Predicting Conference Paper Acceptance

William Jen **Shichang Zhang** **Muyun Chen**

Abstract

In this paper, we examine the possibility of building a model to classify whether a conference paper can be accepted or rejected to a certain conference. We used the PeerRead dataset to build models to classify paper acceptance to ICLR, using 18 features including but not limited to number of authors and figures, abstract bag of words, and whether the abstract contains words like 'deep' or 'neural'. Using accepted and rejected papers from ICLR 2017, we trained the following models on the 172 accepted and 255 rejected papers from ICLR 2017: logistic regression with L2/L1 regression, SVM with the RBF kernel, random forest, AdaBoost, and a fully-connected neural network. We found that the SVM with the RBF kernel performed the best with an accuracy of 71%, an improvement over prior research's best of 65.3%.

1 Introduction

In recent years, there has been an explosion in scientific research applying machine learning onto ever-growing datasets, thanks to recent advances in computational power. In 2017 alone, 3,120 papers were submitted to the Neural Information Processing Systems (NIPS) conference, but only a mere 679 papers were accepted. While the peer review process is the most important way to judge the quality of research work, the scientific community has identified potential issues with the process, ranging from consistency to bias issues. One clear way to avoid these issues is to use a computer to evaluate submissions directly. The goal of this paper is to predict the acceptance of a given academic paper.

We receive raw pdfs and their reviews and labels (accept/reject) as our input, transform them into JSON files using science-parse, a library created from Kang, et. al [2], and then try a variety of models such as logistic regression with L2/L1 regularization, SVM, Random Forests, AdaBoost, and fully-connected neural networks to classify whether a paper will be accepted or rejected. We then look at each model's accuracy.

2 Related Work

Kang, et. al.[2] published initial work on this topic in April 2018 with the public release of PeerRead, a structured dataset that collects several research papers from several ML/AI conferences, such as NIPS, ICML, ICLR, and more. Further, they also developed a Java tool called science-parse to extract useful features from research papers in pdf form, such as specific paper sections, number of figures, equations, and more. These papers are also accompanied by reviewer comments with numerical ratings for the paper as well as confidence ratings for those ratings. The full feature set can be found in Kang, et. al [1].

3 Dataset and Features

We took the 427 papers submitted to ICLR 2017, including 172 accepted and 255 rejected papers. For each paper, we extracted 18 features. To simplify the model, all of our features are numerical or

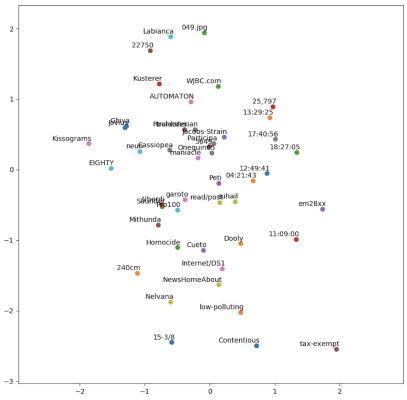

Figure 1: PCA visualization of word2vec: Closer words should appear closer together.

Boolean. Some coarse features include length of the title, the publication year, whether the fancy terms like 'deep' or 'neural' appear in the abstract. There are also more sophisticated lexical features extracted from the abstract of each paper. We used word2vec techniques to capture the information of the abstract. We reconstructed the linguistic contexts of words. In this case, we get a 300-dimensional vector space. All of the words in the abstract get mapped to a vector in this space. Thus, it is much easier for us to measure the similarity between word vectors. We can also get a visualization of the words by dimension reduction technique, e.g. principal component analysis (PCA), linear discriminant analysis (LDA), or t-distributed stochastic neighbor embedding (t-SNE). When we actually put this feature into the model, we only take the word counts to make this feature numerical and be consistent with other features we have. The full feature list can be found in the appendix.

