

# Boosting the distance estimation Application to the $K$ -Nearest Neighbor Classifier

J. Amores<sup>a,\*,1</sup>, N. Sebe<sup>b</sup>, P. Radeva<sup>a</sup>

<sup>a</sup> Department of Computer Science, Universitat Autònoma de Barcelona, Computer Vision Center,  
Edifici O, Campus UAB Bellaterra, 08193 Cerdanyola del Valles, Barcelona, Spain

<sup>b</sup> University of Amsterdam, Amsterdam, The Netherlands

Received 1 February 2005; received in revised form 25 July 2005

Available online 10 October 2005

Communicated by Prof. F. Roli

## Abstract

In this work we introduce a new distance estimation technique by boosting and we apply it to the  $K$ -Nearest Neighbor Classifier ( $K$ -NN). Instead of applying AdaBoost to a typical classification problem, we use it for learning a distance function and the resulting distance is used into  $K$ -NN. The proposed method (Boosted Distance with Nearest Neighbor) outperforms the AdaBoost classifier when the training set is small. It also outperforms the  $K$ -NN classifier used with several different distances and the distances obtained with other estimation methods such as Relevant Component Analysis (RCA) [Duda, R.O., Hart, P.E., Stock, D.G., 2001. Pattern Classification, John Wiley and Sons Inc.]. Furthermore, our distance estimation performs dimension-reduction, being much more efficient in terms of classification accuracy than classical techniques such as PCA, LDA, and NDA. The method has been thoroughly tested on 13 standard databases from the UCI repository, a standard gender recognition database and the MNIST database.

© 2005 Elsevier B.V. All rights reserved.

**Keywords:** Distance estimation; AdaBoost; Dimension reduction

## 1. Introduction

Many applications in computer vision and pattern recognition use a distance function as a means of comparing objects. Well-known examples are the  $K$ -Nearest Neighbor Classifier ( $K$ -NN), unsupervised classifiers such as  $K$ -means, kernel-based classifiers such as the Support Vector Machine (SVM) and ranking-based retrieval in large databases. Obtaining an appropriate distance function can significantly improve the performance of those methods. In this work we propose a new framework for estimating the distance and we apply it to improve the performance of  $K$ -NN. The most common distance is the Euclidean dis-

tance that assumes the data has a Gaussian isotropic distribution. When the feature space has a large number of dimensions, an isotropic assumption is especially inappropriate and the performance of the classifier is decreased.

The typical procedure is to find an accurate distance by considering the family of (anisotropic) Mahalanobis distances  $(\vec{x} - \vec{y})^T W (\vec{x} - \vec{y})$  where the goal is to estimate the weight matrix  $W$ . If  $d$  is the number of dimensions, the matrix  $W$  contains  $d^2$  parameters to be estimated, which is not robust when the training set is small compared to the number of dimensions  $d$ . Furthermore, by using this distance we are still assuming the distribution of the data is Gaussian, which is not necessarily true. In order to reduce the number  $d$  of dimensions, we can apply classical techniques such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) or Non-parametric Discriminant Analysis (NDA) (Duda et al., 2001; Fukunaga and

\* Corresponding author. Fax: +34 935811670.

E-mail address: [jaume@cvc.uab.es](mailto:jaume@cvc.uab.es) (J. Amores).

<sup>1</sup> Partially supported by CICYT TIC2000-1635-C04-04, Spain.

Mantock, 1997). However, these methods again need to estimate a covariance or scatter matrix with  $d^2$  elements, which does not ameliorate the problem in the presence of a small training set.

Instead of estimating a parametric distance (or similarity), we propose to use a functional approximation where the similarity function is estimated by a generalization method (classifier). In particular, we propose to use AdaBoost with decision stumps (Schapire and Singer, 1999) in order to estimate the similarity function. Given a training set with feature vectors  $\Omega = \{\vec{x}_i\}_{i=1}^N$ , the similarity estimation is done by training AdaBoost with differences between vectors in  $\Omega$ , where each difference vector  $\vec{x}_i - \vec{x}_j$  has an associated label (desired value). Based on this training set, a function  $H$  is learnt by AdaBoost, and we derive a similarity  $S(\vec{x}, \vec{y}) \in [0, 1]$  that is accurate in a classification context. Finally, the estimated similarity  $S$  is incorporated into  $K$ -NN. Using  $K$ -NN along with a boosted similarity is especially suitable when the training set is small. Two factors contribute to this. First, if  $N$  is the size of the original training set, the new training set has  $O(N^2)$  relations between the vectors and this makes AdaBoost more robust against over-fitting. Second, AdaBoost complements  $K$ -NN by providing a similarity adapted to the characteristics of the given data. Increasing the effectiveness for small training sets is necessary in many real classification problems in general, and in particular in applications such as retrieval where only a small training set is expected to be provided online by the user.

In summary, the proposed method has three main advantages: (i) it adds effectiveness to the classifier when we have a small training set compared to the number of dimensions; (ii) the estimated similarity only uses a small number of dimensions, resulting in a powerful dimension-reduction technique that does not have the problems highlighted above for classical techniques (e.g. PCA, LDA, NDA); (iii) opposite to other works (Peng et al., 2001; Domeniconi et al., 2002), the proposed similarity estimation is general, i.e. it is not specifically designed for  $K$ -NN.

The rest of the paper is organized as follows: in Section 2 we review previous work on similarity estimation methods, Section 3 describes the proposed method, in Section 4 we provide results comparing our method to other methods, and in Section 5 we conclude and suggest future lines of work.

## 2. Related work

Recently, there have been several works on estimating the distance to ameliorate certain pattern recognition problems (Hertz et al., 2004; Domeniconi et al., 2002; Peng et al., 2001; Xing et al., 2003; Bar-Hillel et al., 2003). Domeniconi et al. (2002) and Peng et al. (2001) propose specific estimations designed for the  $K$ -NN classifier. They obtain an anisotropic distance based on local neighborhoods that are more narrow along relevant dimensions and more elongated along non-relevant ones. Xing et al.

