Introduction to learning and analysis of big data
Exercise 1

Prof. Sivan Sabato

Fall 2020/21

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 following 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 15 points (matlab code) Q.2: 21 points, Q.3: 15 points, Q.4: 18 points. Q.5: 10 points. Q6: 21 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. We will use the Euclidean distance to measure similarity between examples.

The first function, `learnknn`, 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, Xtrain, Ytrain)
```

The input parameters are:

- `k` - the number k to be used in the k-nearest-neighbor algorithm.
- `Xtrain` - a 2-D matrix of size $m \times d$ (rows $\times$ columns), where m is the sample size and d is the dimension of each example. Row i in this matrix is a vector with d coordinates which specifies 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 keeps all the information you need to apply the k-nn prediction rule to new examples. The internal format of this data structure is your choice.

The second function, `predictknn`, uses the classification rule that was outputted by `learnknn` to classify 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, Xtest)
```

The input parameters are:

- `classifier` - the classifier to be used for prediction. Only classifiers that your own code generated using `learnknn` can be used here.
- `Xtest` - a 2-D matrix of size $n \times d$, where n is the number of examples to test. 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;
>> Xtrain=[1,2;3,4;5,6];
>> Ytrain=[1;0;1];
>> classifier = learnknn(k,Xtrain,Ytrain);
>> Xtest=[10,11;3.1,4.2;2.9,4.2;5,6];
>> Ytestprediction = predictknn(classifier, 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. A matlab format file that you can use, called `mnist_all.mat`, can be found on the assignment page in the course website. For this exercise, we will use a smaller data set taken out of MNIST, that only includes images with the digits 1,3,4 and 6, 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.

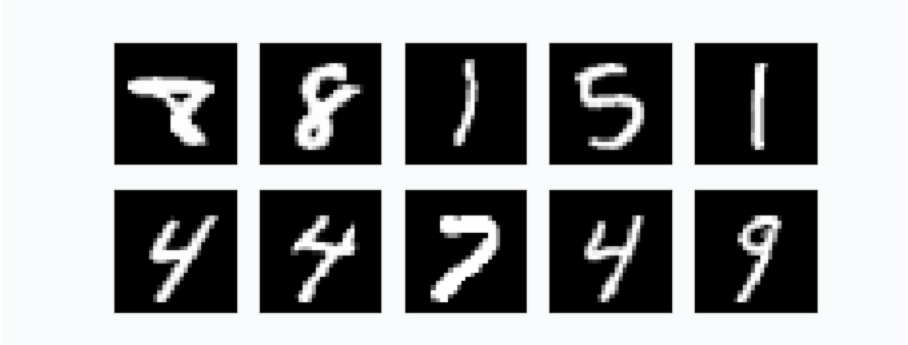

Figure 1: Some examples of images of digits from the data set MNIST

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 a test sample T, which is also a set 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 $\mathrm{err}(\hat{h}_S, T) = \frac{1}{m}\sum_{(x,y)\in T} \mathbb{I}[\hat{h}_S(x) \neq y]$. Since T is an i.i.d. sample from $\mathcal{D}$, $T \sim \mathcal{D}^m$, the error on T is a good estimate of the error of $\hat{h}_S$ on the distribution $\mathrm{err}(\hat{h}_S, \mathcal{D})$.