Table 1: Extracted features from conference papers

Feature Name	Description	Type
Abstract contains ML keyword	Whether abstract contains 'deep', 'neural', etc.	boolean
Title Length	# of characters in title	integer
Authors	Number of authors	integer
Most Recent Reference Year	Latest year that a reference was published	integer
Number of References	How many references this paper uses	integer
Number of Cited References	How many cited references this paper uses	integer
Avg. Length of Mentioned References	How long each reference was talked about (in words)	integer
Number of Recent References	# of recent references (i.e. this year)	integer
Number of Figure/Table/Eqn References	# of references to tables, figures, and equations	integer
Number of Unique Words	How many unique words this paper uses	integer
Number of Sections	How many sections this paper uses (as det. by science-parse)	integer
Average Sentence Length	Avg. length of sentence, in characters	float
Contains Appendix	Does this paper have an appendix?	boolean
Proportion of Frequent Words	Proportion of frequent words	float
Abstract's Bag of Words	Bag of words in abstract	integer
TFIDF-weighted Abstract's Bag of Words	TFIDF-weighted bag of words for importance scaling	float
GloVe	Average GloVe vector embedding of abstract	float
GloVe + TFIDF	Abstract Bag of Words with TFIDF weighting	float

4 Methods

Kang, et. al.[2] trained and tested a logistic regression, SVM, boosting, and a single layer fully-connected neural network based on the extracted features. We first reimplemented their models, and then examined them more in-depth by tuning each model's hyperparameters. We then opted to train a random forest to observe its performance on the classification problem.

Here are all the models that we explored:

- **Logistic regression with L2/L1 regularization** with the regularization hyperparameter λ varied linearly from $[0, 1]$ over steps of 0.1. Kang et. al only examined the set $[0.1, 0.25, 1]$. In logistic regression, we classify a training example as positive if $h(\theta^T x) > 0.5$, negative otherwise. Regularization helps prevent model overfitting.

$$\min_{\theta} \quad \sum_{i=1}^{m} \left\| y^{(i)} - h\left(\theta^T x^{(i)}\right) \right\|^2 + \lambda \left\|\theta\right\|^2$$
$$h(\theta^T x) \equiv \frac{1}{1 + \exp\left(-\theta^T x\right)}$$

- **Random Forest**: We varied the number of trees as well as overall depth to prevent overfitting. Random forests are an ensemble method where each tree is fit to a set of bootstrapped training samples. Each tree has high variance, but random forests reduce the overall variance by averaging across all trees in the random forest.
- **SVM with L2 regularization using a RBF kernel** as defined below:

$$\min_{w,b} \quad \frac{1}{2}||w||^2$$
$$\text{subject to} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 \qquad \forall i = 1, \ldots, m$$
$$K(x, \tilde{x}) = \exp\left(-\frac{||x - \tilde{x}||^2}{\sigma^2}\right)$$

- **AdaBoost**: We used 50 weak classifiers as shown in Figure 2. The main idea behind Adaboost is to take a poorly-performing classifier (one that performs above, but close to 50% accuracy), and then feed the mispredictions to another weak classifier. Each subsequent classifier "fixes" the mispredictions of the previous classifiers through a penalty for mispredictions from the previous classifiers. With a long enough chain, this will eventually result in an accurate end prediction.

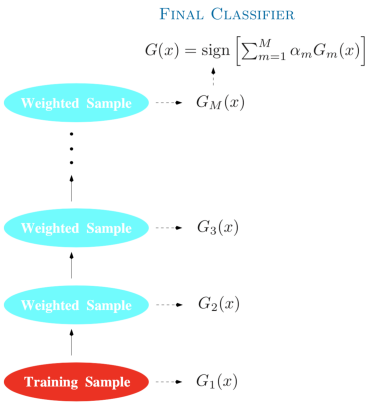

Figure 2: AdaBoost algorithm visualized [1]

- **Fully Connected Neural Network** (Figure 2): We used ReLU ($\max(0, x)$) as our activation function. Kang et. al tried only a single layer with 10 neurons, so we varied the number of neurons from 10 to 100 neurons with a step size of 10, and also repeated with a two layer neural network.

Although a CNN was recommended, we determined it inappropriate based on our features. Typically, convolutional layers are useful for temporal or spatial relationships, such as those that can be found in time series or images. However, our feature set does not include any of these.

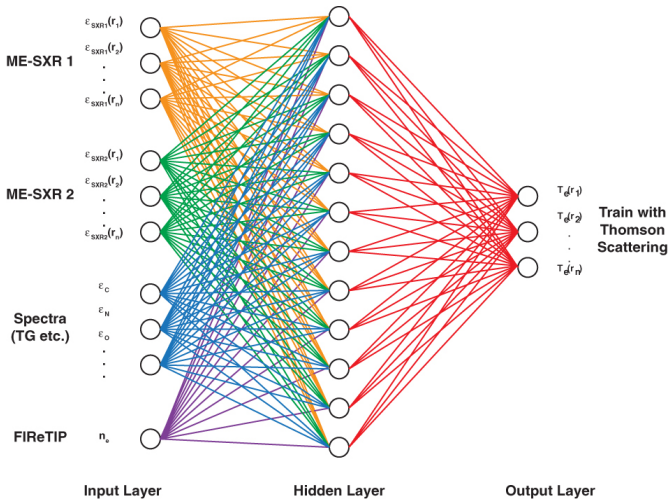

Figure 3: Fully connected neural network diagram [1]

5 Results and Discussion

We used sklearn [3] to implement each model, and used five-fold cross-validation for each model. The result we show in the Table 1 are the models with the best hyperparameter for each model category.

