Mathematics for Machine Learning Coursework 4 - Component Analysis and Optimisation

Stanislas Hannebelle

November 2018

1 Part 1

1.1 PCA

Please check my implementation of PCA in the file PCA.m.
Here is the plot of the recognition error versus the number of components kept corresponding to PCA.

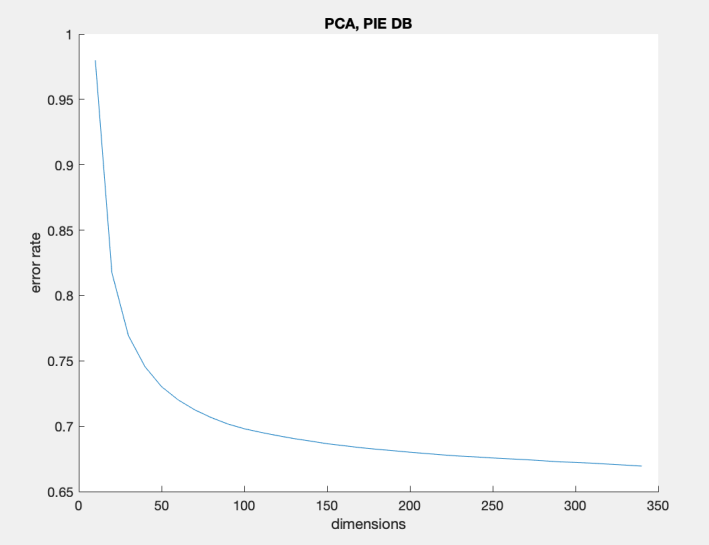

Figure 1: Recognition Error vs Number of Component Kept for PCA

1.2 Whitening PCA

Please check my implementation of Whitening PCA in the file PCA.m.
Here is the plot of the recognition error versus the number of components kept corresponding to Whitening PCA.

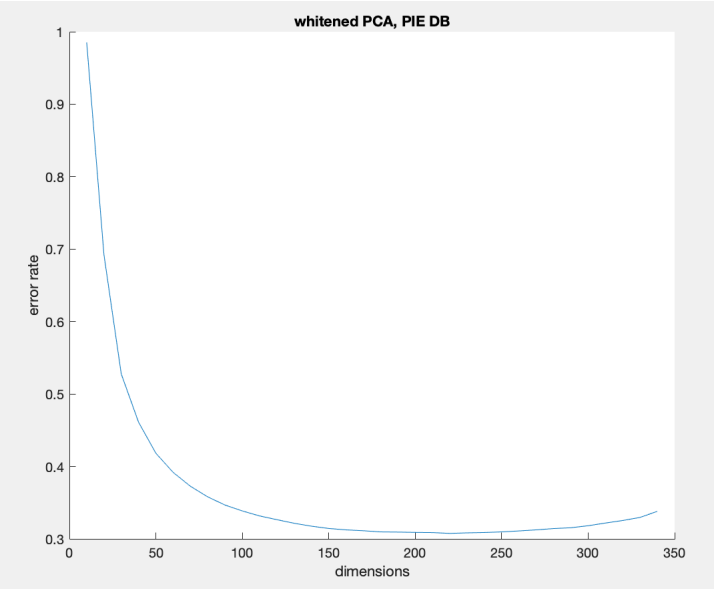

Figure 2: Recognition Error vs Number of Component Kept for Whitening PCA

1.3 LDA

Please check my implementation of LDA in the file LDA.m.
Here is the plot of the recognition error versus the number of components kept corresponding to LDA.