(2003) propose to estimate the matrix  $W$  of a Mahalanobis distance by solving a convex optimization problem. They apply the resulting distance to improve the  $K$ -means behavior. Bar-Hillel et al. (2003) also use a weight matrix  $W$  in order to estimate the distance by Relevant Component Analysis (RCA). They improve the Gaussian Mixture EM algorithm by applying the estimated distance along with equivalence constraints (Bar-Hillel et al., 2003).

In this work we propose a distance estimation method that is general for any kind of classifier (i.e. not specific for  $K$ -NN) and it is not based on a parametric approach (such as the estimation of distances from the Mahalanobis family). For that purpose we propose to use AdaBoost as an effective learner that estimates an accurate distance given labelled data, and at the same time does not assume any particular distribution of the data. The work by Hertz et al. (2004) is the most resembling one to our method, although it is conceptually different. Although they use AdaBoost to estimate a distance function in a product space (with pairs of vectors), the weak classifier minimizes an error in the original feature space. Therefore, the weak classifier minimizes a different error than the one minimized by the strong classifier AdaBoost.

In contrast, our framework utilizes AdaBoost with weak classifiers that minimize the same error as AdaBoost, and in the same space. Here, the weak classifiers learn a (weak) distance function that minimizes the disagreement with desired values given as input. This represents a consistent optimization method where the whole AdaBoost searches for the optimum distance function in a unique functional space. Apart from this conceptual difference, Hertz et al. (2004) use Expectation–Maximization of Gaussian Mixture as weak classifier, where they assume the data has a Gaussian Mixture distribution and estimate several covariance matrices. If  $d$  is the number of dimensions, each matrix contains  $d^2$  parameters to be estimated, which is only robust when the training set is much larger than the dimensionality of the data. If  $d = 1000$ , a common empirical rule says that the training set should contain more than  $3d^2 = 30,000$  samples. In contrast, in our implementation we use decision stumps as base learner, which provides a dimension-reduction behavior and high accuracy in feature spaces with large dimensionality. Furthermore, no assumption on the distribution of the data is made, and the number of parameters estimated by our method does not depend on the number of dimensions  $d$ . Therefore, the proposed method is especially efficient for small training sets and large number of dimensions.

## 3. Distance estimation by boosting

In this section, a similarity estimation method is introduced that uses AdaBoost for learning the similarity. We seek for a similarity function that is particular of each class  $C$  (for example, consider the Mahalanobis distance that depends on the class  $C$  through the covariance  $\Sigma$  of this class). Let  $S_C$  be the similarity function we want to learn

for class  $C$ . First, let us describe the application of a class-dependent similarity  $S_C$  into any distance-based classifier such as the Nearest Neighbor. Let  $\Omega$  be the training set with labelled samples. Let the label of sample  $\vec{y}$  be denoted as  $l(\vec{y})$ . Given a new vector  $\vec{x}$  whose class we want to know, the classifier assigns to  $\vec{x}$  the label

$$l(\vec{x}) = l(\vec{y}^*), \vec{y}^* = \arg \max_{\vec{y} \in \Omega} S_{l(\vec{y})}(\vec{x}, \vec{y}), \quad (1)$$

where  $S_l(\vec{y})$  is the similarity function of the class with label  $l(\vec{y})$ . Let  $l(\vec{y}) = C$ . According to Eq. (1), the function  $S_C$  is applied over pairs  $(\vec{x}, \vec{y})$  where the second vector  $\vec{y}$  always belongs to class  $C$ . Note that, although we used Nearest Neighbor as an example, the application of  $S_C$  is the same in any distance-based classifier.

In this section we explain how to use an efficient learning algorithm such as AdaBoost (Freund and Schapire, 1996) to learn such a function  $S_C$ . AdaBoost has become popular because of its good generalization properties, producing excellent classification rates (Freund and Schapire, 1996). Given a base learner (weak classifier)  $h$ , AdaBoost provides an ensemble of these classifiers, where the classifier  $h_t$  at round  $t$  is focused on the samples that were not successfully learnt by the previous classifiers  $h_1, h_2, \dots, h_{t-1}$ . In this work we use as weak classifier the common decision stump that classifies based on a threshold on a single dimension (Freund and Schapire, 1996). Furthermore, we used the same version of AdaBoost as the one explained in (Viola and Jones, 2004; Freund and Schapire, 1996). AdaBoost with decision stumps is especially effective in high-dimensional feature spaces, selecting the most relevant dimensions that complement each other in the classification accuracy.

Instead of training AdaBoost directly with pairs of vectors, every pair  $(\vec{x}, \vec{y})$  is first mapped to a difference vector<sup>2</sup>  $(\vec{x} - \vec{y})$ . Let  $P = \{(\vec{x} - \vec{y}) : \vec{x}, \vec{y} \in \Omega, l(\vec{x}) = C, l(\vec{y}) = C\}$  be the set of difference vectors between samples of our data that belong to class  $C$  and recall that  $\Omega$  is the original training set. Let  $N = \{(\vec{x} - \vec{y}) : \vec{x}, \vec{y} \in \Omega, l(\vec{x}) \neq C, l(\vec{y}) = C\}$  be the set of difference vectors from samples in other classes to samples in class  $C$ . The input of AdaBoost is the new training set  $\Omega' = P \cup N$  that has associated labels: 1 for difference vectors in  $P$  and 0 for difference vectors in  $N$ . With this input, the same algorithm for AdaBoost as the one explained in (Viola and Jones, 2004) is followed. For completeness, we include the pseudocode of the proposed method in the Appendix A.

As a result, AdaBoost outputs a function  $H_C : \mathbb{R}^d \rightarrow [0, 1]$  expressed as

$$H_C(\vec{x} - \vec{y}) = \sum_{t=1}^T \alpha_t h_t(\vec{x} - \vec{y}), \quad (2)$$

where  $T$  is the number of assembled weak classifiers,  $h_t$  is the  $t$ th base function (weak classifier) chosen by AdaBoost, and  $\alpha_t$  is the weight assigned by AdaBoost to the function  $h_t$  based on its goodness. The pseudocode of AdaBoost is written in the Appendix A. Based on  $H_C$ , the similarity function is expressed as  $S_C(\vec{x}, \vec{y}) = H_C(\vec{x} - \vec{y})$ . Note that given the associated labels for  $P$  and  $N$ , AdaBoost learns a mapping  $H_C$  that provides a value close to 1 for differences  $\vec{x} - \vec{y}$  between samples of class  $C$ , and close to 0 for differences from samples of other classes to samples in  $C$ . The resulting  $S_C(\vec{x}, \vec{y}) = H_C(\vec{x} - \vec{y})$  represents an effective similarity in a classification context.

