Observing Classification Methods for the IMDB Movie Reviews Dataset

Authors: Andrew Chang (14802837), Joyce Tsao (33877931), Kevin Lee (61397357)

Summary

This project is intended to explore the functionality and efficacy of four different classification methods on the IMDB dataset of 50,000 movie reviews. These classifiers include the kNN classifier, logistic regression, feedforward neural network, and recurrent neural network. The results demonstrated the feedforward neural network most effectively predicted the rating of a movie based on the movie review.

Data Description

The dataset we chose for our project is the IMDB dataset. The dataset is formatted into a csv file and has two columns: review and sentiment. The dataset has a total of 50,000 reviews/sentiments and the sentiments had values {positive, negative} shown in figure 1. After applying the 4 classification methods and tuning the hyperparameters, we compared the resulting accuracy measures to determine the best classification method for our dataset.

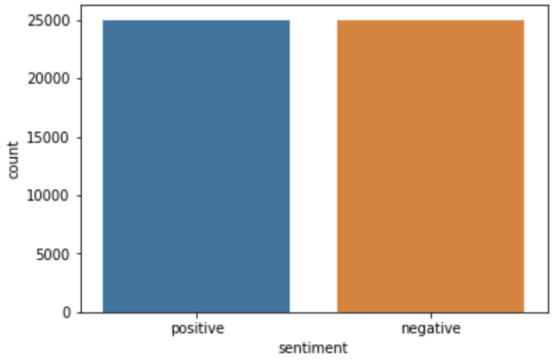

Figure 1. The counts of each sentiment where it equals positive and also negative.

Classifiers

kNN Classifier: The kNN classifier is a classification method in which an unlabeled data point is classified according to its *k* closest neighbors. The labels of these *k* datapoints are considered and the majority label is predicted to be the label of the new datapoint. We were able to train our classifier using the scikit learn's KNeighborsClassifier. Due to the simplicity of this specific classifier, we deemed it necessary to only tune the hyperparameter n_neighbors (*k*). We observed

the effects of setting *k* to: {1, 3, 5, 10, 50, 75, 100, 200}. Upon assessing the mean accuracies, we ultimately used the hyperparameter n_neighbors = 70.

Logistic Regression: Logistic regression is a classification method in machine learning used to classify an object, and in this case, it was used to classify binary data. For the classification of our IMDb dataset, we classified words used in movie reviews and the rating that was provided (1 being positive and 0 being negative) using LogisticRegression from scikit learn. Before tuning, our hyperparameters were penalty = 'l2', solver = lbfgs, and the regularization C = 1.0. We investigated the following values: penalty: {'l1', 'l2', 'elasticnet', 'none'}, solver: {'newton-cg', 'lbfgs', 'liblinear'}, and C: {100, 10, 1.0, 0.1, 0.01} for our experiment.

Feedforward Neural Network: The specific neural network we will be using for our project is a feedforward neural network (the MLP classifier in the scikit learn framework). The feedforward neural network is one of the first and most simplest types of neural networks. The input (data) is moved in only one direction –forward– then passed through a hidden layer(s), and finally to the output. The data we pass through to the neural network is a collection of movie reviews vectorized. The hyperparameters that we decided would be most beneficial for change are the amount of hidden layers (where we range from 1 to 512 hidden layers), the solver (stochastic gradient descent, adam), and also the activation method (relu and logistic).

Recurrent Neural Network: Different from the unidirectional neural network (feedforward), a recurrent neural network's connections between neurons can create a cycle, allowing the output from some neurons to affect the input to the same output. This specific neural network is proven to work better with data that is sequential. Since text is sequential, this makes a recurrent neural network an apparent choice for text-based problems. For the implementation of this neural network we used the framework Tensorflow and its library Keras to implement the "Sequential" neural network model. The hyperparameters for this neural network we mainly tuned for were the number of neurons, in which we decided on 128 neurons.

Experimental Setup

We mainly examined the accuracy rate of the training data and testing data for our classifiers and compared the rates among correlative classifiers with different hyperparameters and also the four distinct classifiers. The IMDB dataset was first partitioned to movie review and if the review was positive or negative. We parsed each movie review and removed all special characters, html formatting, and digits, which resulted in a more accurate keyword set. We vectorized each review by using scikit-learn's CountVectorizer, which gave us the data we can use for passing into our classifiers.

We randomly split our new vectorized data set into an 80/20 split. The 80% is used for our training data where we split again (60/20) where 60% is training and 20% is validation. The last

initial 20% will be used for our testing data set. In all classifiers, we initialize the random state to be 1234, so that our results will be reproducible for future uses.

