CS578 HW2

September 21, 2018

Total Points: 120
Due date: 11:59pm October 5, 2018.

1 Task

1.1 Objective

The objective of this assignment is to experiment on the Perceptron algorithm. You will be classifying the images of handwritten digits using two variations of Perceptron. You will train 10 perceptrons that will combinedly learn to classify the handwritten digits. Each Perceptron will have 785 inputs and one output. Each Perceptron's target is one of the 10 digits. For example, let's index the 10 Perceptrons from 0 to 9. Say we are training for an image of the digit 3. Then only the Perceptron with index 3 will have an expected output 1, all the other Perceptrons will have expected output 0. While predicting the label of an image, the Perceptron which gives the highest value of $w.x$ will be the winner, i.e. the predicted label of the image will be the index of that Perceptron.

1.2 Dataset

You will use the MNIST[1] database of handwritten digits for this task. You will have to download it from the website and parse it. Each digit is represented as a 28x28 dimensional vector here, each cell having a value in range $[0, 255]$. Scale each data value to be 0 or 1 (i.e., divide each value by 255 and round it). Make the 2D vector a 1D vector of size 784 to use in the Perceptrons. Take the bias term as another feature and pass the input value 1 for that feature. So, in total you will have a 785 dimensional feature space. The training examples are in `train-images-idx3-ubyte.gz` and the test examples are in `t10k-images-idx3-ubyte.gz` which you will find on the website containing the dataset. There are $60k$ training examples and $10k$ test examples. You will use the first $10k$ images in the training set for training and all the images in the test examples for testing. Discard the rest $50k$ of the training examples. Shuffle the training examples before training.

[1] http://yann.lecun.com/exdb/mnist/

1.3 Vanilla Perceptron/ Basic Perceptron (40 Points)

- Write down the Vanilla Perceptron algorithm in appropriate algorithm format in LaTeX. **(5 Points)**
- Implement Vanilla Perceptron in the described problem. **(5 Points)**
- Show the effect of the size of training set in learning. Create a learning curve by varying the size of the training set from 500 to 10000 with a step of 250. Use a default number of epoch 50 and default learning rate 0.001. **(8 points)**
- Show the effect of the number of epoch in learning. Create a learning curve by varying the number of epoch from 10 to 100 with a step of 5. Use a default number of training example $10k$ and default learning rate 0.001. **(8 Points)**
- Show the effect of learning rate for the values $.0001, .001, .01, .1$ graphically. Take default number of epochs 50 and default training example size $10k$. **(8 Points)**
- Report your observations on the learning curves. **(6 Points)**

1.3.1 Scripts

Do all the learning curve generation task in background and while submitting your script's in/out format should be as follow.

Input format:

```
$ python vanilla_perceptron.py [size of training set] [no of epochs]
[learning rate] [path to DATA_FOLDER]
```

For example, for the command `"python vanilla_perceptron.py 1000 20 0.001 data"` will take 1000 training examples, will run 20 epochs while training and set a learning rate of 0.001. You can assume that `"train-images-idx3-ubyte.gz", "train-labels-idx1-ubyte.gz", "t10k-images-idx3-ubyte.gz", "t10k-labels-idx1-ubyte.gz"` all reside directly in the `"DATA_FOLDER"`.

Output format:

```
Training F1 score: 0.81
Test F1 score: 0.77
```

1.4 Average Perceptron (40 Points)

In an average Perceptron, instead of returning the updated weights at the end of the training, we return the average of all the weights calculated in each iteration for each data point.

- Write down the Average Perceptron algorithm in appropriate algorithm format in LaTeX. **(5 Points)**
- Implement Average Perceptron in the described problem. **(5 Points)**

- Show the effect of the size of training set in learning. Create a learning curve by varying the size of the training set from 500 to 10000 with a step of 250. Use a default number of epoch 50 and default learning rate 0.001. **(8 points)**
- Show the effect of the number of epoch in learning. Create a learning curve by varying the number of epoch from 10 to 100 with a step of 5. Use a default number of training example $10k$ and default learning rate 0.001. **(8 Points)**
- Show the effect of learning rate for the values $.0001, .001, .01, .1$ graphically. Take default number of epochs 50 and default training example size $10k$. **(8 Points)**
- Report your observations on the learning curves. **(6 Points)**

1.4.1 Scripts

Do all the learning curve generation task in background and while submitting your script's in/out format should be as follow.

