FMAN45 Machine Learning, spring 2020
Assignment 2

***Instructions**: Read this assignment description carefully, solve the tasks stated, and typeset your solutions in, e.g., LaTeX. Download the provided data and skeleton code, and complete the programming tasks. Summarize your solutions and results concisely in a detailed report. All necessary solutions, plots and figures should be in one self-contained pdf-document. It should thus be possible to understand all material presented in the report without running any code. Submit your pdf-document as well as the code you've written and data you've created in a single archive via Moodle before the deadline. Do not include the original data in your archive.*

1 Solving a nonlinear kernel SVM with hard constraints.

In the lecture we have seen the linear Support Vector Machine (SVM) for binary classification, which aims to find parameters $\mathbf{w} \in \mathbb{R}^d$ and a bias term $b \in \mathbb{R}$ such that $\mathbf{w}^\top \mathbf{x}_i + b \geq 1$ if the example $\mathbf{x}_i \in \mathbb{R}^d$ belongs to the positive class ($y_i = +1$), and $\mathbf{w}\mathbf{x}_i + b \leq -1$ when $\mathbf{x}_i$ belongs to the negative class ($y_i = -1$). The size of the margin is inverse proportional to the length of $||\mathbf{w}||$. The margin should be as large as possible, which corresponds to minimizing $||\mathbf{w}||$, or rather $\frac{1}{2}||\mathbf{w}||^2$ for computational reasons. In this section and the next, we will consider the theoretical aspects of SVM classifiers. First, we examine the linear hard margin SVM, which is defined as the solution to the minimization problem:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \ \forall i \end{aligned} \tag{1}$$

Consider the one-dimensional binary classification problem with classes '+1' and '-1':

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

(2)

As can be seen, the dataset is not linearly separable. Instead of trying to solve the above SVM problem, we therefore consider the (non-linear) feature map

$$\boldsymbol{\phi}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \tag{3}$$

and the corresponding kernel is given by $k(x, y) = \boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(y)$ and the kernel matrix by

$$\mathbf{K} = [k(x_i, x_j)]_{1 \leq i,j \leq 4} \tag{4}$$

Task T1: (5 p) Compute the kernel matrix $\mathbf{K}$ using the data from the table.

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

$$\begin{aligned} \underset{\alpha_1,\ldots,\alpha_4}{\text{maximize}} \quad & \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{4} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & \alpha_i \geq 0 \text{ and } \sum_{i=1}^{4} y_i \alpha_i = 0, \quad \forall i \end{aligned} \tag{5}$$

Task T2: (5 p) Solve the maximization problem in (5) for α numerically, using the data in (2). You may use, without a proof that, for the data in (2), the solution satisfies $\alpha = \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$.

Next, we know that, for any support vector x_s, we have

$$y_s \left(\sum_{j=1}^{4} \alpha_j y_j k(x_j, x_s) + b \right) = 1 \tag{6}$$

and that the equation for the classifier is given by

$$g(x) = \sum_{j=1}^{4} \alpha_j y_j k(x_j, x) + b \tag{7}$$

Task T3: (10 p) For the data-target pairs in (2), reduce the classifier function (7) to the most simple the simplest possible form, leading to a simple polynomial in x.

Now consider another binary classification problem with classes '+1' and '-1', and data:

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

(8)

Task T4: (5 p) With the same kernel $k(x, y)$ as above, what is the solution $g(x)$ of the nonlinear kernel SVM with hard constraint on the dataset in (8)? Explain how you got to this solution.

2 The Lagrangian dual of the soft margin SVM

In the soft-margin SVM, we allow errors ξ_i in this classification tasks for each data point $\mathbf{x}_i$. We denote the collection of all ξ_i by $\boldsymbol{\xi}$. The primal formulation of the linear soft margin classifier is given by

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{9}$$

Task T5: (10 p) Show that the Lagrangian dual problem for (9) is given by

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

Task T6: (5 p) Use complementary slackness (of the KKT conditions) to show that support vectors with $y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 1$ have coefficient $\alpha_i = C$.

