Learning Distance Functions: Algorithms and Applications

Thesis for the degree of

DOCTOR of PHILOSOPHY

by

Tomer Hertz

SUBMITTED TO THE SENATE OF

THE HEBREW UNIVERSITY OF JERUSALEM

December 2006

This work was carried out under the supervision of Prof. Daphna Weinshall

Acknowledgements

Daphna Weinshall, who supervised this work, has been deeply involved in shaping my academic journey. Her intensive guidance helped me tremendously to take my first steps along my academic path, and without her shepherding I could have not traveled far. Her academic curiosity and her ability to simultaneously work in several research areas have indeed been a guiding light. I thank her for her patience, judgment, and for the efforts she has made on my behalf. Her contribution to my scientific approach is priceless.

This thesis was written under the auspices and with the support of the Interdisciplinary Center for Computational Neuroscience (ICNC). I thank the center for its financial support. The center had an active and important role in assisting me during the initial stages of my PhD. Some of the work presented in this thesis is the outcome of fruitful collaborations, all of which included other ICNC members from different research areas.

My initial steps in the program were also overseen by Alisa Shadmi, who turned the ICNC into a fun place to visit. I thank her for the personal interest and care she has shown whenever I turned to seek her help. Her advice was always very good. Ruthi Suchi who replaced her, has successfully continued this tradition, and I thank her too for all of the concern and care.

This research began during the summer of 2001, with my joint work with Daphna Weinshall and Misha Pavel. These meetings were my first plunge into the depths of the academic ocean, and the fertile ideas that were produced there led to the development of the first algorithm presented in this thesis. I am deeply indebted to Misha Pavel, who helped me in many ways when I was just starting out, and for supporting me in times when I almost gave up.

The research presented in this thesis is the outcome of several joint efforts with many friends, with whom I have spent the best times in my life:

My best friends Noam Shental and Aharon Bar-Hillel were great partners with whom I have made parts of this journey. The huge differences between us lead to very fruitful collaborations. Our friendship turned our work to be much more than mere academic achievements. I have very good memories of the endless nights we spent together working to meet various deadlines.

Chen Yanover, my friend and roommate, was the one who got me interested in computational immunology - the field I have chosen for my future studies. Our joint work, which started almost by chance, was truly enjoyable. My work with Chen was the my first independent collaboration , and I learned a lot from it. The numerous hours we spent together and the many cups of coffee we drank were very enjoyable.

I also enjoyed working with Rubi Hammer and Shaul Hochstein. Our joint work with Daphna Weinshall in the field of Cognitive Psychology was an interesting and successful interdisciplinary collaboration, which aimed at addressing the same research problem from various disciplines, including cognitive experiments, computer simulations and theoretical work.

My joint project with Daphna Weinshall, Inna Weiner and Israel (Eli) Nelken began after Eli heard a talk that I gave at the ICNC Ein-Gedi seminar. Our joint work was a great experience, and I am thankful to have been given the opportunity to work with with Eli and to learn from him.

I also thank my many friends in the corridor and lab - Ido Omer, Doron Feldman, Tamir Hazan, Amit Gruber, Ana Sorkin, Michael Fink, Shai Shalev-Schwartz and Lavi Shpigelman, with whom it was always fun to talk, drink coffee and discuss various issues.

My interest in computational neuroscience began thanks to my good friend Assaf Krauss, and I am grateful to him for the excitement he kindled in me to work in this area.

Einav, my love, has accompanied me throughout this journey, and provided love, stability and a lot of support. I cannot imagine this journey without her.

My dream to become a researcher was born in the many hours that I spent with my Dad in his cattle observations in Kibbutz Naan and Bet-Dagan, when he was working on his Ph.D. in physiology, and I am deeply indebted to him for this.

This long journey could have not taken place without the love, support and belief of my family, and especially that of my mother, who did everything she could to allow her children to realize their full potential. My grandmother, Rivka Zagon, always helped me with wonderful advice, with much love and care. This thesis is dedicated to my mother and to my grandmother.

Abstract

This thesis presents research in the field of distance learning. Distance functions are extensively used in various application domains and also serve as an important building block in many types of algorithms. Despite their abundance, until recently only canonical distance functions such as the Euclidean distance have been used, or alternatively various application specific distance functions have been suggested, which in most cases were hand-designed to incorporate domain specific knowledge. In the last several years there has been a growing body of work on algorithms for learning distance functions. A considerable amount of different distance learning algorithms have been suggested, most of which aim at learning a restricted form of distance functions called Mahalanobis metrics.

In this thesis I will present three novel distance learning algorithms:

1. **Relevant Component Analysis (RCA)** - An algorithm for learning a Mahalanobis metric using positive equivalence constraints.

2. ***DistBoost*** - A boosting based algorithm which can learn highly non-linear distance functions using equivalence constraints.

3. ***KernelBoost*** - A variant of the *DistBoost* algorithm which learns Kernel functions, which can be used in any kernel-based classifier.

I will then describe their applications to various data domains, which include clustering, image-retrieval, computational immunology, auditory data analysis and kernel-based classification. In all of these application domains, significant improvement is made when using a learned distance function instead of a standard off-the-shelf distance function. These results demonstrate the importance of this growing research field.

The first two chapters of this work present a general introduction to the field of distance functions, and distance function learning, with some additional background on semi-supervised learning:

Chapter 1 - Introduction: In Chapter 1 we provide a general introduction to distance functions, and some reasons why the distance learning problem is an important and interesting learning scenario. We then provide a detailed overview of canonical and hand-designed distance functions. The algorithms presented in this thesis are all from the field of semi-supervised learning. We therefore present a short introduction to the field of semi-supervised learning, with a specific focus on learning using equivalence constraints, which is

the learning setup that is common to many distance learning algorithms, including the ones presented in this thesis.

Chapter 2 - Algorithms for Learning Distance Functions: Chapter 2 provides a detailed overview of current research on distance learning. It suggests a taxonomy of distance learning algorithms covering several subcategories, and describes various distance learning algorithms in each of these subcategories. Additionally, it provides a detailed description of the three distance learning algorithms which are the focus of this thesis.

The remaining chapters present the various publications in which these algorithms have been presented and their use in various application domains. More specifically:

Chapter 3 - The Relevant Component Analysis Algorithm: Chapter 3 presents the Relevant Component Analysis (RCA) algorithm, which is an algorithm for learning a Mahalanobis distance metric using positive equivalence constraints. The paper discusses several theoretical justifications for the RCA algorithm which show that it is the optimal Mahalanobis metric under several interesting criteria including information maximization and maximum likelihood. The algorithm is shown to provide performance boosts when used in a data clustering task.

Chapter 4 - The *DistBoost* algorithm: Chapter 4 presents the *DistBoost* algorithm, which is an algorithm for learning highly non-linear distance functions using equivalence constraints. The algorithm is a semi-supervised boosting algorithm which is based on boosting hypothesis in the product-space (the space of all pairs of points). The algorithm is evaluated for two important applications: clustering and image retrieval, and is shown to improve performance when compared to linear distance learning algorithms such as RCA.

Chapter 5 - The *KernelBoost* algorithm: When given a small sample, we show that classification with SVM can be considerably enhanced by using a kernel function learned from the training data prior to discrimination. This kernel is also shown to enhance retrieval based on data similarity. We describe *KernelBoost* - a boosting algorithm which computes a kernel function as a combination of 'weak' space partitions. The kernel learning method naturally incorporates domain knowledge in the form of unlabeled data (i.e. in semi-supervised or transductive settings), and also in the form of labeled samples from relevant related problems (i.e. in a learning-to-learn scenario). The latter goal is accomplished by learning a *single* kernel function for all

classes. We show comparative evaluations of our method on datasets from the UCI repository. We demonstrate performance enhancement on two challenging tasks: digit classification with kernel SVM, and facial image retrieval based on image similarity as measured by the learnt kernel.

Chapter 6 - Predicting Protein-peptide Binding by Learning Distance Functions: Chapter 6 presents an application of distance learning in the field of computational immunology. In the immune system, the recognition of pathogen peptides begins when they bind to cell membrane Major Histocompatibility Complexes (MHCs). Developing computational methods for predicting protein-peptide binding is important for vaccine design and treatment of diseases like cancer. In this work we propose a novel approach for predicting binding affinity which is based on learning a peptide-peptide distance function. In order to learn these peptide-peptide distance functions, we formalize the problem as a semi-supervised distance learning problem with partial information in the form of equivalence constraints. Specifically we propose to use *DistBoost* which is a semi-supervised distance learning algorithm. We compare our method to various state-of-the-art binding prediction algorithms on MHC class I and MHC class II datasets. In almost all cases, our method outperforms all of its competitors. One of the major advantages of our novel approach is that it can also learn an affinity function over proteins for which only small amounts of labeled peptides exist. In these cases, *DistBoost*'s performance gain, when compared to other computational methods, is even more pronounced.

Chapter 7 - Analyzing Auditory Neurons by Learning Distance Functions Chapter 7 presents another application of distance learning in the field of neuronal data analysis. More specifically, we present a novel approach to the characterization of complex sensory neurons. One of the main goals of characterizing sensory neurons is to characterize dimensions in stimulus space to which the neurons are highly sensitive (causing large gradients in the neural responses) or alternatively dimensions in stimulus space to which the neuronal responses are invariant (defining iso-response manifolds). We formulate this problem as that of learning a geometry on stimulus space that is compatible with the neural responses: the distance between stimuli should be large when the responses they evoke are very different, and small when the responses they evoke are similar. Here we show how to successfully train such distance functions using a rather limited amount of information. The data consisted of the responses of neurons in primary auditory cortex (A1) of anesthetized cats to 32 stimuli derived from natural sounds. For each neuron, a subset of all pairs of stimuli was selected such that the responses of the two stimuli in a pair were either very similar or very dissimilar. The distance function was trained to fit these constraints. The resulting distance functions generalized to

predict the distances between the responses of a test stimulus and the trained stimuli.

Chapter 8 - Epilogue - in this short chapter we provide a brief discussion of the work presented and identify some directions for future research.

Publications included in this thesis:

Chapter 3 [A] Aharon Bar-Hillel, Tomer Hertz, Noam Shental and Daphna Weinshall, **Learning Distance Functions Using Equivalence Relations** *in 20th International Conference on Machine Learning (ICML 2003)*, Washington DC, August 2003.

Chapter 4 [B] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Boosting Margin Based Distance Functions for Clustering**, *in the 21st International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada July 2004.

[C] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Learning Distance Functions for Image Retrieval** *in IEEE Conf. on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

Chapter 5 [D] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Learning a Kernel Function for Classification with Small Training Samples** *in the 23rd International Conference on Machine Learning*, Pittsburgh, Pennsylvania, June 2006.

Chapter 6 [E] Tomer Hertz and Chen Yanover, **PepDist: A New Framework for Protein-Peptide Binding Prediction based on Learning Peptide Distance Functions**, *in BMC Bioinformatics*, Vol. 7 (suppl 1), March 2006.

Chapter 7 [F] Inna Weiner, Tomer Hertz, Israel Nelken and Daphna Weinshall, **Analyzing Auditory Neurons by Learning Distance Functions**, *in the 19th International Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2005.

Contents

Chapter 1

Introduction

Machine Learning is the study of methods for programming computers to learn. This area of research is focused on algorithms which evolve through experience. In the classical scenario a learning algorithm undergoes a learning stage, which makes use of a *training set* and is then evaluated on a *test set* - novel data which were not presented during the learning stage. In the last three decades, considerable advances have been made in the field of Machine learning both in terms of theory and applications. Machine learning algorithms have been successfully applied and used in various application domains varying from text analysis, data mining, computer vision, computational biology, computational neuroscience and many more.

Most of the research in machine learning has focused on *supervised learning*. In this setting, the algorithm is provided with a training set which consists of a set of labeled examples $\{x_i, y_i\}_{i=1}^N$, where $x_i \in \mathcal{X}$ denotes the input objects (or datapoints) and $y_i \in \mathcal{Y}$ denotes the output value associated with x_i. This training set is used to learn a function $f : \mathcal{X} \to \mathcal{Y}$ whose output can be either continuous (regression), or can predict the label of the input object (classification). The key challenge for a supervised learning algorithm is its capability to *generalize* - i.e. to learn a function which works well over any valid input object $x \in \mathcal{X}$ after seeing a (usually) small number of training samples (input-output pairs).

A somewhat less explored area in Machine learning is *unsupervised learning*. The task in this setting is to analyze a set of input objects $\{x_i\}_{i=1}^N$ for which no class labels y_i are provided. This area includes a wide variety of different learning tasks such as data clustering, feature extraction, visualization, density estimation, anomaly detection, information retrieval etc. (Dietterich, 2003). Can anything of value be learned from unlabeled examples? The answer depends critically on the assumptions one is willing to make about the data. If, for example, we assume that our data originate from a set of known underlying probability distributions, there are unsupervised algorithms

which can estimate the parameters of these unknown distributions (Duda and Hart, 1973). In the early 70s, Duda and Hart (1973) put forward three reasons why unsupervised learning is worth exploring: (1) Cost of labelling - Collecting large amounts of labeled data is costly and time consuming. It can therefore be highly beneficial to train a classifier on a small amount of labeled samples and then use large amounts of unlabeled data to "tune up" the learned classifier. (2) Data drifts - In many applications the input characteristics slowly change over time. Tracking these changes in an unsupervised manner can help improve performance. (3) Exploratory data analysis - When working with novel datasets it may be valuable to gain insights into their structure and nature, using unsupervised methods, to choose the correct form of the classifier that will later be trained in a supervised manner.

In recent years, there has been increasing interest in the field of *semi-supervised learning* which lies in between *supervised* and *unsupervised learning*. In this scenario we are provided with large amounts of unlabeled training data, and some limited amount of side-information. This side-information may be of various forms: it may consist of partial labels - labels over small amounts of data, or *equivalence constraints* - information over pairs of datapoints which are known to belong or not to belong to the same class. Interestingly, the two compelling rationales for this field of research are the ones suggested by Duda and Hart ((1) and (2) above) as motivations for *unsupervised learning*. In this setting it is usually (and sometimes implicitly) assumed that while the amount of side-information is not sufficient to apply classical supervised learning methods, this side- information can be used to obtain classifiers which yield better results compared to those that would be obtained without using this side-information.

1.1 Thesis Outline

The research presented in this thesis focuses on the problem of distance function learning. After a detailed introduction of the field of distance learning presented in Chapter 1, I will then provide a review of various distance learning algorithms in Chapter 2, including a detailed description of three novel distance learning algorithms which are the main contribution of this thesis. More specifically I will present the following algorithms:

1. **Relevant Component Analysis (RCA)** - An algorithm for learning a Mahalanobis metric using positive equivalence constraints.

2. ***DistBoost*** - A boosting based algorithm which can learn highly non-linear distance functions using equivalence constraints.

3. ***KernelBoost*** - A variant of the *DistBoost* algorithm which learns Kernel functions, which can be used in any kernel-based classifier.

These algorithms will be described and analyzed in detail in Chapter 2. In the remaining chapters I will proceed to show various applications of these algorithms in a wide variety of application domains, including image retrieval, data clustering, classification, protein-peptide binding prediction and also the analysis of neuronal data recorded from the auditory pathway. The empirical results presented will make a strong case for the applicability and importance of this somewhat new area of research.

In what follows I provide an introduction to the field of distance functions, beginning with some formal definitions and then moving on to a review of various canonical distance functions, hand-designed distance functions and most importantly learned distance functions. Since the important common ground for the distance learning algorithms to be presented here is that they all come from the field of *semi-supervised learning*, a short review of this area will also be presented in Section 1.3

1.1.1 Notations

Throughout this thesis I will use the following notations: vectors will be represented as lowercase letters x and may also be indexed x_i. The j-th component of a vector x_i will be denoted by x_{ij}. Sets are represented by calligraphic uppercase letters $\mathcal{X}$, $\mathcal{Y}$. A distance function will be denoted by the uppercase letter D and a similarity function will be denoted by the uppercase letter S. The symbols $\mathbb{R}$ and $\mathbb{R}^d$ denote the set of reals, and the d-dimensional real vector space respectively. Further, $\mathbb{R}_+$ denotes the set of non-negative real numbers. For $x_i, x_j \in \mathbb{R}^d$, $\|x_i\|$ denotes the L_2 norm, and $\langle x_i, x_j \rangle$, denotes the inner product. Unless otherwise mentioned, log will represent the natural logarithm.

1.2 Learning Distance Functions

The research presented in this work focuses on the problem of learning distance functions. We therefore begin with a formal definition of a *distance function* and discuss its relation to a *distance metric* and a *similarity function* (Section 1.2.1). We will then discuss several important motivations for learning distance functions in Section 1.2.3. While learning distance functions is a somewhat new area of research, much work has been done both using *canonical distance functions*, and using *hand-designed distance functions* in various application domains. A short review of such distance functions will be presented in Sections 1.2.4-1.2.6. The problem of learning distance

functions is also closely related to the problem of data representation and feature selection, and these relations will be discussed in Section 1.2.7. In Section 1.2.8 we discuss the relation between distance learning and multi class classification. A detailed description of the algorithms included in this thesis will be presented in Chapter 2. The remaining chapters present the publications included in this thesis as listed in page viii, in which various applications of these distance learning algorithms are presented.

1.2.1 Distance Functions, Distance Metrics, and Similarity Functions

A distance function is a function defined over pairs of datapoints. Given a pair of datapoints, the function produces a real (and possibly bounded) value, which measures the distance between the pair of points. Intuitively speaking, points that are similar to one another are assigned smaller values than points which are far from one another. More formally, a distance function is a function $\mathcal{D} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which assigns a real valued number for any pair of points from the input space $x_i, x_j \in \mathcal{X}$.

A special form of distance functions are also known as *distance metrics*. A distance metric D is a distance function which maps pairs of points x_i, x_j into the nonnegative reals - $\mathcal{D} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ and obeys the following three properties:

1. **Isolation (also known as 'Identity of indiscernibles') -** $D(x_i, x_j) = 0 \quad \text{iff} \quad x_i = x_j$.
2. **Symmetry -** $D(x_i, x_j) = D(x_j, x_i)$.
3. **Triangular Inequality -** $D(x_i, x_j) + D(x_j, x_k) \geq D(x_i, x_k)$.

A general distance function will not necessarily obey all of these properties. For example, if we allow $D(x_i, x_j) = 0$ for $x_i \neq x_j$, we end up with a *Pseudo-metric*. In all other cases (i.e. if we omit symmetry, or the triangular inequality) we use the general term *distance function*.

An *Ultrametric* is a distance metric which satisfies a strengthened version of the triangular inequality. In a Euclidean coordinate system, this is equivalent to requiring that the triangles of pairwise distances between every three points will be isosceles triangles with the unequal length no longer than the length of the two equal sides (Hastie *et al.*, 2001) - i.e. for any three points $x_i, x_j, x_k \in \mathcal{X}$

$$D(x_i, x_j) \leq \max\{D(x_i, x_k), D(x_j, x_k)\}$$

A concept which is closely related to the a *distance function* is a *similarity function*. A *similarity function* is a function defined over pairs of points which measures the similarity (or resemblance) of the two points. It is

however easy to see that a similarity function is inversely related to a distance function - if a pair of points are very similar to one another, we would expect the distance between them to be small. Therefore, there are several intuitive ways of transforming a similarity function into a distance function and vice-versa. One commonly used example is the following:

$$D(x_i, x_j) = e^{-S(x_i, x_j)}$$

where $D(x_i, x_j)$ is a distance function and $S(x_i, x_j)$ is a similarity function. If we assume that the similarity function is bounded in the range of $[0, 1]$ another widely used transformation is:

$$D(x_i, x_j) = 1 - S(x_i, x_j)$$

Therefore, while we will mostly use the term *distance function*, it should be clear that algorithms which learn distance functions are also used for learning similarity functions.

One important and widely used type of similarity functions are *kernel functions*. A *kernel* $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a function that for any $x_i, x_j \in \mathcal{X}$ satisfies:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

where $\phi : x \rightarrow \phi(x) \in F$ is a mapping from $\mathcal{X}$ to an inner product feature space F. For example, one widely used kernel function is the *polynomial kernel* of degree 2 given by:

$$k(x_i x_j) = \langle x_i, x_j \rangle^2$$

which corresponds to the feature map:

$$\phi(x) = (x_{i_l} \cdot x_{i_m})_{l,m=1}^{n} \in F = \mathbb{R}^{n^2}$$

A kernel function k can be used to define a *Gram matrix* (also known as a *kernel matrix*). Given a set of vectors $S = \{x_1, \ldots x_l\} \in \mathcal{X}$ the *Gram matrix* is defined as the $l \times l$ matrix G whose entries $G_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$. The Gram matrix is in essence a *similarity matrix* of the set of vectors S. It also has several other appealing properties: it is symmetric and *positive semi-definite*. In fact it can be shown that any positive semi-definite symmetric matrix corresponds to some kernel function k (Shawe-Taylor and Cristianini, 2004).

1.2.2 Non-Metric distance functions

While distance metrics and kernels are widely used by various powerful algorithms, they work well only in cases where their axioms hold. However, in some cases, the 'natural' distances between objects do not conform to these strict axioms. For example Jacobs *et al.* (2000) have shown that distance functions which are robust to outliers and irrelevant aspects in the matching of pairs of input objects are not metric, as they tend to violate the triangular inequality. Examples of such distance functions are common in machine vision, in which many times images are compared using part-based comparisons, and also in similarity judgments provided by humans. Human similarity judgments have been extensively studied by Tversky (1977), who showed that they often violate both the symmetry and triangular inequality metric properties. In other contexts, it is sometimes the *'Identity of indiscernibles'* that is violated. For example Mahamud and Hebert (2003b) has shown that the optimal distance function for nearest-neighbor classification violates this property. An additional example is given by Bar-hillel and Weinshall (2003) who analyzed the family of binary distance functions, which also violate this property. Finally, a large number of *hand-designed distance functions* have been suggested in various application domains, as discussed below in Section 1.2.6. In almost all of these cases, these distance functions are far from being metric.

Various works have suggested formulations of more general non-metric similarity functions. Lin (1998) derived a definition of similarity that is based on information theory, for discrete valued vectors. The similarity measure can be learned from event frequencies and has been shown to work well in the context of document retrieval (Aslam and Frost, 2003). A different formulation was suggested by Kemp *et al.* (2005), in which similarity judgements are inferences about generative processes and that the similarity of two objects is defined using the likelihood that they were generated by the same underlying generative process. Bar-Hillel and Weinshall (2006) suggest another information theoretic definition which is similar to the one suggested by Kemp *et al.* (2005), and different from the one considered by Lin (1998). Bar-Hillel and Weinshall (2006) suggest an efficient similarity learning algorithm which can be used for continuous valued vectors. Balcan and Blum (2006) have recently suggested a novel formulation of similarity which provides an alternative to the widely used family of kernel functions. The authors suggest sufficient conditions for a similarity function that will allow one to learn well, which does not require reference to implicit spaces and does not require the function to be positive semi-definite (PSD).

1.2.3 Why are distance functions important?

While the idea of explicitly learning a distance function is rather new, distance functions have been widely used in various application domains and for various computational tasks. This is primarily due to the abundance of algorithms which are *distance based* - i.e. algorithms whose only input requirement is the pairwise distances between the input datapoints. Despite this somewhat straightforward motivation, I believe that there are several additional important reasons why learning a distance function is an interesting and important computational challenge. Let me now present and discuss these motivations:

- **Distance-based algorithms** - Many different supervised and unsupervised learning algorithms make use of the distances between the training datapoints. Examples include graph-based clustering methods such as *average linkage* (Duda and Hart, 1973), normalized-cut (Shi and Malik, 2000), nearest neighbor classifiers (Fukunaga, 1990) and kernel-based classifiers such as support vector machines (SVMs) (Vapnik, 1998). It is widely known that the performance of all of these algorithms is strongly dependent on the distance function used. This is not surprising since in all of these algorithms, the only required input are the distances (or similarities) between datapoints, and the datapoints themselves are not used directly at any stage of the algorithm. An example of this intricate connection will be provided in Chapter 4, publication [B], where we will show that the performance of various hierarchical graph-based algorithms (such as the *average linkage* algorithm), critically relies on the quality of the distance function used.

- **The *curse of dimensionality*** - The curse of dimensionality (Bellman, 1961) refers to the exponential growth of hypervolume as a function of the dimensionality. This is a problem that many learning algorithms suffer from. The importance of choosing the right distance function becomes even more critical when we consider high-dimensional data such as images. It has been recently shown that the quality of the distance function may be strongly affected by the sparsity of the data (Katayama and Satoh, 2001). Moreover, Beyer *et al.* (1999) and Aggarwal *et al.* (2001) have shown that for high-dimensional data when standard L_p-norms are used (such as the L_2 norm which is equivalent to the squared Euclidean distance), the distances between all pairs of points are very similar to one another.

- ***Learning to Learn*** - One of the major current differences between humans and machines in the field of learning is humans' stunning ability to learn new tasks based on previously acquired experience in related tasks. Unlike machines, humans are extremely good at learning new tasks which are similar (or related) to

tasks which they have previously encountered. Moreover, people are able to learn these new tasks with very limited amount of training, sometimes even using a *single* example. This impressive ability to generalize from previously related tasks to a novel task is called *'Learning to Learn'* (Thrun and Pratt, 1998) and is also known as *Inductive transfer* (Caruana, 1997), *Interclass transfer* (Fink *et al.*, 2006) and *'Learning with point sets'* (Minka and Picard, 1997). This field of research has attracted increased attention in recent years from the computational community both in terms of theory (Thrun, 1996; Baxter, 1997; Ben-David and Schuller, 2003) as well as applications (Ferencz *et al.*, 2005; Fink, 2004; Hertz *et al.*, 2006). Various ideas on what knowledge can be transferred between related tasks, and how such knowledge can be transferred have been discussed in the literature, and have resulted in such notions as learning priors (Baxter, 1995), feature selection, and learning distance functions (Caruana, 1996; Thrun, 1996; Hertz *et al.*, 2006).

A good example of how distance functions can be used for interclass transfer can be seen when considering a facial image retrieval system. Suppose that our task is to build a facial image retrieval system in which the user provides a query image and the system retrieves a set of images which are the most similar from a given database of facial images. The retrieval system is powered by some similarity function which is used for measuring the similarity between pairs of images. Faces are in general semi-rigid objects, in which one can easily identify a set of features or parts which are consistent across all faces. These shared features can be exploited for sharing information across different faces. If we train a distance learning algorithm over a set of training images of several subjects, we can use this distance function to measure the distances between novel images of these subjects (which is considered the classical 'test stage' of any learning algorithm), but we can also use it to measure the distances of faces of new subjects which were not presented during the training stage (see Chapter 5 for a detailed empirical study of such an application on a benchmark dataset of facial images).

- **C**apturing relations between datapoints - In the classical learning scenario, we attempt to learn some function f over an input space $\mathcal{X}$, from a given training sample $S = \{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$ where $y_i = f(x_i)$. Distance functions are functions over pairs of points - they provide information about the similarity of pairs of points. Essentially, a distance function captures relations between the input datapoints themselves, and not an input-output relation between the datapoints and their associated label (which can also be continuous). However, in some cases capturing the relations between datapoints can provide information which cannot be easily extracted from directly estimating input-output relations.

An example of such a case can be seen in clustering. Suppose that our input-output function essentially clusters the input data into a set of well separated clusters, each with an associated label. One interesting property of such a function is that we can obtain an identical set of clusters from a large family of other related functions. The commonality of all of these functions is that they capture the same pairwise relations between the input datapoints - i.e. if points x_i and x_j were assigned to the same cluster by f_1 they will also be assigned to the same (but possibly different) cluster by f_2. Capturing the relatedness of these functions can be easily obtained by directly learning these pairwise relations. Therefore, we may be able to capture non-intuitive relations between various input-output functions by characterizing the pairwise relations that they induce on the input objects. One interesting example of such an approach will be presented in Chapter 7 in which we show how such a formulation can be used to characterize auditory neurons using a set of natural bird chirp stimuli.

- **A**n alternative to feature selection - Learning a distance function is closely related to both feature selection and data representation. In fact, as discussed in more detail in Section 1.2.7 these tasks are somewhat interchangeable and therefore learning a distance function can be viewed as an alternative approach to these two important tasks.

Despite the abundance of distance-based learning algorithms, and the additional motivations presented above, until recently the distance functions which were traditionally used were various standard *off-the-shelf* distance metrics such as the Euclidean distance, Mahalanobis distance, or various context dependant distance measures which were constructed by hand. Recently however, various authors have explored the idea of developing methods for learning the distance function using a training data which is either labeled or is augmented with some form of side-information. This research topic has received growing attention in recent years and many different distance learning algorithms have been developed and used successfully in various application domains. However, before we focus on these distance learning algorithms, let us provide a brief introduction to a number of well known distance functions which have been widely used in various application domains and algorithms.

1.2.4 Canonical Distance Functions

Let us now turn to a brief overview of several canonical distance (and similarity) measures:

- **Euclidean Distance** - Perhaps the most well known and widely used distance function (which is also a

metric) is the Euclidean distance defined by

$$D_{Euclidean}(x_i, x_j) = \sqrt{(x_i - x_j)^2} = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2} \tag{1.1}$$

The Euclidean distance is also known as the L_2 distance (or the squared L_2 norm).

- **Mahalanobis Distance** - This distance measure, originally proposed by P.C. Mahalanobis (Mahalanobis, 1936) in the statistics community, is based on correlations between different features within a feature vector. The Mahalanobis distance is a generalization of the Euclidean distance which also takes into account the correlations of the dataset and is scale-invariant. In its original form it measures the distance of any two vectors, based on the assumption that they originate from the same underlying distribution. Formally, given a distribution p which has a covariance matrix Σ, the Mahalanobis distance between two vectors x_i, x_j is given by:

$$D_{Mahalanobis}(x_i, x_j) = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)} = \sqrt{\sum_{k=1}^{d}\sum_{l=1}^{d} x_{ik}\Sigma^{-1}_{lk} x_{jl}} \tag{1.2}$$

  Note that if the covariance matrix Σ is the identity matrix, then the Mahalanobis distance becomes the Euclidean distance. If we restrict the covariance matrix to be diagonal, we now have what is called a *normalized Euclidean distance*. While in its original form the Mahalanobis distance assumes that the datapoints originate from a probability distribution with a covariance matrix Σ, it can be shown that the distance is well defined for any positive semi-definite (PSD) matrix A. We will therefore denote a general Mahalanobis matrix by the symbol A. We will describe several algorithms for learning a Mahalanobis metric in Chapter 2.

- **Manhattan (or City-Block) Distance, L_1 distance** - This distance, originally proposed by Minkowsky is defined by

$$D_{Manhattan}(x_i, x_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}| \tag{1.3}$$

  The distance measures the shortest distance (in "city blocks") that one would be required to walk between the two points x_i and x_j if a city is laid out in square blocks. More formally, it is the sum of the lengths of the projections of the line segments between the points onto the coordinate axes of the coordinate system.

- **Chebychev distance (or chessboard distance)** - The Chebychev distance between two points is the maximum distance between the points in any single dimension:

$$D_{Chebychev}(x_i, x_j) = \max_k |x_{ik} - x_{jk}| \tag{1.4}$$

This distance measure is a special case of the L_∞ norm. It may be appropriate if the difference between points is reflected more by differences in individual dimensions rather than all the dimensions considered together.

- **Minkowski distance** - The Minkowski distance is a generalization of several other canonical distances and is also known as the L_p norm distance. Note that unlike the previous distance metrics, this metric has a free parameter p which must be defined. The distance is given by

$$D_{Minkowski}(x_i, x_j) = \sqrt[p]{\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p} \tag{1.5}$$

When $p = 1$ this yields the Manhattan distance. When $p = 2$ we obtain the Euclidean distance and finally when $p = \infty$ we obtain the Chebychev distance. However, we can also pick different values for p. In general, as the value of p increases the metric tends towards a Chebychev result. Therefore by increasing p, one can assign greater numerical value to the largest distance (in terms of elements in the two vectors in question).

- **Hamming Distance** - The Hamming distance originally introduced by Richard Hamming in the field of information theory (Hamming, 1950) is a distance measure between two strings of equal length, which is defined as the number of positions for which the corresponding symbols are different. Put differently, it measures the number of "errors" that transformed one string into the other. More formally it is given by:

$$D_{Hamming}(x_i, x_j) = \sum_{k=1}^{d} \mathbf{1}(x_i \neq x_j) \tag{1.6}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function ($\mathbf{1}\{True\} = 1, \mathbf{1}\{False\} = 0$).

It can be easily shown that for a fixed length n the Hamming distance is a metric on the vector space of all words of that length. Additionally the Hamming distance of binary strings $x_{ik} \in 0, 1$ is equivalent to the Manhattan distance between the two points in a d-dimensional hypercube, where d is the length of the words.

- **Correlation Distance (or Pearson correlation distance)** - The correlation distance measures the similarity in shape of two feature vectors. More formally it is the dot product of the Z-scores of the two vectors. x_i, x_j given by

$$D_{Correlation}(x_i, x_j) = 1 - \frac{\langle Z(x_i), Z(x_j) \rangle}{N} = 1 - \frac{\sum_{k=1}^{d} Z(x_{ik}) Z(x_{jk})}{N} \tag{1.7}$$

where $Z(x_{ik}) = (x_{ik} - \mu_k)/\sigma_k$, μ is the empirical mean of the data, and σ is the standard deviation. The term $\langle Z(x_i), Z(x_j)\rangle/N$ is sometimes referred to as the *Correlation similarity*.

- **The Jaccard Similarity Coefficient (or Index) -** The Jaccard index is a statistic for computing the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets. More formally, given two sample sets A and B the Jaccard similarity coefficient is given by:

$$S_{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1.8}$$

  This similarity measure has been widely used in areas such as text classification, where one natural way of representing a document is as a "bag of words", which is simply the set of all words within the document (sometimes omitting frequently used words).

  A closely related measure is the *Jaccard distance*, which is obtained by subtracting the size of the intersection of the sets by the size of the union and dividing the result by the size of the union:

$$D_{Jaccard}(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} = 1 - S_{Jaccard}(A, B) \tag{1.9}$$

- **Cosine Similarity** - The Cosine similarity is a similarity which is widely used for clustering *directional data* - data, that deals only with the direction of unit vectors, i.e. which only measures the relative direction between pairs of vectors. More formally it is given by

$$S_{Cosine}(x_i, x_j) = \frac{\langle x_i, x_j\rangle}{\|x_i\| \cdot \|x_j\|} = \sum_{k=1}^{d} \frac{x_{ik} \cdot x_{jk}}{\sqrt{\sum_{k=1}^{d} x_{ik}^2} \cdot \sqrt{\sum_{k=1}^{d} x_{jk}^2}} \tag{1.10}$$

  It has been shown that the Cosine similarity measure is the natural distortion measure for prototype-based clustering under the assumption that the data were generated by a mixture of von-Mises Fisher distributions (Banerjee *et al.*, 2003). This similarity measure has been widely used in information retrieval applications including text analysis (Aggarwal, 2003), bioinformatics and collaborative filtering (Banerjee *et al.,* 2003). It has also been shown by Banerjee *et al.* (2003) that the Pearson correlation is a form of cosine similarity.

  The **Weighted (or Parametrized) Cosine similarity** is a generalization of the Cosine similarity in which a positive-definite matrix A is used:

$$D_{WeightedCosine}(x_i, x_j) = \frac{x_i A x_j}{\|x_i\|_A \cdot \|x_j\|_A} \tag{1.11}$$

  where $\|x\|_A$ is the weighted L_2 norm: $\|x\|_A = \sqrt{x^T A x}$.

Let us now review another family of distance measures, which are defined over pairs of probability distribution functions.

1.2.5 Distances between distributions

Many different approaches have been suggested for measuring the distance between a pair of probability distributions p and q. This is an important problem, which has been extensively studied in many application domains such as information theory, image retrieval etc. For purposes of simplicity let us focus on the case of discrete probability distributions. Furthermore, we will represent each probability function using an empirical histogram H.

A histogram $H = \{h_i\}_{i=1}^{k}$ is a mapping from a set of d-dimensional integer vectors to the set of non-negative reals. These d-dimensional integer vectors typically represent bins (or their centers) in a fixed partitioning of the relevant region of the underlying feature space, and the associated reals are a measure of the mass of the distribution that falls into the corresponding bin. For example, in a grey-level histogram of an image, d is equal to one, the set of possible grey values is split into k intervals, and h_i is the number of pixels in an image that have a grey value in the interval indexed by i (a scalar in this case) (Rubner *et al.*, 2000).

We now review several important examples of such distance measures. In the following, H^1 and H^2 denote histograms and h_i^1 denotes the i'th bin in histogram H^1.

- χ^2 **statistic** - The Chi-Square statistic (Schervish, 1995) measures how unlikely it is that one distribution was drawn from the population represented by the other distribution. More formally it is given by:

$$D_{\chi^2}(H^1, H^2) = \sum_i \frac{(h_i - m_i)^2}{m_i} \tag{1.12}$$

  where $m_i = \frac{h_i^1 + h_i^2}{2}$. This is a widely used distance measure in the statistics community and has also been widely used for the analysis of neuronal data (Bar-Yosef and Nelken, 2006).

- **Kullback-Leibiler Divergence** - The Kullback-Leibiler divergence (Cover and Thomas, 1991) is defined by:

$$D_{KL}(H^1, H^2) = \sum_i h_i^1 log \frac{h_i^1}{h_i^2} \tag{1.13}$$

  This divergence, which originated in information theory, measures how inefficient on average it would be to code one histogram using the other histogram as the code-book. This divergence is non-symmetric and

also sensitive to the histogram binning. One way to overcome these possible shortcomings is to use the empirically derived *Jefferey divergence* (also known as the symmetric KL-divergence) given by:

$$D_{Symmetric-KL}(H^1, H^2) = \sum_i \left(h_i^1 \log \frac{h_i^1}{m_i} + h_i^2 \log \frac{h_i^2}{m_i} \right) \tag{1.14}$$

where $m_i = \frac{h_i^1 + h_i^2}{2}$.

- **Earth Mover's Distance (EMD)** - This distance measure, originally used by Peleg *et al.* (1989) and Rubner *et al.* (2000) for measuring distance between images, is based on the minimal cost that must be paid to transform one distribution into the other. Unlike the *KL divergence* or the χ^2 *distance* it is a distance measure which also compares non-corresponding bins within the histogram. If we define the ground distance between a pair of bins $i \in H^1$ and $j \in H^2$ to be d_{ij} and by f_{ij} the flow between bins i and j the EMD distance is given by:

$$D_{EMD}(H^1, H^2) = \frac{\sum_i \sum_j d_{ij} f_{ij}}{\sum_i sum_j f_{ij}} \tag{1.15}$$

  Intuitively, given two distributions, one can be seen as a mass of earth properly spread in space, the other as a collection of holes in that same space. The EMD measures the least amount of work needed to fill the holes with earth. Here, a unit of work corresponds to transporting a unit of earth by a unit of ground distance. Computing the EMD is based on a solution to the well-known transportation problem (Rubner *et al.*, 2000).

Despite the extensive use of these (and various other) canonical distance functions in various application domains, in the last three decades, considerable efforts have been invested in hand designing application specific distance functions which incorporate some form of domain knowledge into the distance computation. Let us now review several well known hand-designed application specific distance functions.

1.2.6 Application specific Distance Functions

According to Aggarwal (2003), the process of designing application specific distance functions has intrigued researchers for over three decades. While there are several well known distance functions for datapoints that are represented in some vector space $\mathbb{R}^N$, there are many application domains in which the data instances are various objects which cannot be readily represented by a feature vector in some vector space. Classical examples are images, biological sequences, and text documents.

Consider for example the problem of measuring the distance (or similarity) between a pair of images. Each image is represented by a matrix where the value in position (i, j) denotes the gray-scale (or color) value of the (i, j)'th pixel. If we assume that a pair of images I_1 and I_2 are of equal size, then we can readily transform each image into a vector by a simple concatenation of its rows. However, this can be easily shown to be a fairly bad representation of the image since it is very sensitive to small changes in the image. If for example we are given two images of the same object in which in one image the object has been slightly moved to the right (or left), the location of object pixels within this vector representation will be very different. This would cause any canonical distance measure (e.g. the Euclidean distance) to define a 'large' distance between the vectors representing these two images of the same object. Similar effects will also occur if we allow the object to be slightly rotated, or scaled.

In these structured domains, where the objects which we would like to classify (or cluster etc.) cannot be naturally represented by some feature vector, considerable work has focused on the following approaches: (1) Finding good feature representations, and (2) Hand-designing various distance measures which can measure the distances between such objects. The intricate relationship between these two approaches will be further discussed in Section 1.2.7, but before doing this, let us focus on these *hand-designed* distance functions.

The basic idea of a hand-designed distance measure is to tailor a distance measure which incorporates some form of domain knowledge. It should be made clear that almost all of these functions are heuristic in that they are "optimized" manually, and in many cases focus on specific characteristics of the data which were hand-picked. Their "claim-to-fame" is therefore usually based on empirical evaluations and comparisons. Despite the heuristic nature of most of these distance functions, it should be noted that in some applications, these hand-designed distance functions demonstrate state-of-the art performance. One well known example is the *Shape-Context Matching* distance proposed by Belongie *et al.* (2002) (see details below) which when used with a 1-Nearest Neighbor classifier on the MNIST (LeCun *et al.*, 1998) dataset has been shown to outperform all other classification methods, most of which include a training phase which makes use of 60000 datapoints.

Let us now describe several prototypical examples of such hand-designed distance functions for various application domains:

- **Shape-context matching** - Shape context matching is a similarity measure between shapes which is used to measure the similarity of pairs of images (Belongie *et al.*, 2002). This similarity measurement makes use of various image processing and computer vision techniques which can be used automatically without the

need of any form of feedback, or learning stage. The basic idea is to use hand-defined image features which are called *shape-context* features. The shape context of a reference point captures the distribution of the remaining points relative to it, and therefore offers some form of global characteristic of the image (hence the use of the term "context"). After extracting these features from a pair of images, the method then finds the correspondences between the feature points on the two shapes, and then uses these correspondences to estimate a global transform which best aligns the two shapes. The dissimilarity (or distance) between the two shapes is computed as the sum of matching errors between the corresponding points, together with an additional term that measures the magnitude of the aligning transform.

Computing this distance on a pair of images is a computationally costly process, which can take up to several seconds, depending on the number of sample points used to select the features and the size of the images. This is therefore one classical example of a complex highly specific distance measure which is computationally intense, but generates impressive empirical results on various image retrieval and classification applications (Belongie *et al.*, 2002; Zhang and Malik, 2003; Berg *et al.*, 2005).

- **Chamfer distance** - The Chamfer distance is another shape matching algorithm which is used for shape-based object detection and recognition. At the core of this distance measure is a *distance transform*. Chamfer distance transformations rely on the assumption that it is possible to deduce the value of the distance at a pixel from the value of the distance at its neighbors. In order to compute the distance transform on an image, the image is first transformed into a feature image, which is done using some feature extraction method over the image such as edge detection. The feature image is then transformed into a distance image using the distance transform. Several distance transforms have been suggested and analyzed. Fundamentally they are all based on measuring the minimal distance of every "on" pixel to the closest "off" pixel given a binary image. For example the d_4 distance transform measures this distance based solely on the 4 close neighbors of a given pixel. Matching is then done by aligning a pair of distance images to one another, or by attempting to align a model (or template) with a single distance image.

- **Edit-distance (also known as the Levenshtein Distance)** - The edit distance was originally defined by Levenshtein (1965) as a method for measuring the similarity of pairs of strings. The basic idea is to find a sequence of *edit* operations which transform one string into the other, at a minimal cost. The elementary operations are substitution, deletion and insertion of string symbols. This distance can also be considered a generalization of the Hamming distance, which is defined only on strings of equal length and only con-

siders substitution edits. The edit-distance is usually computed by a bottom-up approach using dynamic programming. Various variants of the basic algorithm have been suggested, which in general refine the cost of various edit operations. A domain in which many of these improvements have been considered is in algorithms for aligning biological sequences such as proteins.

- **Alignment scores for biological sequences** - Aligning pairs and sets of biological sequences such as proteins, genes, etc, is a problem for which various algorithms have been proposed (Durbin *et al.*, 1998). This is a fundamental problem in computational biology, and serves as a building block for many other computational tasks which have been addressed. Alignment algorithms are in most cases specific variants of the basic edit-distance method, which make use of domain knowledge to provide refined definitions of the cost of various edit operations. These models are all based on evolutionary models which attempt to mimic the natural (and slow) process of biological evolution. When a protein is manufactured in a cell, it will sometimes undergo various mutations - which can basically consist of a substitution of one amino-acid by another, or omitting or inserting a single amino acid (or a sequence of such amino-acids). Since some amino-acids have similar chemical and electrical properties, we can define a different substitution cost for each pair of amino acids which is based on their pairwise similarity. Intuitively, substituting an amino-acid for a similar amino-acid would result in a lower cost than replacing it with a very dissimilar amino-acid.

- **Cophenetic distance** - the cophenetic distance is used for measuring the distances between clusters generated by a hierarchical clustering algorithm. Given a dendrogram D of N points the cophenetic distance (or dissimilarity) between points x_i and x_j is the inter-group distance of the cluster C_k in which points x_i and x_j are first joined together in the dendrogram. This is a very restrictive distance measure to begin with, since only $N-1$ values within the entire $N(N-1)/2$ values are distinct. Moreover, the cophenetic distances obey the *ultrametric inequality* (Hastie *et al.*, 2001), i.e. for any three points x_i, x_j, x_k we have:

$$D_{cophonetic}(x_i, x_j) \leq max\{D_{cophenetic}(x_i, x_k), D_{cophenetic}(x_j, x_k)\} \tag{1.16}$$

  This distance is typically used to measure the quality of the dendrogram provided by some clustering algorithm. This is done by measuring the *cophenetic correlation coefficient* which is the correlation between the pairwise distances used by the clustering algorithm and the corresponding cophenetic distances.

- **String Kernels** - String kernels are sequence similarity measures which have been widely used in text classification. These kernels are an excellent example of how one may define a similarity (or distance) measure

over objects which cannot be naturally represented by feature vectors. Representing a text document in order to classify documents is a hard problem for which various solutions have been proposed. Among these are the *Bag of Words* model and a mapping of each document into a high-dimensional binary vector in which each entry represents the presence or absence of a feature (usually simply a word, excluding specific 'stop words'), (Salton *et al.*, 1975).

String kernels take a somewhat different approach: each document is represented as a "string" which is simply a sequence of symbols from a given (and finite) alphabet. The main idea of string kernels is to compare documents not by words, but by the substrings they contain (Lodhi *et al.*, 2002). These substrings do not need to be contiguous, but they receive different weighting according to the degree of contiguity. For example: the substring "c-a-r" is present both in the word "card" and "custard" but with different weightings. The main advantage of this approach is that it can detect words with different suffixes or prefixes: the words "microcomputers", "computers" and "computer-based" all share common substrings. Several string kernels have also been suggested for measuring the similarities of biological sequences. Examples are the *spectrum kernel* (Leslie *et al.*, 2002) and the *mismatch kernel* (Leslie *et al.*, 2003).

- **Measuring distances between silhouettes** - Gdalyahu and Weinshall (1999) suggested an algorithm for classifying silhouettes based on measuring the similarity between pairs of silhouettes. Each image of a silhouette is first represented using a set of contours (or line segments) which are segmented using the K-means algorithm. These provide a syntactic representation of the image. The algorithm then identifies points of high curvature which are used as feature points to represent each contour. A global alignment procedure is then used, which is based on an *edit-distance*, which is specifically tailored for the vision domain: substitution is interpreted as matching two shape primitives and the substitution cost is defined by the distance between the matched primitives. Additional operations are added which include a gap insertion and the merging of primitives. Moreover, the distance between the line segments is hand-defined and depends on the scale and orientation of the segments. This similarity is based on the (plausible) assumption that an object may be pictured at different scales and rotations. Making use of this domain-specific knowledge enables the definition of a similarity measure which is based on a flexible matching algorithm. This flexible matching algorithm can match curves which are only weakly similar to one another.

- **Kernels for measuring the similarity between images** - Measuring the similarity between a pair of images is a hard problem which has been recently addressed by several authors (Grauman and Darrell, 2005;

Wallraven *et al.*, 2003; Lyu, 2005; Kondor and Jebara, 2003; Wolf and Shashua, 2003; Moreno *et al.*, 2003; Shashua and Hazan, 2005). Many image representations consist of unordered sets of features or parts, where the sets may be of varying cardinality. For example, an image may be represented by a set of local features which are detected using some interest point operator (see for example (Lowe, 2004)). An object may be represented by a set of patches which correspond to various object parts (Bar-Hillel *et al.*, 2005c,a). The major challenge of using these representations is that most machine learning algorithms are designed to operate on fixed-length input vectors, and not on a set of unordered features. A classical way to address this problem is to compute pairwise correspondences over these feature sets (see for example the *Shape-Context Matching* method described above). Another way to address this challenge is to define a kernel over these unordered feature sets which can measure the similarity of these sets. By defining such a kernel, we can then use any kernel-based classifier over this kernel without the explicit need to solve the correspondence problem between these unordered sets of features. For example Grauman and Darrell (2005) proposed a *pyramid match kernel* - which measures the similarity of a pair of unordered sets of features by mapping each feature set into a multi-resolution histogram and then compares these histograms using a weighted histogram intersection computation. They show how this distance measure can be successfully used on a challenging object recognition task.

1.2.7 The relation to data representation, feature selection and feature weighting

Many classification and clustering algorithms operate over a dataset S in which each datapoint is represented by some *feature vector*. In most cases these vectors are assumed to be ordered, i.e. each dimension within the vector is assumed to represent some measurement or a specific feature (e.g. color, size) of the input instance. Finding a good representation of the input data is known as the *data representation problem*. This problem has been extensively studied over the last several decades, and numerous representation schemes have been suggested for various input domains. For example, if we want to represent an image of some object, we can represent the image using a vector of its pixels, use some dimensionality reduction method such as Principal Component Analysis (PCA) over the original pixel vectors, represent the image using a color (or grayscale) histogram, or represent the image using a set of wavelet coefficients. These are a few of the numerous vector representations which have been suggested and explored for images. It should be clear that for each different input domain, different representations can be defined, with varying degrees of complexity and incorporation of domain knowledge.

The reason the representation problem has received enormous attention is that by improving the representation of the data, one can significantly enhance the performance of the clustering (or classification) algorithm over which it is applied. In fact, finding the optimal (or *ideal*) representation can eliminate the need for further clustering or classification the data, since such an ideal representation would map each input instance into its cluster or class label. Suppose for example that our task is to classify a set of input images into two categories - 'facial images' and 'non-facial images'. If we had a good representation of these facial images - i.e. one that would represent facial images in a very different way than non-facial images , the classification task would become very easy, and a very simple classifier such as a linear separator would easily provide perfect classification performance. However, if our representation mapped facial images and non-facial images very similarly, the classification task would become much harder, and in order to obtain good performance we would probably need to use more sophisticated classifiers to solve the problem [1].