To load all the MNIST data to Matlab, run the following command (after making sure that the `mnist_all.mat` file is in your matlab path):

```
>> 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 function `gensmallm` which is provided in the assignment page on the course website. The function 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, each time with a different random training sample, and average the 10 error values you got. 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. Present also error bars, which show what is the minimal and maximal error value that you got for each sample size. You can use Matlab's `plot` command (or any other plotting software).

(b) Do you observe a trend in the average error reported in the graph? What is it? How would you explain it?

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

(d) Does the size of the error bars change with the sample size? What trend to you see? What do you think is the reason for this trend?

(e) Run your k-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.

(f) Now, check what happens if the labels are corrupted, as follows: repeat the experiment on the values of k as in the previous item, but this time, for every training set and test set that you feed into your functions, first select a random 20% of the examples and change their label to a different label, which you will randomly select from the three other possible labels. Plot the graph of the error as a function of k for this set of experiments.

(g) Compare the two graphs you got for the two experiments on the size of k. What is the optimal value of k for each experiment? Is there a difference between the two experiments? How do you explain it?

Question 3. Consider a distribution over rabbits and food preferences. Each rabbit likes either carrot or lettuce. For each rabbit, we measure their weight in kg and their age in months. In principle, a rabbit can live until age 48 months and weigh as much as 4kg.

(a) What should $\mathcal{X}$ and $\mathcal{Y}$ be in this problem? Define $\mathcal{X}$ to be as specific as possible given the problem description.

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

age (months)	weight (kg)	preferred food	probability
7	1	carrot	0%
7	1	lettuce	10%
7	2	carrot	0%
7	2	lettuce	50%
13	1	carrot	15%
13	1	lettuce	0%
13	2	carrot	0%
13	2	lettuce	25%

Denote the Bayes-optimal predictor for $\mathcal{D}$ by h_{bayes}. Write the value of $h_{\text{bayes}}(x)$ for each x in the support of $\mathcal{D}$. What is the Bayes-optimal error of $\mathcal{D}$?

(c) Suppose we now only measure the age of the rabbits, and so we have a distribution $\mathcal{D}'$ which is exactly like $\mathcal{D}$ except that only age is measured. Provide a table of probabilities for this distribution.

(d) Denote the Bayes-optimal predictor for $\mathcal{D}'$ by h'_{bayes}. Write the value of $h'_{\text{bayes}}(x)$ for each x in the support of $\mathcal{D}'$. What is the Bayes-optimal error of $\mathcal{D}'$?

(e) Suppose that the Memorize learning algorithm that we saw in class gets a sample $S \sim \mathcal{D}^m$ and outputs a predictor. Give a formula for the expected error of Memorize as a function of m for the distribution $\mathcal{D}$ that we defined above. Recall that this expected error is formally denoted $\mathbb{E}_{S\sim\mathcal{D}^m}[\text{err}(\hat{h}_S, \mathcal{D})]$. Calculate the value of the expected error for $m = 2$.

Question 4. In this question, you will show that even in a very simple distribution, 1-nearest-neighbor does not necessarily approach the Bayes optimal error, and can have an error that is arbitrarily close to twice the Bayes-optimal error. You should prove your claims, using basic probability and expectation properties.

Let $\mathcal{X} = \{a\}$, and let $\mathcal{Y} = \{0, 1\}$. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Recall the definition of the function η for $\mathcal{D}$: $\eta(x) := \mathbb{P}_{(X,Y)\sim\mathcal{D}}[Y = 1 \mid X = x]$. Denote $\psi := \eta(a)$.

