks can serve as an inductive bias, helping to guide the learning of one task with the knowledge and information obtained from the others. This can be done in different ways, depending on the type of task relatedness. The main guiding principles are that the predictive models of related tasks should have similar parameters or structure.

Combining techniques of both binary decomposition and multitask learning has great synergistic potential. The first produces multiple tasks that are intrinsically related — an unavoidable consequence of the fact they were formed from a common original problem. Multitask learning can use this shared common structure and help to build better predictive models for the decomposed tasks.

4.1 Binary decomposition

Binary decomposition is a variant of divide-and-conquer, an algorithm design strategy often used in various fields of computer science. It has been used in machine learning in a very similar form even before much dedicated research was done in the area of ranking. When classification problems are generalized from binary classification with positive and negative labels to multiclass classification with $k > 2$ possible labels, some methods can not be directly adapted for solving the more general type of problems. A common solution is to transform the multiclass problem into several binary problems, which the existing methods can handle [44, 72, 21].

The same approach can also be used for multipartite instance ranking [22, 8, 27, 73, 18, 74]. One way to represent this problem is as prediction of ordinal labels. In such setting most binary decomposition techniques that were designed for multiclass classification — prediction of categorical labels — can be applied directly. These techniques do not use the ordinal structure of the labels, but can still work well [73, 8]. It is also possible to use the additional ordinal structure not found in multiclass problems and construct a ranking specific decomposition technique. The best example, and also one of the first, is the ranking method based on binary decomposition proposed

by Frank and Hall [22].

Binary decomposition implementations differ in the details of the transformation from the original problem into binary subproblems. The data sets of decomposed problems can be disjoint or overlapping, they may contain instances from only two classes (ranks) or more, and the number of tasks produced with respect to the number of classes can vary from quadratic to linear or even logarithmic. Further details accompany descriptions of specific implementations below.

The primary motivation for using binary decomposition is to enable solving problems with methods that were not originally designed for them, but there might be additional benefits. Fürnkranz [75] argues that, for solving multiclass problems, binary decomposition can increase performance even for methods that can already directly handle multiclass data. He proposes that decomposition can be viewed as a general ensemble technique. He also mentions that a similar case could be made for ordered classes and ranking. An idea that is further examined in later papers [8, 73], and which we also build upon in our work.

When discussing binary decomposition methods, the decomposition part of the process is mostly in the forefront. However, to be able to completely solve the original problem, it is also necessary to combine the results of subproblems back into the final solution of the original problem. This process, known as *aggregation*, is also an integral part of the whole procedure and needs to be specified together with the decomposition strategy.

Several popular approaches for aggregation functions are based on voting schemes [72, 75, 18, 21]. Classifiers for subproblems can provide class predictions (positive or negative) or confidence values (*e.g.* probabilities, margins) which then determine the final output. Because of a strong connection between decomposition and aggregation, the details of both are typically given together as a pair that forms the complete method.

Unlike utility and pairwise ranking approaches that can also be used for continuous ranking and to some extent bipartite ranking, binary decomposition specifically targets multipartite ranking.

In the following, the most used binary decomposition implementations are reviewed and discussed in the light of the problems from ranking.

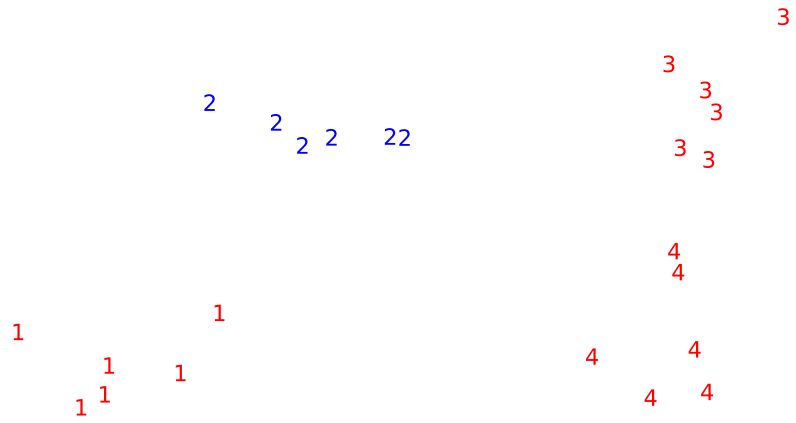


Figure 4.1: Data instances shown in a scatterplot, with the colour marking the label in the second binary task of a decomposed multipartite ranking problem. Instances from the second rank class are positive (blue) and others negative (red).

4.1.1 One-against-all

One-against-all (OAA) is one of the two basic variants (along with one-against-one) of binary decomposition that are also used for multiclass classification. It can be used for ranking problems as well, but does not make use of the ordinal structure of classes.

The approach transforms a k-class multipartite ranking problem into k binary tasks. Each task (i) includes all original instances, which are relabelled so that instances from the i-th class are positive, and all others are negative. Figure 4.1 shows an example for task (2) of a four class multipartite ranking problem. Compared with some other approaches, such decomposition results in a relatively low number of tasks with more data instances per task. The resulting binary tasks also have an unbalanced class distribution — few positive (one class) and many negative (all other classes) instances. This could be more or less of a problem, depending on the base learner used for solving binary classification tasks [76, 77].

Aggregation

Ideally, for a given data instance, only one of the binary classifiers should classify this instance to a positive class. In practice it can happen that none or multiple predictions are positive, making the aggregation non-trivial. When only the predicted values are available, ties can be broken randomly. Base learners that also provide some sort of confidence estimates are much more effective in this case, as the prediction with the highest confidence can be chosen.

In addition to this approach which also works for multiclass problems, ranking problems allow another solution. Because we are not limited to predicting only the labels, as for categorical values, the aggregation can result in new intermediate values. For example, when both classifiers (1) and (2) make positive predictions, a multiclass method would have to break the tie and predict either value 1 or value 2. But a ranking method can predict the average value of 1.5 for the instance, which might result in a better ranking. This also works with confidence values, which enable a weighted average to be used instead.

4.1.2 One-against-one

Also known as round robin classification or pairwise classification [72, 75], one-against-one (OAO) is another general binary decomposition technique that can be used for both multiclass and ranking problems.

It produces $\frac{k(k-1)}{2}$ binary tasks. Each task (i, j) for $i < j$ has instances from classes i and j as positive and negative examples, respectively. Example of a $(2, 4)$ task is shown on Figure 4.2. Only instances from classes 2 (positive, blue) and 4 (negative, red) are present in this task, while all others (grey) are omitted.

Compared with one-against-all, this means more tasks (quadratic instead of linear), but each has only $\frac{2}{k}$ of all instances on average. As Fürnkranz [72] showed, this actually results in lower or equal overall asymptotic time complexity required for learning all tasks.

Less instances per task has another important benefit in addition to reducing the time complexity. Each individual task may be simpler, and thus allow a less complex model to perform well. For example, a linear

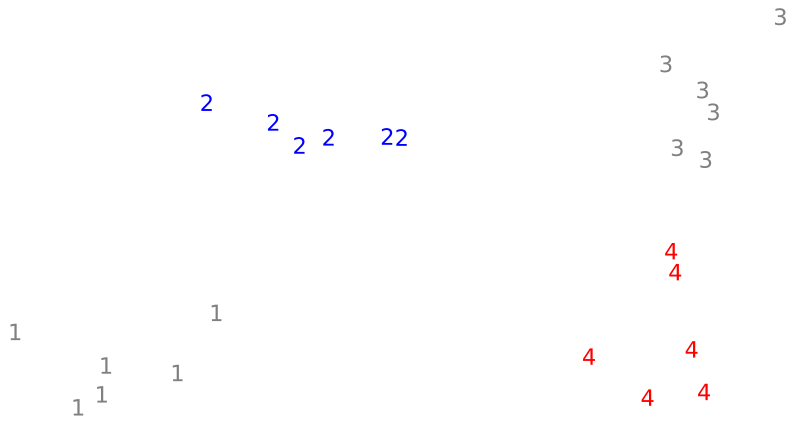


Figure 4.2: Example showing binary task $(2, 4)$ separating the second and fourth rank class, which form the positive (blue) and negative (red) binary classes, respectively. Other instances are not used in this task (grey).

classifier might be able to separate two classes at a time, while it could not separate one class from all the others. See section 4.4.2 for a further study of this property. Also, the class distributions in individual tasks are much more balanced, as only one rank class is compared with one other.

A possible downside of the low number of instances per class is that subproblem data sets may be prohibitively small. For example, when there are five instances per rank class, the resulting subproblems have only ten learning instances. This might not be enough for some methods, especially in the case of many, possibly irrelevant, features.

Aggregation

The standard approach is to let binary classifiers cast votes and predict the majority label. This can again be expanded to a weighted majority, when classifiers also provide confidence estimates.

There are several other choices [78], but despite its simplicity, the voting scheme has been shown to perform well. Hüllermeier et al. [17] even show

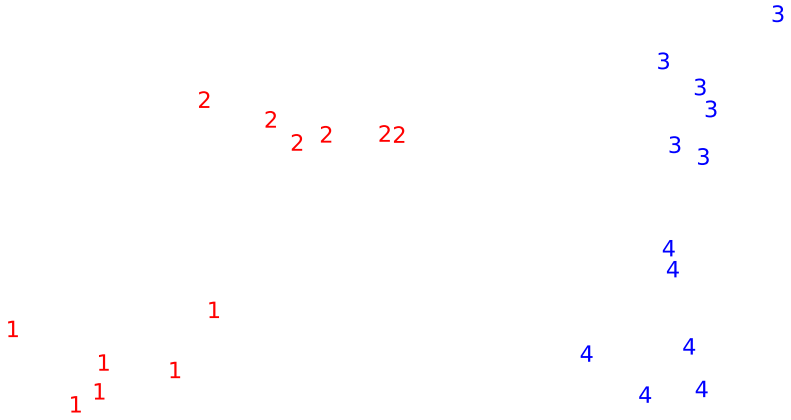


Figure 4.3: Example showing binary task (> 2), with data instances above the threshold forming the positive class (blue) and those below or equal to the threshold forming the negative class (red).

that under certain conditions the voting scheme optimizes the Spearman rank correlation, a measure very relevant to ranking.

4.1.3 Thresholding

Proposed by Frank and Hall [22], the approach we will refer to as thresholding, was one of the first binary decomposition methods designed specifically for ranking. It makes use of ordinal class structure to produce a simple yet effective method. Its idea is based on the fact that classes are ordered and can be split into two groups by selecting a threshold. Classes above and below the threshold form the positive and negative classes of the new binary task. Repeating this for all possible splits produces $k-1$ binary classification tasks.

In our work we will name the tasks by the highest rank class in the lower group. Binary task ($> i$) thus differentiates between instances from the first to i-th rank class, and instances from higher rank classes. See Figure 4.3 for an example of the data for task (> 2).

The resulting binary tasks have similar characteristics as those of the one-against-all approach. Each task contains all instances, and there is a linear number of tasks. But because the ordinal nature of data was used in the transformation, the positive and negative instances of tasks should be more homogeneous. This avoids the situations from one-against-all, where the negative instances come from low as well as high rank classes, which could be vastly different, and make separation from a single chosen class more difficult. Another possible improvement compared with one-against-all is in more balanced class distributions of binary tasks. Only the first and last task match one rank class against all others, while thresholds closer to the middle result in more equal distributions.

Aggregation

The approach proposed in the original paper [22] seeks to find the most probable class. The probability that an instance x belongs to class i is computed from binary classifiers of two successive tasks $(> i-1)$ and $(> i)$:

$$P(x \in R_1) = 1 - P(x \in R_{>1}) \tag{4.1}$$

$$P(x \in R_i) = P(x \in R_{>i-1}) - P(x \in R_{>i}), \quad 1 < i < k \tag{4.2}$$

$$P(x \in R_k) = P(x \in R_{>k-1}) \tag{4.3}$$

The above aggregation rule was proposed for ordinal classification where one of the classes needs to be predicted. When the goal is to rank instances, a different aggregation function can be used to get the utility scores of instances for the ranking. Fürnkranz et al. [8] propose to use the sum of predictions $f_i(x)$:

$$f(x) = \sum_{i=1}^{k-1} f_i(x) \tag{4.4}$$

It is worth noting that this function works well even if the predictions are not probabilities, but for example margins of a decision function or, with some loss of precision, even binary 0/1 predictions.

4.1.4 Generalizations

In addition to the most widespread variants described above, there have been other proposals and efforts to construct a formal framework that gen-

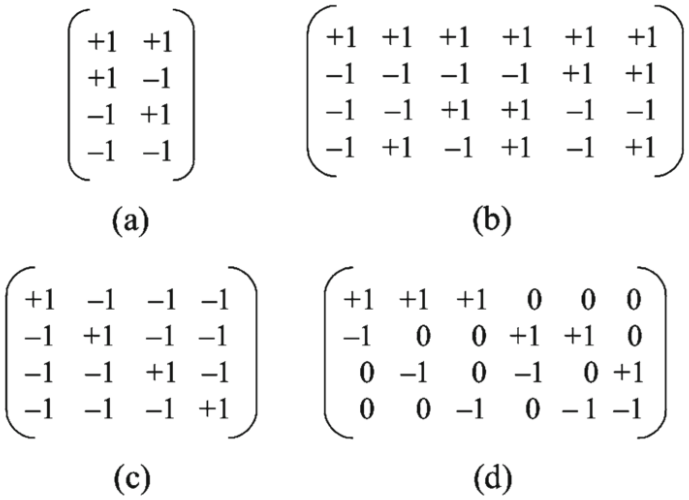


Figure 4.4: Example from Lorena et al. [21] showing matrices describing different decompositions of four classes: the most compact (a), ECOC-based (b), OAA (c) and OAO (d).

eralizes these and other specific methods. One of the generalizations represents different decompositions in terms of error correcting output codes (ECOC) [79, 80]. These can be described with a matrix $\mathbf{M} \in \{-1, 0, 1\}^{k \times l}$ as proposed by Allwein et al. [80]. Each column of the matrix represents one of l binary tasks, and each row one of k original classes. A column vector describes which classes belong to the negative (-1) and positive (1) examples, and which are not used (0) in that binary task.

Figure 4.4 shows matrices describing different decompositions of four classes. The most compact decomposition uses only $\log_2 4 = 2$ binary tasks (a), but does not have much capacity for error correction (less robustness). ECOC try to increase stability and performance through redundancy with more tasks (b). Previously mentioned strategies one-against-all (c) and one-against-one (d) can also be described by matrices.

Aggregation has a natural interpretation as the decoding of ECOC. If the predictions of binary classifiers are considered as the received transmission, they must be decoded to the closest codeword representing one of the original classes. This can be done by measuring the Hamming distance between the vector of predictions and rows in the matrix [79].

Another direction of research focuses on approaches that do not need to

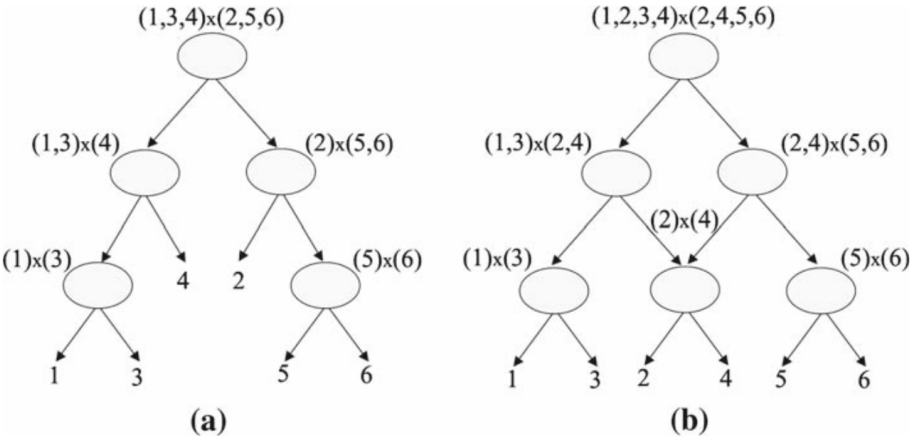


Figure 4.5: Example from Lorena et al. [21] showing tree (a) and DAG (b) structured decompositions. They describe the sequence of queried binary models used to determine the class. For example, the first model in (a) decides between classes (1,3,4) and (2,5,6).

use all binary classifiers for prediction of a new instance. For example, when using one-against-one, classifying an instance from class 2 involves making predictions with many classifiers that are deciding between two choices of which neither is correct (*e.g.* $(3, 4)$, $(4, 5)$, etc.). To limit predictions to relevant classifiers only, it has been proposed to organize the classifiers hierarchically into a tree structure [81] or more generally a directed acyclic graph (DAG) [82, 18]. During the prediction process the structure is traversed and the next classifiers are dynamically chosen based on previous results. Figure 4.5 shows an example of a DAG used to decide which classifiers to use.

4.2 Multitask learning

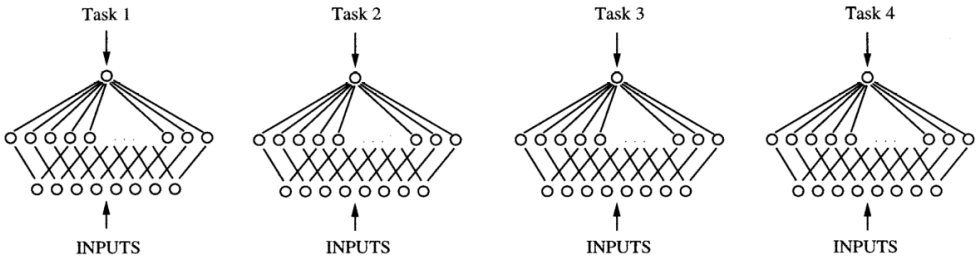
Multitask learning (MTL) is an area of machine learning that focuses on simultaneous learning of multiple tasks. Its goal is to improve the performance that could be achieved by learning each task independently. When learning from data on one task, this is possible by utilizing the data from other related tasks and the knowledge extracted from them.

Multitask learning shares many similarities with *transfer learning* [83], another machine learning discipline concerned with using knowledge extracted from some tasks and transferring it to others. The main difference between both fields is in the treatment of learning tasks. Transfer learning distinguishes between tasks, learning from source tasks and transferring some extracted knowledge to target tasks, making the learning procedure sequential. Multitask learning, on the other hand, considers the tasks equal and does not differentiate between them. The learning process involves all tasks simultaneously, and the knowledge is shared between all of them.

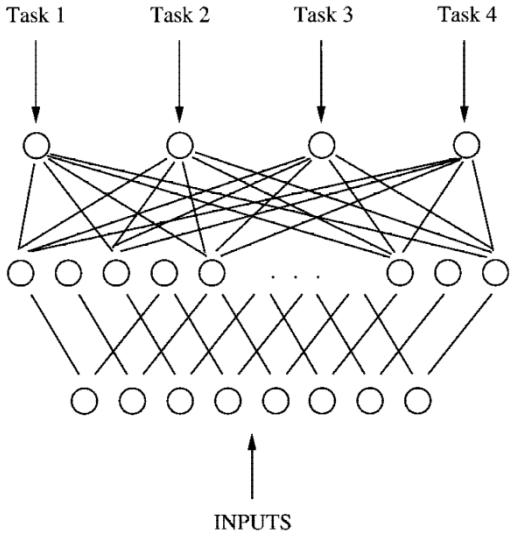
Another term that has been used for similar work, often even interchangeably with multitask learning is *learning to learn* [84, 85, 86]. Caruana started using the term *multitask learning* to describe his and similar approaches using the paradigm of learning from multiple related tasks (see [24] and references therein). He proposed a learning architecture consisting of a single artificial neural network with multitask backpropagation to learn a model for multiple tasks. The approach was favourably compared to learning with several independent neural network models, each one employed for its own task (see Figure 4.6).