In general the success of many learning algorithms is often strongly dependent on various assumptions which they make about the data representation - i.e. about the feature space in which the input data objects are represented: Classes are assumed to be convex, or at least continuous, and at least some of the features are expected to be relevant for predicting the class label of the input instances. However, in most cases, the data representations used in various application domains are far from ideal, and some of these assumptions do not hold. Put differently, in most cases, the data representation is rather 'weak' and is usually based on some standard off-the-shelf domain specific features.

One way to improve such 'weak' representations is to apply some pre-processing transformation F to the input datapoints. The transformation attempts to map the datapoints into a feature space in which the data are 'better' represented. In many cases, this pre-processing transformation can also be used to reduce the dimensionality of the input space, by selecting 'relevant' dimensions, or by eliminating 'non-relevant' dimensions. This problem is also known as the *feature selection* problem, for which various algorithms have been suggested (see Langley (1994) for a review of traditional methods and Guyon *et al.* (2006) for a more recent summary).

The representation problem is closely related to the distance learning problem. Since, as noted above in Section 1.2.3 many algorithms are *distance based* (i.e. they only require as input the distances between datapoints), selecting a 'better' distance function will improve the performance of these algorithms. In other words, there is an analogy here - finding a better distance function for a distance based algorithm is just like finding a better

[1] This is one of the main motivations for *kernel-based classifiers* which attempt to map the original datapoints into some high-dimensional (possibly infinite) feature space, in which the data would hopefully become linearly separable.

representation for a feature-based algorithm.

However, in some cases, the connections between data representation and distance functions can be made more explicit. Consider for example distance-based clustering algorithms such as linkage algorithms. It is well known that the performance of these algorithms depends to a great extent on the quality of the distance function used. Therefore, one way to improve the performance of these algorithms is to improve the quality of the distance function used to compute their input distance matrix. However, note that if we had an *ideal* representation of our data (i.e. one that would map every datapoint into a set of well separated feature vectors, each of which would represent a single class within our data), *any* simple (and non-trivial) distance function we chose would provide perfect clustering results.

Another more formal example of the connection between distance functions and data representation can be seen when considering the relation of the Mahalanobis distance metric to linear transformations. Since a Mahalanobis distance matrix A is a symmetric P.S.D matrix, it can be decomposed using singular value decomposition (SVD) as follows:

$$A = U\Sigma U^T = (U\Sigma^{-\frac{1}{2}})(\Sigma^{-\frac{1}{2}}U^T) = B^T B \tag{1.17}$$

where U is an orthonormal matrix ($UU^T = I$) where each column is an *eigen-vector* of the matrix A, and Σ is a diagonal matrix which holds the *singular values* of the matrix A (that are equal to the squared *eigen-values* of the matrix), and $B = A^{\frac{1}{2}}$.

It can therefore clearly be seen that using the Mahalanobis distance metric defined by A is equivalent to applying a linear transformation of the data using the matrix B and then measuring the Euclidean distance between the transformed datapoints:

$$x^T Ax = x^T(B^T B)x = (x^T B^T)(Bx) = (Bx)^T(Bx) \tag{1.18}$$

Therefore, finding an optimal Mahalanobis metric A is equivalent to finding an optimal linear transformation B and then using the Euclidean distance metric over the transformed space.

When the Mahalanobis metric considered is of low rank, it is equivalent to a *linear projection* of the data. Linear projections have been widely used in various application domains. One well known supervised algorithm for learning a linear projection of the data is Linear Discriminant Analysis (LDA, also known as FLD), which was originally suggested by Fisher (1936). Learning LDA from equivalence constraints has been suggested by Bar-Hillel *et al.* (2003) and Bie *et al.* (2003a) and also by Bar-Hillel and Weinshall (2006) (see chapter 2 for more details).

However, in the case of a general distance function D there is no principled way to find the (possibly non-linear) transformation which can be used to represent the data in some feature space where the Euclidean distance between the data points would provide the same pairwise distances. One possible approach to this problem, which has been explored in the literature, is based on *embedding*. The classical problem of embedding is to take a set of datapoints for which pairwise distances are provided, and to *embed* them into some low-dimensional Euclidean space, in which the pairwise Euclidean distances between the datapoints would be minimally distorted. Several algorithms have been suggested for this problem, including Local Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum *et al.*, 2000) and BoostMap (Athitsos *et al.*, 2004).

Distance learning algorithms are a principled way for going in the opposite direction - they are provided with a set of datapoints that lie in some vectorial feature space, and with some additional side-information on the distances (or more commonly relations) between some pairs of datapoints, and attempt to learn a distance function in which the distances between pairs of datapoints will reflect the side information provided. Learning distance functions can therefore be seen as an alternative to finding a good representation of the data, using some standard representation of the data and some additional side-information. Therefore, distance learning can also be seen as an alternative approach to feature selection, or to finding a strong data representation.

The problem of learning distance functions is also closely related to the problem of *feature weighting*. In this setting, each data object is represented using a set of features (or a feature vector), and the objective is to learn a set of weights over these input features. The distances between a pair of objects are defined as the weighted sum of their corresponding feature vectors. In most cases, feature weighting methods are used in the context of K-Nearest-Neighbor (KNN) classifiers (see (Wettschereck *et al.*, 1997) for a review). Feature weighting can be seen as a generalization of feature selection. More importantly, it can be seen as a special case of distance function learning, in which a diagonal Mahalanobis metric is learnt. Therefore recent works in this area are sometimes described as "feature selection" algorithms (e.g. the *Simba* algorithm suggested by Gilad-Bachrach *et al.* (2004)), and sometimes described as "distance-learning" algorithms (see for example (Schultz and Joachim, 2003; Xing *et al.*, 2002)).

1.2.8 The relation to multiclass Learning

In *Multiclass learning* (or classification) we are required to learn a function over an input space $\mathcal{X}$ which takes a discrete set of values $\mathcal{Y} = \{0, 1, 2, \ldots M-1\}$ where M is the number of classes in our data. At first

glance, multiclass learning appears to be a straightforward generalization of binary classification (in which our function can only take two values $\{0, 1\}$. However, while there has been a considerable amount of theoretical and algorithmic work done on binary classification, the problem of multiclass learning is less well understood. Most available algorithms are usually tailored for binary classification, for example SVM's (Vapnik, 1998) and Boosting (Schapire *et al.*, 1997). Several recipes have been suggested which make it possible to combine binary classifiers in order to produce a multiclass classifier, such as a 'winner-takes-all' approach (Dietterich and Bakiri, 1995) and the use of error correcting output codes (Dietterich and Bakiri, 1995; Har-Peled *et al.*, 2002). Other approaches have been based on generative models, e.g. a Gaussian mixture model, which strongly depend on the assumption that the data distribution is known in advance.

The multiclass classification problem is closely related to the problem of *data partitioning* in which we attempt to learn a partitioning of the data into M discrete sets. In fact data partitioning is simply unsupervised multiclass learning. This observation can provide an alternative approach to multiclass classification, based on defining an equivalent binary classification problem which turns out to be a special form of distance function. Specifically if the original problem is to learn a multiclass classifier over some input data space $\mathcal{X}$, we can pose a binary classification problem over the product space $\mathcal{X} \times \mathcal{X}$ in which each pair is assigned a value of 1 if the two points originate from the same class, and a value of 0 if they originate from different classes. This binary function is in fact a *binary distance function* - i.e. it is a distance function which only assigns a value of 0 or 1 for every pair of points.

Intuitively, if we can learn an *ideal* distance function which assigns a value of 1 to every pair of points that originate from the same class, and 0 to every pair of points that originate from different classes, this function also provides an optimal multi-class classifier of our data. However, the question still remains as to how these problems are related when a non-optimal solution is obtained.

Bar-hillel and Weinshall (2003) provided a formal analysis of the relation between these two concepts. Specifically, they analyzed the relation between the family of binary distance functions (i.e. distance functions whose output is constrained to be 0 or 1) and multi-class classification. They showed that the solutions to these two problems are related, in the sense that finding a good solution to the binary distance learning problem entails the existence of a good solution to the multi-class problem, and vice-versa. More formally they showed that for any solution of the binary product space with error e_{pr} there exists a solution of the multiclass problem with error e_o

such that

$$\frac{e_{pr}}{e_o} < e_o < \sqrt{2Me_{pr}} \tag{1.19}$$

They further showed that under mild assumptions, a stronger version of the right side inequality exists, which shows that these two types of errors are linearly related to one another:

$$e_o < \frac{e_{pr}}{K} \tag{1.20}$$

where K is the frequency of the smallest class.

They also showed that the sample complexity of these two problems is similar, by comparing the VC-dimension S_{VC} of the binary distance learning problem to the Natarajan-dimension S_N of the multi-class problem. More specifically they showed that

$$\frac{S_N}{f_1(M)} - 1 \leq S_{VC} \leq f_2(M)S_N \tag{1.21}$$

where $f_1(M)$ is $O(M^2)$ and $f_2(M)$ is $O(logM)$.

They then concluded by suggesting an algorithm for learning a multi-class classifier by solving the equivalent binary distance learning problem. The algorithm suggests a way in which a product-space hypothesis (or distance function) can be greedily used to define a partition of the data, which in turn is equivalent to a multiclass classifier.

The relation between distance function learning and multiclass classification shows that in essence, distance function learning can be viewed as another way of addressing the problem of multiclass classification as well as the problem of data clustering. However, in many cases, being able to pose the same question in different formulations can lead to very different solutions, and can sometimes help to identify intuitive ways of solving a problem which in its original formulation was harder to approach. Distance function learning can certainly be seen as such a case, and as some of the works presented in this thesis show, it can be used to obtain significant performance improvement in various machine learning applications which have been extensively studied.

1.3 Semi-Supervised Learning

Semi-supervised learning is a topic which has attracted great interest in recent years (see (Zhu, 2005) for a detailed review). In this scenario, we are provided with an unlabeled dataset $\{x_i\}_{i=1}^N$ and with some additional side- information. The two most common types of side-information considered in the literature are partial-labels, and equivalence constraints. In the following sections we will provide a detailed account of these two approaches

and describe some related work which has taken place in these two sub-areas. Before doing so, let us focus on why *semi-supervised learning* is a promising field of research.

As noted above, the main motivation for *semi-supervised learning* lies in the fact that labelling data may be both costly and time-consuming, while obtaining unlabeled data is usually cheap and fast. The main underlying assumption is that since *unsupervised learning* algorithms can only help discover the underlying inherent properties of the data, augmenting the unlabeled data with some form of side-information can enable the use of more powerful *supervised learning* algorithms, and may therefore hopefully lead to improved performance. However, Cozman *et al.* (2003) have shown that this is not always the case, and that if wrong assumptions are made regarding the model used to describe the data, adding unlabeled data can actually degrade the performance of a classifier, when compared to its performance when trained using only labeled data [2]. Several different families of *semi-supervised learning* algorithms have been described in the literature and have been successfully used in various application domains. These are briefly described below.

1.3.1 Semi-supervised learning using partial labels

Semi-supervised learning using partial labels is the scenario in which the additional side -information provided to the algorithm apart from a set of N unlabeled datapoints $X_U = \{x_1 \cdots x_N\}$ is an additional (usually) small number L of input datapoints for which labels are provided $X_L = \{x_{N+1} \cdots x_{N+l}\}$. This scenario is well suited for applications in which obtaining unlabeled data is cheap, while labelling each data point may be expensive and time-consuming. Examples of such cases are the problem of predicting the three-dimensional structure of proteins. Despite the fact that roughly a million proteins have been sequenced, we only know the structure of about $30,000$ of them.

A closely related scenario is *transductive learning* , in which we augment a set of labeled training data with a set of unlabeled points. It is clear that the only real difference between *transductive learning* and *semi-supervised learning* then becomes the amount of labeled data which is provided to the algorithm. An additional (and sometimes not merely technical) difference between the two is that transductive learning algorithms usually require that the test data be provided at training time, a requirement which cannot always be satisfied.

Algorithms which have been developed in this area are usually adaptations of well-known supervised learning algorithms, which have been augmented to make use of the set of unlabeled points X_U. Here are several examples

[2] Note that this comparison is valid only for semi-supervised algorithms which make use of partial labels, and not necessarily those which use equivalence constraints.

of such algorithms:

1. *Semi-Supervised EM* - Both Miller and Uyar (1997) and Nigam *et al.* (1998) have suggested enhancements of the EM algorithm for a Gaussian Mixture Model (GMM) which incorporates labelled data. The algorithm uses the labeled data by defining an objective function which has the standard unsupervised log-likelihood term for the unlabeled datapoints and an additional term for the labeled data which directly represents posterior probabilities. A different formulation in which the labeled data are used to seed the clusters used as initial conditions was suggested by Basu *et al.* (2002).

2. *Semi-supervised K-means* - Demiriz *et al.* (1999) suggested an augmentation of the K-means algorithm which is based on defining an objective function that combines cluster purity and cluster compactness, where the purity term is evaluated using the set of labeled points.

3. *Semi-Supervised EM of a Hidden Markov Random Field* - Lange *et al.* (2005) proposed an approach that incorporates labeled and unlabeled data within an HMRF-like model, while a mean field approximation method for posterior inference is used in the E-step of the algorithm. A similar formulation was suggested by Basu *et al.* (2004).

4. *Semi-supervised Graph-Cuts* - Boykov *et al.* (1999) and Blum and Chawla (2001) suggested an augmentation of graph-cut algorithms which minimizes the cost of a cut in the graph using a set of labeled points. An extension of the normalized-cut (Shi and Malik, 2000) was suggested both by Yu and Shi (2001) and Joachims (2003). Another related formulation was suggested in (Zhu *et al.*, 2003; Zhou *et al.*, 2003) in which a quadratic cost function is minimized. An extension of the typical-cut algorithm (Gdalyahu *et al.*, 2001) was suggested by Getz *et al.* (2005). In this work instead of searching for a *single* cut which may not be robust to noise, a number of cuts are considered.

5. *Partially labeled random walks* - Szummer and Jaakkola (2002) suggests how a small number of labeled points can be used to augment a set of unlabeled points in a *Markov random walk* over a weighted graph which is constructed using a given distance metric D. Given a partially labeled dataset $X = \{X_L \cup X_U\}$, the conditional probabilities that the random process started from point x_i given that it ended in point x_j after t time steps is used to define an N-dimensional representation of each datapoint, where N is the number of datapoints. This representation is then used to estimate the posterior probability of a label for each datapoint.

The labeled points are used to learn the parameters of the prior distribution over the class labels for each given point which is required for estimating the posterior probabilities.

1.3.2 Semi-supervised learning using equivalence constraints

In this second sub-area of *semi-supervised learning* the side-information provided to the algorithm is a set of equivalence constraints. Equivalence constraints are relations between pairs of data points that indicate whether the points belong to the same category or not. We term a constraint 'positive' when the points are known to be from the same class, and 'negative' otherwise.

Equivalence constraints carry *less* information than explicit labels on the original datapoints. This can be seen by observing that a set of labeled points can be easily used to extract a set of equivalence constraints: any pair of points that belong to the same label form a positive constraint, while any pair of points that belong to different labels form a negative equivalence constraint. Note however that the opposite is not true - equivalence constraints cannot usually be transformed into labels, since this requires that the entire set of pairwise constraints be provided, a requirement which is usually far from being fulfilled.

1.3.2.1 Obtaining Equivalence Constraints

In contrast to explicit labels that are usually provided by a human instructor, in some scenarios, equivalence constraints may be extracted with minimal effort or even automatically. Two examples of such scenarios are described below:

1. **Temporal continuity** - In this scenario, we consider cases where the data are inherently sequential and can be modeled by a Markovian process. In these cases we can automatically obtain positive equivalence constraints by considering a set of samples which are temporally close to one another. In some cases, we can also use this scenario to obtain negative constraints. Examples are:

   (a) *Movie segmentation*: The objective of a movie segmentation task is to find all the frames in which the same actor appears (Boreczky and Rowe, 1996). Due to the continuous nature of most movies, faces extracted from successive frames in roughly the same location can be assumed to come from the same person, and thus provide a set of positive equivalence constraints[3]. Yan *et al.* (2004) have presented an interesting application of video object classification using this approach.

[3]This is true as long as there is no scene change, which can be robustly detected (Boreczky and Rowe, 1996)

(b) *Image Surveillance*: In surveillance applications (Shental *et al.*, 2002) we automatically obtain small sequences of images that are known to contain the same intruder, and therefore define a set of positive equivalence constraints. Alternatively, when two people simultaneously walk in front of two *distinct* cameras, the corresponding images are known to contain different people, thus providing negative equivalence constraints.

(c) *Speaker segmentation and Recognition*: In the task of speaker segmentation and recognition, the conversation between several speakers needs to be segmented and clustered according to speaker identity. Here, it may be possible to automatically identify small segments of speech which are likely to contain data points from a single but*unknown* speaker.

2. **Generalized Relevance Feedback** - Anonymous users of a retrieval system can be asked to help annotate the data by providing information about small portions of the data that they see [4]. We can use these user annotations to define equivalence constraints. Examples are:

(a) *Image Retrieval*: We can ask the users of an image retrieval engine to annotate the set of images retrieved as an answer to their query (Bar-Hillel *et al.*, 2005b). Thus, each of these cooperative users will provide a collection of small sets of images which belong to the same category. Moreover, different sets provided by the same user are known to belong to different categories. Note however that we cannot use the explicit labels provided by the different users because we cannot assume that the subjective labels of each user are consistent with one another: A certain user may label a set of images as "F-16" images, and another (less 'wanna be pilot') user may label another set of F-16 images as "Airplane" images.

(b) *Facial image recognition*: Suppose we are provided with a large database of facial images, which we would like to use to train a facial recognition engine. Due to its vast size, the database cannot be labeled by a small number of human teachers. In order to obtain some form of partial information in a relatively short amount of time, we can take the following approach: We arbitrarily divide the database into P parts (where P is very large), which are then given to P teachers to annotate. The labels provided by the different teachers may be inconsistent: because images of the same person appear in more than one part of the database, they are likely to be given different names. Coordinating the labels

[4] This scenario is also known as the 'distributed learning scenario'.

of the different teachers is almost as daunting as labelling the original data set. However, equivalence constraints can be easily extracted, since points which were given the same tag by a certain teacher are known to originate from the same class.

(c) *Text Classification (the "Yahoo!" problem)*: In this scenario we are provided with very large corpora of text documents (papers, newsgroup articles, web pages etc.) and asked to group them into classes, or into a hierarchy in which related documents will be grouped together. The purpose of creating such a taxonomy would be to allow these corpora to be browsed and accessed efficiently. It may also be the case that there is no clear cut definition of how to create such a taxonomy, despite the fact that some general criteria are provided. Cohn *et al.* (2003) who were the first to address this scenario, suggested the following iterative solution:

   i. Cluster the documents using some unsupervised clustering algorithm.

   ii. Browse the resulting clusters and provide some user feedback on which clusters you like and which you don't like. This does not have to be done for all clusters. More specifically feedback can be in one of the following forms:

      - Identify a document which does not belong to the cluster in which it was placed.
      - Move a document from one cluster to another.
      - Identify a pair of documents which do/do not belong together.

(d) Recluster the documents after allowing the clustering algorithm to modify the distance metric, in a way which would satisfy the constraints provided by the user.

(e) Repeat this process until results are satisfactory.

The underlying motivation here (as noted in (Cohn *et al.*, 2003)) is that "it is easier to criticize than to construct" [5].

1.3.2.2 Learning from Equivalence Constraints

Recently, a growing number of papers have suggested learning algorithms which make use of side -information in the form of equivalence constraints. Both positive ('a is similar to b') and negative ('a is dissimilar from b')

[5] To the best of my knowledge this is the first paper which presented the semi-supervised learning scenario of using side-information in the form of equivalence constraints. Interestingly, this paper, which was clearly ahead of its time, was not accepted for publication and only appears as a tech report.

equivalence constraints were considered. Most of these works have focused on two specific learning scenarios:

1. **Semi-supervised clustering** - Several authors have suggested how to adapt classical clustering algorithms of various types in order to make use of additional side-information in the form of equivalence constraints. More specifically constraints were introduced into the following algorithms:

   (a) *Semi-supervised Clustering with User Feedback* - Cohn *et al.* (2003) were actually the first to suggest a semi-supervised technique which is trained using equivalence constraints. The suggested method is an EM algorithm in which the distances between datapoints are based on a weighted Jensen-Shannon divergence. Equivalence constraints are used to learn the weights using gradient descent.

   (b) *Constrained Complete-Linkage* - Klein *et al.* (2002) introduced equivalence constraints into the complete-linkage algorithm by a simple modification of the similarity matrix provided as input to the algorithm: The similarity between all pairs of points which are positively constrained is set to ∞, thus ensuring that they will be merged into a single cluster before any other pairs of points. Similarly, the similarity value between any pair of points which are negatively constrained is set to $-\infty$, which ensures that they will be merged only at the final merge steps of the algorithm.

   (c) *Constrained K-means (COP K-means, MPCK-means)* - Wagstaff *et al.* (2001) suggested a heuristic for incorporating both types of equivalence constraints into the K-means algorithm. Constraints are incorporated as follows: for each datapoint the algorithm attempts to assign the point to the closest cluster to it such that its assignment does not violate any of the constraints to which it belongs. If such a cluster cannot be found (i.e. assigning the point to any of the clusters will violate some pairwise constraint) the algorithm fails. It can be easily shown that only incorporating negative constraints may cause the algorithm to fail. Moreover, as the number of negative constraints increases, the probability of failure increases dramatically. A more principled approach was suggested by Bilenko *et al.* (2004) (see below).

   (d) *Constrained EM* - Shental *et al.* (2004b) introduced equivalence constraints into the Expectation-Maximization (EM) algorithm of a Gaussian Mixture-Model (GMM). The algorithm makes use of clear probabilistic semantics, and therefore introduces constraints in a principled way, which overcomes many of the drawbacks of previous heuristic approaches. Equivalence constraints are introduced by modifying the Expectation step of the EM algorithm: instead of considering (summing) over all of

the possible assignments of datapoints to clusters (partitions), the algorithm only considers partitions which do not violate the constraints. When positive constraints are introduced, this turns out to be a simple modification which leads to closed-form update rules. However, when negative constraints are introduced, the problem becomes computationally hard, and in most cases an approximation scheme is used. A more detailed description of the algorithm will be provided in Chapter 2, in the context of the *DistBoost* algorithm.

(e) *Spectral Clustering* - Kamvar *et al.* (2003) introduced pairwise constraints into spectral clustering by modifying the similarity matrix in a similar way to that suggested in (Klein *et al.*, 2002). This work is also closely related to the work of Yu and Shi (2001). An alternative formulation was presented by Bie *et al.* (2003b) who incorporated a separate label constraint matrix into the objective function of a spectral clustering algorithm such as the normalized-cut (Shi and Malik, 2000).

(f) *Correlation Clustering* - Motivated by the connection between spectral clustering and graph-cut algorithms, Bansal *et al.* (2002) have suggested a general graph-based algorithm which attempts to incorporate both positive and negative constraints. In this formulation a graph is formed in which datapoints are represented by vertices and the constraints correspond to edge labels between the vertices.

2. **Learning distance functions** - Since a distance function is a function defined over pairs of points, equivalence constraints are in fact the natural form of supervision in this context. This may be seen by observing that equivalence constraints are binary labels in the *product space* $\mathcal{X} \times \mathcal{X}$ - i.e. the space of all pairs of points. This can be done by labeling pairs of points which are negatively constrained by -1, and labeling points which are positively constrained by 1. Therefore it is not surprising that the most of the works done on learning distance functions have suggested algorithms which learn distance functions using equivalence constraints.

   Most of the work done on learning distance functions using equivalence constraints has focused on learning a Mahalanobis metrics. More recently several authors have suggested algorithms for learning non-linear distance functions. A detailed description of distance learning algorithms will be provided in Chapter 2.

Several authors have also presented hybrid approaches which combine semi-supervised clustering and distance function learning. The work of Cohn *et al.* (2003) was perhaps the first to take this approach. More recently, Bilenko *et al.* (2004) suggested the MPCK-Means algorithm, which is an adaptation of the K-means algorithm

that integrates a metric learning step in each clustering iteration. More specifically, constraints are utilized both for cluster initialization and for learning a cluster specific Mahalanobis weight matrix A_h. A more general framework was suggested by Basu *et al.* (2004) that presents and EM algorithm over a Hidden Markov Random Field (HMRF), which incorporates both positive and negative equivalence constraints. Using this formulation the algorithm can be used with various distortion measures, all of which are Bregman divergences.

1.3.2.3 Types of equivalence constraints

Equivalence constraints considered in this work are treated as *hard* constraints in the sense that it is assumed that there is no noise in the constraints provided. Recently Law *et al.* (2004, 2005) and Lu and Leen (2005) have suggested two different formalisms which introduce soft constraints into clustering algorithms. In these works, constraints are assumed to be probabilistic, where the probability of each constraint denotes our belief in its truth. It is important to note that these *soft* constraints cannot be naturally obtained in real-life scenarios, and are in general harder to incorporate.

An additional form of side-information which has been recently considered is *relative comparisons*. This form of side-information which is even weaker than equivalence constraints, consists of triplets of the form *'A is more similar to B than to C'*. Relative comparisons can be naturally extracted from labels, and in some cases also from a set of positive and negative equivalence constraints. This form of side-information has been used for distance learning mainly in various retrieval contexts (Athitsos *et al.*, 2004; Rosales and Fung, 2006; Schultz and Joachim, 2003), in which they are a natural form of supervision, since the main objective is to rank objects in the 'correct' order.

Chapter 2

Algorithms for Learning Distance Functions

As noted in the Introduction, for many years only canonical distance functions or hand-designed distance functions were used. Recently, a growing body of work has addressed the problem of learning distance functions. This somewhat new area of research has its roots in works on supervised learning of distance functions for nearest-neighbor classification (Short and Fukunaga, 1981; Hastie and Tibshirani, 1996).

There are several ways to categorize distance learning algorithms. In this thesis, I suggest roughly splitting the distance learning family into the following four sub-categories:

1. **Nearest-Neighbor distance learning algorithms** - These algorithms are designed to improve nearest-neighbor classification by learning the local neighborhood structure of the data. The learnt distance function is then used to retrieve nearest neighbors. These algorithms have been motivated by various application domains such as information retrieval and more specifically image retrieval and document retrieval.

2. **Mahalanobis metric learning algorithms** - Research on learning Mahalanobis metrics is by far the most advanced, and numerous algorithms have been suggested for this task. These metrics are rather simple and easy to interpret and can also in some cases be easily kernelized - i.e. they can be used over any valid kernel.

3. **Non-linear distance learning algorithms** - This category includes algorithms that go beyond Mahalanobis metric learning, in an attempt to model the non-linearity of the distances between input datapoints. Despite the fact that empirically these algorithms have shown performance which in most cases is superior to Mahalanobis distance learning algorithms, only a few non-linear distance learning algorithms currently exist.

4. **Kernel learning algorithms** - These algorithms aim at learning the kernel used to measure the similarity

between datapoints in kernel-based classifiers directly from the data, instead of simply using a standard off-the-shelf kernel such as an RBF kernel.

Algorithms for learning distance functions can also be characterized by the side-information that they use in order to learn the distance function: Algorithms for learning nearest-neighbor classifiers and kernels are usually trained using labeled data, while most Mahalanobis metric learning algorithms and non-linear learning algorithms are trained using equivalence constraints. In the following sections, we will review each of these families of distance learning algorithms in more detail.

2.1 Nearest-Neighbor Distance learning algorithms

K-Nearest-Neighbor (KNN) classifiers have been popularly and successfully used in various application domains. There are several properties that make KNN classifiers appealing. To begin with they are conceptually simple, and do not require any learning stage. Additionally, they can be used when very few examples are present (i.e. when provided with a small sample), and also have been shown to work well even for moderate values of K. But perhaps the most attractive property of these classifiers is that they only rely on the local neighborhood of each datapoint in order to classify it. This means that for a KNN classifier to perform well, only the distances between each point and its local neighborhood need be 'good'. Due to this 'locality' property, KNN classifiers can also be successfully used when the classes are non-convex, or even in cases where each class is represented using a set of distinct clusters.

Another set of appealing properties of KNN classifiers is the asymptotic theoretical guarantees that can be shown for these classifiers: it can be shown that for any separable metric space, [1] as the training set size grows to infinity, for every ε there is an n_ε (which is dependent on the specific sequence of training datapoints) such that for all $n > n_\varepsilon$ the distance between any point in the training sample and its nearest neighbor is less than ε. Additionally Cover and Hart (1967) showed that the generalization error of the one-nearest-neighbor classifier is bounded above by twice the error obtained by the optimal Bayes classifier.

Clearly since KNN classifiers rely on the distances between each point and its neighbors, improving the distance function used for selecting the nearest neighbors can significantly improve the classifier's performance. Short and Fukunaga (1981) were the first to consider distance learning in the context of KNN classification. They characterized the optimal metric for NN classification in terms of the local class densities and their gradients.

[1] a metric space $\mathcal{X}$ is *separable* if it has a countable dense subset.

They then provided an algorithm which learns a local distance function which can be estimated separately for each datapoint. Another influential method for adapting a local metric was proposed by Hastie and Tibshirani (1996) who suggested a local LDA algorithm. More recently, Domeniconi and Gunopulos (2001) suggested a local metric which is computed using an SVM, in which the maximum margin boundary found by the SVM is used to determine the most discriminant direction over the query's neighborhood. Vincent and Bengio (2002) suggested a method for computing distances from local-dependant hyperplanes. Several distance learning algorithms which are aimed at improving KNN classifiers that are based on learning a Mahalanobis metric have been put forward (Lowe, 1995; Grauman and Darrell, 2005; Globerson and Roweis, 2005; Weinberger *et al.*, 2006; Zhang *et al.*, 2003; Shalev-Shwartz *et al.*, 2004) and we therefore defer their description to Section 2.2.

2.2 Mahalanobis Metric Learning Algorithms

To date, most distance learning algorithms proposed in the literature suggest various formulations of learning a Mahalanobis metric. This is probably due to several important properties of Mahalanobis metrics - their positive semi-definiteness and their relation to linear transformations and feature weighting (as discussed in Sec. 1.2.7). Due to these properties, and the simplicity of this distance metric, a large and growing corpus of work in the last few years has addressed the problem of learning Mahalanobis metrics. As noted above, some of these algorithms use side-information in the form of labeled data, and others make use of equivalence constraints. In what follows we provide a brief review of the various techniques which have been suggested for learning Mahalanobis metrics.

Lowe (1995) was perhaps one of the first to suggest a diagonal Mahalanobis distance learning algorithm in the context of kernel learning. He defines the probability of a given label for each point by computing the normalized average distance of the first K-neighbors of that point, where the distance assigned to each neighbor is determined by a Gaussian kernel centered around the point. The distances used to compute the Gaussian probabilities are a diagonal Mahalanobis metric, i.e. they are parametrized by weight vector W which weights the different input dimensions, which is learned during training. Another somewhat more general formulation was presented by Aggarwal (2003) who suggests learning a weighted Minkowsky distance (or L_p norm), of which the Mahalanobis distance is a special case. A simple gradient descent algorithm is presented. Aggarwal (2003) also considers a parametric Cosine model.

Xing *et al.* (2002) were one of the first to suggest a method for learning a full Mahalanobis metric. Their method attempts to find a Mahalanobis metric in which the distances between pairs of positively constrained points is as

small as possible, and the distances between negatively constrained pairs are larger than some constant factor. They then suggest an iterative gradient ascent algorithm for optimizing their suggested criterion. However, in order to ensure positive semi-definiteness, their method requires projection of the Mahalanobis matrix into the PSD cone, in every iteration. All in all, the suggested algorithm's complexity is $O(d^6)$ where d is the dimensionality of the data[2]. Another disadvantage of their method is that it can only work on pairs of positively constrained points, and does not explicitly exploit the transitivity property of positive equivalence constraints.

Bilenko *et al.* (2004) suggested an algorithm which combines Mahalanobis metric learning with semi-supervised clustering. Their algorithm can learn a separate Mahalanobis metric for each of the clusters, which in effect provides a generalized version of the K-means algorithm, very similar to an EM algorithm for a Gaussian Mixture Model. Their suggested algorithm also varies the weight of each constraint with respect to the distance between the constrained points under the Mahalanobis metric. The rationale is that violating a positive constraint between distant points is worse than violating a positive constraint between points that are close to one another, since the former would require a more aggressive modification of the current distance metric.

Shalev-Shwartz *et al.* (2004) suggest a Pseudo Metric Online Learning algorithm (POLA) for learning a Mahalanobis metric, which makes use of pairs of positively and negatively constrained points. Unlike most other distance learning algorithms, POLA can be trained in an online fashion, where at each time step a pair of points is received and the algorithm predicts whether they are similar to each other or not, by measuring the distance between them using the current metric, and determining if it is smaller or larger than a threshold parameter, which is also learned. Following feedback, the algorithm updates the metric and the threshold parameter, in an attempt to correct classification errors. Similar to Xing *et al.* (2002), in order to ensure that the computed distance metric is PSD, a projection operation is required after each update step. Additionally since the PSD matrix that is learned is a linear combination of rank-one matrices defined by vectors in the span of the input instances, the paper also suggests a kernelized version of the algorithm. By showing that the PSD matrices learned are norm bounded, Shalev-Shwartz *et al.* (2004) also provide an online error bound for the algorithm.

Goldberger *et al.* (2004) suggest the Neighborhood Component Analysis Algorithm (NCA), which derives its motivation from Nearest-Neighbor classifiers. The algorithm directly maximizes a leave-one-out KNN score on the training set. The papers suggests a cost function which is based on maximizing the expected number of points which are correctly classified under stochastic (soft) nearest-neighbor assignments. The stochastic soft assign-

[2]As noted by Bie *et al.* (2003a) the paper attempts to address this issue by considering a gradient descent algorithm instead of the standard Newton algorithms, but this may sometimes lead to convergence problems, when the dimensionality of the data is high.

ments are defined using a softmax over the Mahalanobis distances between each pair of points. The algorithms can also be used for dimensionality reduction and low rank projections by restricting the dimensions of the learnt matrix. The objective function, which is non-convex, is optimized using gradient descent.

Globerson and Roweis (2005) present the Maximally Collapsing Metric Learning algorithm (MCML). The algorithm attempts to find a Mahalanobis metric in which each class would be mapped (or collapsed) into a single location in feature space, which would differ for each class. This is an ideal approximation of the equivalence relation the algorithm makes use of. This scenario is approximated using the same stochastic selection rule of the NCA algorithm (Goldberger *et al.*, 2004). However, unlike NCA, the optimization objective is convex and therefore has a single and unique solution. In order to find a metric which approximates the ideal metric, the algorithm tries to minimize the KL-divergence between the ideal bi-level distribution which maps all points from the same class into a single point, infinitely far from points in different classes and the distribution that is defined by the learned metric.

Weinberger *et al.* (2006) suggest the Large Margin Nearest Neighbor algorithm (LMNN), which learns a Mahalanobis metric based on large-margin intuitions. More specifically, the algorithm attempts to ensure that the K-nearest neighbors of each data point always belong to the same class, while examples from different classes are separated by a large margin. Unlike previous approaches that attempt to minimize the pairwise distances between all points within the same class, their method only focuses on the K-near neighbors of each data point. The problem is formulated as a semi-definite program, in which the objective function is similar to the classical SVM objective function and has two competing terms: the first term penalizes large distances between each input point and its neighbors, and the second term penalizes small distances between each input point and all other points that do not originate from the same label.

Zhang *et al.* (2003) propose a parametric distance learning algorithm which uses labeled data. The algorithm is based on defining a similarity measure over pairs of points in the input space, in which the within- class similarity is always greater than the between- class similarity. This similarity measure is then approximated by a regression model which embeds the original input points in a Euclidean low dimensional space. The regression parameters are estimated using the iterative majorization algorithm.

Bie *et al.* (2003a) - Suggest an approximation of Linear Discriminant Analysis (LDA) which makes use of positive equivalence constraints. They suggest using a parametrized version of the data label matrix, which explicitly realizes the equivalence constraints provided. They then derive a cost function which is equivalent to the LDA cost

function, but can be written in terms of this parametrization. They then suggest maximizing the expected value of this cost function with respect to the parameters in their parametrization. Note, however, that the RCA algorithm presented in Section 2.2.1 is also closely related to LDA, and more specifically in (Bar-Hillel *et al.*, 2005b) we suggest a simple derivation of LDA which can be learned solely using positive equivalence constraints.

2.2.1 The Relevant Component Analysis (RCA) Algorithm

The Relevant Component Analysis Algorithm (RCA) was one of the first algorithms for Mahalanobis metric learning that was suggested and analyzed. The original motivations that led to the development of the algorithm were from the field of computer vision. As noted in the Introduction, in several classical computer vision applications such as image retrieval and video surveillance, equivalence constraints can be obtained automatically or with a minimal amount of supervision (Shental *et al.*, 2002). In Chapter 3 (publication [A]), we present several theoretical justifications for the algorithm. We show that RCA is the closed form solution of several interesting optimization problems whose computation is no more complex than a single matrix inversion. We also provide a detailed analytical and empirical comparison between RCA and the Mahalanobis metric algorithm suggested by Xing *et al.* (2002). We now turn to a detailed description of the algorithm.

RCA is a method that seeks to identify and down-scale global unwanted variability within the data. The method changes the feature space used for data representation by a global linear transformation which assigns large weights to "relevant dimensions" and low weights to "irrelevant dimensions" (Tenenbaum and Freeman, 2000, see). These "relevant dimensions" are estimated using *chunklets*; that is, small subsets of points that are known to belong to the same although *unknown* class. A *chunklet* is formed by applying a transitive closure over a set of pairs of points which are positively constrained. For example, if points x_1 and x_2 are related by a positive constraint, and x_2 and x_3 are also related by a positive constraint, then a chunklet $\{x_1, x_2, x_3\}$ is formed. The algorithm is presented below as Algorithm 1.

The RCA transformation is intended to reduce clutter, so that in the new feature space, the inherent structure of the data can be more easily unravelled (see illustrations in Figure 2.1 (a)-(f)). This is obtained by estimating the within class covariance of the data $cov(X|Z)$ where X and Z describe the data points and their labels respectively. The estimation is based on positive equivalence constraints only, and does not use any explicit label information. In high dimensional data, the estimated matrix can be used for semi-supervised dimensionality reduction. After estimating the within class covariance matrix $cov(X|Z)$, the dataset is whitened with respect to

Algorithm 1 The RCA algorithm

Given a data set $X = \{x_i\}_{i=1}^{N}$ and n chunklets $C_j = \{x_{ji}\}_{i=1}^{n_j} \;\; j = 1 \ldots n, \;\;$ do

1. Compute the within chunklet covariance matrix (Figure 2.1 (d)).

$$\hat{C} = \frac{1}{N} \sum_{j=1}^{n} \sum_{i=1}^{n_j} (x_{ji} - m_j)(x_{ji} - m_j)^t \tag{2.1}$$

   where m_j denotes the mean of the j'th chunklet.

2. If needed, apply dimensionality reduction to the data using $\hat{C}$ as described in Algorithm 2 (see Section 2.2.1.2).

3. Compute the whitening transformation associated with $\hat{C}$: $W = \hat{C}^{-\frac{1}{2}}$ (Figure 2.1 (e)), and apply it to the data points: $X_{new} = WX$ (Figure 2.1 (f)), where X refers to the data points after dimensionality reduction when applicable. Alternatively, use the inverse of $\hat{C}$ in the Mahalanobis distance: $d(x_1, x_2) = (x_1 - x_2)^t \hat{C}^{-1} (x_1 - x_2)$.

---

the estimated within class covariance matrix. The whitening transformation W (in Step 3 of Algorithm 1) assigns lower weights to directions of large variability, since this variability is mainly due to within class changes and is therefore "irrelevant" to the task of classification.

Step 2 of the RCA algorithm applies dimensionality reduction to the data if needed. In high dimensional spaces dimensionality reduction is almost always essential for the success of the algorithm, because the whitening transformation essentially rescales the variability in all directions so as to equalize them. Consequently, dimensions with small total variability cause instability and, in the zero limit, singularity. Section 2.2.1.2 describes this issue in more detail.

2.2.1.1 Theoretical justification of the RCA algorithm

While the RCA algorithm can be intuitively understood as an algorithm which seeks to reduce the effect of unwanted variability within the data, it can also be theoretically justified from three different perspectives, as shown in (Bar-Hillel *et al.*, 2003) and in (Bar-Hillel *et al.*, 2005b). Let us now examine these theoretical justifications in more detail.

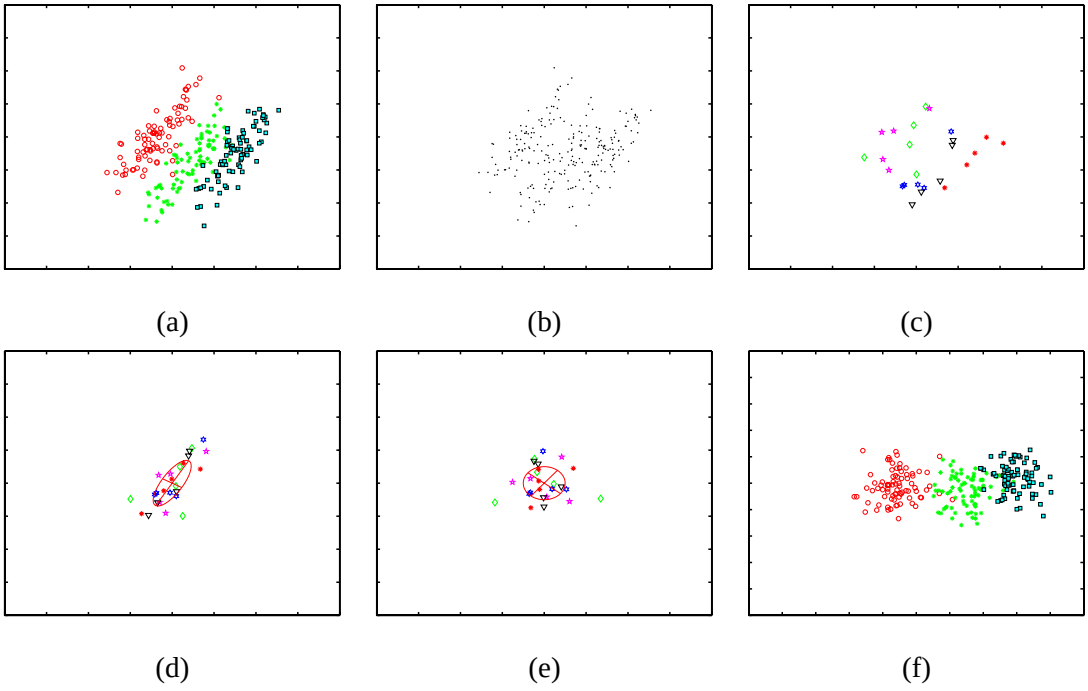

Figure 2.1. An illustrative example of the RCA algorithm applied to synthetic Gaussian data. (a) The fully labelled data set with 3 classes. (b) Same data unlabelled; clearly the class structure is less evident. (c) The set of chunklets that are provided to the RCA algorithm (points that share the same color and marker type form a chunklet). (d) The centered chunklets, and their empirical covariance. (e) The whitening transformation applied to the chunklets. (f) The original data after applying the RCA transformation.

RCA from an information theoretic perspective RCA can be shown to be derived from an information theoretic criterion. Following Linsker (1989), an information theoretic criterion states that an optimal transformation f of the input $\mathcal{X}$ into its new representation $f(\mathcal{X})$, should seek to maximize the mutual information $I(\mathcal{X}, f(\mathcal{X}))$ under suitable constraints. It can be shown that RCA is the solution to the following constrained optimization problem

$$\max_{f \in F} I(\mathcal{X}, f(\mathcal{X})) \quad s.t. \ \frac{1}{p} \sum_{j=1}^{p} \sum_{i=1}^{n_j} ||y_{ji} - \hat{m}_j^y||^2 \leq K$$

where F are invertible linear transformations, $\{y_{ji}\}_{j=1,i=1}^{p\ ,\ n_j}$ denote the set of points in p chunklets after the transformation, $\hat{m}_j^y$ denotes the mean of the points in chunklet j after the transformation, and K denotes some constant threshold.

Under Gaussian assumptions, the optimization problem stated above can be rewritten as:

$$\max_{A} log|A| \quad s.t. \ \frac{1}{N} \sum_{j=1}^{p} \sum_{i=1}^{n_j} ||y_{ji} - \hat{m}_j^y||_2^2 \leq K \tag{2.2}$$

As shown in Chapter 3, the solution to this problem is identical to the Mahalanobis matrix computed by RCA up to a global scale factor. When dimensionality reduction is also required, the solution of this problem becomes Fisher's Linear Discriminant (FLD) followed by the whitening of the chunklet covariance matrix in the reduced space (see Bar-Hillel *et al.* (2005b) Appendix A).

RCA and the minimization of inner chunklet distances RCA can also be shown to be a result of another constrained optimization problem, which tries to minimize the inner class distances.

$$\min_A \frac{1}{p} \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_{ji} - \hat{m}_j||_A^2 \quad s.t. \ |A| \geq 1 \tag{2.3}$$

This optimization problem can be interpreted as follows: we seek a Mahalanobis distance A, which minimizes the sum of all inner chunklet squared distances. The constraint $|A| \geq 1$ prevents the trivial solution of "shrinking" the entire space.

RCA and Maximum Likelihood It can also be shown that when the data consist of several normally distributed classes sharing the same covariance matrix, RCA can be interpreted as the maximum-likelihood (ML) estimator of the within-class covariance matrix. If we assume chunklets are sampled i.i.d. and that points within each chunklet are also sampled i.i.d., the likelihood of the chunklet distribution can be written as:

$$\prod_{j=1}^{n} \prod_{i=1}^{n_j} \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x_{ji}-m_j)^t \Sigma^{-1} (x_{ji}-m_j)\right) \tag{2.4}$$

If we take the log of Equation. 2.4, neglecting constant terms and denoting $A = \Sigma^{-1}$, we obtain:

$$\sum_{j=1}^{n} \sum_{i=1}^{n_j} ||x_{ji} - m_j||_A^2 - N \log |A| \tag{2.5}$$

where N is the total number of points in the chunklets. Maximizing the log-likelihood is equivalent to minimizing (2.5), whose minimum is obtained when A equals the RCA Mahalanobis matrix (2.1). Under these assumptions, we also provide a bound over the variance of this estimator, showing that it is at most twice the variance of the ML estimator obtained using labeled data.

2.2.1.2 Dimensionality reduction and the Constrained Fisher Linear Discriminant Algorithm (cFLD)

As noted above, RCA may include dimensionality reduction, which is some cases may be essential to its performance. We now address this issue in detail. We begin by presenting a detailed version of the dimensionality

reduction step of the RCA algorithm (Step 2), which is presented in Algorithm 2. We then provide an analysis which formally shows when dimensionality reduction is required by analyzing the within-class covariance matrix before and after applying RCA. Finally, we describe the constrained Fisher Linear Discriminant Algorithm (cFLD), which is also used in the dimensionality reduction step of the RCA algorithm.

Algorithm 2 Dimensionality reduction: Step 2 of RCA

Denote by D the original data dimensionality. Given a set of chunklets $\{C_j\}_{j=1}^{n}$ do

1. Compute the rank of the estimated within chunklet covariance matrix $R = \sum_{j=1}^{n}(|C_j| - 1)$, where $|C_j|$ denotes the size of the j'th chunklet.

2. If ($D > R$), apply PCA to reduce the data dimensionality to αR, where $0 < \alpha < 1$ (to ensure that cFLD provides stable results).

3. Compute the total covariance matrix estimate S_t, and estimate the within class covariance matrix using $S_w = \hat{C}$ from (2.1). Solve (2.7), and use the resulting A to achieve the target data dimensionality.

As shown in Chapter 3 the optimal dimensionality reduction often starts with Principal Component Analysis (PCA). PCA may appear contradictory to RCA, since it eliminates principal dimensions with small variability, while RCA emphasizes principal dimensions with small variability. One should note, however, that the principal dimensions are computed in different spaces. The dimensions eliminated by PCA have small variability in the original data space (corresponding to $Cov(X)$), while the dimensions emphasized by RCA have low variability in a space where each point is translated according to the centroid of its own chunklet (corresponding to $Cov(X|Z)$). As a result, the method ideally emphasizes those dimensions with large total variance, but small within -class variance.

Why is dimensionality reduction required? Step 3 of the RCA algorithm decreases the weight of principal directions along which the within- class covariance matrix is relatively high, and increases the weight of directions along which it is low. This intuition can be made precise in the following sense:

Denote by $\{\lambda^i\}_{i=1}^{D}$ the eigenvalues of the within- class covariance matrix, and consider the squared distance between two points from the same class $||x_1 - x_2||^2$. We can diagonalize the within-class covariance matrix using an orthonormal transformation which does not change the distance. Therefore, let us assume without loss of generality that the covariance matrix is diagonal.

Before whitening, the average squared distance is $E[||x_1 - x_2||^2] = 2\sum_{j=1}^{D} \lambda^j$ and the average squared distance in direction i is $E[(x_1^i - x_2^i)^2] = 2\lambda^i$. After whitening these values become $2D$ and 2, respectively. Let us define the weight of dimension i, $W(i) \in [0, 1]$, as

$$W(i) = \frac{E[(x_1^i - x_2^i)^2]}{E[||x_1 - x_2||^2]}$$

Now the ratio between the weight of each dimension before and after whitening is given by

$$\frac{W_{before}(i)}{W_{after}(i)} = \frac{\lambda^i}{\frac{1}{D}\sum_{j=1}^{D} \lambda^j} \tag{2.6}$$

In Equation (2.6) we observe that the weight of each principal dimension increases if its initial within-class variance was lower than the average, and vice versa. When there is high irrelevant noise along several dimensions, the algorithm will indeed scale down noise dimensions. However, when the irrelevant noise is scattered among many dimensions with low amplitude in each of them, whitening will amplify these noisy dimensions, which is potentially harmful. Therefore, when the data are initially embedded in a high dimensional space, the optional dimensionality reduction in RCA (Step 2) becomes mandatory.

The cFLD algorithm As stated above, and shown in Chapter 3, FLD is the dimensionality reduction technique which maximizes the mutual information under Gaussian assumptions, and is therefore part of the RCA algorithm when dimensionality reduction is desired. Traditionally FLD is computed from a fully labelled training data set, and the method therefore falls within supervised learning. We can extend FLD, using the same information theoretic criterion, to the case of partial supervision in the form of equivalence constraints. Specifically, denote by S_t and S_w the estimators of the total covariance and the within -class covariance respectively. FLD maximizes the following determinant ratio

$$\max_{A \in \mathcal{M}_{K \times D}} \frac{AS_tA^t}{AS_wA^t} \tag{2.7}$$

by solving a generalized eigenvector problem. The row vectors of the optimal matrix A are the first K eigenvectors of $S_w^{-1}S_t$. In our case the optimization problem is of the same form as in (2.7), with the within chunklet covariance matrix from (2.1) playing the role of S_w. We compute the projection matrix using SVD in the usual way, and term this FLD variant cFLD.

To understand the intuition behind cFLD, note that both PCA and cFLD remove dimensions with small total variance, and hence reduce the risk of RCA amplifying irrelevant dimensions with small variance. However,

unsupervised PCA may remove dimensions that are important for the discrimination between classes, if their total variability is low. Intuitively, better dimensionality reduction can be obtained by comparing the total covariance matrix (used by PCA) to the within- class covariance matrix (used by RCA), and this is exactly what the partially supervised cFLD tries to accomplish in (2.7).