Table 2: Train and test accuracies for the various models used.

Model	Train Accuracy(%)	Test Accuracy(%)
Majority	60.17	60.53
Logistic L2	42.41	42.10
Logistic L1	68.48	68.42
SVM RBF	**72.49**	**71.05**
Random Forest	99.43	63.16
AdaBoost	96.56	50.00
Neural Network	63.04	60.53

For the ICLR 2017 dataset, Kang et. al reports an test accuracy of 65.3% with a 7% standard deviation, but does not report which method.

In our case, we performed the highest performing model was the SVM model with RBF kernel, but we expected the neural network to perform better. One reason for this is that our dataset is relatively small with only 427 samples. Typically, neural networks require at least an order of magnitude larger dataset for good accuracy.

We also note that the AdaBoost and Random Forest models are significantly overfit. Indeed, our experiments showed that Adaboost with 50 weak classifiers is no better than random guessing! For the random forest model, even when we limiting tree depth and number of trees, we still observed some overfitting with even poorer accuracy.

We also used PCA to visualize to the ICLR dataset given our feature set in an attempt to understand the relatively poor classification accuracies that we observed. In Figure 3, we see that our current feature set does not effectively distinguish between accepted and rejected papers. From this, it does make sense why model classification accuracies are not much better than random guessing. This is good news - this implies that a conference paper's contents are the driving factor for acceptance or rejection, exactly how the peer review process should function.

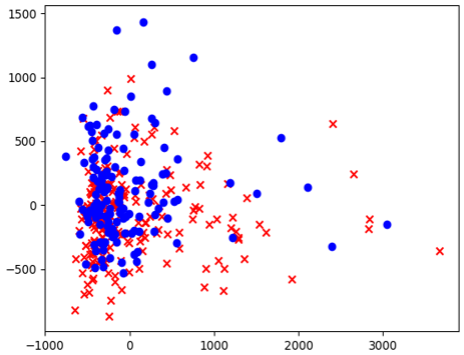

Figure 4: 2D visualization of the ICLR 2017 dataset, where blue dots are accepted papers and red x's are rejected papers.

6 Future Work

Our work focused on the ICLR dataset, which has limited examples. Similar studies can be done on other conferences with more submissions, like NIPS, or for the same conference but with submissions across years. One interesting experiment we could try is to use our trained model from one conference to predict the acceptance of a paper for another conference. This will tell us the preference of different conferences. If two conferences are looking for similar values, the model should provide an equally well prediction result. However, it could also be the case that the model performs poorly, from which we can conclude that two conferences prefer different styles of papers.

To improve classification accuracy on the current ICLR 2017 dataset, we need to employ NLP techniques to extract features to represent the core paper content. We can also extract additional features, such as figure and table content, but this will require additional modifications to the science-parser tool.

A significant issue that we ran into was the lack of labels for papers. Kang et.al artificially expanded their training set through a set of heuristics using review comments as well as looking for references to an unlabelled paper from a known, published paper. We believe that a semi-supervised algorithm (e.g. semi-supervised EM) can potentially use these unlabelled papers to improve predictions.

7 Contributions

- William (wjen) worked through the public PeerRead dataset and code and set up the necessary infrastructure to process the raw dataset. He also verified Kang, et. al.'s results on their prediction methods, and helped implement the models.
- Shichang (shichang) examined the data in depth, performed PCA visualizations to better understand the data, and gave recommendations based on the data. He also provided insight into feature extraction, as well as background on the AdaBoost algorithm, which was not covered in depth during class.
- Muyun (muyunc) worked on feature extraction, and worked with GloVe to gain a better understanding of the framework. She also implemented the various models when we met in person on Shichang's laptop.

Finally, all members collaborated in writing the final report. The repository can be found here: https://github.com/collielimabean/PeerRead

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[2] Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA, June 2018. URL `https://arxiv.org/abs/1804.09635`.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.