FMAN45 Machine Learning - Assignment 2

Erik Waldemarson

1 Solving a nonlinear kernel SVM with hard constraints

This section is about solving a binary classification problem by using Support Vector Machines (SVM) with hard constraints and the kernel trick with a non-linear kernel.

1.1 Exercise 1

When using the feature map

$$\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}, \tag{1}$$

each element in the kernel $\mathbf{K}$ is given by

$$(\mathbf{K})_{i,j} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j) = x_i x_j + x_i^2 x_j^2, \tag{2}$$

which for the data given in instructions becomes

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

1.2 Exercise 2

The Lagrangian dual problem for the (hard margin) SVM is given by:

$$\max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{4} \alpha_i \alpha_j y_i y_j k(x_i, x_j), \tag{3}$$

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

Assuming that $\alpha_i = \alpha$, for every i, I maximize $g(\boldsymbol{\alpha}) = g(\alpha)$ by finding its stationary point (since it's convex bla bla add)

$$\frac{df}{d\alpha} = 4 - \alpha \sum_{i,j=1}^{4} y_i y_j k(x_i, x_j) = 0 \quad => \quad \alpha = \frac{4}{\sum_{i,j=1}^{4} y_i y_j k(x_i, x_j)},$$

and from the data I get $\alpha = 1/9$.

1.3 Exercise 3

After putting the in data, setting $\alpha_j = \alpha = 1/9$ and simplifying I get

$$g(x) = \frac{2x^2 - 5}{3}. \tag{4}$$

1.4 Exercise 4

As shown in Figure 1, the new data falls within the same decision boundary as determined by (4). Thus the solution is the same as before, $g(x) = (2x^2 - 5)/3$.

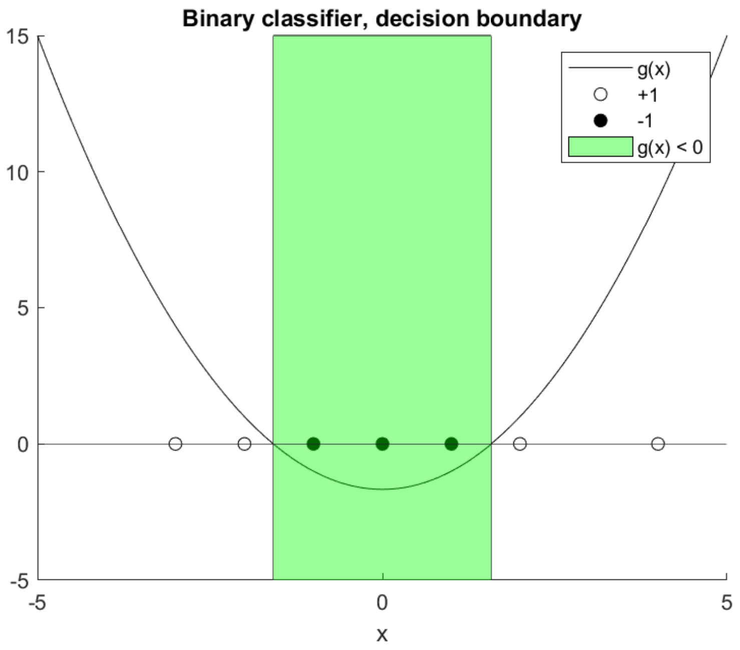

Figure 1: Binary classifier with $y_i \in [-1, +1]$ with decision boundary given by $g(x) = (2x^2 - 5)/3$.

2 The Lagrangian dual of the soft margin SVM

The following section is about deriving the Lagrangian dual problem for the soft margin SVM.

2.1 Exercise 5

The Lagrangian for the soft margin SVM is given by

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda}) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(w^T\mathbf{x_i} + b) - 1 + \xi_i) - \sum_{i=1}^{n}\lambda_i\xi_i, \tag{5}$$

where $\alpha_i \geq 0$ and $\lambda_i \geq 0$. As usual I need to minimize L over $\mathbf{w}$, b and $\boldsymbol{\xi}$ to get the Lagrange dual function $g(\boldsymbol{\alpha}, \boldsymbol{\lambda})$. The dual problem formulation is then maximizing g over $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$. I attempt to do this by finding stationary points:

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{n}\alpha_i y_i x_j^{(i)} = 0, \quad \forall j \quad => \qquad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}, \tag{6}$$

(here $x_j^{(i)}$ means element x_j of the vector $\mathbf{x_i}$).

