Learning Kernel Expansions for Image Classification

Fernando De la Torre **Oriol Vinyals**
Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.
ftorre@cs.cmu.edu vinyals@cs.cmu.edu

Abstract

Kernel machines (e.g. SVM, KLDA) have shown state-of-the-art performance in several visual classification tasks. The classification performance of kernel machines greatly depends on the choice of kernels and its parameters. In this paper, we propose a method to search over a space of parameterized kernels using a gradient-descent based method. Our method effectively learns a non-linear representation of the data useful for classification and simultaneously performs dimensionality reduction. In addition, we suggest a new matrix formulation that simplifies and unifies previous approaches. The effectiveness and robustness of the proposed algorithm is demonstrated in both synthetic and real examples of pedestrian and mouth detection in images.

1. Introduction

Kernel methods [21, 22] are increasingly used for data clustering, modeling and classification problems because of their state-of-the-art performance, simplicity, and lack of local minima problems. Kernel machines such as SVM, KPCA, or KLDA project data into (usually) high dimensional feature spaces, where linear decision surfaces correspond to non-linear decision surfaces in the original input space. The performance of any kernel machine greatly depends on the type of kernel and its parameters. The kernel explicitly defines a similarity measure between two samples and implicitly represents the mapping of the input space to the feature space. In general, different problems require different feature spaces, and a domain-specific kernel is a useful feature for an algorithm to have. In this paper, we propose a method to learn a non-linear mapping of the data useful to improve classification in kernel machines. Besides improving performance, learning a good similarity measure which reflects the structure of the data can be useful for other tasks such as visualization or clustering.

Fig. 1 shows the main aim of this paper. We have synthetically generated two multimodal three-dimensional Gaussian classes. Two of the dimensions are relevant for classification and the other dimension is high-variance

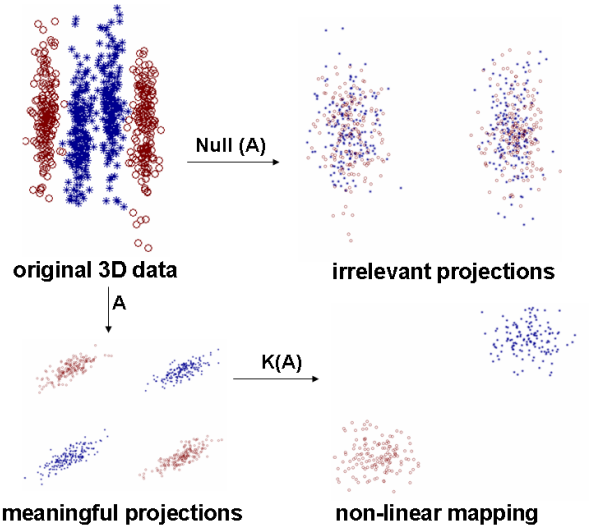


Figure 1. Learning a non-linear mapping useful for classification. Top left: original data. Top right: non-relevant features for classification. Bottom left: meaningful projections that preserve discriminability. Bottom right: final non-linear mapping.

Gaussian noise with the same mean. In this example, our algorithm finds a low dimensional non-linear embedding (a quadratic mapping) where the data is linearly separable. Moreover, it discards the undesirable dimension not relevant for classification. Observe that with a Gaussian Radial Basis Function (RBF) kernel, we could achieve similar classification performance in this particular problem; however, the Gaussian kernel hides the simplicity of the solution found by our algorithm (a quadratic mapping). Moreover, the Gaussian RBF kernel does not provide a mechanism to remove irrelevant features to increase classification accuracy.

The remainder of the paper is organized as follows: section 2 reviews previous work, section 3 introduces a parameterized kernel, and section 4 proposes an error function to fit it. Section 5 explores the relation of our parameterization with standard methods. Section 6 describes a gradient-based algorithm to learn the kernel. Section 7 provides the experiments and section 8 summarizes the conclusions.

2. Previous work

Since the introduction of kernel machines in the 90s, literature on metric and kernel learning(including learning the parameters and hyperparameters) has increased. It is beyond the scope of this paper to review all previous work along these lines, but a good review on metric learning can be found in [27], and for kernel selection methods see [21, 22].

A common approach for kernel matrix learning employs semi-definite programming (SDP) [16, 17, 5] to maximize some type of alignment with respect to an ideal kernel. A major limitation of SDP is its computational complexity [1]. This limitation has restricted its application to small scale problems. Recently, [15] posed the kernel selection for kernel linear discriminant analysis as a convex optimization problem. To optimize over a positive combination of known kernels, the authors used interior point methods with a computational cost of $O(d^3 + n^3)$, where d is the dimension of the samples and n the number of samples. Along these lines, [25] learns a Mahalanobis distance metric in the kNN classification setting by SDP. The learned distance metric enforces the k-nearest neighbors to belong always to the same class, while examples from different classes are separated by a large margin. A less computationally expensive approach, [13] utilize a boosting algorithm to learn "weak" kernel functions. In different but related work in computer vision, [26] learns a kernel matrix for non-linear dimensionality reduction via SDP. In the context of SVM, [3] employed a gradient descent algorithm to minimize the leave-one-out estimation of the generalization error over the kernel parameters.

