Imperial College London

COURSEWORK # 4

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Mathematics for Machine Learning

Author:
Alexandre Allani (CID: 1797836)

Date: December 2, 2019

1 Part 1

With the maltab code joint to this report, I obtained the following results:

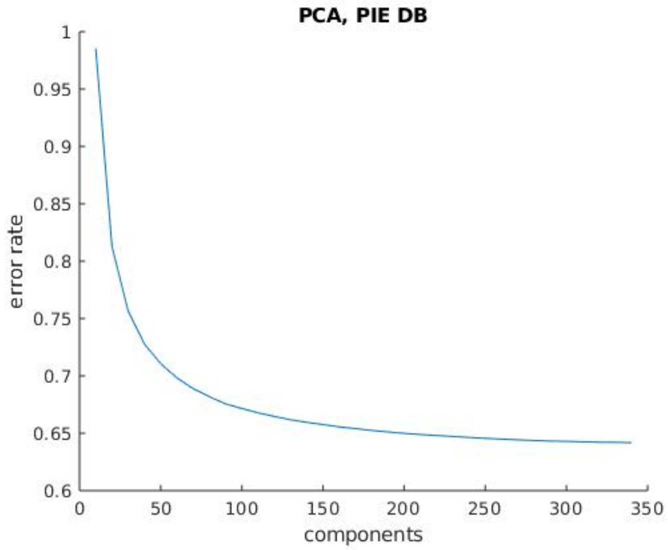

Figure 1: PCA

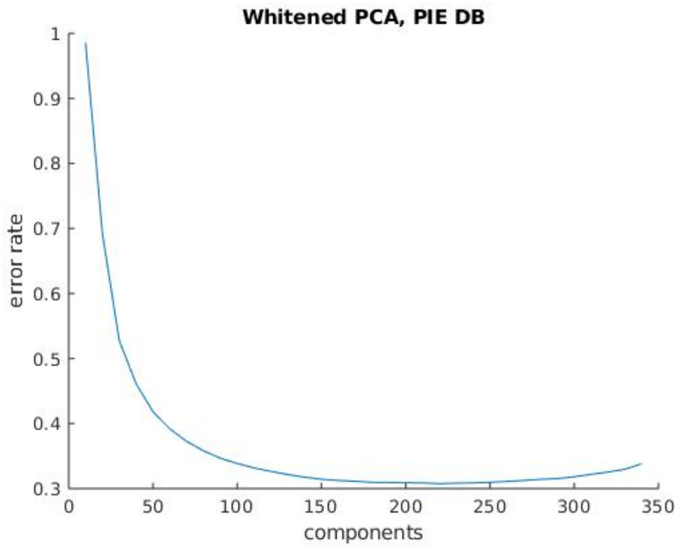

Figure 2: Whitened PCA

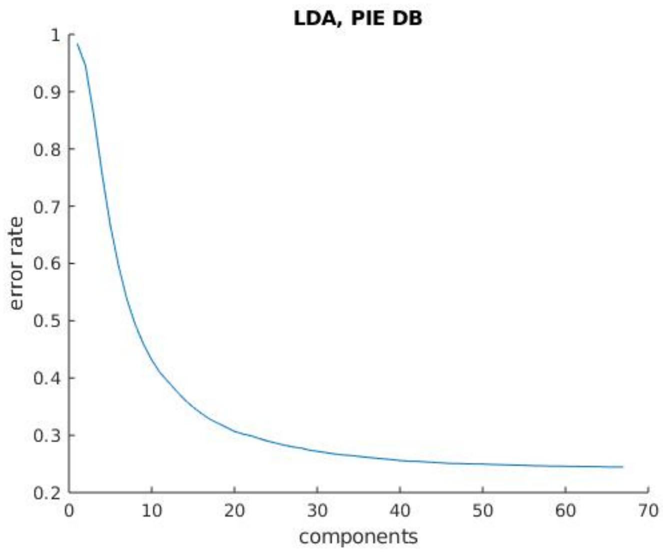

Figure 3: LDA

We can see with the comparison of the figures, that the LDA is better in this case than the PCA/ whitened PCA. This can be explained by the fact that the LDA actually takes into account the number of existing classes giving then, a better projection of the data.
Besides, I find it quite surprising that the error rate of the whitened PCA increases in the end. I think KNN actually overfits in the end, which is usually the case when you take too many components.

2 Part 2

2.1 Question 1

We have the following optimization problem :

$$\begin{array}{ll} \min_{\mathbf{w},b,\xi_i} & \frac{1}{2}\mathbf{w}^T\mathbf{S}_t\mathbf{w} + C\sum_{i=1}^n \xi_i \\ \text{subject to} & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \forall i \in \{1,\dots,n\} \end{array} \tag{1}$$

To solve it, let's express the Lagrangian. We will have here two vectors of Lagrangian multipliers since there are two constraints.
Then:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^\mathbf{T} S_t \mathbf{w} + C\sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^n \beta_i\xi_i \tag{2}$$

$$\begin{aligned} \text{s.t. } & \alpha_i \geq 0, \beta_i \geq 0 \\ & \alpha_i * (y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - (1 - \xi_i)) = 0 \\ & \beta_i * \xi_i = 0 \end{aligned}$$