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0, \tag{7}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0, \quad \forall i \quad => \qquad \alpha_i = C - \lambda_i. \tag{8}$$

It's clear that every $\alpha_i \leq C$ since $\lambda_i \geq 0$. In addition to this I maximize g over $\boldsymbol{\lambda}$:

$$\frac{\partial L}{\partial \lambda_i} = -\xi_i = 0 \quad \forall i \quad => \qquad \boldsymbol{\xi} = \mathbf{0}. \tag{9}$$

Putting all of this together (remember to use different indices in the sum when putting in expression for $\mathbf{w}$), I arrive at the Lagrangian dual problem over $\boldsymbol{\alpha}$:

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_n} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad \forall i \\ & \sum_{i=1}^{n}\alpha_i y_i, \quad \forall i. \end{aligned} \tag{10}$$

□

2.2 Exercise 6

Every optimal α_i^* and λ_i^* must abide the following condition:

$$\text{KKT Complementary Slackness} = \begin{cases} \alpha_i^*(y_i(w^T\mathbf{x_i} + b) - 1 + \xi_i) = 0, \\ \lambda_i^*\xi_i = 0, \quad \forall i. \end{cases} \tag{11}$$

Now by definition, every support vector fulfills $y_i(w^T\mathbf{x_i} + b) - 1 + \xi_i = 0$, so for those $\alpha_i^* \neq 0$. Furthermore, for every support vector with $y_i(w^T\mathbf{x_i} + b) - 1 < 0$ must have $\xi_i > 0$ which means $\lambda_i^* = C - \alpha_i^* = 0 => \alpha_i^* = C$ for every such support vector. □

3 Dimensionality reduction on MNIST using PCA

Principal component analysis (PCA) is a dimensionality-reduction method which involves finding the components with the largest explained variance for the data X. In simpler terms this means finding some orthogonal basis that the data can be projected on while minimizing the amount of information lost. In practice this is done by finding the eigenvalues of the covariance matrix $C = \frac{1}{N}X^TX$.

In this section I will perform (PCA) on a data set of MNIST-images.

3.1 Exercise 7

One can perform PCA with the singular value decomposition on $X = USV^T$ and then finding the projection $\pi(X)$ of X on the d number of left singular vectors (the columns of U):

$$\pi(X) = U_{\text{reduced}}^T X. \tag{12}$$

This can be shown to be equivalent to finding the eigenvectors of C. An important detail is that each row in X needs to be zero-mean.

PCA was performed on the test-data with $d = 2$ components which has been plotted in Figure 2.

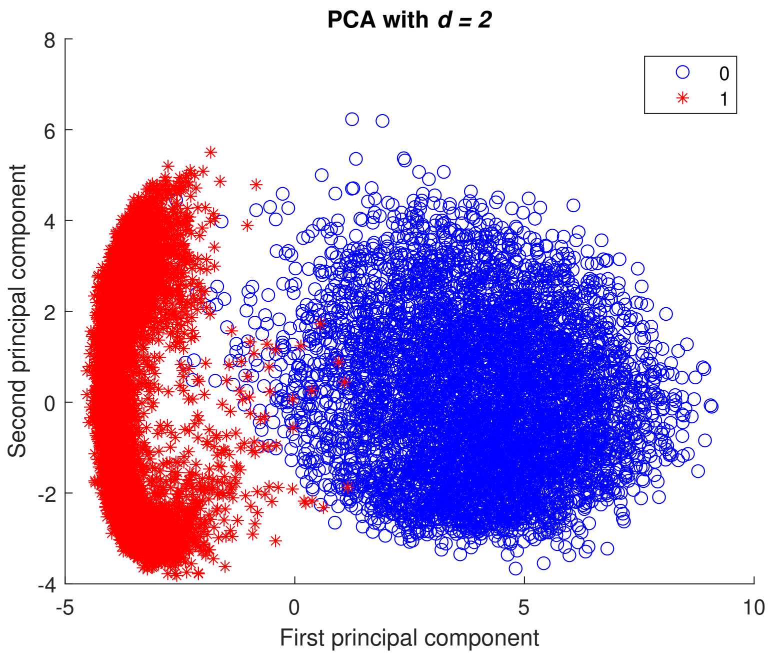

Figure 2: PCA of the MNIST_01 test-data with $d = 2$ components.

