Assignment 2 - Machine Learning

Theodora Gaiceanu

Task T1

From the assignment description, we know that:

$$\mathbf{K} = [k(x_i, x_j)]_{1\leq i,j\leq 4} = [\phi(x_i)^T\phi(x_j)]_{1\leq i,j\leq 4} = [(x_i, x_i^2)\begin{pmatrix} x_j \\ x_j^2 \end{pmatrix}]_{1\leq i,j\leq 4}$$

By replacing the given values in the expression above, we end up with:

$$\mathbf{K} = [x_ix_j + x_i^2x_j^2]_{1\leq i,j\leq 4} = \begin{pmatrix} (-2)\cdot(-2)+4\cdot 4 & (-2)\cdot(-1)+4\cdot 1 & (-2)\cdot(1)+4\cdot 1 & (-2)\cdot 2+4\cdot 4 \\ (-2)\cdot(-1)+1\cdot 4 & (-1)\cdot(-1)+1\cdot 1 & (-1)\cdot 1+1\cdot 1 & (-1)\cdot 2+1\cdot 4 \\ 1\cdot(-2)+1\cdot 4 & 1\cdot(-1)+1\cdot 1 & 1\cdot 1+1\cdot 1 & 1\cdot 2+1\cdot 4 \\ 2\cdot(-2)+4\cdot 4 & 2\cdot(-1)+4\cdot 1 & 2\cdot 1+4\cdot 1 & 2\cdot 2+4\cdot 4 \end{pmatrix}$$

$$\mathbf{K} = \begin{pmatrix} 20 & 6 & 2 & 12 \\ 6 & 2 & 0 & 2 \\ 2 & 0 & 2 & 6 \\ 12 & 2 & 6 & 20 \end{pmatrix}$$

Task T2

We know that $\alpha = \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$. Using this, we can rewrite the Lagrangian dual problem for the (hard margin) SVM.

$$max_{\alpha_1,\dots,\alpha_4}(\sum_{i=1}^{4}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{4}\alpha_i\alpha_jy_iy_jk(x_i,x_j)) = max_\alpha(4\alpha_i - \frac{1}{2}\sum_{i,j=1}^{4}\alpha^2y_iy_jk(x_i,x_j))$$

$$= max_\alpha(4\alpha - \frac{\alpha^2}{2}\sum_{i,j=1}^{4}y_iy_jk(x_i,x_j))$$

By doing the computations for the given data, it holds that $\sum_{i,j=1}^{4} y_iy_jk(x_i,x_j) = 36$. Therefore, the Lagrangian dual problem becomes:

$$max_\alpha(4\alpha - 36\frac{\alpha^2}{2}) = max_\alpha(4\alpha - 18\alpha^2)$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{4} y_i\alpha_i = 0$.

Now, one can actually compute the optimal α by taking the first derivative of the simplified Lagrangian dual problem expression with respect to α and equalising it to 0.

$$\frac{\partial}{\partial\alpha}(4\alpha - 18\alpha^2) = 4 - 36\alpha$$

$$\frac{\partial}{\partial\alpha}(4 - 36\alpha) = \frac{\partial^2}{\partial\alpha^2}(4\alpha - 18\alpha^2) = -36 < 0$$

As $\frac{\partial^2}{\partial\alpha^2}(4\alpha - 18\alpha^2) < 0$, it means that the expression is concave. We have also the constraint $\alpha \geq 0$.

$$0 = \frac{\partial}{\partial\alpha}(4\alpha - 18\alpha^2)$$

$$0 = 4 - 36\alpha \implies \alpha = \frac{1}{9}$$

Task T3

As we obtained from Task T2 that $\alpha = \frac{1}{9}$, we can rewrite $g(x)$ as:

$$g(x) = \frac{1}{9}\sum_{j=1}^{4} y_j k(x_j, x) + b$$

By using the data from the assignment description, one gets the following expression for $g(x)$:

$$g(x) = \frac{1}{9}[1(-2x + 4x^2) - 1(-x + x^2) - 1(x + x^2) + 1(2x + 4x^2)] + b$$

$$= \frac{1}{9}(-2x + 4x^2 + x - x^2 - x - x^2 + 2x + 4x^2) + b = \frac{1}{9}6x^2 + b$$

$$g(x) = \frac{2x^2}{3} + b$$

In order to find the value for b, one can replace $g(x)$ in the expression $y_s(\sum_{j=1}^{4} \alpha_j y_j k(x_j, x_s) + b) = 1$.

