Introduction to learning and analysis of big data
Exercise 1

Dr. Sivan Sabato

Fall 2019/20

Submission guidelines, **please read and follow carefully**:

- The exercise is submitted in pairs.
- Submit using the submission system.
- The submission should be a zip file named "ex1.zip".
- The zip file should include **only the follwing files in the root - no subdirectories please**.
  1. A file called "answers.pdf" - The answers to the questions, including the graphs.
  2. Matlab files (with extension ".m") - The Matlab code for the requested functions in the first question below. Note that you can put several auxiliary functions in a matlab file after the definition of the main function. **Make sure that the code works in Matlab/Octave before you submit it.**

  Anywhere in the exercise where Matlab is mentioned, you can use the free software Octave instead.
- Getting started with Matlab: You can use the guide in this link: `http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf?s_tid=int_tut`. There are plenty of other tutorials, documentation and examples on the web. You can run Matlab on the lab computers. Octave is free to download and use.
- For questions use the course Forum, or if they are not of public interest, send them via the course requests system.
- Grading: Q.1 (matlab code): 20 points, Q.2: 20 points, Q.3: 24 points, Q.4: 16 points. Q.5: 20 points.

Question 1. Implement a function that runs the **k-nearest-neighbors** algorithm that we saw in class on a given training sample, and a second function that uses the output classifier of the first function to predict the label of test examples. The first function creates the classification rule. It should be submitted in a matlab file called **"learnknn.m"**. The first line in the file (the signature of the function) should be:

```
function classifier = learnknn(k, d, m, Xtrain, Ytrain)
```

The input parameters are:

- `k` - the number k to be used in the k-nearest-neighbor algorithm. You may assume that k is an odd integer.
- `d` - the number of features in each example in the training sample, an integer $d \geq 1$. So $\mathcal{X} = \mathbb{R}^d$.
- `m` - the size of the training sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, an integer $m \geq 1$.
- `Xtrain` - a 2-D matrix of size $m \times d$ (rows $\times$ columns). Row i in this matrix is a vector with d coordinates which is example x_i from the training sample.
- `Ytrain` - a column vector of length m (that is, a matrix of size $m \times 1$). The i's number in this vector is the label y_i from the training sample. You can assume that each label is an integer between 0 and 9.

The output of this function is `classifier`: a data structure that keepd all the information you need to apply the k-nn prediction rule on new examples. The internal format of this data structure is your choice. The second function uses the classification rule that the first function learned for new examples. It should be submitted in a matlab file called **"predictknn.m"**. The first line in the file (the signature of the function) should be:

```
function Ytestprediction = predictknn(classifier, n, Xtest)
```

The input parameters are:

- `classifier` - the classifier to be used for prediction. Only classifiers that your own code generated using the first function can be used here.
- `n` - the number of test examples to classify.
- `Xtest` - a 2-D matrix of size $n \times d$. Each row in this matrix is a vector with d coordinates that describes one example that the function needs to label.

The output is `Ytestprediction`. This is a column vector of length `n`. Label i in this vector describes the label that `classifier` predicts for the example in row i of the matrix `Xtest`.

Important notes:

- You may assume all the input parameters are legal.
- You do not need to come up with a very efficient implementation, implement a naive solution that works.
- The Euclidean distance between two vectors `z1,z2` of the same length can be calculated in Matlab/Octave using `norm(z1-z2)`.

Example for using the functions (here there are only two labels, 0 and 1):

```
>> k=1;
>> m=3;
>> d=2;
>> Xtrain=[1,2;3,4;5,6];
>> Ytrain=[1;0;1];
>> classifier = learnknn(k,d,m, Xtrain,Ytrain);
>> n=4;
>> Xtest=[10,11;3.1,4.2;2.9,4.2;5,6];
>> Ytestprediction = predictknn(classifier, n, Xtest)
>> Ytestprediction

Ytestprediction =

1
0
0
1
```

Question 2. Test your k-nearest-neighbor implementation on the **hand-written digits recognition** learning problem: In this problem, the examples are images of hand-written digits, and the labels indicate which digit is written in the image. The full data set of images, called MNIST, is free on the web. It includes 70,000 images with the digits 0-9. For this exercise, we will use a smaller data set taken out of MNIST, that only includes images with the digits 0,3,5 and 8, so that there are only four possible labels.