The cFLD dimensionality reduction can only be used if the rank of the within chunklet covariance matrix is higher than the dimensionality of the initial data space. If this condition does not hold, we use PCA to reduce the original data dimensionality as needed.

2.2.1.3 Extensions of RCA

Recently several works have suggested various augmentations of the RCA algorithm. Specifically Wu *et al.* (2004) suggest the Self-enhanced Relevant Component Analysis Algorithm (SERCA). SERCA employs a boosting procedure in product-space, which makes use of both positive and negative equivalence constraints, and unlabeled data. The algorithm uses a boosting process similar to the one used by the *DistBoost* algorithm (Hertz *et al.*, 2004a) (See Section 2.3.1). As in (Hertz *et al.*, 2004a), the weak learner is a constrained Gaussian Mixture model (Shental *et al.*, 2004b), and boosting weights are also updated for the unlabeled points. The boosting process is then used to build new candidate sets for the positive constraints - that is, after the boosting process, a new set of positive constraints is obtained, and this set is then used to perform RCA in the original space, based on the augmented set of chunklets. The paper compares the performance of RCA with SERCA, and improvements are shown on two datasets from the UCI repository.

More recently, two authors have suggested a kernelized version of the RCA algorithm (Tsang *et al.*, 2005; Wolf, 2006). More specifically, Tsang *et al.* (2005) show how the chunklet covariance matrix can be computed using inner products between the data matrix and a binary matrix which encapsulates chunklet information. They then use the Woodbury formula to compute the inverse of this matrix. Their experimental comparisons, over a set of UCI datasets, the SCOP protein family dataset and on the USPS digit dataset show that in many cases kernel RCA outperforms RCA. Wolf (2006) proposes a different kernelized version, which is motivated by quantum mechanics.

2.3 Non-Linear Distance Function Learning Algorithms

As noted above, most of the research on distance learning has focused on learning a Mahalanobis metric. However, several papers have also suggested non-linear methods for distance learning, which are the focus of this section. Most of the research in this area was motivated by, or specifically designed for various image retrieval applications. There are two main reasons for this somewhat surprising connection. To begin with, unlike various other data domains in which the input data can be naturally represented by some predefined feature vectors, there is no such natural representation for images. Additionally, while in most application domains, the number of features which exist are usually small (< 100), images are usually represented using thousands of features. For these reasons, most of the work on learning distance functions in the context of image retrieval and classification has focused on non-metric distance functions, which are usually also highly non-linear. Another issue which also naturally arises in image applications is inter-class transfer, i.e. the ability to transfer knowledge between related tasks. One classic example is facial retrieval and verification, in which several works have considered inter-class transfer [3].

Phillips (1999) suggested a facial image representation formulated in *difference space*, which explicitly captures the dissimilarities between two facial images. An SVM binary classifier is trained to discriminate dissimilarities between images of the same individual vs. dissimilarities between images of different people. The decision boundary of the SVM is reinterpreted to produce a similarity metric between facial images. Each facial image is represented using PCA coefficients, after aligning all of the facial images in the dataset.

Mahamud and Hebert (2003b,a) presented a non-linear distance measure for object discrimination, which can be shown to be optimal under a nearest neighbor framework. The authors suggest a distance function which minimizes the mis-classification risk of the one-nearest-neighbor classifier, which is shown to be the probability that a pair of input points originate from different classes. This distance function is modeled using a linear logistic model that combines a set of elementary distance functions which operate in feature spaces such as color, texture and local shape properties. The distance function proposed does not satisfy the self-similarity property, but does satisfy the triangular inequality, and can also be shown to be optimal when compared to any metric distance measure (in the limit where the training set size grows to infinity), and also is tightly bounded from below by the

[3] Despite all of the above said, several Mahalanobis metric algorithms were also tested in visual recognition tasks. Among these are RCA, POLA and various others. However in these cases simple vectorial representations of the images were used. Fink *et al.* (2006) suggested the use of POLA for transferring knowledge between related classification tasks.

Bayes optimal classifier. Finding the optimal linear combination is a convex problem, and for purposes of speedup, it is optimized in a greedy stepwise manner. They present results on a database of everyday objects with varying backgrounds (Mahamud and Hebert, 2003b) and also on a face recognition task on the FERET dataset.

Athitsos *et al.* (2004) suggest an approach similar to the one presented by Mahamud and Hebert (2003b). More specifically, they present the BoostMap algorithm, which is a boosting process over the product space used to greedily combine a set of 1-dimensional embeddings into a multidimensional embedding. The boosting process is optimized using relative comparison triplets. The algorithm is designed to efficiently approximate a pre-defined similarity measure that is computationally intensive for purposes of speeding up retrieval performance.

In the context of face verification, Chopra *et al.* (2005) suggest a discriminative non-linear similarity learning algorithm, which is trained using equivalence constraints. Their method attempts to map the input patterns into a target space in which the L_1 norm approximates the "semantic" distance in the input space. The algorithms minimizes a discriminative loss function that penalizes small distances between negatively constrained pairs of points, and large distances between positively constrained pairs of points. Optimization is done using a convolutional network designed to be robust to geometric distortions. The metric is parametrized by pairs of identical convolutional neural nets. Their method can be applied on datasets where the number of categories is very large and not known in advance.

Chang and Yeung (2004, 2005b) proposed the Locally Linear Metric Adaptation algorithm (LLMA), which is a non-linear metric learning algorithm that is trained using positive equivalence constraints. LLMA attempts to transform the original datapoints into a new space in which similar points are closer to one another. However, in order to preserve the topological relationships between points, they apply the transformation not only to the similar point pairs, but also to other close points, in a varying manner which is dependent on their distance with respect to the constrained points. LLMA applies a linear transformation to each local neighborhood, but a different linear transformation is applied to different local neighborhoods, thus resulting in global non-linearity. A kernel based version of the algorithm was presented in (Chang and Yeung, 2005a). Results were presented on semi-supervised clustering (Chang and Yeung, 2004) and on image retrieval (Chang and Yeung, 2005b).

More recently Bar-Hillel and Weinshall (2006) presented the Gaussian Coding Similarity algorithm (GCS). The algorithm outputs a non-metric distance function which is learned from a set of positive equivalence constraints. The similarity of a pair of points is defined using information-theoretic principles. More specifically, it is defined as the gain in coding length which can be obtained when shifting from encoding each point independently to

jointly encoding the pair of points. Under simple Gaussian assumptions, the formulation provides a non-linear metric which is efficient and simple to learn. GCS can be viewed as a likelihood ratio test, and can be shown to be a variant of the FLD algorithm. Bar-Hillel and Weinshall (2006) also showed that under rather simple sampling assumptions of equivalence constraints, GCS converges to the RCA algorithm. The GCS method is a relatively simple and efficient technique, requiring only the estimation and inverse of two covariance matrices. It was used to improve graph-based clustering results on UCI datasets and the MNIST digit dataset and also for image retrieval on a facial image database and on a database of animal images.

2.3.1 The DistBoost Algorithm

The *DistBoost* algorithm is a distance function algorithm which can learn highly non-linear distance functions. The algorithm, originally presented in Hertz *et al.* (2004a,b), has been successfully applied in various application domains including image retrieval (Hertz *et al.*, 2004b), data clustering (Hertz *et al.*, 2004a), in computational immunology (Yanover and Hertz, 2005; Hertz and Yanover, 2006a,b) and also in the analysis of neuronal data (Weiner *et al.*, 2005). This wide variety of applications demonstrate not only that many different problems can be formulated as distance learning problems, but also that in many of these applications the distances between the data instances are highly non-linear and cannot be successfully modeled using the simpler linear model of a Mahalanobis metric. We now turn to a detailed description of the algorithm and its various components.

Recall that a distance function $\mathcal{D}$ is a function which maps every pair of points into some positive real number. The key observation that led to the development of the *DistBoost* algorithm is that we can learn such a distance function by posing a related binary classification problem over the product space $\mathcal{X} \times \mathcal{X}$, and solving it using margin-based classification techniques. The binary problem is the problem of distinguishing between pairs of points that belong to the same class and pairs of points that belong to different classes[4]. Moreover, note that equivalence constraints can be formally regarded as binary labels on points in $\mathcal{X} \times \mathcal{X}$: If we label pairs of points from the same class by 0 and pairs of points belonging to different classes by 1, we can interpret the classifier's margin as the required distance function.

Having reduced distance learning to binary classification with margins, we can now attempt to solve this problem using standard powerful margin-based classifiers. The *DistBoost* algorithm is a non-linear distance learning

[4]Note that this problem is closely related to the multi class classification problem: if we can correctly generate a binary partition of the data in product space, we implicitly define a multi-class classifier in the original vector space $\mathcal{X}$. The relations between the learnability of these two problems is discussed in Bar-hillel and Weinshall (2003) and in Section 1.2.8.

algorithm which is powered by a margin-based binary classifier. Before introducing the algorithm in detail, we briefly note that prior to suggesting this algorithm, we also explored various other adaptations of classical margin-based binary classifiers for learning distance functions. More specifically, we explored both support vector machines (SVM's) and boosting of decision trees (Hertz *et al.*, 2004a). These led us to realize that although the distance learning problem can be cast as a binary classification in the product space, it has some unique features which require special treatment:

1. The product space binary function we wish to learn has some unique structure which may lead to 'unnatural' partitions of the space between the labels. The concept we wish to learn is an indicator of an equivalence relation over the original space. Thus the properties of transitivity and symmetry of the relation place geometrical constraints on the binary hypothesis. If for example we represent a product space point as a concatenation of the two original space points $[x, y]$, then the function should be symmetric with respect to a 'hyper diagonal', i.e $d([x, y]) = d([y, x])$. The transitivity requirement leads to further non-intuitive constraints. Obviously, traditional families of hypotheses, such as linear separators or decision trees, are not limited to equivalence relation indicators, and it is not easy to enforce these constraints when such classifiers are used.

2. In the semi-supervised learning setting, we are provided with N datapoints in $\mathcal{X}$ and with a sparse set of equivalence constraints (or labels in product space) over some pairs of points in our data. We assume that the number of equivalence constraints provided is much smaller than the total number of equivalence constraints $O(N^2)$, and is of order $O(N)$. We therefore have access to large amounts of unlabeled data, and hence semi-supervised learning seems like an attractive option. However, classical binary classifiers like SVM and boosting methods are trained using labeled data alone.

These considerations led us to develop the *DistBoost* algorithm. *DistBoost* is a distance learning algorithm which attempts to address the issues discussed above. The algorithm learns a distance function using a well known machine learning technique called Boosting (Schapire *et al.*, 1997; Schapire and Singer, 1999). In Boosting, a set of "weak" learners are iteratively trained and then linearly combined to produce a "strong" learner. Specifically, DistBoost's weak learner is based on the constrained Expectation Maximization (cEM) algorithm suggested by Shental *et al.* (2004a). The cEM algorithm is used to generate a "weak" distance function. The final ("strong") distance function is a weighted sum of a set of such "weak" distance functions. The algorithm is presented in

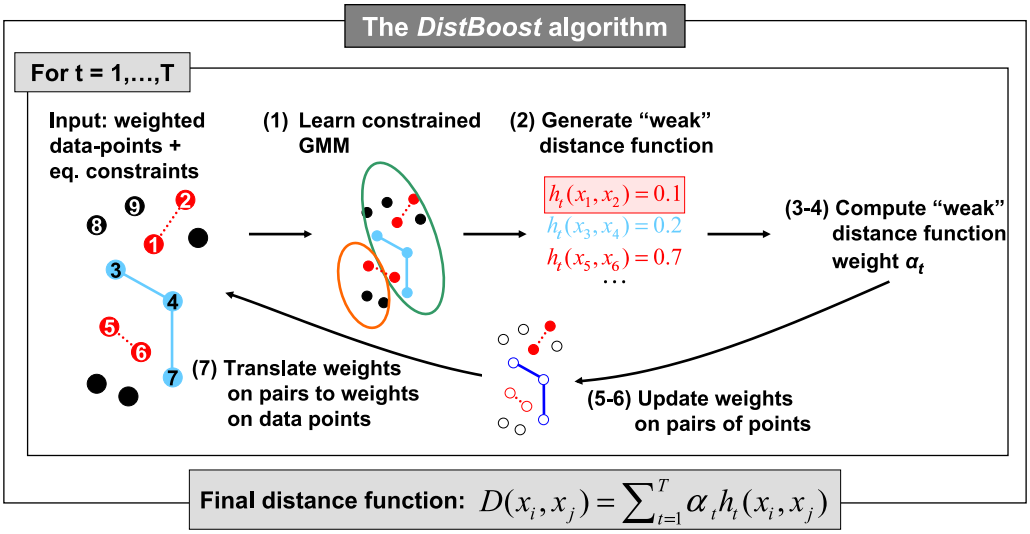

Figure 2.2. An illustration of the *DistBoost* algorithm. At each boosting round t the weak learner is trained using weighted input points and some equivalence constraints. In the example above, points $1,2$ and $5,6$ are negatively constrained (belong to different classes) and points $3,4$ and $4,7$ are positively constrained (belong to the same class). All other pairs of points (e.g. $8,9$ and $1,4$) are unconstrained. The constrained EM algorithm is used to learn a GMM (step (1)). This GMM is then used to generate a "weak" distance function (step (2)) that assigns a value in $[0,1]$ to each pair of points. The distance function is assigned a hypothesis weight (steps (3-4)) which corresponds to its success in satisfying the current weighted constraints. The weights of the equivalence constraints are updated (steps (5-6)) – increasing the weights of constraints that were unsatisfied by the current weak learner. Finally, the weights on pairs are translated into weights on data points (step (7)). In the example above, the distance between the negatively constrained points $1,2$ is small (0.1) and therefore the weight of this constraint is enhanced.

Alg. 3 and illustrated in Fig 2.2. In order to make use of unlabeled data points, *DistBoost*'s weak learner is trained in the original space, and is then used to generate a "weak distance function" on the product space.

The *DistBoost* algorithm builds distance functions based on the weighted majority vote of a set of original space soft partitions. The weak learner's task in this framework is to find plausible partitions of the space that comply with the given equivalence constraints. In this task, the unlabeled data can be of considerable help, as they can be used to define a prior on putative 'plausible partitions'. In order to incorporate the unlabeled data into the boosting process, we augmented the 'Adaboost with confidence intervals' algorithm presented in (Schapire and Singer, 1999). The details of this augmentation are presented in Section 2.3.1.1. The details of the weak learner we use are presented in Section 2.3.1.2.

Algorithm 3 The *DistBoost* algorithm.

Input:

Data points: $(x_1, ..., x_n)$, $x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points:$(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)$ $i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)

  $w_k = 1/n$ $\quad k = 1, \ldots, n$ (weights over data points)

- For $t = 1, .., T$

  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak hypothesis $\tilde{h}_t : \mathcal{X} \times \mathcal{X} \to [-\infty, \infty]$ and define a weak distance function as $h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h}_t(x_i, x_j)\right) \in [0, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i \tilde{h}_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2}\ln(\frac{1+r_t}{1-r_t})$
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:

$$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{h}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\alpha_t) & y_i = * \end{cases}$$

  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final distance function $\mathcal{D}(x_i, x_j) = \sum_{t=1}^{T} \alpha_t h_t(x_i, x_j)$

2.3.1.1 Semi supervised boosting in product space

Our boosting scheme is an extension of the Adaboost algorithm with confidence intervals (Schapire and Singer, 1999) to handle unsupervised data points. As in Adaboost, we use the boosting process to maximize the margins of the labeled points. The unlabeled points only provide a decaying density prior for the weak learner. Given a partially labeled dataset $\{(x_i, y_i)\}_{i=1}^{N}$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis $f(x) = \sum_{i=1}^{k} \alpha_k h(x)$ which minimizes the following loss function:

$$\sum_{\{i|y_i=1,-1\}} \exp(-y_i h(x_i, x_j)) \tag{2.8}$$

Note that this semi-supervised boosting scheme computes the weighted loss only on **labeled** pairs of points but updates the weights over **all** pairs of points. The unlabeled points serve as a prior on the data's density, which effectively constrains the parameter space of the weak learner in the first boosting rounds, giving priority to hypotheses which both comply with the pairwise constraints and with the data's density. In order to allow the algorithm to focus on the labeled points as the boosting process advances, the weights of the unlabeled points decay at a rate which is controlled by a tradeoff parameter λ and by the weight of each boosting round α_t (see Algorithm 3 step 5).

In the product space there are $O(N^2)$ unlabeled points, which correspond to all the possible pairs of original points, and the number of weights is therefore $O(N^2)$. However, the update rules for the weight of each unlabeled point are identical, and so all the unlabeled points can share the same weight. Hence the number of updates effectively required in each round is proportional to the number of labeled pairs alone. If $\lambda \geq 1$, the weight of the unlabeled pairs is guaranteed to decay at least as fast as the weight of any labeled pair.

Several algorithms which incorporate unlabeled data into the boosting process have been suggested (d'Alche Buc *et al.*, 2002; Grandvalet *et al.*, 2001). In these algorithms, the incorporation of unlabeled points is achieved by extending the 'margin' concept to the unlabeled points. Several margin extensions were suggested, relating the margin of a hypothesis over an unlabeled point to the certainty of the hypothesis regarding the point's classification. The extended margins are then incorporated into the *MarginBoost* algorithm (Mason *et al.*, 2000). Specifically, given a partially labeled dataset $\{(x_i, y_i)\}_{i=1}^{N}$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis $f(x) = \sum_{i=1}^{k} \alpha_k h(x)$ minimizing:

$$\sum_{i=1}^{N} \exp(-\rho_f(x_i, y_i)) = \sum_{\{i|y_i=1,-1\}} \exp(-\rho_f(x_i, y_i)) + \sum_{\{i|y_i=*\}} \exp(-\rho_f(x_i))$$

where $\rho_f(x,y) = yf(x)$ for a labeled point and $\rho_f(x) = |f(x)|$ or $\rho_f(x) = f(x)^2$ for an unlabeled point. The minimization argument hence contains a mixture of the supervised loss (which measures the agreement between the combined hypothesis and the labels) and the unsupervised loss (which measures the certainty of the hypothesis over the unsupervised data).

Minimizing traditional supervised loss is an intuitive goal, which has well known justifications in terms of generalization error (Schapire *et al.*, 1997). In contrast, minimizing the unsupervised loss is not clearly a desired goal, as a hypothesis can be very certain about the classification of unlabeled points even when it classifies them incorrectly. This problem becomes more acute when the number of unsupervised points is much larger than the number of supervised points, as is the case in our application. We have empirically tested some variants of these algorithms and found that minimizing these scores tends to lead to poor generalization performance in our context.

2.3.1.2 DistBoost's weak learner - the constrained EM algorithm (cEM)

The weak learner in *DistBoost* is based on the constrained EM algorithm presented by Shental *et al.* (2004a). This algorithm learns a mixture of Gaussians over the original data space, using unlabeled data and a set of positive and negative constraints. Below we briefly review the basic algorithm, and then show how it can be modified to incorporate weights on sample data points. We also describe how to translate the boosting weights from product space points to original data points, and how to extract a product space hypothesis from the soft partition found by the EM algorithm.

A Gaussian mixture model (GMM) is a parametric statistical model which assumes that the data originate from a weighted sum of several Gaussian sources. More formally, a GMM is given by $p(x|\Theta) = \Sigma_{l=1}^{M} \alpha_l p(x|\theta_l)$, where α_l denotes the weight of each Gaussian, θ_l its respective parameters, and M denotes the number of Gaussian sources in the GMM. EM is a widely used method for estimating the parameter set of the model (Θ) using unlabeled data (Dempster *et al.*, 1977). In the constrained EM algorithm *equivalence constraints* are introduced into the 'E' (Expectation) step, such that the expectation is taken only over assignments which comply with the given constraints (instead of summing over *all* possible assignments of data points to sources).

Assume we are given a set of unlabeled i.i.d. sampled points $X = \{x_i\}_{i=1}^{N}$, and a set of pairwise constraints over these points Ω. Denote the index pairs of positively constrained points by $\{(p_j^1, p_j^2)\}_{j=1}^{N_p}$ and the index pairs of negatively constrained points by $\{(n_k^1, n_k^2)\}_{k=1}^{N_n}$. The GMM model contains a set of discrete hidden variables H, where the Gaussian source of point x_i is determined by the hidden variable h_i. The constrained EM algorithm

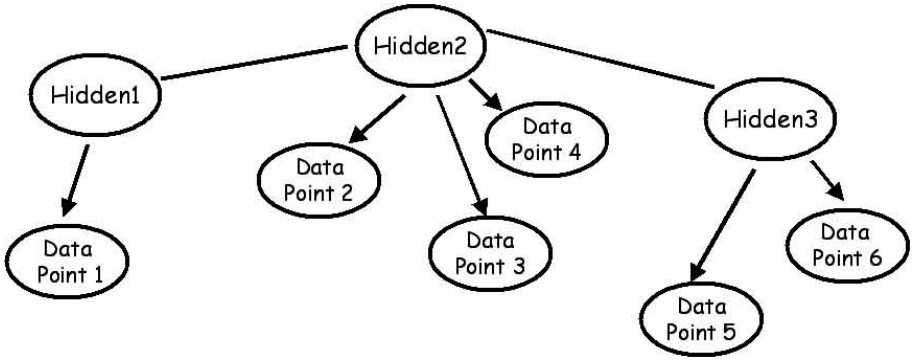

Figure 2.3. A Markov network representation of the constrained mixture setting. Each observable data node has a discrete hidden node as its ancestor. Positively constrained nodes have the same hidden node as their ancestor. Negative constraints are expressed using edges between the hidden nodes of negatively constrained points.Here points 2,3,4 are constrained to be together, and point 1 is constrained to be from a different class.

assumes the following joint distribution of the observables X and the hiddens H:

$$p(X, H|\Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^{n} \alpha_{h_i} p(x_i|\theta_{h_i}) \prod_{j=1}^{N_p} \delta_{h_{p_j^1} h_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{h_{n_k^1} h_{n_k^2}}) \tag{2.9}$$

The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (2.9) with respect to H.

The equivalence constraints create complex dependencies between the hidden variables of different data points. However, the joint distribution can be expressed using a Markov network, as seen in Figure 2.3. In the 'E' step of the algorithm the probabilities $p(h_i|X, \Theta, \Omega)$ are computed by applying a standard inference algorithm to the network. Such inference is feasible if the number of negative constraints is $O(N)$, and the network is sparsely connected. The model parameters are then updated based on the computed probabilities. The update of the Gaussian parameters $\{\theta_l\}$ can be done in closed form, using rules similar to the standard EM update rules. The update of the cluster weights $\{\alpha_l\}_{l=1}^{M}$ is more complicated, since these parameters appear in the normalization constant Z in (2.9), and the solution is found with a gradient descent procedure. The algorithm finds a local maximum of the likelihood, but the partition found is not guaranteed to satisfy any specific constraint. However, since the boosting procedure increases the weights of points which belong to unsatisfied equivalence constraints, it is highly likely that any constraint will be satisfied in one or more partitions.

Incorporating weights into the cEM algorithm We incorporated weights into the constrained EM algorithm according to the following semantics. The algorithm is presented with a virtual sample of size N_v. A training point x_i with weight w_i appears $w_i N_v$ times in this sample. All the repeated tokens of the same point are considered to

be positively constrained, and are therefore assigned to the same source in every evaluation in the 'E' step. In all of our experiments we have set N_v to be the actual sample size.

When introduced this way, the incorporation of weights has some non- trivial consequences. The posterior distribution of the hidden variables, which is computed at the E-step, has a strong dependency on the absolute weight $w_i N_v$ of the point. High values ($w_i N_v \gg 1$) tend to sharpen the distribution, so the point is assigned to a single source with a probability close to 1. Low values ($w_i N_v \ll 1$) have the opposite effect of flattening the point's posterior. Because of these effects the parameter N_v, which controls the absolute size of the weights, has a major impact on the algorithm's behavior. Its role is similar to the role that $\frac{1}{T}$ (where T is the temperature), has in many statistical mechanics models. In all of the experiments with the algorithm (Hertz *et al.*, 2004a,b) we used the actual sample size as the value of this 'virtual sample size' parameter.

Translating pair weights in product-space into singleton weights in the original space While the weak learner accepts a distribution over the original space points, the boosting process described in 2.3.1.1 generates a distribution over the sample product space in each round. The product space distribution is converted to a distribution over the sample points by simple marginalization. Specifically, denote by w_{ij}^p the weight of pair (i, j); the weight w_i^s of point i is defined to be

$$w_i^s = \sum_j w_{ij}^p \tag{2.10}$$

Generating a weak distance function from a GMM The weak learners' task is to provide a weak distance function $h_t(x_i, x_j)$ over the product space . Let us denote by $MAP(x_i)$ the Maximum A-Posteriori assignment of point x_i and by $p^{MAP}(x_i)$ the MAP probability of this point:

$$p^{MAP}(x_i) = \max_p p(h_i = m | x_i, \Theta)$$

We partition the data into M groups using the MAP assignment of the points and define

$$\tilde{h_t}(x_i, x_j) \equiv \begin{cases} +p^{MAP}(x_i) \cdot p^{MAP}(x_j) & \text{if} \quad MAP(x_i) = MAP(x_j) \\ -p^{MAP}(x_i) \cdot p^{MAP}(x_j) & \text{if} \quad MAP(x_i) \neq MAP(x_j) \end{cases} \tag{2.11}$$

The weak distance function is given by

$$h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h_t}(x_i, x_j)\right) \in [0, 1] \tag{2.12}$$

It is easy to see that if the MAP assignment of two points is identical their distance will be in $[0, 0.5]$, and if their MAP assignment is different their distance will be in $[0.5, 1]$.

2.4 Kernel Learning algorithms

Kernel based classifiers have been successfully used in various application domains, and have now become part of the standard toolbox of the machine learning community. The popularity of these classifiers derives from the fact that by the use of a kernel, one can in effect bypass the problem of feature selection (or data representation) via the use of some kernel. Each kernel projects the data into some high (and possibly infinite) dimensional space, in which the data are hopefully well separated. However, by using the so called 'kernel trick', the high dimensional representation is only implicit, since only the dot products of the feature mappings are required for training the classifier. While many standard kernels have been successfully used in the literature, such as the *linear kernel*, the *polynomial kernel* and the *RBF kernel*, selecting the right kernel is usually application-dependent, and there is no principled way to select the kernel other than resorting to data driven techniques such as cross-validation. It is now widely recognized that the performance of any kernel-based classifier strongly depends on the kernel used. In fact, in some application domains such as computational biology, hand-designed kernels which incorporate various forms of domain knowledge have been widely used for various applications (Leslie *et al.*, 2002, 2003; Vert *et al.*, 2005).

Recently there has been a growing body of work on learning the kernel directly from the training data. Most of the work in this area focuses on learning the kernel matrix (also known as the Gram matrix) and can therefore only be applied in a transductive learning scenario. A first attempt to address this problem was suggested by Cristianini *et al.* (2001) who introduced the concept of *kernel alignment*, which intuitively measures the similarity between two given kernels. More formally, given two kernels K_1 and K_2 the kernel alignment score is given by

$$Alignment(K_1, K_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}}$$

where $\langle . \rangle_F$ denotes the Frobenius product. Cristianini *et al.* (2001) suggested how this score can be used to measure the alignment of a given kernel to a training set $S = (x_i, y_i)$ by measuring the alignment of a given kernel to the *ideal kernel* given by $K_{ideal} = YY'$:

$$Alignment(K, S) = \frac{\langle K, K_{ideal} \rangle_F}{\sqrt{\langle K, K \rangle_F \langle K_{ideal}, K_{ideal} \rangle_F}}$$

This score can then be used to choose a good kernel, by selecting the kernel which has the highest alignment with the given training data. Cristianini *et al.* (2001) then considered several algorithms which can learn the Gram matrix. Several other works have also considered learning the Gram matrix: Crammer *et al.* (2002) have

suggested a boosting based formulation of the problem. Lanckriet *et al.* (2002) have suggested an algorithm which is based on semi-definite programming, and Bousquet and Herrmann (2002) put forward a gradient descent approach over generalization bounds. Finally, Zhang *et al.* (2006) proposed a formulation based on generative modeling. Learning of kernel functions in the context of learning-to-learn is discussed in Yu *et al.* (2005).

Based on the idea of kernel alignment (Cristianini *et al.*, 2001), Kwok and Tsang (2003) suggested a kernel learning algorithm which attempts to learn the *ideal kernel* using equivalence constraints. The paper proposes a method for "idealizing" a given kernel K by adding the ideal kernel K^* to it $\tilde{K} = K + \gamma K^*$, where the parameter γ is optimized. They then show how this problem can be formulated as a distance learning problem in which a Mahalanobis metric is learned that can also be used to provide similarity values over unseen patterns (i.e. in an inductive setting). The metric learned requires that the distance between pairs of dissimilar points be greater than a certain margin, which can be shown to increase its alignment. This leads to a quadratic optimization which is similar to the ν-SVM algorithm, over pairs of points. An additional and similar formulation was suggested by Wu *et al.* (2005) which adapts an existing kernel matrix using a set of equivalence constraints. Features are weighted in the induced feature space by learning a Mahalanobis metric in the induced space.

Another avenue of research on kernel learning has focused on learning combination kernels, which are linear combinations of a set of predefined kernels where the weight of each kernel is learned. More specifically, Lanckriet *et al.* (2002) have suggested a semi-definite programming formulation of the problem, and Zien and Ong (2006) have presented another formulation in the context of biological sequence kernels.

Some recent works have considered the problems of learning the parameters of a given pre-defined kernel family that is not based on cross validation (as usually done). For example, Ong *et al.* (2005) suggest a *superkernel* approach, which relies on a statistical estimation of the kernel parameters. Chapelle *et al.* (2002) suggest a gradient descent approach over the estimates of the generalization error, which is estimated over hold out data, or uses a leave-one-out approach.

Another interesting kernel that has been suggested for image retrieval (or more generally for unordered feature sets) is the pyramid match kernel (Grauman and Darrell, 2005). The kernel maps unordered feature sets into multi-resolution histograms and then computes a weighted histogram intersection in this histogram feature space. The kernel can be shown to approximate the similarity measured by the optimal partial matching (correspondences between sets of unequal cardinality).

Despite the growing number of works on learning kernels, all of the works described above have focused on

learning the kernel matrix, rather than learning a kernel function which is defined for every pair of points in the product space. Unlike most other algorithms, the *KernelBoost* algorithm which will be presented in Sec. 2.4.1 learns kernel functions, and can therefore be used in an inductive learning scenario. As will be demonstrated in Chapter 5 the algorithm can be successfully applied in a "learning to learn" setting in which knowledge is transferred between related classification tasks, when using very limited amounts of training data. However, when considering the more standard learning scenario, when there are large amounts of training data, to date none of the suggested methods have been shown to provide significant performance boosts as compared to standard off-the-shelf kernels. Recently, Srebro and Ben-David (2006) have provided a theoretical analysis of the generalization error of a kernel-based classifier when using a learned kernel which is a linear combination of pre-defined kernels, or a convex combination of these. Specifically, they show that for a kernel family with a given *pseudodimension* d_ϕ the estimation error of a kernel-based classifier with margin γ is given by $\sqrt{\tilde{\mathcal{O}}(d_\phi + 1/\gamma^2)/n}$ where n denotes the sample size. Unlike previous bounds in which the relationship between the margin term and kernel-family term is multiplicative, this recent bound is additive. Srebro and Ben-David (2006) also compute the *pseudodimension* of several well known and widely used kernel families.

2.4.1 The KernelBoost Algorithm

The *KernelBoost* is a variant of the *DistBoost* algorithm, which learns distance functions that are Mercer kernels. While the *DistBoost* algorithm described in Section 2.3.1 has been shown to enhance clustering and retrieval performance, it has never been used in the context of classification, mainly due to the fact that the learnt distance function is not a kernel (and is not necessarily metric). Therefore it cannot be used by the large variety of kernel-based classifiers that have been shown to be highly successful in fully labeled classification scenarios. *KernelBoost* alleviates this problem by modifying the weak learner of *DistBoost* to produce a 'weak' kernel function. The 'weak' kernel has an intuitive probabilistic interpretation - the similarity between two points is defined by the probability that they both belong to the same Gaussian component within the GMM learned by the weak learner. An additional important advantage of *KernelBoost* over *DistBoost* is that it is not restricted to model each class at each round using a single Gaussian model, therefore removing the assumption that classes are convex. This restriction is dealt with by using an adaptive label dissolve mechanism, which splits the labeled points from each class into several local subsets, as described in Sec. 2.4.1.2. An important inherited feature of *KernelBoost* is that it is semi-supervised, and can naturally accommodate unlabeled data in the learning process. As our empirical

results show, the ability to use unlabeled data in the training process proves to be very important when learning from small samples. Additionally, as our experiments show (Hertz *et al.*, 2006), the algorithm can be trained with very small amounts of labeled data, and can be used in "learning to learn" scenarios.

Let us denote by $\{x_i\}_{i=1}^n$ the set of input data points which belong to some vector space $\mathcal{X}$, and by $\mathcal{X} \times \mathcal{X}$ the "product space" of all pairs of points in $\mathcal{X}$. An equivalence constraint is denoted by (x_{i_1}, x_{i_2}, y_i), where $y_i = 1$ if points (x_{i_1}, x_{i_2}) belong to the same class (positive constraint) and $y_i = -1$ if these points belong to different classes (negative constraint). $(x_{i_1}, x_{i_2}, *)$ denotes an unlabeled pair.

As in *DistBooost*, the algorithm makes use of the observation that equivalence constraints on points in $\mathcal{X}$ are binary labels in the product space, $\mathcal{X} \times \mathcal{X}$. Thus, by posing the problem in product space the problem is transformed into a classical binary classification problem, for which an optimal classifier should assign $+1$ to all pairs of points that come from the same class, and -1 to all pairs of points that come from different classes [5]. The weak learner itself is trained in the original space $\mathcal{X}$, which allows it to make use of unlabeled data points in a semi-supervised manner. The weak learner is then used to generate a "weak kernel function" on the product space.

The *KernelBoost* algorithm (described in Algorithm 4) learns a Kernel function of the following form:

$$K(x_1, x_2) = \sum_{t=1}^{T} \alpha_t K_t(x_1, x_2) \tag{2.13}$$

which is a linear combination of "weak kernel functions" K_t with coefficients α_t, which is optimized using the same semi-supervised Adaboost extension described in Section 2.3.1.1.

2.4.1.1 KernelBoost's weak learner

As in *DistBoost*, *KernelBoost*'s weak learner is based on the constrained Expectation Maximization (cEM) algorithm (Shental *et al.*, 2004a). The algorithm uses unlabeled data points and a set of equivalence constraints to find a Gaussian Mixture Model (GMM) that complies with these constraints. The difference between the two algorithms lies in the way in which the weak learner is used in order to generate a weak distance function. We therefore now describe this issue in more detail.

Generating a Weak Kernel from a GMM Given the mixture Θ^t at round t, we construct a 'weak kernel' which essentially measures the probability that two points belong to the same Gaussian component. Denoting the hidden

[5] Also referred to as the *ideal* kernel (Cristianini *et al.*, 2001).

Algorithm 4 The *KernelBoost* algorithm.

Input:

Data points: $(x_1, ..., x_n)$, $x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points:$(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)$ $i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)

  $w_k = 1/n \qquad k = 1, \ldots, n$ (weights over data points)

- For $t = 1, .., T$

  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak kernel function $K_t : \mathcal{X} \times \mathcal{X} \to [\prime, \infty]$ and define a weak hypothesis as $\tilde{K}_t(x_i, x_j) = 2K_t(x_i, x_j) - 1 \in [-1, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i \tilde{K}_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2} \ln(\frac{1+r_t}{1-r_t})$.
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:

  $$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{K}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\lambda * \alpha_t) & y_i = * \end{cases}$$

  where λ is a tradeoff parameter that determines the decay rate of the unlabeled points in the boosting process.

  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final Kernel function of the form $K(x_i, x_j) = \sum_{t=1}^{T} \alpha_t K_t(x_i, x_j)$.

label of a point according to the mixture by $l(x)$, the kernel is given by

$$K_t(x_1, x_2) = p[l(x_1) = l(x_2)|\Theta] = \sum_{j=1}^{M^t} p(l(x_1) = j|\Theta)p(l(x_2) = j|\Theta) \tag{2.14}$$

The combined 'strong' kernel therefore becomes:

$$K(x_1, x_2) = \sum_{t=1}^{T}\sum_{k=1}^{M^t} \sqrt{\alpha_t}p[l(x_1) = k|\Theta^t] \cdot \sqrt{\alpha_t}p[l(x_2) = k|\Theta^t] \tag{2.15}$$

If we think of each element in the sum in Equation (2.15) as a feature in a feature-space of dimension $\sum_{t=1}^{T} M^t$, then the coordinate corresponding to the pair (t, k) holds a feature of the form

$$\Phi_{t,k}(x) = \sqrt{\alpha_t}\frac{\pi_k G(x|\mu_k^t, \Sigma_k^t)}{\sum_{j=1}^{M^t} \pi_j G(x|\mu_j^t, \Sigma_j^t)} \tag{2.16}$$

These features can be interpreted as soft Voronoi cell indicators: a high value for feature $\Phi_{t,k}$ indicates that the point lies in cell k of the partition induced by mixture t. These features are rather different from the prototype-like RBF features. Specifically, their response does not necessarily decay rapidly with the distance from the Gaussian's center. Decay only happens in regions where other Gaussians from the same mixture are more likely.

2.4.1.2 The Label Dissolving Mechanism

The weak learner of the *KernelBoost* algorithm treats all constraints as hard constraints; in particular, since all positive constraints are always satisfied in the cEM algorithm, its only option is to attempt to place all of the points from the same label in a single Gaussian at every iteration. This is very problematic for non-convex classes generated by non-Gaussian distributions (see Figure 2.4, left plot). Therefore, in order to enrich the expressive power of *KernelBoost* and to allow it to model classes of these types, the algorithm is augmented by a label-dissolving mechanism, which relies on the boosting weights. This mechanism splits sets of points with the same label into several local subsets, which allows the algorithm to model each of these subsets separately, using a different Gaussian model.

The intuition leading to the proposed mechanism is the following. We would like to model each non-convex class, using several local Gaussians. The attempt to model a highly non-Gaussian, or non-convex class using a single Gaussian will fail, and cause some of the pairwise constraints to be unsatisfied. The boosting process focuses each new weak learner on those harder pairs still inconsistent with the current hypothesis. The adaptive dissolve mechanism uses these pairwise weights to eliminate edges already consistent with the current hypothesis

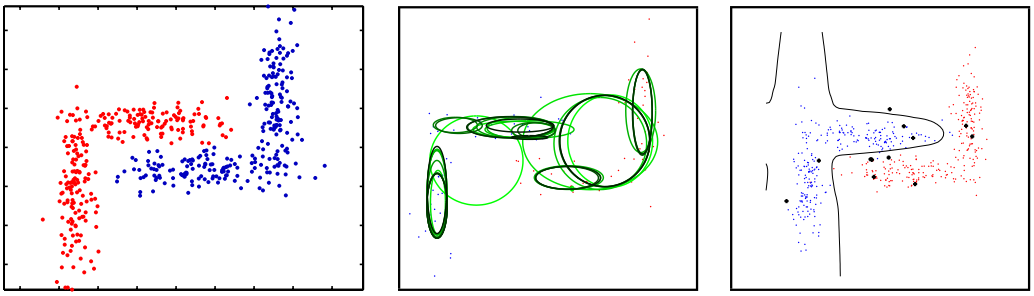

Figure 2.4. **Left**: A 2-d synthetic example of a non-convex and non-Gaussian dataset. **Center:** The Gaussians learnt by *KernelBoost*-dissolve (presented in Sec. 2.4.1.2). The Ellipses mark 1-std contours. Darker ellipses show Gaussians obtained at later boosting rounds. **Right:** The separator induced by the Gaussians for this example. Support vectors are marked by black dots.

from a local neighborhood graph. Classes are therefore split into small local subsets. The dissolve mechanism proposed is presented below in Algorithm 5.

Algorithm 5 The adaptive label-dissolve mechanism.

Preprocess: For each label l, compute a local neighborhood graph where each labeled datapoint is connected to all of its mutual neighbors from the first N_{mutual} neighbors.

For $t = 1 \dots T$ do

For each label l do

1. Define the edge weights on the graph to be the pairwise weights $W^t_{i_1,i_2}$ computed by the boosting process.
2. Threshold edges by removing all edges whose weight is smaller than the average edge weight given by $\frac{1}{|l|} \sum_{(i_1,i_2)\in l} W^t_{i_1,i_2}$.
3. Compute the connected components of the graph and use them to define a partition of the labels from the current class into small and local subsets.

This mechanism has one tunable parameter N_{mutual}, which determines the pre-computed neighborhood graph for each of the labels [6]. This parameter implicitly affects the number of subsets obtained at each boosting round. The effect of using this mechanism on a non-linear non-convex dataset can be seen in Figure 2.4 (center and right plots).

[6] Neighbors are defined as "mutual" iff i is within the first N neighbors of j and vice-versa.

Chapter 3

The RCA algorithm - Learning a Mahalanobis Metric using Equivalence Relations

This chapter includes the following publications

[A] Aharon Bar-Hillel, Tomer Hertz, Noam Shental and Daphna Weinshall, **Learning Distance Functions Using Equivalence Relations** *in 20th International Conference on Machine Learning (ICML 2003)*, Washington DC, August 2003.

Learning Distance Functions using Equivalence Relations

Aharon Bar-Hillel AHARONBH@CS.HUJI.AC.IL
Tomer Hertz TOMBOY@CS.HUJI.AC.IL
Noam Shental FENOAM@CS.HUJI.AC.IL
Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
School of Computer Science and Engineering and the Center for Neural Computation, The Hebrew University of Jerusalem, Jerusalem, Israel 91904

Abstract

We address the problem of learning distance metrics using side-information in the form of groups of "similar" points. We propose to use the RCA algorithm, which is a simple and efficient algorithm for learning a full ranked Mahalanobis metric (Shental et al., 2002). We first show that RCA obtains the solution to an interesting optimization problem, founded on an information theoretic basis. If the Mahalanobis matrix is allowed to be singular, we show that Fisher's linear discriminant followed by RCA is the optimal dimensionality reduction algorithm under the same criterion. We then show how this optimization problem is related to the criterion optimized by another recent algorithm for metric learning (Xing et al., 2002), which uses the same kind of side information. We empirically demonstrate that learning a distance metric using the RCA algorithm significantly improves clustering performance, similarly to the alternative algorithm. Since the RCA algorithm is much more efficient and cost effective than the alternative, as it only uses closed form expressions of the data, it seems like a preferable choice for the learning of full rank Mahalanobis distances.

Keywords: Learning from partial knowledge, semi-supervised learning, feature selection, clustering

1. Introduction

Many learning algorithms use a distance function over the input space as a principal tool, and their performance critically depends on the quality of the metric. Learning a "good" metric from examples may therefore be the key to a successful application of these algorithms. In many cases choosing the right metric may be more important than the specific algorithm which is later used.

Choosing the right metric is especially important in the unsupervised setting of clustering tasks, for such clustering algorithms as K-means and graph based methods. There are also supervised classification techniques which are distance based such as K-Nearest-Neighbors. Kernel machines use inner-product functions which are closely related to the Euclidean distance metric. In this wide variety of algorithms the problem of finding a good metric is equivalent to the problem of finding a good *representation function* $f : X \rightarrow Y$, transferring the data X into representation Y. We will therefore discuss the two problems interchangeably. Our main goal in this paper is to design a simple method for learning a metric, in order to improve the subsequent performance of unsupervised learning techniques. This is accomplished using side-information in the form of equivalence relations. Equivalence relations provide us with small groups of data points that are known to be similar (or dissimilar).

A key observation is that in many unsupervised learning tasks, such groups of similar points may be extracted from the data with minimal effort and possibly automatically, without the need for labels. This occurs when the data originates from a natural sequence that can be modeled as a Markovian process. Consider for example the task of movie segmentation, where the objective is to find all the frames in which the same actor appears. Due to the continuous nature of most movies, faces extracted from successive frames in roughly the same location can be assumed to come from the same person. This is true as long as there is no scene change, which can be automatically and robustly detected (Boreczky & Rowe, 1996). Another analogous example is speaker segmentation and recognition, in which a conversation between several

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

speakers needs to be segmented and clustered according to the speaker identity. Here, it may be possible to automatically identify small segments of speech which are likely to contain data points from a single *unknown* speaker.

In this paper we discuss the problem of learning linear *representation functions*, or equivalently an optimal Mahalanobis distance between data points, using equivalence relations. Specifically, we focus here on the Relevant Component Analysis (RCA) algorithm, which was first introduced in (Shental et al., 2002); the algorithm is reviewed in Section 2. In Section 3 we present a new analysis, based on a novel information theoretic optimality criterion. RCA is shown to be an optimal learning procedure in this sense. We show that Fisher's linear discriminant function followed by RCA optimizes the same criterion if dimensionality reduction is allowed.

In Section 4 we show that RCA can be presented as an optimal solution to a problem of minimizing inner class distances. Viewed this way, RCA can be directly compared with the approach proposed in (Xing et al., 2002), which is another recent algorithm for metric learning with side information. The comparison shows that the optimality criteria of the two algorithms are similar, but some arbitrary aspects of the criterion presented in (Xing et al., 2002) do not exist in RCA. Our empirical study also shows that the results of the algorithms are comparable: We empirically tested the RCA algorithm on a number of databases from the UCI repository, showing significant improvement in clustering performance which is similar or better than the improvement reported in (Xing et al., 2002). The major difference between the two algorithms is computational: RCA is robust and efficient since it only uses closed-form expressions of the data; the algorithm described in (Xing et al., 2002), on the other hand, uses iterative methods which are sensitive to parameter tuning and which are very demanding computationally.

Related work

There has been much work on learning representations and distance functions in the supervised learning setting, and we can just briefly mention some examples. (Hastie & Tibshirani, 1996) and (Jaakkola & Haussler, 1998) use labeled data to learn good metrics for classification. In (Thrun, 1996) a distance function (or a representation function) is learned for classification using a "leaning-to-learn" paradigm. In this setting several related classification tasks are learned using several labeled data sets, and algorithms are proposed which learn representations and distance functions in a way that allows for the transfer of knowledge between the tasks. In (Tishby et al., 1999) the joint distribution of two random variables X and Y is assumed to be known, and the problem is reduced to the learning of a compact representation of X which bears high relevance to Y. This work, which is further developed in (Chechik & Tishby, 2002), can be viewed as supervised representation learning. Information theoretic criteria for unsupervised learning in neural networks were first suggested by (Linsker, 1989), and has been used since in several tasks in the neural network literature, e.g., (Bell & Sejnowski, 1995).

In recent years some work has been done using equivalence relations as side information. In (Wagstaff et al., 2001) equivalence relations were introduced into the K-means clustering algorithm. Both positive ('a is similar to b') and negative ('a is dissimilar from b') relations were used. The problem of finding a better Mahalanobis metric using equivalence relations was addressed in (Xing et al., 2002), in conjunction with the constrained K-means algorithm. We compare this algorithm to our current work in Section 4, and compare our empirical results with the results of both algorithms in section 6. We have also recently developed a way to introduce both positive and negative equivalence relations into the EM algorithm for the estimation of a mixture of Gaussian models (Hertz et al., 2002; Shental et al., 2003).

2. Relevant Component Analysis

Relevant Component Analysis (RCA) is a method that seeks to identify and down-scale global unwanted variability within the data. The method changes the feature space used for data representation, by a global linear transformation which assigns large weights to "relevant dimensions" and low weights to "irrelevant dimensions" (cf. (Tenenbaum & Freeman, 2000)). These "relevant dimensions" are estimated using chunklets. We define a *chunklet* as a subset of points that are known to belong to the same although *unknown* class; chunklets are obtained from equivalence relations by applying a transitive closure. The RCA transformation is intended to reduce clutter, so that in the new feature space, the inherent structure of the data can be more easily unraveled. The method can be used as a preprocessing step for the unsupervised clustering of the data or nearest neighbor classification.

Specifically, RCA does the following (see illustration in Fig. 1a-f):

1. For each chunklet, subtract the chunklet's mean

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

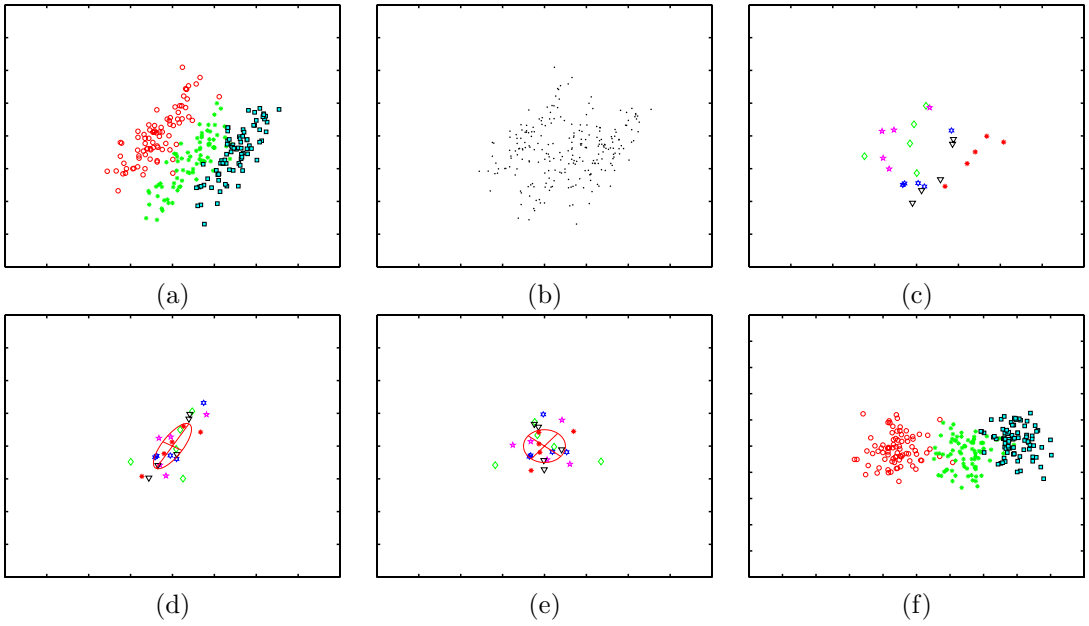

Figure 1. An illustrative example of the RCA algorithm applied to synthetic Gaussian data. (a) The fully labeled data set with 3 classes. (b) Same data unlabeled; clearly the classes' structure is less evident. (c) The set of chunklets that are provided to the RCA algorithm (points that share the same color and marker type form a chunklet). (d) The centered chunklets, and their empirical covariance. (e) The whitening transformation applied to the chunklets. (f) The original data after applying the RCA transformation.

from all of the points it contains (Fig. 1d).

2. Compute the covariance matrix of all the centered data-points in chunklets (Fig. 1d). Assume a total of p points in k chunklets, where chunklet j consists of points $\{x_{ji}\}_{i=1}^{n_j}$ and its mean is $\hat{m}_j$. RCA computes the following matrix:

$$\hat{C} = \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ji}-\hat{m}_j)(x_{ji}-\hat{m}_j)^t \qquad (1)$$

3. Compute the whitening transformation $W = \hat{C}^{-\frac{1}{2}}$ associated with this covariance matrix (Fig. 1e), and apply it to the original data points: $x_{new} = Wx$ (Fig. 1f). Alternatively, use the inverse of $\hat{C}$ as a Mahalanobis distance.

