COURSEWORK 4

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Maths for Machine Learning

Author:
Alexander Gaskell (CID: 01813313)
Email:
aeg19@imperial.ac.uk

Date: December 1, 2019

I

LDA performs better for classification than PCA/whitened PCA, as seen by the error rate being lower for Figure 3 than for Figure 1 and Figure 2. This is because LDA is a supervised learning method hence the class labels exploited during optimization. PCA/WPCA is unsupervised hence the labels are not used in the problem and PCA is not explicitly defined for classification. LDA is designed to make data samples the same class look more similar while samples from different classes are made to look as dissimilar as possible. This means LDA performs better in dimensionality reduction for classification tasks than PCA/WPCA, as illustrated in this problem.

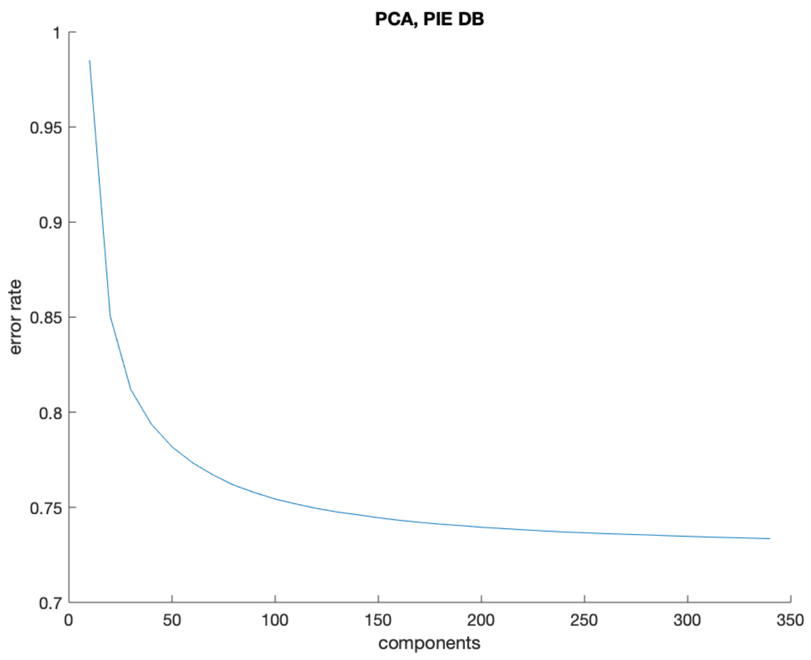

Figure 1: Error rate vs number of components used in model, after extracting features using PCA. Error rate is classification error rate for facial recognition protocol using KNN with features as the components extracted after preforming PCA.

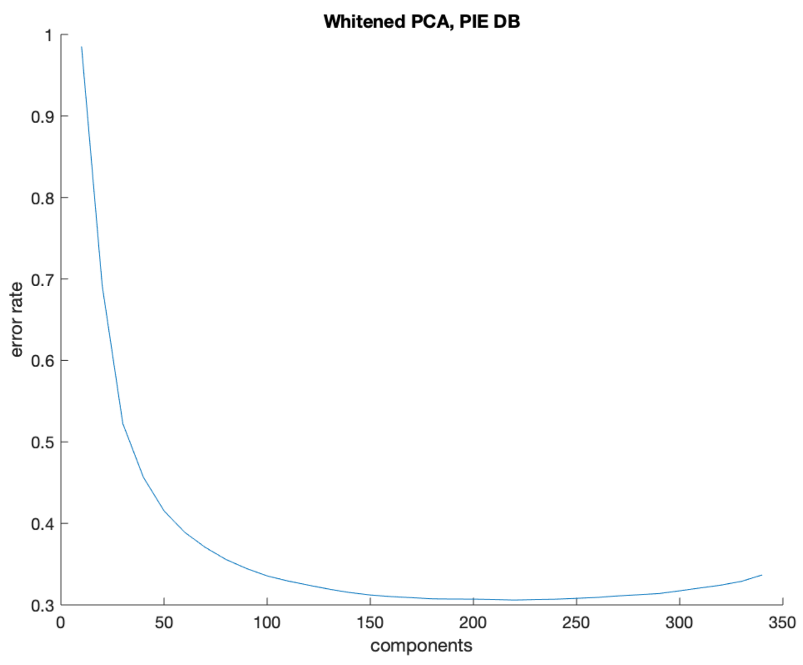

Figure 2: Equivalent to Figure 1 but using whitened PCA to extract features.

