Intro to Machine Learning
Programming Assingment - 2

Group 15

Bhavin Jawade CSE, University at Buffalo

Anuj Narayanswami CSE,University at Buffalo
Akhil Singh Chauhan IE, University at Buffalo

April 2020

1 Part 1 - review Analysis

1.1 Approach 1 - Word Count Vectorization

Task 1: Implement a simple hand-crafted classifier that can label a movie review as positive or negative.
Report 1: Describe your word features and the classification model and report the accuracy of your model on the test data.

The word feature in this problem is the log of positive-to-negative ratio.

1. $x_t \in X_t$ Total count of word x_i = Number of times a word x_i occurs in any review.
2. $x_p \in X_p$ Positive count of word x_i = Number of times a word x_i occurs in a positive review.
3. $x_n \in X_n$ Negative count of word x_i = Number of times a word x_i occurs in a Negative review.
4. $x_r \in X_r = \frac{X_p}{X_n}$ Ratio of positive word count of x_i to Negative word count of x_i.
5. $x_{log(r)} \in X_l = \log X_r$ Log of positive to negative ratio.

Once we have these above features, we find the top 100 most common word from X_t vector. For the top 100 most common words, we find the ratio of positive to negative count. And then calculate the log of this value to get vector $X_l or X_{word\ feature}$

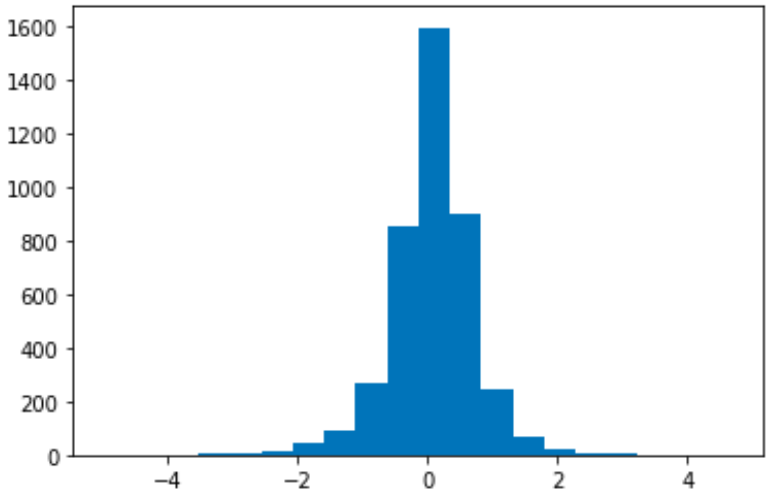
So the word feature used for classification is:

$$X_{word\ feature} = \log \frac{X_p}{X_n}$$

Classification model:

To classify any review, we first split the review into words and then for each word we look for log of ratio of positive to negative for this word in $X_{word\ feature}$ vector. Then we take sum of all these values, and check if, the total value is less than 0, then the feature is negative else it is positive.

Figure 1: The below histogram give us the distribution of the values of $X_{word\ feature}$ vector.



Accuracy of the above model on test data: 0.794

1.2 Approach 2 - Neural Network Based Sentiment Classification

Report 2: Report the accuracy and running time of the vanilla neural network. Is it better or worse than your rule-based algorithm implemented in Approach 1?
Using a vanilla Neural network:

1. **Training accuracy:** 0.928208
2. **Testing accuracy:** 0.857500
3. **Time elapsed:** 34.335062421463205 seconds.

This is better than our rule based classifier made in approach one. This better because, in the previous case instead of using a hard coded rule, the rule over the feature (count of each word in review) is found by the neural network.

Report 3: Report the performance of the model (both in terms of accuracy and time) using different values for the hidden layer width, number of hidden layers, number of epochs and a set of ignored words. What has the most impact on the performance?

1. For improvement Neural network | Change width:
Hyperparameters:

1. Hidden Layer 1: 20
2. Number of epochs: 50
3. Number of Hidden layers: 1
4. Results:

(a) **Training accuracy:** 0.909583

(b) **Testing accuracy:** 0.831250

(c) **Time elapsed:** 18.63854193687439 seconds.

2. For improvement Neural network | Increase Layers:
Hyperparameters:

1. Hidden Layer 1: 20
2. Layer 2: 20
3. Number of epochs: 50
4. Number of Hidden layers: 2
5. Results:
   (a) **Training accuracy:** 0.900958
   (b) **Testing accuracy:** 0.852500
   (c) **Time elapsed:** 19.255243062973022 seconds.

3. For improvement Neural network | Increase Epochs:
Hyperparameters:

1. Hidden Layer 1: 20
2. Hidden Layer 2: 20
3. Number of epochs: 200
4. Number of Hidden layers: 2
5. Results:
   (a) **Training accuracy:** 0.902083
   (b) **Testing accuracy:** 0.823750
   (c) **Time elapsed:** 75.48099684715271 seconds.