Each image in MNIST has 28 by 28 pixels, and each pixel has a value, indicating how dark it is. Each example is described by a vector listing the $28 \cdot 28 = 784$ pixel values, so we have $\mathcal{X} = \mathbb{R}^{784}$, every example is described by a 748-coordinate vector.

Figure 1: Some examples of images of digits from the data set MNIST

The images in MNIST are split to training images and test images. The test images are used to estimate the success of the learning algorithm: In addition to the training sample S as we saw in class, we have an additional **test sample**, T which also consists of pairs of labeled examples.

The k-nn algorithm gets S, and decides on $\hat{h}_S$. Then, the prediction function can predict the labels of the test images in T using $\hat{h}_S$. The error of the prediction rule $\hat{h}_S$ on the test images is $\mathrm{err}(\hat{h}_S, T) = \frac{1}{m}\sum_{(x,y)\in T} \mathbb{I}[\hat{h}_S(x) \neq y]$. It is a good estimate of the error of $\hat{h}_S$ on the distribution $\mathrm{err}(\hat{h}_S, \mathcal{D})$.

You will need the following files, which can be found in the Assignment page in the course website:`mnist_all.mat` and `gensmallm.m`.

To load all the MNIST data to Matlab, run the following command (if the file `mnist_all.mat` is not in your Matlab path or current directory, add the file path to the name of the file):

```
>> load('mnist_all.mat');
```

After this command you will have the variables:
`train0,train1,...,train9,test0,test1,...,test9` in your Matlab workspace.
To generate a training sample of size `m` with images only of some of the digits, you can use the provided function `gensmallm`, which is used as follows:

```
>> function [X,Y] = gensmallm(labelAsample,labelBsample,A, B, samplesize)
```

The function `gensmallm` selects a random subset from the provided data of labels A and B and mixes them together in a random order, as well as creates the correct labels for them. This can be used to generate the training sample and the test sample. To mix more than two digits, you can use the function code as a basis and change it as you wish. Be careful to make sure all digits have the same probability of being selected.

Answer the following questions in the file "answers.pdf".

(a) Run your k-nn implementation with $k = 1$, on several training sample sizes between 1 and 100 (select the values of the training sizes that will make the graph most informative). For each sample size that you try, calculate the error on the full test sample. You can calculate this error, for instance, with the command:

```
>> mean(Ytest ~= Ytest_predict).
```

Repeat each sample size 10 times (with a different random training sample) and average the measurements of the error for each size. Submit a plot of the average test error (between 0 and 1) as a function of the training sample size. Don't forget to label the axes. You can use Matlab's `plot` command (or any other plotting software).

(b) Did you get different results in different runs with the same sample size? Why?

(c) Do you observe a trend in the graph? What is it? How do you explain it?

(d) Run your `nn` implementation with a training sample size of a 100, for values of k between 1 and 11. Submit a plot of the test errors as a function of k, again averaging 10 runs for each k.

(e) Do you observe a trend in this graph as a function of k? Try to explain the results that you got.

(f) Choose one of your runs which had a low error. For the output of this run (provide that run's parameters), provide a *confusion matrix*: A table where each column and each row correspond to one of the digits, and the number in cell (digit1,digit2) is the percentage of images in the test set that are actually digit1, and were predicted by the algorithm to be digit2. Sanity check: the sum of all off-diagonal entries should be the test error. Explain the findings you observe in the confusion matrix.

Question 3. In the following, assume $\mathcal{Y} = \{0, 1\}$.

(a) Let (X_1, Y_1) and (X_2, Y_2) be two random labeled examples drawn independently from some distribution $\mathcal{G}$ over $\mathcal{X} \times \mathcal{Y}$. Suppose that for some examples $x_1, x_2 \in \mathcal{X}$, we have $\eta(x_1) = \alpha$ and $\eta(x_2) = \beta$, where η is the regression function for $\mathcal{D}$ as defined in class, and α and β are some numbers in $[0, 1]$. Calculate an expression using α and β for the value of

$$\mathbb{P}[Y_1 \neq Y_2 \mid X_1 = x_1 \wedge X_2 = x_2].$$

(b) Let $\mathcal{D}$ be some distribution over $\mathcal{X} \times \mathcal{Y}$ such that for all $x \in \mathcal{X}$, $\eta(x) = \alpha$ for some $\alpha \in [0,1]$. What is the Bayes-optimal error of $\mathcal{D}$ as a function of α? Prove the claim using the definition of the Bayes-optimal predictor.

(c) Suppose that $\mathcal{X} = [0,1]$ for $\mathcal{D}$ with the same properties of η as defined above, and that the marginal of $\mathcal{D}$ on $\mathcal{X}$ is uniform.. Suppose that we draw a sample $S \sim \mathcal{D}^m$ and run the 1-Nearest-Neighbor algorithm on the sample using the standard distance $\rho(x, x') = |x - x'|$. Denote the output classifier by $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$. Let $x \in \mathcal{X}$ be some example. Prove that

$$\mathbb{P}_{S \sim \mathcal{D}^m}[\exists y \in \mathcal{Y} \text{ s.t. } (x, y) \in S] = 0.$$

Use this to prove that the probability that some example (that is, an element from $\mathcal{X}$) is repeated more than once in S is 0.

(d) We will now calculate the expected error of the nearest-neighbor algorithm for $\mathcal{D}$. Let $S = ((x_1, y_1), \ldots, (x_m, y_m)) \sim \mathcal{D}^m$ be the training sample. Observe first, that once we know the examples $x_1, \ldots, x_m \in \mathcal{X}$, and the example $x \in \mathcal{X}$ for which a prediction is required, we can already tell which of the examples in the sample is the nearest neighbor of x. Whether the prediction will be correct or not now depends on the labels $y_1, \ldots, y_m$ observed in the sample and on the true label of x. Let $x, x_1, \ldots, x_m \in \mathcal{X}$ be examples which are different from each other. Use (a) to find an expression $f(\alpha)$, which depends on α, for

$$\mathbb{P}[h(X) \neq Y \mid X = x \text{ and the examples in } S \text{ are } x_1, \ldots, x_m].$$

Where the probability is over $(X, Y) \in \mathcal{D}$ and $S \sim \mathcal{D}^m$. As you will observe, $f(\alpha)$ is the same for all possible $x, x_1, \ldots, x_m$. Therefore, we can conclude that

$$\begin{aligned}\mathbb{E}_{S \sim \mathcal{D}^m}[\mathrm{err}(\hat{h}_S, \mathcal{D})] &= \mathbb{E}[\mathbb{P}[h(X) \neq Y \mid X \text{ and the examples } X_1, \ldots, X_m \text{ in } S]] \\ &= \mathbb{P}[h(X) \neq Y \mid X = x \text{ and the examples in } S \text{ are } x_1, \ldots, x_m] = f(\alpha).\end{aligned}$$

(e) Plot the expression that you got for $\mathbb{E}_{S \sim \mathcal{D}^m}[\mathrm{err}(\hat{h}_S, \mathcal{D})]$ for values of $\alpha \in [0,1]$. On the same graph, plot the Bayes-optimal error that you found in (b).

(f) Prove that the expected error that you calculated in (e) is always between the Bayes-optimal error and twice the Bayes-optimal error. Which value of α gets exactly the Bayes-optimal error? For which α does the error approach twice the Bayes-optimal error?