4 Clustering of unsupervised data using K-means

In this section I will use clustering and classification of unsupervised data using K-means.

4.1 Exercise 8

I implemented the K_means_clustering function in MATLAB and used it for $K = 2$ and $K = 5$. I then projected it on the already trained PCA matrix to visualize it in 2D which has been plotted in Figure 3 and 4.

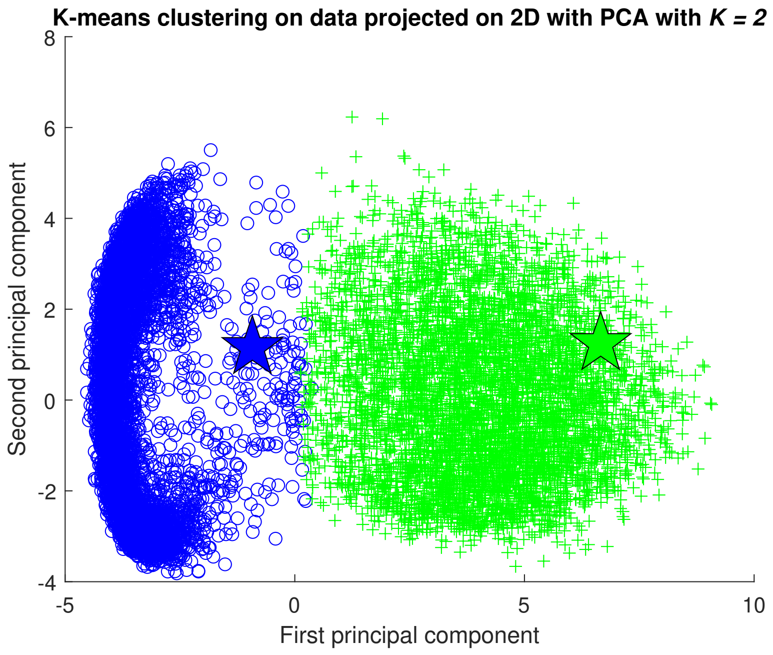

Figure 3: Unsupervised K-means clustering on data projected on 2D with PCA with $K = 2$. The centroids for each class has been marked with a star.

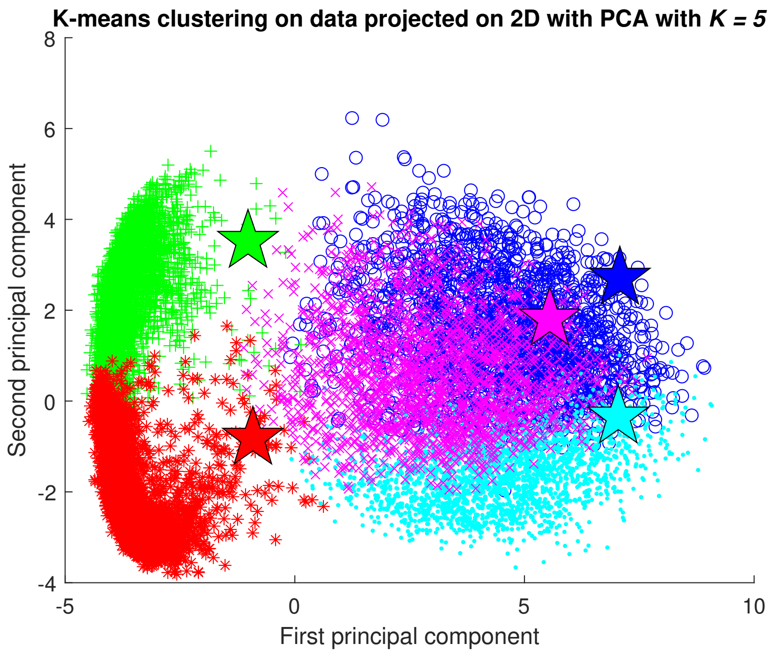

Figure 4: Unsupervised K-means clustering on data projected on 2D with PCA with $K = 5$. The centroids for each class has been marked with a star.

4.2 Exercise 9

The centroids displayed as images have been plotted in Figure 5 and 6 below. As one can see each image is a variation of the number 0 or 1 which is to be expected.

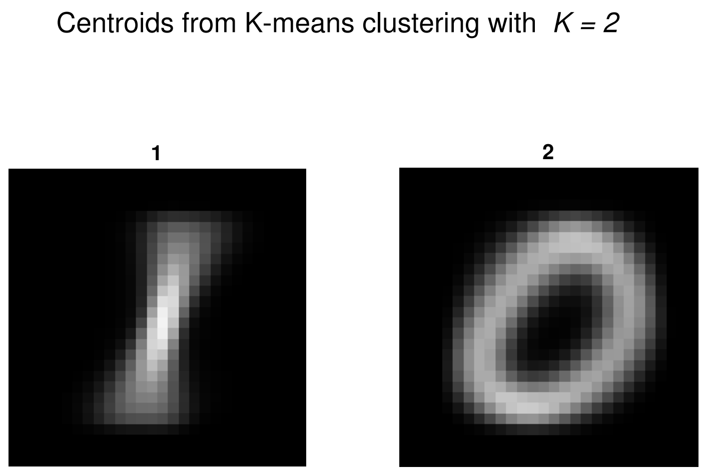

Figure 5: Centroids for binary classifcation using K-means ($K = 2$) displayed as images.

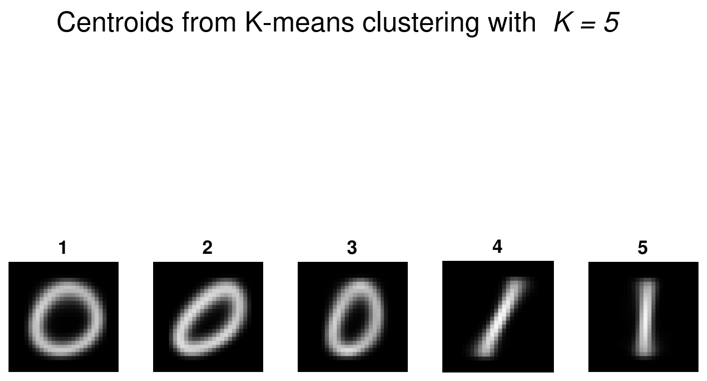

Figure 6: Centroids for binary classifcation using K-means ($K = 5$) displayed as images.

4.3 Exercise 10

Table 1: K-means classification results

Training data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	5811	6	0	6
	2	112	6736	1	112
$N_{\text{train}} = 12665$				Sum misclassified:	118
				Misclassification rate (%):	0.93
Testing data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	968	0	0	0
	2	12	1135	1	12
$N_{\text{test}} = 2115$				Sum misclassified:	12
				Misclassification rate (%):	0.57

4.4 Exercise 11

I got a slightly lower result with $K = 3$ but not by a lot.

Table 2: K-means classification results

Testing data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	513	0	0	0
	2	456	0	0	0
	3	11	1135	1	11
$N_{\text{test}} = 2115$				Sum misclassified:	11
				Misclassification rate (%):	0.52

5 Classification of MNIST digits using SVM

In this section I will use the soft margin support vector machine (SVM) using different kernels for binary classification of MNIST images of numbers.

5.1 Exercise 12

Table 3: 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 (%):	0.095

5.2 Exercise 13

I got a pretty good result on the test data when setting $\beta = e \approx 2.7183$.

Table 4: 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	23
	'1'	0	1112
$N_{\text{test}} = 2115$		Sum misclassified:	23
		Misclassification rate (%):	1.1

5.3 Exercise 14

No because we are tuning the parameter β using the test data which will introduce a bias in the result and will no longer represent the "true" misclassification rate. Normally what you would do in this situation is that you would have three datasets: training, validation (for tuning β) and test.