In effect, the whitening transformation W assigns lower weight to some directions in the original feature space; those are the directions in which the data variability is mainly due to within class variability, and is therefore "irrelevant" for the task of classification.

3. Information maximization under chunklet constraints

In this section we suggest an information theoretic formulation for the problem at hand. The problem is formulated as a constrained search for a good *representation function* . Although it is possible to state the problem for general families of transformations, we treat here only the linear case. In section 3.1 we present and discuss the problem formulation. In 3.2 we show that RCA solves this problem when only linear invertible transformations are considered. In section 3.3 we extend the family of functions considered to include non-invertible linear transformations, which leads to dimensionality reduction. We show that when the data is Gaussian, the solution is given by Fisher's linear discriminant followed by RCA.

3.1. An information theoretic perspective

Following (Linsker, 1989), an information theoretic criterion states that when an input X is transformed into a new representation Y, we should seek to maximize the mutual information $I(X,Y)$ between X and Y under suitable constraints. In the general deterministic case a set $X = \{x_l\}_{l=1}^{n}$ of data points in $\mathcal{R}^N$ is transformed into the set $Y = \{f(x_l)\}_{l=1}^{n}$ of points in $\mathcal{R}^M$. We wish to find a function $f \in F$ that maximizes $I(X,Y)$, where F is the family of allowed transformation functions (the "hypotheses family").

In our case we are also given a set of chunklets of data points from X, $\{x_{ji}\}_{j=1,i=1}^{k\ ,\ n_j}$, which the repre-

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

sentation function f is required to keep close to each other. Therefore, we may pose the problem as:

$$\max_{f \in F} I(X,Y) \quad s.t. \quad \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j} ||y_{ji} - m_j^y||^2 \leq K \quad (2)$$

where m_j^y denotes the mean of points in chunklet j after the transformation, P is the total number of points in chunklets, and K is a constant. The mutual information here is the differential mutual information between two continuous variables X and Y, and it depends on their respective densities. One should note that we can only asses these densities using the provided sample of data points.

Since in our case f is deterministic, the maximization of $I(X,Y)$ is achieved by maximizing the entropy $H(Y)$ alone. To see this, recall that

$$I(X,Y) = H(Y) - H(Y|X)$$

Since f is deterministic, there is no uncertainty concerning Y when X is known. Thus $H(Y|X)$ has its lowest possible value at $-\infty$.[1] However, as noted in (Bell & Sejnowski, 1995), $H(Y|X)$ does not depend on f but on the quantization scale. For every finite quantization of the space this term is a constant. Hence maximizing with respect to f can be done by considering only the first term, $H(Y)$.

It should be noted that $H(Y)$ can be increased by simply 'stretching' the data space (e.g. by choosing $f = \lambda x$, where $\lambda > 1$). Therefore, a constraint that keeps certain points close together is required in order to prevent this trivial scaling solution. Also the family F of *representation functions* should be carefully chosen to avoid trivial solutions.

3.2. RCA from an information theoretic perspective

We now look at the problem posed for the family F of invertible linear functions. When f is an invertible function, the connection between the densities of $Y = f(X)$ and X is expressed by $p_y(y) = \frac{p_x(x)}{|J(x)|}$, where $|J(x)|$ is the Jacobian of the transformation. Noting that $p_y(y)dy = p_x(x)dx$, we can relate $H(Y)$ and $H(X)$ as follows:

$$H(Y) = -\int_y p(y)\log p(y)dy = -\int_x p(x)\log\frac{p(x)}{|J(x)|}dx = H(X) + \langle \log|J(x)|\rangle_x$$

For a linear function $Y = AX$ the Jacobian is constant and equals $|A|$, and it is the only term in $I(X,Y)$ that depends on the transformation A. Hence problem (2) becomes

$$\max_A |A| \quad s.t. \quad \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j} ||x_{ji} - m_j||^2_{A^tA} \leq K \quad (3)$$

Let $B = A^tA$ denote a Mahalanobis distance matrix, where B is positive definite and $\log|A| = \frac{1}{2}\log|B|$. (3) can now be rewritten as

$$\max_B |B| \quad (4)$$
$$s.t. \quad \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j} ||x_{ji} - m_j||^2_B \leq \mathrm{K}\ ,\ B > 0$$

Writing and solving for the Lagrangian, we get the solution $B = \frac{K}{N}\hat{C}^{-1}$ where $\hat{C}$ is the average chunklet covariance matrix (1) and N is the dimension of the data space. The solution is identical to the Mahalanobis matrix proposed by RCA up to a scale factor.[2] Hence RCA is the solution of (4).

3.3. Dimensionality reduction

In this section we analyze the problem posed in Section 3.1 for the case of general linear transformations, i.e. $Y = AX$ where $A \in \mathcal{M}_{M\times N}$ and $M \leq N$. To simplify the analysis, we assume that X is a multivariate Gaussian. As we saw earlier, maximizing $H(Y)$ is equivalent to maximizing $I(X,Y)$ with respect to f. Since X is assumed to be Gaussian, Y is also Gaussian and its entropy is given by

$$\begin{aligned} H(Y) &= \frac{d}{2}\log 2\pi e + \frac{1}{2}\log|\Sigma_y| \\ &= \frac{d}{2}\log 2\pi e + \frac{1}{2}\log|A\Sigma_x A^t| \end{aligned}$$

so that (2) becomes

$$\max_A \log|A\Sigma_x A^t| \quad (5)$$
$$s.t. \quad \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j} ||x_{ji} - m_j||^2_{A^tA} \leq K$$

For a given target dimension M the solution to the problem is Fisher linear discriminant followed by applying RCA in the reduced dimensional space. A sketch of the proof is given in appendix A.

[1] This non-intuitive divergence is a result of the generalization of information theory to continuous variables; specifically, it is a result of ignoring the discretization constant in the definition of differential entropy.

[2] Such a scale constant is not important in classification tasks, i.e. when using relative distances.

4. RCA also minimizes inner class distances

In order to gain some intuition to the solution provided by the information maximization criterion formalized in Eq. (2), let us look at the optimization problem obtained by reversing the roles of the maximization term and the constraint term:

$$\min_B \frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}||x_{ji}-m_j||_B^2 \quad s.t.\ \ |B|\geq 1 \qquad (6)$$

In (6) a Mahalanobis distance B is sought, which minimizes the sum of all inner chunklet squared distances. Demanding that $|B| \geq 1$ amounts to the demand that minimizing the distances will not be achieved by "shrinking" the entire space. Using Kuhn-Tucker theorem, we can reduce (6) to

$$\min_B \sum_{j=1}^{k}\sum_{i=1}^{n_j}||x_{ji}-m_j||_B^2 - \lambda\log|B| \qquad (7)$$
$$s.t.\ \ \lambda\geq 0,\quad \lambda\log|B|=0$$

Differentiating the Lagrangian above shows that the minimum is given by $B = |\hat{C}|^{\frac{1}{d}}\hat{C}^{-1}$, where C is the average chunklet covariance matrix. Once again, the solution is identical to the Mahalanobis matrix proposed by RCA up to a scale factor.

It is interesting, in this respect, to compare RCA and the method proposed recently by (Xing et al., 2002). They also consider the problem of learning a Mahalanobis distance using side information in the form of pairwise similarities.[3] They assume knowledge of a set S of pairs of points known to be similar, and a set D of pairs of points known to be dissimilar. Given these sets, they pose the following optimization problem.

$$\min_B \sum_{(x_1,x_2)\in S}||x_1-x_2||_B^2 \qquad (8)$$
$$s.t. \sum_{(x_1,x_2)\in D}||x_1-x_2||_B,\quad B\geq 0$$

This problem is solved using gradient ascent and iterative projection methods.

To allow a clearer comparison of RCA to Eq. (8), we can cast (6) as a minimization of inner chunklet pairwise distances. For each point x_{ji} in chunklet j we have:

$$x_{ji}-m_j = x_{ji}-\frac{1}{n_j}\sum_{k=1}^{n_j}x_{jk} = \frac{1}{n_j}\sum_{\substack{k=1\\k\neq i}}^{n_j}(x_{ji}-x_{jk})$$

[3] Chunklets of size > 2 are not considered.

Problem (6) can now be rewritten as

$$\min_B \sum_{j=1}^{k}\frac{1}{n_j^2}\sum_{i=1}^{n_j}||\sum_{k\neq i}(x_{ji}-x_{jk})||_B^2 \quad s.t.\ \ |B|\geq 1 \qquad (9)$$

When only chunklets of size 2 are given (as in the case studied by Xing et al.), the problem reduces to

$$\min_B \frac{1}{2}\sum_{j=1}^{k}||x_{j1}-x_{j2}||_B^2 \quad s.t.\ \ |B|\geq 1 \qquad (10)$$

Clearly the minimization terms in problems (10) and (8) are identical up to a constant ($\frac{1}{2}$). The difference between the two problems lies in the constraint term they use. The constraint proposed by Xing et al. tries to use information concerning pairs of dissimilar points, whereas the constraint in the RCA formulation can be interpreted as a pure scale constraint, which does not allow the 'volume' of the Mahalanobis neighborhood to shrink.

Although the constraint used by Xing et al. appears to take into consideration further information, closer look shows that it is somewhat arbitrary. The usage of squared distance in the minimization term and the root of square distance for the constraint term is arbitrary and a-symmetric. Most importantly, it should be noted that in most unsupervised applications dissimilar pairs are not explicitly available. In this case (Xing et al., 2002) recommends to take D to be all the pairs of points that are not in S. This is a problematic choice for two reasons: In most practical scenarios pairs of points which are not in S are not necessarily dissimilar. In addition, this definition usually yields a very large set D, which substantially slows the algorithm's running time. In contrast, the RCA distance computation is simple and fast (requiring a single matrix inversion) without any need for an iterative procedure.

In order to further justify the constraint suggested in problem (6), we proceed to suggest a probabilistic interpretation of the RCA algorithm.

5. RCA and Maximum Likelihood

We now analyze the case of data which consists of several normally distributed classes which share the same covariance matrix. Under the assumption that the chunklets are sampled i.i.d and that points within each chunklet are also sampled i.i.d, the likelihood of the chunklets' distribution can be written as:

$$\prod_{j=1}^{k}\prod_{i=1}^{n_j}\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\tfrac{1}{2}(x_{ji}-m_j)^t\Sigma^{-1}(x_{ji}-m_j)\right) \qquad (11)$$

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

It is easy to see that the RCA Mahalanobis matrix $\hat{C}$ from (1) maximizes (11) over all possible choices of Σ^{-1}, and is therefore the Maximum Likelihood estimator in this setting.

In order to gain further insight into the constraint chosen in (6), we take the log of the likelihood equation (11), drop constant terms and denote $B = \Sigma^{-1}$, to obtain:

$$\hat{C} = \arg\min_{B} \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_{ji} - m_j||_B^2 - p \log |B| \quad (12)$$

where p denotes the total number of points in all chunklets. This equation is closely related to the Lagrangian in (7), but here λ (the Lagrange multiplier) is replaced by the constant p. Hence, under Gaussian assumptions, the solution of problem (7) has a probabilistic justification.

The effect of chunklet size

Under Gaussian assumptions, we can define an *unbiased* version of the RCA estimator. Assume for simplicity that there are p constrained data points divided into n chunklets of size k each. The *unbiased* RCA estimator can be written as follows :

$$\hat{C}(n,k) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{k-1} \sum_{j=1}^{k} (x_i^j - \hat{m}_i)(x_i^j - \hat{m}_i)^t \quad (13)$$

where x_i^j denotes the data point j in the chunklet i, and $\hat{m}_i$ denotes the empirical mean of chunklet i. $\hat{C}(n,k)$ in (13) is the empirical mean of the covariance estimators produced by each chunklet. It can be shown that the variance of the estimator matrix elements $\hat{C}_{ij}$ is bounded by

$$Var(\hat{C}_{ij}(n,k)) \leq \frac{k}{k-1} Var(\hat{C}_{ij}(1,nk)) \quad (14)$$

where $\hat{C}_{ij}(1,nk)$ is the estimator when all the $p = nk$ points are known to belong to the same class, thus forming the best estimate possible when given p points. For proof see (Hertz et al., 2002). The bound shows that the variance of the RCA estimator using small chunklets rapidly converges to the variance of this best estimator.

6. Experimental Results: Application to clustering

As noted in the introduction, the main goal of our method is to use side information in the form of equivalence relations to improve the performance of unsupervised learning techniques. In order to test our proposed RCA algorithm and to compare it with the work presented by Xing et. al, we used six data sets from the UC Irvine repository which were used in (Xing et al., 2002). As in (Xing et al., 2002) we are given a set S of pairwise similarity constraints (or chunklets of size 2).[4] We used the following clustering algorithms:

1. K-means using the default Euclidean metric (i.e. using no side-information).
2. Constrained K-means: K-means subject to points $(x_i, x_j) \in S$ always being assigned to the same cluster (Wagstaff et al., 2001).
3. Constrained K-means + Metric proposed by (Xing et al., 2002): Constrained K-means using the distance metric proposed in (Xing et al., 2002), which is learned from S.
4. Constrained K-means + RCA: Constrained K-means using the RCA distance metric learned from S.
5. EM: Expectation Maximization of a Gaussian Mixture model (using no side-information).
6. Constrained EM: EM using side-information in the form of equivalence constraints (Hertz et al., 2002; Shental et al., 2003), when using the RCA distance metric as an initial metric.

Following (Xing et al., 2002) we will use a normalized accuracy score to evaluate the partitions obtained by the different clustering algorithms presented above. More formally, in the case of 2-cluster data the accuracy measure used can be written as:

$$\sum_{i>j} \frac{1\{1\{c_i = c_j\} = 1\{\hat{c}_i = \hat{c}_j\}\}}{0.5m(m-1)}$$

where $1\{\dot{}\}$ is the indicator function ($1\{True\} = 1), 1\{False\} = 0$), $\{\hat{c}_i\}_{i=1}^m$ is the cluster to which point x_i is assigned by the clustering algorithm, and c_i is the "correct" or desired assignment. The score above is equivalent to computing the probability that the algorithm's assignment $\hat{c}$ of two randomly drawn points x_i and x_j agrees with the "true" assignment c.[5]

[4] To allow for a fair comparison with (Xing et al., 2002), we repeated their exact experimental setup and criteria.

[5] As noted in (Xing et al., 2002), this score needs normalization when the number of clusters is larger than 2. The normalization is achieved by sampling the pairs x_i and x_j from the same cluster (as determined by $\hat{c}$) with probability 0.5 and from different clusters with probability 0.5, so that "matches" and "mismatches" are given the same weight.

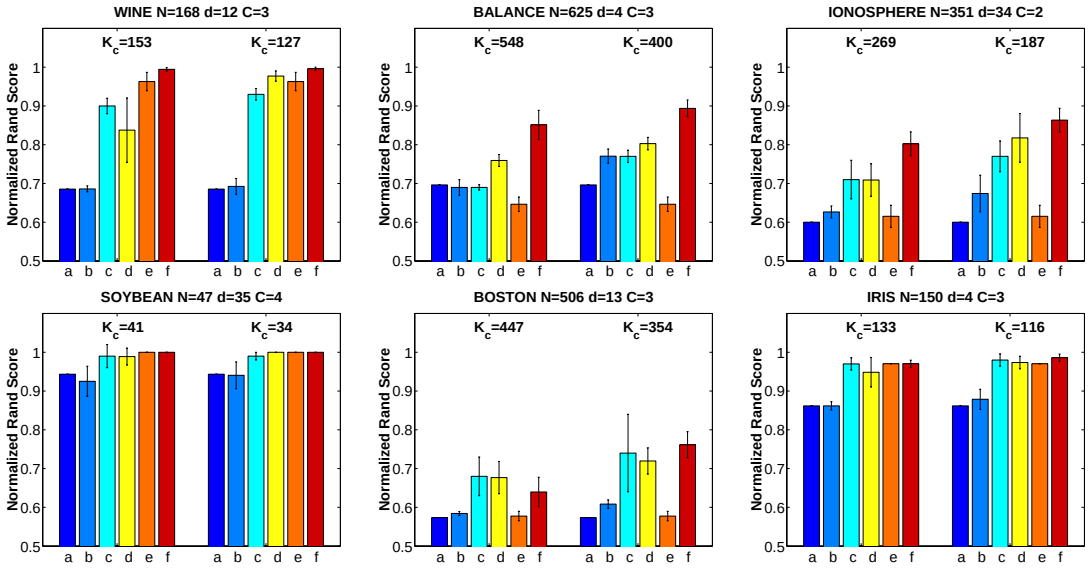

Figure 2. Clustering accuracy on 6 UCI datasets. In each panel, the six bars on the left correspond to an experiment with "little" side-information, and the six bars on the right correspond to "much" side-information. From left to right the six bars are respectively: (a) K-means over the original feature space (without using any side-information). (b) Constrained K-means over the original feature space. (c) Constrained K-means over the feature space suggested by (Xing et al., 2002). (d) Constrained K-means over the feature space created by RCA. (e) EM over the original feature space (without using any side-information). (f) Constrained EM (Shental et al., 2003) over the feature space created by RCA. Also shown are N - the number of points, C - the number of classes, d - the dimension of the feature space, and K_c - the mean number of connected components (see footnote 6). The results were averaged over 20 realizations of side-information.

As in (Xing et al., 2002) we tested our method using two conditions: (1) using "little" side-information S; (2) using "much" side-information.[6] As in (Xing et al., 2002) in all of our experiments we used K-means with multiple restarts.

Fig. 2 shows the results of all algorithms described above when using the two conditions of "little" and "much" side-information.

Clearly using RCA as a distance measure significantly improves the results over the original K-means algorithm. When comparing our results with the results reported in (Xing et al., 2002), we see that RCA achieves similar results. In this respect it should be noted that the RCA metric computation is a single step efficient computation, whereas the method presented in (Xing et al., 2002) requires gradient descent and iterative projections.

[6] S was generated by choosing a random subset of all pairs of points sharing the same class c_i. In the case of little side-information, the size of the subset was chosen so that the resulting number of connected components K_c (using transitive closure over pairs) is roughly 90% of the size of the original dataset. In case of much side information this was changed to 70%.

7. Discussion and Concluding remarks

We have presented an algorithm which makes use of side-information in the form of equivalence relations to learn a Mahalanobis metric. We have shown that our method is optimal under several criteria, and also showed considerable improvement in clustering on several standard datasets.

RCA is one of several techniques which we have developed for using equivalence relations to enhance unsupervised learning. In a related technique, we introduced the constraints into an EM formulation of a Gaussian Mixture Model (Hertz et al., 2002; Shental et al., 2003). This work enhances the power of RCA in two ways: First, it makes it possible to incorporate negative constraints. Second, it allows further improvement of the RCA metric, as may be seen in Fig. 2.

References

Bell, A., & Sejnowski, T. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, *7*, 1129–1159.

Boreczky, J. S., & Rowe, L. A. (1996). Comparison of video shot boundary detection techniques. *SPIE Storage and Retrieval for Still Images and Video Databases IV*, *2664*, 170–179.

Chechik, G., & Tishby, N. (2002). Extracting relevant structures with side information. *NIPS*, *15*.

Fukunaga, K. (1990). *Statistical pattern recognition*. San Diego: Academic Press. 2nd edition.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification and regression. *Advances in Neural Information Processing Systems* (pp. 409–415). The MIT Press.

Hertz, T., Shental, N., Bar-hillel, A., & Weinshall, D. (2002). Enhancing image and video retrieval: Learning via equivalence constraints. *http://www.cs.huji.ac.il/ daphna/*.

Jaakkola, T., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers.

Linsker, R. (1989). An application of the principle of maximum information preservation to linear systems. *NIPS* (pp. 186–194). Morgan Kaufmann.

Shental, N., Hertz, T., Bar-Hilel, A., & Weinshall, D. (2003). Computing gaussian mixture models with EM using equivalence constraints.

Shental, N., Hertz, T., Weinshall, D., & Pavel, M. (2002). Adjustment learning and relevant component analysis. *Computer Vision - ECCV*.

Tenenbaum, J., & Freeman, W. (2000). Separating style and content with bilinear models. *Neural Computation*, *12*, 1247–1283.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems* (pp. 640–646). The MIT Press.

Tishby, N., Pereira, F., & Bialek, W. (1999). The information bottleneck method. *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing* (pp. 368–377).

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-means clustering with background knowledge. *Proc. 18th International Conf. on Machine Learning* (pp. 577–584). Morgan Kaufmann, San Francisco, CA.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2002). Distance metric learnign with application to clustering with side-information. *Advances in Neural Information Processing Systems*. The MIT Press.

Appendix A: Information Maximization in the case of non invertible linear transformation

Here we briefly sketch the proof of the claim made in Section 3.3. As before, we denote by C the average covariance matrix of the chunklets. We can rewrite the constrained expression as:

$$\frac{1}{p}\sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ji}-m_j)^t A^t A(x_{ji}-m_j) = tr(A^t AC) = tr(A^t CA)$$

Hence the Lagrangian may be written as:

$$\log|A\Sigma_x A^t| - \lambda(tr(ACA^t) - K)$$

Differentiating the Lagrangian w.r.t A leads to

$$\Sigma_x A(A^t\Sigma_x A)^{-1} = \lambda CA \tag{15}$$

Multiplying by A^t and rearranging we get: $\frac{I}{\lambda} = A^t CA$. This equation does not give us information concerning the subspace to which the optimal A takes us. However, A whitens the data with respect to the chunklet covariance C in this subspace, similarly to RCA. From $\lambda \neq 0$ it then follows that the inequality constraint is an equality, which can be used to find λ.

$$tr(ACA^t) = tr(\frac{I}{\lambda}) = \frac{M}{\lambda} = K \implies \lambda = \frac{M}{K}$$
$$\implies ACA^t = \frac{K}{M}I$$

Now, since in our solution space $ACA^t = \frac{K}{M}I$, $\log|ACA^t| = M\log\frac{K}{M}$ holds for all points. Hence we can modify the maximization argument as follows

$$\log|A\Sigma_x A^t| \quad = \quad \log\frac{|A\Sigma_x A^t|}{|ACA^t|} + M\log\frac{K}{M}$$

Now the optimization argument has a familiar form. It is known (Fukunaga, 1990) that maximizing the determinant ratio can be done by projecting the space on the span of the first M eigenvectors of $C^{-1}\Sigma_x$. Denote by B the solution matrix for this unconstrained problem. In order to enforce the constraints we define the matrix $A = \sqrt{\frac{K}{M}}\Lambda_1^{-0.5}B$ and we claim that A is the solution of the constrained problem. Notice that the value of the maximization argument does not change when we switch from A to B since A is a product of B and another full ranked matrix. It can also be shown that A satisfies the constraints and is thus the solution of the problem presented in Eq. (5).

Chapter 4

The DistBoost Algorithm - Learning Non-Linear Distance Functions

This chapter includes the following publications:

[B] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Boosting Margin Based Distance Functions for Clustering**, *in the 21st International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada July 2004.

[C] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Learning Distance Functions for Image Retrieval** *in IEEE Conf. on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

Boosting Margin Based Distance Functions for Clustering

Tomer Hertz TOMBOY@CS.HUJI.AC.IL
Aharon Bar-Hillel AHARONBH@CS.HUJI.AC.IL
Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
Center for Neural Computation and School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel 91904

Abstract

The performance of graph based clustering methods critically depends on the quality of the distance function used to compute similarities between pairs of neighboring nodes. In this paper we learn distance functions by training binary classifiers with margins. The classifiers are defined over the product space of pairs of points and are trained to distinguish whether two points come from the same class or not. The signed margin is used as the distance value. Our main contribution is a distance learning method (*DistBoost*), which combines boosting hypotheses over the product space with a weak learner based on partitioning the original feature space. Each weak hypothesis is a Gaussian mixture model computed using a semi-supervised constrained EM algorithm, which is trained using both unlabeled and labeled data. We also consider SVM and decision trees boosting as margin based classifiers in the product space. We experimentally compare the margin based distance functions with other existing metric learning methods, and with existing techniques for the direct incorporation of constraints into various clustering algorithms. Clustering performance is measured on some benchmark databases from the UCI repository, a sample from the MNIST database, and a data set of color images of animals. In most cases the *DistBoost* algorithm significantly and robustly outperformed its competitors.

1. Introduction

Graph based clustering methods have been widely and successfully used in many domains such as computer vision, bioinformatics and exploratory data analysis. This category spans a wide range of algorithms, from classical agglomerative methods such as *average linkage* (Duda et al., 2001), to the recently developed and more sophisticated spectral methods (Shi & Malik, 2000) and stochastic formulations (Blatt et al., 1997; Gdalyahu et al., 2001). The initial representation in all these methods is a matrix (or graph) of distances between all pairs of datapoints. The computation of this distance matrix is considered a "preprocessing" step, and typically one uses some L_p norm on the feature space (or a related variant).

Despite the important differences between the various graph-based clustering algorithms, it is widely acknowledged that clustering performance critically depends on the quality of the distance function used. Often the quality of the distance function is more important then the specifics of the clustering algorithm. In this paper we focus on the question of how to learn a "good" distance function, which will lead to improved clustering. Our main contribution is *DistBoost* - a novel semi-supervised algorithm for learning distance functions.

We consider a semi-supervised clustering scenario in which the data is augmented by some sparse side information, in the form of equivalence constraints. Equivalence constraints are relations between pairs of data points, which indicate whether the points belong to the same category or not. We term a constraint 'positive' when the points are known to be from the same class, and 'negative' otherwise. Such constraints carry *less* information than explicit labels on the original datapoints, since clearly equivalence constraints can be obtained from explicit labels but **not** vice versa. More importantly, it has been suggested that in some cases equivalence constraints are easier to obtain, especially when the database is very large and contains a large number of categories without pre-defined names (Hertz et al., 2003).

In recent years there has been a growing interest in semi supervised clustering scenarios, leading to two different (and related) lines of research. In the first, the constraints are incorporated directly into the clustering algorithm, limiting the clustering solutions considered to those that comply with the given constraints. Examples are the constrained complete linkage algorithm (Klein et al., 2002),

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

constrained K-means (Wagstaff et al., 2001) and a constrained EM of a Gaussian mixture (Shental et al., 2003). The second line of research, to which this work belongs, uses the constraints to learn an informative distance function (prior to clustering). Most of the work in this area has focused on the learning of Mahalanobis distance functions of the form $(x-y)^T A(x-y)$ (Shental et al., 2002; Xing et al., 2002). In these papers the parametric Mahalanobis metric was used in combination with some suitable parametric clustering algorithm, such as K-means or EM of a mixture of Gaussians. In contrast, we develop in this paper a method that learns a non-parametric distance function, which can be more naturally used in non-parametric graph based clustering.

More formally, let $\mathcal{X}$ denote the original data space, and assume that the data is sampled from M discrete labels. Our goal is to learn a distance function $f : \mathcal{X} \times \mathcal{X} \rightarrow [0,1]$.[1] Our key observation is that we can learn such a function, by posing a related binary classification problem over the product space $\mathcal{X} \times \mathcal{X}$, and solving it using margin based classification techniques. The binary problem is the problem of distinguishing between pairs of points that belong to the same class and pairs of points that belong to different classes.[2] The training data included a set of equivalence constraints, which can be formally regarded as binary labels on points in $\mathcal{X} \times \mathcal{X}$. If we label pairs of points from the same class by 0 and pairs of points belonging to different classes by 1, we can interpret the classifier's margin as the required distance function.

Having reduced distance learning to binary classification with margins, we can now attempt to solve this problem using standard powerful margin based classifiers. We have explored both support vector machines (SVM's) and boosting algorithms. However, experiments with several SVM variants and decision trees (C4.5) boosting have led us to recognize that the specific classification problem we are interested in has some unique features which require special treatment:

1. The product space binary function we wish to learn has some unique structure which may lead to 'unnatural' partitions of the space between the labels. The concept we wish to learn is an indicator of an equivalence relation over the original space. Thus the properties of transitivity and symmetry of the relation place geometrical constraints on the binary hypothesis. Obviously, traditional families of hypotheses, such as linear separators or decision trees, are not limited to equivalence relation indicators, and it's not easy to enforce these constraints when such classifiers are used.

2. In the learning setting we have described above, we are provided with N datapoints in $\mathcal{X}$ and with a sparse set of equivalence constraints (or labels in product space) over some pairs of points in our data. We assume that the number of equivalence constraints provided is much smaller than the total number of equivalence constraints $O(N^2)$, and is of order $O(N)$. We therefore have access to large amounts of unlabeled data, and hence semi-supervised learning seems like an attractive option. However, classical binary classifiers like SVM and boosting methods are trained using labeled data only.

These considerations led us to develop the *DistBoost* algorithm, which is our main contribution in this paper. *DistBoost* is a distance learning algorithm which attempts to address the issues discussed above. It learns a distance function which is based on boosting binary classifiers with a confidence interval in product space, using a weak learner that learns in the *original* feature space (and not in product space). We suggest a boosting scheme that incorporates unlabeled data points. These unlabeled points provide a density prior, and their weights rapidly decay during the boosting process. The weak learner we use is based on a constrained Expectation Maximization (EM) algorithm, which computes a Gaussian mixture model, and hence provides a partition of the original space. The constrained EM procedure uses unlabeled data and equivalence constraints to find a Gaussian mixture that complies with them. A weak product space hypothesis is then formed as the equivalence relation of the computed partition.

We have experimented with *DistBoost* and conducted several empirical comparisons of interest. The first is a comparison of *DistBoost* to other margin based distance functions obtained using the more traditional algorithms of SVM and decision tree boosting. Another comparison is between *DistBoost* and previously suggested distance learning algorithms which are based on Mahalanobis metric estimation. Finally, clustering using the distance function learnt by *DistBoost* is compared to previously suggested methods of incorporating equivalence constraints directly into clustering algorithms. During the comparative assessment *DistBoost* was evaluated with several agglomerative clustering algorithms and with different amounts of equivalence constraints information. We used several datasets from the UCI repository (Blake & Merz, 1998), A sample from the MNIST dataset (LeCun et al., 1998), and a dataset of natural images obtained from a commercial image CD. In most of our experiments the *DistBoost* method outperformed its competitors.

[1] Note that this function is not necessarily a metric, as the triangle inequality may not hold.

[2] Note that this problem is closely related to the multi class classification problem: if we can correctly generate a binary partition of the data in product space, we implicitly define a multi-class classifier in the original vector space $\mathcal{X}$.

2. Boosting original space partitions using *DistBoost*

The *DistBoost* algorithm builds distance functions based on the weighted majority vote of a set of original space soft partitions. The weak learner's task in this framework is to find plausible partitions of the space, which comply with the given equivalence constraints. In this task, the unlabeled data can be of considerable help, as it allows to define a prior on what are 'plausible partitions'. In order to incorporate the unlabeled data into the boosting process, we augmented the Adaboost with confidence intervals presented in (Schapire & Singer, 1999). The details of this augmentation are presented in Section 2.1. The details of the weak learner we use are presented in Section 2.2.

2.1. Semi supervised boosting in product space

Our boosting scheme is an extension of the Adaboost algorithm with confidence intervals (Schapire & Singer, 1999; Schapire et al., 1997) to handle unsupervised data points. As in Adaboost, we use the boosting process to maximize the margins of the labeled points. The unlabeled points only provide a decaying density prior for the weak learner. The algorithm we use is sketched in Fig. 1. Given a partially labeled dataset $\{(x_i, y_i)\}_{i=1}^{N}$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis $f(x) = \sum_{i=1}^{k} \alpha_k h(x)$ which minimizes the following loss function:

$$\sum_{\{i|y_i=1,-1\}} \exp(-y_i f(x_i)) \qquad (1)$$

Note that the unlabeled points do not contribute to the minimization objective (1). Rather, at each boosting round they are given to the weak learner and supply it with some (hopefully useful) information regarding the domain's density. The unlabeled points effectively constrain the search space during the weak learner estimation, giving priority to hypotheses which both comply with the pairwise constraints and with the density information. Since the weak learner's task becomes harder in later boosting rounds, the boosting algorithm slowly reduces the weight of the unlabeled points given to the weak learner. This is accomplished in step 4 of the algorithm (see Fig. 1).

In product space there are $O(N^2)$ unlabeled points, which correspond to all the possible pairs of original points, and the number of weights is therefore $O(N^2)$. However, the update rules for the weight of each unlabeled point are identical, and so all the unlabeled points can share the same weight. Hence the number of updates we effectively do in each round is proportional to the number of labeled pairs only. The weight of the unlabeled pairs is guaranteed to decay at least as fast as the weight of any labeled pair. This immediately follows from the update rule in step 4 of the algorithm (Fig. 1), as each unlabeled pair is treated as a labeled pair with maximal margin of 1.

Algorithm 1 Boosting with unlabeled data

Given $(x_1, y_1), ..., (x_n, y_n)$; $x_i \in X$, $y_i \in \{-1, 1, *\}$
Initialize $D_1(i) = 1/n \;\; i = 1, .., n$

For $t = 1, .., T$

1. Train weak learner using distribution D_t
2. Get weak hypothesis $h_t : X \rightarrow [-1, 1]$ with $r_t = \sum_{i=1}^{n} D_t(i) h_t(i) > 0$.
   If no such hypothesis can be found, terminate the loop and set $T = t$.
3. Choose $\alpha_t = \frac{1}{2} \ln(\frac{1+r}{1-r})$
4. Update:
$$D_{t+1}(i) = \begin{cases} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) & y_i \in \{-1, 1\} \\ D_t(i) \exp(-\alpha_t) & y_i = * \end{cases}$$
5. Normalize: $D_{t+1}(i) = D_{t+1}(i)/Z_{t+1}$
   where $Z_{t+1} = \sum_{i=1}^{n} D_{t+1}(i)$
6. Output the final hypothesis $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

We note in passing that it is possible to incorporate unlabeled data into the boosting process itself, as has been suggested in (d'Alche Buc et al., 2002; Grandvalet et al., 2001). In this work the margin concept was extended to unlabeled data points. The margin for such a point is a positive number related to the confidence the hypothesis has in classifying this point. The algorithm then tries to minimize the total (both labeled and unlabeled) margin cost. The problem with this framework is that a hypothesis can be very certain about the classification of unlabeled points, and hence have low margin costs, even when it classifies these points incorrectly. In the semi supervised clustering context the total margin cost may be dominated by the margins of unconstrained point pairs, and hence minimizing it doesn't necessarily lead to hypotheses that comply with the constraints. Indeed, we have empirically tested some variants of these algorithms and found that they lead to inferior performance.

2.2. Mixtures of Gaussians as weak hypotheses

The weak learner in *DistBoost* is based on the constrained EM algorithm presented by (Shental et al., 2003). This al-

gorithm learns a mixture of Gaussians over the original data space, using unlabeled data and a set of positive and negative constraints. Below we briefly review the basic algorithm, and then show how it can be modified to incorporate weights on sample data points. We also describe how to translate the boosting weights from product space points to original data points, and how to extract a product space hypothesis from the soft partition found by the EM algorithm.

A Gaussian mixture model (GMM) is a parametric statistical model which assumes that the data originates from a weighted sum of several Gaussian sources. More formally, a GMM is given by $p(x|\Theta) = \Sigma_{l=1}^{M} \alpha_l p(x|\theta_l)$, where α_l denotes the weight of each Gaussian, θ_l its respective parameters, and M denotes the number of Gaussian sources in the GMM. EM is a widely used method for estimating the parameter set of the model (Θ) using unlabeled data (Dempster et al., 1977). In the constrained EM algorithm *equivalence constraints* are introduced into the 'E' (Expectation) step, such that the expectation is taken only over assignments which comply with the given constraints (instead of summing over *all* possible assignments of data points to sources).

Assume we are given a set of unlabeled i.i.d. sampled points $X = \{x_i\}_{i=1}^{N}$, and a set of pairwise constraints over these points Ω. Denote the index pairs of positively constrained points by $\{(p_j^1, p_j^2)\}_{j=1}^{N_p}$ and the index pairs of negatively constrained points by $\{(n_k^1, n_k^2)\}_{k=1}^{N_n}$. The GMM model contains a set of discrete hidden variables H, where the Gaussian source of point x_i is determined by the hidden variable h_i. The constrained EM algorithm assumes the following joint distribution of the observables X and the hiddens H:

$$p(X, H|\Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^{n} \alpha_{h_i} p(x_i|\theta_{h_i}) \prod_{j=1}^{N_p} \delta_{h_{p_j^1} h_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{h_{n_k^1} h_{n_k^2}}) \quad (2)$$

The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (2) with respect to H.

The equivalence constraints create complex dependencies between the hidden variables of different data points. However, the joint distribution can be expressed using a Markov network, as seen in Fig. 1. In the 'E' step of the algorithm the probabilities $p(h_i|X, \Theta, \Omega)$ are computed by applying a standard inference algorithm to the network. Such inference is feasible if the number of negative constraints is $O(N)$, and the network is sparsely connected. The model parameters are then updated based on the computed probabilities. The update of the Gaussian parameters $\{\theta_l\}$ can be done in closed form, using rules similar to the standard EM update rules. The update of the cluster weights $\{\alpha_l\}_{l=1}^{M}$ is more complicated, since these parameters appear in the normalization constant Z in (2), and the solution is found

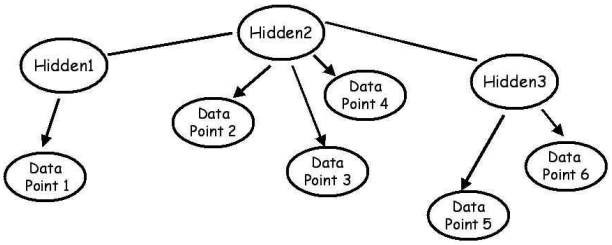

Figure 1. A Markov network representation of the constrained mixture setting. Each observable data node has a discrete hidden node as its ancestor. Positively constrained nodes have the same hidden node as their ancestor. Negative constraints are expressed using edges between the hidden nodes of negatively constrained points.Here points 2,3,4 are constrained to be together, and point 1 is constrained to be from a different class.

with a gradient descent procedure. The algorithm finds a local maximum of the likelihood, but the partition found is not guaranteed to satisfy any specific constraint. However, since the boosting procedure increases the weights of points which belong to unsatisfied equivalence constraints, it is most likely that any constraint will be satisfied in one or more partitions.

We have incorporated weights into the constrained EM procedure according to the following semantics: The algorithm is presented with a virtual sample of size N_v. A training point x_i with weight w_i appears $w_i N_v$ times in this sample. All the repeated tokens of the same point are considered to be positively constrained, and are therefore assigned to the same source in every evaluation in the 'E' step. In all of our experiments we have set N_v to be the actual sample size.

While the weak learner accepts a distribution over the original space points, the boosting process described in 2.1 generates a distribution over the sample product space in each round. The product space distribution is converted to a distribution over the sample points by simple marginalization. Specifically, denote by w_{ij}^p the weight of pair (i, j); the weight w_i^s of point i is defined to be

$$w_i^s = \sum_j w_{ij}^p \quad (3)$$

In each round, the mixture model computed by the constrained EM is used to build a binary function over the product space and a confidence measure. We first derive a partition of the data from the Maximum A Posteriori (MAP) assignment of points. A binary product space hypothesis is then defined by giving the value 1 to pairs of points from the same Gaussian source, and -1 to pairs of points from different sources. This value determines the sign of the hypothesis output. This setting further supports a natural confidence measure - the probability of the pair's

MAP assignment which is:

$$\max_i p(h_1 = i|x_1, \Theta) \cdot \max_i p(h_2 = i|x_2, \Theta)$$

where h_1, h_2 are the hidden variables attached to the two points. The weak hypothesis output is the signed confidence measure in $[-1, 1]$, and so the weak hypothesis can be viewed as a weak "distance function".

3. Learning in the product space using traditional classifiers

We have tried to solve the distance learning problem over the product space using two more traditional margin based classifiers. The first is a support vector machine, that tries to find a linear separator between the data examples in a high dimensional feature space. The second is the AdaBoost algorithm, where the weak hypotheses are decision trees learnt using the C4.5 algorithm. Both algorithms had to be slightly adapted to the task of product space learning, and we have empirically tested possible adaptations using data sets from the UCI repository. Specifically, we had to deal with the following technical issues:

- Product space representation: A pair of original space points must be converted into a single point, which represents this pair in the product space. The simplest representation is the concatenation of the two points. Another intuitive representation is the concatenation of the sum and difference vectors of the two points. Our empirical tests indicated that while SVM works better with the first representation, the C4.5 boosting achieves its best performance with the 'sum and difference' representation.

- Enforcing symmetry: If we want to learn a symmetric distance function satisfying $d(x, y) = d(y, x)$, we have to explicitly enforce this property. With the first representation this can be done simply by doubling the number of training points, introducing each constrained pair twice: as the point $[x, y]$ and as the point $[y, x]$. In this setting the SVM algorithm finds the global optimum of a symmetric Lagrangian and the solution is guaranteed to be symmetric. With the second representation we found that modifying the representation to be symmetrically invariant gave the best results. Specifically, we represent a pair of points x, y using the vector $[x+y, sign(x_1-y_1)*(x-y)]$, where x_1, y_1 are the first coordinates of the points.

- We considered two linear preprocessing transformations of the original data before creating the product space points: the whitening transformation, and the RCA transformation (Bar-Hilel et al., 2003) which uses positive equivalence constraints. In general we found that pre-processing with RCA was most beneficial for both the SVM and C4.5 boosting algorithms.

- Parameter tuning: for the SVM we used the polynomial kernel of order 4, and a trade-off constant of 1 between error and margin. The boosting algorithm was run for 25-150 rounds (depending on the dataset), and the decision trees were built with a stopping criterion of train error smaller than 0.05 in each leaf.

The clustering performance obtained using these two variants is compared to *DistBoost* in section 4. The design issues mentioned above were decided based on the performance over the UCI datasets, and the settings remained fixed for the rest of the experiments.

4. Experimental Results

We compared our *DistBoost* algorithm with other techniques for semi-supervised clustering using equivalence constraints. We used both distance learning techniques, including our two simpler variants for learning in product space (SVM and boosting decision trees), and constrained clustering techniques. We begin by introducing our experimental setup and the evaluated methods. We then present the results of all these methods on several datasets from the UCI repository, a subset of the MNIST letter recognition dataset, and an animal image database.

4.1. Experimental setup

Gathering equivalence constraints: Following (Hertz et al., 2003), we simulated a *distributed learning* scenario, where labels are provided by a number of uncoordinated independent teachers. Accordingly, we randomly chose small subsets of data points from the dataset and partitioned each of the subsets into equivalence classes. The constraints obtained from all the subsets are gathered and used by the various algorithms.

The size of each subset k in these experiments was chosen to be $2M$, where M is the number of classes in the data. In each experiment we used l subsets, and the amount of partial information was controlled by the *constraint index* $P = k \cdot l$; this index measures the amount of points which participate in at least one constraint. In our experiments we used $P = 0.5, 1$. However, it is important to note that the number of equivalence constraints thus provided typically includes only a small subset of all possible pairs of datapoints, which is $\mathcal{O}(N^2)$.

Evaluated Methods: we compared the clustering performance of the following techniques:

1. Our proposed boosting algorithm (*DistBoost*).

2. Mahalanobis distance learning with Relevant Component Analysis (RCA) (Bar-Hilel et al., 2003).

3. Mahalanobis distance learning with non-linear optimization (Xing) (Xing et al., 2002).

4. Margin based distance learning using SVM as a product space learner (SVM) (described in Section 3).

5. Margin based distance learning using product space decision trees boosting (DTboost).

6. Constrained EM of a Gaussian Mixture Model (Constrained EM) (Shental et al., 2003).

7. Constrained Complete Linkage (Constrained Complete Linkage) (Klein et al., 2002).

8. Constrained K-means (COP K-means) (Wagstaff et al., 2001).

Methods 1-5 compute a distance function, and they are evaluated by applying a standard agglomerative clustering algorithm (Ward) to the distance graph they induce. Methods 6-8 incorporate equivalence constraints directly into the clustering process.

All methods were evaluated by clustering the data and measuring the $F_{\frac{1}{2}}$ score defined as

$$F_{\frac{1}{2}} = \frac{2P * R}{R + P} \tag{4}$$

where P denotes precision and R denotes recall. For the distance learning techniques we also show *cumulative neighbor purity* curves. *Cumulative neighbor purity* measures the percentage of correct neighbors up to the K-th neighbor, averaged over all the datapoints. In each experiment we averaged the results over 50 or more different equivalence constraint realizations. Both *DistBoost* and the decision tree boosting algorithms were run for a constant number of boosting iterations $T = 25, 150$ (depending on the dataset). In each realization all the algorithms were given the exact same equivalence constraints.

Dimensionality reduction: the constrained LDA algorithm Some of the datasets reside in a high dimensional space, which must be reduced in order to perform parameter estimation from training data. We used two methods for dimensionality reduction: standard Principal Components Analysis (PCA), and a constrained Linear Discriminant Analysis (LDA) algorithm which is based on equivalence constraints.

Classical LDA (also called FDA, (Fukunaga, 1990)) computes projection directions that minimize the within-class scatter and maximize the between-class scatter. More formally, given a labeled dataset $\{x_i, y_i\}_{i=1}^N$ where $y_i \in \{0, 1, .., M-1\}$ and $x_i \in R^d$, LDA is given by the $k \times d$ matrix W that maximizes

$$J(W) = \frac{W^T S_t W}{W^T S_w W} \tag{5}$$

where $S_t = \sum_{i=1}^N (x_i - m)(x_i - m)^T$ denotes the *total scatter* matrix (m is the data's empirical mean) and $S_w = \sum_{j=0}^{M-1} \sum_{i:y_i=j} (x_i - m_j)(x_i - m_j)^T$ denotes the *within-class scatter* matrix (m_j is the empirical mean of the j-th class).

Since in our semi-supervised learning scenario we have access to equivalence constraints instead of labels, we can write down a constrained LDA algorithm. Thus we estimate the *within class scatter* matrix using positive equivalence constraints instead of labels. Specifically, given a set of positive equivalence constraints, we use transitive closure over this set to obtain small subsets of points that are known to belong to the same class. Denote these subsets by $\{C_j\}_{j=0}^{L-1}$, where each subset C_j is composed of a variable number of data points $C_j = \{x_{j1}, x_{j2}, .., x_{jn_j}\}$. We use these subsets to estimate S_w as follows

$$S_{\hat{w}} = \sum_{j=0}^{L-1} \sum_{i=1}^{N_j} (x_{ji} - m_j)(x_{ji} - m_j)^T \tag{6}$$

where here m_j denotes the mean of subset C_j.

4.2. Results on UCI datasets

We selected several datasets from the UCI data repository and used the experimental setup above to evaluate the various methods. Fig. 2 shows clustering $F_{\frac{1}{2}}$ score plots for several data sets using Ward's agglomerative clustering algorithm. Clearly *DistBoost* achieves significant improvements over Mahalanobis based distance measures and other product space learning methods. Comparing *DistBoost* to methods which incorporate constraints directly, clearly the only true competitor of *DistBoost* is its own weak learner, the constrained EM algorithm. Still, in the vast majority of cases *DistBoost* gives an additional significant improvement over the EM.

4.3. Results on the MNIST letter recognition dataset

We compared all clustering methods on a subset of the MNIST letter recognition dataset (LeCun et al., 1998). We randomly selected 500 training samples (50 from each of the 10 classes). The original data dimension was 784, which was projected by standard PCA to the first 50 principal dimensions. We then further projected the data using the constrained LDA algorithm to 40 dimensions. Clustering and neighbor purity plots are presented on the left side of Fig 3. The clustering performance of the *DistBoost* algorithm is significantly better than the other methods. The

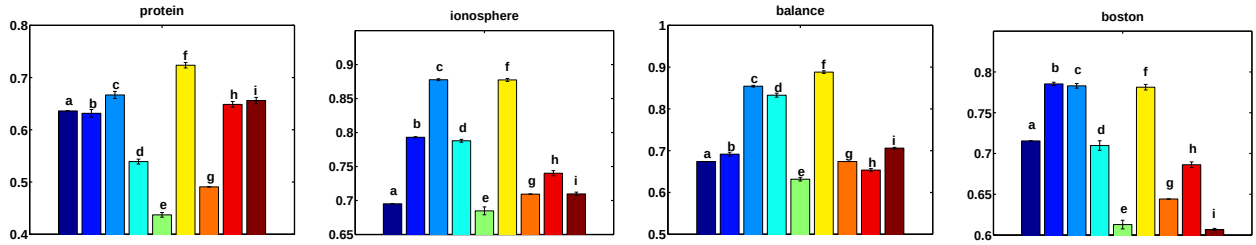

Figure 2. Clustering $F_{\frac{1}{2}}$ score over 4 data sets from the UCI repository using Ward's clustering algorithm. Methods shown are: (a) Euclidean, (b) RCA, (c) constrained EM, (d) SVM, (e) DTboost, (f) DistBoost, (g) Xing, (h) Constrained Complete Linkage, (i) Constrained K-means. The results were averaged over 100 realizations of constraints, and 1-std error bars are shown. The *constraint index* P was 0.5 in all cases.

cumulative purity curves suggest that this success may be related to the slower decay of the neighbor purity scores for *DistBoost*.

4.4. Results on an Animal image dataset

We created an image database which contained images of animals taken from a commercial image CD, and tried to cluster them based on color features. The clustering task in this case is much harder than in the previous applications. The database contained 10 classes with total of 565 images. Fig. 3 shows a few examples of images from the database.

The original images were heavily compressed jpg images. The images were represented using Color Coherence Vectors (Pass et al., 1996) (CCV's). This representation extends the color histogram representation, by capturing some crude spatial properties of the color distribution in an image. Specifically, in a CCV vector each histogram bin is divided into two bins, representing the number of 'Coherent' and 'Non-Coherent' pixels from each color. 'Coherent' pixels are pixels whose neighborhood contains more than τ neighbors which have the same color. We represented the images in HSV color space, quantized the images to $32 * 32 * 32 = 32768$ color bins, and computed the CCV of each image - a $64K$ dimensional vector - using $\tau = 25$.[3]

In order to reduce the dimension of our data, we first removed all zero dimensions and then used the first 100 PCA dimensions, followed by Constrained LDA to further reduce the dimension of the data to $d = 40$. The clustering results and neighbor purity graphs are presented on the right side of Fig 3.[4] The difficulty of the task is well reflected in the low clustering scores of all the methods. However, *DistBoost* still outperforms its competitors, as it did in all previous examples.

[3]The standard distance measure used on CCV features is a Chi-squared distance (also commonly used to measure distance between histograms). We also tried to cluster the data using the Chi-squared distances, and the $F_{\frac{1}{2}}$ score obtained was 0.44.

[4]On this dataset the COP k-means algorithm only converged on 25% of its runs.

5. Discussion

In this paper, we have described *DistBoost* - a novel algorithm which learns distance functions that enhance clustering performance using sparse side information. Our extensive comparisons showed the advantage of our method over many competing methods for learning distance functions and for clustering using equivalence constraints. Another application which we have not explored here, is nearest neighbor classification. Nearest neighbor classification also critically depends on the distance function between datapoints; our hope is that distance functions learned from equivalence constraints can also be used for improving nearest neighbor classification.

