Rasmus Helander

Assignment 2

Machine Learning, FMAN45
May 3, 2020

Task T1

The task was solved using MATLAB and the result is

$$\mathbf{K} = \begin{bmatrix} 20 & 6 & 2 & 12 \\ 6 & 2 & 0 & 2 \\ 2 & 0 & 2 & 6 \\ 12 & 2 & 6 & 20 \end{bmatrix} \tag{1}$$

Task T2

Since a numerical solution was asked for the optimization was solved using the MATLAB function quadprog. For this purpose the problem was somewhat reformulated to match the format of the function arguments:

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{4} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right) = \max_{\boldsymbol{\alpha}} \left(\mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \right) = \min_{\boldsymbol{\alpha}} \left(-\mathbf{1}^T \boldsymbol{\alpha} + \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \right) \tag{2}$$

where $\mathbf{1}^T = [1, 1, 1, 1]$ and $\mathbf{H} = [y_i y_j k(x_i, x_j)]_{1 \leq i,j \leq 4}$. The constraints were rewritten as follows:

$$\sum_{i=1}^{4} y_i \alpha_i = 0 \iff \mathbf{A}_{eq} \boldsymbol{\alpha} = \mathbf{0} \tag{3}$$

where $\mathbf{A}_{eq} = [\mathbf{y}, \mathbf{0}, \mathbf{0}, \mathbf{0}]^T$ and

$$\alpha_i \geq 0 \Leftrightarrow -\mathbf{I} \boldsymbol{\alpha} \leq \mathbf{0} \tag{4}$$

where $\mathbf{I}$ is the appropriate identity matrix. The equalities and inequalities in equations 3 and 4 should be interpreted elementwise.

If the fact that $\alpha_i = \alpha$ for all i, the problem simplifies to

$$\max_{\alpha} \left(4\alpha - \frac{1}{2} \alpha^2 \mathbf{y}^T \mathbf{K} \mathbf{y} \right) \tag{5}$$

which is easy to solve analytically. No matter the approach, the solution is given by

$$\boldsymbol{\alpha} = \frac{4}{36} [1, 1, 1, 1]^T \tag{6}$$

Task T3

First we try to find b:

$$y_s \sum_i \alpha_i y_i k(x_j, x_s) + b) = 1 \Leftrightarrow b = y_s - \sum_i \alpha_i y_i k(x_i, x_s) = y_s - \left(\sum_i \alpha_i y_i \phi(x_i)^T \right) \phi(x_s) \tag{7}$$

Assignment 2

For easier implementation in MATLAB, this is rewritten as

$$b = y_s - \left(\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)\right)^T \boldsymbol{\Phi}^T \phi(x_s) \tag{8}$$

where $\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)$ is a column vector constisting of the diagonal elements of $\boldsymbol{\alpha}\mathbf{y}^T$ (i.e $[\alpha_1 y_1, ...\alpha_4 y_4]^T$) and $\boldsymbol{\Phi} = [\phi(x_1),\ ...\ \phi(x_4)]$. Equation 8 should hold for any support vector $[x_s, x_s^2]^T$ but as suggested in the lecture it might be better to take the average of all support vectors, in which case we need to calculate

$$b = \text{mean}\left(\mathbf{y}_s^T - \left(\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)\right)^T \boldsymbol{\Phi}^T \boldsymbol{\Phi}_s\right) \tag{9}$$

where $\boldsymbol{\Phi}_s$ is the matrix containing the mappings of all support vectors. Note that $\alpha \neq 0$ only for support vectors, meaning that every data point in feature space is a support vector and

$$b = \text{mean}\left(\mathbf{y}_s^T - \left(\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)\right)^T \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right) = \text{mean}\left(\mathbf{y}_s^T - \left(\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)\right)^T \mathbf{K}\right) \tag{10}$$

For the specified dataset this yields $b = -1.667 = \frac{-5}{3}$. If not all points in the feature space are support vector the result is given by omitting the columns of $\mathbf{K}$ where the index does not correspond to the index of a support vector.

The equation for the classifier is then given by

$$g(x) = \left(\text{diag}(\boldsymbol{\alpha}\mathbf{y}^T)\right)^T \boldsymbol{\Phi}^T \phi(x) + b = \frac{2}{3}x^2 - \frac{5}{3} \tag{11}$$

meaning that samples will be classified as positive if $x^2 > \frac{5}{2}$ which is the line right between the positive and negative samples in feature space, as expected (see figure 1 below).