Although this concept has since been expanded to many other methods, the neural network example still captures the basic idea very well. Using one network with multiple layers for all tasks allows the first layer to capture the shared structure by transforming the input variables into more useful intermediate values. After this transformation, which is equal for all tasks, the intermediate values are used in different ways to predict the final outputs of tasks. As we will see later in this section, some recent methods work exactly the same, but use different model representations. For example, in Argyriou et al. [87] the shared hidden layer is replaced by a linear (or kernel) transformation into new features, which are then used to fit task models by optimizing a convex loss function, such as in ridge regression, logistic regression or SVMs.

With increasing availability of related data sets, multitask learning is becoming more useful and widely adopted. The field is also becoming better defined and its concepts introduced into a variety of frameworks. This has led to new approaches for exploiting the benefits of related tasks. Many methods [24, 87, 88, 89] assume that all tasks use the same domain. That



(a) Many models, each used to predict a separate task, trained and used independently.



(b) A single model used to predict many tasks. Training is performed simultaneously with multitask learning.

Figure 4.6: Example from Caruana [24] comparing the architectures of single task learning and multitask learning.

is, their data instances are described with the same set of features. As we plan to use multitask learning on tasks generated from a single ranking data set with binary decomposition, this assumption will hold in our setting.

Multitask methods can be split into two families based on the types of task similarities that they can model and use. The first will assume similar tasks and hence tend towards similar models, as described by their parameters. The second family relaxes this assumption, and expects only the structure of the data and the model to be similar (*e.g.* common subspace or feature subset).

4.2.1 Similar parameters

Often prediction models have to be built for data obtained from distinct, but very similar, data sources. As an example, consider the `school` data set, a popular benchmark data set used for multitask evaluations [90, 89, 87]. It contains data about students, along with their performance, for 139 London schools. Using just one model to predict student performance would ignore the school-specific patterns. But it is also reasonable to assume that school-specific models will be similar, since many of the performance indicators are universal and might only be combined with slightly different weights for different schools. Thus, taking advantage of additional data from other schools when fitting individual models could make the learning process easier and improve the final models.

Similar situations are found more and more frequently with data sets for multiple users becoming available, or experiments being repeated in modified conditions. In these circumstances models are expected to be similar, but not exactly the same.

Formally, the similarity can be modelled and imposed as a common prior for task-specific parameters [90, 91]:

$$w_t \sim p(\theta), \tag{4.5}$$

where w_t is the parameter vector for the task t, and $p(\theta)$ a prior distribution estimated from all tasks.

Another formalization, which is widely used in various optimization methods, is that the parameter vectors should be close in the vector space, that

is, their difference should be small as measured by some norm. For multiple tasks this can be done by minimizing the deviations from the mean parameters. The parameter vector of task t can be written as:

$$w_t = w_0 + v_t, \tag{4.6}$$

where w_0 is the vector of mean parameters all tasks should be close to, and v_t the deviations from this mean for task t. Methods can then simultaneously optimize the task-independent w_0 and task-dependent v_t. Including a regularization term in the optimization function can penalize large norms of v_t, and therefore enforce task similarity. This approach was used to modify the SVM optimization for multitask learning [88], and later adapted to a more general framework for regularized optimizations [89].

Let us just mention that there are also many extensions to this basic approach. For example, when every task is not related to all others, the similarities can be imposed inside smaller task clusters [89, 91].

4.2.2 Similar structure

The fact that tasks are related does not necessarily imply that models' parameters, and consequently their predictions, should be similar. Clearly, predicting y and $-y$ are closely related tasks, but the parameters for both models will not be close. Using the approach from the previous family and forcing parameter similarity would be detrimental, since parameters should in fact be opposite.

This implies other types of task relatedness can also be exploited, but might need a different approach to multitask learning. Thus, the second family consists of methods focused on learning from tasks that are only structurally similar. This is mirrored in the predictive models, which are constrained or encouraged to have similar structure.

For the toy example of predicting a variable and its negative, the similarity in structure would be very strong (parameter values between tasks should differ only in a change of sign). More realistic examples of similar structure can be found in data sets where instances have been measured with the same technique and contain irrelevant or highly correlated features. While the target values that tasks try to predict may be different, they may

all be functions of only a subset of the features. Even when all features seem relevant, the decision functions may be defined in a common subspace and need only a small number of hidden factors to be expressed.

Methods that learn this structure from all tasks can subsequently fit simpler models for individual tasks. While model simplicity is usually desired, this approach can also achieve better predictive performance. Learning the correct features first can greatly alleviate the optimization problems for individual tasks, especially when few instances are available for them.

Examples of methods using this approach include multitask neural networks mentioned at the beginning of this section [24] as well as methods using regularized optimization proposed more recently [87, 89]. We will describe one of them, called *multitask feature learning* and proposed by Argyriou et al. [87], in greater detail as it will be used in the rest of this work. Note that there are some minor changes between the description below and the original paper to conform to our notation and definitions.

Multitask feature learning

As the name implies, multitask feature learning (MTL-FEAT) [87] works by first transforming the original input space into a set of new features. The number and composition of these are learnt with the goal that a small set of new features shared across tasks will be enough to build good individual task models.

Given T supervised learning tasks, for each task $t \in \{1, \ldots, T\}$ the training data set consists of n labelled instances $(x_{t1}, y_{t1}), \ldots, (x_{tn}, y_{tn}) \in \mathbb{R}^d \times \mathbb{R}$ (task training set sizes can in fact vary, and are only assumed to be the same to simplify notation). We wish to fit decision functions $f_t : \mathbb{R}^d \to \mathbb{R}$ for each task t, represented as:

$$f_t(x) = \sum_{i=1}^{m} a_{it} h_i(x), \quad t \in \{1, \ldots, T\}, \tag{4.7}$$

where $h_i : \mathbb{R}^d \to \mathbb{R}$, $i \in \{1, \ldots, m\}$ are the features and $a_{it} \in \mathbb{R}$ the regression parameters, all of which (a_{it}, h_i and m) should be learnt from the data.

Assuming orthogonal linear features $h_i(x) = \langle u_i, x \rangle$, where $u_i \in \mathbb{R}^d$, is similar in spirit to unsupervised methods such as PCA. It is shown later in the paper that kernels can be used to model nonlinear features as well.

Using $U \in \mathbf{O}^d$ to denote the orthogonal $d \times d$ matrix with columns u_i, the task functions can be written as:

$$f_t(x) = \sum_{i=1}^{d} a_{it} \langle u_i, x \rangle = \langle a_t, U^T x \rangle \tag{4.8}$$

The assumption that the tasks share a "small" set of features ($m \ll d$) means that the matrix A has many zero rows and, so, the corresponding features (columns of matrix U) will not be used by any task. Rather than learning the number of features m directly, Argyriou et al. introduce a regularization function which favours a small number of nonzero rows in the matrix A.

$$\mathcal{E}(A, U) = \sum_{t=1}^{T} \sum_{i=1}^{n} L(y_{ti}, \langle a_t, U^T x_{ti} \rangle) + \gamma \|A\|_{2,1}^2 \tag{4.9}$$

where $\gamma > 0$ is a regularization parameter. The first term measures the loss, while the second is a regularization term using the (2,1)-norm of matrix A. The value of $\|A\|_{2,1}$ is obtained by first computing the 2-norms of the (across the tasks) rows a^i (corresponding to feature i) and then the 1-norm of the vector $b(A) = (\|a^1\|_2, \ldots, \|a^d\|_2)$. Because the 1-norm favours zero entries (sparsity), a solution will be sought, where the 2-norms of some rows (features) will be zero, which means they have zero entries for all tasks.

To learn both the features (U) and the task-dependent feature weights (A), the error function is minimized over both U and A.

$$\min\{\mathcal{E}(A, U) : U \in \mathbf{O}^d, A \in \mathbb{R}^{d \times T}\} \tag{4.10}$$

Argyriou et al. note that optimizing (4.10) directly is hard because the problem is not convex and the regularizer $\|A\|_{2,1}^2$ not smooth. Instead, they propose an equivalent convex problem for which a global optimal solution can be found. The equivalence of problems (4.10) and (4.12) is proven in the paper, but skipped here for brevity.

$$\mathcal{R}(W, D) = \sum_{t=1}^{T} \sum_{i=1}^{n} L(y_{ti}, \langle w_t, x_{ti} \rangle) + \gamma \sum_{t=1}^{T} \langle w_t, D^+ w_t \rangle \tag{4.11}$$

$$\begin{aligned} \min\{\mathcal{R}(W, D) : W \in \mathbb{R}^{d \times T}, D \in \mathbf{S}_+^d, \operatorname{trace}(D) \leq 1, \\ \operatorname{range}(W) \subseteq \operatorname{range}(D)\} \end{aligned} \tag{4.12}$$

In this formulation, matrices W and D are optimized instead of A and U, where W contains the weights for original input features ($W = UA$) and D encapsulates the feature transformation.

The proposed algorithm alternately optimizes $\mathcal{R}$ with respect to W and D. To ensure convergence, a perturbation of the objective function (4.11) is minimized instead: $\mathcal{R}_\varepsilon : \mathbb{R}^{d\times T} \times \mathbf{S}^d_{++} \to \mathbb{R}$

$$\mathcal{R}_\varepsilon(W, D) = \sum_{t=1}^{T}\sum_{i=1}^{n} L(y_{ti}, \langle w_t, x_{ti}\rangle) + \gamma \text{ trace}(D^{-1}(WW^T + \varepsilon I)). \quad (4.13)$$

This allows the convergence to an optimal solution of (4.12) to be proven by letting $\varepsilon \to 0$ (refer to the paper for details). The authors note that in practice alternating minimization of the unperturbed objective function also converges to an optimal solution, however in theory this convergence is not guaranteed. As we noticed the same in our experiments, we mostly optimized the unperturbed function directly, avoiding the need for an additional outer loop in the optimization.

Each of the two steps of alternating optimization can be easily solved. In the first, D is kept fixed and the weights W are computed. This can be done by first transforming the data (using the current D), and then computing the weights for each task independently with a standard 2-norm optimization. Therefore, efficient standard solvers can be used for methods such as SVM, logistic regression, ridge regression, etc.

In the second step, the objective is minimized with respect to D for a fixed W. This problem has a closed form solution:

$$D_\varepsilon(W) = \frac{(WW^T + \varepsilon I)^{\frac{1}{2}}}{\text{trace}(WW^T + \varepsilon I)^{\frac{1}{2}}} \quad (4.14)$$

Both steps are repeated until convergence is achieved as shown in Algorithm 1.

4.3 Combining binary decomposition and multitask learning

We hypothesize that binary decomposition and multitask learning techniques could greatly benefit from each other when combined. Intuitively, it makes

Algorithm 1: MTL-FEAT algorithm using alternating optimization

Input: training sets $\{(x_{ti}, y_{ti})\}_{i=1}^{n}, \quad 1 \leq t \leq T$

Parameters: γ, ε, tol

Output: $d \times d$ matrix D,
$d \times T$ regression matrix $W = [w_1, \ldots, w_T]$

$D = \frac{I}{d}$

while $\|W - W_{prev}\| > tol$ **do**

  for $t = 1, \ldots, T$ **do**

    `// compute weight vector for task` t

    $w_t = \text{argmin} \{\sum_{i=1}^{n} L(y_{ti}, \langle w, x_{ti} \rangle) + \gamma \langle w, D^{-1} w \rangle : w \in \mathbb{R}^d\}$

  end

  `// optimize` D `at current` W

  $D = \frac{(WW^T + \varepsilon I)^{\frac{1}{2}}}{\text{trace}(WW^T + \varepsilon I)^{\frac{1}{2}}}$

end

sense that decomposing one problem into several binary tasks will result in some shared structure and allow improvements from learning the tasks simultaneously. However, successfully designing a method that combines binary decomposition and multitask learning requires careful selection of specific methods to maximize the synergy of their combination. There are many details that distinguish different binary decomposition and multitask learning approaches, as was shown in the previous two sections. Due to this, some combinations will work together better than others.

We propose a combination of a binary decomposition and a multitask learning technique that takes advantage of each other's strengths and achieves a good final result. We give some justifications for our choice below as well as in section 4.4, which contains further analysis and experimental validations supporting our decisions. That said, we believe there are many more possible methods that could work well and believe this is a promising direction of future research that could result in a whole family of new methods.

4.3.1 Proposed method

We decided to start by considering the three basic, but most widely used, binary decomposition approaches (one-against-all, one-against-one, thresholding) and enhance them by simultaneously learning the tasks with a multitask method.

All three decomposition approaches can result in tasks that require different values of model parameters. For OAA and thresholding, differentiating between the first rank class and others (task (1) in OAA, and (< 1) in thresholding) might depend on different features and patterns than differentiating between the last rank class and others (tasks (k) or ($> k$)). Similarly for OAO, models for separating low and high rank pairs, *e.g.* tasks $(1, 2)$ and $(k-1, k)$, can require significantly different parameter values. Thus, simple versions of mean-regularized multitask learning, where all models are assumed to be similar, do not present the best choice. Instead, we decided to use a method relying only on similar structure.

More specifically, we selected multitask feature learning [87] (MTL-FEAT) described in the previous section. This method has several desirable properties and makes assumptions that better agree with the problem obtained from binary decomposition. MTL-FEAT does not penalize dissimilar task models. As long as models can be expressed as a combination of few newly learned features, the weights associated with these features do not need to have similar values between different models. As all binary tasks use the same input variables (those from the original data set), the relationships between them also remain the same. Thus, if a group of variables combines into a new feature (hidden factor) important for prediction, this pattern will be present in all tasks. Learning new features shared between tasks is therefore very appropriate and should be more robust than (re)discovering the same features in each task independently.

A secondary benefit of MTL-FEAT, in addition to improving binary classifiers, is its transparency. The transformation from original into new features can be retrieved in matrix form. This makes it possible to inspect how the most useful features are constructed. Similarly, the weight matrix can be used to see which features are used in which task classifiers and how. This multitask method therefore not only exploits the similarities in

Table 4.1: Basic properties of tasks generated with three binary decomposition techniques: one-against-all (OAA), one-against-one (OAO) and thresholding.

decomposition	# tasks	# rank classes per task
OAA	k	k
OAO	$\frac{k(k-1)}{2}$	2
Thresholding	k-1	k

structure, but also allows them to be studied and used to interpret models.

We compared the basic properties of the set of tasks produced by the three considered binary decomposition methods. The most important properties are the number of tasks and the number of rank classes included in each task, which are summarized in Table 4.1. Based on this comparison, we believe that one-against-one decomposition has the biggest potential to benefit from multitask learning. It results in more binary subproblems than one-against-all and thresholding, which provides more tasks for the multitask methods to exploit. Individual tasks also have less learning instances. While the larger number of tasks can counterbalance this and keep OAO competitive, it also enables even greater improvements from multitask learning, which excels when lack of data instances makes single task learning harder. Perhaps most importantly, the binary tasks that result from OAO contain different learning instances. Since MTL-FEAT learns features from all tasks, they can be estimated much more easily, compared with using only a part of all instances (those of a single task). These observations are further studied and tested in section 4.4.1

A voting scheme is used to aggregate predictions of binary classifiers. The most common rank class is declared the winner of the vote and is used as the prediction for the original ranking problem. Both standard and weighted versions of voting were tested (see section 4.4.3 for details). Algorithm 2 gives the pseudo-code for the final ranking algorithm — BDMT (Binary Decomposition and Multi-Task learning) — which consists of functions for fitting the model, and for predicting the ranks of new instances.

The *fit* function accepts a data set *data* which consists of (`instance`, `label`) pairs (x, y) and first decomposes it into binary tasks. These are

Algorithm 2: BDMT ranking algorithm with fit and predict functions.

```
1  Function fit(data)
2      tasks = [ ]
3      ranks = [ ]
4      for i = 1, ..., k do
5          for j = 1, ..., i − 1 do
6              data_{i,j} = {(x, y) | (x, y) ∈ data and y ∈ {i, j}}
7              tasks.append(data_{i,j})
8              ranks.append((i, j))
9      D, W = MTL-FEAT(tasks)
10 Function predict(x)
11     predictions = [ ]
12     for t = 1, ..., T do
13         w_t = W[:, t]
14         i, j = ranks[t]
15         if ⟨w_t, x⟩ > 0 then
16             predictions.append(i)
17         else
18             predictions.append(j)
19     return most_common(predictions)
```

obtained by considering all possible pairs of rank classes (i, j) for $1 \leq j < i \leq k$ (for loops in lines 4 and 5). A subset of data instances whose rank is either i or j is constructed (line 6) and appended to the list of tasks (line 7), the pair of ranks (i, j) is appended to the list of selected ranks (line 8). Finally, models for all tasks are estimated simultaneously with MTL-FEAT (line 9), which learns the task-specific weight coefficients (W) and the space transformation (D).

The *predict* function takes a data instance x and predicts a rank for it. This is done by first collecting all predictions of individual models. The for loop (line 12) iterates over $T = \frac{k(k-1)}{2}$ tasks and uses the previously learned weight vector (line 13) and original ranks (i, j) (line 14) to predict which of the two considered ranks is more likely (lines 15–18). After all predictions have been computed, the one that appears most frequently is returned as the predicted rank for instance x (line 19).

The algorithm does not introduce new parameters apart from those necessary for MTL-FEAT. We estimate the best value for the regularization parameter γ from the data (either on separate data or using an additional internal validation step). When choosing the value of the *tol* parameter, which determines when the iterative optimization should be stopped, we noticed that the size of W (number of weight coefficients) is an important factor. For this reason we modified the stopping criterion from $\|W - W_{prev}\| > tol$ to the normalized version $\frac{\|W - W_{prev}\|}{size(W)} > tol$. After this we obtained good results under various conditions with a fixed value of $tol = 10^{-5}$. Increased optimization precision can be achieved with a lower value (*e.g.* $tol = 10^{-6}$) at the expense of more iterations, but the difference in the final prediction results was minor in most experiments. We chose logistic regression as the base classification learner (among the convex 2-norm optimization algorithms). Another popular choice would be a linear SVM learner, which did not result in significantly different results in our limited tests.

4.4 Analysis and evaluation

We analyse several properties of binary decomposition methods, multitask learning, and their interaction to justify our proposal of combining both fields. An extensive experimental evaluation of BDMT, a new ranking

method based on the proposed combination, is provided. Its performance is compared under different learning conditions that vary in the amount of training data, the number of ranks and the dimensionality of the problem. Its relative improvement over the straightforward use of OAO, on which it is based, is observed, as well as its performance relative to other global ranking methods. We also give examples of other advantages of BDMT, beyond the improvements to predictive performance, such as the good interpretability of its models.

4.4.1 Properties of decomposed tasks

In the previous section we argued that the lower number of data instances per task as well as the larger number of tasks, which are generated by the OAO approach, produce favourable conditions for multitask learning. Here we present experiments that confirm these two hypotheses. We generated related multitask data sets, which allowed us to measure the performance of multitask learning and its relative improvement with respect to learning the tasks independently.

