Machine Learning and Intelligent Systems

Project 1: Understanding k-NNs

kNN is considered a non-parametric method given that it makes few assumptions about the form of the data distribution. This approach is *memory-based* as it requires no model to be fit.

Nearest-neighbor methods use the observations from the training set closest in input space to x to form $\hat{y}$. It assumes that if a sample's features are similar to the ones of points of one particular class then it belongs to that class. These points are known as nearest neighbors.

The specific case where k=1 is denoted the nearest neighbor algorithm. Here $\hat{y}$ is assigned the value y_l of the closest point x_l to x in the training data. This corresponds to a *Voronoi tessellation* of the training data.

Objectives

By executing this project, you will be able to

1. Have a better understanding of the mechanisms behind the k nearest neighbor algorithm
2. Be aware of the problems associated with the curse of dimensionality.
3. Familiarize with the process of training, validation and testing
4. Gain proficiency in the use of python, Jupyter, libraries used in machine learning (pandas, numpy, matplotlib, etc) and programming , in general.

Part I – Implementing a kNN from scratch:

Task 1. Your first task will be to code the k NN learning algorithm. In attachment you have been provided with the file `knn.py`. This contains the skeleton of a Python class that you will need to complete with the necessary code. In particular, you will need to implement the following functions:

1. `train` – with the necessary steps to train a kNN
2. `predict` – with the necessary steps to predict the labels y of a group of samples X.
3. `minkowski_dist` – which implements the Minkowski distance as seen during the first lecture.

Make sure you debug your code, to verify that it works accordingly. You may consider comparing it against scikit-learns implementation for validation.

Task 2. You have also been provided with two files: `training.csv` and `validation.csv`. Use them to do hyper-parameter tuning. In other words, use the data to choose the best k, according to your data.

The `experiment.ipynb` notebook contains some pre-coded routines that may help you to quickly plot your data (as long as it is 2D) and display it.

Part II – The curse of dimensionality:

You will dig into the concept of the curse of dimensionality. To get familiar with it, first read section 2.5 from the book The Elements of Statistical Learning. After reading, choose one of the two tasks below.

Task 3. Suppose you have a D dimension hypercube with all sides of length 1 in the Cartesian map, i.e. $[0,1]^D$. You sample the training data **uniformly** from this hypercube, i.e. $\forall i$, $\mathbf{x}_i \in [0,1]^D$. Assume k=10 to define the label of a test point.

Let l be the edge length of the smallest hypercube that contains all k-nearest neighbor of a test point. What is the approximate volume of the hypercube? What is the length of l in terms of k, D (the dimensions) and N the number of training points?

Using the expression you found for l in the previous question, estimate the size of l as a fuction of D, while assuming a training set of size N=1000. Plot it in `experiment.ipynb`. What consequences this may have for the k nearest neighbor algorithm?

Task 4. Solve exercises 2.3 and 2.4 from The Elements of Statistical Learning.

Report:

You need to prepare a 1-page report explaining how your model was trained, the results obtained in the validation set and the strategy you used to choose k. Then, you need to discuss how the curse of dimensionality affects kNNs and present/discuss the task you chose for Part II as a way to support your claims.

Deliverables:

Upload a zip file named <group-name>.zip containing the report, the knn.py file and any instructions required to run it.

Evaluation:

Criteria	Score
Your code runs and works as expected on any given dataset	6
Your model achieves a good accuracy on a test set (not seen at training), i.e. above 80%	2
Among all submissions, your model achieves the highest accuracy	1
Among all submissions, your model is the fastest during testing	2
Your answers to part II are correct (task 3 or task 4)	4
Your report	5

Important:

- Failing to submit a report leads to a mark of zero (0).
- If ChatGPT is used, failing to report it and explaining its use leads to a mark of zero (0).
- A group will be chosen at random to present their solution during the lecture. Failing to justify the submitted solution leads to a mark of zero (0).