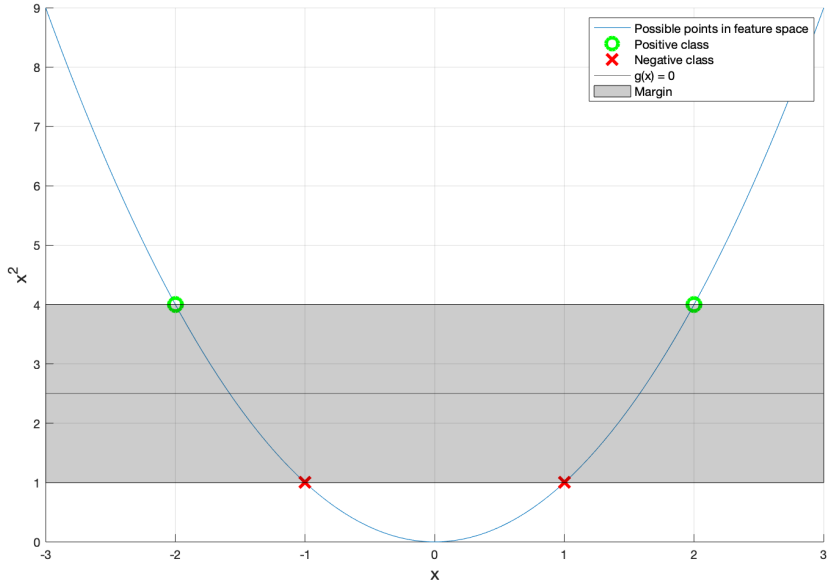

Figure 1: Graphical representation of the SVM

Task T4

Neither of the new data points are located within the margin, meaning that the discriminatory function will be the same as previously, i.e.

$$g(x) = \frac{2}{3}x^2 - \frac{5}{3} \tag{12}$$

This is illustrated in figure 2.

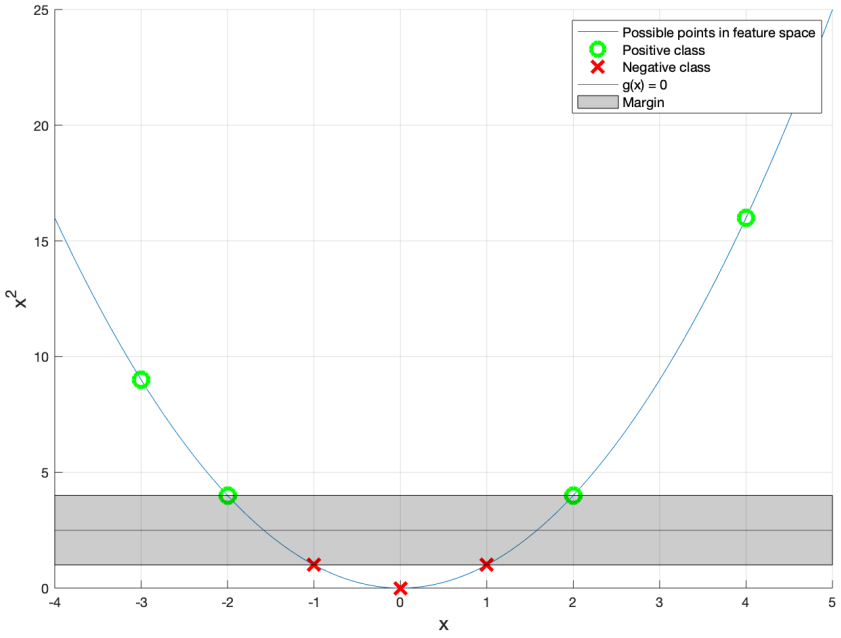

Figure 2: Graphical representation of the SVM, including the new data points.