In the literature of metric learning, Goldberger et al. [12] proposed Neighbourhood Component Analysis that computes the Mahalanobis distance that minimizes an approximation of the classification error. Similarly, [23] optimizes the linear discriminant analysis (LDA) criteria in a semi-supervised manner to estimate the metric. Conversely, other attempts such as those in [14, 9] find a distance metric to improve classification performance.

In previous work, typically, a parameterized family of linear or non-linear kernels (e.g. Gaussian, polynomial) is chosen, and the kernel parameters are tuned with some sort of cross-validation. In this paper, we consider the more generic problem of finding a functional mapping of the data. Moreover, we simultaneously reduce the dimensionality of the data and find the non-linear mapping using an algorithm of complexity $O(dn^2)$.

3. Parameterizing the Kernel

Many visual classification tasks (e.g. object recognition) are highly complex and non-linear kernels provide a good tool to model illumination, view-point, or internal object variability. Learning a non-linear kernel is a relatively difficult problem. For instance, proving that a function is a kernel is a challenging mathematical task. A given function is a kernel if, and only if, the value it produces for two vectors corresponds to a dot product in some feature Hilbert space. This is the well-known Mercer's theorem: *"Every positive definite, symmetric function is a kernel. For every kernel* $\mathbf{K}$*, there is a function* $\varphi(\mathbf{x}) : k(\mathbf{d}_1, \mathbf{d}_2) = \langle \varphi(\mathbf{d}_1), \varphi(\mathbf{d}_2) \rangle$*."*, where $\langle\rangle$ denotes dot product. To avoid the problem of proving that a similarity function is a kernel, it is common to parameterize the kernel as a positive combination of existing kernels (e.g. Gaussian, polynomial) [15].

In this paper, we propose to parameterize a kernel as a positive combination of normalized kernels as follows:

$$\begin{aligned} \mathbf{T} &= \mathbf{D}^T\mathbf{A}\mathbf{D} \qquad \hat{\mathbf{T}} = dm(\mathbf{T})^{-\frac{1}{2}}\mathbf{T}dm(\mathbf{T})^{-\frac{1}{2}} \\ \mathbf{T}_t &= \mathbf{D}^T\mathbf{A}_t\mathbf{D} \qquad \hat{\mathbf{T}}_t = dm(\mathbf{T}_t)^{-\frac{1}{2}}\mathbf{T}_t dm(\mathbf{T}_t)^{-\frac{1}{2}} \\ \mathbf{K}_1(\mathbf{A}, \boldsymbol{\alpha}) &= \textstyle\sum_{t=0}^{p} \alpha_t \hat{\mathbf{T}}^{\odot t} \\ \mathbf{K}_2(\mathbf{A}_1, \cdots, \mathbf{A}_p, \boldsymbol{\alpha}) &= \textstyle\sum_{t=0}^{p} \alpha_t \hat{\mathbf{T}}_t^{\odot t} \end{aligned} \tag{1}$$

where $\alpha_t \geq 0 \; \forall t$, p is the number of terms in the expansion, the columns of $\mathbf{D} \in \Re^{d \times n}$(see notation [1]) contain the original data points, d denotes the dimension of the data and n the number of samples. Each element ij of the matrix $\mathbf{T}$, $t_{ij} = \mathbf{d}_i^T\mathbf{A}\mathbf{d}_j$ contains the weighted dot product between the samples i and j. Each element ij of the matrix $\hat{\mathbf{T}}$ represents the weighted cosine of the angle between the samples i and j (i.e. $\hat{t}_{ij} = \frac{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j}{\sqrt{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i \mathbf{d}_j^T\mathbf{A}\mathbf{d}_j}}$). $\hat{\mathbf{T}}^{\odot k}$ exponentiates each of the entries in $\mathbf{T}$. $\mathbf{K}_1$ is a positive combination of $\hat{\mathbf{T}}^{\odot k}$, and if $\mathbf{A}$ is positive definite, $\mathbf{K}_1$ will be a valid kernel because of the closure properties of kernels [22]. The rank of each of the matrices $\hat{\mathbf{T}}$ and $\hat{\mathbf{T}}_t$ is less or equal to $\min\{d, n\}$, but $\mathbf{K}_1$ might be full rank. The same interpretation holds for $\mathbf{K}_2$ with the difference that for each kernel $\hat{\mathbf{T}}_t$ there is a different metric matrix $\mathbf{A}_t$, that is, each kernel might have a different subspace to project the data onto.

