**Sentiment Analysis**

**Word feature:** A word can be either classified as a positive, negative or a neutral word.

The word features for the given training data:

* Positive word count - 54252
* Negative word count - 52652
* Total word count - 72782

The positive to negative ratio for 3 words in the data:

* Pos-to-neg ratio for 'the' = 1.0618578344986418
* Pos-to-neg ratio for 'amazing' = 4.019607843137255
* Pos-to-neg ratio for 'terrible' = 0.17317612380250552

Few words with **neutral** score:

* realize
* hands
* extreme
* beat
* onto
* psycho
* test
* obsessed
* choose
* speech

**Classification model:**

The classifier **nonml\_classifier()** function works in such a way that it iterates through every word in the review.

The positive to negative ratio output range for the words are:

* **Positive words range : >0.3**
* **Neutral words range: -0.3 to 0.3**
* **Negative range : < -0.3**

So for every word if pos\_neg\_ratios is greater than 0.3 it is classified as a positive word. If pos\_neg\_ratios is less than -0.3 it is classified as a negative word. For a review, if the majority of the words is positive then it is considered as a positive review and “POSITIVE” is returned, else it is considered as negative and “NEGATIVE” is returned in the function.

**Accuracy of the model on test data = 0.761**

**REPORT 2.**

learning\_rate = 0.01

batch\_size = 400

num\_epochs = 50

Train acc: 0.568292, Test\_acc: 0.568333 - Epoch 1

Train acc: 0.884833, **Test\_acc: 0.846250** - Epoch 50

Time elapsed - 25.921778440475464 seconds.

From report 1 and report 2, we can deduce that **vanilla neural networks are better than rule-based algorithms implemented in Approach 1** because it produces better accuracy.

The word level features developed in Approach 1 would be helpful here. The pos-neg-ratios calculated for Approach 1 are used here to decide whether to ignore a word or not. All the words identified as “neutral” will be ignored. If the pos-neg-ratio of a word is in the range [-0.3,0.3] it is considered as a neutral word.

The find\_ignore\_words() checks the pos-neg-ratio of every word in each of the reviews and adds only those words to vocab\_size whose pos-neg-ratio does not lie in [-0.3,0.3].

**REPORT 3.**

**Changing the width of the hidden layer:**

**Width = 10**

Train acc: 0.525042, Test\_acc: 0.518750 - Epoch 1

Train acc: 0.890958, Test\_acc: 0.838750 - Epoch 50

Time elapsed - 37.93610978126526 seconds.

**Width = 20**

Train acc: 0.585625, Test\_acc: 0.605000 - Epoch 1

Train acc: 0.870875, Test\_acc: 0.818750 - Epoch 50

Time elapsed - 36.803650856018066 seconds.

**Width = 30**

Train acc: 0.520833, Test\_acc: 0.562500 - Epoch 1

Train acc: 0.864833, Test\_acc: 0.835000 - Epoch 50

Time elapsed - 40.840527296066284 seconds.

**Width = 128**

Train acc: 0.508417, Test\_acc: 0.491250 - Epoch 1

Train acc: 0.859500, Test\_acc: 0.810000 - Epoch 50

Time elapsed - 56.61194205284119 seconds.

**Width = 256**

Train acc: 0.533417, Test\_acc: 0.572500 - Epoch 1

Train acc: 0.872208, Test\_acc: 0.826250 - Epoch 50

Time elapsed - 63.52945804595947 seconds.

**Impact on the performance:**

As the width of hidden layers increases we observe a fluctuation in the accuracy. So, a general trend could not be observed.

**Changing the Epochs:**

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 50**

n\_input = vocab\_size

n\_classes = 2

Train acc: 0.614917, Test\_acc: 0.595000 - First epoch

Train acc: 0.889958, **Test\_acc: 0.826250** - Last epoch

Time elapsed - 41.0543794631958 seconds

--------------------------------------------------------

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 60**

n\_input = vocab\_size

n\_classes = 2

Train acc: 0.528458, Test\_acc: 0.485000 - First epoch

Train acc: 0.883667, **Test\_acc: 0.827500 -** Last epoch

Time elapsed - 47.117942571640015 seconds.

--------------------------------------------------------

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 70**

n\_input = vocab\_size

n\_classes = 2

Train acc: 0.470250, Test\_acc: 0.488750 - First epoch

Train acc: 0.893167, **Test\_acc: 0.827500** - Last epoch

Time elapsed - 55.41533660888672 seconds.