Task T5

For the problem

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \right) \tag{13}$$

with the constraints specified in the assignment we get the Lagrangian function

$$L(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i - \sum_i \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \right) - \sum_i \beta_i \xi_i \tag{14}$$

and the optimization objective is

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\mathbf{w},\boldsymbol{\xi},b} L(\mathbf{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{15}$$

such that $\alpha_i \geq 0$ and $\beta_i \geq 0$. Minimizing over $\boldsymbol{\xi}$ gives

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \beta_i = 0 \Leftrightarrow \alpha_i = C - \beta_i \tag{16}$$

The fact that $C = \alpha_i - \beta_i$ lets us rewrite the Lagrangian function:

$$L = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) \tag{17}$$

Note that this is the same as for the hard margin problem, meaning that the optimization objective simplifies to the quadratic problem

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \tag{18}$$

subject to $\alpha_i \geq 0$, $\sum_i \alpha_i y_i = 0$. The difference from the hard margin problem is that we also have the constraints $\alpha_i = C - \beta_i$ and $\beta_i \geq 0$. These can be conveniently rewritten as $o \leq \alpha_i \leq \beta_i$ and we have arrived at the Lagrangian dual specified in the task.

Task T6

The complementary slackness states that

$$\beta_i\xi_i = 0 \tag{19}$$

We also know that

$$\xi_i = \max(0,\ 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) = [y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1] = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) > 0 \tag{20}$$

so for these support vectors we must have $\beta_i = 0$ which, combined with equation 16, gives

$$\alpha_i = C \tag{21}$$

□

Task E1

Firstly, the data was centered around the origin in order to do the PCA. This was done by calculating the sample mean for all features, and then updating the data so that

$$\mathbf{x}_{new}^{(j)} = [x_1^{(j)}, x_2^{(j)}, ..., x_d^{(j)}]^T - [\bar{x}_1, \bar{x}_2, ..., \bar{x}_d]^T \tag{22}$$

where d is the dimensionality of the data i.e. the number of features (28x28). Then, the eigenvectors $\mathbf{w}_i$ of the covariance matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ were calculated and sorted by the size of

the respective eigenvector - this was done with the built-in MATLAB function eig. Lastly, the data was projected on the first two eigenvectors, i.e. calculating

$$[\mathbf{w}_1, \mathbf{w}_2]^T \mathbf{X} \tag{23}$$

Figure 3 shows the two first principal components, indicating which class the samples belong to.

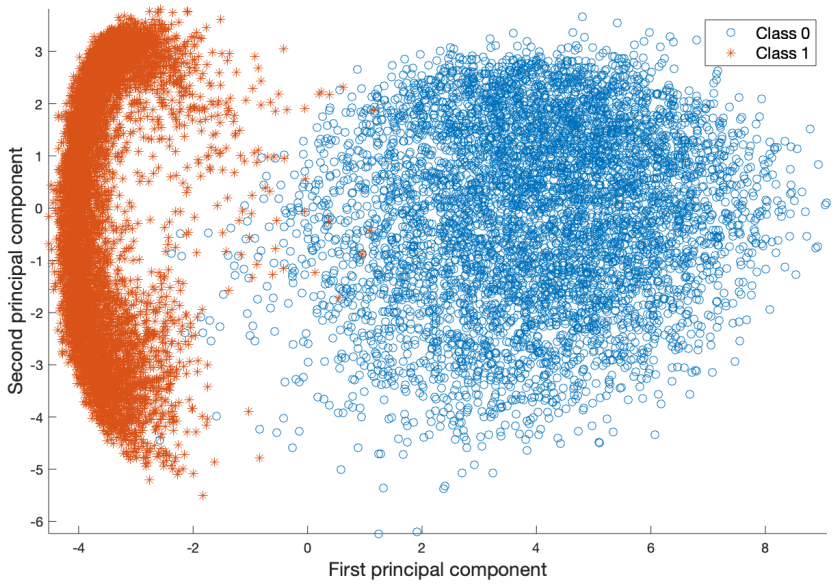

Figure 3: First two principal components of the data

Task E2

The result of the K-means clustering for $K = 2$ and $K = 5$ are shown in figures 4 and 5, respectively - where the two first principal components of the data are plotted for each cluster.