In our context, the weak classifier  $h_t(\vec{x} - \vec{y})$  can be seen as a “weak similarity” between  $\vec{x}$  and  $\vec{y}$ . If we use decision stumps,  $h_t$  is expressed as

$$h_t(\vec{x} - \vec{y}) = \begin{cases} 1 & \text{if } p_t(x_{i_t} - y_{i_t}) < p_t \theta_t, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where  $i_t$  is the chosen dimension for  $h_t$  (i.e.  $x_{i_t}$  denotes the  $i_t$  component of vector  $\vec{x}$ ),  $p_t \in \{-1, 1\}$  is the polarity, and  $\theta_t$  is the threshold of  $h_t$ . Based on Eqs. (2) and (3) we can express  $S_C$  as a function of the chosen dimensions:

$$S_C(\vec{x}, \vec{y}) = \sum_{t=1}^T \alpha_t h_t(x_{i_t} - y_{i_t}), \quad (4)$$

which is nothing more than a weighted sum of non-linear functions  $h_t$  over differences of individual elements, analogous to the weighted Euclidean distance. If we use a number of rounds  $T \ll d$  lower than the number of dimensions  $d$ , we obtain a powerful reduction of dimensionality, where the first dimensions are the most relevant chosen by boosting, and the rest of dimensions complement the ones chosen until that moment.

Finally, note that the resulting similarity  $S_C$  may not be a true metric, but this is not necessarily a disadvantage. Indeed, non-metric distances can be more accurate for comparing complex objects, as studied recently in (Jacobs et al., 2000).

## 4. Results

We tested the performance of the proposed distance estimation into the Nearest Neighbor classifier, and compared it with the classical AdaBoost. We also compared our method with Nearest Neighbor using several other distances and dimension-reduction methods such as PCA, LDA (Duda et al., 2001) and NDA (Fukunaga and Mantock, 1997). In order to test the performance of the different methods, we used 13 standard data-sets from the UCI repository (Merz and Murphy, 1996) and two other known databases for classification of images: the MNIST (LeCun, 1998) addressing hand-written digit recognition (see Fig. 1(a)), and a database with faces addressing the

<sup>2</sup> In order to learn the similarity between vectors  $(\vec{x}, \vec{y})$  the base learner must be based on comparisons between  $\vec{x}$  and  $\vec{y}$ . In case we use decision stumps, each base learner is based on single elements of each vector, therefore it does not take into account the relation between  $\vec{x}$  and  $\vec{y}$ . We use the difference vectors  $\vec{x} - \vec{y}$  as objects of the training set in order to force the base learner be based on comparisons (i.e. differences) between each element of  $\vec{x}$  and the same element of  $\vec{y}$ .

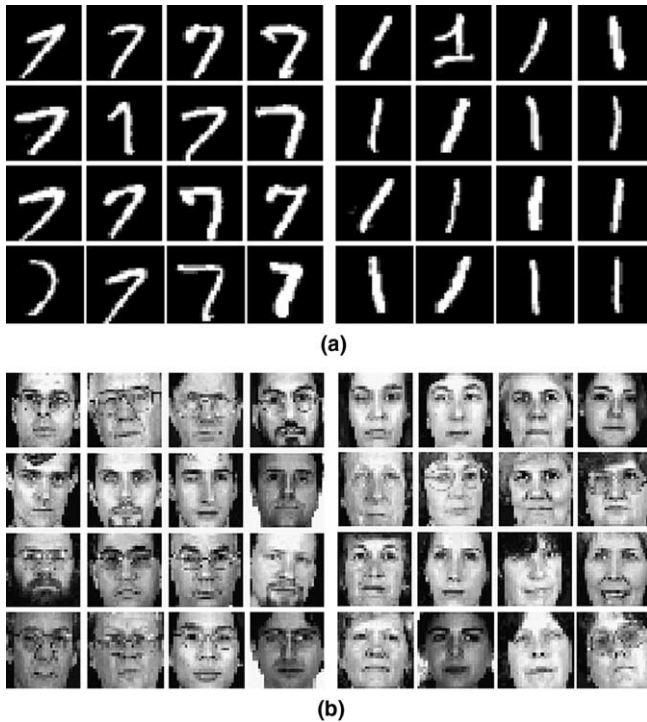


Fig. 1. (a) Digits ‘1’ and ‘7’ from MNIST database; (b) gender recognition database.

problem of gender-recognition (see Fig. 1(b)). The gender recognition database contains images of faces from the AR database (Martinez et al., 1998) and the XM2TVS database (Matas et al., 1999). The images were pre-processed by applying registration and standard normalization (see Fig. 1(b)).

In order to have a fair comparison against the original AdaBoost, the chosen data-sets correspond to binary classification problems. The MNIST database is transformed into a two-class problem by restricting the classification to digits ‘1’ and ‘7’, chosen because of their similar appearance (see Fig. 1(a)). The rest of data-sets contained originally two classes. Table 1 lists the data-sets, the number

Table 1  
Data-sets used in the experiments

Data-set	Observations	Dimensions	Nominal values
ad	2359	1558	✓
gender-recognition	2462	784	
MNIST	12,000	784	
arrhythmia	420	278	✓
splice	1500	60	✓
sonar	208	60	
spectf	349	44	✓
ionosphere	351	34	
wdbc	569	30	
German	1000	24	✓
vot1	435	16	✓
credit	690	15	✓
wbc	699	9	
pima	768	8	
liver	345	6	✓

of samples and the number of dimensions of each one. The column “nominal values” indicates whether the database contains symbolic values, which we transformed into integer values. The categories are sorted in descending order of the number of dimensions.

#### 4.1. Experimental setup

We were mostly interested in problems with a large number of dimensions, because an appropriate distance function is very important in those cases. The first four categories in Table 1 correspond to databases where the samples have a large number of dimensions. In those categories we chose a small training set that contains always 200 samples, very few as compared to the number of dimensions. The samples of the training set are randomly chosen from the data-set and the rest of the data-set is used as test set. This is repeated 100 times, we report results that are the average over these 100 repetitions. For the MNIST database, separate training and test sets are included in the standard database (LeCun, 1998). In this case we also performed 100 repetitions where each time we picked randomly 200 samples from the training set provided in (LeCun, 1998). Each time we evaluate the classification on the same test set provided in (LeCun, 1998).

The other 11 data-sets are included in order to test whether the proposed method also performs well with data-sets with a small number of dimensions. In those cases, we chose a training set that contains 20% of the data and the test set contains the other 80% of the data-set. As before, we repeated the experiments 100 times, each time we picked randomly the training and test sets, and we report as result the average over these 100 repetitions.

#### 4.2. Results on high-dimensional feature spaces

In all the data-sets, we tested  $K$ -NN with  $K = 1$  (i.e. Nearest Neighbor). First we tested the performance of the proposed method on high-dimensional feature spaces, which correspond to the first four data-sets in Table 1. In Fig. 2, we compared the performance of our method (Nearest Neighbor with Boosted Distance) against the performance of the standard Nearest Neighbor with Euclidean distance. Both AdaBoost and our distance boosting use decision stumps over single dimensions. Therefore, the number of dimensions used by our classifier and AdaBoost is *at most* the number of weak classifiers added. These boosting methods can use less dimensions than weak classifiers due to the fact that several classifiers can share the same dimension. Fig. 2 shows the accuracy as a function of the number of assembled weak classifiers. In all the cases, NN with Boosted Distance outperforms both AdaBoost and NN with Euclidean distance.

As mentioned before, the proposed distance estimation behaves like a dimension-reduction technique. Fig. 3 shows a comparison between our method and reduction methods

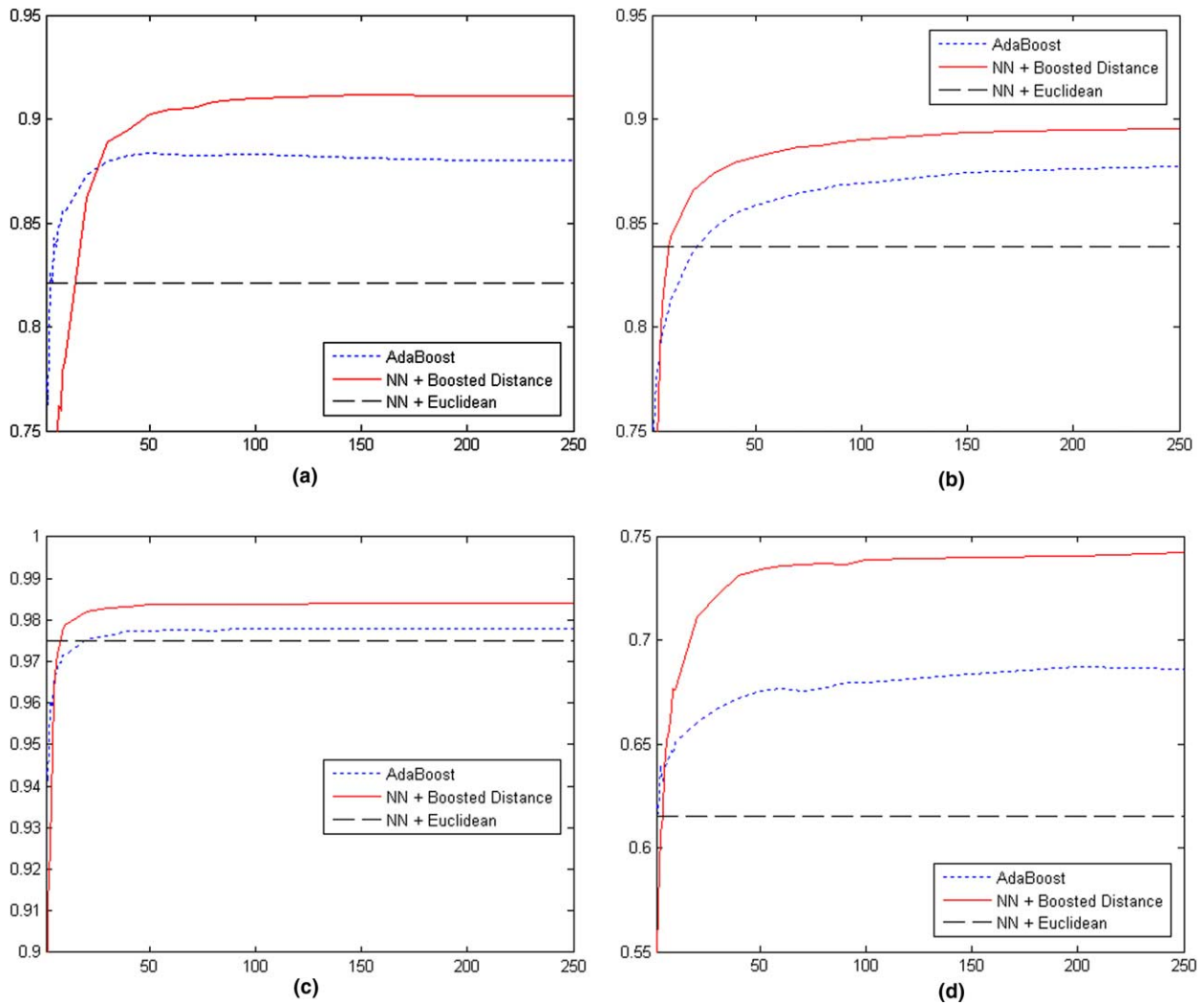


Fig. 2. Performance in high-dimensional data-sets. The horizontal axis corresponds to the number of rounds in AdaBoost and Boosted Distance, and the vertical to the accuracy of the classifiers. (a)–(d) correspond respectively to data-sets “ad”, “gender”, “MNIST” and “arrhythmia” (see Table 1).

such as PCA, LDA (Duda et al., 2001) and NDA. In Fig. 3 the accuracy of the different methods is displayed as a function of the number of dimensions. Fig. 3(b) and (c) do not show the LDA accuracy, which is very low in those cases: 50.08% for the gender database and 49.96% for the MNIST database. The number of tested dimensions shown in the figures is different for each data-set because in NDA we do not take dimensions corresponding to eigenvalues lower than  $1.0e-8$ , due to precision problems with the inverse of the matrix caused by the small size of the training set. However, PCA and LDA never achieved accuracy values higher than the ones displayed in the graphs. LDA does not vary because for 2 classes the resulting dimension is 1.

Classical methods perform very poorly due to the fact that we use a very small database compared to the dimensionality of the data. Note that all these methods rely on estimating a covariance or scatter matrix with  $d^2$  elements, where  $d$  is the number of dimensions. A typical rule says that we need a training set of size greater than  $3d^2$  in order

to obtain a robust estimation of  $d^2$  parameters. To verify this, we tested the same algorithms using 80% of the data as training set. In this case the dimension-reduction methods performed favorably compared with NN without dimension-reduction.

With small training sets, the proposed distance-estimation clearly outperforms other dimension reduction techniques in all the cases (see Fig. 3(a)–(d)).

### 4.3. Results on all the databases

Finally, we compared our Boosted Distance against the original AdaBoost using 15 well-known data-sets from the UCI repository. We also compared our Boosted Distance against several other distances: L2, L1, and distances that use a weight-matrix: RCA (Bar-Hillel et al., 2003), Mahalanobis with the same covariance matrix for all the classes (Mah) and Mahalanobis with a different covariance matrix for every class (Mah-C). The last three distances are based

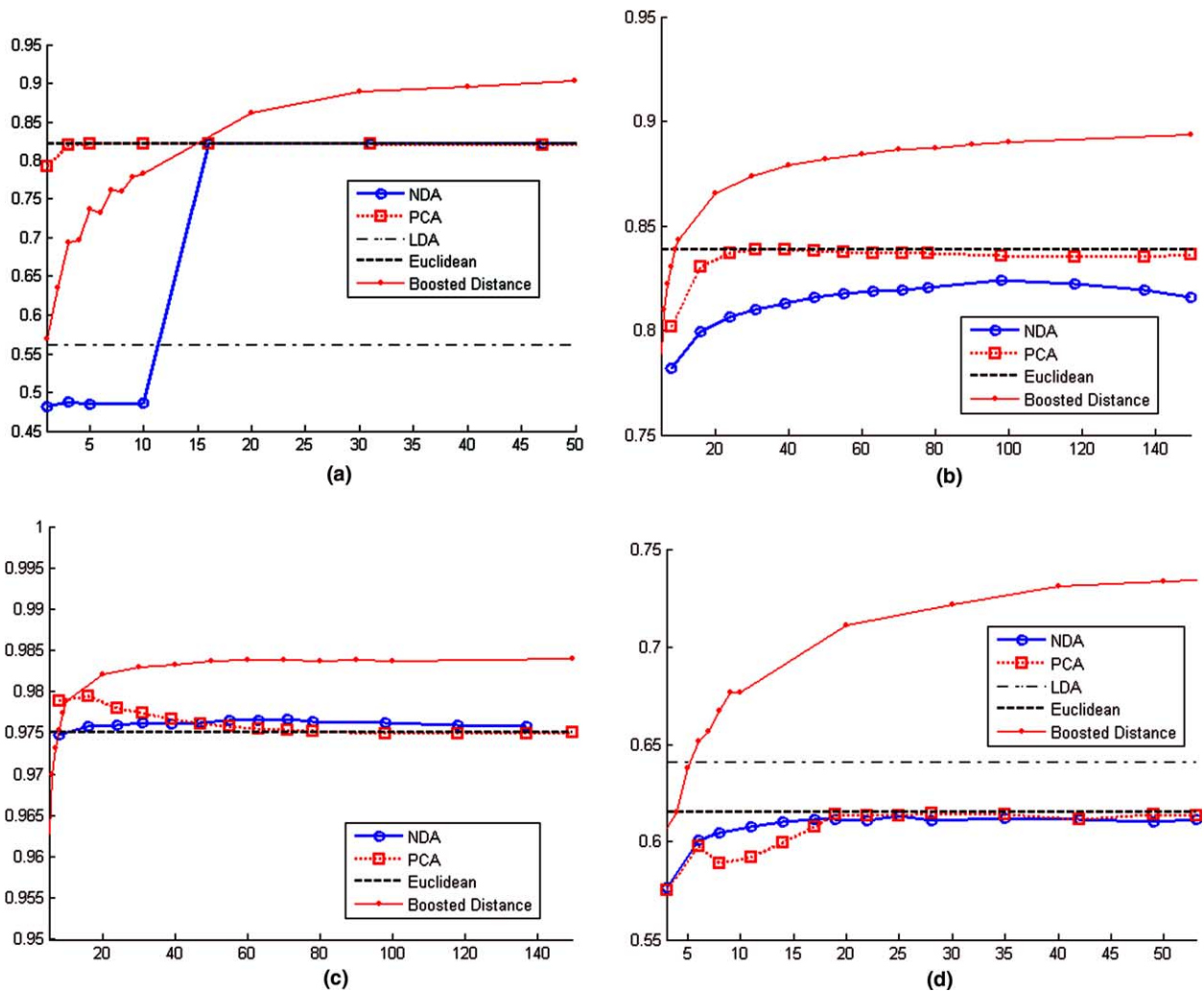


Fig. 3. Comparison against dimension-reduction techniques. The horizontal axis corresponds to the number of dimensions and the vertical to the accuracy of the classifiers. In the case of Boosted Distance, the indicated number of dimensions represents an upper bound on the real number of dimensions. (a)–(d) correspond respectively to data-sets “ad”, “gender”, “MNIST” and “arrhythmia”.

on computing a covariance or scatter matrix. When the size of the training set is small, the resulting distance is not robust. In order to ameliorate this, we included three more distances where we only estimate the diagonal elements of the covariance matrix. The resulting distances receive the same name as the originals, adding a ‘D’, e.g. Mah-D means Mahalanobis with diagonal matrix.

Table 2 shows the error percentages for every data-set and every method. The second column (NN) shows the error when NN is used with the best performing distance (chosen among L1, L2, RCA, Mah, Mah-C, RCA-D, Mah-D, Mah-CD) for every data-set. In parentheses we indicate what the best distance is for the corresponding data-set. The third column indicates the error (in percentage) of AdaBoost with decision-stumps. The fourth column shows the error (in percentage) of NN with our Boosted Distance. Let  $e_A$ ,  $e_D$  be the errors percentages respectively obtained with AdaBoost and our Boosted Distance method. The fifth column shows the difference between

AdaBoost and our method:  $(e_A - e_D)$ . The sixth column shows the relative improvement achieved by our method compared to AdaBoost:  $100 \frac{e_A - e_D}{e_A}$ . We use bold letters for highlighting the method that achieved the best performance for each data-set. Fig. 5 shows 95% confidence intervals, where all the methods of Table 2 are included, and we also include AdaBoost with C4.5 as base learner.

In 13 out of 15 data-sets, our method outperforms AdaBoost. This shows that the proposed method is consistently better than AdaBoost when the training set is small. The mean relative improvement of our method over AdaBoost is 12.87%. Compared against NN with other distances, the proposed distance estimation is significantly better in all the data-sets, except for the ionosphere data-set. This also proves that this method achieves a highly discriminative way of estimating the distance as compared to traditional methods.

In order to see if the new method consistently outperforms AdaBoost even when a stronger classifier is used,

Table 2

Error percentages of different methods. From column 2 to 6: NN using the best distance, AdaBoost, our Boosted Distance used into NN, the difference between our method and AdaBoost, and the relative improvement of our method over AdaBoost

Data-set	NN	AdaB.	Boost. Dis.	Diff	Rel. Improv.
ad	(L1) 17.31	12.00	<b>8.88</b>	3.12	26.00
gender	(L1) 15.38	12.27	<b>10.45</b>	1.82	14.83
MNIST	(RCA-D) 2.34	2.22	<b>1.60</b>	0.62	27.93
arrhythmia	(Mah-D) 37.02	31.39	<b>25.78</b>	5.61	17.87
splice	(Mah-CD) 10.55	5.94	<b>4.73</b>	1.21	20.37
sonar	(Mah-D) 26.10	25.95	<b>25.67</b>	0.28	1.08
spectf	(RCA) 31.16	28.65	<b>27.10</b>	1.55	5.41
ionosphere	(Mah-CD) <b>10.78</b>	19.92	16.27	3.65	18.32
wdbc	(Mah-D) 6.83	5.81	<b>4.67</b>	1.14	19.62
german	(Mah-CD) 38.74	34.31	<b>32.40</b>	1.91	5.57
vote1	(L1) 9.07	<b>6.37</b>	6.86	-0.49	-7.69
credit	(Mah-CD) 19.18	<b>17.97</b>	18.33	-0.36	-2.00
wbc	(RCA) 5.25	5.70	<b>3.79</b>	1.91	33.51
pima	(Mah-CD) 34.55	31.02	<b>28.91</b>	2.11	6.80
liver	(Mah) 41.11	35.51	<b>33.58</b>	1.93	5.44

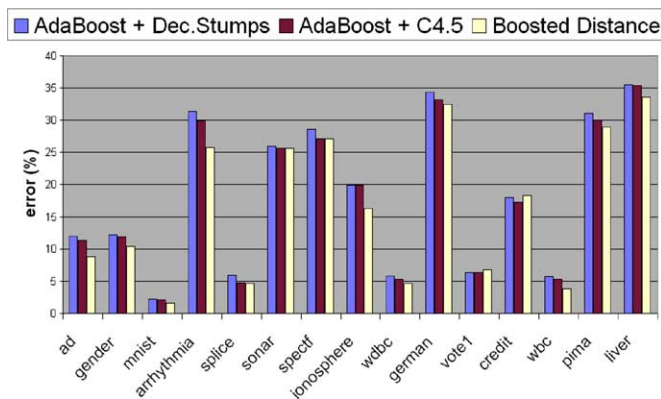


Fig. 4. From left to right: AdaBoost with decision stumps, AdaBoost with C4.5, and Boosted Distance (using decision stumps) with NN.

we also tested the performance of AdaBoost using the C4.5 decision tree (Quinlan, 1996) as base learner.<sup>3</sup> In Fig. 4 AdaBoost with C4.5 is compared against Boosted Distance (using decision stumps) and against AdaBoost with decision stumps. Fig. 5 shows 95% confidence intervals. The performance of AdaBoost is improved when C4.5 is used as base learner. However, boosting the distance applied to Nearest Neighbor still outperforms AdaBoost with C4.5 in the same data-sets as before. The difference is smaller in most cases, but still significant in many of them (e.g. ad, gender, arrhythmia, ionosphere). This is in spite of the fact that we are still using decision stumps in the Boosted Distance, so that the proposed method performs better even when the base learner is worse.

## 5. Discussion

In this work, we provided an efficient method for estimating the distance by boosting. We showed that the

<sup>3</sup> We used the Matlab® classification toolbox included in (Stork and Yom-Tov, 2004).

Nearest Neighbor classifier used along with the proposed (Boosted) Distance outperforms the popular AdaBoost classifier. In particular, we used AdaBoost with decision stumps for learning the distance, and showed that NN with the Boosted Distance is an efficient classifier in the presence of high-dimensional spaces and small training sets. This is very important in many applications where we do not have a large training set.

We performed an exhaustive evaluation of the method using 15 standard data-sets. The proposed method consistently outperformed AdaBoost. We also compared the proposed distance estimation method with several other distances (including distances estimated from the data) and showed that the proposed method is significantly better than the best distance. Finally, the proposed method also behaves as a dimension-reduction technique. Compared with PCA, LDA or NDA, our method proved to be significantly more efficient.

We think it would be interesting to implement the Boosted Distance using C4.5 in order to see if the proposed method is farther improved. Note that the general proposed method for boosting the distance does not depend on the weak classifier. This can be seen from the pseudocode in the Appendix A, where the Boosted Distance algorithm calls a generic AdaBoost function. We let for future research an evaluation of Boosted Distance using different versions of boosting. We would also like to test the method in content-based image retrieval, where the size of the training set is small because it is provided on-line by the user.

## Appendix A

### Pseudocode of boosting the distance estimation

The specific pseudocode of the algorithms used for Boosted Distance with nearest neighbor is provided in this appendix. Let  $\Omega = \{\vec{x}_i\}_{i=1}^n$  be a training set with  $n$  vectors, let  $l_i \in \{1, \dots, M\}$  be the label associated with  $\vec{x}_i$ , where  $M$  is the number of classes. Let  $T$  be the number of rounds

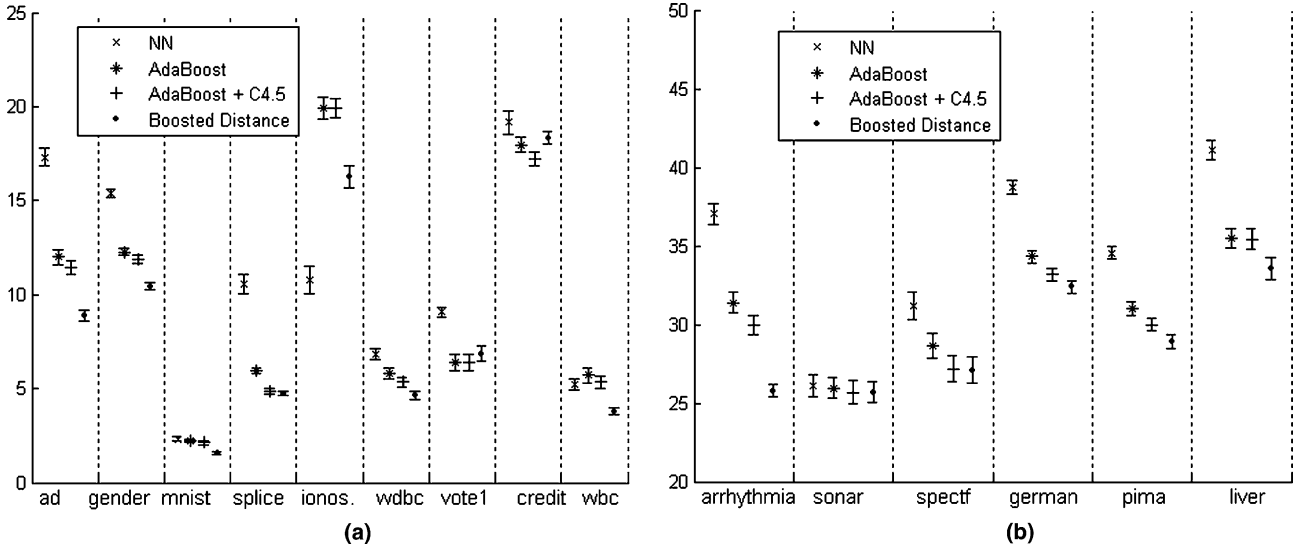


Fig. 5. 95% confidence intervals of error, shown in percentages. From left to right, the bars correspond to NN with best distance (as indicated in Table 2), AdaBoost + decision stumps, AdaBoost + C4.5, and Boosted Distance with Nearest Neighbor. (a) Data-sets with error lower than 25%, (b) those with error higher.

in boosting. Let  $\vec{y}$  be a new vector whose label we want to obtain by our classifier. We distinguish two steps: training the classifier (Algorithm A.1):

$$\{\Phi_C\}_{C=1}^M \leftarrow \text{BoostedDistance}(\Omega, \{l_i\}_{i=1}^n, T),$$

where  $\{\Phi_C\}_{C=1}^M$  are  $M$  sets of parameters obtained by the training stage. We then classify the new vector  $\vec{y}$  (Algorithm A.3):

$$l_y \leftarrow \text{NearestNeighborBoostedDistance}(\vec{y}, \{\Phi_C\}_{C=1}^M, \Omega).$$

**Algorithm A.1.** BoostedDistance( $\Omega, \{l_i\}_{i=1}^n, T$ )

**Input:**

Training set  $\Omega = \{\vec{x}_i\}_{i=1}^n$

Associated labels  $l_i \in \{1, \dots, M\}$ , where  $M$  is the number of classes.

Number of rounds  $T$

**Output:**  $\{\Phi_C\}_{C=1}^M$ : Parameters of Boosted Distance for each class  $C = 1, \dots, M$ .

For  $C = 1, \dots, M$

$P \leftarrow \{(\vec{x}_i - \vec{x}_j) : l_i = C, l_j = C, i, j = 1, \dots, n\}$

$N \leftarrow \{(\vec{x}_i - \vec{x}_j) : l_i \neq C, l_j = C, i, j = 1, \dots, n\}$

$\Phi_C \leftarrow \text{AdaBoost}(P, N, T)$

End loop.

Return  $\{\Phi_C\}_{C=1}^M$ .

AdaBoost obtains an ensemble of weak classifiers given the positive and negative training data  $P$  and  $N$ . Boosted Distance estimation does not depend on the weak classifier, as it is based on a generic ensemble (AdaBoost). In this work we use as weak classifier the decision stump, defined as

$$h(\vec{x}, f, p, \theta) = \begin{cases} 1 & \text{if } px_f < p\theta, \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.1})$$

**Algorithm A.2.** AdaBoost( $P, N, T$ )

**Input:**

Positive set  $P \subset \mathbb{R}^d$ , where  $d$  is the number of dimensions of the data.

Negative set  $N \subset \mathbb{R}^d$

Number of rounds  $T$ .

**Output:** Parameters  $\Phi = \{\alpha_t, f_t, p_t, \theta_t\}_{t=1}^T$ .

Let  $n = |P| + |N|$

$X \leftarrow P \cup N$ , let us express this set as  $X = \{\vec{x}_i\}_{i=1}^n$

Let  $l_i = 1$  if  $\vec{x}_i \in P$  and  $l_i = 0$  if  $\vec{x}_i \in N$ .

Initialize weights:  $w_{1,i} = \frac{1}{2|P|}, \frac{1}{2|N|}$  if  $l_i = 1, 0$  respectively.

For  $t = 1, \dots, T$

Normalize the weights,  $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$  for all  $i = 1, \dots, n$

Select the best weak classifier with respect to the weighted error

$$\epsilon_t = \min_{f,p,\theta} \sum_{i=1}^n w_{t,i} |h(f, p, \theta, \vec{x}_i) - l_i|.$$

Define  $h_t(\vec{x}) = h(\vec{x}, f_t, p_t, \theta_t)$ , where  $f_t, p_t$  and  $\theta_t$  are the minimizers of  $\epsilon_t$

$$\beta_t \leftarrow \frac{\epsilon_t}{1 - \epsilon_t}$$

$$e_i \leftarrow |h_t(\vec{x}_i) - l_i| \text{ for all } i = 1, \dots, n.$$

Update weights:

$$w_{t+1,i} \leftarrow w_{t,i} \beta_t^{1-e_i}, \text{ for all } i = 1, \dots, n$$

Set the weight of the weak weight classifier  $h_t$ :

$$\alpha_t \leftarrow \frac{1}{\log(\beta_t)}.$$

End loop.

The final ensemble  $H(x)$  is the linear combination:

$$H(\vec{x}) = \sum_{t=1}^T \alpha_t h_t(\vec{x}) = \sum_{t=1}^T \alpha_t h(\vec{x}, f_t, p_t, \theta_t)$$

Return parameters  $\Phi = \{\alpha_t, f_t, p_t, \theta_t\}_{t=1}^T$ .

The step where the weak classifier is selected requires that the weighted error is computed for all possible parameters  $f, p, \theta$  and the ones that minimize the weighted error are chosen. This computation can be done efficiently with cost  $O(nd)$ , where  $n$  is the number of positive and negative vectors and  $d$  the dimensionality of the vectors, we refer to Viola and Jones (2004) for a discussion on the implementation. Once we have estimated the Boosted distance parameters  $\{\Phi_C\}_{C=1}^M$ , classification of a new vector  $\vec{y}$  is performed as follows:

**Algorithm A.3.** Nearest Neighbor Boosted Distance ( $\vec{y}, \{\Phi_C\}_{C=1}^M, \Omega$ )

**Input:**

Data point to classify:  $\vec{y}$ .

Parameters of Boosted distance  $\{\Phi_C\}_{C=1}^M$ ,

where  $\Phi_C = \{\alpha_{C,t}, f_{C,t}, p_{C,t}, \theta_{C,t}\}_{t=1}^T$

Training set:  $\Omega = \{\vec{x}_i\}_{i=1}^n$

**Output:** Label assigned to  $\vec{y}$ :  $l_y \in \{1, \dots, M\}$

$s_{\max} \leftarrow -\infty$

for  $i = 1, \dots, n$

$C \leftarrow l_i$

$s_i \leftarrow \sum_{t=1}^T \alpha_{C,t} h(\vec{y} - \vec{x}_i, f_{C,t}, p_{C,t}, \theta_{C,t})$  (Eq. (A.1))

if  $s_i > s_{\max}$

$s_{\max} \leftarrow s_i$

$l_y \leftarrow C$

end if

end for

return  $l_y$ .

## References

- Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D., 2003. Learning distance functions using equivalence relations. *Int'l Proc. ICML*, 11–18.
- Domeniconi, C., Peng, J., Gunopulos, D., 2002. Locally adaptive metric nearest-neighbor classification. *IEEE TPAMI* 24 (9), 1281–1285.
- Duda, R.O., Hart, P.E., Stock, D.G., 2001. *Pattern Classification*. John Wiley and Sons Inc.
- Freund, Y., Schapire, R.E., 1996. Experiments with a new boosting algorithm. *Proc. Int'l Conf. Mach. Learn.*, 148–156.
- Fukunaga, K., Mantock, J., 1997. Nonparametric discriminant analysis. *IEEE TPAMI* 19 (2), 671–678.
- Hertz, T., Bar-Hillel, A., Weinshall, D., 2004. Learning distance functions for image retrieval. *IEEE Proc. CVPR*, 570–577.
- Jacobs, D.W., Weinshall, D., Gdalyahu, Y., 2000. Classification with nonmetric distances: Image retrieval and class representation. *IEEE TPAMI* 22 (6), 583–600.
- LeCun, Y., 1998. MNIST database. Available from: <<http://yann.lecun.com/exdb/mnist/>>.
- Martinez, A., Benavente, R., 1998. The AR face database, Tech. Rep. 24, Computer Vision Center.
- Matas, J., Hamouz, M., Jonsson, K., Kittler, J., Li, Y., Kotropoulos, C., Tefas, A., Pitas, I., Tan, T., Yan, H., Smeraldi, F., Bigun, J., Capdevielle, N., Gerstner, W., Ben-Yacoub, S., Abdeljaoued, T., Mayoraz, E., 1999. Comparison of face verification results on the XM2VTS database. *ICPR*, 858–863.
- Merz, C., Murphy, P., 1996. UCI repository of machine learning databases. Available from: <<http://www.ics.uci.edu/mllearn/MLRepository.html>>.
- Peng, J., Heisterkamp, D.R., Dai, H.K., 2001. LDA/SVM driven nearest neighbor classification. *IEEE Proc. CVPR*, 940–942.
- Quinlan, J.R., 1996. Bagging, boosting, and c4.5. *Proc. Ntl. Conf. Artif. Intell.*, 725–730.
- Schapire, R.E., Singer, Y., 1999. Improved boosting using confidence-rated predictions. *Mach. Learn.* 37 (3), 297–336.
- Stork, D.G., Yom-Tov, E., 2004. *Computer Manual in MATLAB to Accompany Pattern Classification*. John Wiley and Sons Inc.
- Viola, P., Jones, M.J., 2004. Robust-real time face detection. *Int'l J. Comput. Vis.* 57 (2), 137–154.
- Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S., 2003. Distance metric learning, with application to clustering with side-information. *Proc. NIPS*, 505–512.