Fisher Linear Discriminant Analysis

Max Welling
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, M5S 3G5 Canada
welling@cs.toronto.edu

Abstract

This is a note to explain Fisher linear discriminant analysis.

1 Fisher LDA

The most famous example of dimensionality reduction is "principal components analysis". This technique searches for directions in the data that have largest variance and subsequently project the data onto it. In this way, we obtain a lower dimensional representation of the data, that removes some of the "noisy" directions. There are many difficult issues with how many directions one needs to choose, but that is beyond the scope of this note.

PCA is an unsupervised technique and as such does not include label information of the data. For instance, if we imagine 2 cigar like clusters in 2 dimensions, one cigar has $y = 1$ and the other $y = -1$. The cigars are positioned in parallel and very closely together, such that the variance in the total data-set, ignoring the labels, is in the direction of the cigars. For classification, this would be a terrible projection, because all labels get evenly mixed and we destroy the useful information. A much more useful projection is orthogonal to the cigars, i.e. in the direction of least overall variance, which would perfectly separate the data-cases (obviously, we would still need to perform classification in this 1-D space).

So the question is, how do we utilize the label information in finding informative projections? To that purpose Fisher-LDA considers maximizing the following objective:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{1}$$

where S_B is the "between classes scatter matrix" and S_W is the "within classes scatter matrix". Note that due to the fact that scatter matrices are proportional to the covariance matrices we could have defined J using covariance matrices – the proportionality constant would have no effect on the solution. The definitions of the scatter matrices are:

$$S_B = \sum_c (\boldsymbol{\mu}_c - \bar{\mathbf{x}})(\boldsymbol{\mu}_c - \bar{\mathbf{x}})^T \tag{2}$$

$$S_W = \sum_c \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T \tag{3}$$

where $\bar{x}$ is the overall mean of the data-cases. Oftentimes you will see that for 2 classes S_B is defined as $S'_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$. This is the scatter of class 1 with with respect to

the scatter of class 2 and hence corresponds to computing the scatter relative to a different vector. By using the general transformation rule for scatter matrices:

$$S_{\mu+\nu} = S_\mu + N\boldsymbol{\nu}\boldsymbol{\nu}^T + 2N\boldsymbol{\nu}(\boldsymbol{\mu} - \bar{x})^T \tag{4}$$

with $S_\mu = \sum_i (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$ we can deduce that the only difference is a constant shift not depending on any relative distances between point. It will therefore have no impact on the final solution.

Why does this objective make sense. Well, it says that a good solution is one where the class-means are well separated, measured relative to the (sum of the) variances of the data assigned to a particular class. This is precisely what we want, because it implies that the gap between the classes is expected to be big.

An important property to notice about the objective J is that is is invariant w.r.t. rescalings of the vectors $\mathbf{w} \rightarrow \alpha\mathbf{w}$. Hence, we can always choose $\mathbf{w}$ such that the denominator is simply $\mathbf{w}^T S_W \mathbf{w} = 1$, since it is a scalar itself. For this reason we can transform the problem of maximizing J into the following constrained optimization problem,

$$\min_{\mathbf{w}} \quad -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \tag{5}$$

$$\text{s.t.} \quad \mathbf{w}^T S_W \mathbf{w} = 1 \tag{6}$$

corresponding to the lagrangian,

$$\mathcal{L}_P = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1) \tag{7}$$

(the halves are added for convenience). The KKT conditions tell us that the following equation needs to hold at the solution,

$$S_B \mathbf{w} = \lambda S_W \mathbf{w} \quad \Rightarrow \quad S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w} \tag{8}$$

This almost looks like an eigen-value equation, if the matrix $S_W^{-1} S_B$ would have been symmetric (in fact, it is called a generalized eigen-problem). However, we can apply the following transformation, using the fact that S_B is symmetric positive definite and can hence be written as $S_B^{\frac{1}{2}} S_B^{\frac{1}{2}}$, where $S_B^{\frac{1}{2}}$ is constructed from its eigenvalue decomposition as $S_B = U\Lambda U^T \rightarrow S_B^{\frac{1}{2}} = U\Lambda^{\frac{1}{2}} U^T$. Defining $\mathbf{v} = S_B^{\frac{1}{2}}\mathbf{w}$ we get,

$$S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}} \mathbf{v} = \lambda \mathbf{v} \tag{9}$$

This problem is a regular eigenvalue problem for a symmetric, positive definite matrix $S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}}$ and for which we can can find solution λ_k and $\mathbf{v}_k$ that would correspond to solutions $\mathbf{w}_k = S_B^{-\frac{1}{2}} \mathbf{v}_k$.

