

Name: _____

Spire ID: _____

CMPSCI 389 Homework 1

Spring 2022

Assigned: Feb 1, 2022; Due: Feb 10, 2022 @ 4:00pm Eastern

Instructions: This assignment has two parts. In the first part you will provide written answers to questions. In the second part you will implement weighted k -Nearest Neighbors and will apply it to predict student GPAs at university from information available when they applied. Both parts of this assignment should be completed independently (that is, you should not work with others). The written portion of this assignment should be handed in at the start of class on the day the assignment is due. The programming portion will be submitted using Moodle. An auto-grader will be used to grade your code. As such, your program must meet the requirements specified below.

1 Written (50 Points)

1. (5 pts) What is the difference between parametric and nonparametric machine learning algorithms? Give examples of parametric and nonparametric methods.

2. (5 pts) What is the difference between a model parameter and a hyperparameter?

3. (5 pts) Give an example of an algorithm (perhaps hypothetical) that falls within the field of *artificial intelligence* (AI) but not the field of *machine learning* (ML). Explain why this algorithm is an AI algorithm but not an ML algorithm.
4. (5 pts) Consider two vectors, u and v , that describe two different people. Each vector has three elements: the persons age (in years), yearly income (in dollars), and whether or not they like boats (0 = no, 1 = yes). Let $u = [27, 37200, 1]$ and $v = [42, 1250000, 1]$. What is the Euclidean distance between u and v ?
5. (10 pts) Consider using k -nearest neighbors to predict whether someone will buy a boat, using the features described in the previous problem. Explain why using Euclidean distance likely would not work well.

6. (10 pts) Consider using k -nearest neighbors to predict whether someone will buy a boat, using the same features as before. Describe how weighted k -nearest neighbors could be modified to be more effective. Hint: You might propose a specific new way of measuring distances, or you might preprocess the data in some way.

7. (10 pts) Compute the following derivatives (this is a refresher, as we will use these concepts in coming lectures). Recall that \circ denotes function composition. That is, $(f \circ g)(x) = f(g(x))$.

- Let $f(x) = 3e^x$ and $g(x) = \sin(x)$. What is $\frac{d(f \circ g)(x)}{dx}$?

- Let $f(x) = x^2$ and $g(y) = 3y$. What is $\frac{d(f \circ g)(z)}{dz}$?¹

¹This symbols x , y , and z have been carefully checked in this problem. They are not typos.

- Let v and a be two vectors, each of length n . Let $f(a, v) = \sum_{i=1}^n \left(i - \sum_{j=1}^n a_j v_i \right)^2$. If $n > 3$, what is $\frac{\partial f(a, v)}{\partial v_3}$?

- Let $f(x) = x^2$, $g(x) = \sin(x)$, and $h(x) = e^x$. What is $\frac{d(f \circ (g \circ h))}{dx}$?

- Let $f(x) = \frac{\sin(x)}{e^{\cos(x)}}$. What is $\frac{df(x)}{dx}$?

2 Programming (50 Points)

For this assignment you will implement the weighted k -nearest neighbor algorithm described in class and in the course notes.

Language: For this assignment you must use Python 3, and you may use any of the Python 3 default libraries (e.g., math). The only additional library (that requires a pip install command) your code may use is the `numpy` library.

Submitting: Your code should be submitted on Moodle, and should be in a single file named `main.py`.

File Descriptions: Your program will be placed in the same directory as the following three input files (these should be available for download as a .zip from the course page): `train.csv`, `test.csv`, and `hyper.csv`:

- `train.csv`: This file contains the *training data*—the data that weighted k -nearest neighbors should use to make predictions.
- `test.csv`: This file contains the *testing data*. Your program will first try to predict the labels for these points, outputting the predictions to a new file, `out.csv`. Your program should then use the labels from `test.csv` to determine how good its predictions were. Precisely how this is done using the *root mean squared error* (RMSE) is described below.
- `hyper.csv`: This file contains the hyperparameters for weighted k -nearest neighbors.