References

Bar-Hilel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Blatt, M., Wiseman, S., & Domany, E. (1997). Data clustering using a model granular magnet. *Neural Computation, 9*, 1805–1842.

d'Alche Buc, F., Grandvalet, Y., & Ambroise, C. (2002). Semi-supervised marginboost.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JRSSB, 39*, 1–38.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*. John Wiley and Sons Inc.

Fukunaga, K. (1990). *Statistical pattern recognition*. San Diego: Academic Press. 2nd edition.

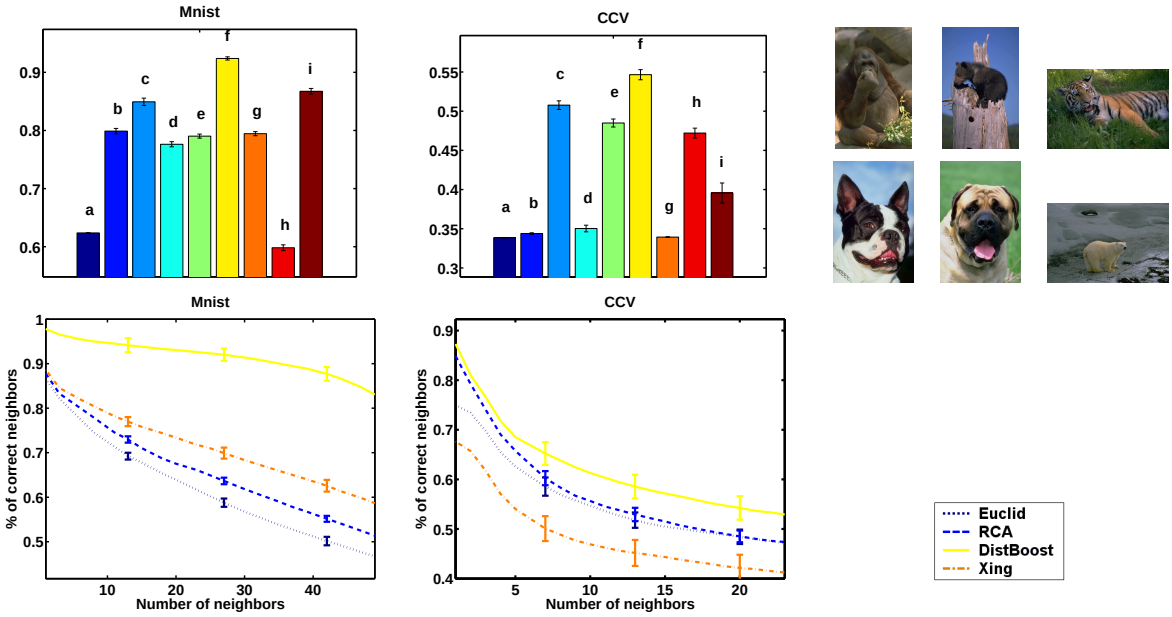

Figure 3. Top Clustering results using Ward's algorithm on a subset of the MNIST dataset (500 datapoints, 10 classes) and on the animal color image database (565 images, 10 classes). Bottom: Cumulative neighbor purity graphs on the same datasets. Methods shown are: (a) Euclidean, (b) RCA, (c) constrained EM, (d) SVM, (e) DTboost, (f) DistBoost, (g) Xing, (h) Constrained Complete Linkage, (i) Constrained K-means. Results were averaged over 50 realizations. The constraint index P is 1 in all cases.

Gdalyahu, Y., Weinshall, D., & Werman., M. (2001). Self organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization.

Grandvalet, Y., d'Alche Buc, F., & Ambroise, C. (2001). Boosting mixture models for semi supervised learning.

Hertz, T., Bar-Hillel, A., Shental, N., & Weinshall, D. (2003). Enhancing image and video retrieval: Learning via equivalence constraints. *IEEE Conf. on Computer Vision and Pattern Recognition, Madison WI, June 2003*.

Klein, D., Kamvar, S., & Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278–2324.

Pass, G., Zabih, R., & Miller, J. (1996). Comparing images using color coherence vectors. *ACM Multimedia* (pp. 65–73).

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. *Proc. 14th International Conference on Machine Learning* (pp. 322–330). Morgan Kaufmann.

Schapire, R. E., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning, 37*, 297–336.

Shental, N., Hertz, T., Bar-Hilel, A., & Weinshall, D. (2003). Computing gaussian mixture models with EM using equivalence constraints.

Shental, N., Hertz, T., Weinshall, D., & Pavel, M. (2002). Adjustment learning and relevant component analysis. *Computer Vision - ECCV*.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*, 888–905.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-means clustering with background knowledge. *Proc. 18th International Conf. on Machine Learning* (pp. 577–584). Morgan Kaufmann, San Francisco, CA.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2002). Distance metric learnign with application to clustering with side-information. *Advances in Neural Information Processing Systems*. The MIT Press.

Learning Distance Functions for Image Retrieval

Tomer Hertz, Aharon Bar-Hillel and Daphna Weinshall
{email: tomboy,aharonbh,daphna@cs.huji.ac.il}
School of Computer Science and Engineering and the Center for Neural Computation
The Hebrew University of Jerusalem, Jerusalem Israel 91904

Abstract

Image retrieval critically relies on the distance function used to compare a query image to images in the database. We suggest to learn such distance functions by training binary classifiers with margins, where the classifiers are defined over the product space of pairs of images. The classifiers are trained to distinguish between pairs in which the images are from the same class and pairs which contain images from different classes. The signed margin is used as a distance function. We explore several variants of this idea, based on using SVM and Boosting algorithms as product space classifiers. Our main contribution is a distance learning method which combines boosting hypotheses over the product space with a weak learner based on partitioning the original feature space. The weak learner used is a Gaussian mixture model computed using a constrained EM algorithm, where the constraints are equivalence constraints on pairs of data points. This approach allows us to incorporate unlabeled data into the training process. Using some benchmark databases from the UCI repository, we show that our margin based methods significantly outperform existing metric learning methods, which are based on learning a Mahalanobis distance. We then show comparative results of image retrieval in a distributed learning paradigm, using two databases: a large database of facial images (YaleB), and a database of natural images taken from a commercial CD. In both cases our GMM based boosting method outperforms all other methods, and its generalization to unseen classes is superior.

1. Introduction

Image retrieval is often done by computing the distance from a query image to images in the database, followed by the retrieval of nearest neighbors. The retrieval performance mainly depends on two related components: the image representation, and the distance function used. Given a specific image representation, the quality of the distance function used is the main key to a successful system.[1]. In this paper we focus on learning 'good' distance functions, that will improve the performance of content based image retrieval. The quality of an image retrieval system also depends on its ability to adapt to the intentions of the user as in relevance feedback methods [1]. Learning distance functions can be useful in this context for training user dependent distance functions.

Formally, let $\mathcal{X}$ denote the original data space, and assume that the data is sampled from M discrete labels where M is large and unknown. Our goal is to learn a distance function $f : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$. In order to learn such a function, we pose a related binary classification problem over product space, and solve it using margin based classification techniques. The binary problem is the problem of distinguishing between pairs of points that belong to the same class and pairs of points that belong to different classes.[2] If we label pairs of points from the same class by 0 and pairs of points belonging to different classes by 1, we can then view the classifier's margin as the required distance function.

The training data we consider is composed of binary labels on points in $\mathcal{X} \times \mathcal{X}$. The labels describe equivalence constraints between datapoints in the original space $\mathcal{X}$. Equivalence constraints are relations between pairs of datapoints, which indicate whether the point in the pair belong to the same category or not. We term a constraint 'positive' when the points are known to be from the same class, and 'negative' in the opposite case. Such constraints carry *less* information than explicit labels of the original images in $\mathcal{X}$, since clearly equivalence constraints can be obtained from M explicit labels on points in $\mathcal{X}$, but **not** vice versa. More importantly, we observe that equivalence constraints are easier to obtain, especially when the image database is very large and contains a large number of categories without pre-defined names.

To understand this observation, ask yourself how can you obtain training data for a large facial images database? You may ask a single person to label the images, but as the size of the database grows this quickly becomes impractical. Another approach is the *distributed learning* approach [9]: divide the data into small subsets of images and ask a

[1] A distance function is a function from pairs of datapoints to the positive real numbers, usually (but not necessarily) symmetric with respect to its arguments. We do not require that the triangle inequality holds, and thus our distance functions are *not* necessarily metrics.

[2] Note that this problem is closely related to the multi class classification problem: if we can correctly generate a binary partition of the data in product space, we implicitly define a multi-class classifier in the original vector space $\mathcal{X}$.The relations between the learnability of these two problems is discussed in [4].

number of people to label each subset. Note that you are still left with the problem of coordinating the labels provided by each of the labellers, since these are arbitrary. To illustrate the arbitrariness of tags, imagine a database containing all existing police facial images. While in one folder all the pictures of a certain individual may be called 'Insurance Fraud 205', different pictures of the same individual in another folder may be called 'Terrorist A'. In this distributed scenario, full labels are hard to obtain, but 'local' equivalence information can be easily gathered. [3]

Learning binary functions with margins over an input space is a well studied problem in the machine learning literature. We have explored two popular and powerful classifiers which incorporate margins: support vector machines (SVM's) and boosting algorithms. However, experiments with several SVM variants and Boosting decision trees (C4.5) have led us to recognize that the specific classification problem we are interested in has some unique features which require special treatment.

1. The product space binary function we wish to learn has some unique structure which may lead to 'unnatural' partitions of the space between the labels. The concept we learn is an indicator of an equivalence relation over the original space. Thus the properties of transitivity and symmetry of the relation place geometrical constraints on the binary hypothesis. Obviously, traditional families of hypotheses, such as linear separators or decision trees, are not limited to equivalence relation indicators, and it's not easy to enforce the constraints when such classifiers are used.

2. In the learning setting we have described above, we are provided with N datapoints in $\mathcal{X}$ and with equivalence constraints (or labels in product space) over some pairs of points in our data. We assume that the number of equivalence constraints provided is much smaller than the total number of equivalence constraints $O(N^2)$. We therefore have access to large amounts of unlabeled data, and hence semi-supervised learning seems an attractive option. However, classical SVM and boosting methods are trained using labeled data only.

These considerations led us to the development of the *DistBoost* algorithm, which is our main contribution in this paper. *DistBoost* is a distance learning algorithm which attempts to address all of the issues discussed above. It learns a distance function which is based on boosting binary classifiers with a confidence interval in product space, using a weak learner that learns in the *original* feature space (and not in product space). We suggest a boosting scheme that incorporates unlabeled data points. These unlabeled points provide a density prior, and their weights rapidly decay during the boosting process. The weak learner we use is based on a constrained EM algorithm, which computes a Gaussian mixture model, and hence provides a partition of the original space. The constrained EM procedure uses unlabeled data and equivalence constraints to find a Gaussian mixture that complies with them. A product space hypothesis is then formed based on the computed partition.

There has been little work on learning distance functions in the machine learning literature. Most of this work has been restricted to learning Mahalanobis distance functions of the form $(x-y)^T A(x-y)$. The use of labels for the computation of the weight matrix A has been discussed in [10]; the computation of A from equivalence constraints was discussed in [17, 13]. Yianilos [18] has proposed to fit a generative Gaussian mixture model to the data, and use the probability that two points were generated by the same source as a measure of the distance between them. Schemes for incorporating unlabeled data into the boosting process were introduced by Ambroise et. al [5, 19]. We discuss the relation between these schemes and our *DistBoost* algorithm in Section 3.

We have experimented with the *DistBoost* algorithm as well as other margin based distance learning algorithms, and compared them to perviously suggested methods which are based on Mahalanobis metric learning. We used several datasets from the UCI repository [15], the yaleB facial image dataset, and a dataset of natural images obtained from a commercial image CD. The results clearly indicate that our margin based distance functions provide much better retrieval results than all other distance learning methods. Furthermore, on all these datasets the *DistBoost* method outperforms all other methods, including our earlier margin based methods which use state of the art binary classifiers.

2. Learning in the product space using traditional classifiers

We have tried to solve the distance learning problem over the product space using two of the most powerful margin based classifiers. The first is a support vector machine, that tries to find a linear separator between the data examples in a high dimensional feature space. The second is the AdaBoost algorithm, where the weak hypotheses are decision trees learnt using the C4.5 algorithm. Both algorithms had to be slightly adapted to the task of product space learning, and we have empirically tested possible adaptations using data sets from the UCI repository. Specifically, we had to deal with the following technical issues:

- Product space representation: A pair of original space

[3] Inconsistencies which arise due to different definitions of distinct categories by different teachers are more fundamental, and are not addressed in this paper. Another way to solve the problem of tag arbitrariness is to use pre-defined category names, like letters or digits. Unfortunately this is not always possible, especially when the number of categories in the database is large and the specific categories are unknown apriori.

points must be converted into a single which represents this pair in the product space. The simplest representation is the concatenation of the two points. Another intuitive representation is the concatenation of the sum and difference vectors of the two points. Our empirical tests indicated that while the SVM works better with the first representation, the C4.5 boosting achieves its best performance with the 'sum and difference' representation.

- Enforcing symmetry: If we want to learn a symmetric distance function satisfying $d(x, y) = d(y, x)$, we have to explicitly enforce this property. With the first representation this can be done simply by doubling the number of training points, introducing each constrained pair twice: as the point $[x, y]$ and as the point $[y, x]$. In this setting the SVM algorithm finds the global optimum of a symmetric Lagrangian and the solution is guaranteed to be symmetric. With the second representation we found that modifying the representation to be symmetrically invariant gave the best results. Specifically, we represent a pair of points x, y using the vector $[x + y, sign(x_1 - y_1) * (x - y)]$, where x_1, y_1 are the first coordinates of the points.

- Preprocessing transformation in the original space: We considered two possible linear transformation of the data before creating the product space points: the whitening transformation, and the RCA transformation [9] which uses positive equivalence constraints. In general we found that pre-processing with RCA was most beneficial for both the SVM and C4.5 boosting algorithms.

- Parameter tuning: for the SVM we used the polynomial kernel of order 4, and a trade-off constant of 1 between error and margin. The boosting algorithm was run for 50 rounds, and the decision trees were built with a stopping criterion of train error smaller than 0.05 in each leaf.

These design issues were decided based on the performance over the UCI datasets, and all settings remained fixed for all further experiments.

3. Boosting original space partitions using *DistBoost*

Our *DistBoost* algorithm builds distance functions based on the weighted majority vote of a set of original space soft partitions. The weak learner's task in this framework is to find plausible partitions of the space, which comply with the given equivalence constraints. In this task, unlabeled data can be of considerable help, as it allows to define a prior on what are 'plausible partitions'. In order to incorporate the unlabeled data into the boosting process, we augmented an existing boosting version. The details of this augmentation are presented in Section 3.1. The details of our weak learner are presented in Section 3.2.

3.1. Semi supervised boosting in product space

Our boosting scheme is an extension of the Adaboost algorithm with confidence intervals [11] to handle unsupervised data points. As in Adaboost, we use the boosting process to maximize the margins of the labeled points. The unlabeled points only provide a decaying density prior for the weak learner. The algorithm we use is sketched in Fig. 1. Given a partially labeled dataset $\{(x_i, y_i)\}_{i=1}^N$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis $f(x) = \sum_{i=1}^{k} \alpha_k h(x)$ which minimizes the following loss function:

$$\sum_{\{i|y_i=1,-1\}} \exp(-y_i f(x_i)) \qquad (1)$$

Algorithm 1 Boosting with unlabeled data

Given $(x_1, y_1), ..., (x_n, y_n)$; $x_i \in X$, $y_i \in \{-1, 1, *\}$
Initialize $D_1(i) = 1/n \;\; i = 1, .., n$

For $t = 1, .., T$

1. Train weak learner using distribution D_t
2. Get weak hypothesis $h_t : X \to [-1, 1]$ with $r_t = \sum_{i=1}^n D_t(i) h_t(i) > 0$.
   If no such hypothesis can be found, terminate the loop and set $T = t$.
3. Choose $\alpha_t = \frac{1}{2} \ln(\frac{1+r}{1-r})$
4. Update:
$$D_{t+1}(i) = \begin{cases} D_t(i) \exp(-\alpha_t y_i h_t(x_i)) & y_i \in \{-1, 1\} \\ D_t(i) \exp(-\alpha_t) & y_i = * \end{cases}$$
5. Normalize: $D_{t+1}(i) = D_{t+1}(i)/Z_{t+1}$ where $Z_{t+1} = \sum_{i=1}^n D_{t+1}(i)$
6. Output the final hypothesis $f(x) = \sum_{t=1}^T \alpha_t h_t(x)$

Note that the unlabeled points do not contribute to the minimization objective of the product space boosting in (1). Rather, at each boosting round they are given to the weak learner and supply it with some (hopefully useful) information regarding the domain density. The unlabeled points effectively constrain the search space during the weak learner

estimation, giving priority to hypotheses which both comply with the pairwise constraints and with the density information. Since the weak learner's task becomes harder in later boosting rounds, the boosting algorithm slowly reduces the weight of the unlabeled points given to the weak learner. This is accomplished in step 4 of the algorithm (see Fig. 1).

In product space there are $O(N^2)$ unlabeled points, which correspond to all the possible pairs of original points, and the number of weights is therefore $O(N^2)$. However, the update rules for the weight of each unlabeled point are identical, and so all the unlabeled points can share the same weight. Hence the number of updates we effectively do in each round is proportional to the number of labeled pairs only. The weight of the unlabeled pairs is guaranteed to decay at least as fast as the weight of any labeled pair. This immediately follows from the update rule in step 4 of the algorithm (Fig. 1), as each unlabeled pair is treated as a labeled pair with maximal margin of 1.

We note in passing that it is possible to incorporate unlabeled data into the boosting process itself, as has been suggested in [5, 19]. Their idea was to extend the margin concept to unlabeled data points. The algorithm then tries to minimize the total (both labeled and unlabeled) margin cost. The problem with this framework is that a hypothesis can be very certain about the classification of unlabeled points , and hence have large margins, even when it classifies these points incorrectly. Indeed, we have empirically tested some variants of these algorithms and found poor generalization performance in our context.

3.2. Mixtures of Gaussians as product space weak hypotheses

The weak learner in *DistBoost* is based on the constrained EM algorithm presented in [9]. This algorithm learns a mixture of Gaussians over the original data space, using unlabeled data and a set of positive and negative constraints. In this section we briefly review the basic algorithm, and then show how it can be extended to incorporate weights on sample data points. We describe how to translate the boosting weights from product space points to original data points, and how to generate a product space hypothesis from the soft partition found by the EM algorithm.

A Gaussian mixture model (GMM) is a parametric statistical model which assumes that the data originates from a weighted sum of several Gaussian sources. More formally, a GMM is given by $p(x|\Theta) = \Sigma_{l=1}^{M} \alpha_l p(x|\theta_l)$, where α_l denotes the weight of each Gaussian, θ_l its respective parameters, and M denotes the number of Gaussian sources in the GMM. EM is a widely used method for estimating the parameter set of the model (Θ) using unlabeled data [6]. In the constrained EM algorithm *equivalence constraints* are introduced into the 'E' (Expectation) step, such that the expectation is taken only over assignments which comply with the given constraints (instead of summing over *all* possible assignments of data points to sources).

Assume we are given a set of unlabeled i.i.d. sampled points $X = \{x_i\}_{i=1}^{N}$, and a set of pairwise constraints over these points Ω. Denote the index pairs of positively constrained points by $\{(p_j^1, p_j^2)\}_{j=1}^{N_p}$ and the index pairs of negatively constrained points by $\{(n_k^1, n_k^2)\}_{k=1}^{N_n}$. The GMM model contains a set of discrete hidden variables H, where the Gaussian source of point x_i is determined by the hidden variable h_i. The constrained EM algorithm assumes the following joint distribution of the observables X and the hiddens H:

$$p(X, H|\Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^{n} \alpha_{h_i} p(x_i|\theta_{h_i}) \prod_{j=1}^{N_p} \delta_{h_{p_j^1} h_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{h_{n_j^1} h_{n_j^2}}) \quad (2)$$

The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (2) with respect to H.

The equivalence constraints create complex dependencies between the hidden variables of different data points. However, the joint distribution can be expressed using a Markov network, as seen in Fig. 1. In the 'E' step of the algorithm the probabilities $p(h_i|X, \Theta, \Omega)$ are computed by applying a standard inference algorithm to the network. Such an inference is feasible if the number of negative constraints is $O(N)$, and the network is sparsely connected. The model parameters are then updated based on the computed probabilities. The update of the Gaussian parameters $\{\theta_l\}$ can be done in closed form, using rules similar to the standard EM update rules. The update of the cluster weights $\{\alpha_l\}_{l=1}^{M}$ is more complicated, since these parameters appear in the normalization constant Z in (2), and it requires a gradient descent procedure. The algorithm finds a local maximum of the likelihood, but the partition found is not guaranteed to satisfy any specific constraint. However, since the boosting procedure increases the weights of points which belong to unsatisfied equivalence constraints, it is most likely that any constraint will be satisfied in some partitions.

We have incorporated weights into the constrained EM procedure according to the following semantics: The algorithm is presented with a virtual sample of size N_v. A training point x_i with weight w_i appears $w_i N_v$ times in this sample. All the repeated tokens of the same point are considered to be positively constrained, and are therefore assigned to the same source in every evaluation in the 'E' step. In all of our experiments we have set N_v to be the actual sample size.

While the weak learner accepts a distribution over original space points, the boosting process described in 3.1 generates a distribution over the sample product space in each

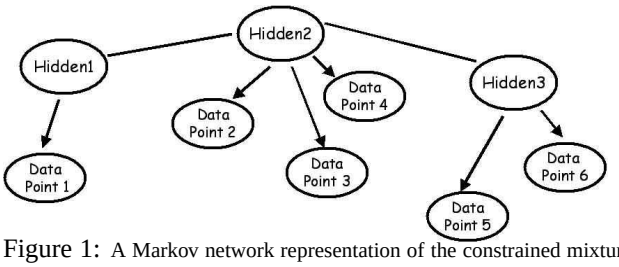

Figure 1: A Markov network representation of the constrained mixture setting. Each observable data node has a discrete hidden node as a father. Positively constrained nodes have the same hidden node as father. Negative constraints are expressed using edges between the hidden nodes of negatively constrained points.Here points 2,3,4 are known to be together, and point 1 is known to be from a different class.

round. The product space distribution is converted to a distribution over the sample points by simple summation. Denoting by w_{ij}^p the weight of pair (i,j), the weight w_i^s of point i is defined to be

$$w_i^s = \sum_j w_{ij}^p \tag{3}$$

In each round, the mixture model computed by the constrained EM is used to build a binary function over the product space and a confidence measure. We first derive a partition of the data from the Maximum A Posteriori (MAP) assignment of points. A binary product space hypothesis is then defined by giving the value 1 to pairs of points from the same Gaussian source, and -1 to pairs of points from different sources. This value determines the sign of the hypothesis output.

This setting further supports a natural confidence measure - the probability of the pair's MAP assignment which is:

$$\max_i p(h_1 = i|x_1, \Theta) \cdot \max_i p(h_2 = i|x_2, \Theta)$$

where h_1, h_2 are the hidden variables attached to the two points. The weak hypothesis output is the signed confidence measure in $[-1, 1]$, and so the weak hypothesis can be viewed as a 'weak distance function'.

4. Learning distance functions: comparative results

In this section we compare our *DistBoost* algorithm with other distance learning techniques, including our two other proposed methods for learning in product space (SVM and boosting decision trees). We begin by introducing our experimental setup. We then show results on several datasets from the UCI repository, which serve as benchmark to evaluate the different distance learning methods.

4.1. Experimental setup

Gathering equivalence constraints: we simulated a *distributed learning* scenario [9], where labels are provided by a number of uncoordinated independent teachers. Accordingly, we randomly chose small subsets of data points from the dataset and partitioned each of the subsets into equivalence classes.

The size of each subset k in these experiments was chosen to be $2M$, where M is the number of classes in the data. In each experiment we used l subsets, and the amount of partial information was controlled by the *constraint index* $P = k \cdot l$; this index measures the amount of points which participate in at least one constraint. In our experiments we used $P = 0.3, 0.5$. However, it is important to note that the number of equivalence constraints thus provided typically includes only a small subset of all possible pairs of datapoints, which is $\mathcal{O}(N^2)$.[4]

Evaluated Methods: we compared the performance of the following distance learning methods:

- Our proposed *DistBoost* algorithm.
- Mahalanobis distance learning with Relevant Component Analysis (RCA) [3].
- Mahalanobis distance learning with non-linear optimization [17].
- SVM for direct discrimination in product space.
- Boosting decision trees in product space.

In order to set a lower bound on performance, we also compared with the whitened Mahalanobis distance, where the weight matrix A is taken to be the data's global covariance matrix.

We present our results using ROC curves and *cumulative neighbor purity* graphs. *Cumulative neighbor purity* measures the percentage of correct neighbors up to the Kth neighbor, averaged over all the queries. In each experiment we averaged the results over 50 different equivalence constraint realizations. Both *DistBoost* and the decision tree boosting algorithms were run for a constant number of boosting iterations $T = 50$. In each realization all the algorithms were given the exact same equivalence constraints.

4.2. Results on UCI datasets

We selected several standard datasets from the UCI data repository and used the experimental setup above to evaluate our proposed methods. The cumulative purity was computed using all the points in the data as queries.

Fig. 2 shows neighbor purity plots for each of these data sets. As can be readily seen, *DistBoost* achieves significant improvements over Mahalanobis based distance measures,

[4] It can be readily shown that by wisely selecting $\mathcal{O}(NM)$ equivalence constraints, one can label the entire dataset. This follows from the transitive nature of positive equivalence constraints.

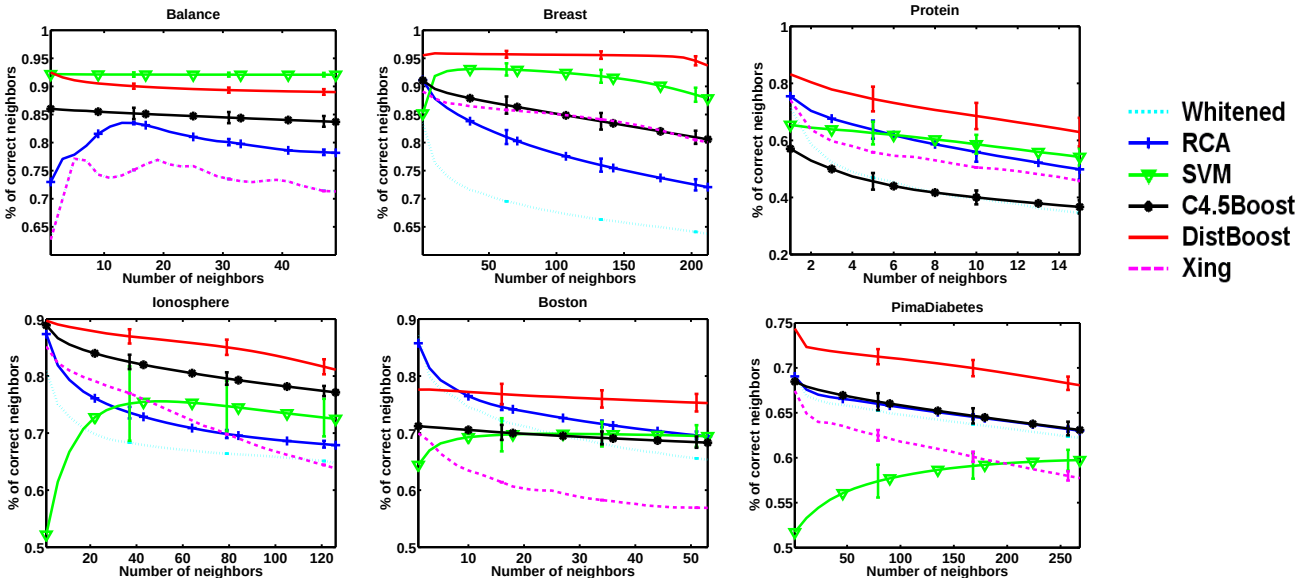

Figure 2: Cumulative neighbor purity plots over 6 data sets from the UCI repository. The UCI results were averaged over 50 realizations of constraints, and 1-std error bars are shown. The percentage of data in constraints was 50% in all cases.

and also outperforms all other product space learning methods (except SVM in the 'balance' dataset).

5. Experiments on image retrieval

We ran experiments on two image retrieval tasks: facial image retrieval using the YaleB dataset, and color based image retrieval using pictures from a commercial image CD. The evaluated methods are described in Section 4.1.

In our experiments we randomly selected from each dataset a subset of images, to be the retrieval database, and this subset was used as the training set. We then followed the same experimental setup of distributed learning (described in Section 4.1) for the generation of equivalence constraints, and trained all methods on the selected data. Retrieval performance was measured using test images which were not presented during training.

5.1 Facial image retrieval - YaleB

As an image retrieval example with known ground-truth and a clear definition of categories, we used a subset of the YaleB facial image database [7]. The dataset contains a total of 1920 images, including 64 frontal pose images of 30 different subjects. In this database the variability between images of the same person is mainly due to different lighting conditions. We automatically centered all the images using optical flow. Images were then converted to vectors, and each image was represented using its first 60 PCA coefficients. From each class, we used 22 images (a third) as a data base training set, and 42 images were used as test queries. In order to check different types of generalization,we used a slightly modified policy for constraint sampling. specifically, constraints were drawn from 20 out of the 30 classes in the dataset, and in the constrained classes p was set to 1 (which means that all the training points in these classes were divided between uncoordinated labellers). When testing the learnt distance functions measurements were done separately for test images from the first 20 classes and for the last 10. Notice that in this scenario images from the 10 unconstrained classes were not helpful in any way to the traditional algorithms, but they were used by *DistBoost* as unlabeled data. On the left in Fig. 3 we present the ROC curves of the different methods on test data from the constrained classes. We can see that the margin based distance functions give very good results, indicating an adaptation of the distance function to these classes. On the right we present the ROC curves when the queries are from unconstrained classes. It can be seen that the performance of SVM and C4.5Boost severely degrades, indicating strong overfit behavior. The *DistBoost*, on the other hand, degrades gracefully and is still better then the other alternatives.

5.2 Color based image retrieval

We created a picture database which contained images from 16 animal classes taken from a commercial image CD. The retrieval task in this case is much harder then in the facial retrieval application. We used 70% of the data from each class as our dataset (training data), and the remaining 30% as our test data. We experimented with two scenarios varying in their difficulty level. In the first scenario we used 10 classes with a total of 405 images. In the second scenario

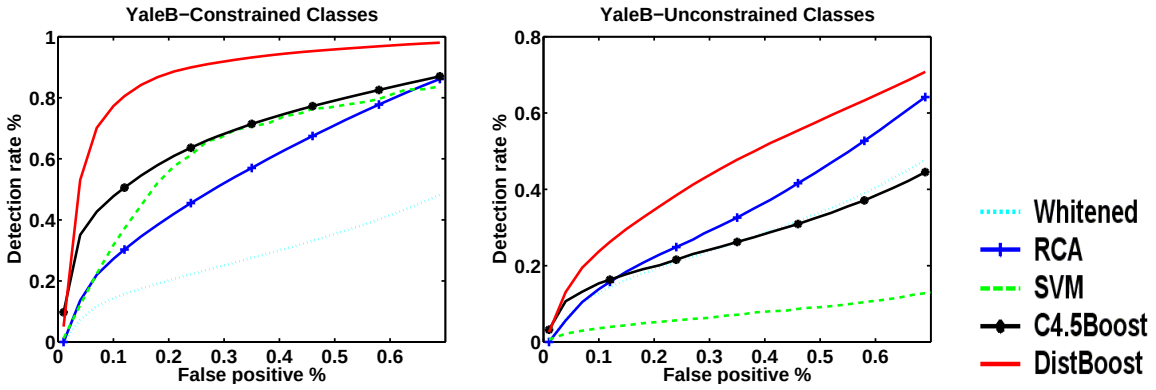

Figure 3: ROC curves of different methods on the YaleB facial image database. Left: retrieval performance on classes on which constraints we are provided. Right: retrieval performance on classes on which no constraints were provided. Results were averaged over 80 realizations

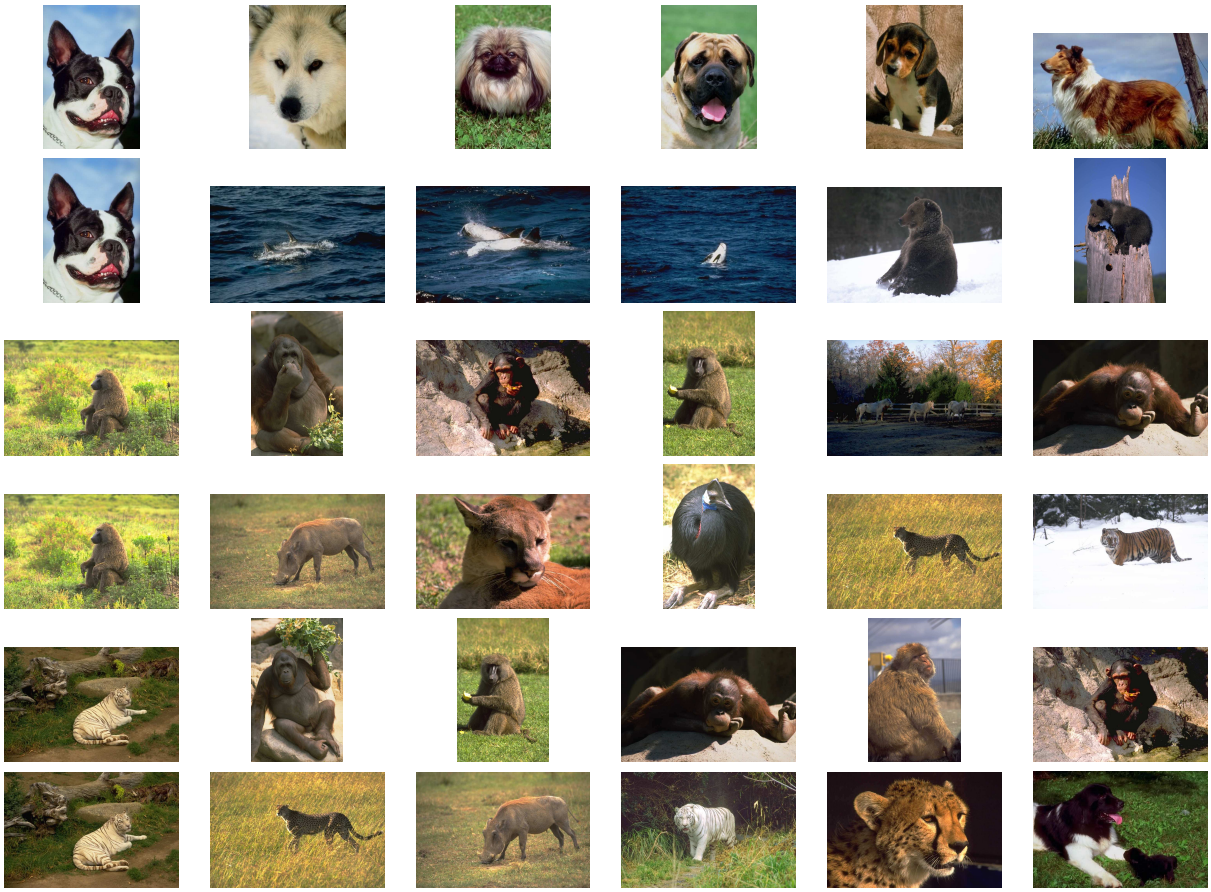

Figure 4: Typical retrieval results on the Animal image database. Each row presents a query image and its first 5 nearest neighbors comparing DistBoost and normalized $L1$ CCV distance (baseline measure). Results appear in pairs of rows: Top row: DistBoost results, Bottom row: normalized $L1$ CCV distance. Results are best seen in color.

the database contained 16 classes with 565 images, and 600 'clutter' images from unrelated classes were added to the data base. The clutter included non-animal categories, such as 'landscapes' and 'buildings'.

The original images were heavily compressed jpg images. The images were represented using Color Coherence Vectors [2] (CCV's). This representation extend the color histogram, by capturing some crude spatial properties of the color distribution in an image. Specifically, in a CC vector each histogram bin is divided into two bins, representing the number of 'Coherent' and 'Non-Coherent' pixels from each color. 'Coherent' pixels are pixels whose neighborhood contains more than τ neighbors which have the same color. Following [2] we represented the images in HSV

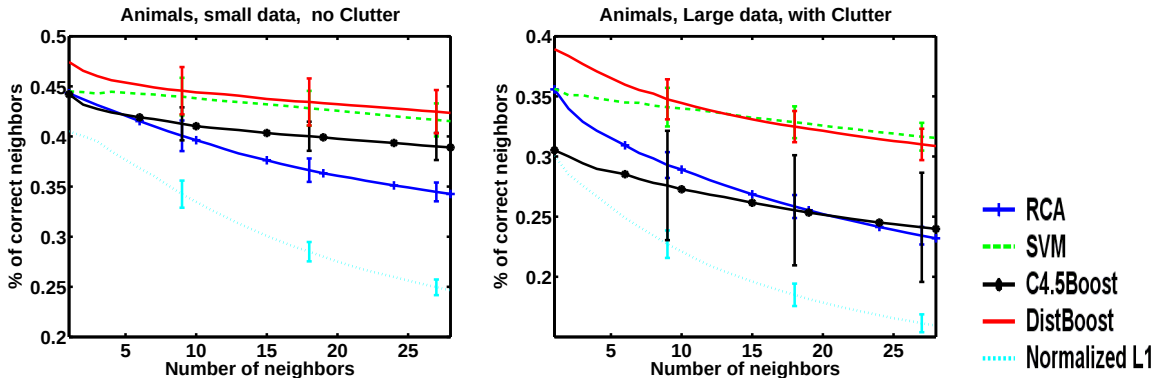

Figure 5: Neighbor purity results on color animal database. Left: results when on a database of 10 classes, 405 images and no clutter. Right: results with 16 classes, 565 images and 600 clutter images. The clutter was added to the database after the training stage. Results were averaged over 50 realizations

color space quantized the images to $4 * 2 * 4 = 32$ color bins, and computed the CCV of each image - a 64 dimensional vector - using $\tau = 25$.

Fig. 5 shows neighbor purity plots of all different distance learning methods. As our baseline measure, we used the normalized $L1$ distance measure suggested in [2]. Clearly the *DistBoost* algorithm and our product space SVM methods outperformed all other distance learning methods. The C4.5Boost performs less well, and it suffers from a relatively high degradation in performance when the task becomes harder. Retrieval results are presented in Fig 4 for our proposed *DistBoost* algorithm (Top row) and for the baseline normalized $L1$ distance over CCV's (bottom row). As can be seen our algorithm seems to group images which do not appear trivially similar in CCV space.

References

[1] Cox, I.J., Miller, M.L., Minka, T.P., Papathornas, T.V., Yianilos, P.N. The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments. In *IEEE Tran. On Image Processing*, Volume 9, Issue 1, pp. 20-37, Jan. 2000.

[2] Greg Pass and Ramin Zabih and Justin Miller. Comparing Images Using Color Coherence Vectors. In *ACM Multimedia*, 65-73, 1996.

[3] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall. Learning Distance Functions using Equivalence Relations. In *Proc. of ICML*, 2003.

[4] A. Bar-hillel , D. Weinshall Learning with Equivalence Constraints, and the relation to Multiclass Classification In *Proc. of COLT* , 2003

[5] F. d'Alche-Bue, Y. Grandvalet, and C. Ambroise. Semi supervised margin boost. in Proc. of Nips, 2001.

[6] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *J. Royal Statistical Society*, B, 39, 1-38, 1977.

[7] Georghiades, A.S. and Belhumeur, P.N. and Kriegman, D.J., From Few To Many: Generative Models For Recognition Under Variable Pose and Illumination", IEEE Int. Conf. on Automatic Face and Gesture Recognition, page 277-284, 2000.

[8] K. Fukunaga. Introduction to statistical pattern recognition. Academic press, 1990.

[9] T. Hertz, N. Shental, A. Bar-Hillel, and D. Weinshall. Enhancing Image and Video Retrieval: Learning via Equivalence Constraints. In Proc. of CVPR, 2003.

[10] D. G. Lowe. Simlarity metric learning for a variable-kernel classifier. Neural Computation 7:72-85, 1995.

[11] R. E. Schapire and Y. Singer Improved Boosting Algorithms using Confidence-Rated Predictions. In *Proc. of COLT* 1998 , 80-91.

[12] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. of ICML* 1997, 322-330.

[13] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *Proc. of ECCV*, Copenhagen, 2002.

[14] K. Tieu and P. Viola. Boosting image retrieval. In Proc. of CVPR, 2000.

[15] http://www.ics.uci.edu/ mlearn/MLRepository.html

[16] V. Vapnik Statistical learning theory Wilay, Chichester, GB 1998.

[17] E. P. Xing, A. Y. Ng, M. I. Jordan and S. Russell. Distance Metric learning with application to clustering with side-information. In *Proc. of NIPS*, 2002.

[18] P. N. Yianilos. Metric learning via normal mixtures. *NECRI TR*, 1995.

[19] Y. Grandvalet, F. d'Alche-Bue, and C. Ambroise. Boosting mixture models for semi supervised learning in ICANN 2001,Vienne, Austria, 41-48, Springer 2001.

Chapter 5

The KernelBoost Algorithm - Learning Kernel Functions from Small Training Samples

This chapter includes the following publications:

[D] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall, **Learning a Kernel Function for Classification with Small Training Samples** *in the 23rd International Conference on Machine Learning*, Pittsburgh, Pennsylvania, June 2006.

Learning a Kernel Function for Classification with Small Training Samples

Tomer Hertz TOMBOY@CS.HUJI.AC.IL
Aharon Bar Hillel AHARONBH@CS.HUJI.AC.IL
Daphna Weinshall DAPHNA@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Center for Neural Computation,
The Hebrew University of Jerusalem, Jerusalem, Israel 91904.

Abstract

When given a small sample, we show that classification with SVM can be considerably enhanced by using a kernel function learned from the training data prior to discrimination. This kernel is also shown to enhance retrieval based on data similarity. Specifically, we describe *KernelBoost* - a boosting algorithm which computes a kernel function as a combination of 'weak' space partitions. The kernel learning method naturally incorporates domain knowledge in the form of unlabeled data (i.e. in a semi-supervised or transductive settings), and also in the form of labeled samples from relevant related problems (i.e. in a learning-to-learn scenario). The latter goal is accomplished by learning a *single* kernel function for all classes. We show comparative evaluations of our method on datasets from the UCI repository. We demonstrate performance enhancement on two challenging tasks: digit classification with kernel SVM, and facial image retrieval based on image similarity as measured by the learnt kernel.

1. Introduction

Learning from small samples is an important problem, where machine learning tools can in general provide very few guarantees. This problem has received considerable attention recently in the context of object recognition and classification (see for example (Li et al., 2004)). Successful generalization from a very small number of training samples often requires the introduction of a certain 'hypotheses space bias' (Baxter, 1997) using additional available information. One such source of information may be unlabeled data, leading to semi-supervised or transductive learning (Chapelle et al., 2006). Another possible source of information is to use labeled samples from related problems, and try to achieve "inter-class transfer", also known as "learning to learn". "Learning to learn" may be expected when there is some shared within-class structure between various classes. The idea is to learn from very small samples by making use of information provided from other related classes, for which sufficient amounts of labeled data are present. There are a number of different methods which have been previously suggested for exploiting the shared structure between related classes (Thrun & Pratt, 1998). These include the selection of priors (Baxter, 1997), hierarchical modeling, and learning transformations between class instances (Sali & Ullman, 1998; Ferencz et al., 2005; Miller et al., 2000).

In this paper, we suggest to learn distance functions, and show that such functions can provide a plausible alternative for transferring inter-class structure. In particular, we describe *KernelBoost* - an algorithm that learns non-parametric kernel functions. These kernels can then be used for classification with kernel SVM. They can also be used directly for retrieval based on similarity (as measured by the kernel). The algorithm is semi-supervised and can naturally handle unlabeled data. The direct input of the algorithm is actually equivalence constraints - relations defined on pairs of data points that indicate whether the pair belongs to the same class or not. When provided with labeled data, such constraints may be automatically extracted from the labels.

The learning algorithm we suggest is based on boosting. In each round, the weak learner computes a Gaussian Mixture Model (GMM) of the data using some of the equivalence constraints and weights on the labeled and unlabeled data points. The mixture is optimized using EM to find a partition which complies with the data density, as well as with the equivalence constraints provided. A 'weak kernel' hypothesis, defined over pairs of points, is then formed based on the probability that the two points originate from the same cluster in the learnt model. The boosting process accumulates a weighted linear combination of such 'weak kernels', which define the final kernel. Note that this final kernel is a function, defined for any pair of data points. Details are presented in Sec. 2.

Appearing in *Proceedings of the 23^{rd} International Conference on Machine Learning*, Pittsburgh, PA, 2006.

We test our proposed algorithm both on classification and retrieval tasks. We first tested the algorithm without using any additional domain knowledge on several UCI datasets (see Sec. 3). We then present results on the task of classifying digit images, and on facial image retrieval using additional domain knowledge (see Sec. 4). Both tasks were selected because there are good reasons to expect some form of "inter-class-transfer" between the different classes (digits or faces). In the classification tasks, the kernel function is used in a standard 'soft margin' SVM algorithm. Multiclass problems are addressed using the 'all-pairs' Error-Correcting Output Codes (ECOC) technique (Dietterich & Bakiri, 1995), in which the full set of binary classifiers over pairs are combined to form an *m*-class classifier. In order to try and make use of the relatedness of these binary classification problems, a **single** kernel function is trained on the entire *m*-class training set. This single kernel function is used (up to truncation as described in Sec. 2.4), by all of the pairwise binary classifiers trained.

In an image retrieval task, the system is presented with a query image and is required to return the images in the database that are most similar to the query image. Performance therefore relies on the quality of the similarity function used to retrieve images. The similarity measure can be hand-defined, or learnt using a set of labeled training images. Ultimately a good similarity function could be trained on a set of labeled images from a small set of classes, and could then be used to measure similarity between images from novel classes. In general, this is a very challenging and currently unsolved problem. However, as we show, on the more specific task of facial image retrieval, our proposed algorithm learns a similarity function which also generalizes well to faces of subjects who were not presented during training at all.

1.1. Related Work

There is a growing literature on the learning of distance functions and kernels, two problems that are typically treated quite differently. For example, learning a Mahalanobis metric from equivalence constraints is discussed in (Xing et al., 2002; Bar-Hillel et al., 2005), while *DistBoost* (Hertz et al., 2004) uses boosting to learn generative distance functions which are not necessarily metric. The question of how to learn a new kernel from a set of existing kernels and a training set of labeled data is discussed in a number of recent papers, for example, (Cristianini et al., 2002; Zhang et al., 2006; Lanckriet et al., 2002; Crammer et al., 2002; Ong et al., 2005). Finally, learning of kernel functions in the context of learning-to-learn is discussed in (Yu et al., 2005).

We note however, that most of these kernel learning methods learn a kernel matrix (rather than a function), and therefore typically use the transductive framework which makes it possible to learn a kernel matrix over the set of all data, train and test. Without making the transductive assumption, most earlier methods have dealt with the estimation of kernel parameters. Our method, on the other hand, learns a non-parametric kernel function defined over all pairs of data points. The proposed method is semi-supervised and can also make use of unlabeled data (which may not necessarily come from the test set).

2. *KernelBoost*: Kernel Learning by Product Space Boosting

KernelBoost is a variant of the *DistBoost* algorithm (Hertz et al., 2004) which is a semi-supervised distance learning algorithm that learns distance functions using unlabeled datapoints and equivalence constraints. While the *DistBoost* algorithm has been shown to enhance clustering and retrieval performance, it was never used in the context of classification, mainly due to the fact that the learnt distance function is not a kernel (and is not necessarily metric). Therefore it cannot be used by the large variety of kernel based classifiers that have shown to be highly successful in fully labeled classification scenarios. *KernelBoost* alleviates this problem by modifying the weak learner of *DistBoost* to produce a 'weak' kernel function. The 'weak' kernel has an intuitive probabilistic interpretation - the similarity between two points is defined by the probability that they both belong to the same Gaussian component within the GMM learned by the weak learner. An additional important advantage of *KernelBoost* over *DistBoost* is that it is not restricted to model each class at each round using a single Gaussian model, therefore removing the assumption that classes are convex. This restriction is dealt with by using an adaptive label dissolve mechanism, which splits the labeled points from each class into several local subsets, as described in Sec. 2.5. An important inherited feature of *KernelBoost* is that it is semi-supervised, and can naturally accommodate unlabeled data in the learning process. As our empirical results show, the ability to use unlabeled data in the training process proves to be very important when learning from small samples.

2.1. The *KernelBoost* Algorithm

Let us denote by $\{x_i\}_{i=1}^n$ the set of input data points which belong to some vector space $\mathcal{X}$, and by $\mathcal{X} \times \mathcal{X}$ the "product space" of all pairs of points in $\mathcal{X}$. An equivalence constraint is denoted by (x_{i_1}, x_{i_2}, y_i), where $y_i = 1$ if points (x_{i_1}, x_{i_2}) belong to the same class (positive constraint) and $y_i = -1$ if these points belong to different classes (negative constraint). $(x_{i_1}, x_{i_2}, *)$ denotes an unlabeled pair.

The algorithm makes use of the observation that equivalence constraints on points in $\mathcal{X}$ are binary labels in the

Algorithm 1 The *KernelBoost* algorithm.

Input:

Data points: $(x_1, ..., x_n)$, $x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points: $(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)$ $i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)
  $w_k = 1/n$ $k = 1, \ldots, n$ (weights over data points)
- For $t = 1, .., T$
  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak kernel function $K_t : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$ and define a weak hypothesis as $\tilde{K}_t(x_i, x_j) = 2K_t(x_i, x_j) - 1 \in [-1, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i \tilde{K}_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2} \ln(\frac{1+r_t}{1-r_t})$.
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:

$$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{K}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\lambda * \alpha_t) & y_i = * \end{cases}$$

  where λ is a tradeoff parameter that determines the decay rate of the unlabeled points in the boosting process.
  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final Kernel function of the form $K(x_i, x_j) = \sum_{t=1}^{T} \alpha_t K_t(x_i, x_j)$.

product space, $\mathcal{X} \times \mathcal{X}$. Thus, by posing the problem in product space the problem is transformed into a classical binary classification problem, for which an optimal classifier should assign $+1$ to all pairs of points that come from the same class, and -1 to all pairs of points that come from different classes [1]. The weak learner itself is trained in the original space $\mathcal{X}$, which allows it to make use of unlabeled data points in a semi-supervised manner. The weak learner is then used to generate a "weak kernel function" on the product space.