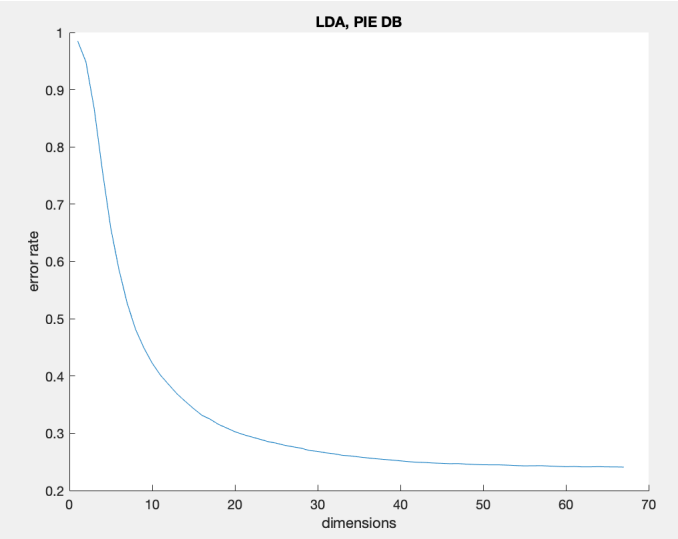

Figure 3: Recognition Error vs Number of Component Kept for LDA

1.4 Comparing Methods

Comparing the three graphs enables us to observe that LDA is probably the best method here.
Indeed, it is the only method getting an error rate of 0.3 for a dimension lower or equal to 20. The error rate even decreases below 0.3 when the number of dimension is reaches 30.
We observe that whitening PCA only becomes efficient when the number of dimension gets close to 100 and that the error rate never goes under 0.3 using this method.
The results of the PCA method are even worst here as it is also efficient when the number of dimension gets close to 100 but the error rate never goes below 0.65.
To conclude, LDA seems to be the best method here and PCA seems to be the worst.

2 Part 2

2.1 Question 1

The minimum enclosing hyper-sphere is defined by the following constrained optimisation problem:

$$\min_{R,a,\xi} R^2 + C\sum_{i=1}^{n}\xi_i$$

subject to $\forall i, (x_i - a)^T(x_i - a) \leq R^2 + \xi_i$

and $\forall i, \xi_i \geq 0$

This problem is equivalent to the following one:

$$\min_{R,a,\xi} R^2 + C\sum_{i=1}^{n}\xi_i$$

subject to $(\mathrm{x}_i - a)^T(x_i - a) - R^2 - \xi_i \leq 0$

and $\forall i, -\xi_i \leq 0$

Then, we can easily formulate the Lagrangian of this optimisation problem:

$$L(R, a, \xi, \lambda, r) = R^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\lambda_i((x_i - a)^T(x_i - a) - R^2 - \xi_i) - \sum_{i=1}^{n} r_i\xi_i$$

with the λ_i corresponding to the $(x_i - a)^T(x_i - a) \leq R^2 + \xi_i$ constraints and the r_i corresponding to the constraints $\xi_i \geq 0$.
Then, the solution of the dual problem is:

$$\max_{\lambda\geq 0, r\geq 0} \min_{R,a,\xi} L(R, a, \xi, \lambda, r)$$

We are optimising a convex function with linear constraints so, the dual solution will equal the primal solution.
To optimise the dual, let's compute the derivatives of L corresponding to R, a and ξ:

$$\frac{\partial L(R, a, \xi, \lambda, r)}{\partial R} = 2R(1 - \sum_{i=1}^{n}\lambda_i)$$

$$\nabla_a L(R, a, \xi, \lambda, r) = 2\sum_{i=1}^{n}\lambda_i(a - x_i)$$

and, $\forall i$,

$$\frac{\partial L(R, a, \xi, \lambda, r)}{\partial \xi_i} = C - \lambda_i - r_i$$

To minimise L, we need these expressions to equal 0. So, we get the following conditions:

$$2R(1 - \sum_{i=1}^{n} \lambda_i) = 0$$

The case of a null radius R is not interesting so, we will consider $R \neq 0$. So,

$$1 - \sum_{i=1}^{n} \lambda_i = 0$$

So,

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{1}$$

Also,

$$2\sum_{i=1}^{n} \lambda_i(a - x_i) = 0$$

So,

$$\sum_{i=1}^{n} \lambda_i a = \sum_{i=1}^{n} \lambda_i x_i$$

Yet,$\sum_{i=1}^{n} \lambda_i = 1$ so,

$$a = \sum_{i=1}^{n} \lambda_i x_i \tag{2}$$

Finally,$\forall i$,

$$C - \lambda_i - r_i = 0 \tag{3}$$

Let's find the new expression of the Lagrangian considering these constraints(1,2,3).