--------------------------------------------------------

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 80**

n\_input = vocab\_size

n\_classes = 2

Train acc: 0.494958, Test\_acc: 0.487500 - First epoch

Train acc: 0.883208, **Test\_acc: 0.850000** - Last epoch

Time elapsed - 71.49704504013062 seconds.

--------------------------------------------------------

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 90**

n\_input = vocab\_size

n\_classes = 2

Train acc: 0.551500, Test\_acc: 0.576250 - First epoch

Train acc: 0.892375, Test\_acc: 0.837500 - Last epoch

Time elapsed - 83.68542885780334 seconds.

--------------------------------------------------------

learning\_rate = 0.001

batch\_size = 400

**num\_epochs = 95**

n\_input = vocab\_size

n\_classes = 2

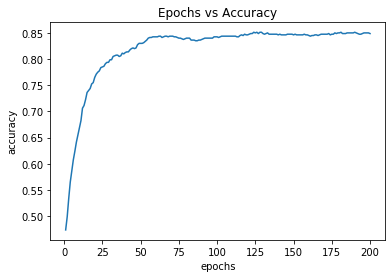
Train acc: 0.575083, Test\_acc: 0.580000

Train acc: 0.894875, **Test\_acc: 0.843750**

Time elapsed - 75.72408819198608 seconds.

**Impact on the performance:**

As the number of epochs increase, we observe an increase in accuracy to an extent. After a point the accuracy fluctuates with increase in epochs giving approximately the same accuracy.



**Changing the number of hidden layers:**

**1 Hidden layer:**

N\_hidden\_1 = 10

Train acc: 0.525708, Test\_acc: 0.483333 - Epoch 1

Train acc: 0.879458, Test\_acc: 0.828750 - Epoch 50

Time elapsed - 38.644378423690796 seconds.

**2 Hidden layers:**

n\_hidden\_1 = 50 # 1st layer number of neurons

n\_hidden\_2 = 25

Train acc: 0.794375, Test\_acc: 0.796667

Time elapsed - 51.20625448226929 seconds.

**4 Hidden layers:**

n\_hidden\_1 = 50 # 1st layer number of neurons

n\_hidden\_2 = 40 # 2nd layer number of neurons

n\_hidden\_3 = 40 # 3rd layer number of neurons

n\_hidden\_4 = 40 # 4th layer number of neurons

Train acc: 0.491625, Test\_acc: 0.488333 - Epoch 1

Train acc: 0.617875, Test\_acc: 0.605000 - Epoch 50

Time elapsed - 41.730207204818726 seconds.

**Impact on the performance:**

As the number of hidden layers increases we observe a decrease in the accuracy.

**Set of ignored words:**

**Range: [-0.2,0.2] -**

Train acc: 0.582708, Test\_acc: 0.643750 - Epoch 1

Train acc: 0.950542, Test\_acc: 0.957500 - Epoch 50

Time elapsed - 38.95972681045532 seconds.

**Range: [-0.3,0.3] -**

Train acc: 0.464833, Test\_acc: 0.512500 - Epoch 1

Train acc: 0.987667, Test\_acc: 0.980000 - Epoch 50

Time elapsed - 38.69011068344116 seconds.

**Range: [-0.4,0.4] -**

Train acc: 0.496958, Test\_acc: 0.535000 - Epoch 1

Train acc: 0.956708, Test\_acc: 0.953750 - Epoch 50

Time elapsed - 43.66018223762512 seconds.

**Range: [-0.5,0.5] -**

Train acc: 0.549917, Test\_acc: 0.527500 - Epoch 1

Train acc: 0.844833, Test\_acc: 0.800000 - Epoch 50

Time elapsed - 48.049405574798584 seconds.

**Impact on the performance:**

As the range of Pos-to-neg ratio varies we observe a decrease in the accuracy.

* Among all we could observe that changing the number of hidden layersfrom 1 hidden layer to 4 hidden layers the test accuracy decreased from 0.82 to 0.60
  + 1 Hidden layer- Test\_acc: 0.828750
  + 2 Hidden layers- Test\_acc:0.796667
  + 4 Hidden layers- Test\_acc:0.605000
* So we could deduce that, **“changing the number of hidden layers has the most impact on the performance ”.**