Yggdrasil:
Natural speech learning with random forests

Joseph D. Romano*[1] and Alexandre Yahi†[1]

[1]Departments of Biomedical Informatics, Systems Biology, and Medicine, Columbia University

May 5, 2016

1 Introduction

Natural language processing and speech recognition comprise one of the largest applications of machine learning and data mining techniques, spanning diverse industries such as economics, healthcare, information retrieval, and personal computing [4]. One famous speech recognition dataset — the HCRC Map Task Corpus — was created in 1992 for the purpose of classifying short discussions between two people about directions on a map[1]. We performed binary classification on a feature-mapped version of this dataset, demonstrating that classification tasks can perform well in spite of a lack of much semantic knowledge of the underlying data. This project, named "Yggdrasil", was used as a submission for a Kaggle-in-class predictive modeling competition focused on binary classification of this dataset.

2 Description of provided data

The original input we were provided with consisted of 126837 data records and 52 features, 36 of which were numeric (in some cases, binary) and 16 categorical. Each categorial feature had a variable number of possible values. It is unclear what each feature vector represents, but the labels corresponding to some of the categorical feature vectors provide clues as to their meanings. For example, the categorical feature vector labeled "26" includes terms such as `vacknoweldge_acknowledge`, `vacknowledge_explain`, and `vacknowledge_clarify`, suggesting that it encodes the type of response given by one of the two parties holding the conversation. Other features seem to encode information about prepositional phrases, object comparisons, negation, and others. Some further information about the original encoding of the data is available in[1].

3 Learning approach

Our optimal learning approach for the binary speech classification problem involved two main steps:

1. Encoding all categorical feature vectors as sets of binary vectors via one-hot expansion.

*jdr2160@cumc.columbia.edu
†ay2318@cumc.columbia.edu

2. Training a random forest classifier and predicting over unlabeled data.

Reaching this strategy required a trial-and-error process where we performed cursory analyses on the dataset with a number of popular classifiers, as described below.

3.1 Data preprocessing and feature design

One noteworthy characteristic of most random forest classifiers is that they are unaffected by scaling, centering, and other monotonic transformations, since all operations on the data are simply linear partitioning (although it is possible to design non-linear decision boundaries for partitioning the data, doing so is generally unnecessary with random forests).

In order to adapt our data to the available random forest algorithm, we performed one-hot encoding to expand all categorical features. This preprocessing step transform a categorical feature with n possible values into n distinct binary features where only one of them can be positive for each learning instance.

For consistency purposes, we performed the feature expansion with both training and quiz dataset to keep track of the expansion order of the features and make sure we kept track of the final feature list. We then separated the datasets back and split numerical and expanded binary features for each set.

3.2 Model description

Random forest is a meta-algorithm from the family of ensemble classifiers that utilize bootstrap aggregation ("bagging" classifiers). Its goal is to perform bootstrap sampling of the dataset to train a large number of decision trees, the output of which is a "forest" of decision trees whose weighted vote on a new datapoint will, in expectation, have a substantially lower error rate than the prediction of any single tree[3]. Although random forest can accept any type of data (e.g., ordinal, nominal, or continuous), the implementation of the random forest algorithm in scikit-learn only accepts numerical input, hence our decision to encode the features via one-hot encoding. Following cross-validation, we decided to build a random forest classifier with 600 estimators, which offered a good compromise between both computing-time and performance. We noticed that the performance of the classifier tended to plateau following approximately 500 trees, and even begin to perform marginally worse after about 1000 trees. The quality of the split of each tree was assessed with the Gini impurity function, which measures the frequency of an incorrect label given a new datapoint that is assigned a label at random from the distribution implied by the tree partition[5]. The stopping criterion was that nodes would be expanded until all the leaves are pure (i.e., all trees are "completely formed" — no futher partitioning is necessary).

3.3 Model selection

For the initial exploration of the data, we set up a pipeline with 10-fold cross validation to get a sense of which learning algorithm would perform the best. We found that random forest made decent predictions when trained on the numerical features alone (accuracy of 0.893). compared to MLP 0.74). An SVM (Support Vector Machine) classifier, although it may potentially perform quite well given the type of classification task, was not appropriate for this dataset due to the massive number of features. We attempted to perform kernelized and non-kernelized SVM, but in each case the dataset was far too large to run the algorithm in a reasonable amount of time. LDA and other linear methods performed even more poorly — yielding about 65% accuracy on the data set in our exploratory analysis of the data. We also attempted to use AdaBoost (a boosting ensemble classifier) to see how it compared to random forests, and saw that it also performed suboptimally. We feel this may be due to overfitting, given the massive number of features after expansion into the encoded feature space.

4 Results

4.1 Predictor evaluation

As described above, our final classifier entailed one-hot encoding of categorical features followed by a standard random forest classifier of 600 trees to predict the binary labels on the quiz dataset. We randomly split the training data into a training set and a holdout set (67%/33%) to evaluate the performances and tune our random forest model parameters. We performed a 10-fold cross validation on the hold out set.

