PARAMETERIZED KERNELS FOR SUPPORT VECTOR MACHINE CLASSIFICATION

Fernando De la Torre and Oriol Vinyals
Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

Keywords: Visual Learning, Kernel methods, Support Vector Machines, Metric learning.

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 the space of parameterized kernels using a gradient-based method. Our method effectively learns a non-linear representation of the data useful for classification and simultaneously performs dimensionality reduction. In addition, we introduce 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 (Schlkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) 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 mostly 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 (i.e. a kernel) useful to improve classification in kernel machines.

Fig. 1 shows the main point of this paper. We have synthetically generated 2 multimodal three dimensional Gaussian classes. Two of the dimensions are relevant for classification and the other dimension is high-variance random Gaussian noise. Our algorithm finds a low dimensional non-linear embedding of the data where the data is linearly separable. In this particular example, our algorithm automatically finds that the best mapping is a quadratic one, while discarding the undesirable dimension not relevant for

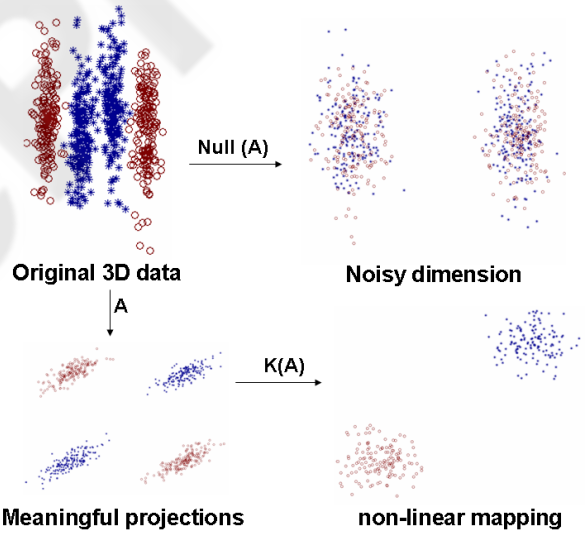

Figure 1: Learning a non-linear mapping optimal for classification. Top left: original data. Top right: useless features for classification. Bottom left: meaningful projections that preserve discriminability. Bottom right: final non-linear learned mapping.

De la Torre F. and Vinyals O. (2007).
PARAMETERIZED KERNELS FOR SUPPORT VECTOR MACHINE CLASSIFICATION.
In *Proceedings of the Second International Conference on Computer Vision Theory and Applications - IU/MTSV*, pages 116-121

classification. Observe that with a Gaussian kernel we could achieve similar classification performance in this particular problem (assuming some tuning of the parameters is done); however, the exponential kernel hides the simplicity of the solution found by our algorithm (a quadratic mapping).

2 PREVIOUS WORK

Since the introduction of kernel machines in the 90's, there has been growing literature on metric and kernel learning. 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 (Yang, 2006), and for kernel selection methods see (Schlkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004).

A common approach to learn a kernel matrix uses semi-definite programming (SDP) (Lanckriet et al., 2004; Cristianini et al., 2001) to maximize some sort of alignment with respect to an ideal kernel. A major limitation of SDP is its computational complexity, since it scales $O(n^6)$, where n is the number of samples (Boyd and Vandenberghe, 2004). This limitation has restricted its application to small scale problems. Recently, (Kim et al., 2006) posed the kernel selection for kernel linear discriminant analysis as a convex optimization problem. To optimize over a positive combination of known kernels, they use interior point methods with a computational cost of $O(d^3+n^3)$, where d is the dimension of the samples. Along these lines, (Weinberger et al., 2006) learns a Mahalanobis distance metric in the kNN classification setting by SDP. The learned distance metric enforces the k-nearest neighbors to belong to the same class, while examples from different classes are separated by a large margin.

In the literature of metric learning, Goldberger et al. (Goldberger et al., 2004) have proposed Neighbor component analysis that computes the Mahalanobis distance that minimizes an approximation of the classification error. Similarly, (Shental et al., 2002) optimizes the linear discriminant analysis (LDA) criteria in a semi-supervised manner to learn the metric.