The kernel expansion suggested in eq. 1 is in spirit similar to the Taylor series expansion of a multivariate function. In fact, $\mathbf{K}_1$ and $\mathbf{K}_2$ can directly represent the polynomial kernel and are closely related to the exponential one. Consider a set of normalized samples (i.e. $\hat{\mathbf{d}}_i = \mathbf{d}_i / ||\mathbf{d}_i||_2$), the Taylor series expansion of two elements of the exponential

[1] Bold capital letters denote a matrix $\mathbf{D}$, bold lower-case letters a column vector $\mathbf{d}$. $\mathbf{d}_j$ represents the j column of the matrix $\mathbf{D}$. d_{ij} denotes the scalar in the row i and column j of the matrix $\mathbf{D}$ and the scalar i-th element of a column vector $\mathbf{d}_j$. All non-bold letters represent variables of scalar nature. $diag$ is an operator that transforms a vector to a diagonal matrix or takes the diagonal of the matrix into a vector. $dm(\mathbf{A})$ is a matrix that contains just the diagonal elements of $\mathbf{A}$. $\circ$ denotes the Hadamard or point-wise product. $\mathbf{1}_k \in \Re^{k \times 1}$ is a vector of ones. $\mathbf{I}_k \in \Re^{k \times k}$ is the identity matrix. $tr(\mathbf{A}) = \sum_i a_{ii}$ is the trace of the matrix $\mathbf{A}$ and $|\mathbf{A}|$ denotes the determinant. $||\mathbf{A}||_F = tr(\mathbf{A}^T\mathbf{A})$ designates the Frobenious norm of a matrix. $\mathbf{A}^{\odot k}$ denotes point-wise power, i.e. $a_{ij}^k \; \forall i, j$.

kernel will be $k_{ij} = e^{\frac{||\hat{\mathbf{d}}_i - \hat{\mathbf{d}}_j||_2^2}{\sigma^2}} = \sum_{r=0}^{\infty} \frac{(2)^r}{\sigma^{2r} r!}(1 - \hat{\mathbf{d}}_i^T \hat{\mathbf{d}}_j)^r$, which has same form as the expansion proposed in eq. 1 if $\mathbf{A}_t = \mathbf{I}_d \; \forall t$. On the other hand, $\mathbf{K}_1$ and $\mathbf{K}_2$ are not translational invariant kernels (unlike Gaussian RBF).

3.1. Dealing with high dimensional data

$\mathbf{A}_t \in \Re^{d \times d}$ is a matrix that captures the correlation relation between features. For high dimensional data (e.g. images) $\mathbf{A}_t$ is a very large matrix. For instance, consider a set of images of 100×100 pixels (vectors in $\Re^{10000}$), the metric matrix $\mathbf{A}_t$ will have dimensions of $\Re^{10000 \times 10000}$. For $\mathbf{A}_t$ to be full rank, we would need at least 10000 independent samples, and even that would result in a poor estimate. In our context, working with these very high dimensional matrices presents two problems: computational tractability (storage and efficiency) and lack of generalization (rank deficiency).

In order to generalize better and alleviate storage and computational demands, we follow recent work [7] and factorize the matrix $\mathbf{A}_t$ as a low dimensional subspace plus a noise term (scaled identity matrix). That is, we approximate each matrix $\mathbf{A}_t$ as $\mathbf{A}_t \approx \mathbf{B}_t \mathbf{B}_t^T + \lambda_t \mathbf{I}_d$ where $\lambda_t \geq 0 \in \Re$ and $\mathbf{B}_t \in \Re^{d \times k}$. Where k is the dimension of the subspace where to project the data. Factorizing the covariance as the sum of outer products and a diagonal matrix is an efficient (in space and time) manner to reduce the dimensionality of the data. If $\mathbf{A}_t$ is a covariance matrix, this factorization is the same as Probabilistic Principal Component Analysis [20, 24]. It is worthwhile to point out two important aspects of the previous factorizations. Firstly, observe that to compute $\mathbf{A}_t \mathbf{d}_i \approx \mathbf{B}_t(\mathbf{B}_t^T \mathbf{d}_i) + \lambda_t \mathbf{d}_i$ storing/computing the full $d \times d$ covariance is not required. Secondly, the original matrix $\mathbf{A}_t$ has $d(d+1)/2$ free parameters; whereas, the number of parameters after the factorization is reduced to $k(2d - k + 1)/2$ (assuming orthogonality of $\mathbf{B}_t$), and hence, it is not so prone to over-fitting.

4. Learning from an ideal kernel

In the previous section, we have proposed a possible expansion of a parameterized kernel. Ideally, we would like to directly optimize the kernel parameters to minimize the Bayes classification error; however, this is usually a hard task because the underlying distribution of the data is unknown and usually some sort of bounds are optimized instead. In this section, we explore the use of an ideal reference kernel to learn the kernel parameters.

In the ideal case, we would like to estimate the parameters of the kernel $(\mathbf{A}, \boldsymbol{\alpha})$ to produce a block diagonal matrix (assuming samples in the same class are contiguous). That is, if two samples belong to the same class, the kernel function should output a similarity of 1 and 0 otherwise. This ideal matrix can be expressed as the following factorization: $\mathbf{F} = \mathbf{G}\mathbf{G}^T$, where $\mathbf{G} \in \Re^{n \times c}$ is a dummy indicator matrix such that $\sum_j g_{ij} = 1$, $g_{ij} \in \{0, 1\}$ and g_{ij} is 1 if $\mathbf{d}_i$ belongs to class C_j. Recall that n refers to the number of samples and c the number of classes. A reasonable measure of distance between this "ideal" kernel matrix and the parameterized one is given by:

$$E_1(\mathbf{A}, \boldsymbol{\alpha}) = ||\mathbf{F} - \mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})||_F \propto \\ tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})^2) - 2tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})\mathbf{F}) \quad (2)$$