For the kNN classifier, the hyperparameters were chosen based on default values, except for n_neighbors which was chosen by trial and error. For logistic regression, the original hyperparameters were chosen based on default values. For the feed-forward neural network, hyperparameters were selected based on class examples and also trial and error. For the recurrent neural network, hyperparameters were selected based on default.

Experimental Results

kNN classifier: For our final *k* value of 70, the kNN classifier yielded a testing accuracy of 69.08, training accuracy of 71.16, and validation accuracy of 69.95. Figure 2 demonstrates the change in training accuracy, testing accuracy, and error rate with changing values of k, according to the range {1, 3, 5, 10, 50, 75, 100, 200}. The error rates were calculated according to the mean accuracies.

k	Training Accuracy	Testing Accuracy
1	100	61.11
3	82.17	63.55
5	78.16	64.58
10	76.65	66.30
50	71.86	69.00
75	70.37	68.20
100	70.33	68.72
200	68.83	67.54

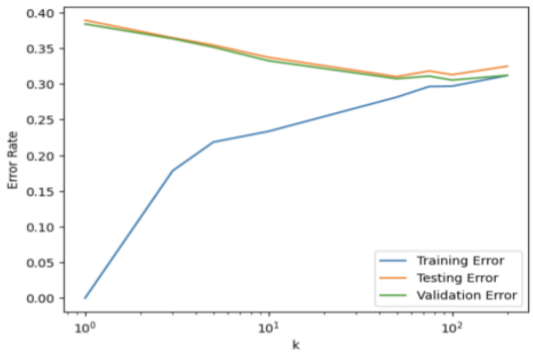

Figure 2. Accuracies/Error rates for increasingly large k values

As the graph displays the lowest error rate for the testing data at k = 70, we selected this value for the single hyperparameter.

Logistic regression: The logistic regression classifier yielded a testing accuracy of 84.38 (figure 3). After tuning our hyperparameters, the model yielded a testing accuracy of 84.38 using the different hyperparameters: solver = 'liblinear', penalty = 'l1', and C = 0.1 (figure 4). However, the tuning still resulted in the same accuracies as the original classifier. The testing accuracy of 84.38 represents the total correct predicted sentiments (positive/negative) based on review words, over the total predictions made by the model.

Training Accuracy	Validation Accuracy	Testing Accuracy
85.97	84.69	84.38

Training Accuracy	Validation Accuracy	Testing Accuracy
85.97	84.69	84.38

Figure 3 (left). Accuracy scores before tuning hyperparameters. Figure 4 (right). Accuracy scores after tuning.

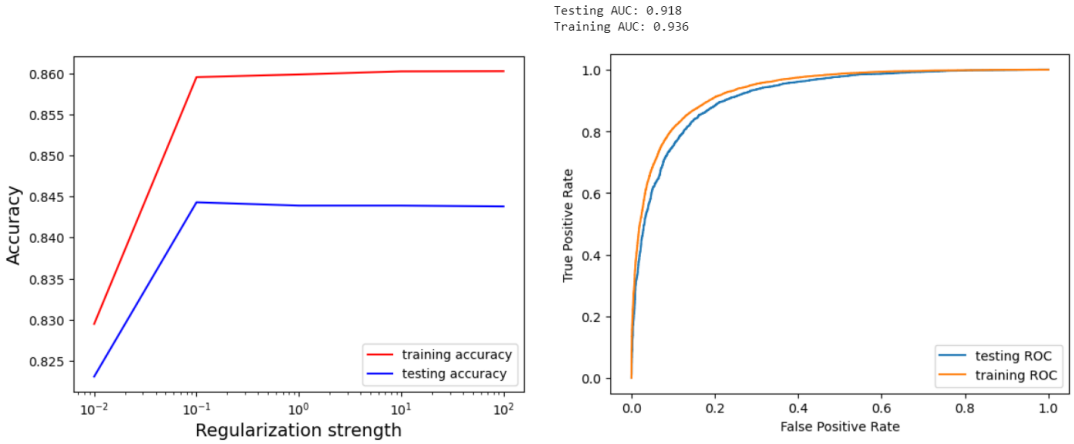

Figure 5 (left). L2 regularization plot against accuracy. Figure 6 (right). ROC curve plots.

As regularization (C) decreases, the model overfits more on the training data, thus the gap between testing and training accuracies (figure 5). A regularization rate of 0.1 provided the highest classification values for both datasets.