Input format:

```
$ python average_perceptron.py [size of training set] [no of epochs]
[learning rate] [DATA_FOLDER]
```

For example, for the command `"python average_perceptron.py 1000 20 0.001"` will take 1000 training examples, will run 20 epochs while training and set a learning rate of 0.001. You can assume that `"train-images-idx3-ubyte.gz", "train-labels-idx1-ubyte.gz", "t10k-images-idx3-ubyte.gz", "t10k-labels-idx1-ubyte.gz"` all reside directly in the `"DATA_FOLDER"`.

Output format:

```
Training F1 score: 0.78
Test F1 score: 0.76
```

1.5 Winnow (20 Points)

Now, design the above problem using Winnow algorithm. Perform the whole learning task which includes the all of the following.

- Write down the Winnow algorithm you used in appropriate algorithm format in LaTeX. **(5 Points)**
- Implement Winnow in the described problem instead of Perceptron. **(5 Points)**
- Which hyper-parameters did you tune and how? Show the learning curves. State what default values you used while tuning one hyper-parameter. **(5 points)**
- Report your observations on the learning curves. **(5 Points)**

1.5.1 Scripts

Do all the learning curve generation task in background and while submitting your script's in/out format should be as follow.

Input format: Your script containing the implementation of Winnow will be named as `winnow.py`. State what other command line arguments are needed to run this file in your report. (Hint: Follow the format of the previous two tasks.)

Output format:

```
Training F1 score: 0.78
Test F1 score: 0.76
```

1.6 Important Notes

- You need to implement this assignment from scratch in `Python 2.7`.
- For all kind of learning curves Macro F1 score will be the performance measure in this assignment.
- You should calculate the Macro F1 score this time also by yourself without using any external package (ex. `sklearn`).
- All the learning curves should show graphs for both the training and test set.
- Place all the learning curves in your report.
- Initialize all the weights randomly.
- For pre-processing and loading the dataset you can use any package installed in `data.cs`. So check in `data.cs` before using any package.

2 Open Ended Questions (20 Points)

Briefly answer the following questions in your report.

- Compare Vanilla Perceptron, Average Perceptron and Winnow algorithm based on your experiment above and state your observations. **(3 Points)**
- Define in one sentence: Mistake bound (your answer should include all the components described in class). **(4 Points)**
- Suggest a mistake bound algorithm for learning Boolean conjunctions (hint: recall the elimination algorithm for monotone conjunctions). Show that your algorithm is a mistake bound algorithm for Boolean conjunctions. **(5 Points)**
- Given a linearly separable dataset consisting of 1000 positive examples and 1000 negative examples, we train two linear classifier using the perceptron algorithm. We provide the first classifier with a sorted dataset in which all the positive examples appear first, and then the negative examples appear. The second classifier is trained by randomly selecting examples at each training iteration. (1) Will

both classifiers converge? (2) what will be the training error of each one of the classifiers? **(4+4=8 Points)**

3 Submission Procedure

You are required to use LaTeX to type your solutions to questions, and report of your programming as well. https://www.overleaf.com/ is a website you can use freely as a Purdue student. Other formats of submission will **not** be accepted. A template named "homework_template.tex" is also provided for your convenience. Create a file named `"README"` and write down anything you want to let the TAs know about your program. Especially, when the program does not run without any bug.

Your code will be tested on `data.cs.purdue.edu`, where you submit your homework as well. **Make sure that your program runs properly on `data.cs.purdue.edu`.** After logging into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely, as you are all granted the accounts to CS data machines during this class), please follow these steps to submit your assignment:

1. Make a directory named *'yourname_yoursurname'* (all letters in lower case) and copy all of your files there: `report.pdf`, `vanilla_perceptron.py`, `average_perceptron.py`, `winnow.py` and `README` (optional) shall reside directly in this directory. **Don't put the dataset files.**
2. While in the upper level directory (if the files are in /homes/dan/dan_goldwasser, go to/homes/dan), execute the following command:

   ```
   turnin -c cs578 -p HW2 *your_folder_name*
   ```

   (e.g. your instructor would use: `turnin -c cs578 -p HW2 dan_goldwasser` to submit his homework)

   Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

3. You can verify the contents of your submission by executing the following command:

   ```
   turnin -v -c cs578 -p HW2
   ```

   Do **not** forget the -v flag here, as otherwise your submission would be replaced with an empty one.

Failure to follow the above instructions will incur the penalty when your homework is being graded.

3.1 Late Policies

As declared in the class.

3.2 Plagiarism Policies

Seriuosly, no cheating. If plagiarism was found, both you and the one whose homework you "referred to" receive 0 point on that homework. If you were found to have plagiarised more than once, we will report to the instructor and the department (and your home department if you are not a CS student).