This measure of distance between kernels is closely related to the one proposed by [5]. Cristianini et al. [5] propose to minimize the alignment between kernels with: $E_2(\mathbf{A}, \boldsymbol{\alpha}) = \frac{tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})\mathbf{F})}{\sqrt{tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha}))}}$. Minimization of E_1 is more convenient to optimize and very similar (but not equivalent) to maximization of E_2. Observe that if we take the log of E_2 and change the sign, $E_2 \propto 0.5 log(tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})^2)) - log(tr(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha})\mathbf{F})$, which is closely related to minimization of E_1.

One drawback of eq. 2 is that it enforces the same similarity measure (i.e. 1) for two samples of the same class whether they lie near or far away in the input space. This behavior can produce over-fitting and remove important information regarding class discriminability. Moreover, we can have an unbalanced problem where a particular class has more samples than another class, and we would like to have a mechanism to compensate for that. Furthermore, any real data set contains a number of outliers that can bias the solution. To account for these situations, we introduce a matrix $\mathbf{W} \in \Re^{n \times n}$ that will individually weight each pairwise points. For instance, to account for outliers we will weight all the rows and columns of the outlying data as 0. To compensate for the fact that two samples have large dissimilarity in the input space, we will enforce a small link between these samples e.g. $w_{ij} = e^{-\frac{||\mathbf{d}_i - \mathbf{d}_j||_2^2}{\beta^2}} \quad \forall i \neq j$. To incorporate $\mathbf{W}$ in the formulation, we modify eq. 2 as:

$$E_3(\mathbf{A}, \boldsymbol{\alpha}) = ||\mathbf{W} \circ (\mathbf{F} - \mathbf{K}(\mathbf{A}, \boldsymbol{\alpha}))||_F \propto \\ tr((\mathbf{W} \circ \mathbf{K}(\mathbf{A}, \boldsymbol{\alpha}))(\mathbf{K}(\mathbf{A}, \boldsymbol{\alpha}) \circ \mathbf{W})^T) \\ -2tr((\mathbf{W} \circ \mathbf{K}(\mathbf{A}, \boldsymbol{\alpha}))(\mathbf{W} \circ \mathbf{F})) \quad (3)$$

5. Relationship with Linear Discriminant Analysis

It is interesting to point out the relationship between the proposed kernel expansion and Linear Discriminant Analysis (LDA) [8, 11]. LDA finds a low dimensional space where the means of the classes are far from each other while the within-class covariance is as compact as possible. The linear projection matrix $\mathbf{B}^{lda}$ can be computed in closed form by solving a generalized eigen-value problem. The

eigenvalue problem is defined between the second-order covariance matrices, conveniently expressed in matrix form as [7]:

$$(n-1)\mathbf{S}_t = \sum_{j=1}^{n}(\mathbf{d}_j - \mathbf{m})(\mathbf{d}_j - \mathbf{m})^T = \mathbf{D}\mathbf{P}_1\mathbf{D}^T$$

$$(n-1)\mathbf{S}_w = \sum_{i=1}^{c}\sum_{\mathbf{d}_j \in C_i}(\mathbf{d}_j - \mathbf{m}_i)(\mathbf{d}_j - \mathbf{m}_i)^T = \mathbf{D}\mathbf{P}_2\mathbf{D}^T$$

$$(n-1)\mathbf{S}_b = \sum_{i=1}^{c} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T = \mathbf{D}\mathbf{P}_3\mathbf{D}^T$$

where $\mathbf{S}_b$ is the between-class covariance matrix and represents the average of the distances between the mean of the classes. $\mathbf{S}_w$ represents the within-class covariance matrix, and it is a measure of the average compactness of each class. $\mathbf{S}_t$ is the total covariance matrix. $\mathbf{m} = \frac{1}{n}\mathbf{D}\mathbf{1}_n$ is the mean vector of all the samples, and $\mathbf{m}_i$ is the mean vector for the class i. n_i is the number of samples for class i and $\sum_{i=1}^{c} n_i = n$. $\mathbf{P}_i$ are projection matrices ($\mathbf{P}_i^T = \mathbf{P}_i$ and $\mathbf{P}_i^2 = \mathbf{P}_i$) with the following expressions [7]:

$$\mathbf{P}_1 = \mathbf{I} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^T \quad \mathbf{P}_2 = \mathbf{I} - \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$$
$$\mathbf{P}_3 = \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^T \tag{4}$$

One possible optimization criteria for LDA minimizes the following expression:

$$tr\big((\mathbf{B}^{lda^T}\mathbf{D}\mathbf{D}^T\mathbf{B}^{lda})^{-1}(\mathbf{B}^{lda^T}\mathbf{D}\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{D}^T\mathbf{B}^{lda})\big)$$

where $(\mathbf{G}^T\mathbf{G})^{-1} \in \Re^{c\times c}$ is a diagonal matrix containing the inverse of the number of samples per class. A closed form solution for $\mathbf{B}^{lda}$ can be obtained by solving the following generalized eigenvalue problem:

$$\mathbf{D}\mathbf{D}^T\mathbf{B}^{lda} = \mathbf{D}\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{D}^T\mathbf{B}^{lda}\mathbf{\Lambda} \tag{5}$$