$$L(R, a, \xi, \lambda, r) = R^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \lambda_i((x_i - a)^T(x_i - a) - R^2 - \xi_i) - \sum_{i=1}^{n} r_i\xi_i$$

So,

$$L(R, a, \xi, \lambda, r) = R^2(1 - \sum_{i=1}^{n} \lambda_i) + \sum_{i=1}^{n} \xi_i(C - \lambda_i - r_i) + \sum_{i=1}^{n} \lambda_i((x_i - a)^T(x_i - a))$$

Thanks to equations 1 and 3,

$$L(R, a, \xi, \lambda, r) = \sum_{i=1}^{n} \lambda_i(x_i - a)^T(x_i - a)$$

So,

$$L(R, a, \xi, \lambda, r) = \sum_{i=1}^{n} \lambda_i x_i^T x_i - 2a^T (\sum_{i=1}^{n} \lambda_i x_i) + \sum_{i=1}^{n} \lambda_i a^T a$$

So, thanks to equation 1:

$$L(R, a, \xi, \lambda, r) = \sum_{i=1}^{n} \lambda_i x_i^T x_i - 2a^T (\sum_{i=1}^{n} \lambda_i x_i) + a^T a$$

And thanks to equation 2,

$$L(R, a, \xi, \lambda, r) = \sum_{i=1}^{n} \lambda_i x_i^T x_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^T x_j$$

The dual problem becomes:

$$\max_{\lambda \geq 0, r \geq 0} \sum_{i=1}^{n} \lambda_i x_i^T x_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^T x_j$$

subject to $\forall i, C - \lambda_i - r_i = 0$
and $\sum_{i=1}^{n} \lambda_i = 1$

which is equivalent to the following problem:

$$\max_{0 \leq \lambda \leq C} \sum_{i=1}^{n} \lambda_i x_i^T x_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^T x_j$$

subject to $\sum_{i=1}^{n} \lambda_i = 1$

which is also equivalent to the following problem:

$$\min_{0 \leq \lambda \leq C} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^T x_j - \sum_{i=1}^{n} \lambda_i x_i^T x_i$$

subject to $\sum_{i=1}^{n} \lambda_i = 1$

2.2 Question 2

If we replace all the x_i by $\phi(x_i)$, the optimisation problem of question 1 corresponds to the one of question 2.
Thanks to the same method used in question one, we get that the corresponding dual optimisation problem is:

$$\min_{0\leq\lambda\leq C} \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j\phi(x_i)^T\phi(x_j) - \sum_{i=1}^{n} \lambda_i\phi(x_i)^T\phi(x_i)$$

subject to $\sum_{i=1}^{n} \lambda_i = 1$

2.3 Question 3

In this question, we will use a linear kernel. So, the corresponding optimisation problem to solve is:

$$\min_{0\leq\lambda\leq C} \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j x_i^T x_j - \sum_{i=1}^{n} \lambda_i x_i^T x_i$$

subject to $\sum_{i=1}^{n} \lambda_i = 1$

This problem can be written has:

$$\min_{0\leq\lambda\leq C} \lambda^T H\lambda + F^T\lambda$$

subject to $\sum_{i=1}^{n} \lambda_i = 1$

with $H_{i,j} = x_i^T x_j$ and $F_i = x_i^T x_i$

This is a quadratic problem that we can solve using the Matlab function quadprog which gives us λ.
Knowing λ, we can compute $a = \sum_{i=1}^{n} \lambda_i x_i$.
Then, we need to find the radius of the hyper-sphere.
To do that, we need to remember that the elements λ are the Lagrange multipliers corresponding to the $(x_i - a)^T(x_i - a) \leq R^2 + \xi_i$ constraints.
So according to the KKT conditions, if $\lambda_i > 0$, then $(x_i - a)^T(x_i - a) = R^2 + \xi_i$
Also, the r_i correspond to the Lagrange multipliers of constraints $\xi_i \geq 0$. So, if $r_i > 0$, then $\xi_i = 0$.
Moreover, $\forall i$,C - $\lambda_i - r_i = 0$ according to equation 3. So $r_i > 0$ is equivalent to $\lambda_i < C$. So, we can conclude that if $0 < \lambda_i < C$, $R^2 = (x_i - a)^T(x_i - a)$.

In practice, we consider S to be the set of i such that $0 < \lambda_i < C$ and N_S the number of elements in S.
We compute R by averaging:

$$R = \sqrt{\frac{\sum_{i \in S}(x_i - a)^T(x_i - a)}{N_S}}$$

Thanks to this method, we can compute the optimal enclosing hyper-sphere for our data and for different values of C.

Firstly, for $C = 10$:

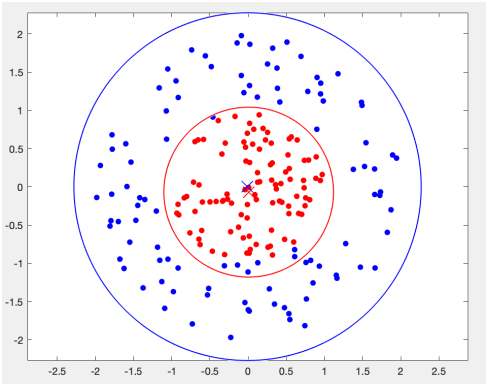

Figure 4: Optimal enclosing hyper-spheres for C=10

We observe that with this value of C, we get satisfactory results. However, the blue circle seems to be a bit large. The radius seems to be a bit to high. This is due to a too high value of C. Let's try a smaller value of C

Then for $C = 0.3$:

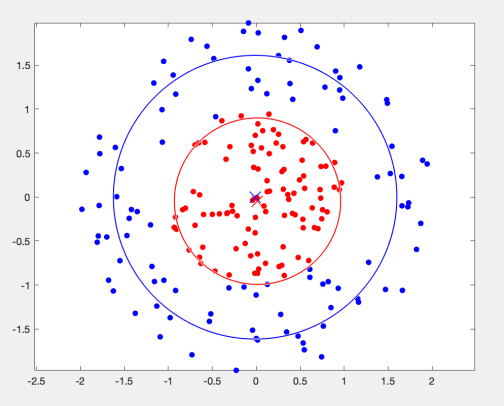

Figure 5: Optimal enclosing hyper-spheres for C=0.3

Here, we observe that, because of a too small value of see, a lot of blue points are clearly outside of the blue circle which is not satisfactory. We need to increase the value of C.

Finally for $C = 0.45$:

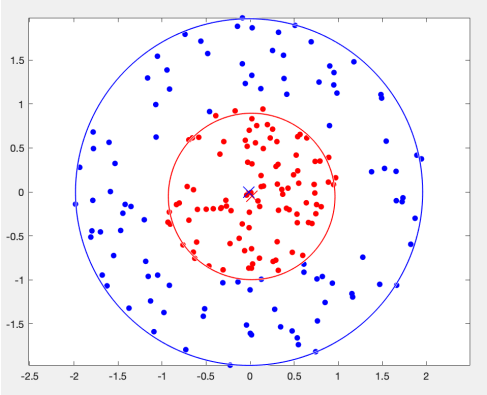

Figure 6: Optimal enclosing hyper-spheres for C=0.45

We observe here that only a few point are outside of their respective circles and that these outside points are still close from the edges of the enclosing circle. This illustrates that $C = 0.45$ is an optimal value for C.

Here, the enclosing edges of both data sets seem to be circles. So, the use of non-linear kernel would not be relevant here.