CS339: Artificial Intelligence
Spring 2021
Project 1: K-Nearest Neighbor Algorithm

Overview:

The purpose of this project is to learn the k-nearest neighbors algorithm, to implement the algorithm in code, and to apply the algorithm to a moderately easy dataset. This project will be an individual project that you should finish on your own.

Implementation:

You are to code the k-nn algorithm in a python jupyter notebook. You are to write the k-nn algorithm from scratch; you may not use a built-in algorithm in an AI toolbox, nor may you borrow or copy code from another source. The course textbook has a good python algorithm that you may follow if you wish.

You will have to decide how to store the data. You may use python lists, numpy arrays, pandas dataframes, or whatever format you wish. numpy arrays have some built-in support that makes the coding slightly easier and it is the basis for the algorithm in our course textbook.

You will have to appropriately pre-process the data. We will have in-class discussions about pre-processing where algorithms use distance metrics between data points. You should follow and implement these ideas to have correct operation of your algorithm.

Dataset:

We will use the famous Fisher Iris dataset. The raw datafile is provided as a link on this assignment (in notebowl). I suggest you convert the categories to numeric values to make the coding easier.

The iris data is in four dimensions. A description of the data is provided with the assignment so that you can understand the feature in each dimension.

You should use 60% of the data as a training dataset and 40% as a testing dataset. Follow appropriate guidelines for selecting the sample points for training and test sets.

Metrics and Graphics:

You will want to measure your algorithm's performance in correctly predicting the category for the testing points. Besides percent correct, how else might you represent accuracy to your reader? How consistent is the algorithm if you make different cuts of the data for training and testing?

The data is in four dimensions. You might look into a plot matrix which can show 4d data as a matrix of 2d plots. How might you graphically display the different categories? How might you visually separate the testing and training points? How might you indicate those testing points which have been correctly classified from those which have not? Are there particular samples in the dataset which are problematic? If so, why?

Report:

This will be the only project this semester where we will not write a paper. We will instead submit our project entirely using a jupyter notebook. The notebook will contain the code, the narrative of the project, and the graphical results.

You want your notebook to tell a data story. Think very carefully about how to format the notebook. Think of the notebook as a research paper which happens to contain code and not as a program which happens to have more than the usual number of comments. You will want to weave together code segments with markup cells that alternate between computation, analysis and your data story narrative. Your notebook should be well organized, should be easy to understand at multiple levels of sophistication, should have compelling graphics, and should have a clear data story that flows from the start of the notebook to the end. I suggest that you put large "service routines" in an "appendix" at the end of the notebook so as not to interrupt the important aspects of the data story with unnecessarily large code blocks.

Plan your project timeline accordingly. I think that about 40% of your time will be developing the algorithm, about 30% for graphics and results analysis, and 30% for writing and formatting the notebook. Do not mistake the project for mostly coding with a few minutes extra for graphics and writing.