4. For improvement Neural network | Ignore words:
Hyperparameters:

1. Hidden Layer 1: 20
2. Hidden Layer 2: 20
3. Number of epochs: 50
4. Number of Hidden layers: 2
5. Results:
   (a) **Training accuracy:** 0.899958
   (b) **Testing accuracy:** 0.841250

(c) **Time elapsed:** 17.923603296279907 seconds.

Increasing the epochs has the most impact on the testing error. This could be caused because of over-fitting, as over-fitting generally decreases the testing accuracy while increasing the training accuracy. Apart from this, even using ignore words, to ignore neutral words, also had a significant impact on the accuracy.

2 Part II - Image Classification on the AI Quick Draw Dataset

1. Run the evaluation of the 1 hidden layer neural network in the notebook - PA2-Part2.ipynb and report the test accuracy and the run time.

1 Hidden Layer Parameters:

1. Hidden Layer 1: 256, relu
2. Number of epochs: 100, 500
3. Number of Hidden layers: 1
4. Input image size: 28 x 28
5. Results for 100 epochs:
   (a) **Training accuracy:** 0.7990
   (b) **Testing accuracy:** 0.7056
   (c) **Training Time:** 2013.1764841079712 seconds.
6. Results for 500 epochs:
   (a) **Training accuracy:** 0.8095
   (b) **Testing accuracy:** 0.68948
   (c) **Training Time:** 6056.42804479599 seconds.

Compare the performance when the number of hidden layers are increased to 3 and 5.

3 Hidden Layer Parameters:

1. Hidden Layer 1: 256, relu
2. Hidden Layer 2: 128, relu
3. Hidden Layer 3: 64, relu
4. Number of epochs: 500
5. Number of Hidden layers: 3
6. Input image size: 28 x 28
7. Results for 100 epochs:
   (a) **Training accuracy:** 0.2215
   (b) **Testing accuracy:** 0.224

(c) **Training Time:** 5431.980283975601 seconds.

8. Results for 500 epochs:

   (a) **Training accuracy:** 0.1659

   (b) **Testing accuracy:** 0.16648

   (c) **Training Time:** 6016.644160985947 seconds.

5 Hidden Layer Parameters:

1. Hidden Layer 1: 256, relu
2. Hidden Layer 2: 128, relu
3. Hidden Layer 3: 64, relu
4. Hidden Layer 4: 32, relu
5. Hidden Layer 5: 16, relu
6. Number of epochs: 100
7. Number of Hidden layers: 5
8. Input image size: 28 x 28
9. Results for 100 epochs:

   (a) **Training accuracy:** 0.0985

   (b) **Testing accuracy:** 0.1

   (c) **Training Time:** 1194.9640402793884 seconds.

10. Results for 500 epochs:

    (a) **Training accuracy:** 0.0993

    (b) **Testing accuracy:** 0.1

    (c) **Training Time:** 6005.83696103096 seconds.

3. Use the resize images() function to reduce the resolution of the images to (20 20), (15 15), (10 10) and (5 5). Using the 1 hidden layer architecture, compare the performance at different resolutions, including the original (28 28) resolution, both in terms of test accuracy and time.

Parameters for different image resolution:

1. Hidden Layer 1: 256, relu
2. Number of epochs: 100
3. Number of Hidden layers: 1

Task: Experimenting with layers, width (complexity of model), and resizing the image, below is a sample result:

Parameters:

Size	Training Accuracy (in %)	Testing Accuracy (in %)	Testing Time (in secs)
(28 x 28)	79.90	70.56	2013.17
(20 x 20)	75.18	70.404	790.54
(15 x 15)	75.19	71.28	627.71
(10 x 10)	73.46	71.064	552.96
(5 x 5)	73.02	72.404	446.34

Table 1: Comparison of performance for various image input sizes, for 100 epochs of training

Size	Training Accuracy (in %)	Testing Accuracy (in %)	Testing Time (in secs)
(28 x 28)	80.95	68.94	6056.42
(20 x 20)	73.65	66.648	3875.18
(15 x 15)	75.03	68.96	3465.18
(10 x 10)	70.64	68.692	2716.55
(5 x 5)	70.57	70.764	2415.12

Table 2: Comparison of performance for various image input sizes, for 500 epochs of training

1. Hidden Layer 1: 100, relu
2. Hidden Layer 2: 64, relu
3. Number of epochs: 100
4. Number of Hidden layers: 2
5. Input image size: 20 x 20
6. Results:
   (a) **Training accuracy:** 0.4239
   (b) **Testing accuracy:** 0.38884
   (c) **Training Time:** 1407.510