The synthetic data sets were generated for k tasks, and for each task contained n training data instances and a separate set of 100 instances per task for testing. Each instance $x \in \mathbb{R}^m$ was described by $m = 32$ features, each drawn from the normal distribution $N(0, 1)$. To make the tasks related, data features $f_1, \ldots, f_m$ were used to compute three hidden factors F_1, F_2 and F_3 that were present in all tasks:

$$F_1 = f_1 + f_2 + f_3 \ , \tag{4.15}$$

$$F_2 = 2f_4 - \frac{1}{2}f_5 - \frac{1}{2}f_6 \ , \tag{4.16}$$

$$F_3 = 0.1 \sum_{i=7}^{16} f_i \ . \tag{4.17}$$

All task models were defined as logistic models using only the three hidden factors with weights drawn randomly from $N(0, 1)$ for each task. This resulted in different task-specific linear models with shared structure — for example, the weights for the first input feature were different in each task, but always equal to the weights of the second and third input features. Another structural similarity was that only the first 16 input features were used

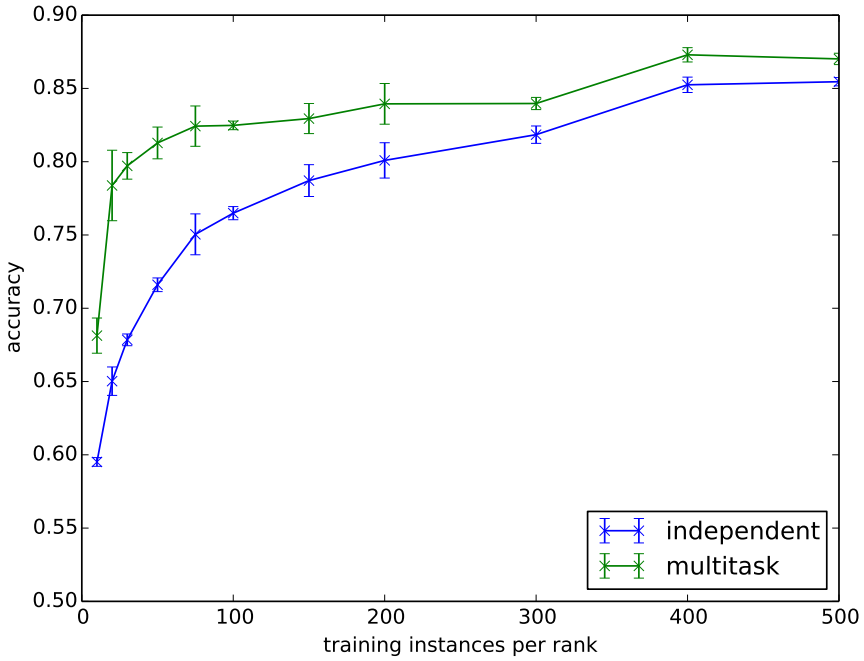


Figure 4.7: Accuracy of multitask learning compared to learning the tasks independently at different training set sizes. Bigger improvements can be seen at lower training set sizes.

in the generative model, while features $f_{17}, \ldots, f_{32}$ were not related to the class value.

First, we tested the hypothesis that multitask learning has a bigger impact when the number of data instances is lower. For this we kept the number of tasks fixed at $k = 45$ and varied the number of instances per rank from $n = 10$ through $n = 500$. The experiment was repeated 20 times for each setting and the results averaged (with standard deviations also shown). The multitask learning method MTL-FEAT was compared with logistic regression models developed independently for each task. Figure 4.7 shows the overall accuracy (fraction of correctly predicted instances across all tasks) at different values of n.

The results from Figure 4.7 confirm that multitask learning can substantially improve the prediction accuracy when only few learning instances are available for each task, as related data from other tasks can be used.

Learning each model independently requires more data to achieve comparable performance. With more data, independent learning did eventually reduce the difference with multitask learning and produce accurate models as well. Asymptotically, we expect the difference between both methods to approach zero, but due to time limitations we limited the experiments to the reported values of n.

The second experiment was performed to confirm the hypothesis that more tasks can lead to larger improvements by multitask learning. This time the number of learning instances per task was fixed at $n = 20$ and the number of tasks ranged from $k = 10$ to $k = 100$. Results were again averaged over 20 repetitions. The accuracy of the multitask representative MTL-FEAT and fitting independent logistic regression models are shown in Figure 4.8. Results clearly confirm that performance of multitask learning increases with the availability of more tasks. As expected, the performance when learning the tasks independently remained in a tight range throughout this experiment.

From both experiments we can conclude that the higher number of tasks and smaller task data sets resulting from one-against-one decomposition produce a set of learning problems where multitask learning is expected to have a greater impact.

4.4.2 Binary decomposition of nonlinear problems

The fact that one-against-one binary decomposition produces tasks with data instances from only two rank classes has another major advantage. It can reduce a nonlinear ranking problem to a set of linear binary classification tasks. Data instances of the same rank classes often form homogeneous clusters and the nonlinearity occurs due to the distribution of rank classes in the feature space. The decomposition breaks up the original complexity by considering only two clusters at a time, which can usually be much more easily separated. This allows simple and efficient base learners such as logistic regression or linear SVMs to be used for modelling decomposed tasks. In addition to achieving good performance, such construction also allows the ranking method and its predictions to be easily explained as both OAO binary decomposition and linear models have simple interpretations.

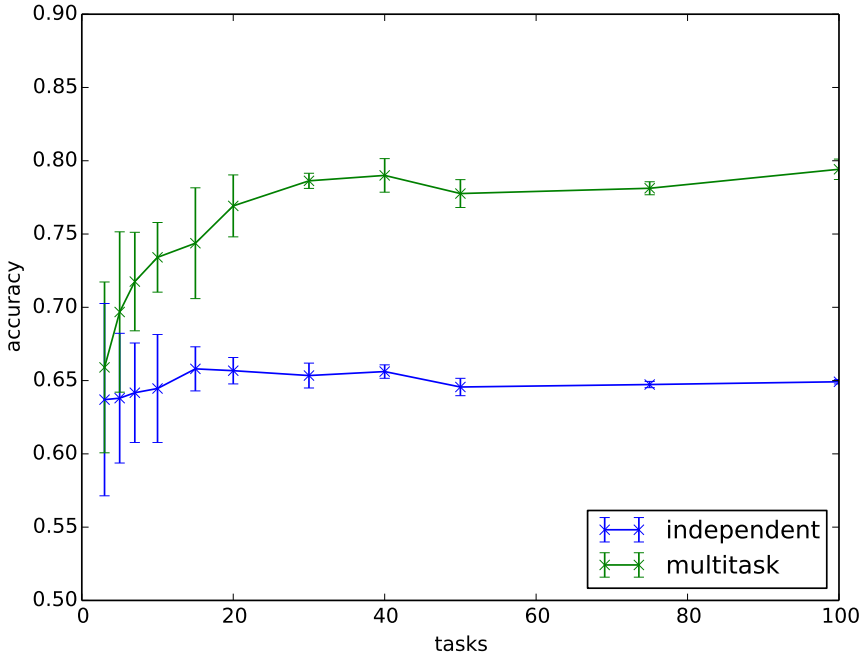


Figure 4.8: Accuracy of multitask learning compared to learning the tasks independently for different numbers of tasks. Standard deviations (shown with error bars) are larger at the beginning, because the results are averaged across less tasks.

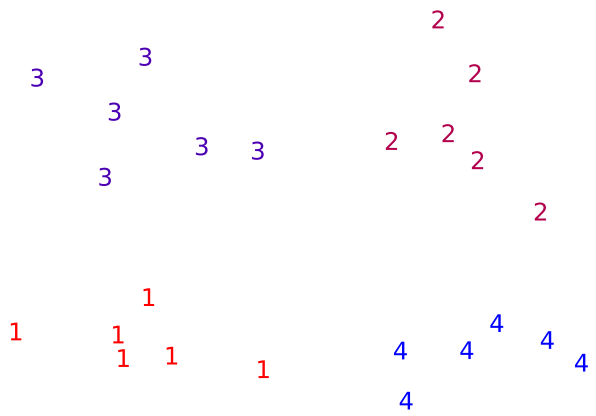


Figure 4.9: Data instances belonging to ranks 1, 2, 3 and 4 forming clusters in an "xor" pattern. While no direction ranks the data well, any two clusters can be linearly separated.

Let us illustrate this in Figure 4.9, where data instances are distributed in a two-dimensional space with clusters forming an "xor" pattern — the first two ranks occupying one diagonal (positions $(0, 0)$ and $(1, 1)$, where the value of the xor function would be 0), and the last two ranks occupying the other diagonal (positions $(0, 1)$ and $(1, 0)$, where the value of the xor function would be 1). A nonlinear method is needed to rank the points as no single direction corresponds to increasing ranks. However, OAO decomposition takes care of the nonlinearity and transforms the problem to simpler binary classification tasks that can be solved with linear methods.

To show the possible performance improvements in a more complex and slightly less engineered setting, we used the `normal-10` and `normal-100` data sets from the previous chapter (Section 3.5.3). The rank classes are randomly distributed in a higher dimensional space instead of being hand placed as in the example above. Refer to the description in the previous chapter (page 28) for details of the data model. OAO decomposition coupled with the linear logistic regression method used for binary classification tasks outperformed the global models of both ridge regression and even the specialized SVM^{rank} (see Table 4.2 for results). In the higher dimensional

Table 4.2: One-against-one (OAO) binary decomposition compared with ridge regression and SVM^{rank} on multipartite data with randomly distributed rank classes.

	`normal-10`		`normal-100`	
method	C	ρ	C	ρ
ridge-ind	0.61	0.31	0.69	0.49
SVM^{rank}	0.72	0.57	0.92	0.93
OAO	0.84	0.77	1.0	1.0

setting of `normal-100`, where any two clusters of data instances from the same rank could be linearly separated, the decomposition approach achieved a perfect ranking performance.

4.4.3 Analysis of BDMT

To analyse the proposed BDMT method we construct a more realistic ranking data model, which will allow us to study various properties of the learning algorithm and performance improvements. In real data sets the rank classes are rarely distributed randomly. For example, the differentiation of stem cells studied in the previous chapter occurs gradually and in consecutive stages. Therefore, as long as the sampling is frequent enough, successive ranks should be relatively similar and their instances close in the feature space. We sought to emulate this by placing the rank classes sequentially along a curve. Specifically, we chose the arc of an interrupted circle as shown in Figure 4.10 to model the structure of rank classes in a latent space. While it is easy to visualize the data in two dimensions, we generate a more general data set described with m features. A linear projection from this higher dimensional space into a hidden two-dimensional subspace would reveal the structure shown in Figure 4.10, but this projection is not given. Thus, the ranking is defined using two unknown hidden factors, which is a plausible setting that we can also expect in real-world data sets.

For the basic version of the `circle` data set we chose $m = 20$ features and sampled the instances from the hypercube $[-1, 1]^m$. The two factors are defined using the first and second set of 10 features as $F_1 = \sum_{i=1}^{10} \alpha_i f_i$ and $F_2 = \sum_{i=10}^{20} \alpha_i f_i$ with random weights $\alpha_i \in [-1, 1]$. Selecting only

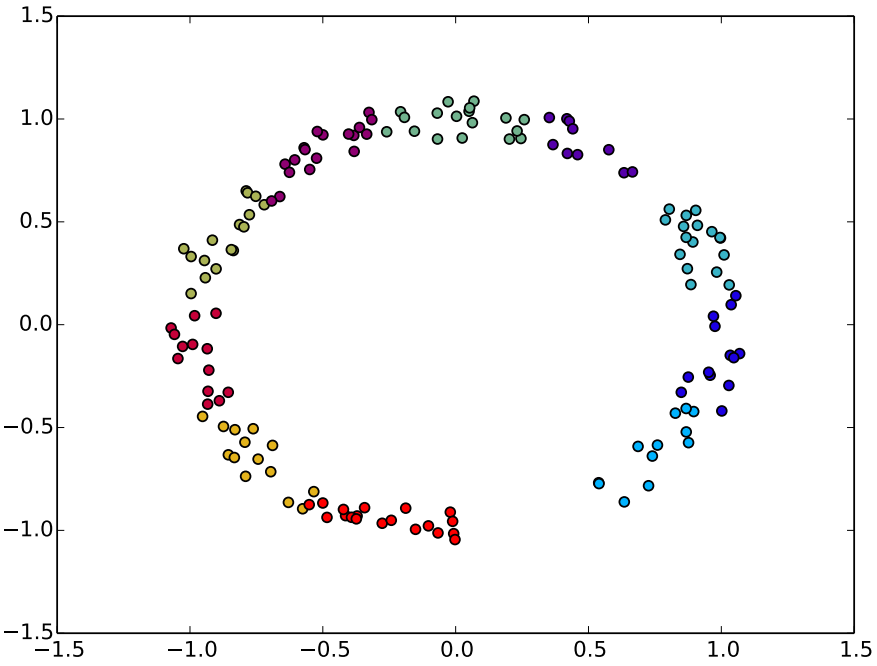


Figure 4.10: Sequence of rank classes distributed in a circular arc (ranks increase in a clockwise direction). All data instances of the same colour have the same rank.

Table 4.3: Average C-index scores and their standard deviations for four ranking methods evaluated on the variants of the `circle` data set. Ridge regression and SVM^{rank} build global ranking models, while one-against-one (OAO) and its enhancement using multitask learning (BDMT) decompose the problem and build many local models.

	$n = 100$		$n = 250$	
method	$m = 20$	$m = 30$	$m = 20$	$m = 30$
ridge	0.75 (0.03)	0.70 (0.04)	0.79 (0.03)	0.75 (0.02)
SVM^{rank}	0.81 (0.02)	0.79 (0.02)	0.81 (0.03)	0.81 (0.02)
OAO	0.79 (0.04)	0.71 (0.04)	0.94 (0.02)	0.86 (0.03)
BDMT	0.91 (0.01)	0.83 (0.02)	0.97 (0.01)	0.94 (0.01)

points whose distance to the origin in the projection plane is close to 1 ($\sqrt{F_1^2 + F_2^2} \approx 1$) results in the circular structure. To define $k = 10$ ranks, the angle is computed $\phi = \arctan \frac{F_1}{F_2} + \pi$ and converted to an integer $y = \lfloor \frac{\phi}{2\pi}(k+1) \rfloor$, where $\phi \in [0, 2\pi)$ and $y \in \{0, \ldots, k\}$. We remove instances with $y = k$ to obtain a gap in the circle, leaving k distinct rank classes. A second variant ($m = 30$) contained an additional set of ten random features unrelated to the ranking.

We compared four ranking methods: linear ridge regression (used to predict ranks), SVM^{rank} using a linear kernel, OAO decomposition with logistic regression for learning binary tasks, and the proposed method BDMT, which also uses logistic regression as a base learner. Each learning algorithm was given a set of n training instances for learning and then used to rank a separate test set of 100 instances. This was repeated 10 times and we report the average C-index scores along with the standard deviations. Different settings were tested by changing the amount of training data available ($n = 100$ and $n = 250$) and including 10 irrelevant features or not, for a total of $m = 30$ and $m = 20$ features, respectively. Results of all four possible combinations are reported in Table 4.3.

Several interesting observations can be made about the results. We start by confirming the expected benefit of our proposed approach, which achieved the highest performance scores in all settings. Especially when compared

with OAO, on which BDMT is based, we see significant improvements can be achieved by combining the decomposition approach with multitask learning. When comparing both decomposition approaches together against ridge regression and SVM^{rank}, which build a single global model, we notice an increased sensitivity to the training set size. OAO and BDMT decompose the ranking problem into many tasks that contain only a fraction of the data, which can lead to underdetermined learning problems that are hard to solve. This resulted in OAO achieving lower performance than SVM^{rank} despite its more suitable model. Because OAO is able to model the type of nonlinearity present in this data, it was able to achieve very good performance when enough data was available ($n = 250$), but could not fulfil its potential on a smaller data set ($n = 100$). Multitask learning used in BDMT alleviates this problem and significantly lowers the required number of data instances for decomposition to be competitive. However, the trend remains the same, and BDMT also experiences a drop in performance when data sets for decomposed tasks get too small. Contrast this with SVM^{rank}, that learns one model from all data and showed almost no difference in both settings ($n = 100$ and $n = 250$).

Introducing additional irrelevant features (variants with $m = 30$) results in a harder learning problem. To successfully estimate model parameters in a higher dimensional space would require more training instances. When the amount of training data is kept fixed, adding features has a similar effect as decreasing the number of training instances. This is confirmed by the results, which show the decomposition approaches (OAO and BDMT) responding more strongly to the change.

Performance curves

To better expose the relationships to various parameters we provide performance curves, which plot the achieved performance as a function of a selected parameter. All charts show the performance of four methods (ridge regression, SVM^{rank}, OAO, BDMT) in terms of the average C-index scores achieved in 20 repetitions along with the standard deviations.

Figure 4.11 shows the standard learning curves, which plot performance with respect to the number of training instances available. Evaluation was

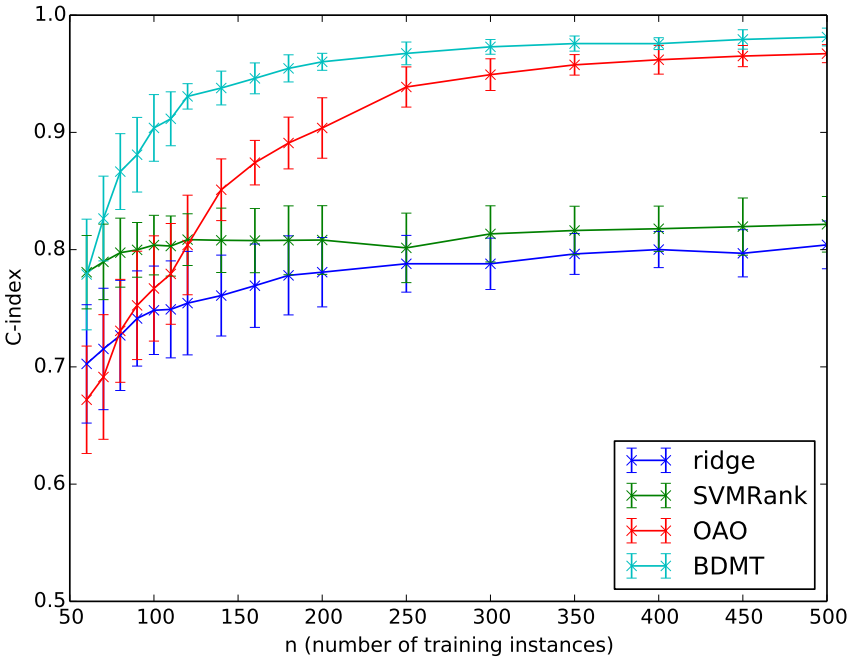


Figure 4.11: Learning curves showing performance of methods at different numbers of training instances.

performed using $m = 20$ features and $k = 10$ rank classes. The learning curves confirm our previous observations about OAO and BDMT, which struggle to fit local models when few training instances are available. The increased accuracy when more instances become available is exposed clearly, as is the faster climb of BDMT, which quickly rises above the $C = 0.95$ mark. We can also observe that ridge regression and SVM^{rank} have a lower limit to their performance due to the inability of their linear models to model the data. OAO and BDMT, although also using linear logistic regression base learners, can achieve almost perfect performance with sufficiently large data sets.

