582631 Introduction to Machine Learning, Autumn 2013
Exercise set 6 (based mainly on the fifth week of lectures)

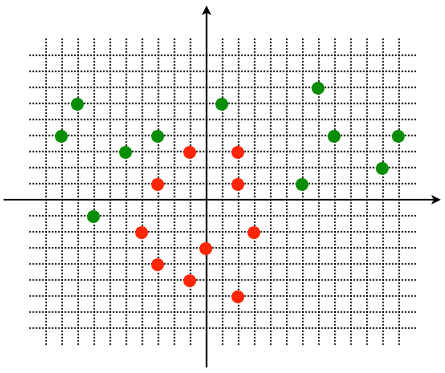
Since Friday 6 December is a public holiday, there will be *no exercise session* for discussing this final problem set. However, you get credit for this set as usual by handing in your written solutions.

The deadline for handing in your solutions is **9:00am on Wednesday, 4 December**.

Send your solutions to the course assistant (Yuan.Zou@cs.helsinki.fi) by e-mail. You should send one PDF file including your solutions to the pen-and-paper problems and the report for the programming problem, and a single compressed file including the code for the programming problem.

Pen-and-paper problems

Problem 1 (3 points) In the figure below, the red points correspond to class $y = +1$ and the green points to class $y = -1$, and the horizontal axis to the attribute x_1 and the vertical axis to the attribute x_2. Is this problem linearly separable, i.e. can you draw a straight line which separates the red from the green points? If not, try to find the linear decision boundary which produces the fewest misclassifications; what is the minimum number of errors? Can you nonlinearly transform the input attributes such that the problem becomes linearly separable? That is, give a function $\mathbf{z} = g(\mathbf{x})$ where $\mathbf{x} = (x_1, x_2)^T$ is the original input vector, such that a linear classifier applied to $\mathbf{z}$ can perfectly separate the two classes.



Problem 2 (3 points)

The *K-medoids* algorithm is similar to K-means but can work with any data objects for which a pairwise similarity measure is defined, including cases for which taking the mean of a set of objects does not make sense. Instead of selecting the *mean* of the objects belonging to a cluster as the prototype for the cluster, the K-medoids algorithm selects the most representative of the objects belonging to the cluster as the prototype. Thus the K-medoids algorithm is exactly like the K-means algorithm (Algorithm 8.1 in the textbook, also presented in the slides for lecture 10), except that in line 4 of the algorithm, in K-medoids the cluster prototype is selected to equal one of the objects assigned to that cluster, such that the *sum* of the similarities between the prototype and the objects in the cluster is maximized.

(a) Is this algorithm guaranteed to converge after a finite number of steps? If so, show this. (Hint: what is the corresponding objective function that the algorithm optimizes?)

(b) If the data objects are points in the Euclidean space, and the similarity measure for the K-medoids algorithm is taken to be the negative of the squared Euclidean distance, in general, which algorithm (K-means or K-medoids) would typically result in a lower sum of squared errors?

Problem 3 (3 points)

One practical problem with K-means is that it is possible for clusters to become empty. That is, in some cases the centroids $\mathbf{c}_j$ may end up in such locations that no points are assigned to one (or several) of the clusters! Such an example is trivial to construct with random initial locations for the centroids (since a centroid can simply be initialized very far from the data), but can also happen when centroids are initialized to equal randomly chosen datapoints (even when no two centroids are initialized to the exact same location).

(a) Manually construct such an example. (Hint: This problem does not happen in the first iteration since at least the datapoint which is equal to the centroid will be assigned to that centroid at first. But it can happen in the second iteration. It is probably simplest to construct with one-dimensional samples $x_i \in R$. You will need at least $K = 3$. Six datapoints are sufficient if suitably placed, but feel free to use more if you wish.) If you find it hard to manually construct such a case, try to find such an example in numerical simulations (using the K-means algorithm that you construct in Problem 4 below).

(b) What are some strategies that you could employ to handle this problem when it occurs in the K-means algorithm?

Computer problem

General instructions:

- Return a brief written report (as PDF, included in the same file as your pen-and-paper solutions) and one directory (as a zip or tar.gz file) including all your code.
- Do not include any of the data files in your solution file.
- Always mention your name and student ID in the report.
- We use the report as the main basis for grading: All your results should be in the report. We also look at the code, but we won't however go fishing for results in your code.
- The code needs to be submitted as a runnable file or set of files (command to run it given in the report).
- In your report, the results will be mostly either in the form of a figure or program output. In both cases, add some sentences which explain what you are showing and why the results are the answer to the question.
- If we ask you to test whether an algorithm works, always give a few examples showing that the algorithm indeed works

Problem 4 (15 points)

In this problem we will implement the K-means algorithm and the K-medoids algorithm (see Problem 2) and test them on the MNIST handwritten digit data.

(a) Write your own implementation of the K-means algorithm (Algorithm 8.1 in the textbook, also presented in the slides of lecture 10). This should be a function that takes as inputs the data matrix and initial centroids, and outputs the final centroids and the assignments specifying which data vectors are assigned to which centroids 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.)

(b) Load the first 500 of the MNIST digits. Run your K-means algorithm, using $K = 10$ clusters, with the initial centroids equal to the first 10 images in the dataset. After convergence, for each cluster, visualize the centroid of that cluster as well as all images assigned to that cluster (using the provided `visual()` function). To what extent do the 10 clusters correspond to the 10 different digits? Re-run K-means but selecting the first instance of each class as the initial centroids (so that the initial centroids all represent distinct digits), and compare with the previous results. (Hint: There should be some correpondence, but

some digits are definitely clustered together, and it should be clear from the visualization that the clustered images are 'similar' even if they do not necessarily represent the same digits.)

(c) Write your own implementation of the K-medoids algorithm (see Problem 2 above). This should be a function that takes as inputs a similarity matrix and the initial medoids, and outputs the final medoids and the assignments specifying which data vectors are assigned to which clusters after convergence of the algorithm. (As above, use matrix operations to avoid explicit loops as much as opssible.)

(d) Run your K-medoids implementation on the first 500 digits of the MNIST dataset, mirroring what you did in part (b) above. For simplicity, use the negative of the euclidean distance as the similarity measure, but you can also try other similarity measures if you wish. Just as in part (b), try running the algorithm with the first 10 images as the initial medoids, and compare with selecting the initial medoids so that they all represent distinct digits. As in (b), visualize your results by, for each cluster, showing the cluster prototype (the medoid), as well as all images belonging to the cluster.

(e) What do your results say about the sensitivity of K-means and K-medoids to the specification of the initial cluster prototypes? What would be one way of dealing with this issue?