# FMAN45 Machine Learning

Assignment 2

**(Van Duy Dang - va7200da-s)**

May 2, 2023

## 1 Solving a nonlinear kernel SVM with hard constraints.

**Task 1:**

| **i** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| $x_i$ | -2 | -1 | +1 | 2 |
| $y_i$ | +1 | -1 | -1 | +1 |

Table 1: Dataset with classes '+1' and '-1'.

Using the dataset from the table above, we compute the kernel matrix K

$$K = \phi(x_i)^T \phi(y_j)$$

$$= x_i x_j + (x_i x_j)^2 = \begin{bmatrix} 20 & 6 & 2 & 12 \\ 6 & 2 & 0 & 2 \\ 2 & 0 & 2 & 6 \\ 12 & 2 & 6 & 20 \end{bmatrix}$$

**Task 2:**

Since the solution satisfies $\alpha = \alpha 1 = \alpha 2 = \alpha 3 = \alpha 4$, the maximization for $\alpha$ is given by:

$$\underset{\alpha}{\text{maximize}} = 4\alpha - \frac{\alpha^2}{2} \sum_{i,j=1}^{4} y_i y_j k(x_i, x_j)$$

$$= 4\alpha - \frac{\alpha^2}{2}(20 - 6 - 2 + 12 - 6 + 2 - 2 - 2 + 2 - 6 + 12 - 2 - 6 + 20) = 4\alpha - 18\alpha^2$$

We solve this problem by differentiating with respect to $\alpha$ and solve the equation

$$\frac{d}{d\alpha}(4\alpha - 18\alpha^2) = 0$$
$$\iff 4 - 36\alpha = 0$$
$$\iff \alpha = \frac{1}{9}$$

**Task 3:**

Using the equation for the classifier from the requirement and $\alpha$ from the previous task, we have

$$g(x) = \sum_{j=1}^{4} \alpha_i y_i k(x_j, x) + b$$

$$= \frac{1}{9}(\begin{bmatrix} -2 & 4 \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix} - \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix} + \begin{bmatrix} 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix})$$

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

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

From the requirement, we have

$$y_s(\sum_{j=1}^{4} \alpha_i y_i k(x_j, x_s) + b) = 1$$

We choose $x_s = x_1 = -2, y_s = y_1 = 1$ hence

$$1(\frac{2}{3}(-2)^2 + b) = 1$$
$$\iff b = -\frac{5}{3}$$

Applying this result to (1), the simplest form is given by

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

**Task 4:**

| **i** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| $x_i$ | -3 | -2 | -1 | 0 | +1 | 2 | 4 |
| $y_i$ | +1 | +1 | -1 | -1 | -1 | +1 | +1 |

Table 2: New dataset with classes '+1' and '-1'.

We notice that the data table in task 1 is the subset of the new data table in task 4. All the data from the old table is the same as the new one. In addition, $x_1, x_4$ and $x_7$ are separated into classes '+1' and '-1' using (2). Therefore we can use the same classifier g(x) as the previous task.

## 2 The Lagrangian dual of the soft margin SVM

**Task 5:**

From the requirement, the primal formulation of the linear soft margin classifier is given by

$$\underset{w,b,\xi}{\text{minimize}} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \tag{3}$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

To show the Lagrangian dual problem, we need to simplify

$$\underset{\alpha 1,\ldots,\alpha n}{\text{maximize}}\ \underset{w,b,\xi}{\text{minimize}}\ \mathcal{L}(w,b,\xi)=\frac{1}{2}||w||^2+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)+\xi_i-1)-\sum_{i=1}^{n}\lambda_i\xi_i \quad (4)$$

We minimize $\mathcal{L}$ with respect $w, b, \xi$ by differentiating as below

$$\frac{d\mathcal{L}}{dw}=0 \iff w=\sum_{i=1}^{n}\alpha_i y_i x_i \quad (5)$$

$$\frac{d\mathcal{L}}{db}=0 \iff \sum_{i=1}^{n}\alpha_i y_i=0 \quad (6)$$

$$\frac{d\mathcal{L}}{d\xi}=0 \iff \lambda_i=C-\alpha_i \quad (7)$$

Using the above three equations, we can rewrite (4) as

$$\underset{\alpha 1,\ldots,\alpha n}{\text{maximize}}\ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j+C\sum_{i=1}^{n}\xi_i-(\sum_{i=1}^{n}\alpha_i y_i(\sum_{j=1}^{n}\alpha_j y_j x_j)^T x_i+\alpha_i y_i b+\alpha_i\xi_i-\alpha_i)-\sum_{i=1}^{n}(C-\alpha_i)\xi_i$$

$$\iff \underset{\alpha 1,\ldots,\alpha n}{\text{maximize}}\ \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j \quad (8)$$