The second performance curve (Figure 4.12) was designed to study the impact of the number of irrelevant features. For this experiment, we started with $m = 20$ features used to determine the ranking model and added up to 80 irrelevant features. The number of training instances remained constant at $n = 250$ and the number of rank classes at $k = 10$. We see the performance

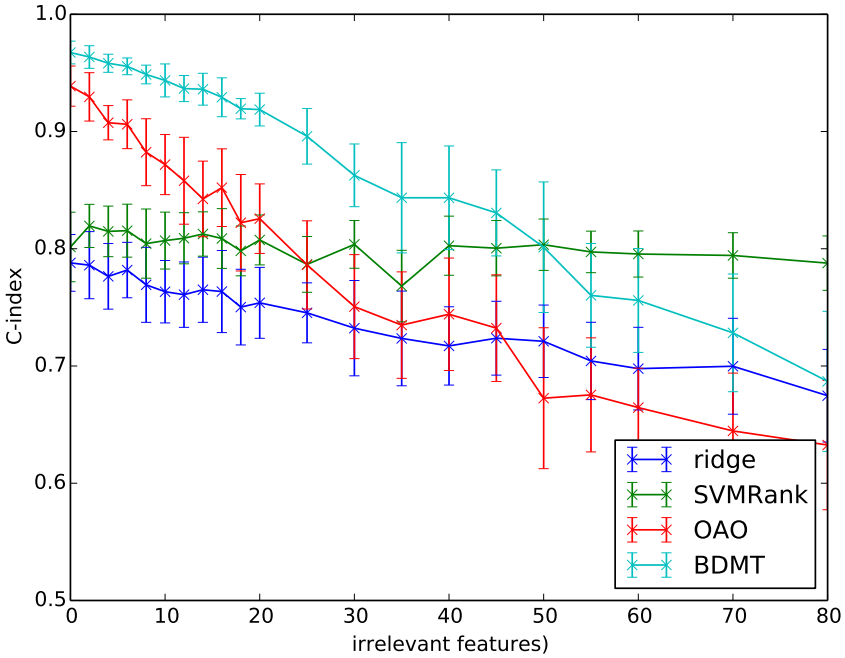


Figure 4.12: Learning curves showing performance of methods in the presence of irrelevant features.

steadily drops for most methods when the number of irrelevant features increases. An exception in this experiment was the SVM^{rank} method, which was able to keep the performance near its maximum achievable levels. We believe the amount of training data and the high bias of its global linear model allowed the ranking-oriented optimization to correctly estimate the parameters even in the presence of irrelevant features, at least for the tested ranges. The performance of OAO and BDMT drops more significantly as after the decomposition the individual tasks contain on average only $\frac{2}{k}n = 50$ data instances. Therefore, adding a fixed number of features has a bigger impact on the proportion between the number of training instances and the number of parameters that need to be estimated.

The last performance curve (Figure 4.13) shows how k, the number of rank classes in multipartite ranking data, influences the performance of different methods. The number of all training instances was equal to $n = 10k$, thus keeping the average sizes of rank classes constant. The number of features was fixed at $m = 20$. This is perhaps the most interesting chart, as the curves do not only differ quantitatively (speed and amount of change), but also qualitatively (direction of change). As we have shown for standard multitask learning settings, more tasks can lead to bigger improvements over single task learning. The same property is also confirmed in this experiment, where the performance of BDMT increases with more rank classes, because that relates to more tasks after the decomposition. As expected, OAO, which learns each task independently, does not exhibit a strong dependence on the value of k. This difference between BDMT and OAO presents an important observation on the nature of BDMT's enhancement. It is able to retain the benefits of decomposition, but can partly avoid the disadvantages due to fragmentation of data. This effect, as is seen from the results, increases with more ranks and thus so does the advantage of BDMT over OAO. On the other hand, the performance of SVM^{rank} clearly decreases with more rank classes, especially at low values of k. We believe that this is because the linear model used in SVM^{rank} is better able to approximate data with fewer ranks. As k increases, so does the complexity and nonlinearity of the data, limiting the best performance SVM^{rank} is able to achieve. The same observations should also apply to ridge regression. But, as was seen in previous experiments, it does not maximize the potential of a linear model unless it

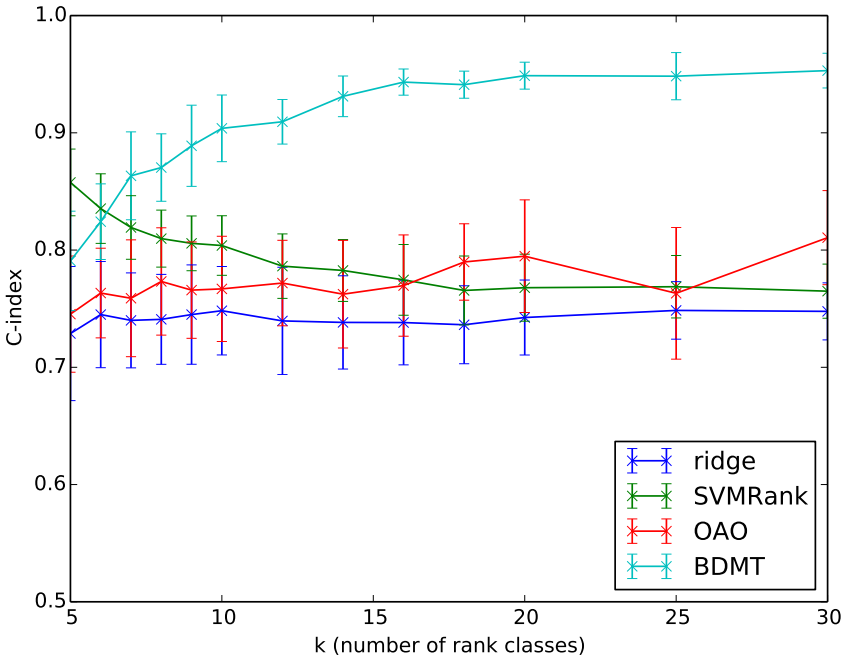


Figure 4.13: Learning curves showing performance of methods evaluated on multipartite data sets with different numbers of ranks.

has more training data. So the disadvantage of less training data overall (number of instances per rank class is constant) probably negates the possibly higher performance that could be achieved due to simpler data. Note that the decomposition methods (OAO and BDMT) do not suffer from this increase in data complexity due to more ranks, because they do not construct a global model, but only ever consider two ranks at a time.

Aggregation variants

We tested alternative variants that can be used to aggregate results of individual task models. In addition to our default voting approach, which selects the most frequently predicted rank class, we also tested its weighted counterpart. In it, each classifier contributes its support for a class proportional to the computed margin of the classified data instance, instead of a simple 0/1 vote. The third considered aggregation function assigns a utility score to each instance equal to the sum of predictions in favour of the higher class [8]. Ranking by utilities should result in instances that were assigned to the higher class more often and more confidently to also be ranked higher.

We compared all three aggregation approaches using the `circle` data set ($n = 100$, $m = 20$) and the results showed that both alternatives performed worse than the chosen majority voting scheme. The weighted voting and sum of predictions variants achieved average C-index scores of 0.88 and 0.82, respectively, compared with 0.91 achieved with the majority vote.

Both alternatives use margins instead of predicted values, which is often beneficial, but might be one of the reasons for their underperformance in this case. In one-against-one decomposition individual models compare two rank classes and are trained with only the relevant subset of instances. However, at prediction time we use all models when considering an unknown new instance and many of them are deciding between two incorrect choices. This is also known as the "non-competence" problem [8] and can be mitigated to a degree by the ensemble aggregation of the results. Although there are wrong predictions, the correct one is still expected to be the most frequent. This does not hold when margins are used. Even in the ideal case when all models make correct predictions, the correct class could receive the maximum possible $k - 1$ votes with individual margins close to $+1$. But a

non-competent model, could predict another (incorrect) class with a margin of +100 and decide the weighted vote with that prediction. Unfortunately, large margins for incorrect predictions are not unexpected, because a data instance that does not belong to the classes on which the model was trained can lie in another part of the feature space, far from the decision boundary.

We believe further study of aggregation could lead to other solutions with better performance. However, the simple majority voting scheme has been frequently used and performs well enough for our current requirements. We therefore limited other experiments in this work to this aggregation approach.

ESC differentiation

Biological experiments often involve only a few repetitions or replicates resulting in data sets that are too small for some approaches. The embryonic stem cell (ESC) differentiation data that we analysed in the previous chapter also has this problem, with most data sets containing only three or four instances per rank class for a total of 18–44 data instances. Nevertheless, we include here an evaluation of decomposition approaches (OAO and BDMT) on ten GEO data sets under the same conditions as were used in section 3.6.3 (with results in Table 3.3). As results have shown that both feature selection methods used previously perform similarly, we only report results with the faster FC method. Table 4.4 contains C-index scores of the leave-pair-out internal validation (on rank classes) for the newly tested methods together with those of SVM^{rank} and the best performing MCE-euclid for comparison.

Results show that in these settings decomposition approaches can not achieve the same performance as the global models. Taking into consideration that a task usually consisted of three positive and three negative data instances and required the estimation of weights for $m = 1000$ features, the results are not unexpected. The ability to model nonlinearly distributed rank classes might provide an advantage in similar problems. However, this process could be modelled well even by simple linear methods such as PCA and PLS (see complete results from the previous chapter) indicating strong linear patterns. While not competitive compared to global models, BDMT

Table 4.4: Comparison of ranking methods on ten stem cell differentiation data sets distinguished by extremely low data set sizes. Decomposition approaches (OAO and BDMT) were not able to compete with global models (SVM^{rank} and MCE).

	OAO	BDMT	SVM^{rank}	MCE-euclid
GDS2431	0.674	0.696	0.844	0.993
GDS2666	0.818	0.824	0.915	0.972
GDS2667	0.818	0.820	0.907	0.964
GDS2668	0.796	0.798	0.889	0.897
GDS2669	0.741	0.741	0.883	0.895
GDS2671	0.812	0.830	0.893	0.964
GDS2672	0.772	0.788	0.897	0.939
GDS2688	0.552	0.560	0.551	0.750
GDS586	0.786	0.774	0.972	0.853
GDS587	0.603	0.577	0.857	0.825

did compare favourably to OAO (its score was lower only on the last two out of ten data sets), but the improvements were mostly minor.

Discretized regression data sets

Due to a lack of standard ordinal benchmark data sets we use regression data sets and discretization to obtain a collection of data sets that can be used to evaluate ranking methods. Table 4.5 lists 20 data sets from Orange (`http://orange.biolab.si/datasets.psp`) and Weka (`http://www.cs.waikato.ac.nz/ml/weka/datasets.html`) repositories, of which the first 16 are discretized regression data sets and the last 4 ordinal by nature. The data sets vary in the number of instances and contain different numbers of categorical and/or numerical features. Each categorical feature with k values was replaced with $k-1$ binary features. The final data set sizes are shown in Table 4.5. The numerical classes of regression data sets were discretized into ordinal bins of equal sizes, targeting 20 instances per bin, but limited to a maximum of 20 bins.

C-index scores were computed using 5-fold cross-validation for the BDMT method and compared with scores of OAO and SVM^{rank}. As can be seen

Table 4.5: Evaluation on 16 discretized regression and 4 ordinal data sets. For each data set the number of data instances (n) and features (m) is reported along with the C-index scores of three ranking methods (OAO, BDMT and SVM^{rank}). Methods were ranked by performance on each data set. The average ranks of methods are given in the last row.

	size		C-index		
data set	n	m	OAO	BDMT	SVM^{rank}
abalone	4177	9	0.799	0.797	0.815
bank8FM	8192	8	0.947	0.948	0.951
bank32NH	8192	32	0.757	0.757	0.811
baskball	96	4	0.670	0.713	0.704
bodyfat	252	14	0.789	0.795	0.909
cloud	108	8	0.749	0.883	0.927
delta_ailerons	7129	5	0.763	0.766	0.834
galaxy	323	4	0.800	0.954	0.949
kin8nm	8192	8	0.744	0.745	0.744
lowbwt	189	16	0.715	0.785	0.757
prostate	97	8	0.705	0.744	0.829
puma8NH	8192	8	0.764	0.763	0.744
pwLinear	200	10	0.855	0.859	0.860
pyrim	74	27	0.754	0.758	0.846
servo	167	10	0.855	0.888	0.861
stock	950	9	0.883	0.947	0.895
ERA	1000	4	0.715	0.715	0.738
ESL	488	4	0.920	0.923	0.962
LEV	1000	4	0.819	0.815	0.874
SWD	1000	10	0.750	0.760	0.819
$\overline{rank}$			2.75	1.8	1.45

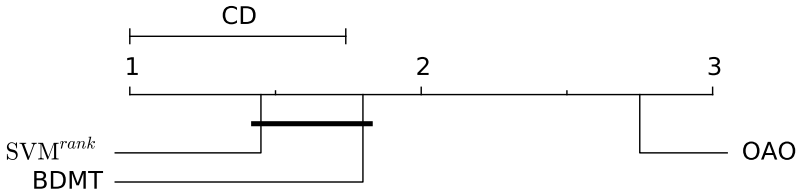


Figure 4.14: Critical difference graph using the Friedman-Nemenyi test ($p < 0.05$) shows that BDMT statistically significantly outperforms OAO, while the difference between BDMT and SVM^{rank} is not statistically significant.

from results reported in Table 4.5, BDMT either improved upon the results of OAO (often noticeably, *e.g.* galaxy, stock, lowbwt, cloud), or retained approximately equal performance, but did not perform much worse on any data set. This is another confirmation showing that the proposed use of multitask learning can improve the learning of decomposed tasks and thus the final ranking results. The different approach of SVM^{rank} was often able to achieve better results than both OAO and BDMT, however, the improved decomposition approach of BDMT was close enough to be considered competitive over the whole collection of data sets. A statistical study of the differences in average ranks, achieved by methods over all data sets, revealed that both SVM^{rank} and BDMT performed significantly better than OAO (Friedman-Nemenyi test as described by Demšar [68], $p < 0.05$). The difference between SVM^{rank} and BDMT was, however, not found to be significant. The critical difference graph (Figure 4.14) shows the comparison of average ranks.

4.5 Additional benefits of the proposed approach

The main motivation for designing BDMT was to improve the process of learning the set of tasks that result from binary decomposition. We have shown that the proposed approach, with its addition of multitask learning, improves the predictive performance of the standard implementation of OAO decomposition. However, predictive performance is not always the only criterion that influences the choice of the most suitable learning method. A faster method may be chosen, even if it is not the most accurate, when time

is limited. In other situations stability and consistent models for similar inputs are desired, or the ability to interpret the model's predictions and gain insight into the underlying process. In the following sections we discuss these additional benefits, which also result from BDMT's design.

4.5.1 Stability

One-against-one decomposition splits the ranking problem into many small tasks, which has been shown to have its advantages (easier to learn simple models) and disadvantages (harder to estimate parameters because of a lack of data). Besides the possible decrease in accuracy, the disadvantages include issues with the stability of models. When learning from small data sets, many of the available data instances define the decision boundary and even minor perturbations of the training set might lead to noticeable changes in the final models. The proposed enhancement with multitask learning adds additional regularization based on data from other tasks and diminishes the influence a few instances can have on an individual model. This results in a more stable learning process and consistent performance.

Results reported in the previous section (Table 4.3) show that BDMT consistently had the lowest standard deviations of scores achieved in repetitions of the same experiment (data was randomly generated with different random seeds). The consistency improved considerably from the basic OAO, which had the largest standard deviations of scores, especially when less training data was available. Perhaps surprisingly, BDMT even achieved more consistent results than ridge regression and SVM^{rank}, despite the fact they only estimate one model using all available data. Furthermore, the BDMT performance curves shown in Figures 4.11–4.13 appear more smooth than those corresponding to OAO.

4.5.2 Efficiency

Ranking methods using binary decomposition are able to achieve comparable predictive performance to complex ranking methods with lower training times. Similar speed benefits have been observed in the use of decomposition for multiclass classification. There, binary decomposition does increase the required training times compared with learning from binary (two-class)

data of the same size. But modifying an algorithm to solve a multiclass problem directly can make it even more computationally expensive. For the case of SVMs, it has been shown that methods using decomposition require less time for training than several algorithms that solve a single optimization problem [44].

We came to similar conclusions based on the performed ranking experiments. While training a single regression method can be the fastest option, more time is usually needed to train specialized ranking methods that are needed for more complex data sets. In these situations, learning multiple models with smaller data sets can be faster than building a single model that is more complex and has more training data. Because of the differences in implementations we do not provide an exact comparison of training times. However, during various experiments OAO and BDMT, which we implemented in Python, consistently achieved lower times than SVM^{rank}, which is implemented in C. On a desktop PC (Core i7-2600 3.40GHz) the orders of magnitudes for training times on a data set with $n = 250$ instances and $m = 30$ features were approximately 0.1s for OAO, 1s for BDMT and up to 100s and more for SVM^{rank}, whose speed depended strongly on the value of its regularization parameter. The publicly available implementation of SVM^{rank} does not have the optimal time complexity that has been shown to be possible [38], but is still a popular benchmark [11, 8].

There is a computational penalty for using multitask learning in BDMT compared with OAO. While OAO has to train a model once for every binary task, BDMT uses MTL-FEAT, which trains all models in an iterative procedure. In each iteration, all models have to be retrained after a space transformation. BDMT is therefore slower than OAO by approximately a factor equal to the number of iterations needed until convergence. In our experiments we observed 10–50 iterations were usually needed (the authors reported 20–100 was sufficient in their experiments).

A benefit common to both OAO and BDMT is that the process of training task models, where most of the time is spent, can be easily parallelized. This can reduce the time needed considerably and presents another advantage over single optimization algorithms, where parallelization is harder or impossible.

4.5.3 Interpretability

The usefulness of a predictive model can often depend on the possibility to interpret the predictions and understand the process that led to them. We will show that BDMT offers good interpretability even when used to model complex data. BDMT consists of two components, binary decomposition and multitask learning, which have clear interpretations of their own. These can supplement each other to provide a more complete understanding of the whole method.

Recovery of hidden factors

BDMT incorporates multitask feature learning and can access the extracted knowledge that becomes available with it. The new features inferred from all tasks do not act only as a regularizer that improves prediction performance, but also provide insight into the problem domain by recovering a small set of features that are sufficient to construct all models and define the ranking process.

We performed an experiment to test the usefulness of feature learning. The data instances in the `circle` data set are ranked according to their position in a two-dimensional subspace, which is randomly defined at generation time and then "forgotten". We test if the two hidden factors, which define the subspace and therefore the ranking, can be recovered from the observable data only, without any additional information.

We trained the BDMT algorithm on a data set with $n = 150$ instances described by $m = 20$ input features. In addition to the learned weights, we also obtain a transformation into the new (learned) features, with properties similar to those of the principal components found by PCA. That is, the features are orthogonal and the first few contain most of the necessary information (in this case for classification instead of retaining the variance in the data). Figure 4.15 shows the data instances projected into the true hidden subspace where the ranking was defined, and into the subspace defined by the first two learned features. Although not identical, the projections are very similar, with the learned projection clearly showing the structure of the data and keeping the rank classes well separated.