Let's derive the Lagrangian:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w}^T S_t - \sum_{i=1}^n \alpha_i y_i \boldsymbol{x_i}^T \qquad \Longrightarrow \mathbf{w}^T = \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T\right) S_t^{-1} \tag{3}$$

$$\frac{\partial\mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i \qquad \Longrightarrow \sum_{i=1}^n \alpha_i y_i = 0 \tag{4}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = C - \alpha_i - \beta_i \qquad \Longrightarrow \beta_i = C - \alpha_i \tag{5}$$

This gives us the value of $\mathbf{w}$:

$$\mathbf{w} = S_t^{-1}\left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T\right) \tag{6}$$

By replacing the value of $\mathbf{w}$ inside (3):

$$\begin{aligned} \mathcal{D}(b,\xi) =& \frac{1}{2} * \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} - \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} \\ & + C\sum_{i=1}^n \xi_i - b\sum_{i=1}^n y_i\alpha_i \sum_{i=1}^n \alpha_i - \sum_{i=1}^n \alpha_i\xi_i - \sum_{i=1}^n \beta_i\xi_i \\ =& -\frac{1}{2} * \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} + \sum_{i=1}^n \alpha_i + \sum_{i=1}^n (C - \alpha_i - \beta_i)\xi_i \\ & ((4) \text{ gives } C - \alpha_i - \beta_i = 0) \\ =& -\frac{1}{2} * \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} + \sum_{i=1}^n \alpha_i \end{aligned} \tag{7}$$

This gives us a new optimization problem:

$$\max_{b,\xi}(\mathcal{D}(b,\xi)) = -\frac{1}{2} * \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} + \sum_{i=1}^{n}\alpha_i \tag{8}$$

$$s.t. \sum_{i=1}^{n} y_i\alpha_i = 0$$
$$\alpha_i \geq 0 \text{ Because } \alpha_i \text{ is a Lagrangian multiplier}$$
$$\alpha_i \leq C \text{ Because (5), and } \beta_i \geq 0 \text{ since it is a Lagrangian multiplier}$$

Let's note:

$$K^y_{(i,j)} = [y_i y_j x_i S_t^{-1} x_j] \quad \mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{9}$$

Then the maximization problem can be written as (10) which can be solved in matlab

$$\max_{b,\xi}(\mathcal{D}(b,\xi)) = -\frac{1}{2}\mathbf{a}^T K^y \mathbf{a} + \mathbf{1}^T a \tag{10}$$
$$s.t. \mathbf{y}^T \mathbf{a} = 0 \tag{11}$$
$$0 \leq \alpha_i \leq C$$

Now let's find the optimal b called b^* and ξ_i called ξ_i^*. When $\alpha_i \neq C$, (5) forces $\beta_i \geq 0$. However, because of the Lagrangian optimization problem : $\beta_i\xi_i = 0 \implies \xi_i = 0$
Hence $\alpha_i \neq C \implies \xi_i = 0$
So for all $\alpha_i \geq 0$:

$$y_i(\mathbf{w}^T\mathbf{x_i} + b^*) - 1 = 0 \implies b^* = y_i - \mathbf{w}^T\mathbf{x_i} \tag{12}$$

Then for all $\alpha_i = C$:

$$y_i(\mathbf{w}^T x_i + b^*) = 1 - \xi_i^* \implies \xi_i^* = 1 - y_i(\mathbf{w}^T\mathbf{x_i} + b^*) \tag{13}$$

Results so far
Lagrangian optimization problem :

$$\mathcal{L}(\mathbf{w},b,\xi) = \frac{1}{2}\mathbf{w^T}S_t\mathbf{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i$$

$$\text{s.t. } \alpha_i \geq 0, \beta_i \geq 0$$
$$\alpha_i * (y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - (1 - \xi_i)) = 0$$
$$\beta_i * \xi_i = 0$$

Dual of the Lagrangian optimization problem:

$$\max_{b,\xi}(\mathcal{D}(b,\xi)) = -\frac{1}{2} * \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^T S_t \mathbf{x_j} + \sum_{i=1}^{n} \alpha_i$$
$$s.t. \sum_{i=1}^{n} y_i \alpha_i = 0$$
$$0 \leq \alpha_i \leq C$$

Optimal solutions for $\mathbf{w}, b, \xi_i$:

$$\mathbf{w} = S_t^{-1}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T\right)$$
$$b = y_i - \mathbf{w}^T \mathbf{x_i} \text{ (For an } i \text{ where } \alpha_i \neq C)$$
$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T \mathbf{x_i} + b) & \text{if } \alpha_i = C \\ 0, & \text{otherwise} \end{cases}$$

2.2 Question 3

In the case of S_t singular, the issue would be that S_t is not invertible. And we need it in order to compute the best projection space of the data $\mathbf{w}$.
To make it invertible, I would do a PCA on the matrix, and keep only the non zero eigenvalues. This would give me a new matrix that is this time invertible and that retains most information. However this only works if S_t is a squared matrix. In the case it isn't, it is possible to make the matrix invertible by adding a bit of noise. We would change a bit the information given by S_t but reasonably enough to still have a decent SVM application.