FACULTY OF DATA
AND DECISION SCIENCES
MAKING SENSE OF IT

הטכניון
מכון טכנולוגי
לישראל

למידה חישובית 1 (096411)

אביב תשפ"ד 2024

תרגיל בית 1

תאריך אחרון להגשה: 01/07/2024 בשעה 23:55

Instructions - Read before you start the exercise

- **Submission is in pairs -** exceptions must be specifically approved by the course staff. Submission not in pairs, which hasn't been approved, **will be graded 0 automatically**.
- You are to submit a **zip file** with the name **HW1_ID1_ID2.zip,** where ID1 and ID2 are your student ids.
- The zip should contain **two** files:
  - A *HW1_ID1_ID2.py* file containing the code for Question 2.
  - A **single** *HW1_ID1_ID2.pdf* with a (detailed) solution of all of the written exercises (Questions 1 + 3).
- Submissions not in pairs please follow this convention:
  - *HW1_ID1.zip*
- One submission per team – only one team member should submit.
- For questions with plotting (Question 1) you can submit a **static** jupyter / google colab notebook (static == pdf format **only**). Written answers can be written inside text cells. You must **combine all the written questions to a single PDF file** (e.g. *question_1.pdf*, *question*_3*.pdf,...* etc is **not allowed**).
- All code (inside notebooks and .py files) must be clear and concise (documented, using meaningful variable names, etc.)
- Every plot must contain at least the following: axis labels, units and legend.
- We will run your code using:
  - python 3.8.8
  - numpy 1.21.2
  - pandas 1.3.4
- Using other versions of python or python libraries is not recommended. It is at your own risk if the code fails to run due to version compatibility issues. We recommend using a clean anaconda virtual environment with the exact python and python modules + versions above installed.
- **No cheating** - if you are to copy your answers from other students and/or online references, you risk getting 0 for the submission and a disciplinary board. You may consult each other, but you are expected to write your own answers.
- Please use the HW forum for questions. Your questions could be helpful for other classmates. Generally, we will not answer questions sent by email to the course staff (unless there is a good reason to).

Good luck

And may the force be with you

Question 1

Let

$$\mu_1 = [-1,1]^T, \Sigma_1 = I_{2\,x\,2}$$

$$\mu_2 = [-2.5,2.5]^T, \Sigma_2 = I_{2\,x\,2}$$

$$\mu_2 = [-4.5,4.5]^T, \Sigma_2 = I_{2\,x\,2}$$

where I is the identity matrix.

1. Create a total of 700 synthetic samples from the 3 Gaussian distributions according to the above parameters. Assume that the probability of sampling from each gaussian is uniform (=1/3).
2. Plot all the samples on a scatter plot. Each Gaussian should be plotted in a different color.
3. Generate 300 additional samples as a test set and plot them on an additional scatter plot.
4. Train a k-NN classifier using k=1 with Euclidean distance on the 700 samples (train set) and evaluate it on the 300 examples (test set).

   We define the classification test error rate as:

   $$Classification\ Error\ Rate = \frac{1}{m}\sum_{i=1}^{m} 1[h(x_i) \neq y_i]$$

   where m is the number of data points and $h(x_i)$ is the model's prediction on x_i .

   What is the classification error rate of your 1-NN model on the **train** and **test** sets? Is there a gap between the two error rates? If so, explain why.
5. Repeat step 4 for $k = 1,2,\ldots,20$ and plot the train and test errors as a function of k in a single line plot (with two lines, one for train and one for test). Does the test error decrease with k? should the test error *always* decrease with k? if yes, explain why, if not, provide a counterexample.
6. Let m_{train} be the number of training examples. For $m_{train_i} \in \{10, 15, 20, \ldots, 40\}$, Fix $k = 10$ and $m_{test} = 100$ and sample **new** m_{train_i} samples (in the same way as you sampled them in step 1). Plot the train and test errors a function of m_{train_i}. How do you expect this graph to behave? Does it match your expectation?
7. Repeat step 6 for 5-10 trials (no need to show the plots for your trials). Do the line plots change between trials? Explain. Do they meet your expectations (from step 6) at every trial?
8. Suggest a variation of a k-NN classifier where the distances of the neighbors are taken into account in the prediction (No need to implement anything in code).