$$1 = y_s(g(x)) = y_s(\frac{2x_s^2}{3} + b)$$

Consequently, we have that:

$$1 = y_s\frac{2x^2}{3} + y_s b \implies b = \frac{1 - y_s\frac{2x^2}{3}}{y_s}$$

But, for computing b, it is actually better to take the average over all support vectors, which means that:

$$b = \frac{1}{N_{SV}}\sum_{x_s \in SV} \frac{1 - y_s\frac{2x^2}{3}}{y_s}$$

Therefore, we have that:

$$b = \frac{1}{4}(\frac{1 - 1\frac{2\cdot4}{3}}{1} + \frac{1 - (-1)\frac{2\cdot1}{3}}{-1} + \frac{1 - (-1)\frac{2\cdot1}{3}}{-1} + \frac{1 - 1\frac{2\cdot4}{3}}{1}) = -\frac{5}{3}$$

Therefore, the simplest possible form for $g(x)$ is $g(x) = \frac{2x^2}{3} - \frac{5}{3}$.

Task T4

One can make a simple script in order to analyse the results. First, one can plot the equation for the classifier $g(x) = \frac{2x^2}{3} - \frac{5}{3}$. This is the orange line from Figure 1. Next, one can plot the points of the second data set (the blue circles).

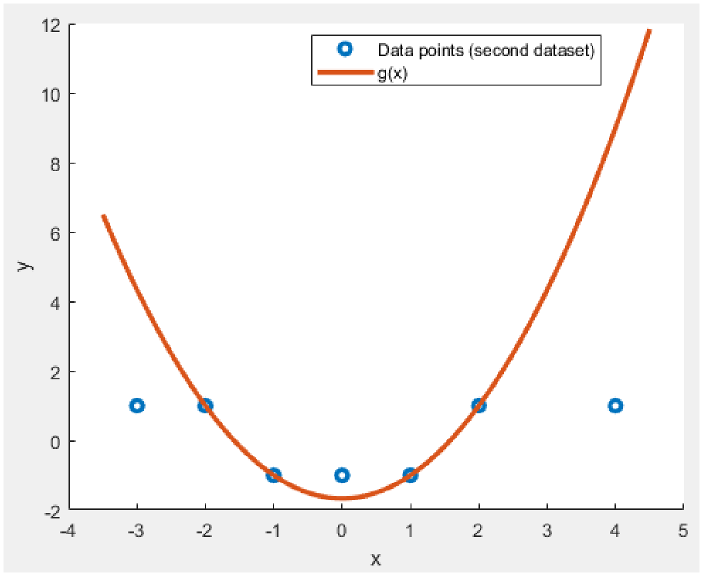


Figure 1: The equation for the classifier $g(x)$ and the new data points

From Figure 1, it can be seen that the points x_2 and x_6 (that have $y_i = 1$) and the points x_3 and x_5 (that have $y_i = -1$) are placed exactly on $g(x)$. Actually, $g(x)$ separates perfectly the two classes, so the same $g(x)$ can be used as a classifier for both datasets.

Task T5

We can write the first constraint as:

$$0 \geq 1 - \xi_i - y_i(w^T x_i + b)$$

$$1 - \xi_i - y_i(w^T x_i + b) \leq 0$$

The second constrained can be rewritten as:

$$-\xi \leq 0$$

Thus, the Lagrangian can be formulated as:

$$L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i(w^T x_i + b)) + \sum_{i=1}^{n}\beta_i(-\xi_i)$$

$$L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(\xi_i + y_i(w^T x_i + b) - 1) - \sum_{i=1}^{n}\beta_i\xi_i$$

Next, one can compute the gradients:

$$\frac{\partial L}{\partial w} = 0 \implies \frac{1}{2}2w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \implies w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \implies -\sum_{i=1}^{n} \alpha_i y_i = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \implies C - \alpha_i - \beta_i = 0 \implies \beta_i = C - \alpha_i$$

After computing the gradients, one can replace the found solutions in the Lagrangian expression:

$$L = \frac{1}{2}(\sum_{i=1}^{n} \alpha_i y_i x_i^T) \sum_{i=1}^{n} \alpha_i y_i x_i + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i(\xi_i + y_i((\sum_{i=1}^{n} \alpha_i y_i x_i^T) x_i + b) - 1) - \sum_{i=1}^{n}(C - \alpha_i)\xi_i$$

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + C\sum_{i=1}^{n} \xi_i - C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \alpha_i y_i((\sum_{i=1}^{n} \alpha_i y_i x_i^T) x_i + b) + \sum_{i=1}^{n} \alpha_i$$

$$L = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i y_i \alpha_j y_j x_j^T x_i - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i$$

But we know that $\sum_{i=1}^{n} \alpha_i y_i = 0$. Therefore, we have that:

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Moreover, we know that $\alpha_i \geq 0$ and $\beta_i \geq 0$. From this and the fact that $\beta_i = C - \alpha_i$, it means that $0 \leq \alpha_i \leq C$.

