CS440/ECE448 Spring 2024

MP04: Perceptron

The perceptron model is a linear function that tries to separate data into two or more classes. It does this by learning a set of weight coefficients w_i and then adding a bias b . Suppose you have features $x_1, \cdots, x_n$ then this can be expressed in the following fashion:

$$f_{w,b}(x) = \sum_{i=1}^{n} w_i x_i + b$$

You will use the perceptron learning algorithm to find good weight parameters w_i and b such that $sign(f_{w,b(x)}) > 0$ when there is an animal in the image and $sign(f_{w,b}(x)) \leq 0$ when there is a no animal in the image.

Your function classifyPerceptron() will take as input the training data, training labels, development data, learning rate, and maximum number of iterations. It should return a list of labels for the development data. And you are supposed to tune the learning rate inside trainPerceptron function.

Training and Development

Please see the textbook and lecture notes for the perceptron algorithm. You will be using a single classical perceptron whose output is either positive or negative (i.e. sign/step activation function).

You may use NumPy to program your solution. Aside from that library, no other outside non-standard libraries can be used (PyTorch is NOT allowed for MP4).

The autograder will supply the set of test images as one giant numpy array. The development data is in the same format. So your code must read the dimensions from the input numpy array.

In the dataset, each image is $32 * 32$ and has three (RGB) color channels, yielding $32 * 32 * 3 = 3072$ features. However, be aware that synthetic tests in the autograder may have different dimensions. So do not hardcode the dimensions.

- Training: To train the perceptron you are going to need to implement the perceptron learning algorithm on the training set. Each pixel of the image is a feature in this case. Be sure to initialize weights and biases to zero. Note: To get full points on the

autograder, use a constant learning rate (no decay) and do not shuffle the training data
- Development: After you have trained your perceptron classifier, you will have your model decide whether or not each image in the development set contains animals. In order to do this take the sign of the function $f_{w,b}(x)$. If it is negative or zero then classify as 0. If it is positive then classify as 1.

Use only the training set to learn the weights.

If designed correctly, the perceptron model should give you accuracy of around 0.78 to 0.80 on development set.

Some comments on Using Numpy

For this mp, you are allowed to use only the NumPy. Note that it is much easier to write fast code by using numpy operations. Your data is provided as a numpy array. Use numpy operations as much as possible, until right at the end when you choose a label for each image and produce the output list of labels.

NumPy Tips:

- Running computations on arrays tend to be faster when run using NumPy arrays. If you are having issues with timing out, then consider using a NumPy implementation
- Linear algebra operators (dot product, vector norm, matrix multiplication) are immensely simplified when using NumPy. Consider looking up methods to perform some of these operations if they are needed in your solution.
- NumPy Broadcasting may make your life easier for this assignment.

Provided Code Skeleton

- reader.py - This file is responsible for reading in the data set. It creates a giant NumPy array of feature vectors corresponding to each image.
- mp4.py - This is the main file that starts the program of mp4, and computes the accuracy, precision, recall, and F1-score using your implementation of mp4.
- perceptron.py is file where you will be doing all of your work for mp4.
- mp4_data is the file of data set.

The only file you will need to modify is perceptron.py

To run the mp4:

```
In [1]: ! python mp4.py
```

```
Testing your perceptron model on linearly separable data
shape of training data:  (50, 2)
Accuracy: 1.0
F1-Score: 1.0
Precision: 1.0
Recall: 1.0
Test passed!
Your accuracy: 1.00.
Testing your perceptron model on the actual data
shape of training data:  (7500, 3072)
Perceptron
Accuracy: 0.8016
F1-Score: 0.8462492250464972
Precision: 0.7867435158501441
Recall: 0.9154929577464789
A correct implementation of perceptron should achieve accuracy of around 0.7
8 to 0.80 on dev. data.
Time taken for perceptron to execute training and classification: 1.25702190
39916992
Time taken for our model implementation is approximately 1.8
```

Deliverables

MP4 will be submitted via Gradescope; please upload perceptron.py.