CSE 255 Assignment 8

Alexander Asplund, William Fedus

September 25, 2015

1 Introduction

In this paper we train an L1-regularized linear support vector machine (SVM) to determine whether the sentiment of a movie review is positive or negative. We train and test on the movie review polarity dataset introduced by Pang and Lee, 2004 [2]. Classification accuracy of the linear SVM is improved through a series of experiments for various data preprocessing techniques and data transformations. Classification accuracy is found to be maximum on the 10 cross-validation folds after removing numerical entries and performing log odds weighting of terms. Our final linear SVM with per-example regularization cost $c = 1.00$ generates an 0.877 classification accuracy; this compares favorably to the 0.864 accuracy using subjectivity extracts (Pang and Lee, 2004) and the 0.905 accuracy using linguisitic knowledge sources (Ng et al, 2006 [4]).

2 Data Statistics

The corpus in our paper is the movie review polarity data set version 2.0 which consists of 1,000 positive and 1,000 negative movie reviews derived from unprocessed and unlabeled html files from IMDB. The reviews are stored as downcased text files and labeled as positive and negative according to the methodology outlined in Appendix A on the Movie Review Data website. Additionally, preliminary steps have already been taken to remove rating information. We then partition this corpus of documents into 10 separate folds for cross-validation, each fold with an equal number of positive and negative examples. We use the same 10 folds as Peng/Lee ACL 2004[2]. The movie reviews in their dataset follows a naming convention which partions reviews into folds.

With the raw data provided and partitioned, we run a series of experiments to determine which data transformations will result in the highest classification accuracy. Each document i within the corpus is represented by vector $\boldsymbol{x_i}$, where x_{ij} represents the number of occurences of word j in document i. The length of $\boldsymbol{x_i}$ is the equal to the vocabulary of the defined corpus, which is discussed in further detail in Section 3 and is the same for each document i. For any given document i in our movie review corpus, the vocabulary b will be greatly exceed

the unique word count in the document and thus the vector $\boldsymbol{x_i}$ is highly sparse with $\approx 99\%$ of the entries equal to 0.

3 Training Methodology

3.1 Training Methodology

We use a linear SVM model to create a binary classifier. 10-fold cross-validation is performed to select the regularization hyperparameter, C_l, which optimizes the 0/1 accuracy metric of classification. We use the R package LiblineaR. Since it does not accept sparse matrices training is too slow to allow for a extensive grid search of the cost hyperparameter C. Instead we use the function heuristicC in the LiblineaR package to determine a good candidate C_l. We then perform a limited gridsearch in the range $C_l \cdot 10^{-2:2}$. The objective function LiblineaR optimizes is

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{I} \text{L1-loss}(w; x_i, y_i) \tag{1}$$

3.2 Data Preprocessing

The preprocessing procedures on the raw data are selected via a series of experiments and the 0/1 accuracy metric of each transformed model is compared against the results of the baseline model without the transformation described in Section 3.1. With the baseline model, without further preprocessing and transformation, we achieve a mean 0/1 accuracy of 0.853.

First, we test the removal of a set of predefined stop words from the corpus. The stop word collection we remove from the corpus is the default stop words list in the package 'tm', which is simply a list of common english words. To assess the performance of this preprocessing step, we perform 10-fold cross-validation and achieve an average 0/1 accuracy metric of 0.858 on the cross-validation folds; therefore, we choose to use this list of stop words.

Second, we test the removal of numerical entries from the corpus. As above, we perform 10-fold cross-validation and achieve an average 0/1 accuracy metric of 0.849 on the cross-validation folds. Despite this small drop in accuracy we choose to include number removal as a preprocessing step. The dataset author warns in the readme accompanying the dataset that there might be leakage in the form of unremoved ratings in the data set. Removing all numbers will allow us greater confidence in future accuracy.

Third, we test word stemming, where suffixes are removed to generate a word stem in order to increase the relevance of similar groups of words. We used the Porter stemming algorithm[3] on the corpus and achieve an average 0/1 accuracy metric of 0.846 on the 10 cross-validation folds; therefore we choose not to perform word stemming in preprocessing.

Finally, we test the removal of punctuation from the corpus and achieve a 0/1 accuracy metric of 0.852 on the 10 cross-validation folds, therefore we choose not remove punctuation in preprocessing.

With these four experiments complete, we assume an approximately linear interaction of each of these preprocessing procedures and therefore encorporate the better of the two results from each experiment in our final data preprocessing procudure. The final preprocessing procedure achieves a 0/1 accuracy metric of 0.858 on the 10 cross-validation folds.

3.3 Data Transformations

Using the final preprocessing procedure outlined in Section 3.2 on our corpus, we now determine the optimal data transformations by performing experiments for various techniques and selecting the one that achieves the highest 0/1 accuracy metric on the 10 cross-validation folds.

First, we test one of the simplest data transformations methods, an indicator function of the presence of word j. The rational for this technique is to reduce the impact of the frequency of a word count on the final linear model. Intuitively, the presence of word j in document i will often have a higher information content for classification than the frequency of word j. Therefore, to solve this issue, for each x_{ij}, we use the following data tranformation for element x_{ij}

$$x_{ij} \mapsto I(x_{ij} > 0) \tag{2}$$

This transformation achieves an average 0/1 accuracy metric of 0.868 on the 10 cross-validation folds, which is slightly higher than the results from the preprocessed baseline with no transformation of 0.858.

Second, we test a log-based data transformation for the word j in document i. One issue of the previous transformation technique is the destruction of information. The log-based transformation reduces the impact of high frequency occurences of word j in document i while preserving sparsity of the $\boldsymbol{x_i}$ vectors which improves computational time. The log-based data transformation is

$$x_{ij} \mapsto log(x_{ij} + 1) \tag{3}$$

which achieves an average 0/1 accuracy metric of 0.867 on the 10 cross-validation folds.

Third, we test the Term Frequency Inverse Document Frequency (TFIDF) transformation, which is a statistic reflecting how important a word is to a corpus. This value will increase with the term frequency of x_j in document i but will decrease if word j is highly common throughout the entire corpus. The following transformation is applied to our dataset

$$x_{ij} \mapsto x_{ij}/c_j \tag{4}$$

where $c_j = \sum_{All\ docs\ i} I(x_{ij} > 0)$. The TFIDF transformation achieves an average 0/1 accuracy of 0.824 on our 10 cross-validation folds.

To conclude, we test the logodds data transformation, which goes beyond the scope of earlier transformations and assesses the importance of word j on the actual labels of the document i. The transformation is

$$x_j \mapsto I(x_j > 0) \left| log \frac{tp}{fn} - log \frac{fp}{tn} \right| \tag{5}$$

where tp is the number of positive training examples containing word j, fn is the number of these examples not containing the word, fp is the number of negative training examples not containing the word and tn is the number of examples not containing the word. In the event that any of these values is equal to 0, it is set to 0.5. This transformation achieves an average 0/1 accuracy of 0.869 on our 10 cross-validation folds. Additionally, as one final experiment, we test the logodds data transformation without the removal of preprocessing. This final experiment is motivated by the fact that logodds is functionally capable of discriminating the importance of these stop words, so prior removal is not essential. This final transformation achieves an average 0/1 accuracy of 0.877 on our 10 cross-validation folds with per-example cost $c = 1.00$ and represents our best classification model.

The results of these four experiments in summarize in the table below.

Description	Transformation	Accuracy	Std Dev
Baseline	$x_{ij} \mapsto x_{ij}$	0.853	0.0148
Preproc Baseline	$x_{ij} \mapsto x_{ij}$	0.858	0.0279
Indicator	$x_{ij} \mapsto I(x_{ij} > 0)$	0.868	0.0244
Log	$x_{ij} \mapsto log(x_{ij} + 1)$	0.867	0.0313
TFIDF	$x_{ij} \mapsto x_{ij}/c_j$	0.824	0.0275
Logodds	$x_{ij} \mapsto I(x_{ij} > 0) \left\| log \frac{tp}{fn} - log \frac{fp}{tn} \right\|$	0.869	0.0233
Logodds without stop word removal	$x_{ij} \mapsto I(x_{ij} > 0) \left\| log \frac{tp}{fn} - log \frac{fp}{tn} \right\|$	0.877	0.0179

Table 1: Experimental results of our transformations. The best model was found to be one that utilized a logodds feature transformation and all preprocessing steps except stop word removal.

3.4 Additional Design Considerations

In order to obviate any leakage of information from the cross-validation folds, no information from the test components is accessible to the training algorithm. For instance, the vocabulary for any given cross-validation fold is determined from the training folds alone and the vocabulary is recalculated for each fold. If then a word is present in the test fold is not present in the training corpus, they are removed. In this way, we prevent leakage of information between the folds as we proceed with cross-validation.

As described in 3.2 we also remove all numeric terms to safeguard against any numerical ratings not removed from the review texts. This ensures that we have a more robust model, since predictions are never made from simply considering the presence of a numerical rating in the text.