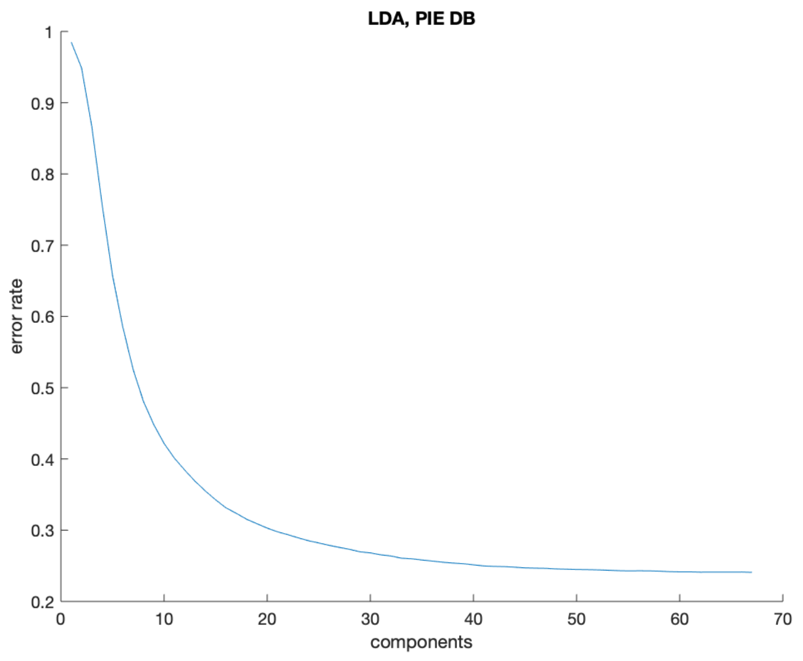

Figure 3: Equivalent to Figure 1 but using LDA to extract features.

II

II.i

The Lagrangian can be formulated as:

$$\mathcal{L}(\mathbf{w}, b, \xi, a_i, r_i) = \frac{1}{2}\mathbf{w}^\top \mathbf{S_t} \mathbf{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n} a_i(y_i(\mathbf{w}^\top \mathbf{x_i} + b) - 1 + \xi) - \sum_{i=1}^{n} r_i \xi_i \tag{1}$$

Taking derivatives and setting equal to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w}\mathbf{S_t} - \sum_{i=1}^{n} a_i y_i \mathbf{x_i} = 0 \iff \mathbf{w_0} = \mathbf{S_t}^{-1}\sum_{i=1}^{n} a_i y_i \mathbf{x}_i \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} a_i y_i = 0 \tag{3}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - a_i - r_i = 0 \tag{4}$$

Substituting these back in we get the dual problem:

$$\max_{\mathbf{a}} \mathcal{L}(\mathbf{a}) = \mathbf{a}^\top \mathbf{1} - \frac{1}{2}\mathbf{a}^\top \mathbf{K}_y \mathbf{a} \tag{5}$$

$$\text{subject to } \mathbf{a}^\top \mathbf{y} = 0, 0 \le a_i \le C \tag{6}$$

Where $\mathbf{K}_y = [y_i y_j \mathbf{x}_i^\top \mathbf{S_t}^{-1} \mathbf{x}_j]$.

Optimal b can be found by first considering the complementary slackness condition:

$$a_i > 0 \implies y_i(\mathbf{w_0}^\top \mathbf{x_i} + b) = 1 \tag{7}$$

This means we can find optimal b, b_0 from any support vector. A more stable solution can be found by taking the average over all support vectors:

$$b_0 = \frac{1}{N_{\mathcal{S}}}\sum_{x_i \in \mathcal{S}} (y_i - \mathbf{w}^\top \mathbf{x_i}) \tag{8}$$

Where $\mathcal{S}$ is the set of support vectors.

We compute ξ_i as follows:

$$\xi_i = \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x_i} + b)), \quad i = 1, \ldots, n \tag{9}$$

Optimal ξ can be computed by plugging w_0, b_0 into this equation.

II.ii

The previous method relies upon S_t being invertible. This may not be the case in small sample size problems, for example, where the number of features exceeds the number of samples. In this case, given we have 2000 samples, if each $\mathbf{x_i}$ had more than 2000 features, we would be faced with this issue. In this case, $S_t = \mathbf{X}\mathbf{X}^\top, \mathbf{X} \in \mathcal{R}^{FxF}$ would be rank deficient as its rank would be at most n and its dimensions would be F, with $F > n$. Here S_t would be singluar so the above method would fail.

A workaround would be to reduce the dimensions of S_t first, it becomes full rank and therefore invertible. One such way would be to perform LDA first. This involves finding a transformation so to maximise the separability of different classes in a low-dimensional latent space while also minimizing the variance of samples from the same class. This reduces the dimensions of $\mathbf{S_t}$ to at most $C - 1$ with C being the number of classes (2 in this case). We could then solve the SVM problem as detailed above to obtain an SVM classifier because the new scatter matrix is invertible.

This approach of first running LDA could be conducted as follows:

1. Perform whitening on $\mathbf{S_t}$ (i.e. compute $\mathbf{U} = \mathbf{X}(\mathbf{I} - \mathbf{M})\mathbf{V_w}\mathbf{\Lambda_w^{-1}}$)
2. Compute projected data $\tilde{\mathbf{X_b}} = \mathbf{U}^\top\mathbf{X}\mathbf{M}$
3. Perform eigenanalysis on $\tilde{\mathbf{X_b}}^\top\tilde{\mathbf{X_b}} = \mathbf{Q}\mathbf{\Lambda_b}\mathbf{Q}^\top$
4. Compute the final transformation $\mathbf{W_0} = \mathbf{U}\mathbf{Q_0}$

Now we could use our SVM classifier using our transformed dataset comprising of $\tilde{\mathbf{X}} \in \mathcal{R}^{nx(C-1)}$ knowing that $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top$ is full rank. In this case $C = 2$ so $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top \in \mathcal{R}$, hence the new scatter matrix would be a scalar, therefore it is invertible.