Using $\alpha_i, \lambda_i \geq 0$ and $\alpha_i = C - \lambda_i$, the constraint can be written as $0 \leq \alpha_i \leq C$
In other words, we can show the Lagrangian dual problem

$$\underset{\alpha 1,\ldots,\alpha n}{\text{maximize}}\ \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$
$$\text{subject to } 0\leq\alpha_i\leq C$$
$$\sum_{i=1}^{n}\alpha_i y_i=0$$

**Task 6:**

Using the KKT conditions, we have:

$$\alpha_i(y_i(w^Tx_i+b)+\xi_i-1)=0 \quad (9)$$

$$\lambda_i\xi_i=0 \quad (10)$$

From (7), we can rewrite (10) as

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

With $\alpha_i = C$ then the above condition is satisfied. In addition, from (9) we have:

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

With $\xi_i \geq 0$ then

$$y_i(w^T x_i + b) < 1$$

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

## 3 Dimensionality reduction on MNIST using PCA

### Task E1:

The train data is normalized so that the mean of each row is 0. Then we calculate the left singular vectors using the svd function. To visualize the whole training data in d = 2 dimension, we calculate the projection of the normalized data onto the two first left singular vectors.

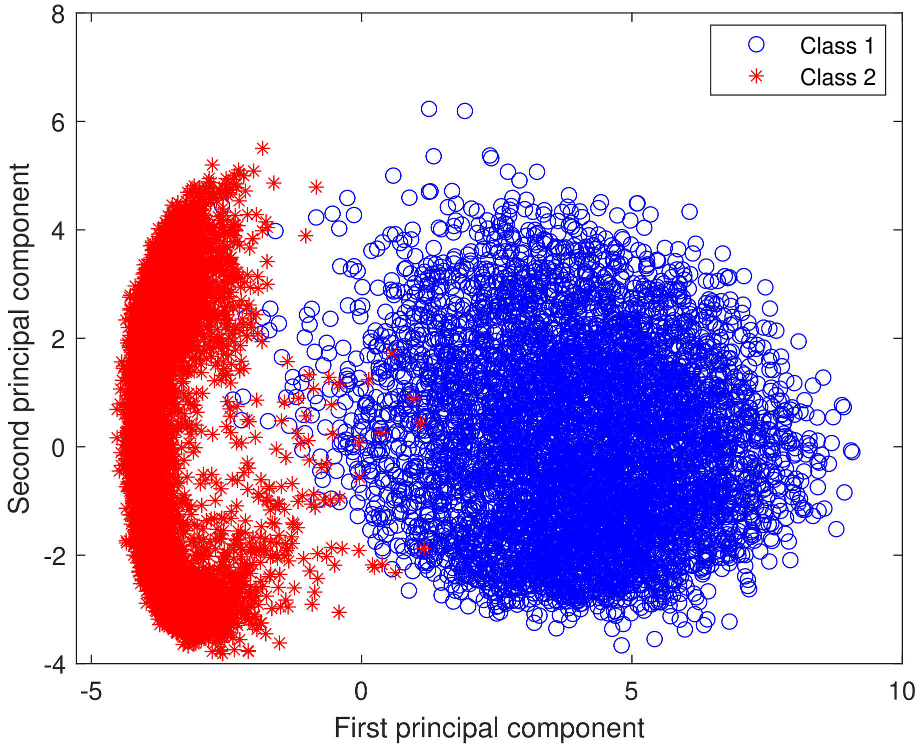

Figure 1: The dimensionality reduction of MNIST to dimension 2

### Task E2:

We use K-means clustering to classify the data with K = 2 and K = 5. With K = 2, the classes are classified quite well. Although some points are misclassified.

