Introduction to machine learning
Exercise 1

Fall 2024/25

Submission guidelines, please read and follow carefully:

- The exercise **must** be submitted in pairs.
- Submit via Moodle.
- The submission should include two separate files:
  1. A pdf file that includes your answers to all the questions.
  2. The code files for the python question. You must submit a copy of the shell python file provided for this exercise in Moodle, with the required functions implemented by you. **Do not change the name of this file!** In addition, you can also submit other code files that are used by the shell file.
- Your python code should follow the course python guidelines. See the Moodle website for guidelines and python resources.
- Before you submit, **make sure that your code works in the course environment**, as explained in the guidelines. Specifically, **make sure that the test `simple_test` provided in the shell file works**.
- You may only use python modules that are explicitly allowed in the exercise or in the guidelines. If you are wondering whether you can use another module, ask a question in the exercise forum. No module containing machine learning algorithms will be allowed.
- For questions, use the exercise forum, or if they are not of public interest, send an email to the course staff at intromlbgu25@gmail.com
- Grading: Q.1 (python code): 10 points, Q.2: 20 points, Q.3: 15 points, Q.4: 19 points, Q.5: 18 points, Q.6: 18 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 the similarity between examples.

The shell Python file `nearest_neighbour.py` is provided for this exercise in Moodle. It contains empty implementations of the functions required below. You should implement them and submit according to the submission instructions.

The first function, `learnknn`, creates the classification rule. Implement the function in the file which can be found on the assignment page in the course's website. The signature of the function should be:

```
def learnknn(k, x_train, y_train)
```

The input parameters are:

- `k` - the number k to be used in the k-nearest-neighbor algorithm.
- `x_train` - 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.
- `y_train` - a column vector of length m (that is, a matrix of size $m \times 1$). The i-th 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 also be implemented in the `nearest_neighbour.py` file. The signature of the function should be:

```
def predictknn(classifier, x_test)
```

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.
- `x_test` - 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 `y_testprediction`, which 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 `x_test`.

Important notes:

- You may assume all the input parameters are legal.
- The Euclidean distance between two vectors z_1, z_2 of the same length can be calculated using `numpy.linalg.norm(z1-z2)` or `scipy.spatial.distance.euclidean(z1, z2)`.

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

```
>>> from nearest_neighbour import learnknn, predictknn
>>> k = 1
>>> x_train = np.array([[1,2], [3,4], [5,6]])
>>> y_train = np.array([1, 0, 1])
>>> classifier = learnknn(k, x_train, y_train)
```

```
>>> x_test = np.array([[10,11], [3.1,4.2], [2.9,4.2], [5,6]])
>>> y_testprediction = predictknn(classifier, x_test)
>>> y_testprediction
[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 dataset of images, called MNIST, is free on the web. It includes 70,000 images with the digits 0-9. A numpy format file that you can use, called `mnist_all.npz`, can be found on the assignment page in the course website. For this exercise, we will use a smaller dataset taken out of MNIST, that only includes images with the digits 2, 3, 5, and 6, so that there are only four possible labels. Each image in MNIST has 28×28 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 $X \in \mathbb{R}^{784}$: every example is described by a 784-coordinate vector.

Figure 1: Some examples of images of digits from the dataset MNIST

The images in MNIST are split into 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:

$$\text{err}(\hat{h}_S, T) = \frac{1}{m} \sum_{(x,y)\in T} \mathbb{I}[\hat{h}_S(x) \neq y].$$

Since T is an i.i.d. sample from D, $T \sim D^m$, the error on T is a good estimate of the error of $\hat{h}_S$ on the distribution $\text{err}(\hat{h}_S, D)$.

To load all the MNIST data, run the following command (after making sure that the `mnist_all.npz` file is in your working directory):

```
>>> data = np.load('mnist_all.npz')
```

This command will load the MNIST data to a Python dictionary called `data`. You can access the data by referencing the dictionary: `data['train0']`, `data['train1']`, ..., `data['train9']` `data['test0']`, `data['test1']`, ..., `data['test9']`.

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 shell file `nearest_neighbour.py`. The function is used as follows:

```
>>> (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.

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:

```
>>> np.mean(y_test != y_testpredict)
```

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

(d) Run your k-nn implementation with a training sample size of 200, 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. What is the optimal value of k you got? Explain the trend of the graph.

(e) 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 30% 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.

(f) 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 small trees and their types. Each tree is either an apple tree or an orange tree. For each tree, we measure its height (in centimeters) and number of leaves. A tree can grow up to 100 cm in height, and the number of leaves can range from 1 to 50. The set of all possible inputs is $\mathcal{X} = \{1, 2, ..., 100\} \times \{1, .., 50\}$ and the set of all possible labels is $\mathcal{Y} = \{\text{Apple}, \text{Orange}\}$

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

Height (cm)	Leaves	Tree type	Probability
20	5	Apple	40%
20	5	Orange	10%
50	10	Apple	20%
50	15	Orange	10%
80	20	Orange	20%

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 $\mathcal{D}$.

(b) Calculate the Bayes-optimal error of $\mathcal{D}$.

(c) Let $\mathcal{H}$ be a hypothesis class that includes only constant functions, that is: functions that assign all examples to the same label. What is the approximation error of $\mathcal{H}$ for the distribution $\mathcal{D}$ defined above?