Question 1 – Important notes:

1. You can use python libraries such as numpy, pandas and sklearn. You do not have to implement your own KNN algorithm and you can import the `KNeighborsClassifier` class from `sklearn.neighbors` for this question.
2. Describe every sampling procedure and error calculation using python code.

Question 2

In this question you will implement a multi-class k-NN algorithm in Python.

Attached to this exercise is a file called ***HW1_ID1_ID2.py***. In this file you will find a class called ***KnnClassifier*** which contains two methods you need to implement: ***fit*** and ***predict***.

- ***fit*** accepts a labeled dataset $S_{train} = \{(x_i, y_i)\}^{m_{train}}{}_{i=1}$ $where\ y_i \in \{0,1,2,\ldots L-1\}$and uses it to train a knn model.
- ***predict*** accepts an un-labeled dataset $S_{test} = \{x_i\}^{m_{test}}{}_{i=1}$ and predict its y labels.

It is guaranteed that after a ***KnnClassifier*** object is instantiated, we will call ***predict*** only after we have called ***fit***. Please refer to the methods' documentation inside ***HW1_ID1_ID2.py*** for detailed input/output specifications.

- Note that it is possible that $m_{test} \neq m_{train}$.
- It is guaranteed that the csv file **will not** contain a header row (column names).
- It is guaranteed that the **last column** in any given **training** dataset (E.g. given to ***fit***) will contain numeric $(0,1,\ldots,L-1)$ class labels.
- Note that the dataset can be multiclass (e.g. the **iris.csv** dataset attached) and your implementation should support an arbitrary number of classes.

Your ***HW1_ID1_ID2.py*** file should be able to run from the **command line** like so:

python HW1_ID1_ID2.py <path_to_csv> <k> <p>

- *<path_to_csv>* - is a path for a local csv file that contains the dataset. For example, ***iris.csv*** which is attached to this exercise.
- *<k>* - number of nearest neighbors to train the knn model with.
- *<p>* - p parameter for Minkowski distance calculation. You can assume that $p \geq 1$.

For example, calling the script from the command line (when ***iris.csv*** is located <u>in the same folder</u> as ***HW1_ID1_ID2.py***) would look like:

python HW1_ID1_ID2.py iris.csv 5 2

This will run (fit and predict) a 5-nn model with Euclidean distance ($p = 2$) on the iris.csv dataset.

<u>Tie breaking</u>

- In case of a tie (e.g. a test point x in a 2-NN binary setting with 1 neighbor from class A and 1 neighbor from class B), your algorithm should break the tie by the **distance** to the nearest neighbor. For the example above, if the (single) nearest neighbor belongs to A, then $h(x) = y_A$.
- If $d(x, A) = d(x, B)$, you should break the tie by the lexicographic ordering of A and B's **labels**. E.g. if $y_A = 1$ and $y_B = 0$, then $h(x) = 0$.
- Note that tie breaking can occur in a multiclass setting.
- If the K and the $K+1$ neighbors have the same distance to a test point x, we consider the neighbor with the lower lexicographic label to be in the K neighborhood of x. E.g. if A and B are the K and $K+1$ neighbors of x (respectively), $d(x, A) = d(x, B)$, $y_A = 1$ and $y_B = 0$, then only B will be in the K neighborhood of x.

- Tie breaking considers only classes tied on the majority vote. E.g. if the majority vote is tied between class A and B yet the closest point to x is from class C, we will predict $h(x) = y_A$ or $h(x) = y_B$ only.