The *KernelBoost* algorithm (described in Alg. 1 above) learns a Kernel function of the following form:

$$K(x_1, x_2) = \sum_{t=1}^{T} \alpha_t K_t(x_1, x_2) \quad (1)$$

which is a linear combination of "weak kernel functions" K_t with coefficients α_t. The algorithm uses an augmentation of the 'Adaboost with confidence intervals' algorithm (Schapire & Singer, 1999) to incorporate unlabeled data into the boosting process. More specifically, given a partially labeled dataset $\{(x_{i_1}, x_{i_2}, y_i)\}_{i=1}^{N}$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis which minimizes the following loss function:

$$\sum_{\{i | y_i = 1, -1\}} \exp(-y_i K(x_{i_1}, x_{i_2})) \quad (2)$$

Note that this semi-supervised boosting scheme computes the weighted loss only on **labeled** pairs of points but updates the weights over **all** pairs of points. The unlabeled points serve as a prior on the data's density, which effectively constrains the parameter space of the weak learner in the first boosting rounds, giving priority to hypotheses which both comply with the pairwise constraints and with the data's density. In order to allow the algorithm to focus on the labeled points as the boosting process advances, the weights of the unlabeled points decay in a rate which is controlled by a tradeoff parameter λ and by the weight of each boosting round α_t (see Alg. 1 step 5). Throughout all experiments reported in this paper, λ was set to 10.

2.2. *KernelBoost*'s Weak Learner

KernelBoost's weak learner is based on the constrained Expectation Maximization (cEM) algorithm (Shental et al., 2003). The algorithm uses unlabeled data points and a

[1] Also referred to as the *ideal* kernel (Cristianini et al., 2002).

set of equivalence constraints to find a Gaussian Mixture Model (GMM) that complies with these constraints.

At each iteration t, the cEM algorithm's uses a set of unlabeled points $X = \{x_i\}_{i=1}^{n}$, and a set of pairwise constraints (Ω) over these points, in order to learn a Gaussian mixture model with parameters $\Theta^t = \{\pi_k^t, \mu_k^t, \Sigma_k^t\}_{k=1}^{M^t}$. We denote positive constraints by $\{(p_j^1, p_j^2)\}_{j=1}^{N_p}$ and negative constraints by $\{(n_k^1, n_k^2)\}_{k=1}^{N_n}$. Let $L = \{l_i\}_{i=1}^{n}$ denote the hidden assignment of each data point x_i to one of the Gaussian sources ($l_i \in \{1, \ldots, M\}$). The constrained EM algorithm assumes the following joint distribution of the observables X and the hiddens L:

$$p(X, L|\Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^{n} \pi_{l_i} p(x_i|\theta_{l_i}) \prod_{j=1}^{N_p} \delta_{l_{p_j^1} l_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{l_{n_k^1} l_{n_k^2}}) \quad (3)$$

where Z is the normalizing factor and δ_{ij} is Kronecker's delta. The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (3) with respect to L. For a more detailed description of this weak learner see (Shental et al., 2003).

2.3. Generating a Weak Kernel from a GMM

Given the mixture Θ^t at round t, we construct a 'weak kernel' which essentially measures the probability that two points belong to the same Gaussian component. Denoting the hidden label of a point according to the mixture by $l(x)$, the kernel is given by

$$\begin{aligned} K_t(x_1, x_2) &= p[l(x_1) = l(x_2)|\Theta] \quad (4) \\ &= \sum_{j=1}^{M^t} p(l(x_1) = j|\Theta) p(l(x_2) = j|\Theta) \end{aligned}$$

where $p[l(x) = j|\Theta] = \frac{\pi_j G(x|\mu_k, \Sigma_k)}{\sum_{k=1}^{M^t} \pi_k G(x|\mu_k, \Sigma_k)}$, and $G(x|\mu, \Sigma)$ denotes the Gaussian probability with parameters μ and Σ.

This "weak kernel" is bounded in the range $[0, 1]$. The weak hypothesis required for updating the sample weights in the boosting process is created by applying the linear transformation $2K(x_1, x_2) - 1 \in [-1, 1]$ to the 'weak kernel'. Note that the final combined kernel is a linear combination of the "weak kernels" (and not the weak hypotheses) in order to ensure positive definiteness.

2.4. Adapting the Learned Kernel Function

As noted above, *KernelBoost* can learn a single kernel function over a multi-class dataset, which can then be used to train both binary classifiers and an *m-class* classifier. When training a binary classifier between any subset of labels from the data, we adapt the learned kernel function. More specifically, we consider all 'truncated' kernel combinations, i.e kernels that are a truncation of the full learned kernel up to some $t' \leq T$. In order to select the optimal truncated kernel for a given binary classification problem, we use the empirical kernel alignment score suggested by (Cristianini et al., 2002) between the learned kernel and the 'ideal' kernel ($K_{ideal} = yy'$) which is given by

$$Alignment(K, S) = \frac{\langle K, K_{ideal} \rangle_F}{\sqrt{\langle K, K \rangle_F \langle K_{ideal}, K_{ideal} \rangle_F}}$$

where $S = (x_i, y_i)$ is the training sample, y denotes the vector of point labels and $\langle . \rangle_F$ denotes the Frobenius product . This score is computed for $t = 1 \ldots T$ and the kernel with the highest score on the training data is selected.

2.5. The Label Dissolving Mechanism

The weak learner of the *KernelBoost* algorithm treats all constraints as hard constraints; in particular, since all positive constraints are always satisfied in the cEM algorithm, its only option is to attempt to place all of the points from the same label in a single Gaussian at every iteration. This is very problematic for non-convex classes generated by non-Gaussian distributions (see Fig. 1). Therefore, in order to enrich the expressive power of *KernelBoost* and to allow it to model classes of these types, the algorithm is augmented by a label-dissolving mechanism, which relies on the boosting weights. This mechanism splits sets of points with the same label into several local subsets, which allows the algorithm to model each of these subsets separately, using a different Gaussian model.

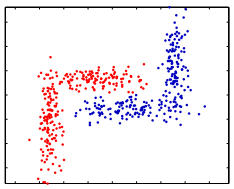

Figure 1. A 2-d synthetic example of a non-convex and non-Gaussian dataset.

The intuition leading to the proposed mechanism is the following: We would like to model each non-convex class, using several local Gaussians. The attempt to model a highly non-Gaussian, or non-convex class using a single Gaussian, will fail, and cause some of the pairwise constraints to be unsatisfied. The boosting process focuses each new weak learner on those harder pairs still inconsistent with the current hypothesis. The adaptive dissolve mechanism uses these pairwise weights to eliminate edges already consistent with the current hypothesis from a local neighborhood graph. Classes are therefore split into small local subsets. The dissolve mechanism proposed is presented below in Alg. 2.

This mechanism has one tunable parameter N_{mutual}, which determines the pre-computed neighborhood graph for each of the labels[2]. This parameter implicitly affects

[2]Neighbors are defined as "mutual" iff i is within the first N

Algorithm 2 The adaptive label-dissolve mechanism.

Preprocess: For each label l, compute a local neighborhood graph where each labeled datapoint is connected to all of its mutual neighbors from the first N_{mutual} neighbors.

For $t = 1 \dots T$ do

For each label l do

1. Define the edge weights on the graph to be the pairwise weights $W^t_{i_1,i_2}$ computed by the boosting process.
2. Threshold edges by removing all edges whose weight is smaller then the average edge weight given by $\frac{1}{|l|}\sum_{(i_1,i_2)\in l} W^t_{i_1,i_2}$.
3. Compute the connected components of the graph and use them to define a partition of the labels from the current class into small and local subsets.

the number of subsets obtained at each boosting round.

2.6. The Kernel's Implicit Representation

The substitution of Equation (4) into (1) yields the structure of the learnt kernel:

$$K(x_1,x_2) = \sum_{t=1}^{T}\sum_{k=1}^{M^t} \sqrt{\alpha_t} p[l(x_1)=k|\Theta^t] \cdot \sqrt{\alpha_t} p[l(x_2)=k|\Theta^t] \quad (5)$$

If we think of each element in the sum in Equation (5) as a feature in a feature-space of dimension $\sum_{t=1}^{T} M^t$, then the coordinate corresponding to the pair (t,k) holds a feature of the form

$$\Phi_{t,k}(x) = \sqrt{\alpha_t}\frac{\pi_k G(x|\mu^t_k,\Sigma^t_k)}{\sum_{j=1}^{M^t}\pi_j G(x|\mu^t_j,\Sigma^t_j)} \quad (6)$$

These features can be interpreted as soft Voronoi cell indicators: a high value for feature $\Phi_{t,k}$ indicates that the point lies in cell k of the partition induced by mixture t. These features are rather different from the prototype-like RBF features. Specifically, their response does not necessarily decay rapidly with the distance from the Gaussian's center. Decay only happens in regions where other Gaussians from the same mixture are more likely.

3. Experiments: Learning from Small Samples

3.1. Visualization using $2D$ Synthetic Datasets

We begin by returning to the non-convex, and non-Gaussian dataset presented in Fig. 1. Each class in this dataset was created using two Gaussians. We compared the performance of *KernelBoost* to several standard kernels. More specifically, we compared the following kernels: (1) KB - *KernelBoost*, (2) KB-dis - *KernelBoost* which includes the label dissolving mechanism described above, (3) the linear kernel, (4) the polynomial kernel of degree 2, (5) the RBF kernel (with σ chosen using cross-validation)

The dataset contains 500 datapoints. In our experiment we randomly selected N_{train} datapoints for training (where $N_{train} = 20$ or 100) and used the remaining datapoints for testing. We uniformly set the SVM tradeoff parameter C to 5 in all these experiments. Each of the two experiments was repeated for 10 random train-test data splits. *KernelBoost* was run for 10 boosting iterations.

Table 1. A comparison of classification accuracy on the non-convex and non-Gaussian dataset shown in Fig. 1. Best Results are highlighted in bold.

N_{train}	KB	KB-dis	Linear	Poly.	RBF
20	17.5	**4.5**	12.0	13.1	6.3
100	17.9	**0.9**	10.4	10.5	1.9

The results are reported in Table 1. As may be seen, *KernelBoost* with the dissolve mechanism outperforms all other kernels on this dataset for both small and large samples. Using the label dissolving mechanism suggested above, *KernelBoost* can generate general non-convex separators, as can be seen from the results in Table 1. Fig. 2 shows the learnt Gaussians and the separating hyper-plane induced by the learnt kernel in a typical experiment on this dataset.

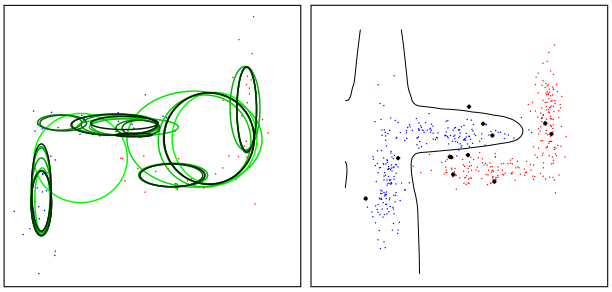

Figure 2. **Left:** The Gaussians learnt by *KernelBoost*-dissolve (presented in Sec. 2.5). The Ellipses mark 1-std contours. Darker ellipses show Gaussians obtained at later boosting rounds. **Right:** The separator induced by the Gaussians for this example. Support vectors are marked by black dots.

3.2. Results on UCI Datasets

We now turn to evaluate the performance of our algorithm on several real datasets from the UCI repository, and also compare its performance to some standard 'off the shelf' kernels.

neighbors of j and vice-versa.

Experimental setup We used 4 datasets from the UCI data repository (Blake & Merz, 1998): wine, ionosphere breast cancer and balance. These experiments were conducted in a transductive setting, i.e. the test data was presented to the *KernelBoost* algorithm as unlabeled data. We used 10% of the data as training sample. For each of these conditions we compare our method with some standard 'off-the-shelf' kernels, and report the best results of (Zhang et al., 2006) on the same experimental setup. The results reported are averages over 10 random train/test splits. In all of these experiments the SVM tradeoff parameter was set to 300. The dissolve neighborhood parameter N_{mutual} was set to 12, and we used $T = 30$ boosting rounds. The results may be seen in Table 2. KernelBoost outperforms other methods on 3 of the 4 datasets.

Table 2. A comparison of classification accuracy of various kernels on 4 UCI datasets. In this experiment 10% of the data was used both for learning the kernel and training the SVM classifier. Results were averaged over 10 different realizations of train and test data. Best results are highlighted in bold.

Data	KB	KB-dis	Lin.	Poly.	RBF	Zhang
wine	95.1	**95.4**	91.9	78.3	90.8	94.6
ionos.	85.9	**90.4**	79.7	72.3	84.5	87.6
breast	94.2	92.6	94.8	93.9	**95.7**	94.6
balance	83.5	**86.4**	84.4	77.4	85.0	—

4. Experiments: Learning to Learn

4.1. MNIST Digit Classification

Various different classification methods have been used on the MNIST dataset, some of which providing almost perfect results (LeCun et al., 1998). However, these methods were all trained and tested on very large samples of training data (usually on the entire training set which consists of $60,000$ datapoints). Since we are interested in testing inter-class transfer, we conducted a set of experiments in which a very limited amount of training data was used.

Experimental setup: In these experiments, we randomly selected 1000 sets of 4 digits from the dataset, and used 5 different train/test splits for each set. For each set of 4 digits, we further split the classes into 2 pairs: one pair was designated to provide large amounts of data for training, while the other pair was designated to provide a very small amount of training data. For each 4-tuple, we considered all 6 possible splits into pairs. For the designated 'large' classes we randomly selected 100 datapoints as train data. For the designated 'small' classes, we randomly selected k labeled points from each class, where $k = 3, 4, 5, 6, 10$ and 20. Additionally, for each of the 4 digits we randomly selected 200 datapoints which were supplied to the learning algorithm as unlabeled data . *KernelBoost* used the training data from all 4 classes to learn a **single** kernel function. Predictions were evaluated on a test set of 200 points from each class, which were not presented during the training stage. Images were represented using the first 30 PCA coefficients of the original vectorized images. The SVM tradeoff parameter C was set to 300, T was set to 10 and the N_{mutual} parameter was uniformly set to 12.

After learning the kernel function, we trained SVM binary classifiers for all 6 digit pairs. As a baseline comparison, we also trained SVM binary classifiers using standard 'off-the-shelf' kernels for all the pairs. We compared our algorithm to the following standard kernels: linear, RBF and a polynomial kernel of degree 5 (which has been shown to perform well on the MNIST dataset (LeCun et al., 1998)). The binary SVM's were also used to create a multiclass classifier using the 'all-pairs' Error Correcting Output Codes (ECOC) scheme (Dietterich & Bakiri, 1995).

These 6 binary classification problems (for each 4 digits) can be divided into 3 subgroups, to allow a more detailed analysis of the effects of "inter-class-transfer":

1. 'small vs. small' - the single binary classifier trained on the two classes for which a very small amounts of labeled points was present (k).
2. 'small vs. large' - the 4 binary classifiers which were trained on two classes, where one had a large amount of labeled points (100) and the other had a very small amount of labeled points (k).
3. 'large vs. large' - the single binary classifier trained on the two classes for which large amounts (100) of labeled data was present.

From these three types, the first two may benefit from inter-class transfer. Clearly the hardest binary classification problem is the 'small vs. small' one in which the total amount of datapoints is only $2k$. However, the 4 'small vs. large' binary problems are also very challenging.

Classification results: The results on the MNIST classification tasks when using $k = 3$ labeled points for the small classes are presented in Table 3. These results demonstrate a clear advantage of *KernelBoost* over all other standard kernels in this difficult classification task. Specifically, in the challenging 'small vs. small' condition, both KernelBoost variants obtain significantly better accuracy scores over all other kernels. Note that the performance of the KB-dis variant is always superior to that of the original KB method. In the 'small vs. large' condition, both *KernelBoost* variants achieve excellent performance with an improvement of roughly 15% in test accuracy over all other kernels. Finally, as expected, in the 'large vs. large' con-

Table 3. A comparison of median classification accuracy of KernelBoost with various other standard kernels on randomly selected subsets of 4 digits from the MNIST dataset. In this experiment the amount of labeled points for the 'small' classes k was 3. Best Result are highlighted in bold.

Type	Kboost	KboostDis	Lin.	Poly(5)	RBF
'small vs. small'	$84.80(\pm 0.40)$	$\mathbf{85.01(\pm 0.46)}$	$81.7(\pm 0.31)$	$56.45(\pm 0.37)$	$79.20(\pm 0.33)$
'small vs. large'	$\mathbf{92.90(\pm 0.13)}$	$89.60(\pm 0.21)$	$72.70(\pm 0.17)$	$51.10(\pm 0.13)$	$77.60(\pm 0.19)$
'large vs. large'	$96.70(\pm 0.15)$	$96.40(\pm 0.23)$	$96.50(\pm 0.10)$	$\mathbf{97.90(\pm 0.08)}$	$97.70(\pm 0.08)$
'ECOC' (multi-class)	$\mathbf{79.30(\pm 0.23)}$	$72.33(\pm 0.27)$	$64.90(\pm 0.22)$	$50.35(\pm 0.17)$	$67.83(\pm 0.23)$

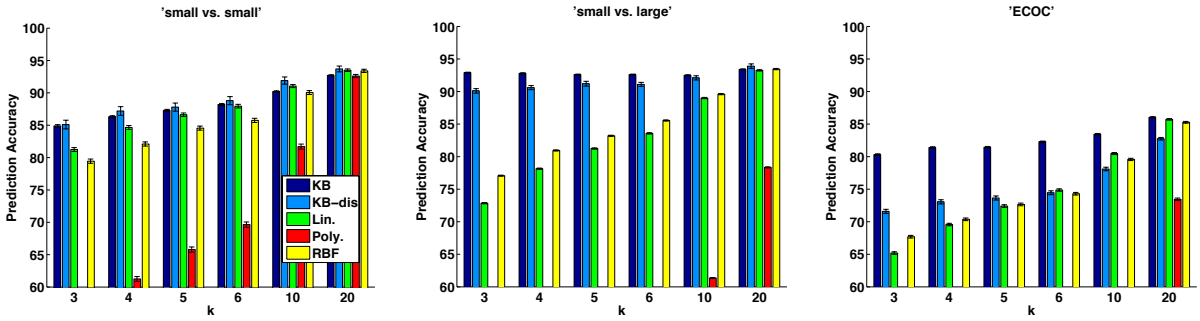

Figure 3. Median classification accuracy on the MNIST dataset as a function of the number of labeled points. Methods compared are: KB - *KernelBoost*, KB-dis - *KernelBoost* with the label dissolve mechanism, Lin. - linear kernel, Poly. - polynomial kernel of degree 5, and RBF - RBF kernel with σ chosen using cross-validation. **Left**: Results on the 'small vs. small' classes. **Middle**: Results on the 'small vs. large' classes. **Right**: multi-class classification results. When the Polynomial kernel is not shown its accuracy $\leq 60\%$.

dition all of the algorithms obtain almost perfect classification accuracy, and the polynomial kernel of degree 5 achieves the best performance. Note however, that on all other tasks the polynomial kernel performs poorly, which implies serious overfitting. Another clear advantage of the *KernelBoost* algorithm is shown in the multi-class classification task, where its performance is significantly better than all other methods.

It is interesting to analyze the results on these 4 classification tasks as the number of labeled points k increases, as shown in Fig. 3. Clearly, as the amount of labeled data increases, the performance of all kernels improves, but *KernelBoost* still maintains a significant advantage over its competitors.

4.2. Facial Image Retrieval Results

In the previous section we have shown that *KernelBoost* makes use of interclass transfer on digits from the MNIST dataset. We now turn to another relevant application of facial image retrieval.

Experimental setup: We used a subset of the YaleB facial image dataset (Georghiades et al., 2000). The dataset contains a total of 1920 images, including 64 frontal pose images of 30 different subjects. In this database the variability between images of the same person is mainly due to different lighting conditions. The images were automatically centered using optical flow. Images were then converted to vectors, and each image was represented using its first 9 Fisher Linear Discriminant coefficients, which were computed over the first 150 PCA coefficients. *KernelBoost* was run for 10 boosting iterations, with a Gaussian Mixture model with a single (shared) covariance matrix. On this dataset, we conducted three experiments:

1. 'Fully supervised' - in which we randomly selected images from 20 of the subjects. We used 50% of their data as training data and tested on the remaining 50%.
2. 'Semi-supervised' setting in which we augmented the train data of experiment 1 with an additional 50% of the data from the remaining 10 classes as unlabeled data, and tested performance on the remaining 50% of the unlabeled classes.
3. 'Unsupervised' setting in which we trained the algorithm using the exact same data of exp. 1 and tested it on images from the remaining 10 classes which were not present during the training stage at all.

In the test stage of each of the experiments, the retrieval database contained images of all 30 subjects, part (or all of which) was used by the learning algorithms. For each image we compute the ROC (Receiver Operating Characteristic) curve and these ROCs are averaged to produce a single ROC curve. The fraction of the area under the curve (AUC score) is indicative of the distinguishing power of the algorithm and is used as its prediction accuracy. We compare

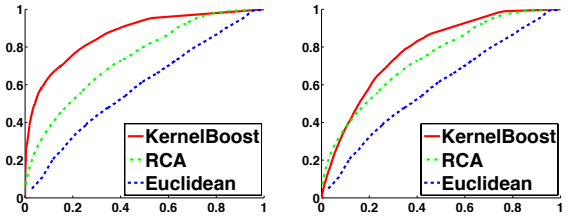

Figure 4. ROC retrieval curves on the YaleB dataset. The graphs compare the performance of KernelBoost to RCA and to the Euclidean distance metric in a "Learning to learn" scenario. Left: Results for classes for which unlabeled data was presented during the training stage. Right: Results on novel classes for which no data at all was present during training.

the performance of our method to the RCA algorithm (Bar-Hillel et al., 2005), which is a Mahalanobis metric learning algorithm that has been shown to work well on image and video retrieval (Bar-Hillel et al., 2005). As a baseline measure we also used the Euclidean metric.

Retrieval results: The results of the 3 experiments described above are presented in Fig 4, and summarized in Table 4. In the 'Fully-supervised' experiment, the KernelBoost method obtains almost perfect performance, with a clear advantage over the simpler RCA algorithm. In the 'Semi-supervised' experiment, both methods' performance degrades, but KernelBoost still performs significantly better than all other methods. In the 'Unsupervised' setting, where the test set consists of faces of individuals not seen during training, the performance degrades some more, but both algorithms still perform significantly better that the Euclidean distance metric.

Table 4. AUC scores (and ste's) for the three image retrieval experiments conducted on the YaleB dataset. See text for details.

Exp No.	KernelBoost	RCA	Euclidean
(1)	$98.94(\pm 0.01)$	$93.88(\pm 0.01)$	$60.84(\pm 0.01)$
(2)	$84.57(\pm 0.04)$	$77.37(\pm 0.03)$	$59.00(\pm 0.00)$
(3)	$79.74(\pm 0.06)$	$76.62(\pm 0.04)$	$58.92(\pm 0.01)$

5. Discussion

The main contribution of this paper lies in the description of a method for learning non-parametric kernel functions. The algorithm presented is semi-supervised (i.e., it benefits from unlabeled data), and can learn from very small samples. When used with kernel SVM, classification performance was shown to be significantly better than various standard kernels. The benefit of learning the kernel function was most evident in the context of "learning to learn", in which information is transferred to classes for which only a few examples are available for training. In future work we hope to explore the benefit of such learned kernels when combined with other kernel-based techniques.

Acknowledgments: This research was supported by the EU under the DIRAC integrated project IST-027787.

References

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2005). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research, 6(Jun)*, 937–965.

Baxter, J. (1997). A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning, 28*, 7–39.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning*. Cambridge: MIT Press. in press.

Crammer, K., Keshet, J., & Singer, Y. (2002). Kernel design using boosting. *Advances in Neural Information Processing Systems*.

Cristianini, N., Kandola, J., Elissee, A., & Shawe-Taylor, J. (2002). On kernel target alignment. *Advances in Neural Information Processing Systems*.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research, 2*, 263–286.

Ferencz, A., Learned-Miller, E., & Malik, J. (2005). Building a classification cascade for visual identification from one example. *International Conference of Computer Vision (ICCV)* (pp. 286–293).

Georghiades, A., Belhumeur, P., & Kriegman, D. (2000). From few to many: Generative models for recognition under variable pose and illumination. *Automatic Face and Gesture Recognition* (pp. 277–284).

Hertz, T., Bar-Hillel, A., & Weinshall, D. (2004). Boosting margin based distance functions for clustering. *21st International Conference on Machine Learning (ICML)*.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2002). Learning the kernel matrix with semi-definite programming. *ICML* (pp. 323–330).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278–2324.

Li, F. F., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *CVPR, Workshop on Generative-Model Based Vision*.

Miller, E., Matsakis, N., & Viola, P. (2000). Learning from one example through shared densities on transforms. *CVPR* (pp. 464–471).

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research, 6*, 1043–1071.

Sali, E., & Ullman, S. (1998). Recognizing novel 3-d objects under new illumination and viewing position using a small number of example views or even a single view. *ICCV* (pp. 153–161).

Schapire, R. E., & Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning, 37*, 297–336.

Shental, N., Hertz, T., Bar-Hilel, A., & Weinshall, D. (2003). Computing gaussian mixture models with EM using equivalence constraints.

Thrun, S., & Pratt, L. (1998). *Learning to learn*. Boston, MA: Kluwer Academic Publishers.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2002). Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems*.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. *Proceedings of the 22 nd International Conference on Machine Learning, Bonn, Germany*.

Zhang, Z., Kwok, J., & Yeung, D. (2006). Model-based transductive learning of the kernel matrix.

Chapter 6

Predicting Protein-Peptide binding by Learning Distance Functions

This chapter includes the following publications:

[E] Tomer Hertz and Chen Yanover, **PepDist: A New Framework for Protein-Peptide Binding Prediction based on Learning Peptide Distance Functions**, *in BMC Bioinformatics*, Vol. 7 (suppl 1), March 2006.

BMC Bioinformatics

Proceedings

PepDist: A New Framework for Protein-Peptide Binding Prediction based on Learning Peptide Distance Functions

Tomer Hertz*[1,2] and Chen Yanover*[1]

Address: [1]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, 91904 and [2]The Center for Neural Computation, The Hebrew University of Jerusalem, Jerusalem, Israel, 91904

Email: Tomer Hertz* - tomboy@cs.huji.ac.il; Chen Yanover* - cheny@cs.huji.ac.il

* Corresponding authors

from NIPS workshop on New Problems and Methods in Computational Biology
Whistler, Canada. 18 December 2004

Published: 20 March 2006

BMC Bioinformatics 2006, **7**(Suppl 1):S3 doi:10.1186/1471-2105-7-S1-S3

Abstract

Background: Many different aspects of cellular signalling, trafficking and targeting mechanisms are mediated by interactions between proteins and peptides. Representative examples are MHC-peptide complexes in the immune system. Developing computational methods for protein-peptide binding prediction is therefore an important task with applications to vaccine and drug design.

Methods: Previous learning approaches address the binding prediction problem using traditional margin based binary classifiers. In this paper we propose *PepDist*: a novel approach for predicting binding affinity. Our approach is based on learning peptide-peptide distance functions. Moreover, we suggest to learn a single peptide-peptide distance function over an **entire** family of proteins (e.g. MHC class I). This distance function can be used to compute the affinity of a novel peptide to any of the proteins in the given family. In order to learn these peptide-peptide distance functions, we formalize the problem as a semi-supervised learning problem with partial information in the form of equivalence constraints. Specifically, we propose to use *DistBoost* [1,2], which is a semi-supervised distance learning algorithm.

Results: We compare our method to various state-of-the-art binding prediction algorithms on MHC class I and MHC class II datasets. In almost all cases, our method outperforms all of its competitors. One of the major advantages of our novel approach is that it can also learn an affinity function over proteins for which only small amounts of labeled peptides exist. In these cases, our method's performance gain, when compared to other computational methods, is even more pronounced. We have recently uploaded the *PepDist* webserver which provides binding prediction of peptides to 35 different MHC class I alleles. The webserver which can be found at http://www.pepdist.cs.huji.ac.il is powered by a prediction engine which was trained using the framework presented in this paper.

Conclusion: The results obtained suggest that learning a *single* distance function over an entire family of proteins achieves higher prediction accuracy than learning a set of binary classifiers for each of the proteins separately. We also show the importance of obtaining information on experimentally determined non-binders. Learning with real non-binders generalizes better than learning with randomly generated peptides that are assumed to be non-binders. This suggests that information about non-binding peptides should also be published and made publicly available.

Background

Many different aspects of cellular signalling, trafficking and targeting mechanisms are mediated by interactions between proteins and peptides. In the immune system, for example, the major task of recognizing foreign pathogen proteins is mediated by interactions between Major Histocompatibility Complex (MHC) molecules and short pathogen-derived peptides (see Fig. 1). T-cells recognize these peptides only when they are bound to MHC molecules. Understanding the underlying principles of MHC-peptide interactions is therefore a problem of fundamental importance, with applications to vaccine and drug design [3]. MHC binding is a challenging problem because MHC molecules exhibit high specificity – it is estimated that each molecule only binds to 1% of all existing peptides [4]. Additionally, MHC molecules are highly polymorphic and polygenic – there are hundreds of different alleles in the entire population while each individual carries a few alleles only (up to 6 MHC class I alleles and up to 12 MHC class II alleles) [5].

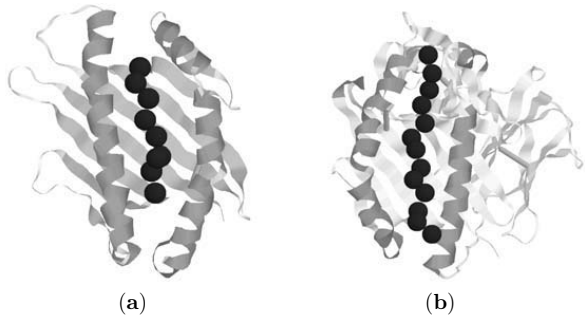

Figure 1
Schematized drawing of a peptide in the binding groove of MHC class I (**a**) and MHC class II (**b**) molecules. The peptide backbone is shown as a string of balls, each of which represents a residue.

Biochemical assays, which empirically test protein-peptide binding affinity, can nowadays provide a rather high throughput rate [6]. However, note that there are 20^L peptides of length L (for 9 amino-acid long peptides as in the MHC proteins this amounts to 10^{12} peptides) and a great number of proteins that need to be considered. Therefore, in recent years, there has been a growing interest in developing computational methods for protein-peptide binding prediction [7-13]. Formally, the protein-peptide binding prediction problem can be stated as follows: given a protein and a peptide, predict the binding affinity of the interaction between the two. Stated this way, the protein-peptide binding prediction is essentially a simplified version of the more general protein docking problem.

What should we expect from a "good" binding prediction algorithm? [8].

1. Classification: A good binding prediction algorithm should first and foremost correctly predict whether a query peptide (which was not provided during the training stage) binds or does not bind to the given protein.

2. Ranking: An even stronger requirement is that the algorithm could also obtain a relative binding score for each peptide that can be used to rank different peptides according to their specificity.

3. Affinity prediction: Ultimately, the algorithm's score would predict the precise binding affinity values as determined experimentally.

Clearly, current state-of-the-art prediction methods obtain promising *classification* results (for a recent comparison between several methods see [14]). Many of these methods also compute binding scores for each peptide, but these scores are in most cases not even compared to the empirically known affinity values, and have even been shown to have poor correspondences in some cases [15] (an interesting exception is a recent work on the PDZ domain [16]).

Most prediction algorithms formalize the protein-peptide binding prediction problem as a binary classification problem: For each protein (i.e. MHC molecule) a classifier is trained to distinguish binding peptides from non-binding peptides. After an initial training stage, the classifier is tested on a set of peptides, which were not presented during the training stage. The training data consists of experimentally determined binders and randomly generated peptides which are assumed to be non-binders. Interestingly enough, only rarely are experimentally determined non-binders used, mainly because a small number of these non-binders have been made publicly available.

In this paper we suggest a novel formulation of the protein-peptide binding prediction problem. Our approach is driven by the following two important observations:

Observation 1 *Peptides that bind to the same protein are "similar" to one another, and different from non-binding peptides.*

This observation underlies most, if not all, computational prediction methods. Motif based methods [8,13] for example, search for a binding motif that captures the similarity of a set of known binding peptides. Prediction is then based on the similarity of a query peptide to the motif, which implicitly measures the similarity of the

query peptide to the peptides in the training set. This observation suggests that given a novel peptide, one can predict its binding affinity to a specific protein, by explicitly measuring the average distance (or similarity) of the novel peptide to a list of known binding peptides. Intuitively speaking, if the peptide is close to the known binders, we would classify it as a binder, and if it is far – we would classify it as a non-binder.

Observation 2 *Peptides binding to different proteins within the same "family" resemble each other*

Proteins from the same family (e.g. MHC class I) are known to have structural and sequential similarity. Therefore they bind to peptides that share common characteristics. Additionally, in MHC class I and MHC class II, many proteins are grouped into supertypes [17] (such as the HLA-A2 MHC class I supertype). A supertype is a collection of proteins whose binding peptide sets are overlapping. This observation implies that we may benefit from *simultaneously* learning a single binding prediction function over an entire family of proteins, instead of independently learning a *single* classifier for each of the proteins within the protein family. At a first glance it might appear that one can recast a set of binary classification problems using a single multi-class classification problem. However, a closer look reveals that the protein-peptide binding problem *is not* a multi-class classification problem due to the following inherent facts: (1) Some peptides bind to several proteins within a family (indeed this information is used to define MHC supertypes). (2) A peptide that does not bind to a specific protein within a family, does not necessarily bind to a different protein within the family.

Our novel approach is based on the two observations described above. We propose to address the protein-peptide binding prediction problem by learning peptide-peptide distance functions. We do not require that the triangle inequality holds, and thus our distance functions are *not* necessarily metrics. Moreover, based on *observation 2*, we suggest to pool together information from an entire protein family and to learn a *single* peptide-peptide distance function (instead of learning a different distance function for every protein independently). Our peptide-peptide distance function is then used to compute protein-peptide binding affinity – the affinity of a query peptide to a given protein is inversely proportional to its average distance from all of the peptides known to bind to that protein. Our proposed learning scheme is summarized in Fig. 2 and elaborated in the following section.

Input:
A dataset of binding and non binding peptides from an entire protein family.

1. For each protein: Extract "positive" and "negative" equivalence constraints using its known binding and non-binding peptides, respectively.
2. Learn a single peptide-peptide distance function over this dataset using the equivalence constraints extracted in step 1.
3. Define a protein-peptide affinity function using the peptide-peptide distance function from step 2.

Output:
A *single* protein-peptide affinity function over the entire protein family.

Figure 2
The *PepDist* framework.

Learning peptide distance functions

As mentioned above, we propose to address the protein-peptide binding affinity prediction problem by learning a peptide-peptide distance function over an *entire* family of proteins. A distance function $\mathcal{D}$ assigns a non-negative value for each pair of points. Most algorithms that learn distance functions make use of equivalence constraints [1,2,18-22]. Equivalence constraints are relations between pairs of data points, which indicate whether the points in the pair belong to the same category or not. We term a constraint *positive* when the points are known to be from the same class, and *negative* in the opposite case. In this setting the goal of the algorithm is to learn a distance function that attempts to comply with the equivalence constraints provided as input.

In our setting, each protein defines a class. Each pair of peptides (data-points) which are known to bind to a specific protein (that is, belong to the same class) defines a positive constraint, while each pair of peptides in which one binds to the protein and the other does not – defines a negative constraint. Therefore, for each protein, our training data consists of a list of binding and non-binding peptides, and the set of equivalence constraints that they induce.

We collect these sets of peptides and equivalence constraints from several proteins within a protein family into a single dataset. We then use this dataset to learn a peptide-peptide distance function (see Fig. 3 left plots). Using this distance function, we can predict the binding affinity of a novel peptide to a specific protein, by measuring its average distance to all of the peptides which are known to bind to that protein (see Fig. 3 right plots). More formally, let us denote by $\mathcal{D}\left(Peptide_i, Peptide_k\right)$ the distance between $Peptide_i$ and $Peptide_k$ and by B_j the group of peptides known to bind to $Protein_j$. We define the affinity between $Peptide_i$ and $Protein_j$ to be:

$$Affinity\left(Peptide_i, Protein_j\right) \equiv \exp\left(-\frac{1}{\left|B_j\right|}\sum_{k\in B_j}\mathcal{D}\left(Peptide_i, Peptide_k\right)\right) \quad (1)$$

In order to learn peptide-peptide distance functions, we use the *DistBoost* algorithm [1,2], which learns distance functions using data and some equivalence constraints (see Methods for the algorithm's description). *DistBoost* requires that the data be represented in some continuous vector feature space. We therefore represent each amino-acid using a 5-dimensional feature vector as suggested by [23], and each peptide by concatenating its amino-acid feature vectors (for further details see the Data representation section). We compare our method to various protein-peptide affinity prediction methods on several datasets of proteins from MHC class I and MHC class II. The results show that our method significantly outperforms all other methods. We also show that on proteins for which small amounts of binding peptides are available the improve-

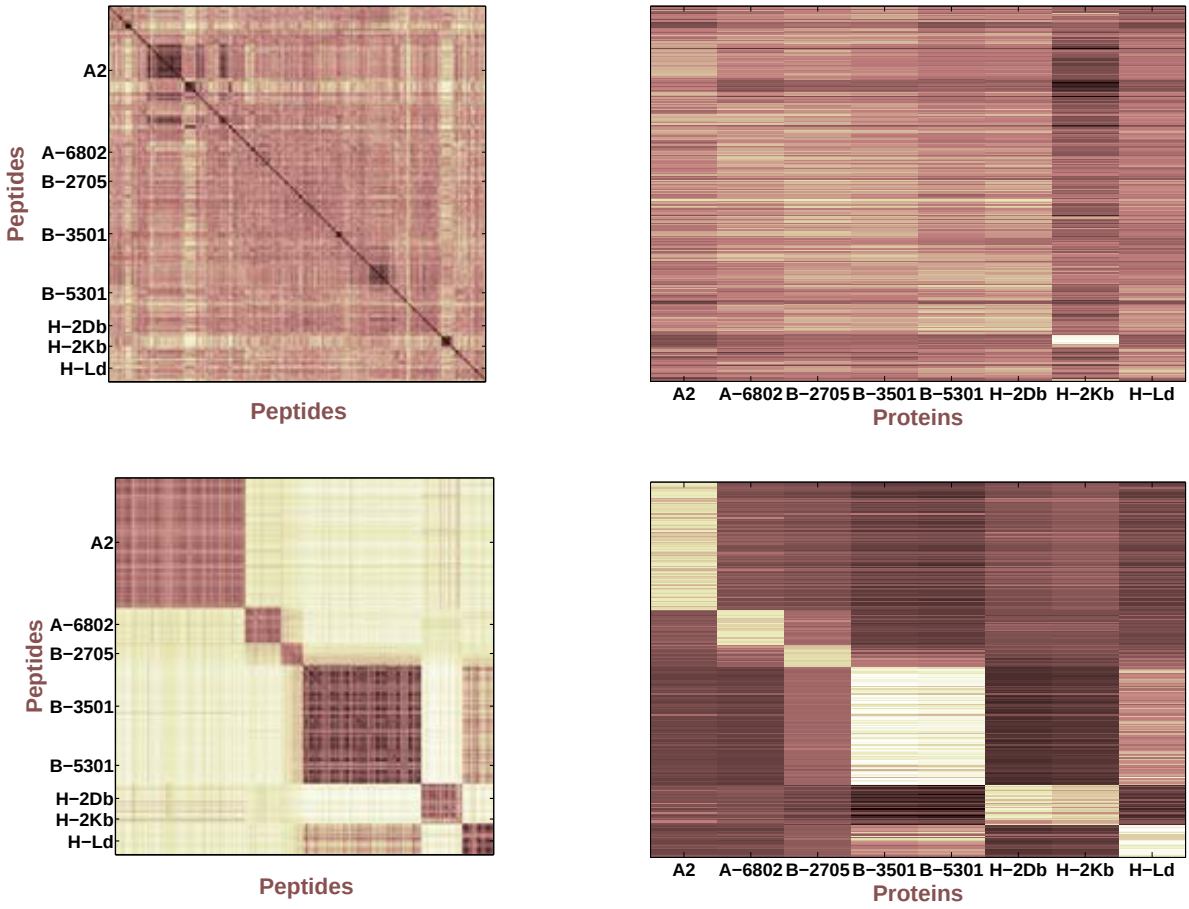

Figure 3

Left: peptide-peptide distance matrices of MHC class I binding peptides, collected from the MHCBN dataset. Peptides that bind to each of the proteins were grouped together and labeled accordingly. Following *Observation 1*, a "good" distance matrix should therefore be block diagonal. Top left: The Euclidean peptide-peptide distance matrix in $\mathbb{R}^{45}$ (see Methods for details). Bottom left: The peptide-peptide distance matrix computed using the *DistBoost* algorithm. **Right**: protein-peptide affinity matrices. The affinity between a peptide and a specific protein is computed by measuring the average distance of the peptide to all peptides known to bind to that protein (see eq. 1). Top right: the Euclidean affinity matrix. Bottom right: the *DistBoost* affinity matrix. *DistBoost* was trained on binding peptides from all of the proteins **simultaneously**.

ment in performance is even more pronounced. This demonstrates one of the important advantages of learning a single peptide distance function on an entire protein family.

Related work

Many different computational approaches have been suggested for the protein-peptide binding prediction problem (see [24] for a recent review). These methods can be roughly divided into three categories:

Motif based methods

Binding *motifs* represent important requirements needed for binding, such as the presence and proper spacing of certain amino acids within the peptide sequence. Prediction of protein-peptide binding is usually performed as motif searches [8,15]. The position specific scoring matrix (PSSM) approach is a statistical extension of the motif search methods, where a matrix represents the frequency of every amino acid in every position along the peptide. Peptide candidates can be assigned scores by summing up the position specific weights. The *RANKPEP* resource [13] uses this approach to predict peptide binding to MHC class I and class II molecules.

Structure based methods

These methods predict binding affinity by evaluating the binding energy of the protein-peptide complex [9]. These methods can be applied only when the three-dimensional structure of the protein-peptide complex is known or when reliable molecular models can be obtained.

Machine learning methods

Many different learning algorithms have been suggested for binding prediction. Among these are artificial neural networks (NetMHC) [12], Hidden Markov Models (HMM's) [10] and support vector machines (SVMHC) [11]. To the best of our knowledge all of these methods are trained separately for each protein (or supertype). Therefore, these methods work well when sufficient amounts of training data (i.e peptides which are known to be binders or non-binders for a given protein) is provided.

Results

We evaluated the performance of our method on several MHC class I and MHC class II datasets, and compared it to various other prediction methods (see Methods for details about these datasets). We begin with a thorough comparison of our method to the recently enhanced RANKPEP method [13] on MHC class I and class II datasets. In order to assess the importance of using experimentally determined non-binders, we tested our method on another MHC class I dataset collected from the MHCBN repository. On this dataset we also compare our method to various other MHC binding prediction methods.

MHC binding prediction on the MHCPEP dataset

We compared our method to the recently enhanced RANKPEP method [13]. We replicated the exact experimental setup described in [13]: (1) We used the exact same MHC class I and class II datasets. (2) Training was performed using 50% of the known binders for each of the MHC molecules. (3) The remaining binding peptides were used as test data to evaluate the algorithm's performance. These binders were tested against randomly generated peptides.

We trained *DistBoost* in two distinct scenarios: (1) Training using only binding peptides (using only positive constraints). (2) Training using both binding and (randomly generated) non-binding peptides (using both positive and negative constraints). In both scenarios *DistBoost* was trained **simultaneously** on all of the MHC molecules in each class. Fig. 4 presents a comparison of *DistBoost* to both of the PSSM's used in [13]. on the H-2Kd MHC class I molecule. Comparative results on the entire MHC class I and class II datasets are presented in Figures 5 and 6, respectively. In all these comparisons, the PSSM AUC scores (See Methods for details) are as reported in [13].

On the MHC class I molecules, our method significantly outperforms both PSSM's used by RANKPEP. On 21 out

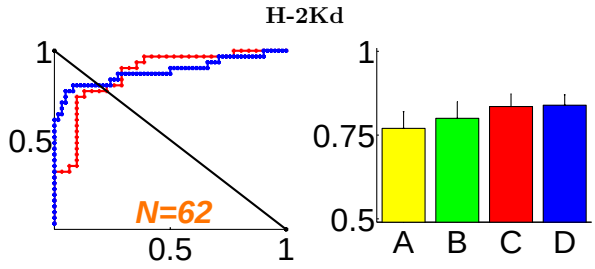

Figure 4

Comparative results of *DistBoost* and RANKPEP on the H-2Kd MHC class I molecule. The left plot presents ROC (see Evaluation methods section for details) curves of the best test score obtained when training on 50% of the entire data (red: using only positive constraints; blue: using both types of constraints). The intersection between the curves and the diagonal line marks the equal error-rate statistic. The right plot presents average AUC scores on test data. We compare the two PSSM methods used by RANKPEP (A: PROFILE-WEIGHT, B: BLK2PSSM) to *DistBoost* when trained using only positive constraints (C) and when trained using both positive and negative constraints (D). The averages were taken over 10 different runs on randomly selected train and test sets. *N* denotes the total number of binding peptides (of which 50% were used in the training phase and the remaining 50% were used in the test phase). For a detailed comparison see Figs. 5-6.

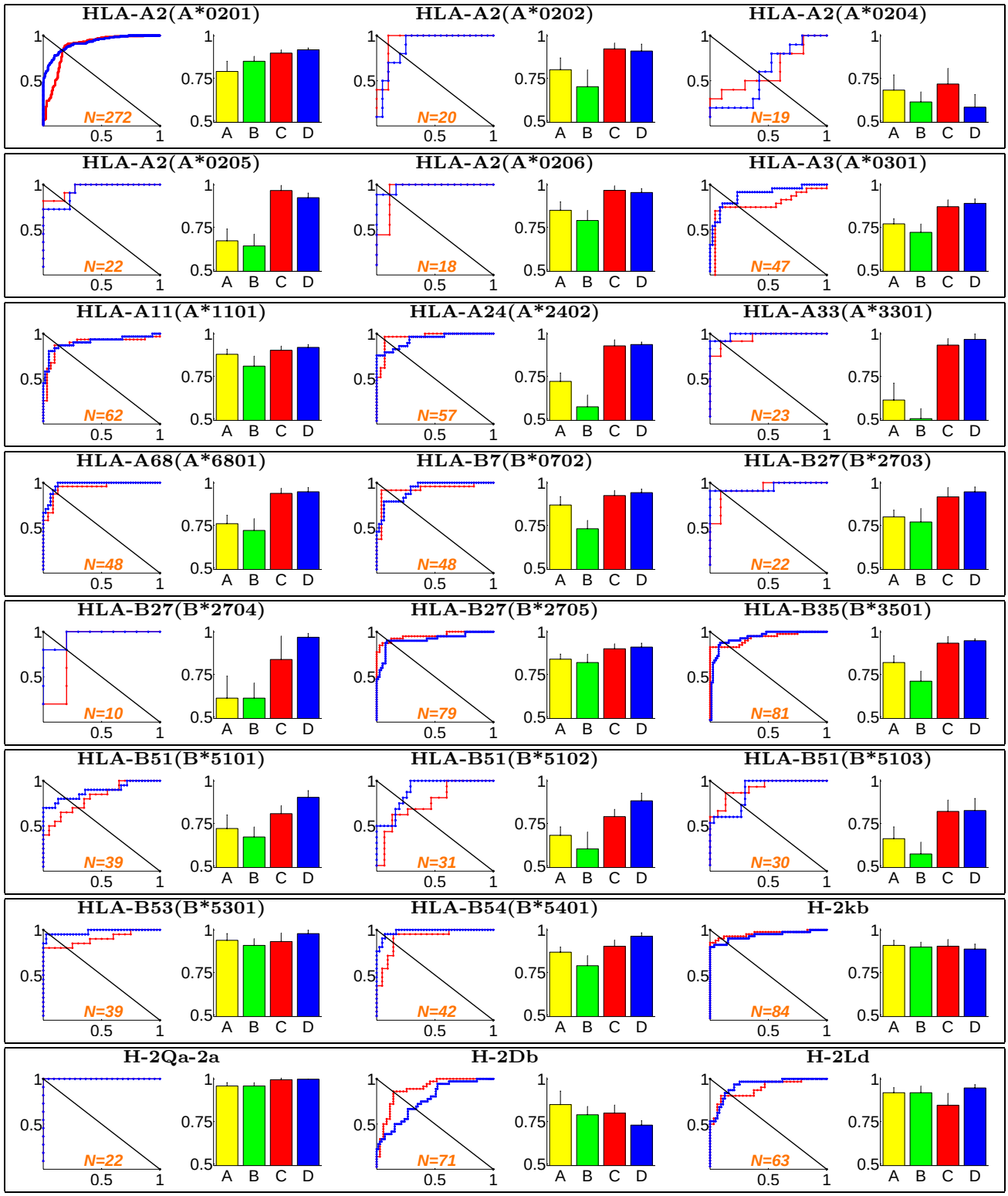

Figure 5
Comparative results of *DistBoost* (left plot and bars C-D) and RANKPEP (bars A-B) on 24 MHC class I molecules. Plot legends are identical to Fig 4. On 21 out of the 25 molecules (including Fig. 4), *DistBoost* outperforms both PSSM methods. On this data the use of negative constraints also improves performance. For numerical comparison, see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 1.