The ROC curve on the right (figure 6), shows the tradeoff between the model trying to maximize the true positive rate while minimizing the false positive rate of predicting the sentiment based on vocabulary words used. The ROC is also used as a method to determine a cutoff value between true positives and false positives. The AUC values for training data is slightly higher than the testing data, but both values are high, and represent the predictive power of the logistic regression model.

Feedforward neural network: Our feedforward neural network with hidden layers 1, 50, and 512 yielded testing accuracies of 86.63, 94.24, and 96.58 respectively. The testing accuracies yielded 85.40, 85.29, and 85.81 respectively (figure 7). The testing accuracies represent the predicted sentiments over the testing data. The feedforward neural network we will be plotting is the one with 1 hidden layer.

Hidden Layers	Training Accuracy	Testing Accuracy
1	86.63	85.40
50	94.24	85.29
512	96.58	85.81

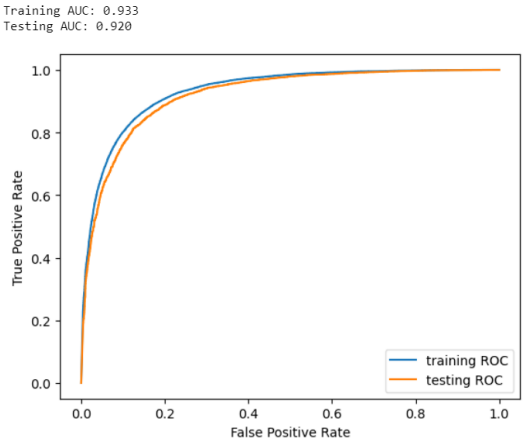

Figure 7 (left). Accuracy rates for hidden layers. Figure 8 (right). ROC curves for training and testing data.

Figure 8 shows the ROC curves for both the training and testing data, with the training ROC curve having a slightly larger AUC. This shows that the neural network has a better predictive power for the training set compared to the testing set.

Recurrent neural network: The recurrent neural network with 128 neurons yielded a training accuracy of 85.40 and a testing accuracy of 85.04. Since our data is similar both in training and testing, we can conclude that our data is not overfitting. This classifier has the lowest gap between training and testing accuracy. The ROC has the lowest lost true positive rate compared to the other classifiers.

Insights

Based on the results of the kNN classifier, we conclude that this specific classifier is not suitable for this specific dataset. Even for the most optimal *k* value, the testing accuracy is far from what is expected of an effective classifier on textual data. Despite tuning the logistic regression classifier, the best accuracy rates remained the same. Our logistic regression had better results in accuracy compared to kNN, but it still is not the best classifier for the IMDb dataset. Out of the two neural networks, we found that the feedforward neural network with 512 hidden layers had the most accurate results with a testing accuracy of 96.58. We did find this surprising because we hypothesized that a recurrent neural network would typically outperform the other classifiers for text datasets. When classification of the data is actually applied (i.e. when the final ratings for films and the order in which they are displayed on the website are decided), the inefficiency of the feedforward neural network may prove a nuisance, especially considering how the IMDb website is constantly updated over time. When people visit the website, they would expect to view quick results, but this may prove difficult for the feedforward neural network. That being said, some of the other classifiers may be better in efficiency but do not yield the high level of accuracy the feedforward neural network does. Thus, tradeoffs must inevitably be made if any of these classification are actually used in practice.

Strengths and weaknesses: kNN classifier works well with our dataset but it is memory intensive and slow for our large dataset. Logistic regression easily classifies binary information but the model also overfits on the training data. Feedforward neural networks are less likely to overfit on the training data. Though statistically effective for this dataset, the feedforward neural network demands optimization of many parameters, making the classifier arguably inefficient relative to the low number of potential labels. Recurrent neural networks process that data in a way where even when the input increases, the model size does not increase. However, it is slow to compute.

Error analysis: We examined a few movie reviews that were incorrectly predicted, and found that reviews that had a high number of negative words were often predicted as negative and vice versa when the actual label was the opposite. However, we did run into a small number of cases

in which the model did incorrectly predict reviews where the sentiment should have been easy to classify. In these cases, it was difficult to determine why the classifiers predicted incorrectly.

Contributions

Andrew Chang - Andrew contributed to the project by training and predicting the dataset for the feedforward neural network and the recurrent neural network classifiers, and figures related to these two classifiers. He also completed the experimental setup section.

Joyce Tsao - Joyce contributed to the project by training and predicting the dataset for the logistic regression classifier, and related figures. She also completed the error analysis and data description.

Kevin Lee - Kevin contributed to the project by training and predicting the dataset for the kNN regression classifier, and related figures. He also completed the summary and insights.

All members contributed equally to the classifiers and experimental results sections.