CSE 142 - Machine Learning

Assignment #3

Due Wednesday, Nov 17th by 11:59pm PST

Notes:

- This assignment is to be done individually. You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the question, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.
- Be sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Be aware of the late policy in the course syllabus.
- Justify every answer you give – show the work that achieves the answer or explain your response.
- Any updates or corrections will be posted both on Piazza and Canvas, so check there occasionally.
- To turn in your assignment:
    - Submit your reports in a PDF file through Gradescope.
    - Submit your code on Codalab. See the problem description for details

For this assignment, we will hold 2 Codalab competitions.

Problem 1. Voted Perceptron (50 points)

For this assignment, we publish a CodaLab competition. *The competition link is https://competitions.codalab.org/competitions/36185?secret_key=a001a20c-c344-45bb-b3aa-6ec23c70b446. For late submission: https://competitions.codalab.org/competitions/36186?secret_key=aeb84069-5392-46a4-b4b0-b783f325373b.*

Perceptron algorithm is one of the classic algorithms which has been used in machine learning from early 1960s. It is an online learning algorithm for learning a linear threshold function which works as a classifier. We will also apply the Voted Perceptron algorithm on the classification task in this assignment.

Let $D = \{(x_i, t_i)\}_{i=1}^{m}$ be the training data, let x_i be the feature vectors and $t_i \in \{-1, +1\}$ be the corresponding labels. The goal of the Perceptron algorithm is to find a vector w which defines a linear function such that $\forall i, t_i(w \cdot x_i) > 0$. For the Voted Perceptron, you need a list of weighted vectors $\{(w_k, c_k)\}_{k=1}^{K}$ where w_k is the vector and c_k is its weight. (Please refer to https://cseweb.ucsd.edu/~yfreund/papers/LargeMarginsUsingPerceptron.pdf for a clear description of voted perceptron.)

```
Initiate k=1, c_1 = 0, w_1 = 0, t = 0;
while t <= T do
    for each training example (x_i, t_i) do
        if y_i (w_k x_i) <= 0 then
            w_k+1 = w_k + t_i x_i;
            c_k+1 = 1;
            k = k + 1
        else
            c_k += 1;
        end
    end
    t = t + 1;
end
```

Then you can use all these vectors and their weight to do the classification: the predicted label $\hat{y}$ for any feature vector would be

$$\hat{y} = sign(\sum_{k=1}^{K} c_k sign(w_k x))$$

Dataset

For this problem, you can download the training data from [https://www.dropbox.com/s/gqye1fydkdg8ig4/hw3*data*q1.zip?dl=0). It contains two CSV files:

- `Xtrain.csv` Each row is a feature vector. The values in the i-th columns are integer values in the i-th dimension.
- `Ytrain.csv` The CSV file provides the binary labels for corresponding feature vectors in the file `Xtrain.csv`.

Submission format

The final submission format should be

```
- submission.zip
    - run.py
    - other python scripts you wrote
```

Note that to create a valid submission, please use the command `zip -r submission.zip run.py [other python scripts]` starting from the directory. **DO NOT zip the directory itself, just its content.** A sample `run.py` file is

```
#!/usr/bin/env python

# import the required packages here

def run(Xtrain_file, Ytrain_file, test_data_file, pred_file):
    '''The function to run your ML algorithm on given datasets, generate th

    Parameters
    ----------
    Xtrain_file: string
        the path to Xtrain csv file
    Ytrain_file: string
        the path to Ytrain csv file
    test_data_file: string
        the path to test data csv file
    pred_file: string
        the prediction file to be saved by your code. You have to save your
    '''

    ## your implementation here
    # read data from Xtrain_file, Ytrain_file and test_data_file

    # your algorithm

    # save your predictions into the file pred_file


# define other functions here
```

Evaluation Criteria

The final score will be a weighted combination of accuracy and F-1 score to evaluate your results:

$$final\_score = 50 \times accuracy + 50 \times F\_score$$

The competition will cover 80% for this problem.

Report (20%)

In addition to the code submission on CodaLab Competitions, you are also supposed to write a PDF report and submit it on canvas. The report should solve the following question:

> First, use the last 10% of the training data as your test data. Compare Voted Perceptron on several fractions of your remaining training data. For this purpose, pick 1%,2%,5%,10%,20% and 100% of the first 90% training data to train and compare the performance of Voted Perceptron on the test data. Plot the accuracy as a function of the size of the fraction you picked (x-axis should be "percent of the remaining training data" and y-axis should be "accuracy").

Problem 2. KNN classifier (50 points)

For this assignment, we publish another CodaLab competition. *The competition link is https://competitions.codalab.org/competitions/36187?secret_key=2d8558a4-921e-4902-a901-bd59c705bb55.* To solve this problem, you should implements a k-nearest neighbor (KNN) classifier. *For late submission https://competitions.codalab.org/competitions/36188?secret_key=7393c532-01b9-456e-90b0-38afe9adf343*

Dataset

For this problem, you can download the training data from [https://www.dropbox.com/s/nzbu1km1010hku7/hw3*data*q2.zip?dl=0]. The training data contains two CSV files:

- `Xtrain.csv` Each row is a feature vector. The values in the i-th columns are **float** numbers in the i-th dimension.
- `Ytrain.csv` The CSV file provides the **multi-class** labels for corresponding feature vectors in the file `Xtrain.csv` . **Please note the labels will be integer numbers between 0 and 10.**

A sample training data `Xtrain.csv` is

```
3.1665,2.9837,2.9480
3.4507,3.1793,2.9028
```

A sample training label `Ytrain.csv` is

```
1
2
```

Please note that there is neither header nor index in the CSV files. The program should use a vote among the k nearest neighbors to determine the output label of a test point; in the case of a tie vote, choose the label of the closest neighbor among the tied exemplars. In the case of a distance tie (e.g., the two nearest neighbors are at the same distance but have two different labels), choose the lowest-numbered label (e.g., choose label 3 over label 7). To determine distance/nearness in this problem, use Euclidian distance. As with the other problems, the output values should be in the appropriate order corresponding to the order of the testing data points.

Submission format

The final submission format should be

```
- submission.zip
    - run.py
    - other python scripts you wrote
```

Note that to create a valid submission, please use the command `zip -r submission.zip run.py [other python scripts]` starting from the directory. **DO NOT zip the directory itself, just its content.** A sample `run.py` file is

```
#!/usr/bin/env python

# import the required packages here

def run(Xtrain_file, Ytrain_file, test_data_file, pred_file):
    '''The function to run your ML algorithm on given datasets, generate th

    Parameters
    ----------
    Xtrain_file: string
        the path to Xtrain csv file
    Ytrain_file: string
        the path to Ytrain csv file
    test_data_file: string
        the path to test data csv file
    pred_file: string
        the prediction file name to be saved by your code. You have to save
    '''

    ## your implementation here
    # read data from Xtrain_file, Ytrain_file and test_data_file

    # your algorithm

    # save your predictions into the file pred_file


# define other functions here
```

Evaluation Criteria

The final score will be accuracy evaluate your results:

$$final\_score = accuracy$$

The competition will cover 80% for this problem.

Report (20%)

In addition to the code submission on CodaLab Competitions, you are also supposed to write a PDF report and submit it on canvas. The report should solve the following questions:

- How will the accuracy vary across different selections of k, i.e., $k = 1, 2, 3, 4...$