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:

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

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:

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

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.

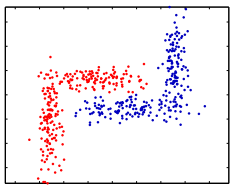

Figure 1. A 2-d synthetic example of a non-convex and non-Gaussian dataset.

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.

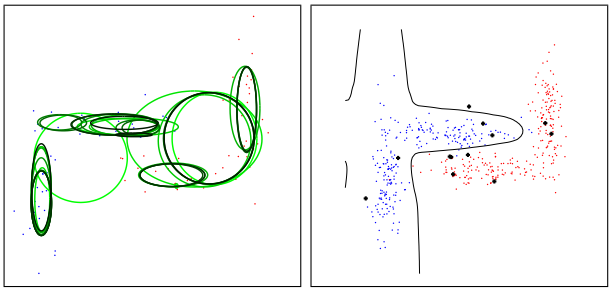

Figure 2. **Left:** The Gaussians learnt by *KernelBoost*-dissolve (presented in Sec. 2.5). The Ellipses mark 1-std contours. Darker ellipses show Gaussians obtained at later boosting rounds. **Right:** The separator induced by the Gaussians for this example. Support vectors are marked by black dots.

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)	$\mathbf{79.30(\pm 0.23)}$	$72.33(\pm 0.27)$	$64.90(\pm 0.22)$	$50.35(\pm 0.17)$	$67.83(\pm 0.23)$

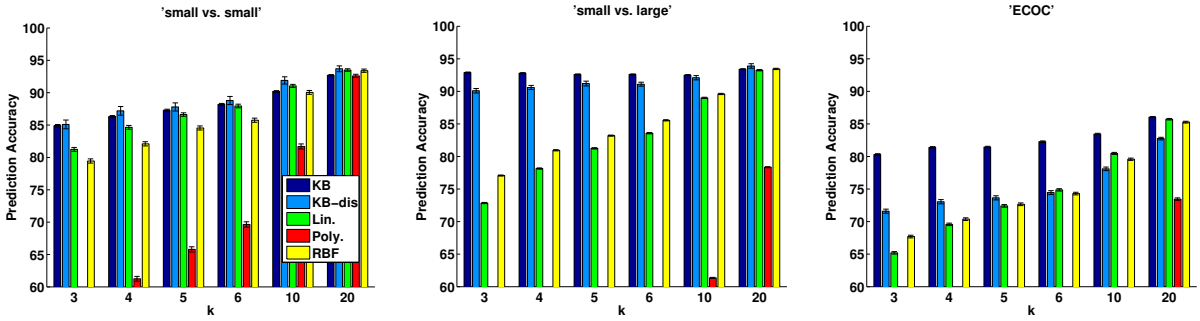

Figure 3. Median classification accuracy on the MNIST dataset as a function of the number of labeled points. Methods compared are: KB - *KernelBoost*, KB-dis - *KernelBoost* with the label dissolve mechanism, Lin. - linear kernel, Poly. - polynomial kernel of degree 5, and RBF - RBF kernel with σ chosen 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

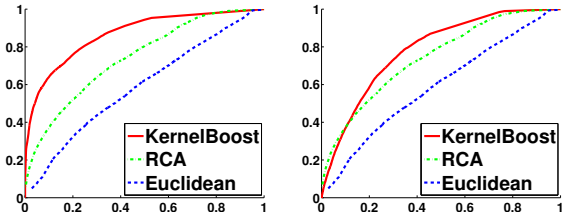

Figure 4. ROC retrieval curves on the YaleB dataset. The graphs compare the performance of KernelBoost to RCA and to the Euclidean distance metric in a "Learning to learn" scenario. Left: Results for classes for which unlabeled data was presented during the training stage. Right: Results on novel classes 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.