Therefore, the Lagrangian dual problem is given by:

$$max_{\alpha_1,\dots,\alpha_n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j,$$

with the constraints $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Task T6

From the KKT conditions (complementary slackness), we know that:

$$\lambda_i \alpha_i(\xi_i + y_i(w^T x_i + b) - 1) = 0$$

and

$$\lambda_i \beta_i \xi_i = 0,$$

with $\lambda \geq 0$.

From Task T5, we know that $\beta_i = C - \alpha_i$. This and the fact that $\lambda_i \beta_i \xi_i = 0$ leads to:

$$\lambda_i (C - \alpha_i) \xi_i = 0$$

But we also have that $\xi_i > 0$ for the support vectors. And we also have that $\lambda \geq 0$. Consequently,

$$C - \alpha_i = 0 \implies \alpha_i = C$$

From the first condition and the fact that $\alpha_i \geq 0$, we have that:

$$\xi_i + y_i(w^T x_i + b) - 1 = 0 \implies \xi_i = 1 - y_i(w^T x_i + b)$$

But, as stated before, for the support vectors it holds that $\xi_i > 0$. Consequently, it holds that:

$$1 - y_i(w^T x_i + b) > 0 \implies y_i(w^T x_i + b) < 1$$

In conclusion, support vectors with $\implies y_i(w^T x_i + b) < 1$ have coefficient $\alpha_i = C$.

Task E1

In order to do PCA, first one needs to make sure that the data $\mathbf{X}$ has mean 0. This is actually the necessary condition for doing PCA. So the mean should be subtracted. Then one can do a singular value decomposition on the data ($[U, S, V] = svd(data)$). After doing this, one uses the first two columns (two dimensions; they represent the two left singular vectors having the highest absolute singular values) of the matrix U and does a projection of $\mathbf{X}$ on the first two principal components.

The results of the algorithm can be seen in Figure 2. As it was expected, the data was separated into two clusters (belonging to class 0, respectively class 1). There are some overlapping points, but, in general, the two classes are clearly delimited.

Task E2

First, one needs to complete the function *K_means_clustering*, which implements the K-means clustering algorithm. One needs a function that computes the distance between a single example and the K centroids. The function that does this is $fxdist$ and it takes as input the example X and the K centroids C. It goes through all the centroids and computes the Euclidean distance between each centroid and the example. Next, one needs a function that computes the Euclidean distance between two cluster centroids. This function is called $fcdist$ and it takes as input two centroids, $C1$ and $C2$. The function *step_compute_mean* takes as input the samples X, the centroid C, the label assigned y and the clusters K. It iterates through all the clusters and it updates every centroid by computing the mean of all points assigned to the centroid. Then it computes a measure of the distance which the centroids have moved. After completing all these functions, the algorithm start with assigning the data to a cluster. Then it updates, assigning new cluster. The algorithm stops when the centroids have changed less than a specified threshold, as it is stated in the assignment description.