Consider the simplified case, where the matrix $\mathbf{A}_t = \mathbf{B}_t\mathbf{B}_t^T$, that is, we only select the second term of the expansion of $\mathbf{K}_2$. Finding the optimal $\mathbf{B}_t$ that minimizes:

$$E_4(\mathbf{B}_t) = ||(\mathbf{D}^T\mathbf{B}_t\mathbf{B}_t^T\mathbf{D}) - \mathbf{F}||_F \tag{6}$$

w.r.t. $\mathbf{B}_t$, and using the fact that $\mathbf{B}_t^T\mathbf{D}\mathbf{D}^T\mathbf{B}_t = \mathbf{\Lambda}_2$, it can be shown that the solution of E_4 satisfies the following eigen-equation:

$$\mathbf{D}\mathbf{D}^T\mathbf{B}_t = \mathbf{D}\mathbf{F}\mathbf{D}^T\mathbf{B}_t\mathbf{\Lambda}_2 \tag{7}$$

Observe that if $\mathbf{F} = \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$, we will have the same solution as LDA, that is, eq. (5) and eq. (7) are equivalent. Our kernel expansion is a consistent non-linear generalization of standard LDA [8, 11].

If $d << n$ and $\mathbf{D}\mathbf{D}^T$ is full rank, standard packages for generalized eigenvalue problems can efficiently solve eq. (5). However, for high dimensional data ($d >> n$) solving the previous equation directly is not computationally efficient in neither space nor time. Fortunately, using the fact that the solutions of $\mathbf{B}$ are linear combinations of the data (i.e. $\mathbf{B}$=$\mathbf{D}\boldsymbol{\alpha}$), multiplying both sides by $\mathbf{D}^T$ and assuming $\mathbf{D}^T\mathbf{D}$ is invertible, the original eigenvalue problem is equivalent to solve $\mathbf{F}\mathbf{D}^T\mathbf{D}\boldsymbol{\alpha} = \mathbf{D}^T\mathbf{D}\boldsymbol{\alpha}\mathbf{\Lambda}_3$, which is of much lower dimension ($n \times n$) [6].

6. Learning the Kernel

In this section, we derive several optimization strategies to learn the parameterized kernel.

6.1. Optimization

In the interest of space, we derive the optimization rules for the most generic error function. We show how to optimize:

$$E_5(\mathbf{A}_1, \cdots, \mathbf{A}_p, \boldsymbol{\alpha}) = ||\mathbf{W} \circ (\mathbf{F} - \mathbf{K}_2)||_F \tag{8}$$

w.r.t. $\mathbf{A}_1, \cdots, \mathbf{A}_p, \boldsymbol{\alpha}$, since optimizing $\mathbf{K}_1$ is very similar. To optimize eq. 8, we use an alternating strategy of fixing $\mathbf{A}_t$ parameters and optimizing w.r.t. $\boldsymbol{\alpha}$ and vice versa. This will monotonically decrease the error of $\mathbf{E}_5$.

We use a gradient descent approach to efficiently and incrementally optimize for $\mathbf{A}_t$. The gradient updates are given by:

$$\mathbf{A}_t^{n+1} = \mathbf{A}_t^n - \eta\tfrac{\partial E_5}{\partial \mathbf{A}_t} \tag{9}$$
$$\tfrac{\partial E_5}{\partial \mathbf{A}_t} = 2\alpha_t t\mathbf{D}(\mathbf{M}_2 - \mathbf{M}_3)\mathbf{D}^T \quad \forall\, t$$
$$\mathbf{M}_1 = (\mathbf{K}_2 - \mathbf{F}) \circ \hat{\mathbf{T}}_t^{\odot(t-1)} \circ \mathbf{W}^{\odot 2}$$
$$\mathbf{M}_2 = dm(\mathbf{T}_t)^{-\frac{1}{2}}\mathbf{M}_1 dm(\mathbf{T}_t)^{-\frac{1}{2}}$$
$$\mathbf{M}_3 = dm(\mathbf{T}_t)^{-\frac{3}{2}} diag\big((dm(\mathbf{T}_t)^{-\frac{1}{2}}\mathbf{M}_1 \circ \mathbf{T}_t)\mathbf{1}_n\big)$$

The major problem with the update of eq. 9 is determining the optimal η. In our case, η is found with a line search strategy [10]. Also, observe that it might be the case that $\mathbf{A}_t$ is no longer positive semi-definite. One possible solution is to compute the eigenvectors of $\mathbf{A}_t$ and add an identity matrix scaled by absolute value of the largest negative eigenvalue. On the other hand, we can guarantee a positive semidefinite matrix by computing the gradient w.r.t. $\mathbf{B}_t$:

$$\mathbf{B}_t^{n+1} = \mathbf{B}_t^n - \eta\tfrac{\partial E_5}{\partial \mathbf{B}_t} \tag{10}$$
$$\tfrac{\partial E_5}{\partial \mathbf{B}_t} = 2\alpha_t t\mathbf{D}(\mathbf{M}_2 - \mathbf{M}_3)\mathbf{D}^T\mathbf{B}_t \;\; \forall t$$

At this point, it is worthwhile to mention that the complexity of the updates is $O(dn^2)$, far less expensive than SDP approaches. λ_t is optimized using the *fmincon* function from $Matlab$.

