DATA11002 Introduction to Machine Learning, Fall 2018
Exercise set 5

Due December 5th–7th. (Note that December 6th is the Independence Day and there are **no exercise sessions on Dec 6**. Please visit the other groups, and if you really cannot go to any other group, send your results by email to Janne Leppä-Aho. Kindly indicate in the body of the email which exercises you have done.)

Continue reading the textbook: Chapter 10 (Clustering and dimension reduction)

Problem 1 (2+2 points)

(a) (2 points) Given a set of n numbers $x_1, \ldots, x_n$, with $x_i \in \mathbb{R}$, show that the value x^* that minimizes the *sum of squared errors*, i.e.

$$x^* = \arg\min_{x'} \sum_{i=1}^{n} (x_i - x')^2 \tag{1}$$

is given by the *average* of the x_i, i.e. $x^* = \sum_i x_i / n$.

Hint: Take the derivative of the sum of squared errors and set it to zero.

(b) (2 points) Do the same for vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where $\mathbf{x}_i \in \mathbb{R}^p$. In other words, show that the minimum sum of squared distances

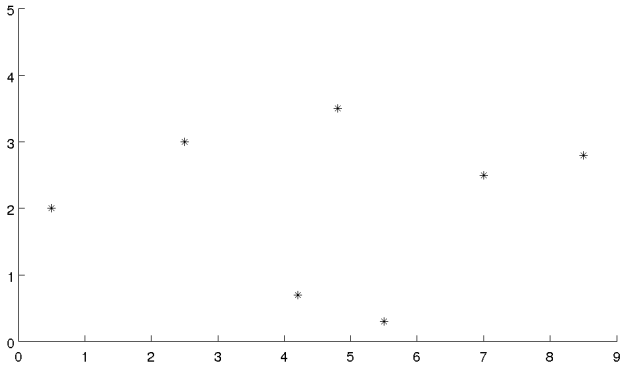
$$\mathbf{x}^* = \arg\min_{\mathbf{x}'} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{x}'\|_2^2 \tag{2}$$

is given by the average vector $\mathbf{x}^* = \sum_i \mathbf{x}_i / n$.

Hint: Recall that the squared Euclidean norm is given by $\|\mathbf{z}\|_2^2 = \sum_{j=1}^{p} \mathbf{z}_j^2$. You can break down the sum of squared errors in terms of the different dimensions and inspect the effect of the choice of $\mathbf{x}^*$ on each of the p parts in the sum.

Problem 2 (3+3 points)

We consider hierarchical clustering on a toy data set consisting of seven data points in the Euclidean plane. The data points are $p_1 = (0.5, 2.0)$, $p_2 = (2.5, 3.0)$, $p_3 = (4.2, 0.7)$, $p_4 = (5.5, 0.3)$, $p_5 = (4.8, 3.5)$, $p_6 = (7.0, 2.5)$ and $p_7 = (8.5, 2.8)$, or as a picture:



Exercises continued on the next page...

The matrix of Euclidean distances between the data points is then as follows:

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1	0	2.24	3.92	5.28	4.55	6.52	8.04
p_2	2.24	0	2.86	4.04	2.35	4.53	6.00
p_3	3.92	2.86	0	1.36	2.86	3.33	4.79
p_4	5.28	4.04	1.36	0	3.28	2.66	3.91
p_5	4.55	2.35	2.86	3.28	0	2.42	3.77
p_6	6.52	4.53	3.33	2.66	2.42	0	1.53
p_7	8.04	6.00	4.79	3.91	3.77	1.53	0

(a) (3 points) Simulate the basic agglomerative hierarchical clustering by hand (so not using R or other software) to this data using the single linkage notion of dissimilarity between clusters. Visualise the result as a dendrogram.

Hint: Feel free to skip some calculations if you can clearly see what the next step in the algorithm is, but always calculate at least the cost of the join that you select.

(b) (3 points) Repeat the clustering using now complete linkage dissimilarity. Compare the results.

Problem 3 (6+2+3+3 points)

In this problem you will implement the K-means algorithm, so don't use an existing implementation (such as `kmeans` in R). Remember that you are allowed to use any programming language, but even in that case, don't use an existing implementation of the algorithm itself.

(a) (6 points) Write your own implementation of the so called Lloyd's algorithm for K-means. The algorithm is explained in the slides and Algorithm 10.1 in the textbook.

This should be a function that takes as inputs the data matrix, and outputs the final cluster means and the assignments specifying which data vectors are assigned to which cluster after convergence of the algorithm. (Use matrix operations wherever possible, avoiding explicit loops, to speed up the algorithm sufficiently for running the algorithm on the MNIST data below.)

Test the algorithm by clustering $n = 100$ random data points drawn from a bivariate standard normal distribution with mean $\boldsymbol{\mu} = (0, 0)$ and covariance $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Plot the results as a scatter plot where different clusters are shown in different colors. Repeat the experiment with the same data to see how much the clusters differ.

Hint: You may recall from the previous exercises that normal distirbuted feature vectors with diagonal covariance can be drawn by independently generating one-dimensional normal distributed features.

(b) (2 points) This wouldn't be a proper ML course if we didn't use the classic MNIST data! Download the classic MNIST handwritten digit database from `http://yann.lecun.com/exdb/mnist/`, and load the data into R.[1] Display the fifth training data instance on the screen to make sure you have succeeded. It should look more or less like a '9' (or a letter 'a' leaning to the right but these are all supposed to be digits 0–9). Verify the that the correct class value, y, of the fifth training instance is indeed 9 by printing the value `train$y[5]`.

Exercises continued on the next page...

[1] Brendan O'Connor has kindly written a handy R script for reading the files: `https://gist.github.com/brendano/39760`. Just remember to put the files in folder `mnist` and unzip them. **NB:** Some systems may put a dot '.' in the file names where there should be a dash '-'. If the data loading script complains, check that the file names in the script match the actual file names.

(c) (3 points) Having loaded the data, discard all but the first 500 training data points.[2] Run your K-means algorithm, using $K = 10$ clusters, with the initial cluster means equal to the first 10 images in the dataset. After convergence, visualize the cluster prototypes as an image showing the mean vector (mean grayscale value of each pixel).

Also visualize some of the images belonging to each cluster.

To what extent do the 10 clusters correspond to the 10 different digits?

(d) (3 points) Re-run K-means but selecting the first instance of each class as the initial cluster mean (so that the initial cluster means all represent distinct digits), and compare with the previous results.

Hint: The correpondence between clusters and digits should now be better than in the previous case, but some images of different digits are definitely clustered together, and it should be clear from the visualization that the clustered images may be quite similar in many ways even if they represent different digits.

[2]You can try including more data if the run-time is tolerable.