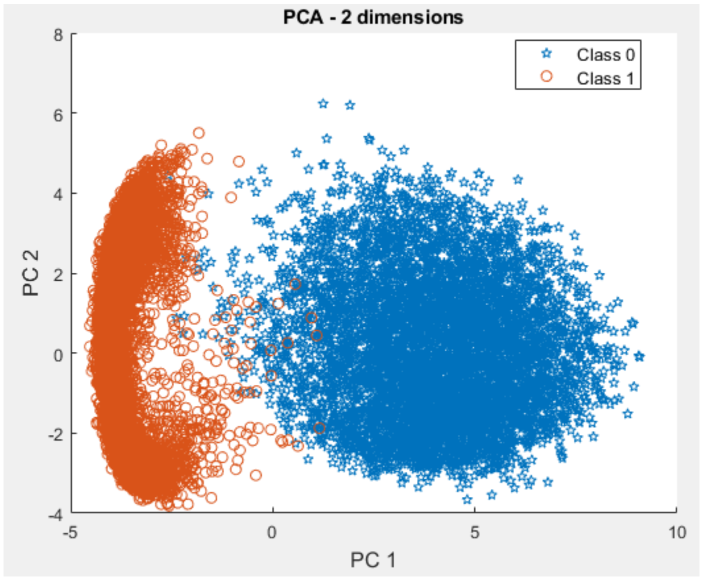
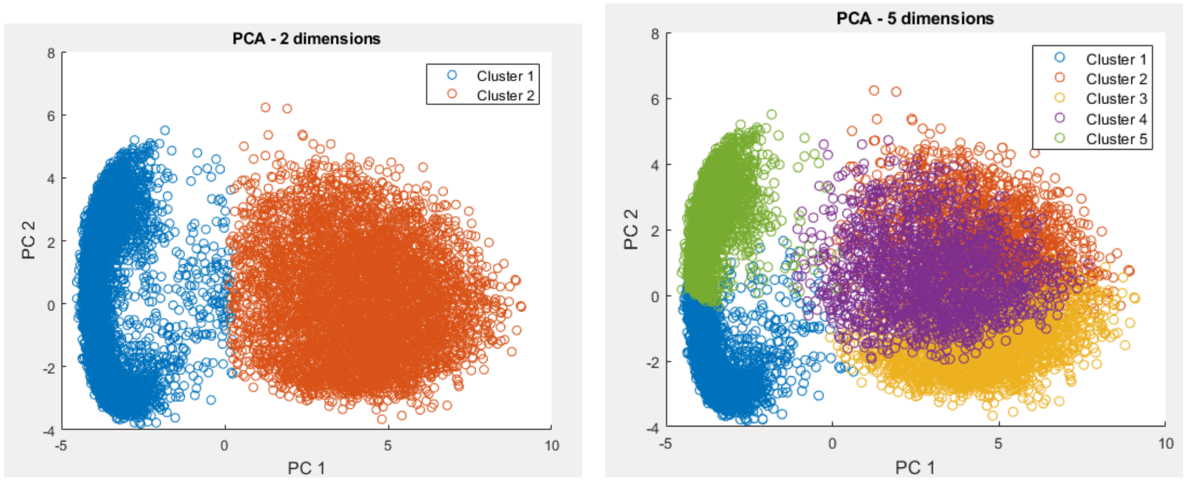


Figure 2: Visualization in 2 dimensions of the training data using linear PCA



(a) 2 clusters

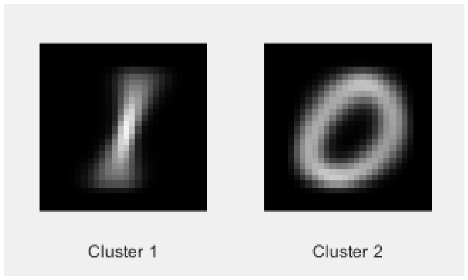
(b) 5 clusters

Figure 3: K-means clustering of the training data using the PCA

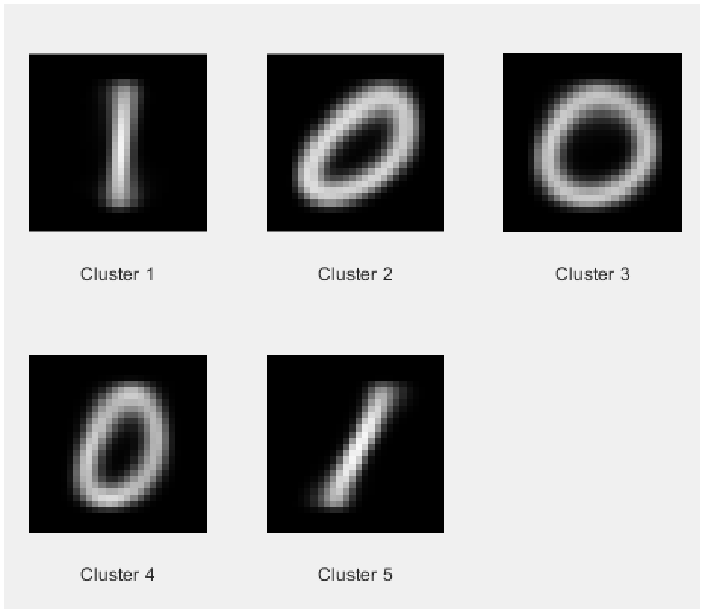
Figure 3a illustrates the results when using $K = 2$ clusters. It looks similar to Figure 2, but the data is delimited even better (there are no overlapping points). Figure 3b presents the results when using $K = 5$ clusters. Here, there can be seen some overlapping clusters. One reason for this behaviour is because we did a PCA dimensionality reduction on the data after the K-means clustering. With PCA for 2 dimensions, we may have samples closer to many clusters, resulting in an overlap.