We wanted to confirm that the learned projection is truly useful and the

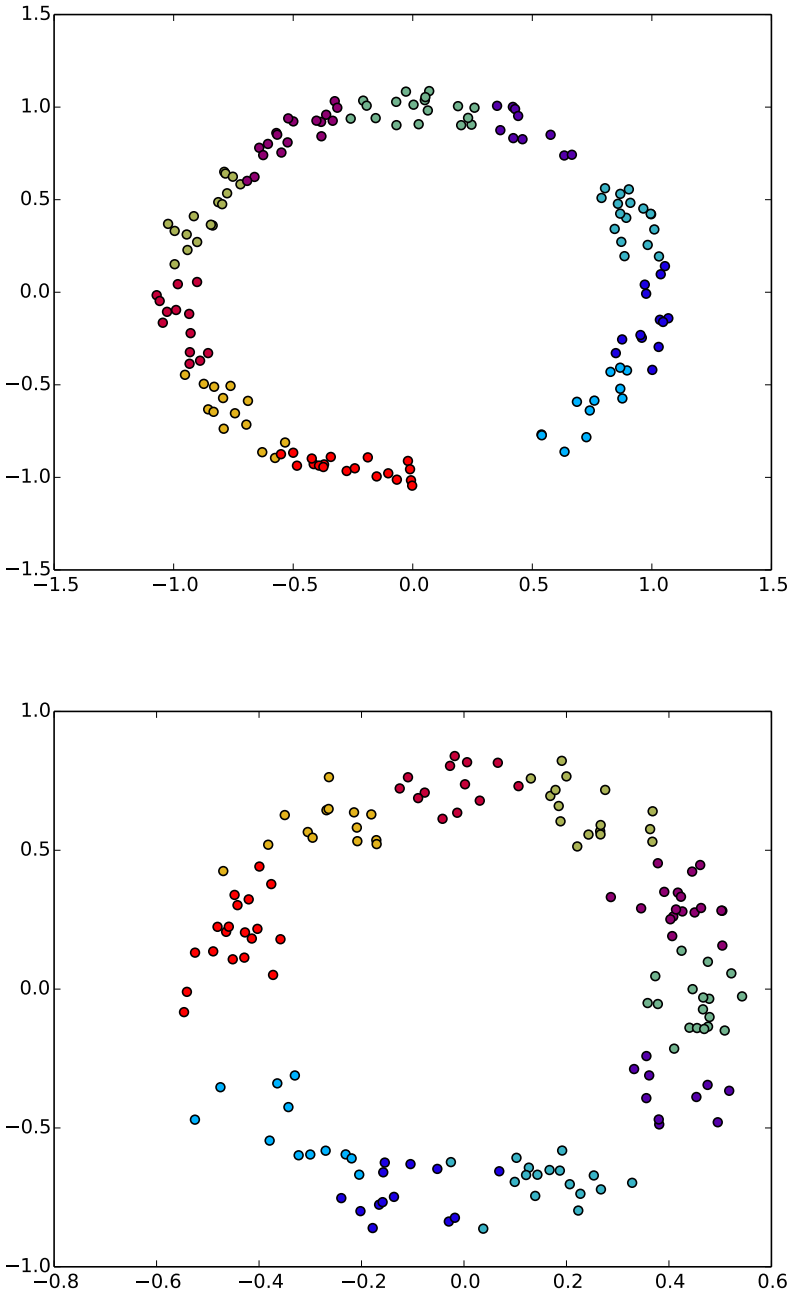


Figure 4.15: Training instances shown in the hidden subspace where the data was defined (top), and in the subspace learned by BDMT using the same data set (bottom).

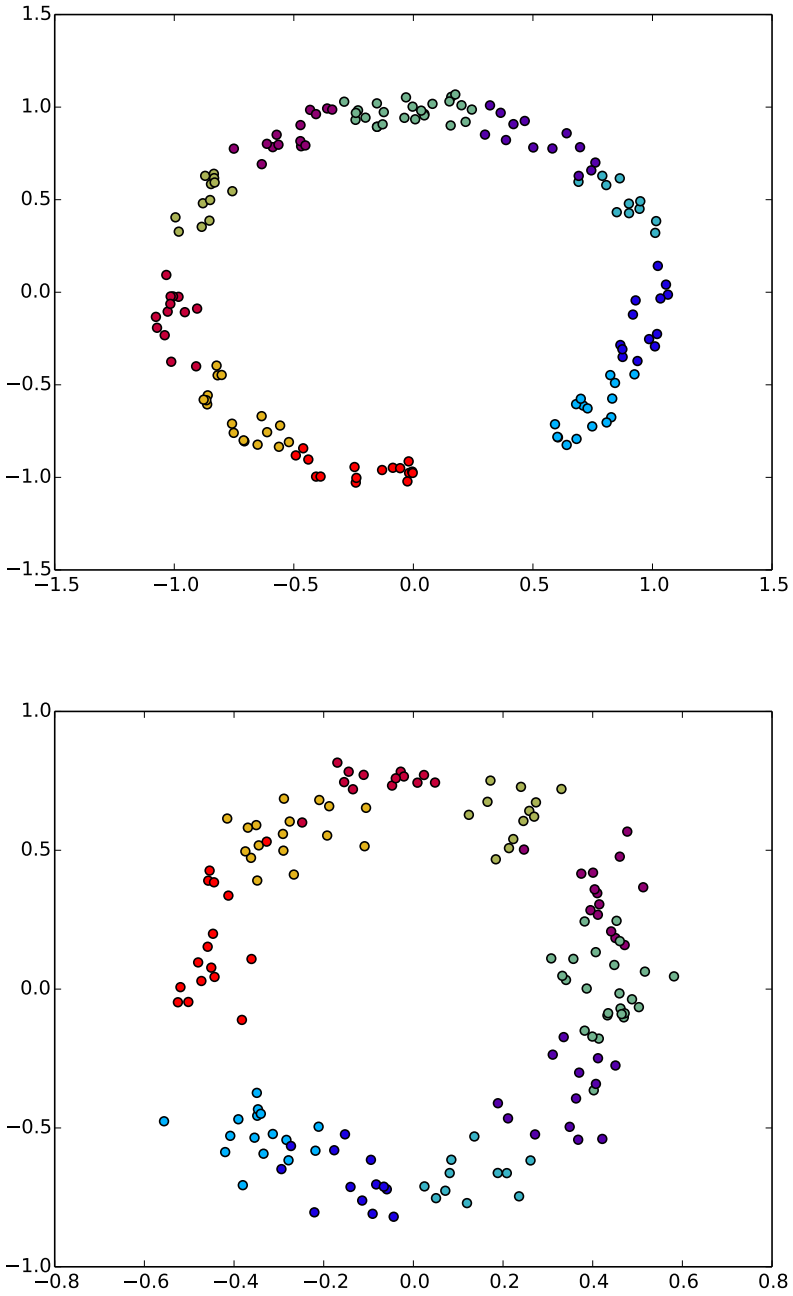


Figure 4.16: Test instances shown in the hidden subspace where the data was defined (top), and in the subspace learned by BDMT on the training set beforehand (bottom).

Table 4.6: Importance of the first six learned features measuring their weights in models of decomposed tasks.

	learned feature					
	1	2	3	4	5	6
importance	0.518	0.386	0.035	0.024	0.019	0.010

separation is not a consequence of overfitting the training data for which the ranks were available during learning. For this reason, a separate test data set was projected to the same learned subspace (Figure 4.16). Close observation reveals more overlap of rank classes than when projecting training instances, which would imply some degree of overfitting to the training data. However, it is still easy to recognize the circular structure and the succession or rank classes.

We can conclude that the recovery of hidden factors allows an informative visualization of the data. Such visualizations or a direct study of the recovered factors could be very beneficial to a researcher studying an unknown process.

Although we have used just the first two features, the algorithm learns $\min(m, k)$ new features, where m is the number of original features and k the number of tasks. The optimization tries to ensure only the first few features are enough to construct accurate models, but it is not known in advance how many will be needed. Fortunately, the "importance" of features is also computed during optimization and can be used to assess the dimensionality of the subspace relevant for ranking. It is measured by the L_2 norm of the vector of weights for a selected feature across all tasks and normalized so that all importances sum up to 1. Table 4.6 lists the importance of the first six learned features from the previous experiment. We could clearly conclude that only the first two features should be investigated, even if we had no prior knowledge of the data generation process.

Task similarity

Studying and comparing models of tasks formed by binary decomposition can help to interpret the complete ranking model and elucidate the data patterns. In this respect, BDMT is similar to OAO, apart from its ability

of expressing task models using the newly learned features. The models can still be expressed with original features as well, but the smaller number and possibly meaningful interpretations of the new ones are usually preferred.

A single linear model can be easily understood, with its weights defining the direction in space in which the ranking progresses. If the ranking process is indeed linear (rank classes distributed along a line), what would the models of decomposed tasks look like? Each model (i, j) would have a similar dividing hyperplane with its normal corresponding to the direction from the lower to the higher rank class. In OAO, weight vectors of all models would be similar. In BDMT it would be even more evident as the most important feature would represent this direction and would be the main factor in all models.

We already showed that OAO decomposition can also model nonlinear relationships. Can these be similarly explained? For such data the weight vectors of different task models would be different. One possibility is to observe tasks $(i, i+1)$ that compare consecutive ranks. Their weight vectors give the main direction of change from the i-th to the $(i+1)$-th rank and could be called the ranking gradient at rank i. Analysing the gradients can be a step towards understanding the ranking process and can be used even when the data can not be visualized (*e.g.* it has more than two relevant factors).

As an example consider the weights learned in the `circle` experiment. We have already seen that only the first two features learned by BDMT are important, so in Table 4.7 we report only their weights (other weights were smaller). Note that by our definition the instances of the lower rank form the positive class, so the weight vectors point from the higher to the lower rank. The plot of the data in the learned subspace (Figure 4.15, bottom) can be used to help understand the weights. For example, the first rank class (red points in the Figure) is distinguished from the second (yellow points) by lower values of both factors corresponding to the vector $[-9.9, -8.8]$. Computing the angle between the weight vectors of tasks $(0, 1)$ and $(6, 7)$ would show that the direction is almost opposite (172°, 168° if complete weight vectors are used). The angles between consecutive gradients, which vary around 32°, would reveal that the changes in this ranking model happen gradually. In another setting, the gradient could remain the same for the first

Table 4.7: Weights for the first two learned features in tasks comparing consecutive ranks. A gradual change of direction can be seen.

	binary task								
	$(0,1)$	$(1,2)$	$(2,3)$	$(3,4)$	$(4,5)$	$(5,6)$	$(6,7)$	$(7,8)$	$(8,9)$
w_1	-9.9	-14.4	-15.3	-11.3	-4.1	6.7	10.4	14.3	11.5
w_2	-8.8	-3.1	2.3	8.6	13.6	10.5	6.9	0.1	-4.1

few ranks and then change suddenly, which would provide us with valuable insight and could indicate a second process is activated at that stage.

Chapter 5

Conclusion

Ranking is developing into a distinct field of machine learning and has a great potential to be used in real-world applications. The field is younger and not as established as, for example, classification or regression, but with current research interest in this topic, ranking should soon be offered as a common tool in machine learning suites. This thesis contributes to the development of the field of ranking, and in particular to its subfield of *multipartite instance ranking*. Our major contributions are: a comparative analysis of existing approaches with a ranking-specific evaluation framework, and a proposal of a new ranking approach that combines binary decomposition and multitask learning.

In the first part of the thesis we categorize different learning approaches, some proposed in other fields, and study their use for instance ranking. We introduce an evaluation framework combining ranking performance measures and validation techniques, which is used for subsequent experimental evaluations. We then show how ranking can be used in practice for solving a relevant problem from the field of bioinformatics. We cast the problem of embryonic stem cell differentiation as multipartite instance ranking, which allows us to use different ranking approaches and compare their success at this task. The results show that the inferred models trained from gene expression data can predict a ranking of biological samples, which accurately corresponds to their stages of differentiation. This is a useful contribution to bioinformatics, where marker genes used to determine the stages were often studied and selected manually, and to the field of ranking as a critical

comparison of its approaches and their potential applications.

In the second part of the thesis we focus on the family of ranking methods based on binary decomposition and aim to address some of its shortcomings. We propose the idea of enhancing the process of learning models for decomposed tasks with the use of multitask learning. This allows the use of the structure shared between similar tasks to arrive at better models. The combination of binary decomposition and multitask learning to improve the design of ranking methods is the main contribution of the dissertation. Based on the conclusions of our study into properties of both components, we propose a concrete implementation of the principle in the form of a new method called BDMT.

We perform an extensive analysis of BDMT and show that it successfully combines one-against-one (OAO) decomposition with multitask feature learning (MTL-FEAT). An important observation is that BDMT almost always outperforms the straightforward use of OAO. It often also outperforms other established ranking methods, even when OAO is unable to achieve this by itself. Its usefulness, compared with other approaches, depends on the properties of the training data set. In particular, and as shown in the experiments, the accuracy of BDMT depends on the number of training instances per rank class, which has to be sufficient to prevent the individual tasks from becoming ill-defined. An advantage of BDMT over global models that we would like to highlight is its ability to model nonlinear patterns. This can be achieved without the cost of extra model complexity through the use of OAO decomposition and linear base learners.

In addition to the improvements to predictive performance, we show that BDMT is also distinguished by models that can be easily interpreted. This makes it very useful for studying new problems with relatively unknown underlying processes, when more focus is on exploratory analysis.

Some details of the proposed method remain open for further study. One of them is the aggregation function used to combine the results after decomposition. As alternatives to the proposed majority voting, we tested some weighted variants that were not successful. The "non-competence" problem (models deciding between two incorrect choices) could perhaps be solved using one-class distribution estimation to gauge the competence of models and incorporate this in the aggregation. Another aspect that we

wish to see in future work is the use of this method and other decomposition ranking approaches in more practical applications. Our focus was mostly on the analysis of various properties and the proof of concept for the idea of combining binary decomposition and multitask learning. With its potential demonstrated, we would like to see how it can benefit different applications.

The described implementation (BDMT) is also not the only possible combination of binary decomposition and multitask learning. We do not see our work merely as the proposal of a specific method, but of a principle that can spawn a whole family of methods. A possible method design that we find promising combines the thresholding approach of binary decomposition and structured multitask learning. The main motivation for this combination are task similarities for successive thresholds. Deciding if an instance belongs to a rank higher than i is similar to deciding if it belongs to a rank higher than $i+1$, as the answer differs only for instances from rank $i+1$. A regularized optimization that would penalize large differences in parameters of models for successive tasks could use this structural similarity to arrive at better parameter estimates.

Appendix A

Additional results of the ESC differentiation study

A.1 Gene selection

Two top ranked methods, MCE-euclid-FC and PCA-FC, were evaluated with different numbers of informative genes (50, 100, 500, 1000, 2000 and 5000). The variations were ranked by the achieved performance on individual data sets and the average ranks computed. Figures A.1 and A.2 plot the performance with respect to the number of used genes for MCE-euclid-FC and PCA-FC, respectively. Results show that using too many or too few genes leads to lower performance.

A.2 Internal validation

Tables A.1, A.2 and A.3 show the entire set of internal validation results, including all possible combinations of gene selection, modelling methods, and distances for MCE.

A.3 External validation

Tables A.4, A.5 and A.6 show the entire set of external validation results, using all possible pairs of data sets for training and testing.

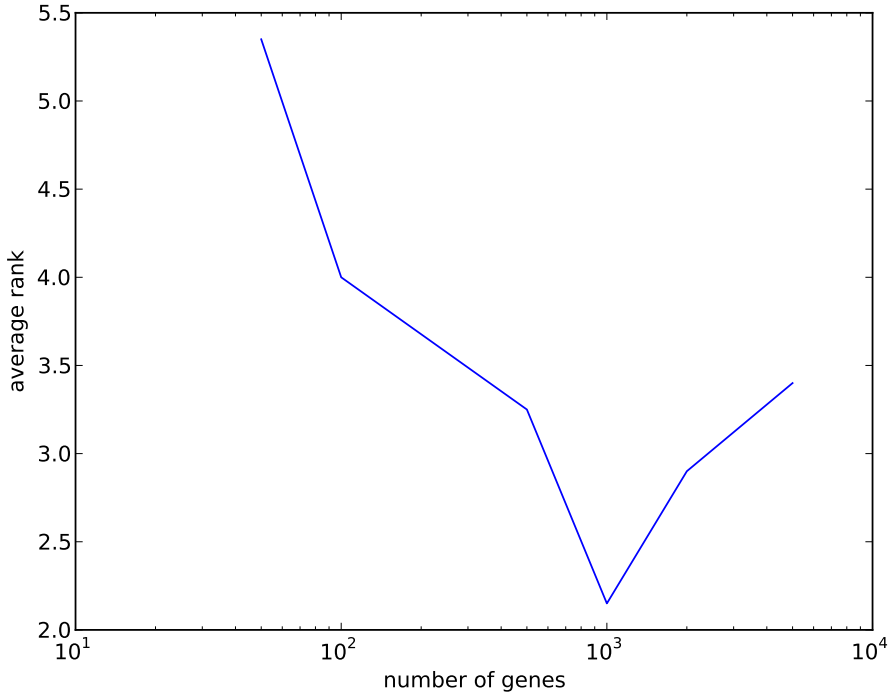


Figure A.1: The average rank (across different data sets) of MCE-euclid-FC with respect to the number of selected genes.

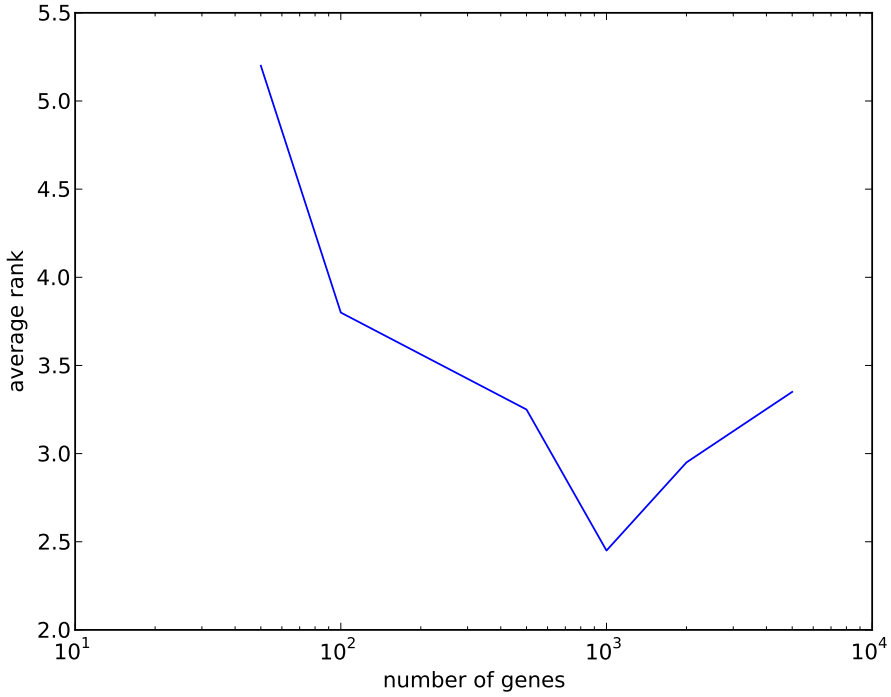


Figure A.2: The average rank (across different data sets) of PCA-FC with respect to the number of selected genes.

Table A.1: C-index scores of leave-pair-out internal validation for each method on 5 of 14 data sets used in our study. Results on the other data sets can be found in Tables A.2 and A.3. For each data set the methods are ranked according to the data set-specific C score. Methods' average ranks and average C scores over all data sets are reported in Table A.3.

	GDS2431	GDS2666	GDS2667	GDS2668	GDS2669
MCE-euclid-FC	0.993	0.972	0.964	0.897	0.895
PLS-AREA	0.867	0.945	0.913	0.923	0.903
MCE-corr-AREA	0.941	0.964	0.929	0.903	0.887
PCA-FC	0.874	0.974	0.899	0.931	0.909
PCA-AREA	0.896	0.952	0.921	0.929	0.889
PLS-FC	0.859	0.962	0.883	0.905	0.909
MCE-corr-FC	0.993	0.972	0.945	0.897	0.810
SVMRank-FC	0.844	0.915	0.907	0.889	0.883
SVMRank-AREA	0.859	0.883	0.881	0.893	0.905
MCE-euclid-AREA	0.941	0.966	0.941	0.901	0.877
PLS-FC-time	0.859	0.966	0.786	0.766	0.903
PLS-AREA-time	0.867	0.952	0.766	0.760	0.798
PLS-Markers	0.896	0.899	0.885	0.893	0.855
MCE-euclid-Markers	0.867	0.943	0.861	0.804	0.832
Pathrecon	0.956	0.840	0.887	0.859	0.812
PCA-Markers	0.600	0.911	0.869	0.877	0.842
MCE-corr-Markers	0.800	0.935	0.836	0.792	0.780
SVMRank-Markers	0.852	0.834	0.836	0.826	0.711

Table A.2: C-index scores of leave-pair-out internal validation for each method on 5 of 14 data sets used in our study. Results on the other data sets can be found in Tables A.1 and A.3. For each data set the methods are ranked according to the data set-specific C score. Methods' average ranks and average C scores over all data sets are reported in Table A.3.