Once all $\mathbf{A}_t$ have been updated, $\boldsymbol{\alpha}$ values can be optimized using quadratic programming. After rearranging, eq. 8 can be expressed as:

$$E_6(\boldsymbol{\alpha}) \propto \boldsymbol{\alpha}^T \mathbf{Z}\boldsymbol{\alpha} - 2\mathbf{p}^T\boldsymbol{\alpha} \quad \boldsymbol{\alpha} \geq \mathbf{0} \tag{11}$$

where $z_{ij} = \sum_{lk} w_{lk}^2 k_{lk}^i k_{lk}^j$ and $p_i = \sum_{lk} w_{lk}^2 f_{lk} k_{lk}^i$. Recall that k_{ij} corresponds to the ij element of $\mathbf{K}_2$. We use the *quadprog* function from $Matlab$ to optimize w.r.t. $\boldsymbol{\alpha}$ while satisfying $\boldsymbol{\alpha} \geq 0$.

6.2. Initialization and other issues

Minimizing eq. (8) with respect to $\boldsymbol{\alpha}, \mathbf{A}_1, \cdots, \mathbf{A}_p$ is a non-convex optimization problem prone to many local minima. Without a good initial estimation, the previous optimization scheme easily converges to a local minima. To get a reasonable estimation, we initialize each of the parameters $\mathbf{A}_1, \cdots, \mathbf{A}_p$ with the LDA solution and the means of the clusters resulting from k-means clustering. The $\boldsymbol{\alpha}$ vector is initialized with a uniform value. Moreover, we start from several random initial points and select the solution with minimum error after convergence.

To avoid over-fitting problems and for computational convenience, we stochastically train the algorithm. That is, we randomly select subsets of training data, run a few iterations of the gradient descent algorithm, select other random subsets of data, and proceed in this manner until convergence.

7. Experiments

In this section, we report comparative results of our algorithm with standard SVM approaches in image classification problems. In all of the experiments, we have used the C-SVM from the LIBSVM [2].

7.1. Synthetic data

Consider fig. (1), where 200 samples have been generated from four $3D$ Gaussians (50 each) for two different classes (XOR problem). For each of the Gaussians, the z coordinate is noise with same mean and high variance. Observe that the XOR problem is not linearly separable and a non-linear kernel is needed.

In this case, we learn a matrix $\mathbf{A} \in \Re^{3\times 3}$ common to all the kernels. After convergence, the rank of matrix $\mathbf{A}$ is 3 with eigenvalues $l_1 = 1.9860$ $l_2 = 0.6843$ $l_3 = 0.0009$. The smallest eigenvalue corresponds to the eigenvector aligned with the z direction, where the non-discriminative information lies. That is, the null space of $\mathbf{A}$ contains the random non-discriminative directions. Even more interesting is the interpretation of the $\boldsymbol{\alpha}$ vector. All of the α_t parameters are close to zero except for the powers of two. This is because, for samples within the same cluster, the cosine of the angle will be approximately 1; between the samples of a different cluster and same class the cosine will be -1; between clusters of different classes will be approximately 0. Hence, by converting the negative -1 values to 1 by powering to an even number, we will achieve the ideal matrix (assuming zero mean data).

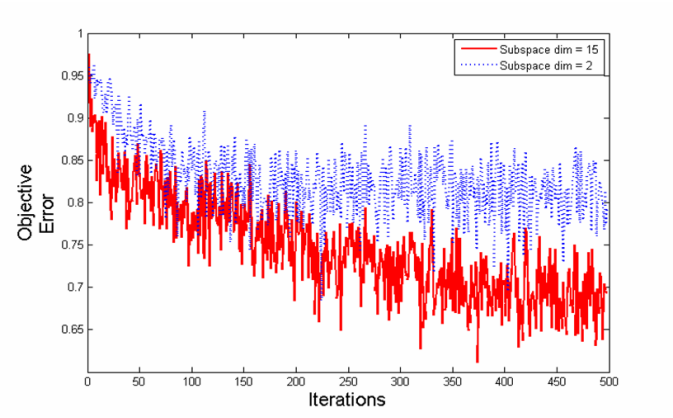
Figure 2. Top row: examples images of pedestrians. Bottom row: examples of non-pedestrian images misclassified as pedestrians by the algorithm.

7.2. Pedestrian detection

Detecting people in images is key to a number of applications, ranging from intelligent vehicles to surveillance systems or robotics. In this experiment, we will make use of a challenging database on pedestrians recently published by [18]. This database is especially difficult because the non-pedestrian examples are the false positives of a shape-based pedestrian detector. The database consists of three training data sets and two testing data sets. Each data set has 4800 images (18×36 pixels) with a pedestrian in the middle and 5000 non-pedestrian examples. Fig. 2 shows a few images from this database.

