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 agglomer-

ative 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),

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 $f : \mathcal{X} \times \mathcal{X} \to [0, 1]$.[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 un-labeled 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:

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

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

---

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 $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

---

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 un-labeled 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:

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

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
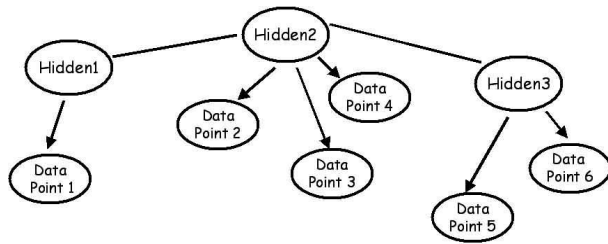

Figure 1. A Markov network representation of the constrained mixture setting. Each observable data node has a discrete hidden node as its ancestor. Positively constrained nodes have the same hidden node as their ancestor. Negative constraints are expressed using edges between the hidden nodes of negatively 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

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

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

$$F_{\frac{1}{2}} = \frac{2P * R}{R + P} \qquad (4)$$

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

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

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

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

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
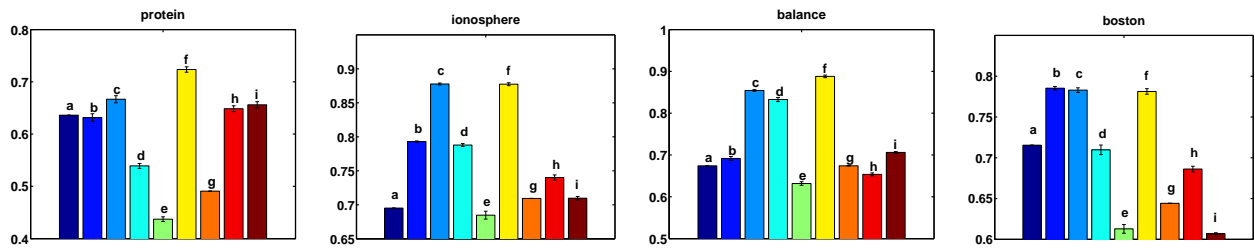
Figure 2. Clustering $F_{\frac{1}{2}}$ score over 4 data sets from the UCI repository using Ward's clustering algorithm. Methods shown are: (a) Euclidean, (b) RCA, (c) constrained EM, (d) SVM, (e) DTboost, (f) DistBoost, (g) Xing, (h) Constrained Complete Linkage, (i) Constrained K-means. The results 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.

[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.

However, *DistBoost* still outperforms its competitors, as it did in all previous examples.

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.

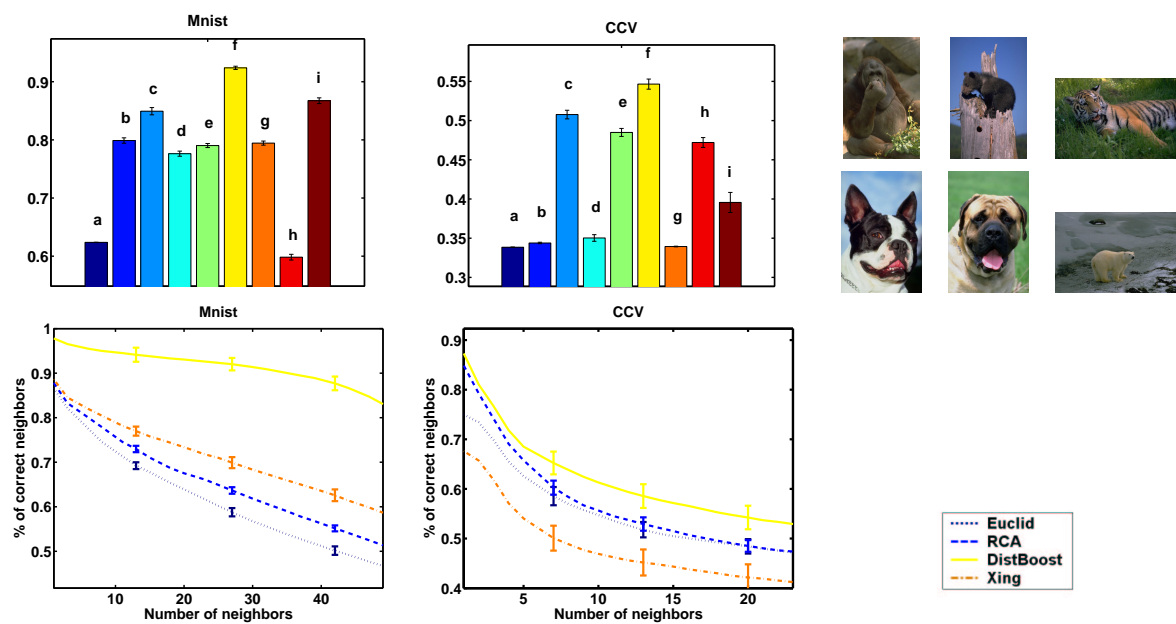Fukunaga, K. (1990). *Statistical pattern recognition*. San Diego: Academic Press. 2nd edition.

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.