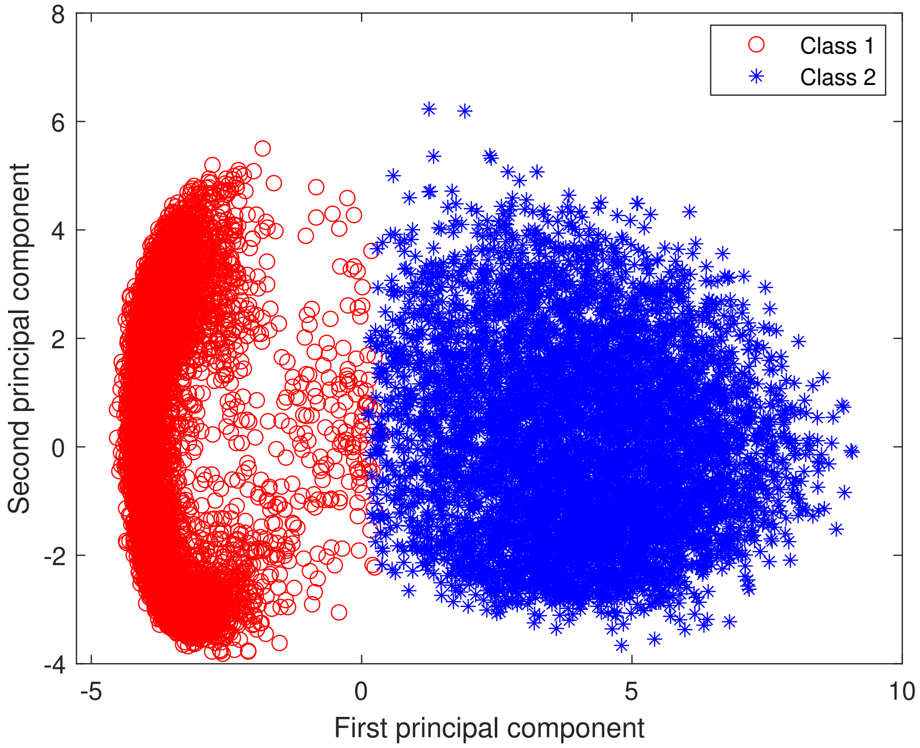

Figure 2: K-means clustering with K = 2

The clusters seem to overlap with K = 5. The possible reason is that we use K-means clustering before PCA. By applying PCA before K-means clustering, we reduce the dimensionality of the data while still retaining most of the important information, allowing K-means clustering to work more effectively in a lower-dimensional space. In addition, K = 5 seems to be a bit large and causes overlap.

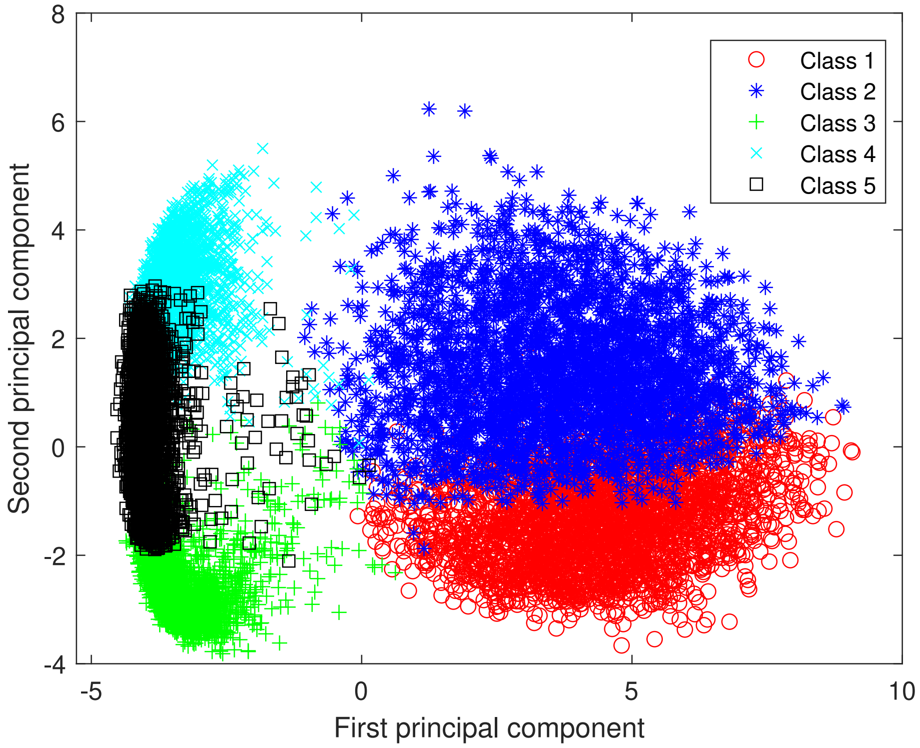

Figure 3: K-means clustering with K = 5

### Task E3:

In this task, we display the K = 2 and k = 5 centroids as images.

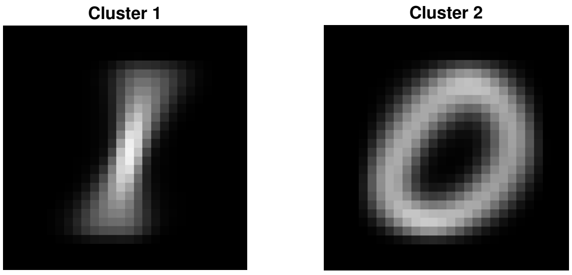

Figure 4: Images of centroids with K = 2

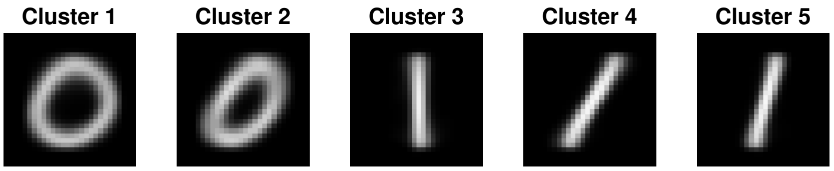

Figure 5: Images of centroids with K = 5

With K = 5, the images are sharper than K = 2. One reason could be that the number of clusters increases so it is possible to focus on numbers with similar properties.

**Task E4:**

Table 3: K-means classification results

| Training data | Cluster | # '0' | # '1' | Assigned to class | # misclassified |
|---|---|---|---|---|---|
| | 1 | 5089 | 6 | 0 | 6 |
| | 2 | 114 | 6736 | 1 | 114 |
| $N_{\text{train}} = 12665$ | | | | Sum misclassified: | 120 |
| | | | | Misclassification rate (%): | 0.95 |
| Testing data | Cluster | # '0' | # '1' | Assigned to class | # misclassified |
| | 1 | 970 | 0 | 0 | 0 |
| | 2 | 10 | 1135 | 1 | 10 |
| $N_{\text{test}} = 2115$ | | | | Sum misclassified: | 10 |
| | | | | Misclassification rate (%): | 0.47 |

The result looks good as the misclassification rates in both training and test data are low.

**Task E5:**

I try K = 3, 5, 7, 9, 11 and 13.

| **K** | **Training data** | **Test data** |
|---|---|---|
| 3 | 0.47 | 0.68 |
| 5 | 0.33 | 0.4 |
| 7 | 0.24 | 0.36 |
| 9 | 0.24 | 0.25 |
| 11 | 0.19 | 0.21 |
| 13 | 0.24 | 0.22 |

Table 4: Misclassification rate (%) using different K values.

With K = 11, the misclassification rate is lowered to 0.21 %.

**Task E6:**

Table 5: 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 |

We can see that the linear SVM classifier gives a better result than the K-means classifier. The misclassification rate is 0% for the training data and 0.095% for the test data.

**Task E7:**

Table 6: Gaussian kernel SVM classification results

| Training data | Predicted class | True class: | # ’0’ | # ’1’ |
|---|---|---|---|---|
| | ’0’ | | 5923 | 0 |
| | ’1’ | | 0 | 6743 |
| $N_{\text{train}} = 12665$ | 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 |

I do a while loop until the misclassification rate of the test data is 0. The training is quite slow so each iteration I increase $\beta$ by 0.2. The best result I got is $\beta = 5$.

**Task E8:**

We cannot expect the same error rate on new images even if we achieve very low misclassification rates on both the train and test data.

The reason for this is that the train and test data may not be fully representative of all image types. The model may be overfitting to the patterns present in the train and test data, resulting in very good performance on these particular images but poor performance on new images that may have different characteristics.

Moreover, if the training data is not diverse enough, the model may not generalize well to new images outside the training set.