Task E3

The corresponding centroids for the clusters obtained in Task E2 can be seen in Figure 4. Figure 4a presents the centroids corresponding to the the two clusters from Figure 3a. Figure 4b illustrates the centroids corresponding to the five clusters from Figure 3b. One aspect that can be noticed is that the images of the centroids for $K = 2$ are slightly more blurry than the



(a) $K = 2$ centroids



(b) $K = 5$ centroids

Figure 4: K centroids displayed as images

images of the centroids for $K = 5$.

Task E4

One can evaluate the performances of the K-means algorithm for $K = 2$. The function $K\_means\_classification$ takes as input the assigned clusters y, the training labels, the testing data, the testing labels ans the assigned centroids C. The function first assigns to a sample the label of the closest centroid. After, it assigns to each cluster centroid the label of which it has the most examples in the training data (the MATLAB function *mode* was used for this). Finally, it computes the performance of the K-means algorithm (for $K = 2$) for training and testing.

The results can be seen in Table 1. 12665 samples were used for training and 2115 samples were used for testing. The training misclassification rate is 0.93 %. The testing misclassification rate is even lower (0.56 %). Anyway, both rates are lower than 1%, which means that the K-means classifier for $K = 2$ has very good results.

Task E5

This task is very similar to the previous one. The only difference is that here one wants to evaluate the evolution of the misclassification rate for some different number of clusters. Therefore, one should use the same procedure as in Task E4, but repeat it for each considered K.

The evolution of the miscalssification rate is plotted in Figure 5. The minimum number of clusters considered is 2 and the maximum is 15. As it can be seen, the misclassification rate

Table 1: K-means classification results

Training data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	112	6736	1	112
	2	5811	6	0	6
$N_{\text{train}} = 12665$				Sum misclassified:	118
				Misclassification rate (%):	0.93
Testing data	**Cluster**	**# '0'**	**# '1'**	**Assigned to class**	**# misclassified**
	1	12	1135	1	12
	2	968	0	0	0
$N_{\text{test}} = 2115$				Sum misclassified:	12
				Misclassification rate (%):	0.56

decreases when increasing the number of clusters. Anyway, the highest misclassification rate is around 0.57%, which is still very low. The minimum value for the misclassification rate is 0.148%, when using 13 clusters.

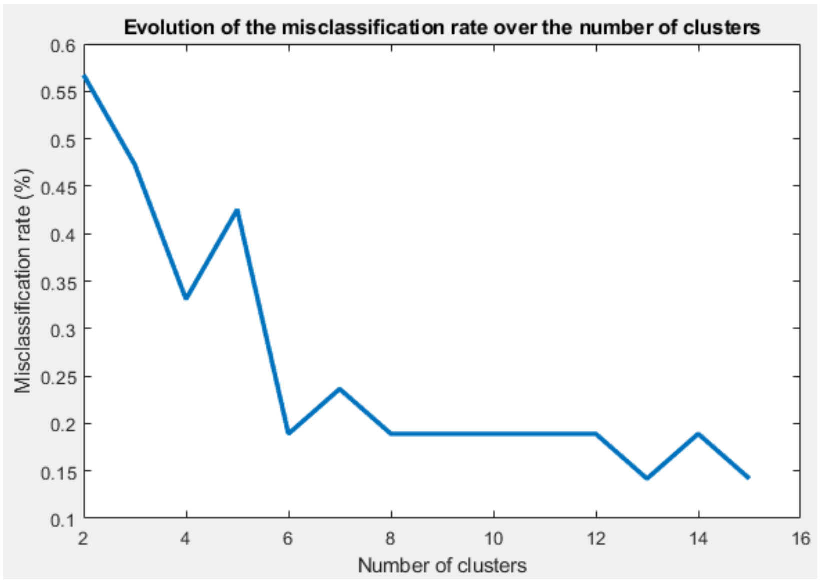


Figure 5: The evolution of the misclassification rate for K = 2, ..., 15

Task E6

Here, one needs to train a SVM model using the *fitcsvm* MATLAB function. Then one predicts the results for both the training and testing data using the *predict* MATLAB function. Then the misclassification rate is analyzed.

