Machine Learning
2nd-Term Semester 2016-2017
Assignment 1

SYM

February 13, 2017

General instructions: All course participants are requested to handle their exercise solutions as follows:

- Write your answers of **theory section** and your report of **programming section**, both as PDF, using one of the following text processing tools: MS-Word, LibreOffice, or Latex.
- Always mention your **name** and **student ID** in the PDF file.
- For programming section, write the source code using programming language that you prefer or familiar with.
- The assignment is designed to be solved in a week. However you could accomplish it less than a week if you allocate your time properly to work on it.
- The deadlines:
  - questions number 1 until 5 is on Thursday 16.2.2017,
  - question number 6 is on Monday 20.2.2017,
  - question number 7 is on Thursday 23.2.2017.

  The time of all deadlines are at 23.59 UTC+7.
- Submit your work (PDF and all your codes) into one directory (as a **zip** file) through email to the **lecturer** and the **teaching assistant** before the deadline. Do not include any of the data files in your solution file.
- In all the exercises, do not just give your answer, but also the derivation of how you obtained it.
- Cheating is strictly forbidden.

Section 1: Theory

1. **(3 points)** What is machine learning?
2. **(3 points)** Give an example of machine learning implementations? (Give other examples than those mentioned on the slides)
3. In this exercise we explore the relationships between the cosine and correlation similarity measures and Euclidean distance for data vectors in R^n.
   (a) **(2 points)** What is the range of values that are possible for the cosine measure?
   (b) **(3 points)** If two objects have a cosine measure of 1, are they necessarily identical? Please explain.
   (c) **(5 points)** What is the relationship of the cosine measure to correlation, if any? (Hint: Look at statistical measures such as mean and standard deviation in cases where cosine and correlation are the same and different.)

(d) **(5 points)** Derive the mathematical relationship between cosine similarity and Euclidean distance when each data object has an L_2 length (norm) of 1.

(e) **(5 points)** Derive the mathematical relationship between correlation and Euclidean distance when each data point has been standardized by subtracting its mean and dividing by its standard deviation.

4. Proximity is typically defined between a pair of objects.

   (a) **(4 points)** Give two ways in which you might define the 'proximity' among a set of (more than two) objects (i.e. a single measure of how similar an arbitrary number of items are all to one another)

   (b) **(4 points)** How might you define the distance between two sets of points in Euclidean space?

   (c) **(4 points)** How might you define the proximity between two sets of data objects? (Make no assumptions about the data objects, except that a proximity measure is defined between any pair of objects.)

5. Consider a document-term matrix, where tf_{ij} is the number of times that the i^{th} word (term) appears in the j_{th} document, and let m be the total number of documents in the collection. Consider the variable transformation that is defined by m,

$$tf'_{ij} = tf_{ij} log \frac{m}{df_i}$$

   where df_i is the number of documents in which the i^{th} term appears, which is known as the *document frequency* of the term. This transformation is known as the *inverse document frequency* transformation.

   (a) **(3 points)** What is the effect of this transformation if a term occurs in only one document?

   (b) **(3 points)** What is the effect of this transformation if a term occurs in every document?

   (c) **(3 points)** What is the overall effect and what might be the purpose of this transformation?

   (d) **(3 points)** Can you think of other (non-document) data in which this transformation might be useful?

Section 2: Programming

Instructions:

- Write a brief written report (as PDF) of programming section.
- 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

6. In this problem we will consider similarity measures for movies on the Movielens dataset.

   Download the Movielens data that we sent to you through email. In addition to the data, the file also contains some functions for easily loading the data into Matlab/Octave/R and some example code that you can use if you wish. See the README files for details.

   (a) We will now construct a similarity measure over the movies. For simplicity, let us first consider a simple measure that does not use the explicit (numerical) ratings given by the users, nor the time stamps of the ratings, but only whether or not a given movie was rated by a given user.

   i. **(5 points)** Create a function that, given two different movie IDs as input, outputs the Jaccard coefficient: the number of users who rated both movies divided by the number of users who rated at least one of the movies. For example, for the movies 'Toy Story' and 'GoldenEye' the coefficient should be 0.217.
   ii. **(1 points)** What is the Jaccard coefficient between 'Three Colors: Red' and 'Three Colors: Blue'?
   iii. **(2 points)** What are the 5 movies with highest Jaccard coefficient to 'Taxi Driver'?
   iv. **(2 points)** Select a movie of your own choosing (which you are familiar with), what are the 5 movies with highest Jaccard coefficient to that movie? Do they make sense?

(b) Now let's try a similarity measure that uses the explicit ratings.

   i. **(5 points)** Create a second function that, given two different movie IDs as input, outputs the correlation coefficient of the ratings given to those two movies by all users which have rated both movies. (Note, the function may need to return 0 when the number of users who have rated both is so low that one cannot compute a correlation coefficient.)
   ii. **(1 points)** What is now the similarity between 'Toy Story' and 'GoldenEye'?
   iii. **(1 points)** How about 'Three Colors: Red' and 'Three Colors: Blue'?
   iv. **(2 points)** What are the 5 movies with highest similarity to 'Taxi Driver'?
   v. **(2 points)** Again, select a movie of your own choosing and list the 5 movies with highest similarity.

(c) **(4 points)** Provide some brief thoughts on which similarity measure seems to work 'better', in the sense that the computed similarity matches your intuitive sense of similarity. Why do you think this is? Explain.

7. In this exercise we implement an extremely simple prototype-based classifier to classify handwritten digits from the MNIST dataset, and compare that to a nearest-neighbor classifier.

   Download the MNIST data that we sent to you through email. In addition to the actual data, the package contains some functions for easily loading the data into Matlab/Octave/R and for displaying digits. See the README files for details.

   (a) **(5 points)** Load the first N=5,000 images using the provided function. Use the provided functions to plot a random sample of 100 handwritten digits, and show the associated labels. Verify that the labels match the digit images. (This is a sanity check that you have the data is in the right format.)

   (b) **(5 points)** Divide the data into two parts: A 'training set' consisting of the first 2,500 images (and associated labels), and a 'test set' containing the remaining 2,500 images (and their associated labels). For each of the ten classes (digits 0-9), compute a class prototype given by the mean of all the images in the training set that belong to this class. That is, select from the training set all images of class '0' and compute the mean image of these; this should look sort of like a zero. Do this for all ten classes, and plot the resulting images. Do they look like what you would expect?

   (c) **(5 points)** For each of the images in the test set, compute the Euclidean distance of the image to all 10 prototypes, and classify the test image into the class for which the distance to the prototype is the smallest. So, if a test image is closer to the prototype for '3' than it is to the prototypes for any of the other digits, predict its class to be '3'. Compute and display the resulting confusion matrix.

   (d) **(5 points)** Classify each of the test images with a nearest neighbor classifer: For each of the test images, compute its Euclidean distance to all (2,500) of the training images, and let the predicted class be the class of the closest training image. Compute and display the resulting confusion matrix.

   (e) **(5 points)** Compute and compare the error rates of both classifiers (the prototype-based classifier and the nearest neighbor classifier). Which is working better? Based on the confusion matrix, which digits are confused with each other? Why do you think this is?