Both `train.csv` and `test.csv` contain data points (inputs and labels), with one point per row. **Notice:** Your code will be tested on other files which may have different numbers of rows and columns, so do not hard-code the number of rows and columns into your program! In all cases, the last column is the label while all other columns are the features. For example, the provided `train.csv` and `test.csv` correspond to the GPA prediction dataset described in class, which has nine features. As a result, each file has ten columns: The first nine contain features and the tenth contains the corresponding label. Lastly, the file `hyper.csv` contains two rows. The first row contains the hyperparameter k and the second contains the hyperparameter σ .

Program Behavior: For each row in `test.csv`, your algorithm should predict the corresponding label using the weighted k -nearest neighbor algorithm (using the hyperparameters specified in `hyper.csv`) and the training data in `train.csv`. These predictions should be printed to a new file, `out.csv`. This output file should contain one number per line—the prediction for the data point on the corresponding line of `test.csv`. For example, if the predictions are stored in a numpy ndarray named `yHat` and you included numpy and os with:

```
import numpy as np
import os
```

then these values can be printed with the line:

```
np.savetxt(os.path.join(os.path.dirname(__file__), 'out.csv'), yHat, delimiter=',')
```

After printing the predictions to `out.csv`, your program will compute an estimate of how good its predictions were. Let y_i and \hat{y}_i be the actual label and the prediction for the i^{th} datapoint in `test.csv`. The *root mean squared error* (RMSE) of these predictions is given by the equation:

$$\text{RMSE} = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2},$$

where n_{test} is the number of points in `test.csv`. Your program should print the RMSE of its predictions to the console. The values printed to the console will *not* be checked by the autograder and so the format is not important. Below you will be asked to report the RMSE value.

Tips: Notice that the provided training and testing datasets are quite large, and so your code might take a long time to run. We therefore recommend that you create new smaller and simpler datasets to debug and test your program, only running your program on the full dataset once when your implementation is complete and debugged.

To help you debug your program, in the `test case` subdirectory of the `.zip` version of the assignment you will find example input and output files.

To help you get started coding, the code below loads `train.csv` into two variables, `Xtrain` and `Ytrain`:

```
import numpy as np      # We will use np.ndarray as our array structure
import os              # We will use os.path to get the path to the current directory

def load_data(file_path: str)->tuple[np.ndarray, np.ndarray]:
    '''
    This function loads and parses text file separated by a ',' character and
    returns a data set as two arrays, an array of features, and an array of labels.

    Parameters
    -----
    file_path : str
        path to the file containing the data set

    Returns
    -----
    features : ndarray
        2D array of shape (n,m) containing features for the data set
    labels : ndarray
        1D array of shape (n,) containing labels for the data set
    '''
    curDirectory = os.path.dirname(__file__)      # Get the directory of this file
    fullPath = os.path.join(curDirectory, file_path) # Get the full path to the data file
    D = np.genfromtxt(fullPath, delimiter=",")    # Load data from the file using numpy "genfromtxt"
    features = D[:, :-1]                         # Store all columns but the last one in features
    labels = D[:, -1]                           # Store the last column in labels
    return features, labels                     # Return features and labels

def main():
    # Load train.csv into [Xtrain, Ytrain]
    print("Loading train.csv")
    Xtrain, Ytrain = load_data("train.csv")
    print("\ttrain.csv loaded.\n\tNum rows: ", Xtrain.shape[0], "\n\tNum features: ", Xtrain.shape[1])

if __name__ == "__main__":
    main()
```

Report: Compute the RMSE of the predictions using the provided files and write it below:

Report your RMSE here:

Next, tune the hyperparameters k and σ to find hyperparameters that work better than the provided values of $k = 10$ and $\sigma = 100$. You should find parameters that result in an RMSE below 0.76. Report the hyperparameters and the resulting RMSE below.

Report your value of k here:

Report your value of σ here:

Report your new RMSE here:

Lastly, write a brief reflection on this process. Were there tricky bugs that you ran into while writing your implementation? How did you tune the hyperparameters? Was it easier or harder than you expected to tune the hyperparameters?

Your reflection: