CS598 PSL Project III Report

Team Members: Name (netID), Name (netID), Name (netID)

Introduction

This project uses data from the Bag of Words Meets Bags of Popcorn Kaggle challenge, with the goal of predicting the sentiment of movie reviews using a binary classification model with a vocabulary size less than or equal to 1K. There are 50K examples collected from IMDB and each entry contains id, sentiment, score, and review. The test ids are pre-determined and are stored in a 25K-by-5 matrix. The 50K movie reviews are splitted into 5 sets of training and test data according to the test ids. The 5 splits of data sets will be used to train and evaluate 5 different models independently. We will first generate a vocabulary of size 1K, and use logistic regression for predicting movie review sentiments.

The evaluation metric used in this project is AUC (Area Under the Curve) on the test data. AUC measures how well the classifier can distinguish between different classes. The higher the AUC, the better the model can distinguish between the positive and negative classes.The ROC (Receiver Operator Characteristic) curve plots the TPR (True Positive Rate) against FPR (False Positive Rate) and is the measurement curve for AUC used in our binary classification. In our model, the smallest AUC score for the 5 splits is still higher than 0.96, which satisfies the performance target. Next, we present a detailed discussion on model interpretability. Finally, we summarize important learnings and interesting findings from this project, along with individual contributions of each team member.

Pre-processing and Model Building

Given the goal of predicting sentiment of movie reviews, the first task is to transform sentences into feature vectors that could be used as input to classification algorithms. The corpus could contain tens of thousands of unique words, leading to a huge document-term matrix that is difficult to process. Therefore, the first step is to generate a vocabulary with a reasonable size. An alternative approach is to use word embeddings to achieve dimensionality reduction, which we didn't explore in this project. Next, using the vocabulary we can generate a document-term matrix, with which we could experiment with different algorithms for binary classification.

Vocabulary Generation

As with most NLP applications, one critical component of the task of sentiment classification is to derive a vocabulary for constructing the document term matrix (DTM), which will be used as input features for the classification task. For example, it corresponds to the design matrix for linear or logistic regression. Given a corpus of reviews and their corresponding sentiment label, we apply the following steps to derive a compact vocabulary for the classification task. Steps 1-2 uses the `text2vec` package.

1. Tokenization and stop word removal using a predefined list of common words.

2. Filtering the vocabulary by removing very frequent and very infrequent terms.
3. Applying a two-sample t-test to keep the top 2K most relevant terms based on t-statistics.
4. Applying lasso (with logistic regression) for feature selection a number of times (50) with randomly sampled observations. For each run, the lambda value is selected such that it leads to the largest number of features less than 1K. Finally, the top 1K most frequently selected features are used to construct the final vocabulary for training and prediction.

Model Building

With a document-term matrix representing movie reviews, we experiment with two popular classification methods, logistic regression and gradient boosted decision tree (GBDT).

- **Logistic Regression**: We first tried logistic regression using the `cv.glmnet` function (`family = "binomial", type.measure = "auc"`). After hyper-parameter tuning, we decide to use `alpha = 0` (i.e. ridge penalty) with `lambda.min` as it leads to the smallest prediction error measure by ROC AUC score.
- **XGBoost**: We also tested GBDT for binary classification using the `xgboost` library. We used the following hyper-parameters for `xgboost`, `nrounds=4000, max_depth=4, eta=0.05, subsample=0.5`. In addition, the `objective` is set as `binary:logistic`, and `eval_metric` is set as `logloss`. The performance (AUC score) of GBDT is slightly inferior to LR but takes much longer to train. For example, for a single split, `xgboost` takes 6x longer than `cv.glmnet`. Therefore, GBDT is less favorable for this task according to our experiments, and its results are not reported.

Results

The computer system used for the experiments is a Macbook Pro with 2.3 GHz Quad-Core Intel Core i5 and 8GB Memory. Running time is recorded for each of the five splits, including both training and prediction time (excluding vocabulary construction time). We report the ROC AUC score from logistic regression for the five splits of test data below, along with their running time:

Test data	Split #1	Split #2	Split #3	Split #4	Split #5
AUC Score	0.9655	0.9639	0.9647	0.9646	0.9638
Running Time	53.73	52.07	52.21	53.24	52.37

Interpretability

Logistic Regression

One major advantage of logistic regression is its interpretability. The log-odds (i.e. $log\frac{p}{1-p}$) for the input labeled “1” is linearly related to the feature coefficients from logistic regression. In our case, the positive coefficient indicates that the feature positively contributes to the probability

that the sentiment is positive, and the negative sign indicates the opposite. Taking one sample (`id=2`) from the test data of Split #1 as an example. The original review writes "*This movie is a disaster within a disaster film. It is full of great action scenes, which are only meaningful if you throw away all sense of reality …… Hard to believe somebody read the scripts for this and allowed all this talent to be wasted. I guess my suggestion would be that if this movie is about to start on TV ... look away! It is like a train wreck: it is so awful that once you know what is coming, you just have to watch. Look away and spend your time on more meaningful content.*" The active terms (covered by the 1K vocabulary) representing this review in the document term matrix and their corresponding weights are shown in Figure 1. We can see that terms with negative weights overwhelm those with positive weights, thereby leading to an overall negative review.

acting	awful	disaster	even	for_this	great	guess	if_this	just	know	let's	mean
-0.0931	-0.5023	-0.1873	-0.0430	-0.1428	0.2429	-0.1234	-0.3512	-0.0180	0.0552	-0.0057	-0.0591
only	reality	see	sense_of	to_be	to_believe	very	very_good	wasted	worse	would	would_be
-0.0628	0.1461	0.0830	0.0963	-0.0411	-0.2037	0.0657	0.2451	-0.4438	-0.4211	-0.0529	-0.0746

Figure 1. Active terms and their weights

It is also important to note that the term frequency also contributes to the predicted probability, since a term can appear multiple times in a document/review. Furthermore, TF-IDF weighting can be employed to downweight popular terms. We can also rank the coefficients by their absolute value and find out which terms have the strongest influence on the final prediction (assuming they only show up once).The most significant 50 terms obtained using Split #1 are shown in Figure 2. We can see that these terms imply strong feelings of the reviewer or are directly related to their rating of the movie.

if_had	film_excellent	7_10	7_out	not_recommend	4_10	grade_d
-1.9410	1.7431	1.4177	1.2861	-1.2526	-1.2193	-1.1895
this_travesty	very_disappointed	should_been	definitely_recommend	had_high	8_10	below_average
-1.1292	-1.0998	1.0844	1.0726	-1.0678	1.0585	-1.0322
definitely_worth	could_been	only_saving	mystery_science	3_10	2_10	if_director
1.0208	-1.0062	-0.9919	-0.9865	-0.9819	-0.9797	-0.9763
not_recommended	of_funniest	8_out	be_missed	doesn't_help	very_poorly	wasn't_funny
-0.9676	0.9664	0.9578	0.9435	-0.9270	-0.9201	-0.9091
only_complaint	wonderful_film	well_worth	favorite_movies	only_positive	give_1	of_favorites
0.9044	0.9017	0.8990	0.8913	-0.8888	-0.8864	0.8777
just_isn't	bad_really	something_better	1_out	don't_miss	this_supposed	this_gem
-0.8744	-0.8719	-0.8706	-0.8660	0.8630	-0.8595	0.8477
save_money	acting_great	olds	can_relate	pleasantly_surprised	stinker	lifeless
-0.8423	0.8265	-0.8256	0.8237	0.8215	-0.8167	-0.8154
not_perfect						
0.8117						

Figure 2. Top 50 terms with largest coefficients (absolute value)

XGBoost

Compared to traditional linear models, XGBoost does not have straightforward explanations. SHapley Additive exPlanations (SHAP) by Lundberg and Lee (2016) is a method to explain individual predictions. SHAP is based on the game theoretically optimal Shapley Values. SHAP is integrated into the tree boosting framework XGBoost. Local Interpretable Model-agnostic Explanations (LIME) are also popular interpretable models that are used to explain individual predictions of black box machine learning models. Instead of training a global model, LIME focuses on training local models to explain individual predictions.

Discussion and Conclusion

In this project, we use logistic regression to solve the challenge of sentiment analysis of movie reviews. The most important step turns out to be the construction of a compact yet representative vocabulary, which is needed to transform a review into a row in the document term matrix. With a two-sample t-test and feature selection with lasso penalty, we were able to reduce the size of the vocabulary to merely 1K, which is still able to achieve the performance target measured by AUC.

Limitations: Compared with other approaches (e.g. GBDT, which we also tried), logistic regression is efficient and interpretable. However, there are also a few limitations. First, its performance is dependent on the input vocabulary, which is used to construct the document term matrix (also called the design matrix). A vocabulary of size 1K does not necessarily capture all the important terms for our tasks and could lead to larger errors. Second, it doesn't capture non-linear relationships between log-odds and the feature variables, which could negatively impact the prediction performance.

Improvements: We can improve the model in terms of vocabulary, word embeddings and network architecture. To more efficiently utilize the limited vocabulary, we can map words that are synonyms into a single term (link). For word embeddings, we can use word2vec or glove, which captures the similarities between terms. For model complexity, we can use deep neural networks with hidden layers and Relu activations, which introduces non-linearity into the model. State-of-the-art networks with pretrained weights, such as BERT (e.g. Hugging Face), can be used with additional fine tuning.

Learnings: From this project, we learned about the tradeoff between model complexity and interpretability. Simple models are easier to interpret and could also be very effective for NLP tasks, which provide a robust baseline for more complicated models. Explainable machine learning models are gaining attention. We need to grasp how to translate real word problems into explainable strategies for deriving actionable items and better educating the non-technical population.

Acknowledgements: We benefited a lot from numerous resources, including posts on Campuswire, office hours and Kaggle discussions. Most of our implementations are inspired by Instructor notes.

Individual contributions:

- **Name (NetID)**: data pre-processing (vocab generation), model training, coding, report writing and final submission.
- **Name (NetID)**: data pre-processing, model training, coding, report writing.
- **Name (NetID)**: model training, hyper-parameter tuning, coding, report writing.