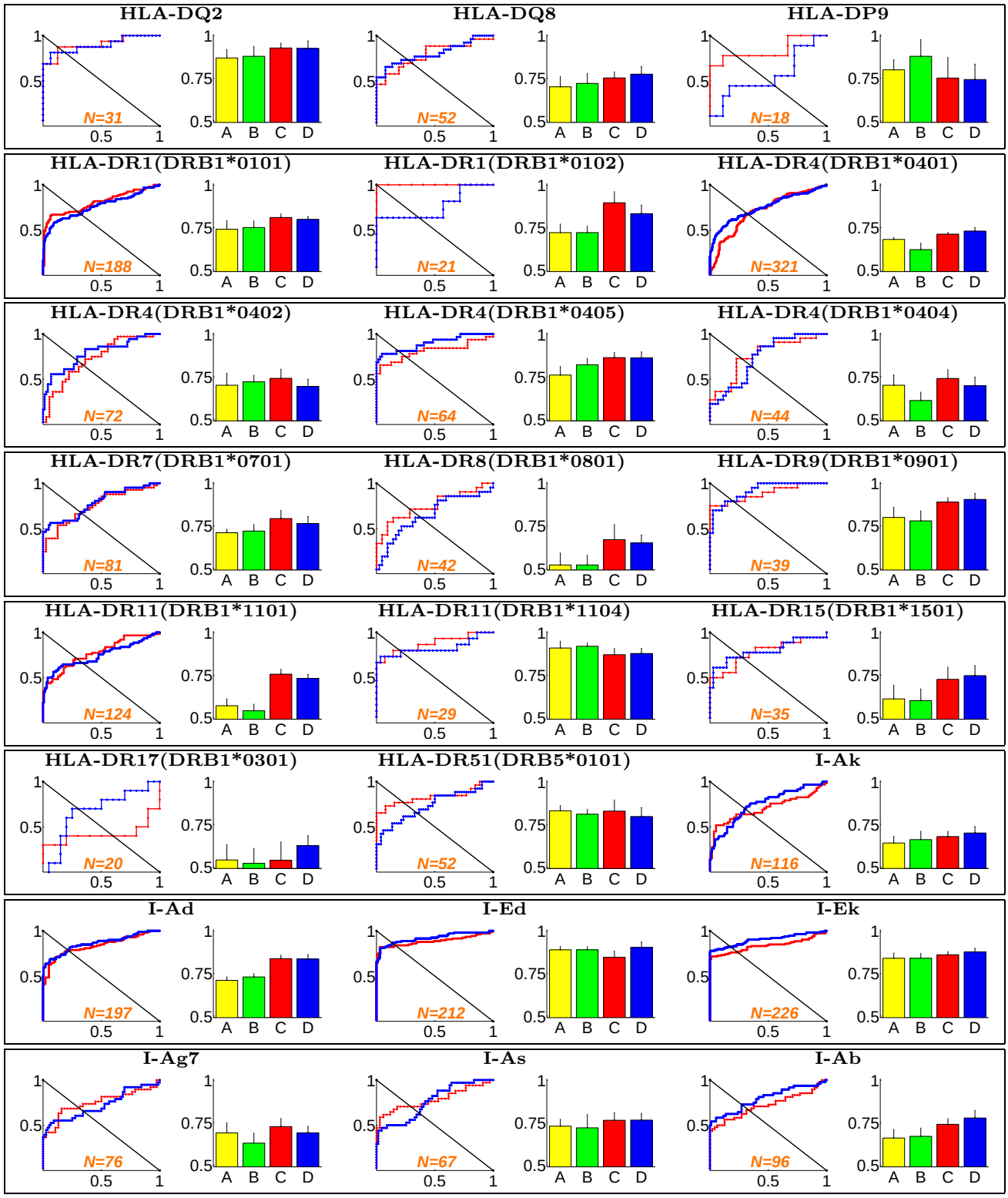

Figure 6
Comparative results of *DistBoost* (left plot and bars C-D) and RANKPEP (bars A-B) on 24 MHC class II molecules. Plot legends are identical to Fig 4. As may be seen, on 19 out of the 24 molecules, *DistBoost* outperforms both PSSM methods. On this data-set the use of negative constraints only slightly improves performance. For numerical comparison, see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 2.

of the 25 molecules *DistBoost's* average AUC score, when trained using only positive constraints, is higher than both PSSM methods. The improvement in performance is more pronounced on molecules with relatively small amounts of known binders (e.g. HLA-B27(B*2704) – 10 binders, HLA-A2(A*0205) – 22 binders and HLA-A33(A*3301) – 23 binders). One possible explanation of these results is that the information provided by other proteins within the protein family is used to enhance prediction accuracy, especially in cases where only small amounts of known binders exist. Additionally, it may be seen that using both positive and negative constraints on this dataset, usually improves the algorithm's performance. Another important advantage of *DistBoost* can be seen when comparing standard deviations (std) of the AUC scores. When *DistBoost* was trained using only positive constraints, on 13 out of the 25 molecules the algorithm's std was lower than the std of both PSSM's. When *DistBoost* was trained using both positive and negative constraints, on 20 out of the 25 molecules the algorithm's std was lower than the std of both PSSM's. These results imply that our method is more robust.

When tested on the MHC class II molecules, our method obtained similar improvements (see Fig. 6): On 19 out of the 24 molecules *DistBoost's* average AUC score when trained using only positive constraints is higher than both PSSM methods. In general, it appears that the performance of all of the compared methods is lower than on the MHC class I dataset. It is known that predicting binding affinity on MHC class II is more challenging, partially due to the fact that peptides that bind to class II molecules are extremely variable in length and share very limited sequence similarity [25]. On this dataset, the use of both positive and negative constraints improved *DistBoost's* performance on only 11 out of 24 molecules.

MHC class I binding prediction on the MHCBN dataset

The MHCPEP dataset only contains information about peptides that bind to various MHC molecules. In contrast, the MHCBN dataset also contains information about non-binding peptides for some MHC class I molecules. We used this dataset to evaluate the importance of learning using experimentally determined non-binders (as opposed to randomly generated non binders).

We compared *DistBoost* to various other computational prediction methods on peptides that bind to the HLA-A2 supertype, collected from the MHCBN repository. Specifically, we compared the performance of the following methods: (1) The *DistBoost* algorithm. (2) The *SVMHC* web server [11]. (3) The *NetMHC* web server [12]. (4) The RANKPEP resource [13] (5) The Euclidean distance metric in $\mathbb{R}^{45}$ Despite the fact that methods (2–4) are protein specific, they also provide predictions on various MHC supertypes including the HLA-A2 supertype.

We note that it is unclear whether the peptides collected from the MHCBN repository are the HLA-A2 supertype binders, or HLA-A*0201 binders which was named HLA-A2 in the older HLA nomenclature. When we compared our predictions to those of the SVMHC and NetMHC methods on the HLA-A*0201, similar results were obtained.

We trained *DistBoost* on 70% of the **entire** *MHCclass1BN* data (including binding and non-binding peptides) and compared its performance to all other methods on the **single** HLA-A2 supertype. The test set, therefore, consists of the remaining 30% of HLA-A2 data. The results are shown in Fig. 7(a). As may be seen, *DistBoost* outperforms all other methods, including SVMHC, NetMHC and RANKPEP, which were trained on this specific supertype. However, it is important to note, that unlike *DistBoost*, all of these methods were trained using randomly generated non-binders. The performance of all of these methods when tested against random peptides is much better – AUC scores of SVMHC: 0.947, NetMHC: 0.93 and RANKPEP: 0.928. When *DistBoost* was trained and tested using randomly generated non-binders it achieved an AUC score of 0.976. Interestingly, when *DistBoost* was trained using real non-binders and tested on randomly generated non-binders it obtained an AUC score of 0.923. These results seem to imply that learning using random non-binders does **not** generalize well to experimentally determined non-binders. On the other hand, learning from "real" non-binders generalizes very well to random non-binders.

Our proposed method is trained **simultaneously** on a number of proteins from the same family, unlike methods (2–4). However, our final predictions are protein specific. As the results reveal, we obtain high binding prediction accuracy when tested on a single protein (see Fig. 7(a)). In order to quantify the overall protein specific binding prediction accuracy, we present ROC curves for *DistBoost* and the Euclidean affinity functions when tested on the **entire** MHCclass1BN dataset (Fig. 7(b)). The peptide-peptide distance matrices and the protein-peptide affinity matrices of these two methods are presented in Fig. 3. On this dataset *DistBoost* obtained excellent performance.

In order to evaluate the stability and learning power of *DistBoost* we ran it on the *MHCclass1BN* dataset, while varying the percentage of training data. Fig. 8 presents the algorithm's learning curves when trained using only positive constraints and when trained using both positive and negative constraints. As may be expected, on average, performance improves as the amount of training data

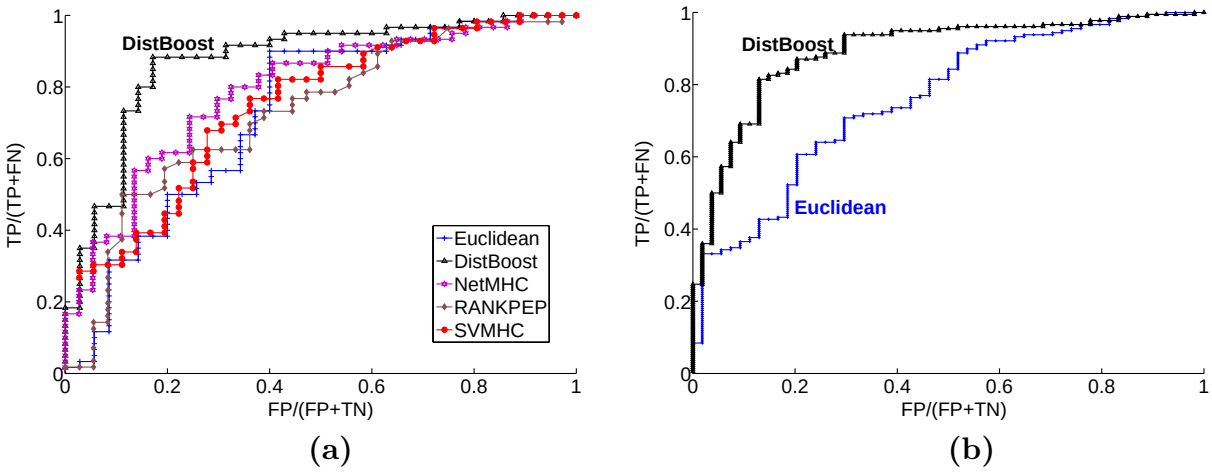

Figure 7

(**a**) ROC curves on test data from the HLA-A2 supertype. *DistBoost* is compared to the following algorithms: the *SVMHC* web server [11], the *NetMHC* web server [12], the RANKPEP resource [13] and the Euclidean distance metric in $\mathbb{R}^{45}$. (**b**) *DistBoost* and the Euclidean affinity ROC curves on test data from the **entire** *MHCclass I BN* dataset. The rest of the methods are not presented since they were not trained in this multi-protein scenario. In both cases, *DistBoost* was trained on 70% of the data and tested on the remaining 30%. Results are best seen in color.

increases. Note that *DistBoost* achieves almost perfect performance with relatively small amounts of training data. Additionally, we can see that on this dataset learning from both types of constraints dramatically improves performance.

The PepDist Webserver

Our proposed method is now publicly available through the *PepDist* webserver, which can be found at http://www.pepdist.cs.huji.ac.il. The current version provides binding predictions of 9-mer peptides to 35 different MHC class I alleles. The engine also supports multiple peptide queries. We hope to enhance the webserver in the near future to provide predictions for more MHC class I alleles and also for MHC class II alleles.

Discussion and Conclusion

In this paper we proposed *PepDist*: a novel formulation of the protein-peptide binding prediction problem that has two foundations. The first is to predict binding affinity by learning peptide-peptide distance functions. The second is to learn a *single* distance function over an entire family of proteins. Our formulation has several advantages over existing computational approaches:

1. Our method also works well on proteins for which small amounts of known binders are currently available.

2. Unlike standard binary classifiers, our method can be trained on an entire protein family using only information about binding peptides (i.e. without using real/randomly generated non-binders).

3. Our method can compute the relative binding affinities of a peptide to several proteins from the same protein family.

In order to learn such distance functions we casted the problem as a semi-supervised learning problem in which equivalence constraints can be naturally obtained from empirical data. Specifically, we used the *DistBoost* algorithm, that learns distance functions using positive and negative equivalence constraints. Our experiments suggest that binding prediction based on such learned distance functions exhibits excellent performance. It should be noted that our proposed learning scheme can be also implemented using other distance learning algorithms and in our future work we also plan to further investigate this idea. We also hope that the *PepDist* formulation will allow addressing the more challenging task of peptide ranking. One way of doing this is by incorporating information about relative binding values into the distance learning algorithm.

Our approach may also be useful for predicting some protein-protein interactions such as PDZ-protein complexes.

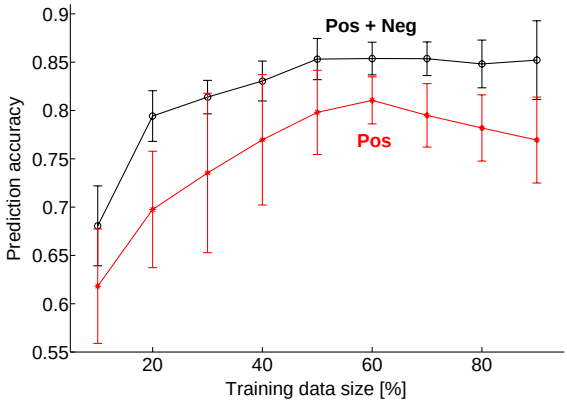

Figure 8
Learning curves of *DistBoost* trained using only positive constraints (*Pos*) and using both types of constraints (*Pos* + *Neg*). Prediction accuracy based on the AUC score, averaged over 20 different randomly selected training sets.

The PDZ domains are frequently occurring interaction domains involved in organizing signal transduction complexes and attaching proteins to the cytoskeleton [26]. In most cases, this is accomplished by specific recognition of the ligands' carboxyl termini (or regions "mimicking" the structure of a carboxyl terminal). Therefore, predicting whether a protein binds to a specific PDZ domain, can be cast as protein-peptide prediction problem where the "peptide" is the short linear sequence (4 – 6 amino acids long) lying at the protein's C-terminal. We are currently examining the feasibility of using the *PepDist* framework for this application.

Our novel formulation of the protein-peptide binding prediction problem and the results obtained suggest two interesting conclusions: The first is that learning a *single* distance function over an entire family of proteins achieves higher prediction accuracy than learning a set of binary classifiers for each of the proteins separately. This effect is even more pronounced on proteins for which only small amounts of binders and non-binders are currently available. The second interesting conclusion, is the importance of obtaining information on experimentally determined non-binders. These non-binders (as opposed to randomly generated non-binders) are usually somewhat similar to known binders, since they were in many cases suspected to be binders. Our results on the MHCBN dataset show that learning with real non-binders generalizes better than learning with randomly generated peptides that are assumed to be non-binders. This suggests that information about non-binding peptides should also be published and made publicly available.

Methods

The DistBoost Algorithm

Our peptide-peptide distance functions are learned using the *DistBoost* algorithm. *DistBoost* is a semi-supervised learning technique that learns a distance function using unlabeled data points and equivalence constraints.

Notations

Let us denote by $\{x_i\}_{i=1}^{n}$ the set of input data points which belong to some vector space $\mathcal{X}$. The space of all pairs of points in $\mathcal{X}$ is called the "product space" and is denoted by $\mathcal{X}\times\mathcal{X}$. An equivalence constraint is denoted by (x_{i1}, x_{i2}, y_i) where $y_i = 1$ if points (x_{i1}, x_{i2}) belong to the same class (positive constraint) and $y_i = -1$ if these points belong to different classes (negative constraint). $(x_{i1}, x_{i2}, *)$ denotes an unlabeled pair. The *DistBoost* algorithm learns a bounded distance function, $D : \mathcal{X}\times\mathcal{X} \to [0,1]$, that maps each pair of points to a real number in $[0,1]$.

Algorithm description

The algorithm makes use of the observation that equivalence constraints on points in $\mathcal{X}$ are binary labels in the product space, $\mathcal{X}\times\mathcal{X}$. By posing the problem in product space we obtain a classical binary classification problem: an optimal classifier should assign +1 to all pairs of points that come from the same class, and -1 to all pairs of points that come from different classes. This binary classification problem can be solved using traditional margin based classification techniques. Note, however, that in many real world problems, we are only provided with a sparse set of equivalence constraints and therefore the margin based binary classification problem is semi-supervised.

DistBoost learns a distance function using a well known machine learning technique, called *Boosting* [27,28]. In Boosting, a set of "weak" learners are iteratively trained and then linearly combined to produce a "strong" learner. Specifically, *DistBoost's* weak learner is based on the constrained Expectation Maximization (cEM) algorithm [29]. The cEM algorithm is used to generate a "weak" distance function. The final ("strong") distance function is a weighted sum of a set of such "weak" distance functions. The algorithm is presented in Fig. 9 and illustrated in Fig. 10.

In order to make use of unlabeled data points, *DistBoost's* weak learner is trained in the original space, $\mathcal{X}$, and is then used to generate a "weak distance function" on the product space. *DistBoost* uses an augmentation of the

Input:

Data points: $(x_1, ..., x_n)$, $x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points: $(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)$ $i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)
  $w_k = 1/n$ $k = 1, \ldots, n$ (weights over data points)
- For $t = 1, .., T$

  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak hypothesis $\tilde{h}_t : \mathcal{X} \times \mathcal{X} \rightarrow [-1, 1]$ and define a weak distance function as $h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h}_t(x_i, x_j)\right) \in [0, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i \tilde{h}_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2}\ln(\frac{1+r_t}{1-r_t})$
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:

$$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{h}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\alpha_t) & y_i = * \end{cases}$$

  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final distance function $\mathcal{D}(x_i, x_j) = \sum_{t=1}^{T} \alpha_t h_t(x_i, x_j)$

Figure 9
The *DistBoost* Algorithm.

'Adaboost with confidence intervals' algorithm [27] to incorporate unlabeled data into the boosting process. More specifically, given a partially labeled dataset $\left\{\left(x_{i_1}, x_{i_2}, y_i\right)\right\}_{i=1}^{N}$ where $y_i \in \{1, -1, *\}$, the algorithm searches for a hypothesis $\mathcal{D}\left(x_{i_1}, x_{i_2}\right) = \sum_{t=1}^{T} \alpha_t h_t\left(x_{i_1}, x_{i_2}\right)$ which minimizes the following loss function:

$$\sum_{\{i|y_i=1,-1\}} \exp\left(-y_i \mathcal{D}\left(x_{i_1}, x_{i_2}\right)\right) \qquad (2)$$

Note that this semi-supervised boosting scheme computes the weighted loss only on **labeled** pairs of points but updates the weights over **all** pairs of points (see Figure 9 steps (3–6)). The unlabeled points effectively constrain the parameter space of the weak learner, giving priority to hypotheses which both comply with the pairwise constraints and with the data's density. Since the weak learner's task becomes harder in later boosting rounds, the boosting algorithm gradually reduces the weights of the unlabeled pairs (see Figure 9 step (5)). While the weak learner accepts a distribution over the points in the original space $\mathcal{X}$, the boosting process described above generates a distribution over pairs of points that belong to the product space $\mathcal{X}\times\mathcal{X}$. The distribution over the product space is converted to a distribution over the sample points by simple marginalization (see Figure 9 step (7) of the algorithm). The translation from the original input space into product space is introduced in step (2) of the algorithm and is further discussed below.

DistBoost's weak learner

DistBoost's weak learner is based on the constrained Expectation Maximization (cEM) algorithm [29]. The algorithm uses unlabeled data points and a set of equivalence constraints to find a Gaussian Mixture Model (GMM) that complies with these constraints. A GMM is a parametric statistical model which is given by $p\left(x|\Theta\right) = \sum_{l=1}^{M} \pi_l p\left(x|\theta_l\right)$, where π_l denotes the weight of each Gaussian, θ_l its parameters, and M denotes the number of Gaussian sources in the GMM. Estimating the parameters (Θ) of a GMM is usually done using the well known EM algorithm [30]. The cEM algorithm introduces equivalence constraints by modifying the 'E' (Expectation) step of the algorithm: instead of summing over *all* possible assignments of data points to sources, the expectation is taken only over assignments which comply with the given equivalence constraints.

The cEM algorithm's input is a set of unlabeled points $X = \{x_i\}_{i=1}^{n}$, and a set of pairwise constraints, Ω, over these points. Denote positive constraints by $\left\{\left(p_j^1, p_j^2\right)\right\}_{j=1}^{N_p}$ and negative constraints by $\left\{\left(n_k^1, n_k^2\right)\right\}_{k=1}^{N_n}$. Let $H = \{h_i\}_{i=1}^{n}$ denote the hidden assignment of each data point x_i to one of the Gaussian sources ($h_i \in \{1,..., M\}$). The constrained EM algorithm assumes the following joint distribution of the observables X and the hiddens H:

$$p\left(X, H|\Theta, \Omega\right) = \frac{1}{Z}\prod_{i=1}^{n} \pi_{h_i} p\left(x_i|\theta_{h_i}\right) \prod_{j=1}^{N_p} \delta_{h_{p_j^1} h_{p_j^2}} \prod_{k=1}^{N_n} \left(1 - \delta_{h_{n_k^1} h_{n_k^2}}\right) \qquad (3)$$

where Z is the normalizing factor and δ_{ij} is Kronecker's delta. The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (3) with respect to H. For a more detailed description of this weak learner see [29].

In order to use the algorithm as a weak learner in our boosting scheme, we modified the algorithm to incorporate weights over the data samples. These weights are provided by the boosting process in each round (see Fig. 9 step 7).

Generating a weak distance function using a GMM

The weak learners' task is to provide a weak distance function $h_t(x_i, x_j)$ over the product space $\mathcal{X}\times\mathcal{X}$. Let us Denote by $MAP(x_i)$ the Maximum A-Posteriori assignment of point x_i and by $p^{MAP}(x_i)$ the MAP probability of this point: $p^{MAP}\left(x_i\right) = \max_m p\left(h_i = m|x_i, \Theta\right)$. We partition the data into M groups using the MAP assignment of the points and define

$$\tilde{h}_t\left(x_i, x_j\right) \equiv \begin{cases} +p^{MAP}\left(x_i\right) \cdot p^{MAP}\left(x_j\right) & \text{if } \mathrm{MAP}\left(x_i\right) = \mathrm{MAP}\left(x_j\right) \\ -p^{MAP}\left(x_i\right) \cdot p^{MAP}\left(x_j\right) & \text{if } \mathrm{MAP}\left(x_i\right) \neq \mathrm{MAP}\left(x_j\right) \end{cases}$$

The weak distance function is given by

$$h_t\left(x_i, x_j\right) = \frac{1}{2}\left(1 - \tilde{h}_t\left(x_i, x_j\right)\right) \in [0,1] \qquad (4)$$

It is easy to see that if the MAP assignment of two points is identical their distance will be in [0, 0.5] and if their MAP assignment is different their distance will be in [0.5, 1].

Datasets

The first two datasets we compiled (MHCclass1 and MHCclass2) were the same as those described in [13]. Fol-

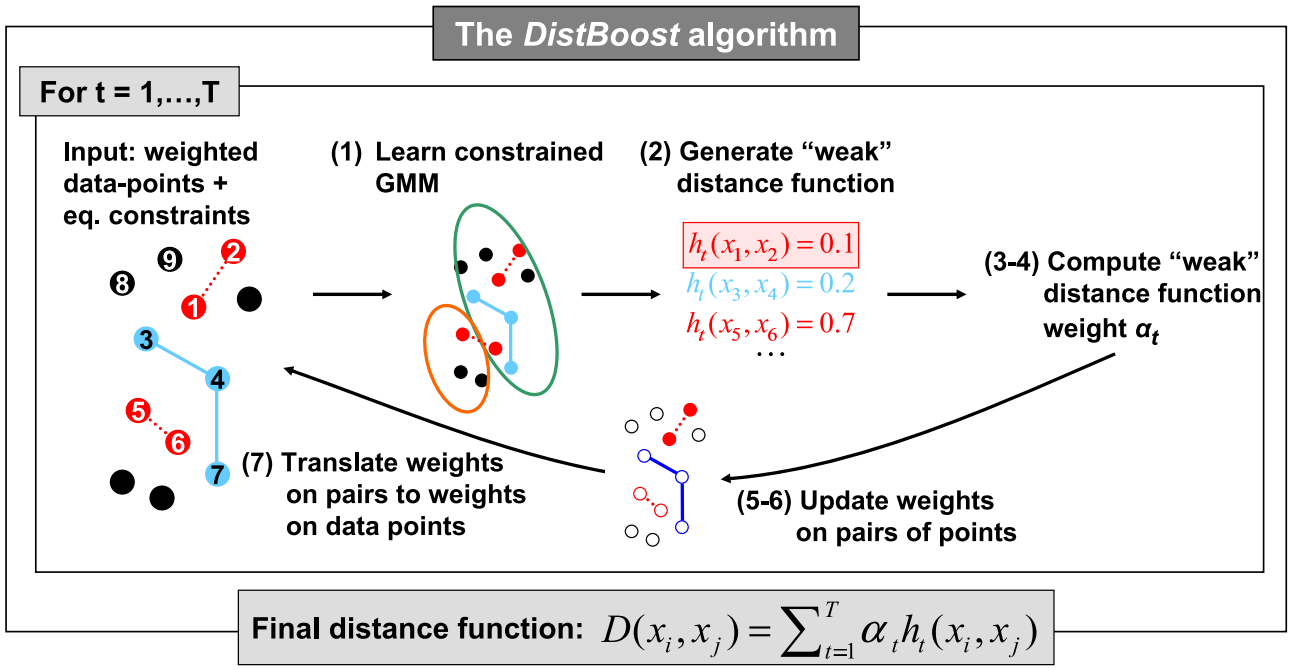

Figure 10
An illustration of the *DistBoost* algorithm. At each boosting round *t* the weak learner is trained using weighted input points and some equivalence constraints. In the example above, points 1, 2 and 5, 6 are negatively constrained (belong to different classes) and points 3, 4 and 4, 7 are positively constrained (belong to the same class). All other pairs of points (e.g. 8, 9 and 1, 4) are unconstrained. The constrained EM algorithm is used to learn a GMM (step (1)). This GMM is then used to generate a "weak" distance function (step (2)) that assigns a value in [0, 1] to each pair of points. The distance function is assigned a hypothesis weight (steps (3–4)) which corresponds to its success in satisfying the current weighted constraints. The weights of the equivalence constraints are updated (steps (5–6)) – increasing the weights of constraints that were unsatisfied by the current weak learner. Finally, the weights on pairs are translated to weights on data points (step (7)). In the example above, the distance between the negatively constrained points 1, 2 is small (0.1) and therefore the weight of this constraint will be enhanced.

lowing the works of [8,9,13] we considered peptides with a fixed sequence length of 9 amino acids. Sequences of peptides, that bind to MHC class I or class II molecules, were collected from the MHCPEP dataset [31]. Each entry in the MHCPEP dataset contains the peptide sequence, its MHC specificity and, where available, observed activity and binding affinity. Peptides, that are classified as low binders or contain undetermined residues (denoted by the letter code *X*), were excluded. We then grouped all 9 amino acid long peptides (9-mers), that bind to MHC class I molecules, to a dataset, called *MHCclass1*. This dataset consists of binding peptides for 25 different MHC class I molecules (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 3).

Unlike MHC class I binding peptides, peptides binding to MHC class II molecules display a great variability in length, although only a peptide core of 9 residues fits into the binding groove. Following [13], we first used the MEME program [32] to align the binding peptides for each molecule, based on a single 9 residues motif. We finally filtered out redundant peptides and obtained the *MHCclass2* dataset. This dataset consists of binding peptides for 24 different MHC class II molecules (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 4).

Since all peptides in the MHCPEP dataset are binders, we added randomly generated peptides as non-binders to both MHCclass1 and MHCclass2 datasets (amino acid frequencies as in the Swiss-Prot database). The number of non-binders used in any test set was twice the number of the binding peptides. During the training phase, the number of non-binders was the same as the number of binders. In order to assess the performance of the prediction algorithms on experimentally determined non-binders, we compiled a third dataset, called *MHCclass1BN*.

This dataset consists of binding and non-binding peptides, for 8 different MHC class I molecules, based on the *MHCBN 3.1* website [33] (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 5).

Data representation

DistBoost requires that the data be represented in some continuous vector feature space. Following [23] each amino acid was encoded using a 5-dimensional property vector. Therefore, each peptide in the MHC datasets is a point in $\mathbb{R}^{45}$. The property vectors for each of the 20 amino acids are based on multidimensional scaling of 237 physical-chemical properties. Venkatarajan and Braun's analysis [23] showed that these 5 properties correlate well with hydrophobicity, size, α-helix preference, number of degenerate triplet codons and the frequency of occurrence of amino acid residues in β-strands. They also showed that the distances between pairs of amino-acids in the 5-dimensional property space are highly correlated with corresponding scores from similarity matrices derived from sequence and 3D structure comparisons.

Evaluation methods

In order to evaluate the algorithms' performance, we measured the affinity of all test peptides to each of the proteins. We present the prediction accuracy (that is how well binders are distinguished from non-binders) of the various algorithms as ROC (Receiver Operating Characteristic) curves. The *X*-axis represents the percentage of "false alarms" which is $FP/(FP + TN)$ (where *FP* denotes False Positives, and *TN* denotes True Negatives). The *Y*-axis represents the percentage of "hits" which is $TP/(TP + FN)$ (where *TP* denotes True Positives and *FN* denotes False Negatives). The fraction of the area under the curve (AUC) is indicative of the distinguishing power of the algorithm and is used as its prediction accuracy.

Authors' contributions

Both authors contributed equally.

Additional material

Additional File 1

This file includes the following tables: 1. Table 1: Average AUC scores and standard deviations obtained by RANKPEP and DistBoost *on MHC class I molecules. 2. Table 2: Average AUC scores and standard deviations obtained by RANKPEP and* DistBoost *on MHC class II molecules. 3. Tables 3, 4, 5, describe the 3 datasets used in the paper.*

Click here for file

[http://www.biomedcentral.com/content/supplementary/1471-2105-7-S1-S3-S1.pdf]

Acknowledgements

We thank Daphna Weinshall for many fruitful discussions and comments. We also thank Itamar Glatzer for his help in the implementation of the *PepDist* webserver. C.Y is supported by Yeshaya Horowitz Association through the Center for Complexity Science.

References

1. Hertz T, Bar-Hillel A, Weinshall D: **Learning Distance Functions for Image Retrieval.** *CVPR, Washington DC* 2004.
2. Hertz T, Bar-Hillel A, Weinshall D: **Boosting Margin Based Distance Functions for Clustering.** *ICML* 2004.
3. Rammensee HG, Friede T, Stevanoviic S: **MHC ligands and peptide motifs: first listing.** *Immunogenetics* 1995, **41(4):**178-228.
4. Yewdell JW, Bennink JR: **Immunodominance in Major Histocompatibility Complex Class I-Restricted T-Lymphocyte Responses.** *Annual Review of Immunology* 1999, **17:**51-88.
5. Janeway CA, Travers P, Walport M, Shlomchik M: *Immunobiology* 5th edition. *New York and London: Garland Publishing*; 2001.
6. Sette A: **Tools of the Trade in Vaccine Design.** *Science* 2000, **290(5499):**2074b-2075.
7. Brusic V, Rudy G, Harrison LC: **Prediction of MHC binding peptides using artificial neural networks.** *Complexity International* 1995, **2:**.
8. Gulukota K, Sidney J, Sette A, DeLisi C: **Two complementary methods for predicting peptides binding major histocompatibility complex molecules.** *Journal of Molecular Biology* 1997, **267:**1258-1267.
9. Schueler-Furman O, Altuvia Y, Sette A, Margalit H: **Structure-based prediction of binding peptides to MHC class I molecules: application to a broad range of MHC alleles.** *Protein Sci* 2000, **9(9):**1838-1846.
10. Mamitsuka H: **Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models.** *Proteins* 1998, **33(4):**460-474.
11. Donnes P, Elofsson A: **Prediction of MHC class I binding peptides, using SVMHC.** *BMC Bioinformatics* 2002, **3:**25. http://www-bs-informatik.uni-tuebingen.de/SVMHC
12. Buus S, Lauemoller S, Worning P, Kesmir C, Frimurer T, Corbet S, Fomsgaard A, Hilden J, Holm A, Brunak S: **Sensitive quantitative predictions of peptide-MHC binding by a 'Query by Committee' artificial neural network approach.** *Tissue Antigens* 2003, **62(5):**378-384. http://www.cbs.dt.dk/services/NetMHC/
13. Reche PA, Glutting JP, Zhang H, Reinher EL: **Enhancement to the RANKPEP resource for the prediction of peptide binding to MHC molecules using profiles.** *Immunogenetics* 2004, **56(6):**405-419. http://mif.dfci.harvard.edu/Tools/rankpep.html
14. Yu K, Petrovsky N, Schonbach C, Koh JL, Brusic V: **Methods for Prediction of Peptide Binding to MHC Molecules: A Comparative Study.** *Molecular Medicine* 2002, **8:**137-148.
15. Andersen M, Tan L, Sondergaard I, Zeuthen J, Elliott T, Haurum J: **Poor correspondence between predicted and experimental binding of peptides to class I MHC molecules.** *Tissue Antigens* 2000, **55(6):**519-531.
16. Wiedemann U, Boisguerin P, Leben R, Leitner D, Krause G, Moelling K, Volkmer-Engert R, Oschkinat H: **Quantification of PDZ Domain Specificity, Prediction of Ligand Affinity and Rational Design of Super-binding Peptides.** *Journal of Molecular Biology* 2004, **343:**703-718.
17. Sette A, Sidney J: **Nine major HLA class I supertypes account for the vast preponderance of HLA-A and -B polymorphism.** *Immunogenetics* 1999, **50:**201-212.
18. Bar-Hilel A, Hertz T, Shental N, Weinshall D: **Learning Distance Functions using Equivalence Relations.** *The 20th International Conference on Machine Learning* 2003.
19. Xing E, Ng A, Jordan M, Russell S: **Distance Metric learnign with application to clustering with side-information.** In *Advances in Neural Information Processing Systems Volume 15. The MIT Press*; 2002.
20. WagstaffF K, Cardie C, Rogers S, Schroedl S: **Constrained K-means Clustering with Background Knowledge.** In *Proc 18th International Conf on Machine Learning Morgan Kaufmann, San Francisco, CA*; 2001:577-584.
21. Bilenko M, Basu S, Mooney R: **Integrating Constraints and Metric Learning in Semi-Supervised Clustering.** In *ICML Banff Canada, AAAI press*; 2004. [citeseer.ist.psu.edu/705723.html]

22. Klein D, Kamvar S, Manning C: **From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering.** 2002. [citeseer.nj.nec.com/klein02from.html]
23. Venkatarajan MS, Braun W: **New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical-chemical properties.** *Journal of Molecular Modeling* 2001, **7(12):**445-453.
24. Flower DR: **Towards in silico prediction of immunogenic epitopes.** *TRENDS in immunology* 2003, **24:**.
25. Madden DR: **The Three-Dimensional Structure of Peptide-MHC Complexes.** *Annual Review of Immunology* 1995, **13:**587-622.
26. Hung AY, Sheng M: **PDZ Domains: Structural Modules for Protein Complex Assembly.** *J Biol Chem* 2002, **277(8):**5699-5702.
27. Schapire RE, Singer Y: **Improved Boosting Using Confidence-rated Predictions.** *Machine Learning* 1999, **37(3):**297-336.
28. Schapire RE, Freund Y, Bartlett P, Lee WS: **Boosting the margin: a new explanation for the effectiveness of voting methods.** In *Proc 14th International Conference on Machine Learning Morgan Kaufmann*; 1997:322-330.
29. Shental N, Bar-Hilel A, Hertz T, Weinshall D: **Computing Gaussian Mixture Models with EM using Equivalence Constraints.** *NIPS* 2003.
30. Dempster AP, Laird NM, Rubin DB: **Maximum Likelihood from incomplete data via the EM algorithm.** *JRSSB* 1977, **39:**1-38.
31. Brusic V, Rudy G, Harrison LC: **MHCPEP, a database of MHC-binding peptides: update 1997.** *Nucl Acids Res* 1998, **26:**368-371.
32. Bailey T, Elkan C: **Fitting a mixture model by expectation maximization to discover motifs in biopolymers.** *ISMB* 1994, **2:**28-36.
33. Bhasin M, Singh H, Raghava GPS: **MHCBN: a comprehensive database of MHC binding and non-binding peptides.** *Bioinformatics* 2003, **19(5):**665-666.

Chapter 7

Analyzing Auditory Neurons by Learning Distance Functions

This chapter includes the following publications:

[F] Inna Weiner, Tomer Hertz, Israel Nelken and Daphna Weinshall, **Analyzing Auditory Neurons by Learning Distance Functions**, *in the 19th International Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2005.

Analyzing Auditory Neurons by Learning Distance Functions

Inna Weiner[1] **Tomer Hertz**[1,2] **Israel Nelken**[2,3] **Daphna Weinshall**[1,2]

[1]School of Computer Science and Engineering,
[2]The Center for Neural Computation, [3]Department of Neurobiology,
The Hebrew University of Jerusalem, Jerusalem, Israel, 91904
weinerin,tomboy,daphna@cs.huji.ac.il,israel@md.huji.ac.il

Abstract

We present a novel approach to the characterization of complex sensory neurons. One of the main goals of characterizing sensory neurons is to characterize dimensions in stimulus space to which the neurons are highly sensitive (causing large gradients in the neural responses) or alternatively dimensions in stimulus space to which the neuronal response are invariant (defining iso-response manifolds). We formulate this problem as that of learning a geometry on stimulus space that is compatible with the neural responses: the distance between stimuli should be large when the responses they evoke are very different, and small when the responses they evoke are similar. Here we show how to successfully train such distance functions using rather limited amount of information. The data consisted of the responses of neurons in primary auditory cortex (A1) of anesthetized cats to 32 stimuli derived from natural sounds. For each neuron, a subset of all pairs of stimuli was selected such that the responses of the two stimuli in a pair were either very similar or very dissimilar. The distance function was trained to fit these constraints. The resulting distance functions generalized to predict the distances between the responses of a test stimulus and the trained stimuli.

1 Introduction

A major challenge in auditory neuroscience is to understand how cortical neurons represent the acoustic environment. Neural responses to complex sounds are idiosyncratic, and small perturbations in the stimuli may give rise to large changes in the responses. Furthermore, different neurons, even with similar frequency response areas, may respond very differently to the same set of stimuli. The dominant approach to the functional characterization of sensory neurons attempts to predict the response of the cortical neuron to a novel stimulus. Prediction is usually estimated from a set of known responses of a given neuron to a set of stimuli (sounds). The most popular approach computes the spectrotemporal receptive field (STRF) of each neuron, and uses this linear model to predict neuronal responses. However, STRFs have been recently shown to have low predictive power [10, 14].

In this paper we take a different approach to the characterization of auditory cortical neurons. Our approach attempts to learn the non-linear warping of stimulus space that is in-

duced by the neuronal responses. This approach is motivated by our previous observations [3] that different neurons impose different partitions of the stimulus space, which are not necessarily simply related to the spectro-temporal structure of the stimuli. More specifically, we characterize a neuron by learning a pairwise distance function over the stimulus domain that will be consistent with the similarities between the responses to different stimuli, see Section 2. Intuitively a good distance function would assign small values to pairs of stimuli that elicit a similar neuronal response, and large values to pairs of stimuli that elicit different neuronal responses.

This approach has a number of potential advantages: First, it allows us to aggregate information from a number of neurons, in order to learn a good distance function even when the number of known stimuli responses per neuron is small, which is a typical concern in the domain of neuronal characterization. Second, unlike most functional characterizations that are limited to linear or weakly non-linear models, distance learning can approximate functions that are highly non-linear. Finally, we explicitly learn a distance function on stimulus space; by examining the properties of such a function, it may be possible to determine the stimulus features that most strongly influence the responses of a cortical neuron. While this information is also implicitly incorporated into functional characterizations such as the STRF, it is much more explicit in our new formulation.

In this paper we therefore focus on two questions: (1) Can we learn distance functions over the stimulus domain for single cells using information extracted from their neuronal responses?? and (2) What is the predictive power of these cell specific distance functions when presented with novel stimuli? In order to address these questions we used extracellular recordings from 22 cells in the auditory cortex of cats in response to natural bird chirps and some modified versions of these chirps [1]. To estimate the distance between responses, we used a normalized distance measure between the peri-stimulus time histograms of the responses to the different stimuli.

Our results, described in Section 4, show that we can learn compatible distance functions on the stimulus domain with relatively low training errors. This result is interesting by itself as a possible characterization of cortical auditory neurons, a goal which eluded many previous studies [3]. Using cross validation, we measure the test error (or predictive power) of our method, and report generalization power which is significantly higher than previously reported for natural stimuli [10]. We then show that performance can be further improved by learning a distance function using information from pairs of related neurons. Finally, we show better generalization performance for wide-band stimuli as compared to narrow-band stimuli. These latter two contributions may have some interesting biological implications regarding the nature of the computations done by auditory cortical neurons.

Related work Recently, considerable attention has been focused on spectrotemporal receptive fields (STRFs) as characterizations of the function of auditory cortical neurons [8, 4, 2, 11, 16]. The STRF model is appealing in several respects: it is a conceptually simple model that provides a linear description of the neuron's behavior. It can be interpreted both as providing the neuron's most efficient stimulus (in the time-frequency domain), and also as the spectro-temporal impulse response of the neuron [10, 12]. Finally, STRFs can be efficiently estimated using simple algebraic techniques.

However, while there were initial hopes that STRFs would uncover relatively complex response properties of cortical neurons, several recent reports of large sets of STRFs of cortical neurons concluded that most STRFs are somewhat too simple [5], and that STRFs are typically rather sluggish in time, therefore missing the highly precise synchronization of some cortical neurons [11]. Furthermore, when STRFs are used to predict neuronal responses to natural stimuli they often fail to predict the correct responses [10, 6]. For example, in Machens et al. only 11% of the response power could be predicted by STRFs on average [10]. Similar results were also reported in [14], who found that STRF models

account for only $18 - 40\%$ (on average) of the stimulus related power in auditory cortical neural responses to dynamic random chord stimuli. Various other studies have shown that there are significant and relevant non-linearities in auditory cortical responses to natural stimuli [13, 1, 9, 10]. Using natural sounds, Bar-Yosef et. al [1] have shown that auditory neurons are extremely sensitive to small perturbations in the (natural) acoustic context. Clearly, these non-linearities cannot be sufficiently explained using linear models such as the STRF.

2 Formalizing the problem as a distance learning problem

Our approach is based on the idea of learning a cell-specific distance function over the space of all possible stimuli, relying on partial information extracted from the neuronal responses of the cell. The initial data consists of stimuli and the resulting neural responses. We use the neuronal responses to identify pairs of stimuli to which the neuron responded similarly and pairs to which the neuron responded very differently. These pairs can be formally described by equivalence constraints. Equivalence constraints are relations between pairs of datapoints, which indicate whether the points in the pair belong to the same category or not. We term a constraint *positive* when they points are known to originate from the same class, and *negative* belong to different classes. In this setting the goal of the algorithm is to learn a distance function that attempts to comply with the equivalence constraints.

This formalism allows us to combine information from a number of cells to improve the resulting characterization. Specifically, we combine equivalence constraints gathered from pairs of cells which have similar responses, and train a single distance function for both cells. Our results demonstrate that this approach improves prediction results of the "weaker" cell, and almost always improves the result of the "stronger" cell in each pair. Another interesting result of this formalism is the ability to classify stimuli based on the responses of the total recorded cortical cell ensemble. For some stimuli, the predictive performance based on the learned inter-stimuli distance was very good, whereas for other stimuli it was rather poor. These differences were correlated with the acoustic structure of the stimuli, partitioning them into narrowband and wideband stimuli.

3 Methods

Experimental setup Extracellular recordings were made in primary auditory cortex of nine halothane-anesthetized cats. Anesthesia was induced by ketamine and xylazine and maintained with halothane (0.25-1.5%) in 70% N_2O using standard protocols authorized by the committee for animal care and ethics of the Hebrew University - Haddasah Medical School. Single neurons were recorded using metal microelectrodes and an online spike sorter (MSD, alpha-omega). All neurons were well separated. Penetrations were performed over the whole dorso-ventral extent of the appropriate frequency slab (between about 2 and 8 kHz). Stimuli were presented 20 times using sealed, calibrated earphones at 60-80 dB SPL, at the preferred aurality of the neurons as determined using broad-band noise bursts. Sounds were taken from the Cornell Laboratory of Ornithology and have been selected as in [1]. Four stimuli, each of length 60-100 ms, consisted of a main tonal component with frequency and amplitude modulation and of a background noise consisting of echoes and unrelated components. Each of these stimuli was further modified by separating the main tonal component from the noise, and by further separating the noise into echoes and background. All possible combinations of these components were used here, in addition to a stylized artificial version that lacked the amplitude modulation of the natural sound. In total, 8 versions of each stimulus were used, and therefore each neuron had a dataset consisting of 32 datapoints. For more detailed methods, see Bar-Yosef et al. [1].

Data representation We used the first 60 ms of each stimulus. Each stimulus was represented using the first d real Cepstral coefficients. The real Cepstrum of a signal x was calculated by taking the natural logarithm of magnitude of the Fourier transform of x and then computing the inverse Fourier transform of the resulting sequence. In our experiments we used the first 21-30 coefficients. Neuronal responses were represented by creating Peri-Stimulus Time Histograms (PSTHs) using 20 repetitions recorded for each stimuli. Response duration was 100 ms.

Obtaining equivalence constraints over stimuli pairs The distances between responses were measured using a normalized χ^2 distance measure. All responses to both stimuli (40 responses in total) were superimposed to generate a single high-resolution PSTH. Then, this PSTH was non-uniformly binned so that each bin contained at least 10 spikes. The same bins were then used to generate the PSTHs of the responses to the two stimuli separately. For similar responses, we would expect that on average each bin in these histograms would contain 5 spikes. Formally, let N denote the number of bins in each histogram, and let r_1^i,r_2^i denote the number of spikes in the i'th bin in each of the two histograms respectively. The distance between pairs of histograms is given by: $\chi^2(r_1^i, r_2^i) = \sum_{i=1}^{N} \frac{(r_1^i - r_2^i)^2}{(r_1^i + r_2^i)/2}/(N-1)$.

In order to identify pairs (or small groups) of similar responses, we computed the normalized χ^2 distance matrix over all pairs of responses, and used the complete-linkage algorithm to cluster the responses into $8-12$ clusters. All of the points in each cluster were marked as similar to one another, thus providing positive equivalence constraints. In order to obtain negative equivalence constraints, for each cluster c_i we used the $2-3$ furthest clusters from it to define negative constraints. All pairs, composed of a point from cluster c_i and another point from these distant clusters, were used as negative constraints.

Distance learning method In this paper, we use the *DistBoost* algorithm [7], which is a semi-supervised boosting learning algorithm that learns a distance function using unlabeled datapoints and equivalence constraints. The algorithm boosts weak learners which are soft partitions of the input space, that are computed using the constrained Expectation-Maximization (cEM) algorithm [15]. The *DistBoost* algorithm, which is briefly summarized in 1, has been previously used in several different applications and has been shown to perform well [7, 17].

Evaluation methods In order to evaluate the quality of the learned distance function, we measured the correlation between the distances computed by our distance learning algorithm to those induced by the χ^2 distance over the responses. For each stimulus we measured the distances to all other stimuli using the learnt distance function. We then computed the rank-order (Spearman) correlation coefficient between these learnt distances in the stimulus domain and the χ^2 distances between the appropriate responses. This procedure produced a single correlation coefficient for each of the 32 stimuli, and the average correlation coefficient across all stimuli was used as the overall performance measure.

Parameter selection The following parameters of the *DistBoost* algorithm can be fine-tuned: (1) the input dimensionality d = 21-30, (2) the number of Gaussian models in each weak learner M = 2-4, (3) the number of clusters used to extract equivalence constraints C = 8-12, and (4) the number of distant clusters used to define negative constraints $numAnti$ = 2-3. Optimal parameters were determined separately for each of the 22 cells, based *solely* on the training data. Specifically, in the cross-validation testing we used a validation paradigm: Using the 31 training stimuli, we removed an additional datapoint and trained our algorithm on the remaining 30 points. We then validated its performance using the left out datapoint. The optimal cell specific parameters were determined using this approach.

Algorithm 1 The *DistBoost* Algorithm

Input:

Data points: $(x_1, ..., x_n),\ \ x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points: $(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)\ \ i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)
  $w_k = 1/n \qquad k = 1, \ldots, n$ (weights over data points)
- For $t = 1, .., T$
  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak hypothesis $\tilde{h}_t : \mathcal{X} \times \mathcal{X} \rightarrow [-1, 1]$ and define a weak distance function as $h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h}_t(x_i, x_j)\right) \in [0, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i h_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2}\ln(\frac{1+r_t}{1-r_t})$
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:
  $$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{h}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\alpha_t) & y_i = * \end{cases}$$
  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final distance function $\mathcal{D}(x_i, x_j) = \sum_{t=1}^{T} \alpha_t h_t(x_i, x_j)$

4 Results

Cell-specific distance functions We begin our analysis with an evaluation of the fitting power of the method, by training with the entire set of 32 stimuli (see Fig. 1). In general almost all of the correlation values are positive and they are quite high. The average correlation over all cells is 0.58 with $ste = 0.023$.

In order to evaluate the generalization potential of our approach, we used a Leave-One-Out (LOU) cross-validation paradigm. In each run, we removed a single stimulus from the dataset, trained our algorithm on the remaining 31 stimuli, and then tested its performance on the datapoint that was left out (see Fig. 3). In each histogram we plot the test correlations of a single cell, obtained when using the LOU paradigm over all of the 32 stimuli. As can be seen, on some cells our algorithm obtains correlations that are as high as 0.41, while for other cells the average test correlation is less then 0.1. The average correlation over all cells is 0.26 with $ste = 0.019$.

Not surprisingly, the train results (Fig. 1) are better than the test results (Fig. 3). Interestingly, however, we found that there was a significant correlation between the training performance and the test performance $C = 0.57, p < 0.05$ (see Fig. 2, left).

Boosting the performance of weak cells In order to boost the performance of cells with low average correlations, we constructed the following experiment: We clustered the responses of each cell, using the complete-linkage algorithm over the χ^2 distances with 4 clusters. We then used the $F_{\frac{1}{2}}$ score that evaluates how well two clustering partitions are in agreement with one another ($F_{\frac{1}{2}} = \frac{2*P*R}{P+R}$, where P denotes precision and R denotes recall.). This measure was used to identify pairs of cells whose partition of the stimuli was most similar to each other. In our experiment we took the four cells with the lowest

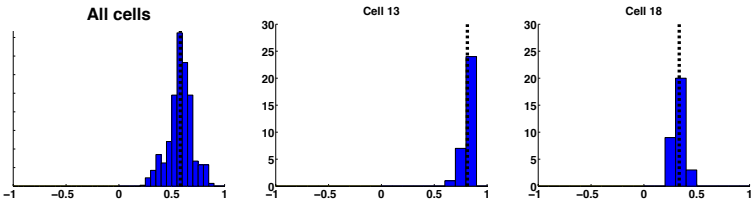

Figure 1: Left: Histogram of train rank-order correlations on the entire ensemble of cells. The rank-order correlations were computed between the learnt distances and the distances between the recorded responses for each single stimulus ($N = 22 * 32$). Center: train correlations for a "strong" cell. Right: train correlations for a "weak" cell. Dotted lines represent average values.

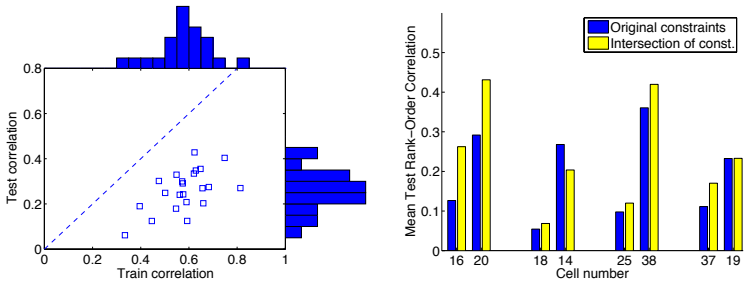

Figure 2: Left: Train vs. test cell specific correlations. Each point marks the average correlation of a single cell. The correlation between train and test is 0.57 with $p = 0.05$. The distribution of train and test correlations is displayed as histograms on the top and on the right respectively. Right: Test rank-order correlations when training using constraints extracted from each cell separately, and when using the intersection of the constraints extracted from a pair of cells. This procedure always improves the performance of the weaker cell, and usually also improves the performance of the stronger cell

performance (right column of Fig 3), and for each of them used the $F_{\frac{1}{2}}$ score to retrieve the most similar cell. For each of these pairs, we trained our algorithm once more, using the constraints obtained by intersecting the constraints derived from the two cells in the pair, in the LOU paradigm. The results can be seen on the right plot in Fig 2. On all four cells, this procedure improved LOUT test results. Interestingly and counter-intuitively, when training the better performing cell in each pair using the intersection of its constraints with those from the poorly performing cell, results deteriorated only for one of the four better performing cells.