The performance of the linear SVM classification can be observed in Table 2. A number of 12665 samples were used for training, and 2115 samples were used for testing. As it can be seen, the algorithm learns very well, the misclassification rate for training is 0%, and the misclassification rate for testing is 0.0945%. Consequently, one can say that the linear SVM classification (supervised) gets better results than the K mean classification (unsupervised).

Table 2: Linear SVM classification results

Training data	Predicted class	True class:	# '0'	# '1'
	'0'		5923	0
	'1'		0	6742
$N_{\text{train}} = 12665$			Sum misclassified:	0
			Misclassification rate (%):	0
Testing data	**Predicted class**	**True class:**	**# '0'**	**# '1'**
	'0'		979	1
	'1'		1	1134
$N_{\text{test}} = 2115$			Sum misclassified:	2
			Misclassification rate (%):	0.0945

Task E7

This task is quite similar to Task E6, but one needs to use a SVM with Gaussian kernel, instead of the linear SVM. One needs to find a suitable value for the β hyperparameter. In order to do so, I have considered values from 1 (the default value) to 15 for the β hyperparameter. For each value of β, one trains the SVM with the Gaussian kernel, computes the class predictions for training and testing and calculates the misclassification rate. The program stop when the misclassification rate for the testing data is 0 or when it has iterated through all the values for β.

As it can be seen from Figure 6, the misclassifcation rate decreases when β increases. Also, from Figure 6, one can observe that the default value for β gives a not so good performance (misclassification rate of approximately 18% for testing). The misclassification rate is significantly reduced when β is approximately 2.5. The best performance is obtained when $\beta = 4.8$. At this point, the misclassification rate for both testing and training data is 0. The detailed results for the SVM with Gaussian kernel classifier when $\beta = 4.8$ can be analyzed in Table 3.

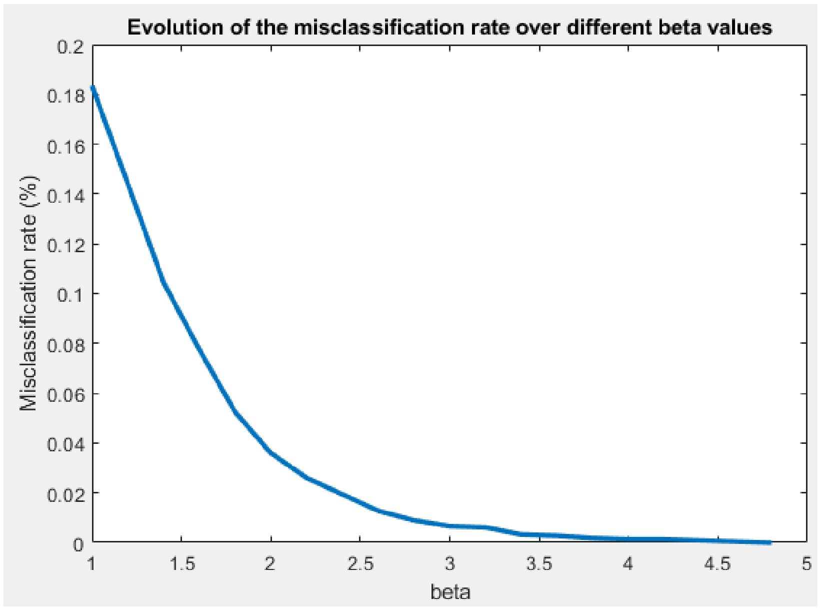


Figure 6: The evolution of the misclassification rate for $\beta = 1, ..., 5$

Table 3: Gaussian kernel SVM classification results

Training data	Predicted class	True class: # '0'	# '1'
	'0'	5923	0
	'1'	0	6742
$N_{\text{train}} = 12665$		Sum misclassified:	0
		Misclassification rate (%):	0
Testing data	Predicted class	True class: # '0'	# '1'
	'0'	980	0
	'1'	0	1135
$N_{\text{test}} = 2115$		Sum misclassified:	0
		Misclassification rate (%):	0

Task E8

It is true that we can achieve a 0 misclassification rate on both train and test data by making a good choice upon the the value of β for the given data. But all the choices we made are based only on the given data. Therefore, it may be a risk of overfitting. On the other hand, the value obtained for the misclassification rate may mean that we could obtain reasonable results with new data. But one should expect the misclassification rate for this new data to be slightly higher.