Note that in our case, there is only one possible example, $x = a$. In the 1-nn algorithm, assume that for any test point x, if the training sample S has more than one point that is nearest to x (there's a tie), one of these points from the sample is selected uniformly at random and its label is chosen as the prediction. Let $\hat{h}_S$ be the (possibly random) prediction rule of the 1-nn algorithm for a given training sample S.

(a) Show that for a fixed S, a single number that depends on S (call it β_S) controls the probability $\mathbb{P}[\hat{h}_S(a) = 1]$. Give a formula for β_S as a function of $S = ((x_1, y_1), \ldots, (x_m, y_m))$. Note that here the probability is over the randomness of $\hat{h}_S$, since S is fixed.

(b) Calculate $\text{err}(\hat{h}_S, \mathcal{D})$ as a function of β_S and ψ.

(c) Calculate the expectation of β_S over samples, denoted $\mathbb{E}_{S\sim\mathcal{D}^m}[\beta_S]$, as a function of ψ.

(d) Calculate the expected error of the 1-nn algorithm for $\mathcal{D}$ over random samples, denoted by $\overline{\text{err}} := \mathbb{E}_{S\sim\mathcal{D}^m}[\text{err}(\hat{h}_S, \mathcal{D})]$, as a function of ψ.

(e) Calculate the error of the Bayes optimal rule for $\mathcal{D}$, denoted by $\mathrm{err}_{\text{bayes}}$, as a function of ψ.

(f) Show that for any $\epsilon > 0$, there exists a value of ψ such that the ratio $\overline{\mathrm{err}}/\mathrm{err}_{\text{bayes}}$ is larger than $2 - \epsilon$. Conclude that without knowing anything about the distribution, we cannot guarantee for a factor lower than two for the Bayes-optimal error using the 1-nn algorithm.

Question 5. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a finite domain, and let $\mathcal{Y} = \{0, 1\}$. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$. Suppose that $\mathcal{D}$ has a Bayes-error of zero and that η of $\mathcal{D}$ is c-Lipschitz with respect to the Euclidean distance.

(a) Let $S \sim \mathcal{D}^m$. Prove that for any two pairs $(x_1, y_1), (x_2, y_2) \in S$, if $y_1 \neq y_2$ then $\|x_1 - x_2\| \geq 1/c$.

(b) Let $\mathcal{B}$ be a set of balls of radius $1/(3c)$ that *cover* the space of points $\mathcal{X}$, meaning that every point from $\mathcal{X}$ is in at least one ball in $\mathcal{B}$. Let $S \sim \mathcal{D}^m$ such that for every ball $B \in \mathcal{B}$, there is some pair $(x, y) \in S$ which satisfies $x \in B$. Prove that under this assumption, and the other assumptions on $\mathcal{D}$ given above, $\mathrm{err}(f_S^{nn}, \mathcal{D}) = 0$, where f_S^{nn} is the 1-nearest-neighbor function defined in class.

Question 6. Let $\mathcal{X} = [0, 1]$ and $\mathcal{Y} = \{0, 1\}$. For an even integer $k \geq 2$, define the hypothesis class

$$\mathcal{H}_k = \{f_{a_1,\ldots,a_k} \mid \forall i \in \{1, \ldots, k\}, a_i \in [0, 1], \text{ and } a_1 \leq a_2 \leq \ldots \leq a_3\},$$

where the function f_a is defined as follows:

$$\forall x \in [0, 1], \quad f_{a_1,a_2,\ldots,a_k}(x) := \mathbb{I}\big[\exists \text{ an odd integer } i \in \{1, \ldots, k-1\} \text{ such that } x \in [a_i, a_{i+1}]\big].$$

Suppose that $\mathcal{D}$ is a distribution over $\mathcal{X} \times \mathcal{Y}$ such that the probability that $X = x$ for $(X, Y) \sim \mathcal{D}$ is uniform over $[0, 1]$.

(a) Among the hypothesis classes $\mathcal{H}_2, \mathcal{H}_4, \mathcal{H}_6$, which guarantees a minimal approximation error on $\mathcal{D}$? Prove your claim.

(b) Fix an even integer $k \geq 2$. Prove that for any $m \leq k$, with probability 1 over $S \sim \mathcal{D}^m$, the hypothesis returned by the 1-nearest-neighbor algorithm satisfies the condition of an ERM algorithm for the hypothesis class $\mathcal{H}_k$.

(c) Fix an even integer $k \geq 2$. Prove that for $m = k+1$, there exists a sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ with no duplicate examples (that is, for all i, j such that $i \neq j$, we have $x_i \neq x_j$), such that the output hypothesis of the 1-nearest-neighbor algorithm if this sample is provided as input does **not** satisfy the condition of an ERM algorithm for the hypothesis class $\mathcal{H}_k$.

(Note: you only need to prove the **existence** of such an S, we are not asking about how likely it is to get such an S.)