Since the amount of available data for training is large ($\approx$ 20000 samples), we use the stochastic version of the algorithm in chunks of 400 samples. We use two of the training sets for learning the parameters and the third one to tune with cross-validation the C parameter in the C-SVM algorithm and the scale parameter in the Gaussian RBF kernel. The two testing data sets are used just for testing purposes. Fig. (3.a) shows the error of eq. 8 versus the number of iterations and fig. (3.b) shows the classification accuracy versus the iterations. As expected, the error decreases on average due to the stochastic behavior of the minimization, and it also provides (on average) a better classification accuracy. In this case, we have augmented the data vector as $[\mathbf{d}_i 1]$ to take into account the mean value. The matrix $\mathbf{W}$ was set all to one.

Having three training sets and two data tests, we can compute a total of six different ROC curves [18]. In fig. 4, we show the ROC mean curve. Fig. (2) shows some true positives (top row) and some false positives (bottom row) produced by our algorithm. In table 1, we show the recognition performance in comparison with linear and Gaussian RBF kernels.



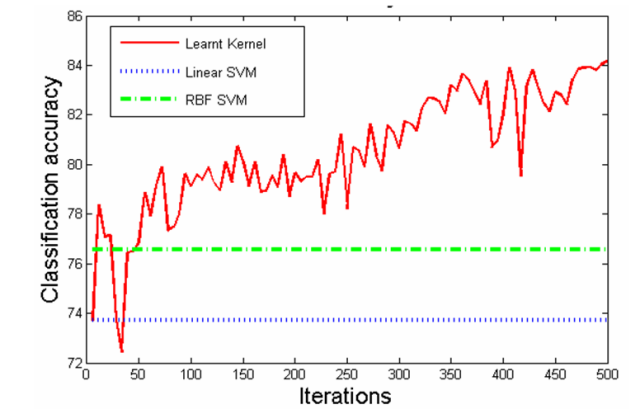


Figure 3. a) Error versus iterations for two values of k (dimension of the subspace). b) Classification accuracy versus iterations.

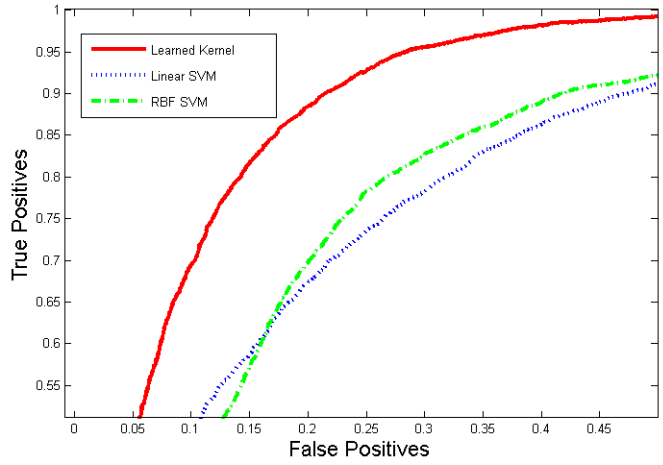


Figure 4. ROC curves for linear, Gaussian RBF and the parameterized kernel for SVM classifier.

7.3. Mouth detector

In our second experiment, we test the performance of the algorithm detecting the mouths in 14 images taken with a regular digital camera (see fig. 6). The kernel has been learned from a set of 2500 mouths (positive examples) taken from the IBM ViaVoice AV database [19], and aligned with Procrustes [4] (see fig. 5). The 2500 negative examples are selected from patches of the face that do not contain the mouth.

Features	Linear	Exponential	Ours
Graylevel	73.9	76.4	**84.2**

Table 1. Average classification results for different kernels

Figure 5. Some training examples of the IBM Database.

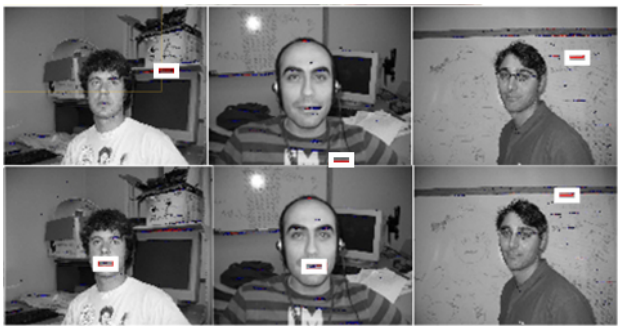


Figure 6. First row: Gaussian RBF SVM does not correctly detect the mouth in three test. Second row: the same images with the learned parameterized kernel SVM (2 out of 3 are correct).

Given an image with one or more frontal faces and an estimate of the scale factor, we search over all possible locations in the image. Evaluating the kernel at each location (x, y) can be computationally expensive. For a particular position (x, y) computing the projection $\mathbf{B}_t^T \mathbf{d}_i$ is equivalent to correlating the image with each basis vector of the subspace $\mathbf{B}_t$, and stacking all the values for each pixel. For large regions, this correlation is performed efficiently in the frequency domain by using the Fast Fourier Transform (FFT) (i.e. $\mathbf{C}_1 = \mathbf{b}_1^T \mathbf{I} = IFFT(FFT(\mathbf{b}_1) \circ FFT(\mathbf{I}))$). This fast search is another advantage of our formulation. Fig. (6) shows some examples of the detection performance of the RBF-SVM versus our parameterized kernel. In the first row, the Gaussian RBF kernel does not correctly detect any of the three mouths in the images; whereas, in the second row our learned kernel detects two. Fig. (7) shows the average ROC curve over 14 images for the learned kernel and RBF kernel. The parameters of the RBF kernel (i.e. $e^{\frac{-||\mathbf{X}_i - \mathbf{X}_j||_2^2}{2\sigma^2}}$, σ and the C in the C-SVM are tuned with cross-validation procedure.

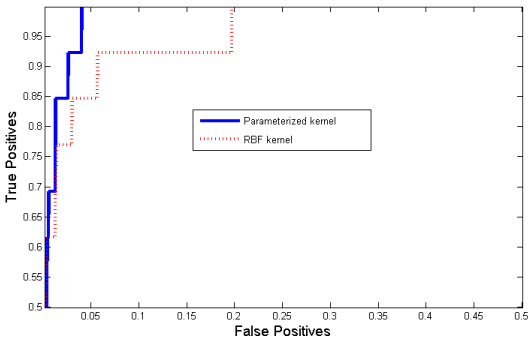


Figure 7. Roc curves for RBF and parameterized kernel.

8. Conclusions and Future Work

In this paper, we have proposed an efficient algorithm to learn a parameterized kernel matrix. In all the experiments, the learned kernel outperformed standard kernels such as polynomial or RBF after cross-validation. Although this preliminary work has shown promising results, several issues to be addressed: the need for better optimization strategies that avoid local minima problems and can process a large amount of data with a computational cost of less than $O(dn^2)$. We are also looking for other possible kernel parameterizations that can extract interesting image features in a supervised or unsupervised manner.

Acknowledgements This work was partially supported by the National Institute of Justice award 2005-IJ-CX-K067 and the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Thanks Minh Hoai and Tomas Simon to for helpful comments.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 2

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm. 5

[3] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 2002. 2

[4] T. Cootes and C. Taylor. Statistical models of appearance for computer vision. tech. report. university of manchester. 2001. 6

[5] N. Cristianini, J. Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *Neural Information Processing Systems*, 2002. 2, 3

[6] F. de la Torre, R. Gross, S. Baker, and V. Kumar. Representational oriented component analysis (roca) for face recognition with one sample image per training class. In *Computer Vision and Pattern Recognition*, 2005. 4

[7] F. de la Torre and T. Kanade. Multimodal oriented discriminant analysis. In *International Conference on Machine Learning*, pages 177–184, 2005. 3, 4

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons Inc., 2001. 3, 4

[9] M. Fink. Object classification from a single example utilizing class relevance metrics. In *Neural Infonation Processing Systems*, pages 449–456, 2005. 2

[10] R. Fletcher. *Practical methods of optimization*. John Wiley and Sons., 1987. 4

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition*. Academic Press.Boston, MA, 1990. 3, 4

[12] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Neural Information Processing Systems*, pages 513–520, 2004. 2

[13] T. Hertz, A. B. Hillel, and D. Weinshall. Learning a kernel function for classification with small training samples. In *International conference on Machine learning*, pages 401–408, 2006. 2

[14] J. Kandola, N. Cristianini, and J. Shawe-Taylor. Learning semantic similarity. In *Neural Infonation Processing Systems*, 2003. 2

[15] S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel fisher discriminant analysis. In *International Conference on Machine Learning*, pages 465–472, 2006. 2

[16] G. Lanckriet, N. Cristianini, and P. Bartlett. Learning the kernel matrix with semi-definite programming. 5:27–72, 2004. 2

[17] G. Lanckriet, N. Cristianini, P. Bartlett, E. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. (5):27–72, 2004. 2

[18] S. Munder and D. Gavrilla. An experimental study on pedestrian classifcation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006. 5

[19] C. Neti, G. Potamianos, J. Luettin, I. Matthews, H. Glotin, D. Vergyri, J. Sison, A. Mashari, and J. Zhou. Audio-visual speech recognition. Technical Report WS00AVSR, Johns Hopkins University, CLSP, 2000. 6

[20] S. Roweis. EM algorithms for PCA and SPCA. In *Neural Information Processing Systems*, pages 626–632, 1997. 3

[21] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002. 1, 2

[22] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. 1, 2

[23] N. Shental, T. Hertz, D. Weinshall, and M. Pavel. Adjustment learning and relevant component analysis. In *European Conferenceon Computer Vision*, pages 776–790, 2002. 2

[24] M. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 61:611–622, 1999. 3

[25] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Neural Information Processing Systems*, 2006. 2

[26] K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Computer Vision and Pattern Recognition*, pages 988–995, 2004. 2

[27] L. Yang. Distance metric leraning: A comprehensive survey, 2006. 2