CS 412 — Introduction to Machine Learning — Fall 2019

Assignment 2

Department of Computer Science, University of Illinois at Chicago

Instructor: Xinhua Zhang

Due: Oct 18, 2019 by 3 PM (CST)
Out: Oct 5, 2019

Instructions

This is an **individual** assignment**.** Your mark will be out of 100, and it contributes 8% to your final course score.

What and where to submit:

1. The solution to the written questions (Q1 – Q3) should be submitted to **Gradescope**. You can submit PDF or image. It can be either typed using WORD or latex, or scanned copies of handwritten submissions provided that the handwriting is neat and legible. CamScanner is a great app for scanning on your smartphone. Please specify clearly which pages correspond to which question.

2. The solution to the programming questions must be submitted to **Blackboard** as a single zipped file (.zip or .tar) **named**

Surname_UIN.zip or Surname_UIN.tar

where Surname is your last name and UIN is your UIC UIN.

This zip/tar file should include the following files:

- **a)** A single Portable Document Format (PDF) file, which answers Question 4. The file name should be Surname_UIN.pdf, where Surname is your last name and UIN is your UIC UIN. It must be typed using WORD or latex.
- **b)** **Code**: All the code you wrote – in a form that the TA can run it. Please include plenty of comments. Put all code in one folder named Code/
- **c)** **ReadMe**: A ReadMe file that explains to your TA how to run your code. Name the file as ReadMe under the Code/ folder.

How to submit on Blackboard: Go to the class web site, and folder Assignments. The entry "Assignment 2" in this folder will have a link through which you can upload your abovementioned zip/tar file for Q4 (programming question). **Do NOT submit your solutions to Q1-Q3 on Blackboard; they should go to Gradescope**.

Late policy: Late submissions will not be accepted in any case, unless there is a documented personal emergency. Arrangements must be made with the instructor as soon as possible after the emergency arises, preferably well before the deadline.

Resubmission: Both Blackboard and Gradescope accept resubmission (upload a new version) until the deadline. Grading will be based on the last version uploaded.

Programming Language: **Python only**.

Do NOT submit a Jupyter Notebook. Please submit your code in a standalone form so that the TA can run it directly.

The grading will be based on the output of executing your code on some test examples, along with the performance of algorithms that you plot in the assignment submission. Clarify and readability of your code will also count.

Cheating and Plagiarism: All assignments must be done individually. Remember that plagiarism is a university offence and will be dealt with according to university procedures. Please refer to the corresponding UIC policies: http://dos.uic.edu/docs/Student%20Disciplinary%20Policy.pdf

Latex primer: http://ctan.mackichan.com/info/lshort/english/lshort.pdf

Problem 1. Multivariate Methods (18 points, 9 points each)

(a) Exercise 5.4 of Alpaydin, but instead of four cases, do it only for the case of $\Sigma_1 \neq \Sigma_2$.
(b) Exercise 5.5 of Alpaydin. In particular, write out $g_i(x)$, and explain if the separating boundary is quadratic or linear.

Problem 2. Clustering (24 points, 8 points each)

(a) Exercise 7.1 of Alpaydin. Instead of writing "computer program", you can write out pseudo-code in the PDF file. The compression rate in this question is the ratio between

1) the number of bits needed to transfer the encoded representation of the image;

and

2) the number of bits used to encode the original image.

The key is to consider each c-by-c window as a data point. Then an a-by-b image has $\frac{a}{c} \times \frac{b}{c}$ windows, i.e., data points. N images of size a-by-b altogether make $N \times \frac{a}{c} \times \frac{b}{c}$ data points, and now do k-means on them to get k centers.

Now let's transmit each image window by window (e.g. scanning by rows). If the image is sized a-by-b, then there are $\frac{a}{c} \times \frac{b}{c}$ windows to send. For each c-by-c window (i.e. data point), find the center (among the k-centers) that is most similar to it, and then transfer the center. But instead of sending the centers themselves, we only need to send the index of the centers, which costs $\log k$ bits for each c-by-c window.

Then there is an overhead of sending these centers at the beginning of the transfer, and this only needs to be done once in the following sense:

- For different c-by-c windows of the same image, they share the same set of centers and therefore only need to send the index of the particular center for each window;
- For different images, their c-by-c windows also share the same set of centers and just send the index for each window.

For this question, you can assume that each (original) pixel is represented by 24 bits.

(b) Exercise 7.7 of Alpaydin. Just briefly explain in 3-5 lines.
(c) Exercise 8.4 of Alpaydin. Just briefly explain in 2-4 lines.

Problem 3. Nearest Neighbors (18 points)

Consider the following dataset with six examples:

x_1	x_2	x_3	x_4	y
3	10	2	11	Red
17	-17	9	-1	Blue
-4	9	-2	-1	Red
4	0	2	-5	Blue
8	-1	6	-12	Blue
19	3	23	14	Red

(a) For a new testing example, $x_1 = 0.0$, $x_2 = 0.0$, $x_3 = 0.0$, $x_4 = 0.0$, write the distance to each of the training examples and indicate the prediction made by 1-NN and 3-NN using Euclidean distance. **(6 points)**

(b) For a new testing example, $x_1 = 0.0$, $x_2 = 0.0$, $x_3 = 0.0$, $x_4 = 0.0$, write the distance to each of the training examples and indicate the prediction made by 1-NN and 3-NN using Manhattan distance. **(6 points)**

(c) What is the Leave-One-Out-Cross-Validation (LOOCV) error rate of 1-NN using Manhattan distance on this dataset? Indicate which examples (if any) contribute to the error rate. **(6 points)**

Problem 4. Nearest Neighbors: Training Set Size and Noise (40 points)

Please note before submitting your code : Since the experiments involve randomness, it is important to ensure that your results are replicable. To this end, your implementation should take one integer (or any numeric value) as a seed that is used to initialize the random number generators. See, e.g. `random.seed` at `https://docs.python.org/3/library/random.html`.

(a) Optical character recognition (OCR)

How does the the classification error of the 1-Nearest Neightbor for the MNIST dataset change with the number of training examples? Plot a figure where the x-asix is number of training examples (e.g. 100, 1000, 2500, 5000, 7500, 10000), and the y-axis is test error.

Then plot the n-fold cross validation error for the first **1000** training training examples where $n \in \{3, 10, 50, 100, 1000\}$. That is, plot a figure where the x-asix is $n = 3; 10; 50; 100; 1000$, and the y-axis is n-fold cross validation error.

Please submit a print out of your plots and relevant code snippet. **(20 points)**

Note: Code example for creating line chart and image plot is given in the last part of `mnist1NNdemo.py`.

(b) Iris plant recognition

Within your working directory, run `iris1NNboundary.py` to produce a plot of the decision boundary of 1-Nearest Neighbors classifier. You will be making use of and modifying this source code for this problem.

The iris data set has 150 examples and 3 classes. Now randomly choose m examples, $m \in \{10, 20, 30, 50\}$ and flip the class label of each of the examples. For instance if the true class label of example i is 1 then replace the class label with either 2 or 3. Now you will have 4 modified datasets. For each of the modified datasets, run the k-NN algorithm with $k = 3$ and then plot the decision boundary. Also report the training error (leave-one-out validation error).

Please submit a print out of your results and relevant code snippet.