Question 4. Let $\mathcal{X} = [0,1]$, $\mathcal{Y} = \{0,1\}$. Consider the hypothesis class of thresholds that we defined in class,

$$\mathcal{H}_{\mathrm{th}} := \{f_a \mid a \in [0,1]\}, \text{ where } f_a(x) := \mathbb{I}[x \geq a].$$

Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Suppose that the marginal distribution of $\mathcal{D}$ on $\mathcal{X}$ is uniform on $[0,1]$, and that for $(X, Y) \sim \mathcal{D}$, for any $x \in \mathcal{X}$, $\mathbb{P}[Y = 1 \mid X = x] = \mathbb{I}[x \geq \beta]$, for some fixed $\beta \in [0,1]$. In other words, the unique Bayes-optimal predictor for $\mathcal{D}$ is f_β.

(a) Let $\epsilon \in [0,1]$. Prove that if $a \in [\beta - \epsilon, \beta + \epsilon]$, then $\mathrm{err}(f_a, \mathcal{D}) \leq \epsilon$.

(b) Suppose that we run some ERM algorithm on a sample $S \sim \mathcal{D}^m$. Prove that if there exists some $(x, y) \in S$ such that $x \in [\beta - \epsilon, \beta]$ and there exists some $(x, y) \in S$ such that $x \in [\beta, \beta + \epsilon]$, then the output classifier returned by the algorithm, denoted $\hat{h}_S$, is equal to f_a for some $a \in [\beta - \epsilon, \beta + \epsilon]$. Conclude that $\mathrm{err}(\hat{h}_S, \mathcal{D}) \leq \epsilon$.

(c) Prove that for $S \sim \mathcal{D}^m$, the probability that there does not exist a $(x, y) \in S$ such that $x \in [\beta, \beta + \epsilon]$ is $(1 - \epsilon)^m$, and that the same holds for the existence of $x \in [\beta - \epsilon, \beta]$. Let $\hat{h}_S$ be the classifier returned by the ERM algorithm as defined above. Conclude, using the above sub-questions and the union bound, that

$$\mathbb{P}_{S \sim \mathcal{D}^m}[\text{err}(\hat{h}_S, \mathcal{D}) \leq \epsilon] \geq 1 - 2(1 - \epsilon)^m.$$

(d) Based on the above inequality, what sample size is needed (that is, what should the value of m be) if we wish to guarantee that there is a probability of at least 95% that the output of the ERM algorithm has an error of at most 3% on the distribution? Explain the calculation.

Question 5. Consider a distribution over patients and the treatment that best suits them, out of two treatments: take medicine, or rest at home. For each patient, we measure their height in cm and their age in years. We allow height up to 2.5 meters and age up to 120.

(a) What are $\mathcal{X}$ and $\mathcal{Y}$ in this problem?

(b) We have a distribution $\mathcal{D}$ over patients with the following probabilities (all other values have zero probability):

height	age	best treatment	probability
160	20	rest	10%
160	40	rest	45%
160	40	medicine	5%
180	25	rest	15%
180	25	medicine	10%
180	35	medicine	15%

What is the Bayes-optimal predictor h^* for distribution $\mathcal{D}$? Write the value of the predictor $h^*(x)$ for each possible x with a non-zero probability in $\mathcal{D}$. What is the Bayes-optimal error of $\mathcal{D}$?

(c) Find a linear predictor (with a bias) that gives the Bayes-optimal error on $\mathcal{D}$. Write down w and b that you found and show that you get the required error value.

(d) Suppose we only had the height of patients. Write down the table of probabilities for the distribution over patient heights and best treatment. What is the Bayes-optimal error for this distribution?

(e) We now have a different distribution $\mathcal{D}'$ over patients with the following probabilities (all other values have zero probability):

height	age	best treatment	probability
160	20	medicine	50%
170	20	rest	25%
180	20	medicine	25%

What is the Bayes-optimal predictor for $\mathcal{D}'$? Can the Bayes-optimal error on $\mathcal{D}'$ be achieved by a linear classifier? Explain.