	GDS2671	GDS2672	GDS2688	GDS586	GDS587
MCE-euclid-FC	0.964	0.939	0.750	0.853	0.825
PLS-AREA	0.909	0.812	0.581	0.944	0.884
MCE-corr-AREA	0.919	0.891	0.724	0.845	0.810
PCA-FC	0.822	0.794	0.732	0.948	0.942
PCA-AREA	0.824	0.798	0.738	0.944	0.937
PLS-FC	0.911	0.764	0.588	0.948	0.857
MCE-corr-FC	0.931	0.596	0.759	0.849	0.799
SVMRank-FC	0.893	0.897	0.551	0.972	0.857
SVMRank-AREA	0.859	0.913	0.542	0.964	0.862
MCE-euclid-AREA	0.911	0.828	0.728	0.817	0.820
PLS-FC-time	0.871	0.782	0.423	0.960	0.815
PLS-AREA-time	0.863	0.842	0.392	0.952	0.841
PLS-Markers	0.855	0.873	N/A	0.913	0.519
MCE-euclid-Markers	0.907	0.903	N/A	0.520	0.651
Pathrecon	0.919	0.784	N/A	0.897	0.804
PCA-Markers	0.887	0.776	N/A	0.730	0.519
MCE-corr-Markers	0.810	0.913	N/A	0.520	0.651
SVMRank-Markers	0.822	0.683	N/A	0.722	0.720

Table A.3: C-index scores of leave-pair-out internal validation for each method on 4 of 14 data sets used in our study. Results on the other data sets can be found in Tables A.1 and A.2. For each data set the methods are ranked according to the data set-specific C score. Methods' average C scores and average ranks over all data sets are reported in the last two columns.

	F	G	N	Z	$\overline{C}$	$\overline{rank}$
MCE-euclid-FC	0.983	0.733	0.905	0.533	0.881	6.500
PLS-AREA	0.950	0.933	0.905	0.950	0.911	7.115
MCE-corr-AREA	1.000	0.717	1.000	0.683	0.884	7.115
PCA-FC	0.883	0.867	0.857	0.983	0.899	7.346
PCA-AREA	0.883	1.000	0.917	0.917	0.908	7.538
PLS-FC	0.883	1.000	0.905	0.983	0.905	7.654
MCE-corr-FC	1.000	0.733	1.000	0.617	0.857	8.000
SVMRank-FC	0.900	1.000	0.905	0.983	0.911	8.038
SVMRank-AREA	0.967	1.000	0.857	0.933	0.906	8.269
MCE-euclid-AREA	0.983	0.700	0.905	0.583	0.860	8.808
PLS-FC-time	0.883	1.000	0.905	0.983	0.883	9.577
PLS-AREA-time	0.950	0.933	0.905	0.950	0.875	10.231
PLS-Markers	0.833	0.817	0.964	0.983	0.860	10.269
MCE-euclid-Markers	0.917	0.717	0.976	0.700	0.815	11.308
Pathrecon	0.633	0.683	0.524	0.700	0.792	12.385
PCA-Markers	0.850	0.900	0.940	0.917	0.817	12.769
MCE-corr-Markers	0.933	0.717	0.976	0.683	0.796	13.154
SVMRank-Markers	0.683	0.700	0.940	0.917	0.788	14.923

Table A.4: C scores for the PCA-AREA inferred models developed on a training set (row label) and tested on an independent test set (column label). Results on the rest of the test sets are continued in Tables A.5 and A.6.

	GDS2431	GDS2666	GDS2667	GDS2668	GDS2669
GDS2431	/	0.838	0.580	0.877	0.477
GDS2666	0.822	/	0.915	0.939	0.901
GDS2667	0.511	0.947	/	0.949	0.909
GDS2668	0.519	0.980	0.891	/	0.893
GDS2669	0.444	0.941	0.954	0.941	/
GDS2671	0.778	0.958	0.921	0.954	0.875
GDS2672	0.237	0.941	0.960	0.935	0.909
GDS2688	0.911	0.483	0.343	0.386	0.552
GDS586	0.674	0.945	0.903	0.887	0.879
GDS587	0.385	0.713	0.426	0.733	0.352
F	0.148	0.970	0.853	0.954	0.806
G	0.326	0.889	0.749	0.800	0.798
N	0.126	0.709	0.879	0.733	0.855
Z	0.148	0.798	0.863	0.844	0.881

Table A.5: C scores for the PCA-AREA inferred models developed on a training set (row label) and tested on an independent test set (column label). Results on the rest of the test sets are found in Tables A.4 and A.6.

	GDS2671	GDS2672	GDS2688	GDS586	GDS587
GDS2431	0.721	0.770	0.651	0.841	0.635
GDS2666	0.840	0.818	0.373	0.821	0.757
GDS2667	0.869	0.857	0.427	0.774	0.762
GDS2668	0.770	0.804	0.399	0.738	0.794
GDS2669	0.828	0.830	0.426	0.817	0.810
GDS2671	/	0.711	0.326	0.869	0.937
GDS2672	0.840	/	0.467	0.567	0.624
GDS2688	0.218	0.467	/	0.119	0.619
GDS586	0.715	0.739	0.474	/	0.884
GDS587	0.717	0.251	0.718	0.913	/
F	0.919	0.800	0.268	0.925	0.757
G	0.859	0.824	0.490	0.710	0.344
N	0.848	0.905	0.436	0.861	0.667
Z	0.760	0.861	0.318	0.381	0.910

Table A.6: C scores for the PCA-AREA inferred models developed on a training set (row label) and tested on an independent test set (column label). Results on the rest of the test sets are found in Tables A.4 and A.5.

	F	G	N	Z
GDS2431	0.950	0.700	0.274	1.000
GDS2666	0.967	1.000	0.940	1.000
GDS2667	0.983	1.000	0.988	1.000
GDS2668	0.983	1.000	0.333	1.000
GDS2669	1.000	0.967	0.952	0.950
GDS2671	0.967	1.000	1.000	1.000
GDS2672	0.967	1.000	0.976	1.000
GDS2688	0.183	0.000	0.095	0.333
GDS586	0.783	0.933	0.310	0.883
GDS587	0.750	0.933	0.940	0.900
F	/	1.000	0.976	1.000
G	0.933	/	1.000	0.950
N	0.850	0.883	/	0.817
Z	0.767	0.750	0.988	/

Dodatek B

Razširjeni povzetek v slovenskem jeziku

B.1 Uvod

S problemi rangiranja, oziroma razvrščanja stvari po nekem kriteriju, se srečujemo vsak dan. Rangiramo na primer filme in se odločimo, kateri se nam zdijo vredni nakupa, enkratnega ogleda in kateri nas ne zanimajo. Pogosto dobimo obstoječe rangiranje od nekoga drugega, najsi bodo to prijateljeva priporočila glede filmov ali razvrstitev spletnih strani iskalnika Google. Dobljena priporočila lahko ocenimo, tako da primerjamo, koliko se prejeto rangiranje ujema s takim, ki bi ga sestavili sami. Glede na podobnost rangiranja in s tem uporabnost vira, se zatem lahko odločimo, ali vir še uporabljati v prihodnosti ali ne.

S strojnim učenjem si pomagamo pri reševanju problemov, ko je količina podatkov prevelika, naloge ponavljajoče se, ali kadar je z računalnikom možno doseči boljše rezultate. Rangiranje v tem pogledu ni izjema in je tudi ena izmed družin problemov, pri katerih si lahko pomagamo z računalnikom in ustreznimi algoritmi strojnega učenja. Ob vedno večjem zanimanju za področje rangiranja (v strojnem učenju) narašča število metod, kakor tudi raznolikost njihovih pristopov.

Najbolj pogosto se metode učijo iz atributnega zapisa primerom napovedati numerično vrednost, glede na katero se lahko primere razvrsti. Drugi pristop deluje na parih primerov in gradi modele, ki napovejo, kateri od

dveh primerov naj bo rangiran višje. Tretji pristop, ki bo tudi osnova naših razširitev, uporablja postopek binarnega razcepa. Pri tem se prvotni problem rangiranja najprej razcepi na več manjših podproblemov uvrščanja v dva razreda. Ti podproblemi so bili že zelo dobro raziskani in za njihovo reševanje obstaja mnogo uveljavljenih metod.

Vzporedno se kot podpodročje strojnega učenja razvija hkratno učenje več nalog (ang. multitask learning). Vprašanje, ki ga rešuje, je, ali se da več sorodnih nalog naučiti bolje, če jih rešujemo hkrati, namesto vsako posebej. Podproblemi, ki nastanejo po binarnem razcepu, so si gotovo podobni. Naša predpostavka je, da se jih da skupaj rešiti bolje, kot če bi jih reševali neodvisno, kot je bila dosedanja praksa.

B.2 Sorodno delo

Rangiranje je v različnih oblikah prisotno že dalj časa in se zato pojavlja kot del različnih področij strojnega učenja, iskanja in izbiranja informacij (ang. information retrieval) ter statističnega modeliranja.

Najprej se je večina raziskav rangiranja pojavljala pod okriljem iskanja in izbiranja informacij [1], kjer se je pojavila potreba po rangiranju zadetkov iskanja glede na njihovo relevantnost. V zadnjem času se rangiranje uveljavlja tudi kot samostojno področje nadzorovanega strojnega učenja, pri čemer je več poudarka na splošnejših raziskavah različnih oblik problema in ne toliko na uporabi za določen cilj [15, 12].

Binarni razcep se je uveljavil že pri reševanju uvrščanja v več razredov [21]. Zelo podobne ali celo enake tehnike, kot so bile razvite tam, se lahko uporabijo tudi pri reševanju problemov rangiranja. Poleg tega obstajajo tudi pristopi, razviti posebej za rangiranje, ki uporabljajo urejenost razredov [22, 8, 18].

Hkratno učenje več nalog je bilo že uporabljeno v povezavi z rangiranjem [23, 24, 25], a ne na enak način kot pri nas. Medtem ko so predhodna dela izboljšala učenje, kadar je bilo na voljo več nalog rangiranja, mi rešujemo en sam problem rangiranja, sorodne naloge pa pridobimo z binarnim razcepom.

B.3 Rangiranje

Rangiranje (ang. ranking) je del nadzorovanega strojnega učenja. Za razliko od bolj znanih problemov klasifikacije in regresije pri rangiranju cilj ni napoved vrednosti odvisne (razredne) spremenljivke. Namesto tega se želimo iz učnih primerov naučiti modela, ki bo znal rangirati nove primere. Čeprav bi glede na pomen lahko namesto izraza rangiranje govorili o razvrščanju, bomo uporabljali prvi izraz, saj se je razvrščanje v strojnem učenju že uveljavilo za nenadzorovano učenje razvrščanja v skupine (ang. clustering). Bolj natančno bi zgoraj opisani problem imenovali *rangiranje primerov* (ang. instance ranking) [15], s čimer se ga loči od sorodnega problema *rangiranja oznak* (ang. label ranking) [16, 17]. Nekatere ideje in pristopi so skupni različnim oblikam rangiranja, a v tem doktorskem delu se bomo omejili na probleme rangiranja primerov.

B.3.1 Vrste

Glede na število rangov uporabljenih pri rangiranju množice primerov, ločimo več družin problemov: zvezno rangiranje, večdelno rangiranje in dvodelno rangiranje.

Zvezno rangiranje

Najbolj osnoven problem, kjer za rangiranje n primerov uporabimo range od 1 do n, bomo poimenovali zvezno rangiranje. Pri problemih te vrste je možna primerjava poljubnih dveh primerov, od katerih mora biti eden rangiran višje od drugega.

Večdelno rangiranje

Pogosto se ukvarjamo s problemi, v katerih so množice učnih primerov, ki se jih med sabo ne da primerjati. Take podatke je vseeno možno rangirati, pri čemer primerom iz neprimerljive skupine (ekvivalenčni razred) priredimo enak rang. Posamezni rangi tako niso več zaporedne številke rangiranih primerov, ampak jih obravnavamo kot množice primerov, problem pa imenujemo večdelno rangiranje (ang. multiparite ranking). Pripadnost primera določenemu rangu bomo zapisali kot $x \in R_i$. Rangi so urejeni (R_1 do R_k)

in disjunktni. Zaradi sorodnosti s pojmom razreda pri problemu uvrščanja lahko namesto izraza rang uporabljamo tudi razred rangiranja. Število rangov k in število učnih primerov na rang sta pomembni lastnosti podatkov večdelnega rangiranja, ki vplivata na učinkovitost učenja z različnimi metodami.

Dvodelno rangiranje

Kadar vsi učni primeri spadajo v enega izmed dveh možnih rangov, govorimo o dvodelnem rangiranju (ang. bipartite ranking). Na ta problem lahko gledamo kot na robni primer večdelnega rangiranja ($k = 2$), vendar se je zaradi močne povezave z uvrščanjem v dva razreda dvodelno rangiranje uveljavilo prej in je bolje raziskano. Metode za dvodelno rangiranje pogosto optimizirajo mero AUC (ang. area under the curve), ki meri ploščino pod ROC (ang. receiver operating characteristic) krivuljo [28, 29].

B.3.2 Opis problema

Omejili se bomo na večdelno rangiranje, ki omogoča modeliranje enakosti med učnimi primeri in se ga da pogosto uporabiti za modeliranje aktualnih problemov. Ker se področje šele uveljavlja, se še ni izoblikoval enoten standarden zapis problema. Več predlaganih metod [8, 7, 18, 11, 13] tako predpostavi rahlo drugačne oblike vhoda in izhoda ter cilje učenja.

Vhodne podatke predstavlja rangirana množica učnih primerov. To je možno podati neposredno, z urejenim zaporedjem primerov. Bolj pogosto in praktično jo je podati posredno, na primer v obliki množice parov `(primer, oznaka)`, kjer je oznaka lahko rang (zaporedna številka) primera ali poljubna numerična vrednost, po kateri je primere možno rangirati. Druga uporabna posredna oblika podatkov je sestavljena iz množice parov primerov, s čimer se prednosti lahko poda paroma. Še posebej je ta oblika uporabna, kadar podatki niso popolni in so znane primerjave samo med nekaterimi pari.

Rezultat učenja je model, ki zna rangirati nove množice primerov. Navadno so modeli predstavljeni v obliki funkcije, ki napove vmesne vrednosti, na podlagi katerih je primere možno rangirati. Prevladujeta dve obliki: *koristnostna funkcija* (ang. utility score function) in *prednostna funkcija* (ang. binary preference function). Prva napove za vsak primer numerično vre-

dnost, glede na katero se primere nato uredi. Druga je definirana nad pari primerov in napove, ali naj bo prvi primer rangiran višje od drugega ali ne.

Cilj učenja je zgraditi model, ki bo znal posploševati iz učnih podatkov in dobro rangirati nove množice primerov. A podrobnosti pri tem splošnem cilju lahko problem bistveno spremenijo. Prva razlika je med popolnim in delnim rangiranjem, ki se razlikujeta po zahtevi, da so vsi primeri rangirani. Pri določitvi cilja je pomembna tudi definicija kvalitete rangiranja. V nadaljevanju bodo natančneje opisane mere, ki jih lahko uporabimo za primerjavo rangiranj, tu pa omenimo le splošnejšo razliko med globalnim in lokalnim ocenjevanjem. Pri prvem stremimo k čim manj napakam v celotnem rangiranju, medtem ko pri drugem več poudarka namenimo določenemu delu rangiranja (pogosto najvišje rangiranim primerom).

B.3.3 Pristopi

Metode za učenje rangiranja se lahko glede na pristop reševanja problema zelo razlikujejo. Večino metod se da uvrstiti v eno izmed naslednjih družin: rangiranje s koristnostno funkcijo, parno rangiranje in binarni razcep.

Rangiranje s koristnostno funkcijo

Model za vsak učni primer napove numerično vrednost imenovano koristnost (ang. utility score). Rangiranje nato dobimo z ureditvijo primerov glede na njihove koristnosti. Učni problem se s tem prevede na modeliranje numerične spremenljivke, kar omogoča uporabo več obstoječih metod [31, 32, 33]. Poleg tega se razvijajo nove metode posebej namenjene rangiranju [7, 38, 39, 10, 40, 9, 41].

Parno rangiranje

Parno rangiranje obravnava dva učna primera hkrati in se trudi naučiti prednostne funkcije, ki bo napovedala, kateri od obeh podanih primerov naj bo rangiran pred drugim. Pri tem se pojavi novo vprašanje predstavitve parov, od katerih sta najbolj razširjeni združitev vektorskih zapisov $x_{1,2} = (x_1, x_2)$ in vektorska razlika $x_{1,2} = x_1 - x_2$. Metode so pogosto izpeljane iz dobro znanih standardnih učnih algoritmov (metoda podpornih vektorjev, nevronske mreže, itd.) [7, 40, 9, 43, 42].

Binarni razcep

Pristopi, ki temeljijo na binarnem razcepu, problem najprej razdelijo na več binarnih podproblemov uvrščanja. Za vsakega zgradijo model in napovedi modelov na koncu združijo v rešitev prvotnega problema. Primeri metod za rangiranje, osnovani na tem principu, vključujejo znani pragovni pristop (Frank in Hall [22]) ter novejše izboljšave [8, 18, 45]. Binarni razcep bo predstavljal izhodišče izboljšavam in predlagani novi metodi v naslednjem poglavju ter bo bolj natančno predstavljen tam.

B.3.4 Vrednotenje

Notranje preverjanje

Podobno kot pri uvrščanju in regresiji se za namene vrednotenja podatke lahko razdeli in kvaliteto modelov oceni na neodvisnih pridržanih podatkih, ki niso bili uporabljeni pri učenju. Večina standardnih postopkov, kot so delitev na učno in testno množico, prečno preverjanje in ostali [46], je primerna tudi za rangiranje.

Pomembna razlika je pri postopku "izpusti enega", ki se ga uporablja, kadar je na voljo majhna množica primerov. Ker se kvalitete napovedi pri rangiranju ne da oceniti ob preverjanju s samo enim učnim primerom, ta postopek tu ni uporaben. Namesto njega je možno uporabiti postopek "izpusti dva", ki pridrži dva primera hkrati in po učenju na ostalih preveri, ali model pravilno rangira pridržana primera.

Mere za ocenjevanje

Najbolj pogosta in znana mera za ocenjevanje rangiranja je AUC, ki pa je v osnovni obliki definirana samo za dvodelno rangiranje. Vrednost AUC tam predstavlja oceno verjetnosti, da bosta primera iz različnih razredov pravilno rangirana. Za večdelno rangiranje je možno uporabiti posplošitve AUC, kot je C-indeks. Ta ocenjuje verjetnost, da bosta dva primera iz poljubnih (različnih) rangov pravilno rangirana.

Za ocenjevanje rangiranja so primerni tudi koeficienti korelacije rangov (ang. rank correlation). Najbolj pogosta sta Spearmanov ρ in Kendallov τ koeficient.

B.3.5 Analiza

Z eksperimenti na sintetičnih podatkih smo raziskali lastnosti rangiranja s koristnostno funkcijo in ta pristop primerjali s parnim rangiranjem. Rezultati so pokazali, da je izbor vrednosti, ki jih koristnostna funkcija modelira, ključnega pomena. Še posebej to velja za splošne regresijske metode, ki stremijo k majhnim odstopanjem od podanih vrednosti. Specializirane metode za rangiranje so pri tem v prednosti, ker dopuščajo večja odstopanja, če to vodi do boljšega modela za rangiranje. V ločenem eksperimentu se je pokazalo, da je parno rangiranje dobra alternativa, kadar je podatkovni model težje opisati s koristnostno funkcijo.

