Assignment 5

Gaurav Baney, Wei-Han Chang gbaney@usc.edu, weihanc@usc.edu
04/09/2020

Neural Networks

1. Implementation

Our implementation of PCA is in the file 'NeuralNetwork.py'
The training and test files have been hardcoded to run upon executing the file.

The results of my implementation were as follows:

```
Output
[ 0.86667137, -0.4962773 ]
[-0.23276482, -0.4924792 ]
[ 0.44124968,  0.71496368]
Accuracy
```

In our implementation we use numpy arrays to improve the speed of the algorithm (SIMD operations) and also for certain math features that can be directly applied to the numpy arrays(Transpose, dot, summations, etc)
Using NumPy I can directly subtract out the means from the parsed data, calculate the covariance and also the eigenvalues and eigenvectors

1.0.1 Data Structures

We used numpy arrays to vectorize all the pmg input and they labels they carry
We use lists to keep track of per epoch data for each of the input
We use a dictionary to store the results of single forward and single backward movement
We save the weights and biases in a separate dictionaries.

1.0.2 Code Level Optimzations

We hardcoded our neural nets architecture according to the input size and given number of layers to make it easier to set up the initials weights and biases

1.1 Software Familiarization

1.1.1 sklearn.Nueral

//TODO

```
[ 0.86667137, -0.4962773 ]
[-0.23276482, -0.4924792 ]
[ 0.44124968,  0.71496368]
```

PCA was pretty straightforward, our answers matched.

2. Applications

2.1 Recurrent Neural Networks

RNNs are a derived form of Neural Networks but the graph formed by the nodes is directed graph along a straight sequence of nodes. They also have access to the state of other internal nodes and because of this are really good at sequence generation. Their primary use cases are handwriting and speech recognition.