3 Dimensionality reduction on MNIST using PCA

In the experimental part of this assignment you shall work with the MNIST image data set of handwritten digits, containing data-target pairs with images $\mathbf{X}_i \in [0,1]^{28\times 28}$ and targets $t_i \in \{0,1,\ldots,9\}$. You will only consider a part of the data set, specifically only images with targets $t_i \in \{0,1\}$. The dataset `MNIST_01.mat` contains 12665 training data-target pairs {`train_data, train_labels` } and 2115 test data-target pairs {`test_data, test_labels` }, where for both sets the images have been stacked column-wise, i.e., $\mathbf{x}_i = \text{vec}(\mathbf{X}_i) \in \mathbb{R}^{28^2}$

The MNIST images are of quite low resolution, but the dimensionality is still quite large, i.e., $D = 28^2 = 784$. One way of visualizing a collection of high-dimensional data examples is to use a dimensionality reduction technique. Principal component analysis (PCA) is a method for reducing the dimensionality of a dataset, while retaining as much of the data's variability as possible. PCA utilizes the singular value decomposition (SVD), which for a matrix of N zero-mean data examples of dimension D, i.e., $\mathbf{X} \in \mathbb{R}^{D\times N}$, is the factorization:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \tag{11}$$

where for $P = \min(D,N)$; $\mathbf{U} \in \mathbb{R}^{D\times P}$, $\mathbf{S} = \text{diag}(\mathbf{s}), \mathbf{s} \in \mathbb{R}^P$, and $\mathbf{V} \in \mathbb{R}^{N\times P}$ are the left singular vectors, the diagonal matrix of P many singular values, and the right singular vectors, respectively. The dimensionality reduction of $\mathbf{X}$ to dimension d is then the projection of $\mathbf{X}$ onto the d left singular vectors with the largest absolute singular values.

Task E1: (5 p) Compute a linear PCA and vizualize the whole training data in $d = 2$ dimensions. Make sure that the necessary condition to apply PCA is met. Display your results in a plot where the data examples are marked with different markers and colors for each of the two classes. Make the plot clearly interpretable by using legends, adjusting font size, etc.

4 Clustering of unsupervised data using K-means

In this section, you shall cluster the MNIST images without using their target values. K-means clustering is an unsupervised approach for computing K many clusters by iteratively determining the so-called centroids of each cluster and assigning each sample to a cluster. The centroid of a cluster is defined as the mean location of all samples within the cluster. In the assignment step, each sample gets assigned to the closest centroid. Algorithm 1 summarizes the K-means clustering approach.

Algorithm 1 K-means clustering

Select number of clusters K, convergence threshold $\epsilon_{\text{tol}} > 0$,
and maximum iterations $j_{\max}$.
Initialize K centroids $C^{(0)} = \left[\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_K^{(0)}\right]$ randomly.
for $j = 1, \ldots, j_{\max}$ **do**
  Step 1: Assign examples to clusters:
  $d^i = f_{\text{xdist}}(\mathbf{x}_i, \mathbf{C}^{(j-1)}), i = 1, \ldots, N$
  $y_i = \arg\min_k d_k^i, i = 1, \ldots, N$
  Step 2: Assign new cluster centroids:
  $N_k = \sum_{i=1}^{N} 1\{y_i = k\}$, k = 1,. . . ,K
  $\mathbf{c}_k = N_k^{-1} \sum_{i=1}^{N} 1\{y_i = k\}\mathbf{x}_i$, k = 1,. . . ,K
  Check convergence:
  if $f_{\text{cdist}}(\mathbf{C}^{(j)}, \mathbf{C}^{(j-1)}) < \epsilon_{\text{tol}}$ **then**
    return $\mathbf{y}^{(j)}, \mathbf{C}^{(j)}$
  end if
end for
return $\mathbf{y}^{(j_{\max})}, \mathbf{C}^{(j_{\max})}$

Task E2: (20 p) Implement K-means clustering for the training data, using the provided function sketch `K_means_clustering`. Do so by completing the following steps:

1. Define and implement a distance function $f_{\text{xdist}}(\mathbf{x}, \mathbf{C})$ which computes a distance of your chosing between a single example and the K centroids.

2. Define and implement a pairwise distance function, $f_{\text{cdist}}(\mathbf{C}, \tilde{\mathbf{C}})$, which computes a distance of your choosing between two cluster centroids, such that the distance is exactly zero when the pairs of centroids agree.

3. Construct the first step of the K-means clustering algorithm: Define and implement a function `step_assign_cluster` which assigns each sample to one of the K clusters.

4. Construct the second step of the K-means clustering algorithm: Define and implement a function `step_compute_mean` which assigns new centroids by computing the means of the newly assigned clusters, outputting both the new centroids and a measure of the distance which the centroids have moved. The latter will be used as a stopping criterion; the algorithm will stop when the centroids have changed less than a specificed threshold.

5. Implement a fully running `K_means_clustering` using the components you've implemented above, and run the algoritm on the training data for $K = 2$ and $K = 5$ clusters. Reuse the pre-trained linear PCA from Task E1, and visualize your results in two dimensions for each K in a plot where the data examples are marked with a different marker and color for each cluster. Make the plot clearly interpretable by using legends, adjusting font size, etc. Explain why the clusters seem to overlap for, e.g., $K = 5$.

We could now check only a few samples from each cluster to assign a suitable label to the centroid. Having assigned a label to a centroid, we assign the same label to all samples in the corresponding cluster. Alternatively, we can plot the centroid image and assign a suitable class manually. These approaches are advantageous if labeling of all samples individually is expensive. Here, we have the privileged situation to know target labels for all samples, so we may also assign the label to a centroid that is the most frequent in the cluster. Assigning the same label to all samples in the cluster will lead to some misclassifications. By summing the number of misclassifications in each cluster, a missclassification

rate may be calculated for the dataset.

Task E3: (5 p) Display the K centroids as images. Make a plot with $1 \times K$ subplots illustrating the centroids from each cluster as an image, using the Matlab function `imshow()`. Make legends, titles or otherwise mark which cluster is displayed. Note that you have to stack the centroids back into the shape $\mathbf{X}_i \in \mathbb{R}^{28\times 28}$ to display the image properly, using, e.g., `reshape()`.

Task E4: (5 p) Now you shall use K-means clustering for classification, by completing the following steps:

1. For $K = 2$, implement a function `K_means_classifier` which assigns to a given example the label of the closest centroid. You may use the distance function calculated in Task E2. Then assign each cluster centroid the label of which it has the most examples of **in the training data**.
2. Evaluate how many misclassifications occur for the train and test set `train_data` and `test_data` by comparing the computed cluster labels to the true labels in `train_labels` and `test_labels`. Then produce and fill in the following table (which is provided in the supplementary material):

Table 1: K-means classification results

Training data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	?	?	?	?
	⋮	⋮	⋮	⋮	⋮
	K	?	?	?	?
$N_{\text{train}} = ?$				Sum misclassified:	?
				Misclassification rate (%):	?
Testing data	Cluster	# '0'	# '1'	Assigned to class	# misclassified
	1	?	?	?	?
	⋮	⋮	⋮	⋮	⋮
	K	?	?	?	?
$N_{\text{test}} = ?$				Sum misclassified:	?
				Misclassification rate (%):	?

Task E5: (5 p) Can you lower the misclassification rate further on test data by considering a different number of clusters K?

5 Classification of MNIST digits using SVM

In the previous section, you implemented a classifier using unsupervised data, by virtue of the data structure itself. In this section, you shall consider a supervised classifier, namely the support vector machine (SVM). Your task is to use the soft-margin SVM for binary classification, which solves the optimization problem

$$\begin{aligned} \underset{\mathbf{w},b,\xi_1,\ldots,\xi_N}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{12}$$

where $y_i = 1$ for target $t_i = 1$, and $y_i = -1$ for target $t_i = 0$.

Task E6: (5 p) Use the supervised training data {`training_data, training_labels`} to train a linear SVM classifier:

1. One trains the soft-margin SVM by solving the Langrangian dual problem derived in Task T5 using numerical methods. Train a binary soft-margin SVM using Matlab's built-in function `fitcsvm(X,T)` on the training data `train_data` and targets `train_labels`, where `X` denotes the $N \times D$ matrix of data examples, and `T` the target vector of length N. Note that the data provided is of size $D \times N$, so you need to transpose it to match the input format. The parameter C in the SVM problem has to be set. The Matlab implementation uses a default value of $C = 1.0$, which you may leave unchanged.

2. Calculate class predictions using the built-in Matlab function `predict(model,X)`, where `model` is the output from `fitcsvm()`, and `X` is the $N \times D$ dataset matrix to do predictions on. Then evaluate the misclassification rate on both the training and test data by filling in the following table (which is provided in the supplementary material):

Table 2: Linear SVM classification results

Training data	Predicted class	True class: # '0'	# '1'
	'0'	?	?
	'1'	?	?
$N_{\text{train}} = ?$		Sum misclassified:	?
		Misclassification rate (%):	?
Testing data	Predicted class	True class: # '0'	# '1'
	'0'	?	?
	'1'	?	?
$N_{\text{test}} = ?$		Sum misclassified:	?
		Misclassification rate (%):	?

Next, you shall look at SVM with kernels, more specifically a Gaussian kernel. The kernel SVM solves the SVM optimization problem not on the data directly, but in a feature space $\mathbf{z} = \boldsymbol{\phi}(\mathbf{x})$, such that the classifier is linear in $\mathbf{z}$. In practice, this is done using the so called kernel trick, such that data is never mapped into this feature space explicitly, but by modifying the dual problem appropriately.

Task E7: (10 p) Use the supervised data again to train a **non-linear** kernel SVM classifier, using a Gaussian kernel:

1. Train an SVM with Gaussian kernel using Matlab's built-in function, by specifying `fitcsvm(X,T,'KernelFunction','gaussian')`.

2. The Gaussian kernel has a scaling parameter σ^2, which in Matlab's SVM estimator is set to $\beta = \sqrt{1/\sigma^2} = 1$ by default, but may be modified by specifying `fitcsvm(X,T, 'KernelFunction','gaussian', 'KernelScale',beta)` for some value `beta`. Can you lower the misclassification rate on the test data by tuning `beta`?

3. Present your best results, similar to Task E5, by filling in the following table (which is provided in the supplementary material):

Table 3: Gaussian kernel SVM classification results

Training data	Predicted class	True class: # '0'	# '1'
	'0'	?	?
	'1'	?	?
$N_{\text{train}} = ?$		Sum misclassified:	?
		Misclassification rate (%):	?
Testing data	Predicted class	True class: # '0'	# '1'
	'0'	?	?
	'1'	?	?
$N_{\text{test}} = ?$		Sum misclassified:	?
		Misclassification rate (%):	?

Task E8: (5 p) We can achieve very low misclassification rate on both train and test data with a good choice of parameters for the Gaussian kernel SVM. Can we therefore expect the same error on new images? Explain why such a conclusion would be false.

A Supplementary material

Contents of data archive `A2_data.mat` used in sections 3-5:

`train_data_01`	MNIST images of '0' and '1' for training, $\mathbf{X} \in \mathbb{R}^{784\times 12665}$
`train_labels_01`	MNIST labels '0' and '1' for training, $\mathbf{t} \in \mathbb{R}^{12665}$
`test_data_01`	MNIST images of '0' and '1' for testing, $\mathbf{X} \in \mathbb{R}^{784\times 2115}$
`test_labels_01`	MNIST labels '0' and '1' for testing, $\mathbf{t} \in \mathbb{R}^{2115}$

Provided files:

- `K_means_clustering.m`

  Matlab function sketch `[y,C] = K_means_clustering(X,K)`, returns cluster assignments `y` and cluster centroids `C` given an input of data `X` the number of clusters `K`.

- `A2_tables.tex`:

  LaTeX-formatted tables from Section 3 and 4, to be filled in and used in the report.