In previous work, typically, a parameterized family of linear or non-linear kernels, (e.g. Gaussian, polynomial) are 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.

3 PARAMETERIZING THE KERNEL

Many visual classification tasks (e.g. object recognition) are highly complex and non-linear kernels are needed to model changes such as illumination, viewpoint 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, ...).

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

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

where $\alpha_t \geq 0\ \forall t$, 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, n the number of samples and p the degree of the polynomial. Each element ij of the matrix $\mathbf{T}$, $t_{ij} = \mathbf{d}_i^T\mathbf{A}\mathbf{d}_j$ contains the dot weighted product between the sample i and j. Each element ij of the matrix $\hat{\mathbf{T}}$ represents the cosine of the angle between the samples i and j (i.e. $\hat{t}_{ij} = \frac{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j}{\sqrt{\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i}}$). $\hat{\mathbf{T}}^{\odot k}$ 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

[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 will 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. $a_{ij}^k\ \forall i,j$.

properties of kernels (Shawe-Taylor and Cristianini, 2004). The rank of each of the matrices $\hat{\mathbf{T}}$ and $\hat{\mathbf{T}}_t$ is $\preceq \min\{d,n\}$, 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 matrix $\mathbf{A}_t$ models correlations between variables.

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 represent directly 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 kernel will be $k_{ij} = e^{\frac{||\hat{\mathbf{d}}_i - \hat{\mathbf{d}}_j||_2^2}{\sigma^2}} = \sum_{i=0}^{\infty} \frac{(2)^i}{\sigma^{2i} i!}(1 - \hat{\mathbf{d}}_i^T \hat{\mathbf{d}}_j)^i$, which has an identical form of the expansion proposed in eq. 1 if $\mathbf{A}_t = \mathbf{I}_d \; \forall t$. Observe that our kernel expansion is more flexible and it will learn a mapping useful for classification from training data. However, $\mathbf{K}_1$ and $\mathbf{K}_2$ are not translational invariant kernels.

3.1 Dealing with High Dimensional Data

For high dimensional data (e.g. images) $\mathbf{A} \in \Re^{d \times d}$ is a big matrix that captures the correlation relation between the features. In our context, working with these very high dimensional matrices presents two problems: computational tractability (storage, efficiency and rank deficiency) and generalization.

In order to be able to generalize better and to not suffer from storage/computational limitations, we follow recent work (de la Torre and Kanade, 2005) and factorize the matrix $\mathbf{A}$ as a low dimensional subspace plus a noisy 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 $\mathbf{B}_t \in \Re^{d \times k}$. It is worthwhile to point out two important aspects of the previous factorizations. 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. Firstly, observe that to compute $\mathbf{A}\mathbf{d}_i \approx \mathbf{B}(\mathbf{B}^T \mathbf{d}_i) + \lambda \mathbf{d}_i$ storing/computing the full $d \times d$ covariance is not required. Secondly, the original matrix $\mathbf{A}$ has $d(d+1)/2$ free parameters, and after the factorization the number of parameters is reduced to $k(2d-k+1)/2$ (assuming orthogonality of $\mathbf{B}$), and hence 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 a ideal reference kernel to learn the kernel parameters.

In the ideal case, we would like to estimate the parameters of the kernel $(\mathbf{A}, \alpha)$ to produce a block diagonal matrix (assuming samples are ordered). That is, in all the samples that belong to the same class the kernel function should output a similarity of 1 and 0 otherwise. This ideal matrix can be computed with the matrix $\mathbf{G}$ as $\mathbf{F} = \mathbf{G}\mathbf{G}^T$. A reasonable measure of distance between the ideal kernel and the parameterized one is given by:

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

This measure of distance between kernels is closely related to the one proposed by (Cristianini et al., 2001). Cristianini et al. propose to minimize the alignment between kernels with: $E_2(\mathbf{A}, \alpha) = \frac{tr(\mathbf{K}(\mathbf{A},\alpha)\mathbf{F})}{\sqrt{tr(\mathbf{K}(\mathbf{A},\alpha)\mathbf{K}(\mathbf{A},\alpha))}}$. Minimization of E_1 is more convenient to optimize and very similar (but not equivalent) to maximization of E_2. Recall that if we take the log E_2 and change the sign, $E_2 \propto 0.5log(tr(\mathbf{K}(\mathbf{A}, \alpha)^2)) - logtr(\mathbf{K}(\mathbf{A}, \alpha)\mathbf{F})$.

One drawback of eq. 2 is that it enforces the same similarity measure (i.e. 1) for two samples of the same class that are near or far away in the input space. This behavior can produce over-fitting and it can remove important information regarding class discriminability. Moreover, we can have an unbalanced problem where a particular class has more samples than another one, and we would like 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 new distance matrix $\mathbf{W} \in \Re^{n \times n}$ that will weight individually each pair-wise points. For instance, to account for outliers we will weight all the rows and columns of the outlying data as 0, or to compensate for the fact that two samples have large dissimilarity in the input space, we 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:

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

5 LEARNING THE KERNEL

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

5.1 Optimizing w.r.t A and α

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

$$E_4(\mathbf{A}_1,\cdots,\mathbf{A}_p,\alpha) = ||\mathbf{W}\circ(\mathbf{F}-\mathbf{K}_2)||_F \quad (4)$$

w.r.t. $\mathbf{A}_1,\cdots,\mathbf{A}_p,\alpha$, since optimizing $\mathbf{K}_1$ is very similar. To optimize eq. 4, we use an alternating strategy of fixing $\mathbf{A}_t$ parameters and optimize w.r.t. α and vice versa. This will monotonically decrease the error of $\mathbf{E}_4$. If α is known, optimizing over $\mathbf{A}_t$ matrix can be formulated as a semi-definite programming problem (SDP); however, the computational complexity is large. Instead, we use a gradient descent approach to incrementally optimize for $\mathbf{A}_t$. The gradient updates are given by:

$$\mathbf{A}_t^{n+1} = \mathbf{A}_t^n - \eta\frac{\partial E_4(\mathbf{K}_2)}{\partial \mathbf{A}_t} \quad (5)$$

$$\frac{\partial E_4}{\partial \mathbf{A}_t} = 2\alpha_t t\mathbf{D}(\mathbf{M}_2-\mathbf{M}_3)\mathbf{D}^T \ \forall t$$

$$\mathbf{M}_1 = (\mathbf{K}_1-\mathbf{F})\circ\hat{\mathbf{T}}_t^{\odot(t-1)}\circ\mathbf{W}^{\odot 2}$$

$$\mathbf{M}_2 = dm(\mathbf{T}_t)^{-\frac{1}{2}}\mathbf{M}_1 dm(\mathbf{T}_t)^{-\frac{1}{2}}$$

$$\mathbf{M}_3 = dm(\mathbf{T}_t)^{-\frac{3}{2}} diag\big((dm(\mathbf{T}_t)^{-\frac{1}{2}}\mathbf{M}_1\circ\mathbf{T}_t)\mathbf{1}_n\big)$$

The major problem with the update of eq. 5 is to determine the optimal η. In our case, η is determined with a line search strategy (Fletcher, 1987). Similarly, for high dimensional data the gradient w.r.t $\mathbf{B}_t$ is:

$$\mathbf{B}_t^{n+1} = \mathbf{B}_t^n - \eta\frac{\partial E_4(\mathbf{K}_2)}{\partial \mathbf{B}_t} \quad (6)$$

$$\frac{\partial E_4}{\partial \mathbf{B}_t} = 2\alpha_t t\mathbf{D}(\mathbf{M}_2-\mathbf{M}_3)\mathbf{D}^T\mathbf{B}_t \ \ \forall t$$

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

Once all $\mathbf{A}_t\ \forall t$ have been updated, α values can be optimized using quadratic programming. After rearranging, eq. 4 can be expressed as:

$$E_5(\alpha) \propto \alpha^T\mathbf{Z}\alpha - 2\mathbf{p}^T\alpha \quad \alpha \geq \mathbf{0} \quad (7)$$

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.α to ensure positiveness.

5.2 Initialization and Other Issues

Minimizing eq. 4 w.r.t to $\alpha,\mathbf{A}_1,\cdots,\mathbf{A}_p$ 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 α values are initialized with the same 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 train the algorithm stochastically. That is, we randomly select subsets of training data, run few iterations of the gradient descent algorithm, select other random subset of data and proceed this way until convergence.

6 EXPERIMENTS

In this section, we report comparative results of our algorithm with standard SVM approaches in image classification problems. In all the experiments we have used the C-SVM implementation (Chang and Lin, 2001).

6.1 Synthetic Data

Consider fig. 1, where 200 samples have been generated from four $3D$ Gaussians (50 each) from 2 different classes (”exclusive or” (XOR) problem). For each of the Gaussians, the z coordinate is random noise of high variance.

In this case, we learn a common matrix $\mathbf{A}\in\Re^{3\times 3}$ and the α values. After convergence, the rank of the matrix $\mathbf{A}$ is 3 with eigenvalues $l_1 = 1.9860$ $l_2 = 0.6843$ $l_3 = 0.0009$, the small 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 α parameters. All the α parameters are close to 0 except for the powers of 2. 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 but of the same class

Table 1: Average classification results for different kernels.

Features	Linear	Exponential	Ours
Graylevel	73.9	76.4	**84.2**
Multiband	72.4	79.1	**84.9**

Figure 5: a) Some training examples of the IBM Database. b) First row the same images with our learned kernel (3/3). Second row some test images using the RBF SVM (1/3).

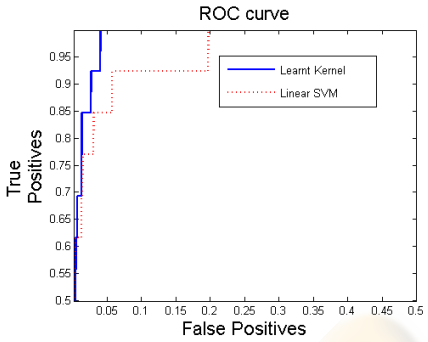

Figure 6: Roc curves of RBF versus our algorithm.

possible locations of 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 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 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 the average ROC curve over 14 images for our 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 a cross validation procedure. Similarly the C parameter in the C-SVM is tuned with cross-validation. Fig. 5.b shows some examples of the detection performance of the RBF-SVM versus our learned kernel. In the first row, our learned kernel detects two out of three mouths, whereas in the second row RBF only detects one.

ACKNOWLEDGEMENTS

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

REFERENCES

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

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

Cootes, T. F. and Taylor, C. J. (2001). Statistical models of appearance for computer vision. In *http://www.isbe.man.ac.uk/bim/refs.html*.

Cristianini, N., Taylor, J. S., and Elisseeff, A. (2001). On kernel-target alignment. In *NIPS*.

de la Torre, F. and Kanade, T. (2005). Multimodal oriented discriminant analysis. In *ICML*.

Fletcher, R. (1987). *Practical methods of optimization*. John Wiley and Sons.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004). Neighbourhood component analysis. In *NIPS*, pages 513–520.

Kim, S.-J., Magnani, A., and Boyd, S. (2006). Optimal kernel selection in kernel fisher discriminant analysis. In *ICML*, pages 465–472.

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

Munder, S. and Gavrilla, D. (2006). An experimental study on pedestrian classifcation.

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

Schlkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge university press.

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

Weinberger, K., Blitzer, J., and Saul, L. (2006). Neighbourhood component analysis. In *NIPS*.

Yang, L. (2006). Distance metric leraning: A comprehensive survey.