4 Results

4.1 Our Classifier

Our series of experiments on data preprocessing and data transformations indicated that the accuracy maximizing linear SVM would be one that used a logodds transformation on a dataset with numerical entries and punctuation removed, and word stemming. This final model achieves an average 0/1 accuracy of 0.877 on our 10 cross-validation folds.

Below we present our confusion matrix for our 10-fold cross validation. We derive it by taking the mean of true positives, true negatives, false positives and false negatives across the 10 cross validation iterations and then normalizing it.

		Predicted	
		Favorable	Unfavorable
Actual	Favorable	0.4405	0.0595
	Unfavorable	0.0635	0.4365

Table 2: Final model confusion matrix averaged across 10 folds

From our confusion matrix it is clear that there is no particular imbalance between false negatives and false positives. The recall of this final model was 0.900 and the precision was 0.882.

Most terms are not particularly surprising, but the presence of *script* and *plot* as unfavorable terms and *allows*, *overall*, *flaws*, *also* and *both* as favorable terms is interesting to note.

4.2 Comparison with Other Published Results

Determining the sentiment of movie reviews has been extensively studied for machine learning experiments since there are large on-line collections of reviews and there is a readily machine-extractable learning parameter, the rating. This IMDB dataset selected by Pang and Lee [2] has been used as a standard for many researchers over the last decade.

Pang et al, 2002 [1] initially determined that a SVM performed better than both a Naive Bayesian classifier and a maximum entropy classifier, achieving an 0.829 accuracy rate with unigrams. This accuracy metric, however, is not directly comparable since it was drawn from a different corpus of 752 negative and 1301 positive reviews. However, Pang and Lee, 2004 [2], and many subsequent papers provided a directly comparable dataset to the one used in this paper. Pang and Lee, 2004 demonstrated an 0.828 accuracy without utilization

Unfavorable	Favorable
bad	hilarious
awful	memorable
mess	wonderfully
ridiculous	terrific
worst	allows
boring	excellent
nothing	overall
wasted	perfectly
waste	others
supposed	flaws
script	also
unfortunately	performances
dull	sometimes
stupid	perfect
ludicrous	outstanding
plot	breathtaking
degenerates	both
lame	many
terrible	best
embarrassing	wonderful

Table 3: 20 Most Favorable and Unfavorable Terms

of subjectivity extracts. The subjectivity extracts employed by Pang and Lee, 2004 improved the performance of the classifier to 0.864, a statistically significant result and one that appears highly additive to this classification process. This additional procedure is logical and well-founded as it effectively filters out the noise associated with objective sentences of the reviews which have little to no bearing on the sentiment of the review.

Finally, Ng et al, 2006, improved performance on this dataset further to 0.905 through the utilization of linguisitic knowledge sources. This result, although one of the strongest of the available papers, does not appear as novel or useful given the manual human input necessary for the process.

5 Conclusions

Our best SVM classifier achieves a 0/1 accuracy of 0.877 across 10-fold cross validation with per-example cost hyperparameter $c = 1.00$. This accuracy was achieved by only removing numbers during preprocessing and then performing the logodds transformation of features.

This performance is commendable considering that we vectorize documents in a bag-of-words representation and destroy all information content in the ordering of the words. The syntax of a sentence is often essential to understand the sentiment of a sentence. For instance, the actual meaning of a sentence is

easily inverted by adding the word *not*. Reviewers frequently utilize a deliberate contrast in their reviews in order to upset reader expectations and create a more interesting structure; however, with the syntax lost in the bag-of-words representation, this confuses the machine learning algorithm which is forced to associate the same word with both positive and negative sentiment examples. Given this practice, it is not surprising that many of the most predictive words will often dictate the polarity of a sentence unambiguously, like 'aweful', 'worst', 'breathtaking', 'memorable' or 'terrific', and are not as easily twisted to thwart reader expectations.

The precision of our final model at 0.900 was higher than the recall of 0.882, which is suitable for this kind of domain. If a user visiting a movie database site and wishes to see the favorable or unfavorable reviews of a movie, he/she will likely care more about the movies presented actually having the expected disposition rather than being offered a complete list of all favorable or unfavorable reviews present in the database.

References

[1] Lillian Lee Bo Pang and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *Proceedings of EMNLP*, pages 79–86, 2002.

[2] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *In Proc. of the ACL*, pages 271–278, 2004.

[3] Martin Porter. `http://tartarus.org/martin/PorterStemmer/`, 2006. The Porter Stemming Algorithm.

[4] Sajib Dasgupta Vincent Ng and S. M. Niaz Arifin. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. *Proceedings of the COLING/ACL Poster Sessions*, 2006.