Stimulus classification The cross-validation results induced a partition of the stimulus space into narrowband and wideband stimuli. We measured the predictability of each stimulus by averaging the LOU test results obtained for the stimulus across all cells (see Fig. 4). Our analysis shows that wideband stimuli are more predictable than narrowband stimuli, despite the fact that the neuronal responses to these two groups are not different as a whole. Whereas the non-linearity in the interactions between narrowband and wideband stimuli has already been noted before [9], here we further refine this observation by demonstrating a significant difference between the behavior of narrow and wideband stimuli with respect to the predictability of the similarity between their responses.

5 Discussion

In the standard approach to auditory modeling, a linear or weakly non-linear model is fitted to the data, and neuronal properties are read from the resulting model. The usefulness of this approach is limited however by the weak predictability of A1 responses when using such models. In order to overcome this limitation, we reformulated the problem of char-

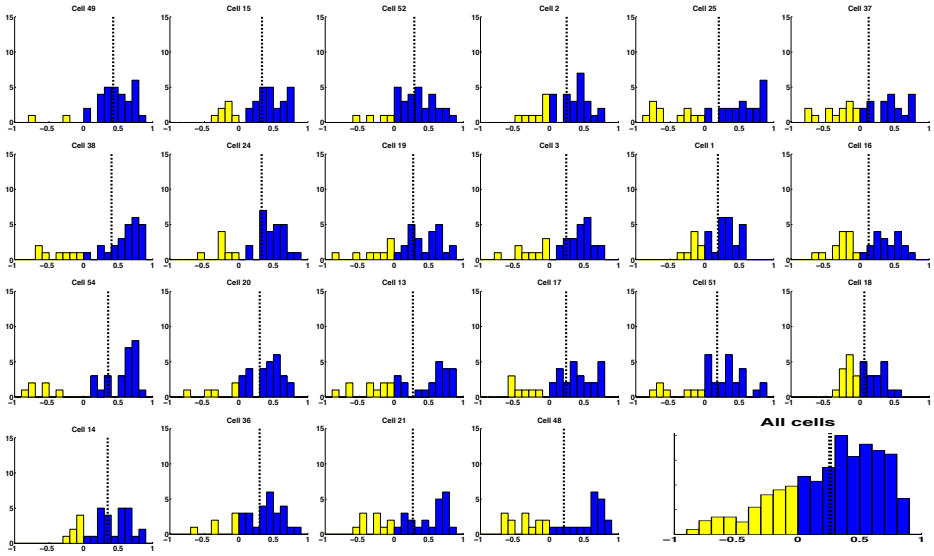

Figure 3: Histograms of cell specific test rank-order correlations for the 22 cells in the dataset. The rank-order correlations compare the predicted distances to the distances between the recorded responses, measured on a single stimulus which was left out during the training stage. For visualization purposes, cells are ordered (columns) by their average test correlation per stimulus in descending order. Negative correlations are in yellow, positive in blue.

acterizing neuronal responses of highly non-linear neurons. We use the neural data as a guide for training a highly non-linear distance function on stimulus space, which is compatible with the neural responses. The main result of this paper is the demonstration of the feasibility of this approach.

Two further results underscore the usefulness of the new formulation. First, we demonstrated that we can improve the test performance of a distance function by using constraints on the similarity or dissimilarity between stimuli derived from the responses of multiple neurons. Whereas we expected this manipulation to improve the test performance of the algorithm on the responses of neurons that were initially poorly predicted, we found that it actually improved the performance of the algorithm also on neurons that were rather well predicted, although we paired them with neurons that were poorly predicted. Thus, it is possible that intersecting constraints derived from multiple neurons uncover regularities that are hard to extract from individual neurons.

Second, it turned out that some stimuli consistently behaved better than others across the neuronal population. This difference was correlated with the acoustic structure of the stimuli: those stimuli that contained the weak background component (wideband stimuli) were generally predicted better. This result is surprising both because background component is substantially weaker than the other acoustic components in the stimuli (by as much as 35-40 dB). It may mean that the relationships between physical structure (as characterized by the Cepstral parameters) and the neuronal responses becomes simpler in the presence of the background component, but is much more idiosyncratic when this component is absent. This result underscores the importance of interactions between narrow and wideband stimuli for understanding the complexity of cortical processing.

The algorithm is fast enough to be used in near real-time. It can therefore be used to guide real experiments. One major problem during an experiment is that of stimulus selection: choosing the best set of stimuli for characterizing the responses of a neuron. The distance functions trained here can be used to direct this process. For example, they can be used to

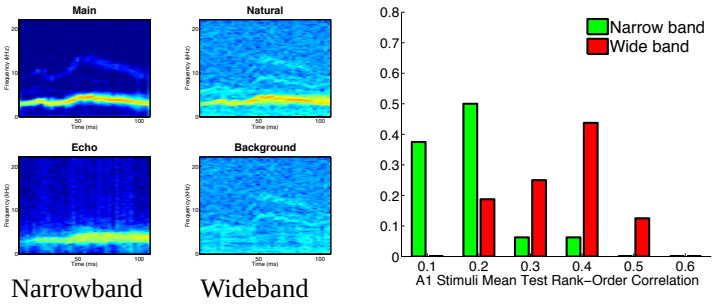

Figure 4: Left: spectrograms of input stimuli, which are four different versions of a single natural bird chirp. Right: Stimuli specific correlation values averaged over the entire ensemble of cells. The predictability of wideband stimuli is clearly better than that of the narrowband stimuli.

find surprising stimuli: either stimuli that are very different in terms of physical structure but that would result in responses that are similar to those already measured, or stimuli that are very similar to already tested stimuli but that are predicted to give rise to very different responses.

References

[1] O. Bar-Yosef, Y. Rotman, and I. Nelken. Responses of Neurons in Cat Primary Auditory Cortex to Bird Chirps: Effects of Temporal and Spectral Context. *J. Neurosci.*, 22(19):8619–8632, 2002.

[2] D. T. Blake and M. M. Merzenich. Changes of AI Receptive Fields With Sound Density. *J Neurophysiol*, 88(6):3409–3420, 2002.

[3] G. Chechik, A. Globerson, M.J. Anderson, E.D. Young, I. Nelken, and N. Tishby. Group redundancy measures reveal redundancy reduction in the auditory pathway. In *NIPS*, 2002.

[4] R. C. deCharms, D. T. Blake, and M. M. Merzenich. Optimizing Sound Features for Cortical Neurons. *Science*, 280(5368):1439–1444, 1998.

[5] D. A. Depireux, J. Z. Simon, D. J. Klein, and S. A. Shamma. Spectro-Temporal Response Field Characterization With Dynamic Ripples in Ferret Primary Auditory Cortex. *J Neurophysiol*, 85(3):1220–1234, 2001.

[6] J. J. Eggermont, P. M. Johannesma, and A. M. Aertsen. Reverse-correlation methods in auditory research. *Q Rev Biophys.*, 16(3):341–414, 1983.

[7] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *ICML*, 2004.

[8] N. Kowalski, D. A. Depireux, and S. A. Shamma. Analysis of dynamic spectra in ferret primary auditory cortex. I. Characteristics of single-unit responses to moving ripple spectra. *J Neurophysiol*, 76(5):3503–3523, 1996.

[9] L. Las, E. A. Stern, and I. Nelken. Representation of Tone in Fluctuating Maskers in the Ascending Auditory System. *J. Neurosci.*, 25(6):1503–1513, 2005.

[10] C. K. Machens, M. S. Wehr, and A. M. Zador. Linearity of Cortical Receptive Fields Measured with Natural Sounds. *J. Neurosci.*, 24(5):1089–1100, 2004.

[11] L. M. Miller, M. A. Escabi, H. L. Read, and C. E. Schreiner. Spectrotemporal Receptive Fields in the Lemniscal Auditory Thalamus and Cortex. *J Neurophysiol*, 87(1):516–527, 2002.

[12] I. Nelken. Processing of complex stimuli and natural scenes in the auditory cortex. *Current Opinion in Neurobiology*, 14(4):474–480, 2004.

[13] Y. Rotman, O. Bar-Yosef, and I. Nelken. Relating cluster and population responses to natural sounds and tonal stimuli in cat primary auditory cortex. *Hearing Research*, 152(1-2):110–127, 2001.

[14] M. Sahani and J. F. Linden. How linear are auditory cortical responses? In *NIPS*, 2003.

[15] N. Shental, A. Bar-Hilel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. In *NIPS*, 2003.

[16] F. E. Theunissen, K. Sen, and A. J. Doupe. Spectral-Temporal Receptive Fields of Nonlinear Auditory Neurons Obtained Using Natural Sounds. *J. Neurosci.*, 20(6):2315–2331, 2000.

[17] C. Yanover and T. Hertz. Predicting protein-peptide binding affinity by learning peptide-peptide distance functions. In *RECOMB*, 2005.

Chapter 8

Epilogue

In this thesis we have presented algorithms for learning distance functions and considered various applications of these algorithms in a wide variety of application domains including data clustering, image retrieval, data classification, prediction of immunological interactions and the analysis of neuronal data recordings. The main contributions of this thesis can be summarized as follows:

- **Distance Learning Algorithms** We present three novel distance learning algorithms:

    1. **Relevant Component Analysis (RCA)** - a Mahalanobis metric learning algorithm which is trained using positive equivalence constraints. This algorithm can be derived both from an information theoretic criterion and from a maximum-likelihood criterion under Gaussian assumptions. The algorithm is very efficient and its computation only requires a single matrix inversion.
    2. ***DistBoost*** **-** a boosting based algorithm for learning non-linear distance functions which is trained using both positive and negative equivalence constraints. The algorithm can be shown to learn highly non-linear distance functions via a semi-supervised boosting over product-space hypotheses.
    3. ***KernelBoost*** **-** a variant of the *DistBoost* algorithm which can learn kernel functions that can be used in any kernel-based classifier. The kernels can be learned using very small sample sizes and can also naturally be used in a learning-to-learn scenario in which information among related classes can be transferred or shared.

- **Applications of Distance learning** Using the above three algorithms, we present their applications in the following application domains:

– **Data clustering** - we show that learning a distance function can significantly improve the performance of various clustering algorithms which are distance based such as the average-linkage algorithm. Improvements in clustering are shown for both the RCA algorithm and the *DistBoost* algorithm.

– **Image retrieval** - We show that a similarity based image retrieval system can be significantly improved by learning the similarity function with which it compares image pairs. Using very simple and standard feature representations of images we show that both for facial images and for images from a database of animal images collected from the web, both the RCA and *DistBoost* algorithm can be used to obtain retrieval results which are superior to standard canonical distance functions such as the Euclidean distance.

– **Data classification** - We show that the *KernelBoost* algorithm can be used to learn kernels that outperform standard off-the-shelf kernels when trained on very small samples consisting of $3-10$ datapoints from each class. This is obtained by training a kernel function for the two classes with small amounts of data using a learning-to-learn scenario in which training data from related classes is also used in the training process.

– **Predicting protein-peptide binding in the immune system** - We suggest formalizing the problem of predicting whether Major Histocompatability Complex (MHC) proteins will bind to a set of peptides as a distance learning problem. More specifically we show that a distance function between peptides can be used to predict whether they will bind to a specific protein. We suggest training a single distance function over a family of related proteins. This enables transfer of information between related classes, which results in significant performance enhancement over previously suggested methods that are separately trained for each MHC protein.

  We have also recently shown that this approach can be used to classify MHC proteins into *Supertypes* - sets of proteins which are known to bind to similar peptides (Hertz and Yanover, 2006a). Using this approach we suggest two different ways of learning distance functions between MHC proteins - a peptide-based approach and a protein-based approach. Comparison with experimental classification of proteins into supertypes shows that our method can compete very successfully with previous supertype classifications.

– **Analysis of neuronal data** - We suggest a novel approach to the characterization of neuronal data recordings which is based on learning a distance function over the input space that is trained using

information about the neuron's responses to the input stimuli. This distance function can then be used to predict whether the response to a novel input stimuli is similar to the responses to other stimuli which have been previously presented to the neuron. This novel approach shows promising results and we hope that in the future we will be able to incorporate it as a useful tool to guide neuronal recording experiments, where the algorithm will be trained in real-time and used to suggest which novel stimuli should be presented to the neuron currently being recorded.

Bibliography

Aggarwal, C. (2003). Towards systematic design of distance functions for data mining applications. In *In Proc. 9th Intl. Conf. on Knowledge Discovery and Data Mining (KDD).*, pages 9–19.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory (ICDT)*.

Aslam, J. A. and Frost, M. (2003). An information-theoretic measure for document similarity. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 449–450. ACM Press.

Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. (2004). Boostmap: A method for efficient approximate similarity rankings. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Balcan, M.-F. and Blum, A. (2006). On a theory of learning with similarity functions. In *International Conference on Machine Learning (ICML)*.

Banerjee, A., Dhillon, I., Ghosh, J., and Sra, S. (2003). Generative model-based clustering of directional data. In *In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 19–28.

Bansal, N., Blum, A., and Chawla, S. (2002). Correlation clustering. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 238–247.

Bar-hillel, A. and Weinshall, D. (2003). Learning with equivalence constraints, and the relation to multiclass classification. In *The Sixteenth Annual Conference On Learning Theory (COLT)*.

Bar-Hillel, A. and Weinshall, D. (2006). Learning distance function by coding similarity. Technical report, The Hebrew Uinversity of Jerusalem Israel.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2003). Learning distance functions using equivalence relations. In *International Conference on Machine Learning (ICML)*, pages 11–18.

Bar-Hillel, A., Hertz, T., and Weinshall, D. (2005a). Efficient learning of relational object class models. In *International Conference on Computer Vision (ICCV)*.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2005b). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, **6**, 937–965.

Bar-Hillel, A., Hertz, T., and Weinshall, D. (2005c). Object class recognition by boosting a part-based model. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Bar-Yosef, O. and Nelken, I. (2006). The effects of background noise on the neuralresponses to natural sounds in cat primary auditory cortex. Submitted to J. Neurophysiology.

Basu, S., Banerjee, A., and Mooney, R. (2002). Semi-supervised clustering by seeding. In *International Conference on Machine Learning (ICML)*, pages 19–26.

Basu, S., Bilenko, M., and Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *KDD04*, pages 59–68.

Baxter, J. (1995). Learning internal representations. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

Baxter, J. (1997). Theoretical models of learning to learn. In T. Mitchell and S. Thrun, editors, Learning to Learn. Kluwer, Boston, 1997.

Bellman, R. (1961). *Adaptive Control Processes*. Princeton University Press.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Ben-David, S. and Schuller, R. (2003). Exploiting task relatedness for multitask learning. In *Proceedings of COLT*.

Berg, A. C., Berg, T. L., and Malik, J. (2005). Shape matching and object recognition using low distortion correspondence. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is nearest neighbors meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT)*.

Bie, T. D., Momma, M., and Cristianini, N. (2003a). Efficiently learning the metric with side-information. In *Algorithmic Learning Theory, 14th International Conference, ALT 2003, Sapporo, Japan, October 2003, Proceedings*, volume 2842 of *Lecture Notes in Artificial Intelligence*, pages 175–189. Springer.

Bie, T. D., Suykens, J., and Moor, B. D. (2003b). Learning from general label constraints. In *Proceedings of the joint IAPR international workshops on Syntactical and Structural Pattern Recognition (SSPR 2004) and Statistical Pattern Recognition (SPR 2004)*, Lisbon.

Bilenko, M., Basu, S., and Mooney, R. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, pages 81–88.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning (ICML)*, pages 19–26. Morgan Kaufmann, San Francisco, CA.

Boreczky, J. S. and Rowe, L. A. (1996). Comparison of video shot boundary detection techniques. *SPIE Storage and Retrieval for Still Images and Video Databases IV*, **2664**, 170–179.

Bousquet, O. and Herrmann, D. J. L. (2002). On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems (NIPS)*.

Boykov, Y., Veksler, O., and Zabih, R. (1999). Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision (ICCV)*, pages 377–384.

Caruana, R. (1996). Algorithms and applications for multitask learning. In *International Conference on Machine Learning (ICML)*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, **28**, 41–75.

Chang, H. and Yeung, D. (2004). Locally linear metric adaptation for semi-supervised clustering. In *International Conference on Machine Learning (ICML)*.

Chang, H. and Yeung, D. (2005a). Kernel-based metric adaptation with pairwise constraints. In *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 721–730.

Chang, H. and Yeung, D. (2005b). Stepwise metric adaptation based on semi-supervised learning for boosting image retrieval performance. In *British Machine Vision Conference (BMVC)*.

Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, **46**(1–3), 131–159.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Cohn, D., Caruana, R., and McCallum, A. (2003). Semi-supervised clustering with user feedback.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley and Sons.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEE Transactions on Information Theory*, **IT-13(1)**, 21–27.

Cozman, F. G., Cohen, I., and Cirelo, M. C. (2003). Semi-supervised learning of mixture models and bayesian networks. In *International Conference on Machine Learning (ICML)*.

Crammer, K., Keshet, J., and Singer, Y. (2002). Kernel design using boosting. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. (2001). On kernel-target alignment. In *Advances in Neural Information Processing Systems (NIPS)*.

d'Alche Buc, F., Grandvalet, Y., and Ambroise, C. (2002). Semi-supervised marginboost. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14.

Demiriz, A., Embrechts, M., and Bennet, K. P. (1999). Semi-supervised clustering using genetic algorithms. In *Artificial Neural Networks in Engineering (ANNIE'99)*.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JRSSB*, **39**, 1–38.

Dietterich, T. (2003). Machine learning. In *Nature Encyclopedia of Cognitive Science*. Macmillan - London.

Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, **2**, 263–286.

Domeniconi, C. and Gunopulos, D. (2001). Adaptive nearest neighbor classification using support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience Publication.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis*. Cambridge University Press.

Ferencz, A., Learned-Miller, E., and Malik, J. (2005). Building a classification cascade for visual identification from one example. In *International Conference on Computer Vision (ICCV)*, pages 286–293.

Fink, M. (2004). Object classification from a single example utilizing class relevance pseudo-metrics. In *Advances in Neural Information Processing Systems (NIPS)*.

Fink, M., Shwatz-Shalev, S., Singer, Y., and Ullman, S. (2006). Online multiclass learning by interclass hypothesis sharing. In *International Conference on Machine Learning (ICML)*.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, 179–188.

Fukunaga, K. (1990). *Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition.

Gdalyahu, Y. and Weinshall, D. (1999). Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **21(12)**.

Gdalyahu, Y., Weinshall, D., and Werman, M. (2001). Self organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **23(10)**, 1053–1074.

Getz, G., Shental, N., and Domany, E. (2005). Semi-supervised learning - a statistical pyhsics approach. In *Proceedings of workshop on "Learning with partially classified training data" - ICML 2005.*

Gilad-Bachrach, R., Navot, A., and Tishby, N. (2004). Margin based feature selection - theory and algorithms. In *International Conference on Machine Learning (ICML)*.

Globerson, A. and Roweis, S. (2005). Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 451–458.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004). Neighbourhood component analysis. In *Advances in Neural Information Processing Systems (NIPS)*.

Grandvalet, Y., d'Alche Buc, F., and Ambroise, C. (2001). Boosting mixture models for semi supervised learning. In *ICANN 2001,Vienne, Austria*, pages 41–48. Springer.

Grauman, K. and Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L., and (editors) (2006). *Feature extraction, foundations and applications*. Springer.

Hamming, R. W. (1950). Error-detecting and error-correcting codes. *Bell System Technical Journal*, **29(2)**, 147–160.

Har-Peled, S., Roth, D., and Zimak, D. (2002). Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems (NIPS)*.

Hastie, T. and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification and regression. In *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 409–415.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer-Verlag.

Hertz, T. and Yanover, C. (2006a). Identifying hla supertypes by learning distance functions. In *European Conference on Computational Biology (ECCB)*.

Hertz, T. and Yanover, C. (2006b). Pepdist: A new framework for protein-peptide binding prediction based on learning peptide distance functions. *BMC Bioinformatics*, **7(Suppl 1):S3**.

Hertz, T., Bar-Hillel, A., and Weinshall, D. (2004a). Boosting margin based distance functions for clustering. In *International Conference on Machine Learning (ICML)*.

Hertz, T., Bar-Hillel, A., and Weinshall, D. (2004b). Learning distance functions for image retrieval. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hertz, T., Hillel, A. B., and Weinshall, D. (2006). Learning a kernel function for classification with small training samples. In *International Conference on Machine Learning (ICML)*.

Jacobs, D. W., Weinshall, D., and Gdalyahu, Y. (2000). Classification with non metric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **22(6)**, 583–600.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning (ICML)*.

Kamvar, S. D., Klein, D., and Manning, C. D. (2003). Spectral learning. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

Katayama, N. and Satoh, S. (2001). Distinctiveness sensitive nearest neighbor search for efficient similarity retrieval of multimedia information. In *International Conference on Data Engineering (ICDE)*.

Kemp, C., Bernstein, A., and Tenenbaum, J. (2005). A generative theory of similarity. In *XXVII Anuual Meeting of the Cognitive Science Society (CogSci)*.

Klein, D., Kamvar, S., and Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *International Conference on Machine Learning (ICML)*.

Kondor, R. and Jebara, T. (2003). A kernel between sets of vectors. In *International Conference on Machine Learning (ICML)*.

Kwok, J. T. and Tsang, I. W. (2003). Learning with idealized kernels. In *International Conference on Machine Learning (ICML)*, pages 400–407.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2002). Learning the kernel matrix with semi-definite programming. In *International Conference on Machine Learning (ICML)*, pages 323–330.

Lange, T., Law, M. H., Jain, A. K., and Buhmann, J. (2005). Learning with constrained and unlabelled data. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 20–25.

Langley, P. (1994). Selection of relevant features in machine learning. In *proceedings of the AAAI symposium on relevance, New Orleans*. LA:AAAI press.

Law, M., Topchy, A., and Jain, A. K. (2004). Clustering with soft and group constraints. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, And Statistical Pattern Recognition (S+SSPR 2004)*, pages 662–670.

Law, M., Topchy, A., and Jain, A. K. (2005). Model-based clustering with probabilistic constraints. In *Proceedings of SIAM Data Mining*, pages 641–645.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Leslie, C., E, E. E., and Noble, W. S. (2002). The spectrum kernel: a string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 564–75.

Leslie, C., Eskin, E., Weston, J., and Noble, W. (2003). Mismatch string kernels for svm protein classification. In *Advances in Neural Information Processing Systems (NIPS)*.

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, **163(4)**, 845–848.

Lin, D. (1998). An information theoretic definition of similarity. In *International Conference on Machine Learning (ICML)*.

Linsker, R. (1989). An application of the principle of maximum information preservation to linear systems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 186–194.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *JMLR*, **2**, 419–444.

Lowe, D. (1995). Similarity metric learning for variable kernel classifier. *Neural Computation*, **7(1)**, 72–85.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60(2)**, 91–110.

Lu, Z. and Leen, T. (2005). Semi-supervised learning with penalized probabilistic clustering. In *NIPS*, pages 849–856.

Lyu, S. (2005). Mercer kernels for object recognition with local features. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, volume 2, pages 49–55.

Mahamud, S. and Hebert, M. (2003a). Minimum risk distance measure for object recognition. In *International Conference on Computer Vision (ICCV)*.

Mahamud, S. and Hebert, M. (2003b). The optimal distance measure for object detection. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages I: 248–255.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 512–518.

Miller, D. and Uyar, S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 571–578.

Minka, T. and Picard, R. (1997). Learning how to learn is learning with point sets. Unpublished manuscript. Available at http://wwwwhite.media.mit.edu/ tpminka/papers/learning.html.

Moreno, P., Ho, P., and Vasconcelos, N. (2003). A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *Advances in Neural Information Processing Systems (NIPS)*.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98*, pages 792–799, Madison, US. AAAI Press, Menlo Park, US.

Ong, C. S., Smola, A. J., and Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, **6**, 1043–1071.

Peleg, S., Werman, M., and Rom, H. (1989). A unified approach to the change of resolution: Space and gray-level. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **11**, 739–742.

Phillips, P. J. (1999). Support vector machines applied to face recognition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 11.

Rosales, R. and Fung, G. (2006). Learning sparse metrics via linear programming. In *The 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 367–373. ACM Press.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**(5500), 2323–2326.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, **40**(2), 99–121.

Salton, G., Wong, A., and Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM*, **18(11)**, 613–620.

Schapire, R. E. and Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, **37**(3), 297–336.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann.

Schervish, M. J. (1995). *Theory of statistics*. Springer-Verlag New York.

Schultz, M. and Joachim, T. (2003). Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*.

Shalev-Shwartz, S., Singer, Y., and Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. In *International Conference on Machine Learning (ICML)*.

Shashua, A. and Hazan, T. (2005). Algebraic set kernels with application to inference over local image representations. In *Advances in Neural Information Processing Systems (NIPS)*.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Shental, N., Hertz, T., Weinshall, D., and Pavel, M. (2002). Adjustment learning and relevant component analysis. In *European Conf. on Computer Vision (ECCV)*, volume 4.

Shental, N., Bar-Hillel, A., Hertz, T., and Weinshall, D. (2004a). Computing gaussian mixture models with em using equivalence constraints. In *Advances in Neural Information Processing Systems (NIPS)*, pages 465–472.

Shental, N., Zomet, A., Hertz, T., and Weiss, Y. (2004b). Pairwise clustering and graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 185–192.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **22**(8), 888–905.

Short, R. D. and Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory*, **27(5)**, 622–627.

Srebro, N. and Ben-David, S. (2006). Learning bounds for support vector machines with learned kernels. In *The 19th Annual Conference on Learning Theory (COLT)*.

Szummer, M. and Jaakkola, T. (2002). Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14. MIT Press.

Tenenbaum, J. and Freeman, W. (2000). Separating style and content with bilinear models. *Neural Computation*, **12**(6), 1247–1283.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**(5500), 2319–2323.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 640–646.

Thrun, S. and Pratt, L. (1998). *Learning To Learn*. Kluwer Academic Publishers, Boston, MA.

Tsang, I. W., Cheung, P.-M., and Kwok, J. T. (2005). Kernel relevant component analysis for distance metric learning. In *Proceedings of International Joint Conference on Neural Networks*.

Tversky, A. (1977). Features of similarity. *Psychological review*, **84(4)**, 327–352.

Vapnik, V. (1998). *The nature of statistical learning theory*. Wilay Chichster, GB.

Vert, J., Thurman, R., and Noble, W. S. (2005). Kernels for gene regulatory regions. In *Advances in Neural Information Processing Systems (NIPS)*.

Vincent, P. and Bengio, Y. (2002). K-local hyperplane and convex distance nearest neighbor algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14.

Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. (2001). Constrained K-means clustering with background knowledge. In *International Conference on Machine Learning (ICML)*, pages 577–584. Morgan Kaufmann, San Francisco, CA.

Wallraven, C., Caputo, B., and Graf, A. (2003). Recognition with local features: the kernel recipe. In *International Conference on Computer Vision (ICCV)*.

Weinberger, K., Blitzer, J., and Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*, Cambridge, MA. MIT Press.

Weiner, I., Hertz, T., Nelken, I., and Weinshall, D. (2005). Analyzing auditory neurons by learning distance functions. In *Advances in Neural Information Processing Systems (NIPS)*.

Wettschereck, D., Aha, D., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificail intelligence review*, **11**, 273–314.

Wolf, L. (2006). Learning using the born rule. Technical Report MIT-CSail-TR-2006-036.

Wolf, L. and Shashua, A. (2003). Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, **4**, 913–931.

Wu, F., Zhou, Y., and Zhang, C. (2004). Self-enhanced relevant component analysis with side-information and unlabeled data. In *International Joint Conference on Neural Networks 2004 (IJCNN 04)*.

Wu, G., Chang, E. Y., and Panda, N. (2005). Formulating distance functions via the kernel trick. In *ACM International Conference on Data Mining and Knowledge Discovery (KDD)*.

Xing, E., Ng, A., Jordan, M., and Russell, S. (2002). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15. The MIT Press.

Yan, R., Zhang, J., Yang, J., and Hauptmann, A. (2004). A discriminative learning framework with pairwise constraints for video object classification. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 02, pages 284–291.

Yanover, C. and Hertz, T. (2005). Predicting protein-peptide binding affinity by learning peptide-peptide distance functions. In *RECOMB*.

Yu, K., Tresp, V., and Schwaighofer, A. (2005). Learning gaussian processes from multiple tasks. In *Proceedings of the 22 nd International Conference on Machine Learning, Bonn, Germany*.

Yu, S. and Shi, J. (2001). Grouping with bias. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhang, H. and Malik, J. (2003). Learning a discriminative classifier using shape context distances. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Z., Kwok, J. T., and Yeung, D.-Y. (2003). Parametric distance metric learning with label information. In *The Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1450–1452.

Zhang, Z., Kwok, J., and Yeung, D. (2006). Model-based transductive learning of the kernel matrix. *Machine Learning*, **63(1)**, 69–101.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schlkopf, B. (2003). Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report Computer Sciences TR 1530 University of Wisconsin Madison.

Zhu, X., Ghahramani, Z., and Laferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*.

Zien, A. and Ong, C. S. (2006). An automated combination of sequence motif kernels for predicting protein subcellular localization.

תוכן העניינים

שלהם תגובה שונה צריך להיות גדול, ואילו המרחק בין גירויים המניבים תגובה דומה צריך להיות קטן. אנו מראים כיצד ניתן ללמוד פונקציות מרחק אלו תוך שימוש בכמות מוגבלת מאד של מידע צדדי. הנתונים שעליהם מופעלת השיטה הם רישומים מתאים בקורטקס השמיעתי הראשוני (primary auditory cortex) של חתולים מורדמים בתגובה ל – 32 גירויים שהופקו מצלילים טבעיים (ציוצי ציפורים). לכל תא אוסף של זוגות של גירויים נבחרים כך שהתגובה לכל זוג תהיה דומה מאד או שונה מאד. פונקצית המרחק אומנה תוך שימוש בזוגות אלו. פונקציות המרחק שנלמדו מציגות יכולת הכללה טובה בחיזוי המרחקים בין גירויי מבחן לגירויי האימון.

אלו הינם כיום פופולריים ביותר, בין השאר בשל הביצועים הטובים שהם מציגים במגוון רחב של אפליקציות כגון קלסיפיקציה ורגרסיה, וכן בשל החסמים התיאורטיים אשר הוכחו לגבי יכולת ההכללה של אלגוריתמים אלו. אלגוריתמים אלו עושים שימוש בפונקצית גרעין (kernel function) אשר בעזרתה הם מודדים את הדמיון בין העצמים אשר עליהם הם פועלים. בחירת פונקצית הגרעין הינה חשובה ביותר, שכן במקרים רבים הביצועים של אלגוריתמים אלו תלויים באופן מהותי בפונקצית הגרעין שבה נעשה שימוש. נכון להיום לא קיימת שיטה סדורה לבחירת פונקצית הגרעין הנכונה, ופעמים רבות זו נבחרת באופן אמפירי ע"י מדידת ביצועים על מדגם האימון. בשנים האחרונות החלו להופיע אלגוריתמים ללמידת פונקציות הגרעין מן הנתונים עצמם. למרות שהוצעו כמה אלגוריתמים כאלו, אין כיום הוכחה ברורה כי לאלגוריתמים אלו יש עדיפות ברורה על פני פונקציות גרעין סטנדרטיות. בעבודה זו, אנו מציעים אלגוריתם ללמידת פונקציות גרעין אשר יכול ללמוד פונקציות כאלו ממדגמי אימון קטנים ביותר הכוללים 2-10 דוגמאות. אנו מראים כי כאשר מדגים האימון קטן מאד, לאלגוריתם שלומד את פונקצית הגרעין יש עדיפות מובהקת על פני שימוש בפונקציות גרעין סטנדרטיות.

ד. **חיזוי הקישור בין חלבונים לפפטידים במערכת החיסונית** – תהליך הזיהוי של פפטידים (רצפים קצרים של חומצות אמינו) זרים (פתוגניים) מתחיל כאשר פפטידים אלו נקשרים לחלבונים במערכת החיסונית הנקראים Major Histocompatability Complexes (או MHC's). חלבוני MHC הם מאד ספציפיים וכל אחד מהם קושר אוסף קטן של פפטידים. לכל אדם יש כ – 12 חלבוני MHC שונים במערכת החיסונית. מספר החלבונים השונים המוכרים באוכלוסיה מונה כמה אלפים. הפפטידים אשר נקשרים לחלבונים אלו הם לרוב באורך 9 חומצות אמינו. משיקולים אלו ברור כי לא ניתן לסרוק את כל מרחב הפפטידים והחלבונים באופן ניסוי, ולכן נעשה שימוש בשיטות חישוביות שונות לחיזוי הקישור. פיתוח שיטות חישוביות לחיזוי הקישור בין פפטיד לחלבון MHC הינו בעל חשיבות לפיתוח חיסונים וטיפול במחלות כגון איידס וסרטן. בעבודה זו אנו מציעים גישה חדשנית לחיזוי הקישור אשר מבוססת על למידת פונקצית מרחק בין פפטידים. כדי ללמוד פונקציות מרחק אלו אנו מנסחים את הבעיה כבעיה של למידה מפוקחת למחצה, שבה המידע הצדדי ניתן על ידי אילוצי שקילות. באופן ספציפי אנו מציעים להשתמש באלגוריתם DistBoost ללמידת פונקציות מרחק אלו. השיטה המוצעת מראה שיפורי ביצועים משמעותיים לעומת שיטות חיזוי אחרות. אחד היתרונות של השיטה המוצעת הוא שהיא מאפשרת ללמוד פונקצית חיזוי גם עבור חלבונים אשר עבורם יש מעט מאד מידע מתויג לצורך של האימון. במקרים אלו, היתרון היחסי של השיטה בהשוואה למתחריה בולט עוד יותר.

ה. **ניתוח מידע עצבי במערכת השמיעתית** – אנו מציעים שיטה חדשנית לניתוח מידע עצבי ולאפיון תאי עצב סנסוריים. אחת המטרות באיפיון תאי עצב סנסוריים היא בזיהוי המימדים במרחב הגירויים שהנוירון רגיש אליהם (כלומר שעבורם קיים שינוי משמעותי בתגובה העצבית). אנחנו מנסחים את הבעיה כבעיה של למידת גאומטריה על מרחב הגירויים אשר תואמת לדמיון הנמדד בין התגובות: המרחק בין גירויים

שני הפרקים הראשונים של העבודה כוללים מבוא מקיף לתחום של פונקציות מרחק, ולמידת פונקציות מרחק, וכן רקע נוסף בתחום של למידה מפוקחת למחצה (Semi-supervised learning). שאר הפרקים מכילים את המאמרים אשר עליהם מבוססת עבודה זו, הכוללים בין השאר את האלגוריתמים המתוארים בעבודה, וכן את היישומים השונים בהם נעשה שימוש באלגוריתמים אלו.

היישומים השונים של אלגוריתמי למידת פונקציות המרחק המוצגים בעבודה זו כוללים בין השאר:

א. **אחזור תמונות (Imag Retrieval)** – במערכת אחזור תמונות המשתמש מגיש תמונה למערכת, ועל המערכת לאחזר למשתמש אוסף של תמונות הדומות לתמונה זו. בכדי לבחור אלו תמונות לאחזר עבור שאילתא מסוימת, על המערכת למדוד את הדמיון בין התמונה המקורית שהגיש המשתמש, לבין התמונות השמורות במאגר המידע של המערכת. הבעיה של הגדרת פונקצית דמיון טובה בין תמונות היא בעיה קשה, אשר רחוקה מלהיות פתורה. עיקרה של הבעיה טמון בעובדה שבכדי להגדיר מדד דמיון טוב בין תמונות נדרשת הבנה של התמונה, או לחילופין יכולת למצוא ייצוג טוב עבור תמונות - בעיה קשה בפני עצמה אשר גם לה אין פיתרון טוב במיוחד. יחד עם זאת, קיימות מערכות אחזור תמונות אשר עובדות ע"י הפעלת פונקציות מרחק פשוטות למדי אשר מודדות את המרחק בין תמונת השאילתא לשאר התמונות במאגר המידע. בעבודה זו אנו מראים כי השימוש באלגוריתמים ללמידת פונקצית המרחק, יכול לשפר באופן משמעותי את הביצועים של מערכות אחזור מידע.

ב. **צברור מידע (data clustering)** – אלגוריתמים לצברור מידע מאפשרים לקבץ אוסף של עצמים לקבוצות כאשר הדמיון בין העצמים השייכים לאותה הקבוצה גדול מהדמיון בין עצמים השייכים לקבוצות שונות. בעית הצברור הינה בעיה שאינה מוגדרת היטב (ill-posed). מאחר שמדובר באלגוריתמים לא מפוקחים, לא קיימת הגדרה ברורה מהו הדמיון בין עצמים, ומעבר לכך, מהו הפיתרון הנכון של הבעיה. יחד עם זאת, קיימים אלגוריתמים רבים לצברור מידע, ונעשה בהם שימוש רחב בתחומי מחקר רבים הכוללים ביולוגיה, ניתוח טקסט, עיבוד תמונה וכד'. אלגוריתמי צברור רבים אינם פועלים ישירות על העצמים, אלא נעזרים במטריצה של מרחקים (או דמיון) בין העצמים, אשר בעזרתה הם מכריעים כיצד לקבץ את הנתונים לצבירים. הביצועים של אלגוריתמים אלו תלויים באופן משמעותי באיכות של מטריצת המרחקים בין העצמים. בעבודה זו אנו מראים כי כאשר לומדים את פונקצית המרחק בין העצמים השונים ניתן להשיג תוצאות צברור טובות יותר, בהשוואה לאלו שמתקבלות כשאר עושים שימוש בפונקציות מרחק קנוניות כגון המרחק האוקלידי.

ג. **למידת פונקציות גרעין** – בשנים האחרונות הוצע אוסף גדול של אלגוריתמים שהם מבוססי גרעין (kernel-based classifiers) כגון אלגוריתם ה – Support Vector Machine (SVM). אלגוריתמים

בתחומים אלו. קיימים סוגים שונים של אלגוריתמים אשר הוצעו בתחום של למידה מפוקחת למחצה. אלו כוללים בין השאר אלגוריתמי קלסיפיקציה, אלגוריתמי צברור וכן אלגוריתמים ללמידת פונקציות מרחק.

עבודה זו עוסקת בבעיה של למידת פונקציות מרחק, בעיה אשר צברה לאחרונה עניין הולך וגובר בקהילייה המדעית. פונקצית מרחק היא פונקציה אשר פועלת על זוגות של נקודות. הפונקציה מחזירה ערך רציף לכל זוג נקודות, אשר מודד את מידת הקרבה (או הדמיון) ביניהן. באופן אינטואיטיבי נצפה כי פונקצית מרחק "טובה" תחזיר ערך קטן יחסית עבור זוגות של נקודות שהן דומות זו לזו, ובהתאמה- תחזיר ערך גדול עבור זוגות של נקודות השונות זו מזו.

פונקציות מרחק הן אבן בנין משמעותית בסוגים שונים של אלגוריתמים ללמידה מפוקחת, לצברור (clustering) ולאחזור מידע. למרות שפע האלגוריתמים מבוססי פונקציות מרחק, עד לאחרונה נעשה בעיקר שימוש בפונקציות מרחק קנוניות כגון המרחק האוקלידי, או בפונקציות מרחק אשר תוכננו באופן ידני עבור יישום ספציפי, תוך שימוש בידע נוסף שקשור לאותו תחום היישום. בשנים האחרונות התעורר עניין הולך וגובר באלגוריתמים ללמידת פונקציות מרחק, ומספר רב של אלגוריתמים כאלו הוצעו בספרות. מרבית האלגוריתמים ללמידת פונקציות מרחק שייכים לתחום של למידה מפוקחת למחצה.

בעבודה זו יוצגו שלושה אלגוריתמים חדשים ללמידת פונקציות מרחק:

1. **Relevant Component Anlaysis (RCA)** – אלגוריתם הלומד פונקציות מרחק מסוג מרחקי מהלנוביס (Mahalanobis) תוך שימוש באילוצי שקילות חיוביים.

2. **DistBoos**t – אלגוריתם שיכול ללמוד פונקציות מרחק לא לינאריות, תוך שימוש באילוצי שקילות.

3. **KernelBoost** – אלגוריתם שהוא וריאציה על DistBoost אשר לומד פונקציות גרעין (kernel functions) שיכולות לשמש כל אלגוריתם קלסיפיקציה מבוסס גרעין (kernel-based classifier).

לאחר הצגת האלגוריתמים, יוצגו שימושים שונים של אלגוריתמים אלו, במספר רב של יישומים הכוללים צברור מידע, אחזור תמונות, אימונלוגיה חישובית, ניתוח רישומים עצביים וקלסיפיקציה מבוססת אלגוריתמי גרעין (kernel-based classification). בכל התחומים הללו הושג שיפור משמעותי בביצועים כאשר נעשה שימוש בפונקצית מרחק נלמדת, בהשוואה לשימוש בפונקציות מרחק סטנדרטיות. תוצאות אלו מדגימות את חשיבותו של תחום מחקר חדש זה.

תקציר העבודה

למידה חישובית היא תחום מחקר שמטרתו פיתוח שיטות המאפשרות למחשבים ללמוד. תחום זה עוסק בעיקרו בפיתוח שיטות המבוססות על ניסוי וטעייה. בפרדיגמה הקלאסית, האלגוריתם עובר שלב אימון שבו הוא עושה שימוש במדגם אימון (training set). לאחר מכן ביצועיו של האלגוריתם נמדדים על מדגם בחינה (test set) אשר מורכב מדוגמאות חדשות שלא הוצגו בשלב האימון. בשלושת העשורים האחרונים הושגה התקדמות מרשימה בתחום הלמידה החישובית הן מבחינה תיאורטית והן מבחינה יישומית. אלגוריתמי למידה הופעלו בהצלחה בתחומים רבים הכוללים ניתוח טקסט, אחזור מידע, ראיה חישובית, ביולוגיה חישובית, חישוביות עצבית ועוד.

המחקר בלמידה חישובית עוסק בעיקרו בתחום של "למידה מפוקחת" (Supervised learning). בתחום זה האלגוריתם מקבל בשלב האימון אוסף של דוגמאות מתויגות – כלומר דוגמאות אשר לכל אחת מהן מצורף גם התיוג (label) הנדרש. מדגם האימון הזה, משמש ללמידת פונקציה אשר ממפה כל דוגמא X לתיוג Y. האתגר העיקרי הינו בניית פונקציות בעלות יכולת הכללה (generalization) כלומר פונקציות בעלות יכולת לתייג נכון נקודות חדשות אשר לא הוצגו בפניהן בשלב האימון.

תחום מחקר נוסף בלמידה חישובית עוסק ב"למידה לא מפוקחת" (Unsupervised learning), שמטרתה לנתח אוסף של דוגמאות בלתי מתויגות ולחלץ מהן מידע כלשהו. תחום זה כולל בין השאר אלגוריתמים לצברור מידע (clustering), לבחירת ייצוג (feature selection), ויזואליזציה, שערוך פונקצית צפיפות, איתור מקרים יוצאי דופן וכדומה. השאלה הנשאלת בתחום זה הנה האם ניתן כלל ללמוד משהו (כלומר להסיק כלל כלשהו) מאוסף של דוגמאות לא מתויגות? התשובה לשאלה זו תלויה באופן קריטי בהנחות שנסכים להניח לגבי הנתונים. לדוגמא, אם נניח כי הנתונים נוצרו מאוסף ידוע של פונקציות התפלגות, קיימים אלגוריתמים לא מפוקחים היכולים לשערך את הפרמטרים של ההתפלגויות הללו.

בשנים האחרונות התעורר עניין הולך וגובר בתחום של "למידה מפוקחת למחצה" (Semi-supervised learning), אשר מצוי בין התחום של למידה מפוקחת לתחום של למידה לא מפוקחת. בתחום זה, בשלב האימון האלגוריתם נחשף למדגם לא מתויג של נקודות וכן לכמות מוגבלת של מידע צדדי (side-information) נוסף. מידע צדדי זה יכול להיות תיוג של חלק מהנקודות במדגם האימון, או להיות אוסף של אילוצי שקילות (equivalence constraints) - מידע לגבי זוגות של נקודות אשר ידועות כשייכות לאותה המחלקה, או שייכות למחלקות שונות. חלק מהמוטיבציות שהובילו לפיתוח תחום זה הנן (1) המחיר היקר הנדרש על מנת לתייג נקודות – במקרים רבים בכדי להשיג מדגם מתויג נדרשת השקעה מרובה של שעות אדם יקרות. (2) בתחומים מסוימים מאפייני הנתונים עליהם פועל האלגוריתם מצויים בשינוי איטי מתמיד. בשל כך, קיימת חשיבות מכרעת לפתח אלגוריתמים אשר מסוגלים לעקוב אחר שינויים אלו תוך שימוש במדגם לא מתויג במטרה לשפר ביצועים

חן ינובר, חברי ושותפי לחדר, היה מי שעניין אותי בתחום של האימונולוגיה החישובית – התחום שאליו פניתי להמשך לימודי. עבודתנו המשותפת, אשר החלה כמעט במקרה, היתה לי לעונג רב. העבודה עם חן היתה שיתוף הפעולה העצמאי הראשון שלי, ולמדתי רבות ממנו. השעות הרבות שבילינו יחדיו, וכוסות הקפה הרבות שגמענו, היו מהנות במיוחד.

נהניתי מאד לעבוד גם עם רובי המר ושאול הוכשטיין. העבודה המשותפת עימם ועם דפנה ויינשל בתחום הפסיכופיסיקה היתה ניסיון מעניין ומוצלח ליצור שיתוף פעולה אינטרדסיפלינרי שמטרתו לעסוק באותה השאלה המחקרית מכיווני מחקר שונים שכללו ניסויים קוגניטיביים, סימולציות מחשב ועבודה תיאורטית.

הפרוייקט המשותף שלי עם דפנה ויינשל, אינה ויינר ואלי נלקן החל לאחר שאלי שמע הרצאה שנתתי בסמינר עין-גדי של המרכז לחישוביות עצבית. העבודה המשותפת עם אלי היתה חויה מיוחדת במינה, ואני שמח שהייתה לי ההזדמנות לשתף עימו פעולה וללמוד ממנו.

תודה רבה גם לחברים הרבים במעבדה ובמסדרון - עידו עומר, דורון פלדמן, תמיר חזן, עמית גרובר, אנה סורקין, מיכאל פינק, שי שלו-שוורץ ולביא שפיגלמן שתמיד היה כיף לשוחח עימם לשתות עימם קפה ולהתייעץ בנושאים שונים ומשונים.

התעניינותי בתחום של חישוביות עצבית החלה תודות לחברי הטוב אסף קראוס, ואני חב לו תודה עמוקה על ההתלהבות שהוא הצית בי לעסוק בתחום זה.

אהובתי עינב, ליוותה אותי במסע הזה וסיפקה לי אהבה, יציבות והמון תמיכה. אינני יכול לדמיין את המסע הזה בלעדיה.

החלום לעסוק במחקר נולד בשעות הרבות שבהן ליוויתי את אבי בתצפיות הבקר ברפת בקיבוץ נען ובמרכז וולקני (בית-דגן) בעת ששקד על סיום לימודי הדוקטורט שלו בפיסיולוגיה, ואני חב לו תודה עמוקה על כך.

המסע הארוך הזה לא היה מתאפשר ללא האהבה התמיכה והאמונה הרבה של משפחתי, ובמיוחד של אימי, אשר עשתה כל שביכולתה כדי לאפשר לילדיה לממש את הפוטנציאל הגלום בהם. סבתי רבקה זגון, תמיד סייעה בעצות נפלאות ובהמון אכפתיות ואהבה. התיזה הזו מוקדשת לאימי ולסבתי.

תודות

דפנה ויינשל אשר הדריכה אותי בעבודתי זו, ליוותה מקרוב את תהליך ההתפתחות האקדמית שלי, תוך מעורבות רבה בכל הנעשה ותמיד מתוך עניין וסקרנות יוצאי דופן. הליווי הצמוד והאינטנסיבי שהיא העניקה לי בתחילת דרכי האקדמית סייע לי רבות, ובלעדיהם לא הייתי יכול לצלוח את המסע הזה. סקרנותה המדעית הרחבה, ויכולתה לעסוק בו זמנית בתחומי מחקר רבים מהווים נר לרגלי. אני מודה לה על סבלנותה, עקשנותה ועל המאמצים הרבים אשר עשתה למעני. תרומתה לעיצוב גישתי למחקר בלתי ניתנת להערכה.

הדוקטורט נעשה במסגרתו ובתמיכתו של המרכז לחישוביות עצבית. אני מודה למרכז על התמיכה הכלכלית שזכיתי לה. המרכז היה מקום פעיל ומהותי בליווי השלבים הראשונים של הדוקטורט. חלק מן העבודות המצורפות לדוקטורט זה הן פרי שיתופי פעולה פוריים ומוצלחים אשר כולם נעשו עם חברי מרכז אחרים מתחומי מחקר שונים.

את תחילת דרכי במסלול ליוותה גם עליזה שדמי, שהפכה את המרכז לחישוביות עצבית למקום שכיף לבקר בו. אני מודה לה מאד על היחס האישי הדאגה והאכפתיות שהיא הפגינה בכל עניין שבו פניתי אליה. עצותיה הרבות היו תמיד טובות במיוחד. רותי סוצ'י, שהחליפה אותה המשיכה את דרכה באופן מוצלח ביותר, ואני מודה גם לה על הדאגה והאכפתיות הרבה.

תחילתו של המחקר שהוביל לתזה הזו היה באביב שנת 2001 בעבודתי המשותפת עם דפנה ויינשל ומישה פאבל. הפגישות המשותפות הללו היו מעין קפיצתי הראשונה למעמקי עולם המחקר, והרעיונות הפוריים שעלו בהן הובילו לפיתוח האלגוריתם הראשון המוצג בתזה הזו. אני חב תודה עמוקה למישה פאבל, אשר סייע לי רבות בתחילת דרכי, ועל שתמך ועודד אותי ברגעים שבהם הייתי כמעט מיואש.

המחקר המוצג בתיזה זו הינו תוצר של שיתופי פעולה מרובים עם חברים רבים, שהתקופה שביליתי עימם היתה היפה בחיי:

חברי הטובים נועם שנטל ואהרון בר-הילל היו שותפים נהדרים למרבית הדרך שצעדתי בה. השוני הרב בין כל אחד מאיתנו הוביל לשיתוף פעולה פורה במיוחד. החברות בינינו הפכה את העבודה יחדיו להרבה מעבר להישגים מדעיים. זכורים לי לטובה הלילות הרבים שבילינו יחדיו לפני הדד-ליינים שעבדנו יחדיו לקראתם.

עבודה זו נעשתה בהדרכתה של פרופסור דפנה ויינשל.

למידת פונקציות מרחק: אלגוריתמים ויישומים

חיבור לשם קבלת תואר דוקטור לפילוסופיה

מאת
תומר הרץ

הוגש לסינאט האוניברסיטה העברית בשנת תשס"ז (2006)