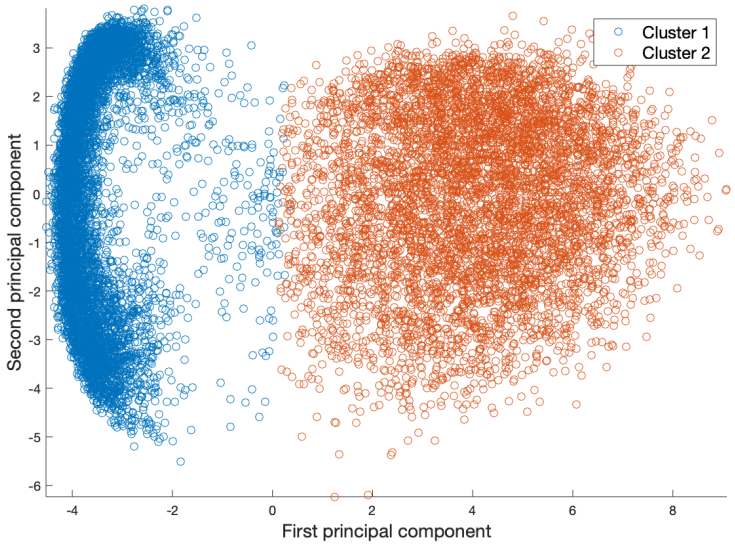

Figure 4: Result of K-mean clustering with 2 clusters, projected on first two principal components

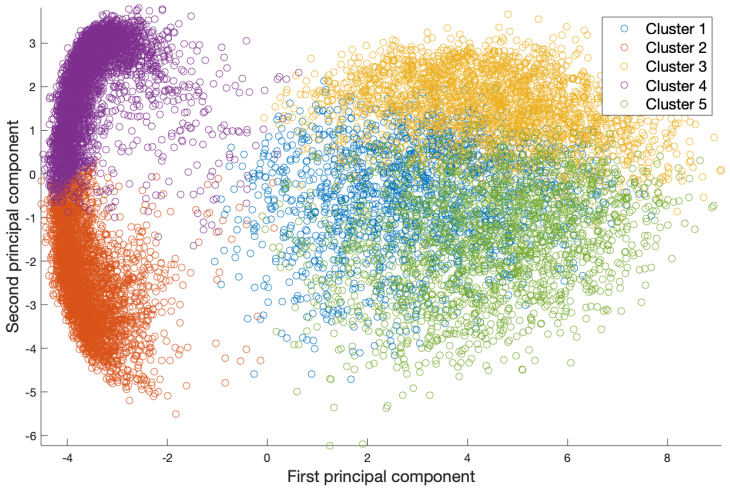

Figure 5: Result of K-mean clustering with 5 clusters, projected on first two principal components

The overlap in the clusters in the figures above can be explained with the fact that only two dimensions are visualized. One can easily imagine a third dimension "in towhich removes the overlap completely. In this data set, there are 782 dimensions which are not visualized that all play a roll in separating the clusters.

Task E3

The figures below show the cluster means of the clusters depicted in figures 4 and 5

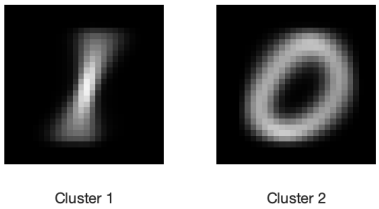

Figure 6: Cluster means for the $K = 2$ training run

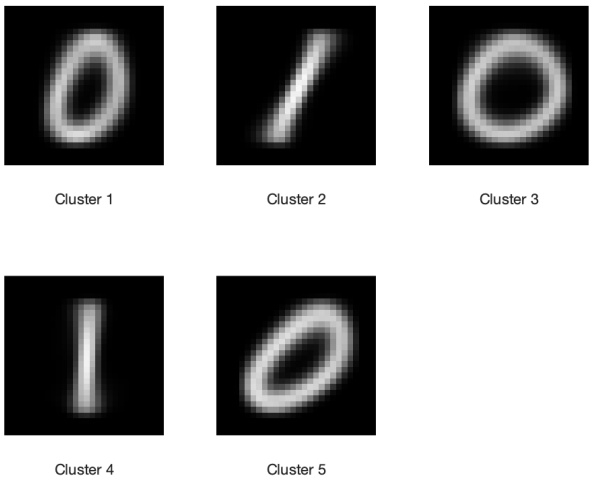

Figure 7: Cluster means for the $K = 5$ training run

Task E4

Table 1: K-means classification results

Training data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	5809	114	0	114
	2	6	6736	1	6
$N_{\text{train}} = 12665$				Sum misclassified:	120
				Misclassification rate (%):	0.95
Testing data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	968	12	0	12
	2	0	1135	1	0
$N_{\text{test}} = 2115$				Sum misclassified:	12
				Misclassification rate (%):	0.57

Task E5

It is possible to decrease the misclassification rate by increasing the number of clusters, see figure below.