4.2 Results of evaluation and analysis

For the evaluation of the random forest method we picked, we had to tune the number of trees (estimators) per iteration. In order to select empirically the best number of estimators based on the labeled set available, we computed the cross-validation score, accuracy, average precision, F1 score, precision, recall, and the area under the receiver operating characteristic (ROC) curve, which is the estimated true positive rate vs. false positive rate given by the out-of-bag decision function computed as part of the random forest training algorithm. AUROC values above %50 indicate performance better than random assignment of labels, with %100 implying perfect accuracy.

Estimators	CV score	Accuracy	Average Precision	F1 score	Precision	Recall	AUROC
5	92.99	92.98	94.01	91.86	93.46	90.31	92.67
10	93.63	93.64	94.96	92.48	95.96	89.24	93.15
50	94.21	94.21	95.42	93.18	96.32	90.24	93.77
100	94.14	94.14	95.35	93.10	96.20	90.20	93.70
250	94.22	94.22	95.43	93.19	96.35	90.23	93.78
500	94.25	94.25	95.45	93.23	96.33	90.32	93.82
600	**94.22**	**94.22**	**95.44**	**93.19**	**96.39**	**90.19**	**93.77**
750	94.22	94.22	95.43	93.20	96.33	90.26	93.79

Table 1: Performance of random forest classifer for a number of different estimators. In all cases, the trees were fully resolved and partition quality was determined at every split via Gini impurity. AUROC was estimated using the decision function given by performing out-of-bag estimation on the resultant classifiers. All other scores were determined via cross-validation. The selected model is shown in bold. 600 trees resulted in a classifier that performed best on the quiz data set.

Our final selected classifier reliably predicts the label for new data with approximately %95 accuracy. This is within 1 percentage point of the highest attained accuracy by any group in the Kaggle-in-class competition, validating the strength of our approach in spite of its relative simplicity.

5 Discussion

This dataset highlights several important concepts in machine learning. First, in agreement with the No Free Lunch Theorem, we can see that there is no perfect learning algorithm that performs flawlessly on any given real-world dataset[6]. In order to discover the optimal classifier, we manually selected and tested a number of different classifiers. Linear classifiers (such as LDA and single-layer perceptron) did not perform well at all, implying that there is not a good linear decision boundary separating the two classes. SVM is highly sensitive to specification of appropriate tuning parameters, and selecting the appropriate kernel function to implictly

map the feature space to a more amenable dimensionality. After discovering good performance with random forests, we tried other powerful methods for completeness. One of these is Multi Layer Perceptron — a neural network approach that did not perform well at first with a quasi-Newton method (0.74 accuracy) and plateaued at 0.93 accuracy once we selected a stochastic gradient-based optimization algorithm[2].

Random forest is widely praised for its resistance to overfitting, something that we saw quite prevalently in our application of a random forest classifier. Random forests do not require centering and/or standardization, due to its reliance on decision trees where partitioning is not relevant to the ability of the partitioning operation to reliably separate the dataset into two groups. This is in stark contrast to many other machine learning algorithms, where normalization is one of the most important central tasks to improve the performance of the model.

It is also important to note that because we were using scikit-learn, one of the few packages allowed for this challenge with Python, we were limited by the fact that all its methods only accept numerical features. This led us too perform a one-hot expansion that resulted in a dramatic increase in features.

We feel that, given the complex nature of speech data and the poor performance we observed with linear classifiers, employing convolutional neural networks would be the logical next step for improving the performance of Yggdrasil. This, unfortunately, was not feasible to attempt at this point, given that neural networks are not represented in allowed library code that was available to us, and that given the nature of the competition we did not have the time to build a convolutional neural network classifier from scratch.

6 Conclusions

Random forests have become flexible and highly popular methods for performing a wide array of classification and regression tasks. Here we demonstrate the utility of random forest — along with appropriate encoding of nominal features — for classifying data from a classical natural speech recognition dataset.

6.1 Naming "Yggdrasil"

Yggdrasil is, in Norse mythology, an immense mythical tree connecting the nine worlds of the cosmos. It plays an important role in numerous Norse epics, including the *Poetic Edda*, the *Prose Edda*, and others. There are many alternative names for Yggdrasil, but the most common translation equates roughly to "gallows". Our choice of "Yggdrasil" as the name for this project comes from numerous claims that "decision trees will be the death of us".

7 Author contributions

JDR and AY jointly performed model selection and designed the final model. JDR wrote, formatted, and documented the source code for Yggdrasil. JDR and AY both participated in analysis of the results and submissions to the Kaggle competition. AY prepared the manuscript. JDR and AY edited the manuscript.

References

[1] A H Anderson, M Bader, E G Bard, and E Boyle. The HCRC map task corpus. *Language and Speech*, 34(4):351–366, 1991.

[2] M G Bello. Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks. *IEEE Transactions on Neural Networks*, 3(6):864–875, 1992.

[3] L Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] E Cambria and B White. Jumping NLP curves: a review of natural language processing research [review article]. *Computational Intelligence Magazine*, 9(2):48–57, 2014.

[5] Johannes L Grabmeier and Larry A Lambe. Decision trees for binary classification variables grow equally with the Gini impurity measure and Pearson's chi-square test. *International Journal of Business Intelligence and Data Mining*, 2(2):213, 2007.

[6] D Wolpert. *No free lunch theorem for optimization*, volume 1. IEEE Transactions on Evolutionary Computation, 1997.