Fisher's linear discriminant analysis

1 How to solve the optimisation problem in Fisher's LDA

In Fisher's linear discriminant analysis (LDA), we aim to find a projection direction to maximise the ratio of between-class scatter and within-class scatter. In this way, we can make the instances from the same class closer to each other, while those from different classes separate further apart. Here we focus on binary classification.

Suppose we have a training set of N instances $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$ is the p-dimensional feature vector of the i-th instance and y_i is the corresponding class label. $\boldsymbol{\mu}_1 \in \mathbb{R}^{p\times 1}$ and $\boldsymbol{\mu}_2 \in \mathbb{R}^{p\times 1}$ are the sample means of the two classes.

We solve the following optimisation problem to get the projection direction $\mathbf{w} \in \mathbb{R}^{p\times 1}$:

$$J(\mathbf{w}) = \max_{\mathbf{w}} \ \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}, \tag{1.1}$$

where

$$\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$$

and

$$\mathbf{S}_W = \sum_{c=1}^{2}\sum_{y_i=c}(\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

are the between-class scatter and the within-class scatter, separately. This ratio is also known as Rayleigh quotient.

Note that $J(\mathbf{w})$ is invariant to the change of scale of $\mathbf{w}$: $J(\mathbf{w}) = J(\alpha\mathbf{w})$. Thus we can write the optimisation problem in (1.1) as

$$\begin{aligned} &\max_{\mathbf{w}} \ \ \mathbf{w}^T\mathbf{S}_B\mathbf{w} \\ &\text{s.t.} \ \ \mathbf{w}^T\mathbf{S}_W\mathbf{w} = 1. \end{aligned} \tag{1.2}$$

To solve the above optimisation problem, we use the following Lagrangian function:

$$L = \mathbf{w}^T\mathbf{S}_B\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{S}_W\mathbf{w} - 1). \tag{1.3}$$

Solving the first-order derivative of L with respect to $\mathbf{w}$, we have

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{S}_B\mathbf{w} - 2\lambda\mathbf{S}_W\mathbf{w} = 0 \rightarrow$$

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

The above is a generalised eigenvalue problem. We can solve $\mathbf{w}$ as the generalised eigenvector of $\mathbf{S}_B$ and $\mathbf{S}_W$ with the largest generalised eigenvalue.

If $\mathbf{S}_W$ is invertible, we have

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w} \rightarrow$$

$$\frac{1}{\lambda}\mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w} = \mathbf{w} \rightarrow$$

$$\frac{1}{\lambda}\mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w}] = \mathbf{w}.$$

Note that $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w}$ is a scalar, then we have

$$\frac{c}{\lambda}\mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = \mathbf{w}.$$

Thus

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2).$$

There is a nice explanation of this process in [2]. You can also find a kernel extension of LDA in this paper.

2 Fisher's LDA as a distance metric learning method

Fisher's LDA projects data to a low-dimensional subspace by using $\mathbf{W} \in \mathbb{R}^{p\times r}$. Here the maximum value that r can take is $C-1$ when there are C classes in the data, i.e. $r \leqslant C-1$, because $\mathbf{S}_B$ is of rank $C-1$.

If we project an instance $\mathbf{x} \in \mathbb{R}^{p\times 1}$ in the original feature space to an r-dimensional subspace, then the projection of $\mathbf{x}$ is

$$\mathbf{x}^P = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^{r\times 1}.$$

In the original feature space, the Euclidean distance between two instances $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)}.$$

After projection, the Euclidean distance between two projected instances $\mathbf{x}_1^P$ and $\mathbf{x}_2^P$ is

$$\begin{aligned} d(\mathbf{x}_1^P, \mathbf{x}_2^P) &= \sqrt{(\mathbf{x}_1^P - \mathbf{x}_2^P)^T(\mathbf{x}_1^P - \mathbf{x}_2^P)} \\ &= \sqrt{(\mathbf{W}^T\mathbf{x}_1 - \mathbf{W}^T\mathbf{x}_2)^T(\mathbf{W}^T\mathbf{x}_1 - \mathbf{W}^T\mathbf{x}_2)} \\ &= \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{W}\mathbf{W}^T(\mathbf{x}_1 - \mathbf{x}_2)} \\ &= \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{M}(\mathbf{x}_1 - \mathbf{x}_2)}, \end{aligned}$$

where $\mathbf{M} = \mathbf{W}\mathbf{W}^T \in \mathbb{R}^{p\times p}$.

We can see that the Euclidean distance in the projected subspace between $\mathbf{x}_1^P$ and $\mathbf{x}_2^P$, $\sqrt{(\mathbf{x}_1^P - \mathbf{x}_2^P)^T(\mathbf{x}_1^P - \mathbf{x}_2^P)}$, is equivalent to the Mahalanobis distance in the original feature space between $\mathbf{x}_1$ and $\mathbf{x}_2$, $\sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{M}(\mathbf{x}_1 - \mathbf{x}_2)}$. Thus LDA actually finds a new distance metric to measure the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. With this new distance metric, we can make the between-class scatter large while the within-class scatter small, in order to make the classes as separate as possible.

Learning the matrix $\mathbf{M} \in \mathbb{R}^{p\times p}$ in the Mahalanobis distance

$$d_M(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{M}(\mathbf{x}_1 - \mathbf{x}_2)}$$

is called distance metric learning. Distance metric learning methods aim to automatically learn tailored distance measures/metrics that can capture the underlying structure of the data. A set of similarity/dissimilarity constraints is required to construct a distance metric learning algorithm: the instances from the same class should be similar while the instances from different classes should be dissimilar. Based on the distance metric learned from the training data, we can make the instances from the same class close together while those from different classes farther away from each other. Thus the classification task becomes easier and we can expect better classification performance using the learned distance metrics.

Distance metric learning methods are usually designed for k nearest neighbours (kNN): we use the new Mahalanobis distance learned from data to determine the k nearest neighbours of a test instance. For example, we can use LDA to find $\mathbf{W}$ and use $\mathbf{M} = \mathbf{W}\mathbf{W}^T$ to get the Mahalanobis distance. Then we adopt this Mahalanobis distance in kNN to find the nearest neighbours.

There is a nice paper on distance metric learning, which presents a nice review of distance metric learning algorithms and proposes one of the most successful method called large margin nearest neighbours (LMNN) [1].

References

[1] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[2] Max Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, 3(1), 2005.