(d) Consider the following distribution $\mathcal{D}''$:

Height (cm)	Leaves	Tree type	Probability
20	5	Apple	40%
50	10	Apple	25%
50	15	Orange	15%
80	20	Orange	20%

Calculate the value of the expected error of the Memorize algorithm for sample size $m = 2$ using the formula we learned in class. Explain why you are allowed to use this formula for $\mathcal{D}''$ but not for the distribution $\mathcal{D}$ defined above.

Question 4. Let $\mathcal{X} = [0, 1]$ and $\mathcal{Y} = \{0, 1\}$. For an even integer $k \geq 2$, define the hypothesis class

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

where the function f_a is defined as follows:

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

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 $\mathcal{H}_k$. For simplicity, let the training sample be represented as $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and you can assume $x_1 < x_2 < .. < x_m$.

(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.

Question 5. Consider the following *union of two segments function* for an input $x \in \mathbb{R}$:

$$f_{a,b,c,d}(x) = \mathbb{I}\big[a \leq x \leq b \text{ or } c \leq x \leq d\big]$$

where $a, b, c, d \in \mathbb{R}$, $a < b < c < d$, are four thresholds that define two segment in $\mathbb{R}$. $\mathbb{I}\big[\text{condition}\big]$ is an indicator function that returns 1 if the condition is satisfied, and 0 otherwise.

We define the hypothesis class

$$\mathcal{H} = \{f_{a,b,c,d} \mid a, b, c, d \in \mathbb{R},\ a < b < c < d\}$$

and consider it for finding a predictor for a binary classification problem where the labels are $\mathcal{Y} = \{0, 1\}$.

What is the VC dimension of $\mathcal{H}$? Prove your answer.

Question 6. Consider a classification problem with an input domain $\mathcal{X}$, a label domain $\mathcal{Y}$, and a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Data scientist A has a sample $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ of m independent and identically distributed (i.i.d.) examples drawn from the distribution $\mathcal{D}$. Data scientist B has a sample $S' = \{(x'_1, y'_1), \cdots, (x'_n, y'_n)\}$ of n i.i.d. examples drawn from the distribution $\mathcal{D}$. The examples in S and S' are statistically independent. The distribution $\mathcal{D}$ is unknown to the two data scientists.

Consider a finite hypothesis class $\mathcal{H}$ for which $\mathcal{D}$ is realizable.

The two data scientists would like to collaborate in order to learn a predictor $\widehat{h} : \mathcal{X} \to \mathcal{Y}$ for the unknown distribution. However, they do not want to share their examples with each other, and therefore they use the following iterative extension of the $\text{ERM}_{\mathcal{H}}$ principle:

0. Initialize: $k = 1$, $\mathcal{W} = \{\}$
1. Data scientist A proposes a predictor $\hat{h}_S^{(k)}$ that corresponds to $\text{ERM}_{\mathcal{H}\setminus\mathcal{W}}$ on the sample S, where the hypothesis class is $\mathcal{H}$ without the predictors in $\mathcal{W}$ that were chosen in previous iterations:

   $$\hat{h}_S^{(k)} \in \operatorname*{argmin}_{h \in \mathcal{H}\setminus\mathcal{W}} \widehat{\text{err}}(h, S)$$

   where $\widehat{\text{err}}(h, S) := \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[h(x_i) \neq y_i]$ is the empirical error of a predictor $h \in \mathcal{H}$ on the sample S. The indicator function $\mathbb{I}\big[\text{condition}\big]$ returns 1 if the condition is satisfied, and 0 otherwise.
2. Data scientist B gets the predictor $\hat{h}_S$ from data scientist A and compute the empirical error of $\hat{h}_S^{(k)}$ on the sample S':

   $$\widehat{\text{err}}(\hat{h}_S^{(k)}, S') := \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[\hat{h}_S^{(k)}(x'_i) \neq y'_i]$$

3. If $\widehat{\text{err}}(\hat{h}_S^{(k)}, S') = 0$:
   - the two data scientists agree to set $\hat{h}_{S,S'} := \hat{h}_S^{(k)}$ and conclude the learning process with $\hat{h}_{S,S'}$ as their chosen predictor.
4. Else (i.e., if $\widehat{\text{err}}(\hat{h}_S^{(k)}, S') > 0$):
   - the data scientists decide not to use $\hat{h}_S^{(k)}$ and to repeat another iteration of the process, i.e.,
     set $\mathcal{W} \leftarrow \mathcal{W} \cup \{\hat{h}_S^{(k)}\}$
     set $k \leftarrow k + 1$
     return to the step 1 of the process.

Assume that $\mathcal{H}$ makes the above iterative process to converge in a reasonable time (number of iterations) and that a predictor $\hat{h}_{S,S'}$ is always returned.

Formulate a condition on the size m of the sample S for which the above collaborative process provides a predictor $\hat{h}_{S,S'}$ such that

$$\mathbb{P}_{S\sim\mathcal{D}^m, S'\sim\mathcal{D}^n}[\mathrm{err}(\hat{h}_{S,S'}, \mathcal{D}) \leq \epsilon] \geq 1 - \delta$$

for $\epsilon, \delta \in (0, 1)$.

Note: The formulated condition on m should express the effect of a given S' according to its fixed size n.

Your answer should include a mathematical proof of the formulated condition.