Yucheng Jiang

# Report

In my Maxent classifier, I add all the training data to a dictionary and a list. I also filtered some noise by excluding several words that is unrelated to sentiment analysis, such as “is”, “are”, “I”, “this”, “that” and etc. Then I implemented scholastic gradient descent to train the data. When the Euclidean distance between old weight and updated weight is larger than epsilon, the training ends. In each while loop, I divided eta by 2 to make sure the gradient descent will converge. The result of Maxent classifier with epsilon 0.001, eta 0.01:

```
C:\Users\User\Downloads\sentimentAnalyzer\python>python Maxent.py ../data/imdb1/ 0.001 0.01 0.1
Training took 193.590000s!
[INFO]  Fold 0 Accuracy: 0.780000
Training took 203.053000s!
[INFO]  Fold 1 Accuracy: 0.845000
Training took 216.427000s!
[INFO]  Fold 2 Accuracy: 0.800000
Training took 201.106000s!
[INFO]  Fold 3 Accuracy: 0.845000
Training took 201.479000s!
[INFO]  Fold 4 Accuracy: 0.845000
Training took 203.899000s!
[INFO]  Fold 5 Accuracy: 0.820000
Training took 200.729000s!
[INFO]  Fold 6 Accuracy: 0.860000
Training took 202.645000s!
[INFO]  Fold 7 Accuracy: 0.790000
Training took 204.611000s!
[INFO]  Fold 8 Accuracy: 0.815000
Training took 204.572000s!
[INFO]  Fold 9 Accuracy: 0.845000
[INFO]  Accuracy: 0.824500
```

I also tried different parameters. When epsilon and eta are larger, for example 0.01 and 0.05, the training is faster but the accuracy is lower, only 78%. When epsilon and eta are smaller number, for example 0.0005 and 0.005, the training took longer time but the accuracy is slightly higher which is 83.25%.

After adding l2 regularizer, with the same input as the previous example, it has slightly better accuracy.

```
C:\Users\User\Downloads\sentimentAnalyzer\python>python RMaxent.py ../data/imdb1/ 0.001 0.01 0.1
Training took 197.975000s!
[INFO]  Fold 0 Accuracy: 0.780000
Training took 204.397000s!
[INFO]  Fold 1 Accuracy: 0.835000
Training took 182.710000s!
[INFO]  Fold 2 Accuracy: 0.850000
Training took 185.450000s!
[INFO]  Fold 3 Accuracy: 0.845000
Training took 195.468000s!
[INFO]  Fold 4 Accuracy: 0.835000
Training took 189.240000s!
[INFO]  Fold 5 Accuracy: 0.825000
Training took 186.720000s!
[INFO]  Fold 6 Accuracy: 0.865000
Training took 5334.757000s!
[INFO]  Fold 7 Accuracy: 0.815000
Training took 228.257000s!
[INFO]  Fold 8 Accuracy: 0.810000
Training took 247.408000s!
[INFO]  Fold 9 Accuracy: 0.835000
[INFO]  Accuracy: 0.829500
```

And the change of epsilon and eta has the same effect on regularized Maxent classifier. Then I tried different value of lambda. When lambda is very small, for example 0.001 or 0.01, the accuracy is very close to the Maxent classifier without regularizer, which is 82.45%. When lambda is larger, for example lambda=1, the accuracy is lower, which is 80.5%. When lambda is 10, the accuracy is 69.75%. Therefore, when lambda is increasing, the accuracy is decreasing.

For the perceptron classifier, I implemented it based on the formula in the slide. The running example:

```
C:\Users\User\Downloads\sentimentAnalyzer\python>python Perceptron.py ../data/imdb1/ 50
Training took 435.095000s!
[INFO]  Fold 0 Accuracy: 0.800000
Training took 437.046000s!
[INFO]  Fold 1 Accuracy: 0.840000
Training took 560.332000s!
[INFO]  Fold 2 Accuracy: 0.845000
Training took 582.978000s!
[INFO]  Fold 3 Accuracy: 0.835000
Training took 580.092000s!
[INFO]  Fold 4 Accuracy: 0.835000
Training took 535.240000s!
[INFO]  Fold 5 Accuracy: 0.835000
Training took 431.361000s!
[INFO]  Fold 6 Accuracy: 0.865000
Training took 433.642000s!
[INFO]  Fold 7 Accuracy: 0.815000
Training took 435.378000s!
[INFO]  Fold 8 Accuracy: 0.835000
Training took 435.806000s!
[INFO]  Fold 9 Accuracy: 0.865000
[INFO]  Accuracy: 0.837000
```

I tried different number of iterations. The accuracy is increasing when the number of iterations get larger but training took longer time. When iteration is 1, the accuracy is 65.35%. When iteration is 10, the accuracy is 78.45%. When iteration is 20, the accuracy is 81.25%. When iteration is 50, the accuracy is 83.7%. When iteration is 100, the accuracy is 84%.