Remains to choose which eigenvalue and eigenvector corresponds to the desired solution. Looking at the original objective J we can guess we want the eigenvector corresponding to the *largest* eigenvalue. This is easily verified by transforming to the dual formulation of the problem, where we use $\mathbf{w}^T S_B \mathbf{w} = 1$ and $\mathbf{w}^T S_W \mathbf{w} = 1/\lambda_k$, resulting in a dual Lagrangian,

$$\mathcal{L}_D = \text{constant} + \frac{1}{2}\lambda_k \tag{10}$$

which we need to *maximize* over λ_k.

2 Kernel Fisher LDA

So how do we kernelize this problem? Unlike SVMs it doesn't seem the dual problem reveal the kernelized problem naturally. But inspired by the SVM case we make the following key assumption,

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i) \tag{11}$$

This is a central recurrent equation that keeps popping up in every kernel machine. It says that although the feature space is very high (or even infinite) dimensional, with a finite number of data-cases the final solution, $\mathbf{w}^*$, can never lie outside of the space spanned by the data-cases. It would not make much sense to do this transformation if the number of data-cases is larger than the number of dimensions, but this is obviously no longer true for kernel-methods. So, we argue that although there are possibly infinite dimensions available a priori, at most N are being occupied by the data, and the solution $\mathbf{w}$ must lie in its span. This is most easily seen by recalling that the solution $\mathbf{W}$ is the solution to some eigenvalue equation, $S_B^{\frac{1}{2}} S_W^{-1} S_B^{\frac{1}{2}} \mathbf{w} = \lambda \mathbf{w}$, where both S_B and S_W (and hence its inverse) lie in the span of the data-cases. Hence, the part $\mathbf{w}^\perp$ that is perpendicular to this span will be projected to zero, and hence the only solution is $\mathbf{w}^\perp = 0$.

In terms of $\boldsymbol{\alpha}$ the objective $J(\alpha)$ becomes,

$$J(\alpha) = \frac{\boldsymbol{\alpha}^T S_B^\Phi \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T S_W^\Phi \boldsymbol{\alpha}} \tag{12}$$

where it is understood that vector notation now applies to a different space, namely the space spanned by the data-vectors, $\mathbb{R}^N$. The scatter matrices in kernel space can expressed in terms of the kernel only as follows (this requires some algebra to verify),

$$S_B^\Phi = \sum_c \left[\boldsymbol{\kappa}_c \boldsymbol{\kappa}_c^T - \boldsymbol{\kappa} \boldsymbol{\kappa}^T \right] \tag{13}$$

$$S_W^\Phi = K^2 - \sum_c N_c \boldsymbol{\kappa}_c \boldsymbol{\kappa}_c^T \tag{14}$$

$$\boldsymbol{\kappa}_c = \frac{1}{N_c} \sum_{i \in c} K_{ij} \tag{15}$$

$$\boldsymbol{\kappa} = \frac{1}{N} \sum_i K_{ij} \tag{16}$$

So, we have managed to express the problem in terms of kernels only which is what we were after. Note that since the objective in terms of $\boldsymbol{\alpha}$ has exactly the same form as that in terms of $\mathbf{w}$, we can solve it by solving the generalized eigenvalue equation. This scales as N^3 which is certainly expensive for many datasets. More efficient optimization schemes solving a slightly different problem and based on efficient quadratic programs exist in the literature.

Projections of new test-points into the solution space can be computed by,

$$\mathbf{w}^T \Phi(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) \tag{17}$$

as usual. In order to classify the test point we still need to divide the space into regions which belong to one class. The easiest possibility is to pick the cluster with smallest Mahalonobis distance: $d(\mathbf{x}, \boldsymbol{\mu}_c^\Phi) = (x^\alpha - \mu_c^\alpha)^2 / (\sigma_c^\alpha)^2$ where μ_c^α and σ_c^α represent the class mean and standard deviation in the 1-d projected space respectively. Alternatively, one could train any classifier in the 1-d subspace.

One very important issue that we did not pay attention to is regularization. Clearly, as it stands the kernel machine will overfit. To regularize we can add a term to the denominator,

$$S_W \rightarrow S_W + \beta \mathbf{I} \tag{18}$$

By adding a diagonal term to this matrix makes sure that very small eigenvalues are bounded away from zero which improves numerical stability in computing the inverse. If we write the Lagrangian formulation where we maximize a constrained quadratic form in α, the extra term appears as a penalty proportional to $||\boldsymbol{\alpha}||^2$ which acts as a weight decay term, favoring smaller values of $\boldsymbol{\alpha}$ over larger ones. Fortunately, the optimization problem has exactly the same form in the regularized case.