B.3.6 Uporaba: razvoj zarodnih celic

Na področju biomedicine je veliko zanimanja za proces razvoja zarodnih celic oziroma njihove diferenciacije v bolj specializirane vrste celic, a vendar veliko podrobnosti ostaja neznanih. V naši raziskavi, ki je bila pred tem objavljena že v reviji *Bioinformatics* [49] in predstavljena na konferenci [50], se problema lotimo s pomočjo strojnega učenja. Različne stopnje diferenciacije obravnavamo kot razrede rangiranja in preverimo, ali se da naučiti modele, ki na podlagi izraženosti genov celicam določijo stopnjo in jih rangirajo.

Zaradi velikega števila značilk (izraženosti 5000–25000 genov), njihovo število najprej omejimo (1000). Nato uporabimo različne nenadzorovane (PCA, Pathrecon, MCE) in nadzorovane metode (PLS, SVM^{rank}) za gradnjo napovednih modelov. Za njihovo primerjavo smo uporabili prosto dostopne podatkovne nabore iz baze podatkov GEO. Ovrednotenje je bilo opravljeno z notranjim preverjanjem z metodo "izpusti dva" ter zunanjim preverjanjem, kjer smo model zgradili na enem podatkovnem naboru in ga ocenili na drugem. Kot mero za ocenjevanje rangiranja smo uporabili C-indeks.

Rezultati so bili zelo dobri in so pogosto presegli C vrednosti 0.9, tako pri notranjem kot tudi zunanjem preverjanju. Posebej pri slednjem je to velik uspeh, saj nakazuje, da so modeli sposobni zajeti širše uporabno znanje, ki se ga da uporabiti na podatkovnih naborih sorodnih eksperimentov. Med metodami se ni pokazalo, da bi bila katera statistično značilno boljša od ostalih.

B.4 Povezava razcepa in hkratnega učenja

B.4.1 Binarni razcep

Binarni razcep je tehnika, ki temelji na principu deli in vladaj. V strojnem učenju se je uveljavila že pred uporabo pri rangiranju za reševanje problemov uvrščanja v več razredov (ang. multiclass classification) [44, 72, 21]. V skoraj enaki obliki se lahko binarni razcep uporabi tudi za večdelno rangiranje [22, 8, 27, 73, 18, 74], kjer je struktura problema podobna, le da so rangi, ki ustrezajo razredom, tu še urejeni. Uporaba lastnosti urejenosti omogoča tudi razvoj različic, namenjenih izključno rangiranju.

Glavna ideja tehnike je v razcepu originalnega problema na več podproblemov uvrščanja v dva razreda. Le-ti so bili dobro raziskani, zato obstaja veliko uveljavljenih metod za njihovo reševanje. Po gradnji modelov za posamezne podprobleme je potrebno njihove rezultate ponovno združiti v rešitev prvotnega problema (agregacija). Nekaj najbolj uveljavljenih verzij binarnega razcepa vključuje naslednje različice:

Eden proti vsem (ang. one-against-all, OAA) razcepi problem s k rangi v k binarnih odločitvenih problemov, v katerih se vsak posamezen rang primerja z vsemi ostalimi.

Eden proti enemu (ang. one-against-one, OAO) razcepi problem s k rangi v $\frac{k(k-1)}{2}$ binarnih odločitvenih primerov, v vsakem izmed katerih se primerja po dva ranga.

Pragovni razcep [22] (ang. thresholding) je eden izmed prvih postopkov razcepa, predlagan posebej za rangiranje. To se odraža v predpostavki, da so razredi urejeni in jih je zato možno z določitvijo praga razdeliti v dve skupini — tiste pod in nad pragom.

Posplošitve zgornjih verzij binarnega razcepa in s tem še cele družine sorodnih različic se da enotno zapisati v obliki matrik, ki zakodirajo, kateri rangi tvorijo pozitivni in kateri negativni razred v posameznih binarnih podproblemih [80]. To omogoča tudi povezavo s področjem kod za odpravljanje napak, ki lahko služijo kot vodilo za dobre izbire sestave matrik [79]. Druga vrsta posplošitev uporablja drevesa ali usmerjene

aciklične grafe za predstavitev zaporedja binarnih podproblemov, ki določijo končno rešitev [81, 82, 18].

B.4.2 Hkratno učenje več nalog

Pogosto imamo na voljo podatke za več sorodnih nalog strojnega učenja, ki jih lahko uporabimo za medsebojno pomoč pri učenju. Na taki predpostavki je osnovano področje hkratnega učenja več nalog (ang. multitask learning). Začetek razvoja v samostojno področje bi lahko postavili okoli objave istoimenskega članka Richarda Caruane [24], ki je predlagal modeliranje več nalog s skupno umetno nevronsko mrežo. Od takrat je bila ideja hkratnega učenja prenesena na mnogo metod in izvedena na različne načine, ki jih glede na vrsto predpostavke o sorodnosti nalog lahko razdelimo v dve družini.

Prva družina vključuje metode, ki predpostavljajo, da bodo modeli sorodnih nalog imeli *podobne parametre* in zaradi tega dajali tudi podobne napovedi. Taka predpostavka je pogosto izpolnjena, kadar naloge temeljijo na podatkih podobnih virov. Na primer, vsaka naloga vključuje gradnjo modela za enega izmed množice podobnih uporabnikov.

Druga družina metod deluje na bolj mili predpostavki in zahteva le *podobno strukturo* podatkov in modelov. Metode te vrste bi delovale tudi pri podatkih uporabnikov, ki uporabljajo enako podmnožico značilk, a so njihovi okusi popolnoma nasprotni in se morajo zato razlikovati tudi napovedni modeli.

Predstavnik druge družine je tudi metoda hkratnega učenja značilk (ang. multitask feature learning) MTL-FEAT [87], ki jo bomo uporabili v nadaljevanju. Glavna predpostavka metode je, da je možno modele posameznih nalog izraziti z manjšim naborom novih značilk. Postopek učenja zato sočasno optimizira tako sestavo novih značilk, kot tudi uteži, s katerimi so uporabljene v modelih. Algoritem 3 prikazuje iterativni postopek izmenične optimizacije uteži (W) in transformacije prostora (D), s katerim sočasno gradimo modele za vse naloge $t \in \{1, \ldots, T\}$. T označuje število vseh nalog, n število učnih primerov za posamezno nalogo in d število vhodnih spremenljivk. Uporabljeni so še parametri γ (koeficient regularizacije), ε (pozitivna spremenljivka, ki gre proti 0 in zagotavlja konvergenco) in *tol* (meja, ki določa kdaj prekinemo iterativni postopek).

Algoritem 3: Psevdo-koda algoritma MTL-FEAT za hkratno učenje značilk.

Input: učni podatki $\{(x_{ti}, y_{ti})\}_{i=1}^{n}, \quad 1 \leq t \leq T$
Parameters: γ, ε, tol
Output: $d \times d$ matrika D,
$d \times T$ matrika uteži $W = [w_1, \ldots, w_T]$

$D = \frac{I}{d}$
while $\|W - W_{prev}\| > tol$ **do**
  for $t = 1, \ldots, T$ **do**
    `// poišči uteži za nalogo` t
    $w_t = \text{argmin} \{\sum_{i=1}^{n} L(y_{ti}, \langle w, x_{ti} \rangle) + \gamma \langle w, D^{-1} w \rangle : w \in \mathbb{R}^d\}$
  end
  `// optimiziraj` D `pri trenutnih utežeh` W
  $D = \frac{(WW^T + \varepsilon I)^{\frac{1}{2}}}{\text{trace}(WW^T + \varepsilon I)^{\frac{1}{2}}}$
end

B.4.3 Povezava binarnega razcepa in hkratnega učenja

Naša predpostavka je, da je učenje nalog, ki nastanejo po binarnem razcepu, mogoče izboljšati s hkratnim učenjem. Pri zasnovi nove metode, ki temelji na povezavi obeh področij, smo želeli izbrati metodi za binarni razcep in hkratno učenje, ki se bosta kar najbolj dopolnjevali.

Vse osnovne različice binarnega razcepa lahko privedejo do posameznih nalog, katerih modeli ne bodo nujno podobni. Zato smo se odločili za metodo hkratnega učenja MTL-FEAT, ki predpostavi, da modeli uporabljajo enak nabor naučenih značilk, ne pa tudi kako jih utežijo. Predpostavka dobro ustreza nalogam razcepa, ki so nastale iz skupnih podatkov in enake domene. Povezave med spremenljivkami se zato prenesejo v vse naloge in jih modeli lažje odkrijejo in uporabijo, če imajo dostop do podatkov vseh nalog.

Pri izboru metode binarnega razcepa smo se odločali med tehnikami eden-proti-ostalim, eden-proti-enemu ter pragovnim razcepom, ki se kljub preprostosti v praksi dobro obnesejo [73, 8]. Izbrali smo tehniko eden-proti-enemu, ki obeta najboljše rezultate ob združitvi s hkratnim učenjem, saj ustvari večje število nalog z manj učnimi primeri v primerjavi z drugima

tehnikama. Kot smo eksperimentalno potrdili, take okoliščine omogočajo večje izboljšave hkratnega učenja. Napovedi posameznih modelov, ki se odločajo med dvema rangoma, združimo z glasovanjem, tako da učnemu primeru priredimo največkrat napovedani rang. Algoritem 4 opisuje predlagano metodo za rangiranje, ki smo jo poimenovali BDMT.

Algoritem 4: Psevdo-koda algoritma za rangiranje BDMT s funkcijama za učenje (fit) in napovedovanje (predict).

```
Function fit(data)
    tasks = [ ]
    ranks = [ ]
    for i = 1, ..., k do
        for j = 1, ..., i − 1 do
            data_{i,j} = {(x, y) | (x, y) ∈ data and y ∈ {i, j}}
            tasks.append(data_{i,j})
            ranks.append((i, j))
    D, W = MTL-FEAT(tasks)
Function predict(x)
    predictions = [ ]
    for t = 1, ..., T do
        w_t = W[:, t]
        i, j = ranks[t]
        prediction = (⟨w_t, x⟩ > 0) ? i : j
        predictions.append(prediction)
    return most_common(predictions)
```

B.4.4 Analiza in ovrednotenje

Lastnosti množice nalog binarnega razcepa

Osnovni lastnosti množice nalog, ki nastanejo z različnimi metodami binarnega razcepa, sta število nalog in število razredov rangiranja, prisotnih v posamezni nalogi. Slednje neposredno določa tudi količino učnih primerov

za vsako nalogo. Ker ti dve lastnosti močno vplivata na sposobnosti metod hkratnega učenja več nalog, smo preverili njun vpliv na hkratno učenje značilk, uporabljeno v BDMT. Potrdili sta se predpostavki, da so večje izboljšave rezultatov mogoče ob večjem številu nalog ter ob manjšem številu učnih primerov za posamezno nalogo.

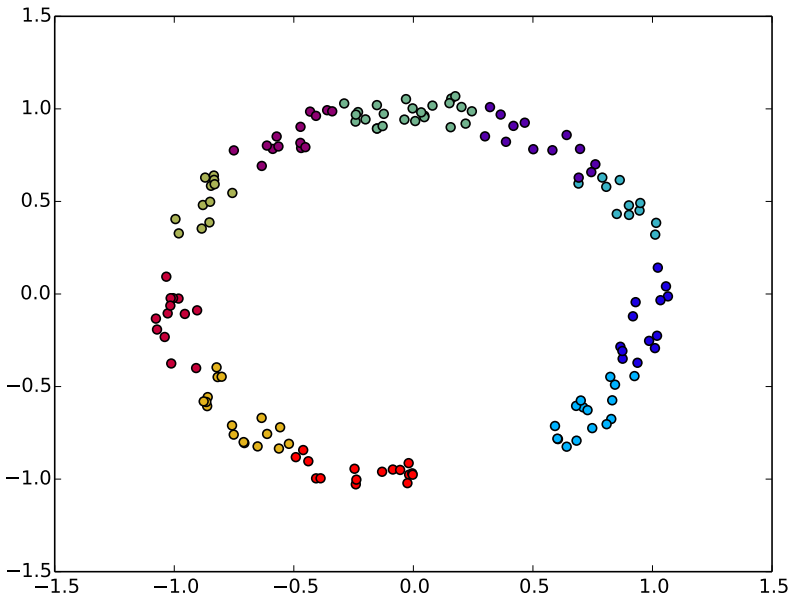
Razcep nelinearnih problemov

Linearne metode učenja so zelo popularne, predvsem zaradi hitrosti ter preprostega in razumljivega modela. A ker so vzorci v podatkih pogosto nelinearni, moramo v teh primerih poseči po bolj zapletenih metodah. Ena izmed prednosti razcepa eden-proti-enemu je tudi, da je z razbitjem problema možno odpraviti nekatere vrste nelinearnosti, kar omogoča uporabo linearnih metod za učenje posameznih nalog. Primer podatkov, kjer tak princip dobro deluje, predstavljajo rangi v obliki gruč primerov, ki se jih da paroma dobro ločiti, čeprav je opis z enotnim modelom bolj zapleten.

Analiza BDMT

Da bomo lahko analizirali različne lastnosti in ovrednotili BDMT, predstavimo model za generiranje podatkov večdelnega rangiranja. Ker ni smiselno pričakovati, da bodo v praksi rangi v prostoru razporejeni naključno, predlagamo vzorec, kjer so si primeri iz zaporednih rangov med sabo bolj podobni. To dosežemo z razporeditvijo rangov v obliki krožnice (slika B.1), ki se nahaja v ravnini, vpeti v celoten m-dimenzionalen učni prostor. Osnovna verzija podatkovnega nabora `krog` vsebuje $m = 20$ spremenljivk, nekateri eksperimenti pa vključujejo tudi dodatne spremenljivke, ki ne vplivajo na določitev rangiranja.

BDMT smo primerjali z osnovno različico metode eden-proti-enemu (OAO) ter z dvema globalnima metodama: grebensko regresijo (ang. ridge regression) in SVM^{rank}. Rezultati, podani v Tabeli B.1, potrjujejo izboljšavo, ki jo z dodatkom hkratnega učenja nalog doseže BDMT v primerjavi z OAO. BDMT uspe preseči tudi globalni metodi, celo na naborih z manj učnimi primeri ($n = 100$), kjer je imela osnovna metoda OAO manj uspeha. Dodatne irelevantne spremenljivke (nabora z $m = 30$) imajo podoben učinek kot zmanjšanje števila učnih primerov, saj naredijo problem težji in bi za enak



Slika B.1: Projekcija primerov, opisanih z 20 spremenljivkami, v 2D podprostor, kjer je definirano rangiranje. Rangi so predstavljeni z različnimi barvami in zaporedoma naraščajo v smeri urinega kazalca.

Tabela B.1: Povprečne vrednosti C-indeksa s standardnimi odkloni za štiri metode rangiranja, ovrednotene na podatkovnih naborih `krog` različnih velikosti.

	$n = 100$		$n = 250$	
method	$m = 20$	$m = 30$	$m = 20$	$m = 30$
ridge	0.75 (0.03)	0.70 (0.04)	0.79 (0.03)	0.75 (0.02)
SVM^{rank}	0.81 (0.02)	0.79 (0.02)	0.81 (0.03)	0.81 (0.02)
OAO	0.79 (0.04)	0.71 (0.04)	0.94 (0.02)	0.86 (0.03)
BDMT	0.91 (0.01)	0.83 (0.02)	0.97 (0.01)	0.94 (0.01)

uspeh potrebovali več podatkov. Vpliv lastnosti podatkovnega nabora na dosežene rezultate smo bolj natančno preverili s pomočjo krivulj učenja. Za najbolj izrazito se je pokazala odvisnost od velikosti učne množice (slika B.2).

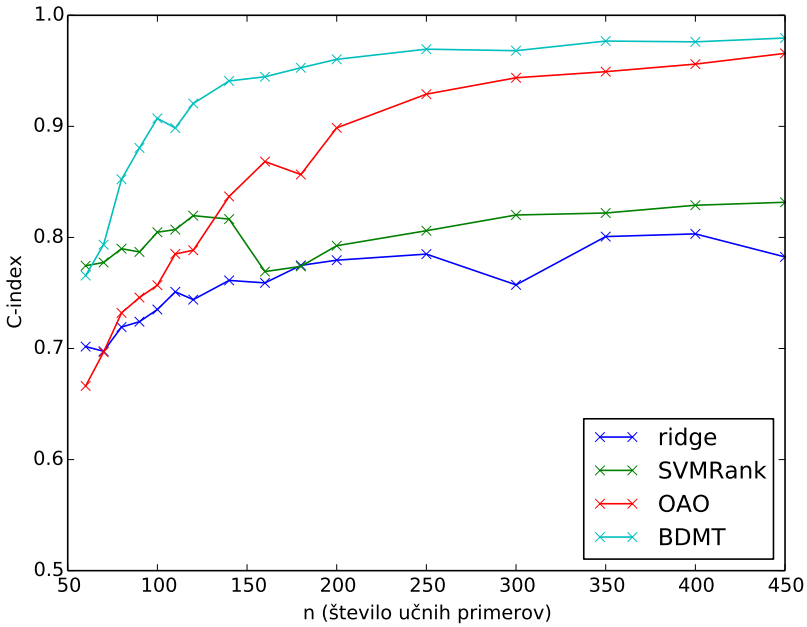
Preizkusili smo dve alternativni metodi agregacije, ki sta dosegli slabše rezultate kot navadno večinsko glasovanje. Obe metodi namesto binarnih odločitev uporabljata oddaljenosti od odločitvene meje. Te so navadno povezane z zanesljivostjo odločitve, a v primeru nalog razcepa eden-proti-enemu lahko dajejo zavajajoče rezultate.

OAO in BDMT smo preizkusili tudi na podatkih o razvoju zarodnih celic, uporabljenih v analizi iz prejšnjega poglavja. Glavno težavo je pri tem predstavljalo majhno število učnih primerov za posamezen rang (2–4). Za metode, ki temeljijo na binarnem razcepu, to predstavlja večjo oviro. Rezultati so potrdili, da se v tem primeru OAO in BDMT ne moreta primerjati z globalnimi metodami. V primerjavi z OAO je BDMT na desetih podatkovnih naborih sedemkrat dosegel boljši rezultat in dvakrat slabši.

B.4.5 Dodatne prednosti predlaganega pristopa

Stabilnost

Po razcepu OAO nastane množica problemov uvrščanja, za katere je uporaben samo majhen del podatkov. Ocenjevanje parametrov modelov je zato težje in veliko bolj občutljivo na spremembe učne množice. Predlagana metoda BDMT dosega boljšo stabilnost pri učenju, saj zaradi uporabe hkra-



Slika B.2: Krivulje učenja prikazujejo učinkovitost metod pri različnih velikostih učne množice.

tnega učenja na ocenjene parametre ne vpliva samo (majhna) množica podatkov enega problema, ampak tudi podatki drugih problemov in znanje, pridobljeno iz njih. To se je potrdilo tudi eksperimentalno, saj smo pri več ponovitvah učenja na naključno generiranih podatkih opažali veliko manjše nihanje rezultatov metode BDMT v primerjavi z OAO. V Tabeli B.1 vidimo, da so standardni odkloni rezultatov najmanjši prav pri BDMT, celo v primerjavi z globalnimi metodami, ki uporabljajo vse podatke.

Časovna zahtevnost

Metode, ki delujejo na osnovi binarnega razcepa, lahko za učenje množice preprostih problemov porabijo manj časa, kot bi ga bolj zapletena metoda za enovito optimizacijo globalnega modela [44]. Med opravljenimi eksperimenti je BDMT dosegal nižje čase učenja v primerjavi z metodo SVM^{rank}, ki se uči enega modela. Zaradi razlik v implementacijah namesto točnih meritev podamo samo okvirne čase, ki se lahko razlikujejo tudi za velikostni razred in več. Na primer, za učenje na naboru `krog` ($n = 250$, $m = 30$) so metode

potrebovale približno 0.1s (OAO), 1s (BDMT) ter do 100s in več za SVM^{rank}.