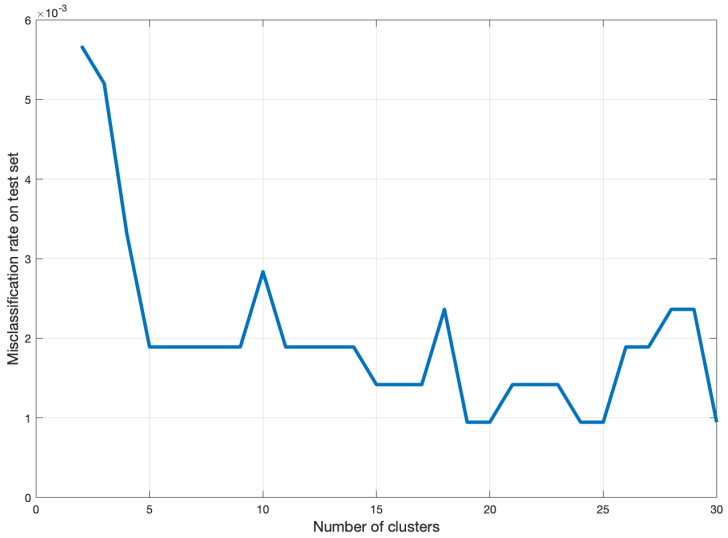

Figure 8: Misclassification rate on test set plotted against number of clusters used in K-means clustering algorithm

Task E6

As can be seen in table 2, the linear SVM classifier performs extremely well on both the training and test set, only misclassifying 2 out of 2115 images in the test set. Normally I would suspect some data leakage from the labels to the feature vectors for such good results, but as the data was provided in the assignment and a builtin MATLAB function was used to train the model and make predictions, I have not looked into it any further.

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 (%):		9.4563e-02

Task E7

Figure 9 visualizes the process of searching for the optimal β parameter. Apparently there are more than one β that decrease the misclassification rate to 0, and the rate stays low for a wide range of β:s. For very large values of β (~ 1000), the classifier once again got significantly worse.

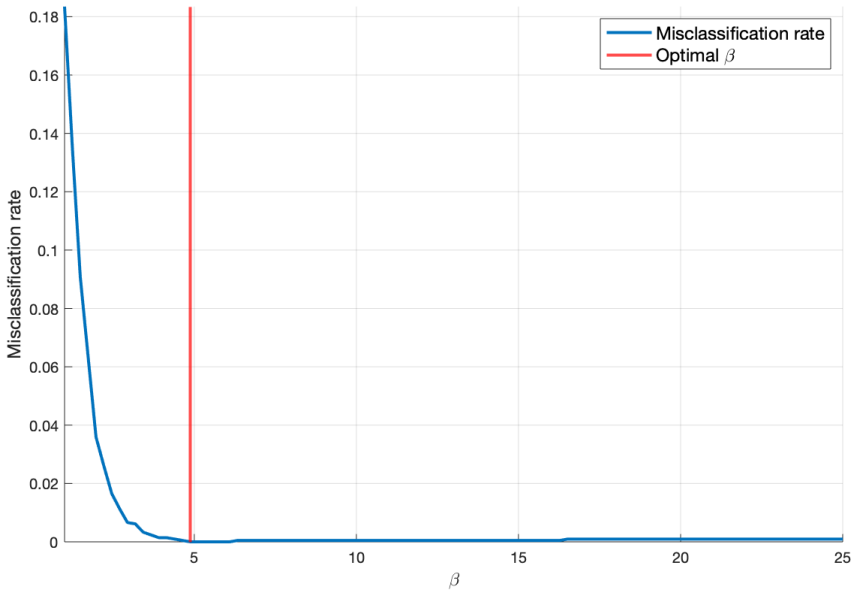

Figure 9: Optimization of the hyperparameter β in the Gaussian kernel

For the optimal parameter indicated in the figure, both classification and training errors were 0, as stated in table 3.

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

As the hyperparameter was tuned to optimize the misclassification rate on the test set, there is a leakage of data from the test set to the model, i.e. there is a risk of that the model

is overfitted to the test data. However, since such a wide range of β:s give good results with respect to misclassification of the test set, I would expect the classifier to work fairly well on unseen data - but the error would likely be slightly larger than zero for the optimal hyperparameter value indicated in figure 9.