Question 2 – Important notes:

1. You must place your student IDs in line 6.
2. You are only allowed to import the following python modules:
   - *numpy*
   - *pandas*
   - native python modules (*os*, *argparse*, etc. Usually libraries that come with a clean python environment. If you are not sure if a library is native python or not - please use the HW forum)

   You are **definitely not allowed** to import modules such as *sklearn* (or any of its submodules).
3. You can add and modify class methods to the ***KnnClassifier*** class as you wish. However, you are not allowed to change the signature of ***__init__***, ***fit*** and ***predict***. Also, your ***predict*** method must return a numpy array. Please refer to the documentation.
4. Your ***KnnClassifier*** class will be evaluated using a different script, so editing the main block in HW1_ID1_ID2.py won't affect your code's performance. Our evaluation will be similar to the main block in HW1_ID1_ID2.py, but it is only there for your own use and self-evaluation.
5. Your code should have a reasonable runtime (reasonable == under 1 minute for a training set of 100 samples).

- **Failing to follow instructions 1-5 will result in a grade of 0 on this question.**
- **A *HW1_ID1_ID2.py* script that raises an exception and/or fails to execute from the command line for any reason will also result in a grade of 0 on this question.**

Evaluation

We will test the correctness of your implementation against multiple datasets and (k, p) configurations. Since k-NN is a deterministic algorithm, for a given dataset and (k, p) parameters there is only one correct solution. You may use the attached ***iris.csv*** dataset and compare your implementation to a sklearn implementation to test your code's correctness.

Bonus - 7 points:

If your ***KnnClassifier*** class runs properly **with NO LOOPS** (Specifically, you don't use any ***for, while*** statements in your ***KnnClassifier*** implementation).

Question 3

1. Given a training set $S = \{(x_i, y_i)\}_{i=1}^{m}$ where $y_i \in \{1, \dots, K\}$, In the multiclass formulation of logistic regression, we assume the following probability mass function (PMF):

$$P_w(Y_i = k|x_i) = \frac{e^{w_k{}^T x_i}}{\sum_{j=1}^{K} e^{w_j{}^T x_i}} =: p_k$$

   Recall that for the binary case ($y_i \in \{0,1\}$) we defined:

$$P_w(Y_i = 1|x_i) = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} =: p$$

   Show that the binary case is a private case of having k classes.

   That is, show that for every $w_1, w_2 \in \mathbb{R}^d$ there exists $w \in \mathbb{R}^d$ such that $p_1 = p, p_2 = 1 - p$.
   In addition, show that the converse is true, that is:

$$\forall w \in \mathbb{R}^d, \exists w_1, w_2 \in \mathbb{R}^d : p = p_1, 1 - p = p_2$$

2. A maximum likelihood estimator for a **multiclass** logistic regression classifier ($y_i \in \{1,2, \dots, K\}$) is obtained by the solving the following optimization problem:

$$w^* = argmax_w \{\sum_{i=1}^{m} log\ (P_w(Y_i = y_i\ |x_i))\}$$

   Derive the moment matching constraint that solves the optimization problem given the formulation from the previous section. Explain.

3. A multiclass logistic regression classifier was fit in a task with $x_i \in R^2$ and $y_i \in \{0,1,2\}$. The obtained parameters were:

$$w_1 = [8, -2.5, 2]$$
$$w_2 = [2, 0.5, -1.5]$$
$$w_3 = [-10, 2, -0.5]$$

   a. Explain why there are three weight vectors w_1, w_2, w_3.
   b. Explain why $x \in R^2$ yet $w_i \in R^3$.
   c. The following three **new** observations were obtained:

i **(observation number)**	$x_{i,1}$	$x_{i,2}$
1	1	8
2	6	-2
3	12	4

   For every observation i calculate $p_i = P_w(Y_i = k\ |x_i)$ for every class k, and then classify each observation with its predicted label based on the probabilities.