V primerjavi z OAO je učenje BDMT počasnejše za faktor, enak številu iteracij v postopku hkratnega učenja. To je bilo v naših eksperimentih večinoma med 10–50. Obe metodi zaradi razcepa omogočata tudi vzporedno grajenje modelov, ki se lahko izvaja na ločenih enotah porazdeljenega sistema.

Tolmačenje modelov

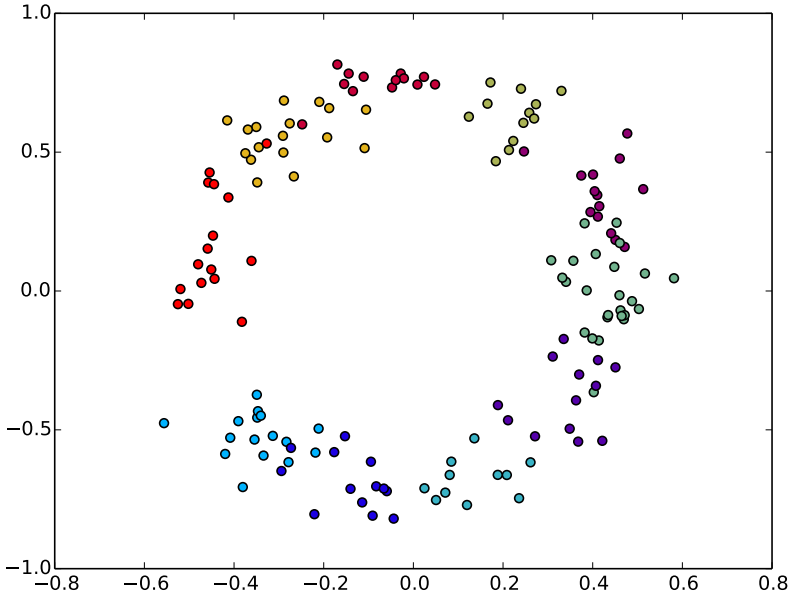
Zaradi sestave iz komponent, ki posamično omogočajo dobro tolmačenje modelov, je to tudi ena izmed prednosti metode BDMT. Preko uporabljenega hkratnega učenja značilk je možno odkriti skrite faktorje, ki vplivajo na rangiranje. Podatki s slike B.1 so na sliki B.3 prikazani še v podprostoru dveh najpomembnejših značilk, naučenih z BDMT. Čeprav prvega podprostora, kjer je bilo rangiranje definirano, metoda ne pozna, se ji je iz podatkov uspelo naučiti zelo dober približek.

Opazovanje in primerjava modelov posameznih problemov binarnega razcepa nam pomagata pri odkrivanju odnosov med razredi rangiranja in prepoznavanju njihove razporeditve po prostoru. Iz uteži naučenih značilk lahko na primer ugotovimo, da se primeri iz drugega razreda razlikujejo od tistih iz prvega po večji vrednosti obeh faktorjev, medtem ko kasneje za prehod iz sedmega v osmi razred velja ravno obratno (faktorja se zmanjšata).

B.5 Zaključek

Rangiranje se trenutno uveljavlja kot samostojno področje strojnega učenja. K temu smo prispevali tudi v tem doktorskem delu s predstavitvijo in analizo metod z več področij, ki se lahko uporabijo za večdelno rangiranje učnih primerov. Njihova primerjava je bila izvedena z metodologijo, prirejeno posebej za problem večdelnega rangiranja. Nova, perspektivna uporaba rangiranja na problemu iz bioinformatike je dokazala uporabnost opisanih metod.

V drugem delu disertacije smo se posvetili družini metod, ki z binarnim razcepom prevedejo problem rangiranja na množico problemov uvrščanja v dva razreda. Da bi izboljšali učenje nastale množice problemov, smo predla-



Slika B.3: Projekcija v podprostor, ki ga določata prvi dve značilki naučeni z BDMT. Podobnost sliki B.1 kaže na dobro ujemanje naučenih značilk in "neznanih" faktorjev, ki sta bila del podatkovnega modela.

gali povezavo s hkratnim učenjem več nalog. Na podlagi pregleda lastnosti različnih tehnik binarnega razcepa in hkratnega učenja smo predlagali novo metodo za rangiranje (BDMT), ki povezuje razcep eden-proti-enemu in hkratno učenje značilk.

Eksperimentalno smo pokazali, da hkratno učenje bistveno izboljša točnost modelov in omili težave zaradi pomanjkanja učnih primerov. BDMT se je izkazal za konkurenčno metodo, ki se pogosto obnese bolje od uveljavljenih pristopov za rangiranje. Izpostavili bi radi tudi njegovo sposobnost modeliranja zapletenih, nelinearnih vzorcev, kljub preprosti sestavi metode. Zaključili smo z opisom ostalih prednosti, kot so stabilnost in hitrost učenja ter dobro tolmačenje naučenih modelov.

Radi bi poudarili, da pomembno novost predstavlja tudi sama ideja povezave binarnega razcepa in hkratnega učenja, ki poleg našega predloga konkretne metode lahko porodi še celo družino novih metod. S tem ostaja odprtih še veliko možnosti za nadaljnje delo na tem področju.

B.5.1 Prispevki k znanosti

Povzetek glavnih prispevkov k znanosti:

- Primerjalna analiza metod za rangiranje, ki je bila predstavljena v tretjem poglavju in vključuje analizo uporabe rangiranja na problemu iz bioinformatike (glej [49, 50]).
- Nova metoda za rangiranje, ki temelji na binarnem razcepu in hkratnem učenju, je bila predlagana in analizirana v četrtem poglavju.

Bibliography

[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 9780521865715.

[2] Nicholas J. Belkin and W. Bruce Croft. Annual Review of Information Science and Technology, Vol. 22. chapter Retrieval Techniques, pages 109–145. Elsevier Science Inc., New York, NY, USA, 1987. ISBN 0-444-70302-0.

[3] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[4] P. Bollmann and S. K. M. Wong. Adaptive linear information retrieval models. In *Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '87, pages 157–163, New York, NY, USA, 1987. ACM. ISBN 0-89791-232-2.

[5] Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. Learning Preference Relations for Information Retrieval. In *ICML-98 Workshop: Text Categorization and Machine Learning*, pages 80–84, 1998.

[6] Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 41–48, New York, NY, USA, 2000. ACM. ISBN 1-58113-226-3.

[7] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X.

[8] Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy. Binary decomposition methods for multipartite ranking. In Wray Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 5781 of *Lecture Notes in Computer Science*, pages 359–374. Springer Berlin / Heidelberg, 2009.

[9] Hwanjo Yu, Youngdae Kim, and Seungwon Hwang. RV-SVM: An efficient method for learning ranking SVM. *Advances in Knowledge Discovery and Data Mining*, 5476:426–438, 2009.

[10] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jouni Järvinen, and Jorma Boberg. An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165, 2009.

[11] Stéphan Clémençon, Sylvain Robbiano, and Nicolas Vayatis. Ranking data with ordinal labels: optimality and pairwise aggregation. *Machine Learning*, 91(1):67–104, 2013.

[12] Eyke Hüllermeier and Johannes Fürnkranz. Editorial: Preference learning and ranking. *Machine Learning*, 93(2–3):185–189, 2013. ISSN 0885-6125.

[13] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14266-6.

[14] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10(1):243–270, 1999.

[15] Johannes Fürnkranz and Eyke Hüllermeier, editors. *Preference Learning*. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14124-9.

[16] Shankar Vembu and Thomas Gärtner. Label ranking algorithms: A survey. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 45–64. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14124-9.

[17] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172 (16-17):1897–1916, 2008.

[18] José Ramón Quevedo, Elena Montañés, Oscar Luaces, and Juan José del Coz. Adapting decision DAGs for multipartite ranking. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 115–130. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15938-1.

[19] Frank E. Harrell, Jr. *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer Series in Statistics. Springer New York, 2001. ISBN 978-0-387-95232-1.

[20] Willem Waegeman and Bernard De Baets. A Survey on ROC-based Ordinal Regression. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 127–154. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14124-9.

[21] Ana Carolina Lorena, André C.P.L.F. Carvalho, and João M.P. Gama. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30:19–37, 2008.

[22] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In Luc Raedt and Peter Flach, editors, *Machine Learning: ECML 2001*, volume 2167 of *Lecture Notes in Computer Science*, pages 145–156. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42536-6.

[23] Rich Caruana, Shumeet Baluja, and Tom Mitchell. Using the Future to "Sort Out" the Present: Rankprop and Multitask Learning for Medical Risk Evaluation. In *Advances in Neural Information Processing Systems 8*, pages 959–965, 1996.

[24] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[25] Adriana Birlutiu, Perry Groot, and Tom Heskes. Efficiently learning the preferences of people. *Machine Learning*, 90(1):1–28, 2013.

[26] Şeyda Ertekin and Cynthia Rudin. On Equivalence Relationships Between Classification and Ranking Algorithms. *Journal of Machine Learning Research*, 12:2905–2929, 2011.

[27] Stéphan Clémençon, Marine Depecker, and Nicolas Vayatis. Adaptive partitioning schemes for bipartite ranking. *Machine Learning*, 83(1):31–69, 2011.

[28] Tapio Pahikkala, Antti Airola, Hanna Suominen, Jorma Boberg, and Tapio Salakoski. Efficient AUC Maximization with Regularized Least-Squares. In *Proceedings of the 2008 conference on Tenth Scandinavian Conference on Artificial Intelligence: SCAI 2008*, pages 12–19, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-867-0.

[29] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-Pass AUC Optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 906–914, Atlanta, Georgia, USA, 2013.

[30] Weiwei Cheng, Michaël Rademaker, Bernard Baets, and Eyke Hüllermeier. Predicting partial orders: Ranking with abstention. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6321 of *Lecture Notes in Computer Science*, pages 215–230. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15879-7.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, 2009. ISBN 978-0-387-84857-0.

[32] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 2000. ISBN 0-521-78019-5.

[33] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. ISBN 978-0412048418.

[34] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1): 29–36, 1982.

[35] Yehuda Koren and Liran Carmel. Robust linear dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):459–470, 2004.

[36] Paul M. Magwene, Paul Lizardi, and Junhyong Kim. Reconstructing the temporal ordering of biological samples using microarray data. *Bioinformatics*, 19(7):842–850, 2003.

[37] Carlo Vittorio Cannistraci, Timothy Ravasi, Franco Maria Montevecchi, Trey Ideker, and Massimo Alessio. Nonlinear dimension reduction and clustering by Minimum Curvilinearity unfold neuropathic pain and tissue embryological classes. *Bioinformatics*, 26(18):i531–i539, 2010.

[38] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5.

[39] Jun Xu and Hang Li. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7.

[40] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 97–102, 1999.

[41] Vikas C. Raykar, Ramani Duraiswami, and Balaji Krishnapuram. A fast algorithm for learning a ranking function from large-scale data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7): 1158–1170, 2008.

[42] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[43] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5.

[44] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2): 415–425, mar 2002.

[45] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In Johannes Fürnkranz and Eyke Hüllermeier,

editors, *Preference Learning*, pages 65–82. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-14124-9.

[46] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 3 edition, 2005.

[47] Tapio Pahikkala, Antti Airola, Jorma Boberg, and Tapio Salakoski. Exact and efficient leave-pair-out cross-validation for ranking RLS. In Timo Honkela, Matti Pöllä, Mari-Sanna Paukkeri, and Olli Simula, editors, *Proceedings of the 2nd International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'08)*, pages 1–8, Helsinki University of Technology, 2008.

[48] Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. An experimental comparison of cross-validation techniques for estimating the area under the ROC curve. *Computational Statistics & Data Analysis*, 55(4):1828–1844, 2011.

[49] Lan Zagar, Francesca Mulas, Silvia Garagna, Maurizio Zuccotti, Riccardo Bellazzi, and Blaz Zupan. Stage prediction of embryonic stem cell differentiation from genome-wide expression data. *Bioinformatics*, 27(18): 2546–2553, 2011.

[50] Lan Zagar, Francesca Mulas, Riccardo Bellazzi, and Blaz Zupan. Ranking and 1-dimensional projection of cell development transcription profiles. In Mor Peleg, Nada Lavrač, and Carlo Combi, editors, *Artificial Intelligence in Medicine*, volume 6747 of *Lecture Notes in Computer Science*, pages 85–89. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22217-7.

[51] Bhaskar Bhattacharya, Sachin Puri, and Raj K. Puri. A review of gene expression profiling of human embryonic stem cell lines and their differentiated progeny. *Current stem cell research & therapy*, 4(2):98–106, 2009.

[52] F.J. Müller, L.C. Laurent, D. Kostka, I. Ulitsky, R Williams, C. Lu, I.H. Park, M.S. Rao, R. Shamir, P.H. Schwartz, N.O. Schmidt, and J.F. Loring. Regulatory networks define phenotypic classes of human stem cell lines. *Nature*, 455(7211):401–405, 2008.

[53] N. Novershtern, A. Subramanian, L.N. Lawton, Mak R.H., W.N. Haining, M.E. McConkey, N. Habib, N. Yosef, C.Y. Chang, T. Shay, G.M. Frampton,

A.C. Drake, I. Leskov, B. Nilsson, F. Preffer, D. Dombkowski, J.W. Evans, T. Liefeld, J.S. Smutko, J. Chen, N. Friedman, R.A. Young, T.R. Golub, A. Regev, and B.L. Ebert. Densely interconnected transcriptional circuits control cell states in human hematopoiesis. *Cell*, 144(2):296–309, 2011.

[54] Juan Mata, Rachel Lyne, Gavin Burns, and Jürg Bähler. The transcriptional program of meiosis and sporulation in fission yeast. *Nature Genetics*, 32(1):143–147, 2002.

[55] Roger A. Wagner, Raymond Tabibiazar, Arnold Liao, and Thomas Quertermous. Genome-wide expression dynamics during mouse embryonic development reveal similarities to drosophila development. *Developmental Biology*, 288(2):595–611, 2005.

[56] Nancy Van Driessche, Chad Shaw, Mariko Katoh, Takahiro Morio, Richard Sucgang, Miroslava Ibarra, Hidekazu Kuwayama, Tamao Saito, Hideko Urushihara, Mineko Maeda, Ikuo Takeuchi, Hiroshi Ochiai, William Eaton, Jeffrey Tollett, John Halter, Adam Kuspa, Yoshimasa Tanaka, and Gad Shaulsky. A transcriptional profile of multicellular development in dictyostelium discoideum. *Development*, 129(7):1543–1552, 2002.

[57] T. Neri, V. Merico, F. Fiordaliso, M. Salio, P. Rebuzzini, L. Sacchi, R. Bellazzi, C.A. Redi, M. Zuccotti, and S. Garagna. The differentiation of cardiomyocytes from mouse embryonic stem cells is altered by dioxin. *Toxicology Letters*, 202(3):226–236, 2011.

[58] Xiangqin Cui and Gary A. Churchill. Statistical tests for differential expression in cDNA microarray experiments. *Genome Biology*, 4(4):210, 2003.

[59] Taesung Park, Sung-Gon Yi, Seungmook Lee, Seung Yeoun Lee, Dong-Hyun Yoo, Jun-Ik Ahn, and Yong-Sung Lee. Statistical tests for identifying differentially expressed genes in time-course microarray experiments. *Bioinformatics*, 19(6):694–703, 2003.

[60] María José Nueda, Ana Conesa, Johan A. Westerhuis, Huub C. J. Hoefsloot, Age K. Smilde, Manuel Talón, and Alberto Ferrer. Discovering gene expression patterns in time course microarray experiments by ANOVA-SCA. *Bioinformatics*, 23(14):1792–1800, 2007.

[61] Barbara Di Camillo, Gianna Toffolo, Sreekumaran K. Nair, Laura J. Greenlund, and Claudio Cobelli. Significance analysis of microarray transcript levels in time series experiments. *BMC Bioinformatics*, 8(Suppl 1):S10, 2007.

[62] Massimiliano Gentile, Leena Latonen, and Marikki Laiho. Cell cycle arrest and apoptosis provoked by UV radiation-induced DNA damage are transcriptionally highly divergent responses. *Nucleic Acids Research*, 31(16): 4779–4790, 2003.

[63] Melissa J. Peart, Gordon K. Smyth, Ryan K. van Laar, David D. Bowtell, Victoria M. Richon, Paul A. Marks, Andrew J. Holloway, and Ricky W. Johnstone. Identification and functional significance of genes regulated by structurally different histone deacetylase inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 102(10): 3697–3702, 2005.

[64] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[65] Agnar Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2 (3):211–228, 1988.

[66] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In Craig Saunders, Marko Grobelnik, Steve Gunn, and John Shawe-Taylor, editors, *Subspace, Latent Structure and Feature Selection*, volume 3940 of *Lecture Notes in Computer Science*, pages 34–51. Springer Berlin / Heidelberg, 2006.

[67] Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. A comparison of AUC estimators in small-sample studies. In Sašo Džeroski, Pierre Geurts, and Juho Rousu, editors, *Machine Learning in Systems Biology*, volume 8, pages 3–13, 2010.

[68] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[69] Kagnew Hailesellasse Sene, Christopher J. Porter, Gareth Palidwor, Carolina Perez-Iratxeta, Enrique M. Muro, Pearl A. Campbell, Michael A.

Rudnicki, and Miguel A. Andrade-Navarro. Gene function in early mouse embryonic stem cell differentiation. *BMC Genomics*, 8:85, 2007.

[70] Kazuhiro Aiba, Timur Nedorezov, Yulan Piao, Akira Nishiyama, Ryo Matoba, Lioudmila V. Sharova, Alexei A. Sharov, Shinya Yamanaka, Hitoshi Niwa, and Minoru S. H. Ko. Defining developmental potency and cell lineage trajectories by expression profiling of differentiating mouse embryonic stem cells. *DNA Research*, 16(1):73–80, 2009.

[71] Sang-Hyeun Park and Johannes Fürnkranz. Efficient prediction algorithms for binary decomposition techniques. *Data Mining and Knowledge Discovery*, 24(1):40–77, 2012.

[72] Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[73] Jens C. Hühn and Eyke Hüllermeier. Is an ordinal class structure useful in classifier learning? *International Journal of Data Mining, Modelling and Management*, 1(1):45–67, 2009.

[74] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In Nada Lavrač, Dragan Gamberger, Hendrik Blockeel, and Ljupčo Todorovski, editors, *Machine Learning: ECML 2003*, volume 2837 of *Lecture Notes in Computer Science*, pages 145–156. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20121-2.

[75] Johannes Fürnkranz. Pairwise classification as an ensemble technique. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 97–110. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44036-9.

[76] Haibo He and E.A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[77] Chris Drummond and Robert C. Holte. Severe class imbalance: Why better algorithms aren't the answer. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 539–546. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29243-2.

[78] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[79] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2(1):263–286, 1995.

[80] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[81] Lin Dong, Eibe Frank, and Stefan Kramer. Ensembles of balanced nested dichotomies for multi-class problems. In Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *Knowledge Discovery in Databases: PKDD 2005*, volume 3721 of *Lecture Notes in Computer Science*, pages 84–95. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29244-9.

[82] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 547–553. MIT Press, 2000.

[83] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[84] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Kluwer Academic Publishers, 1998. ISBN 0-7923-8047-9.

[85] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[86] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 567–580. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40720-1.

[87] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[88] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1.

[89] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[90] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[91] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 1012–1019, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5.