Columbia Unviersity

COMS4721: Kaggle Competition Project

Team: The Weak Learners

Group Members: Daniel Kost-Stephenson,Ryan Walsh, Bob Minnich

Group UNI: dpk2124, rpw2120, rcm2165

May 5, 2016

Contributions: All group members of this project contributed equally to data exploration, feature selection, model selection and writing the report.

Introduction

The objective of the project is to create a binary classifier from a training dataset that minimizes the 1/0 error rate on an unseen test set. The dataset contains both numeric and categorical features that corresponds to dialogue between two individuals and the label we are seeking to predict is whether or not that dialogue led to successful cooperation. Our process involved: preprocessing and computationally eliminating features, training and tuning a selection of classifiers, estimating the test error rate of our classifiers and selecting the best one.

Data preprocessing and feature design

Before being able to run any models, the data required preprocessing. The dataset contained many categorical features which many models, such as those based on distance, can't support. We used one-hot encoding which expanded the number of dimensions from 53 to 7616. We had some concerns about how some models might be affected by this expansion. With so many features relative to the number of observations, the variance of the models could be high, and we believed that many of the newly created features would not be very useful. For example, with random forests using $\sqrt{d}$ features at each split, only 87 features would be randomly selected at a time. If only 100 are useful and the rest is noise, then each split would only have one useful feature on average, and 32% of the splits would contain no useful feature. Much of our remaining feature design focused on reducing the dimensionality of the model. Our first attempt was to apply principal components analysis. This turned out to perform very poorly on this dataset. It still required most of the features to retain 90% of the variance in the dataset. Because most of the features were generated from the one-hot encoding of just a few categorical features with high cardinality, the features were already largely orthogonal. We then turned to methods that fully drop features. We used two embedded methods that use ensembles of classifiers to infer feature importance. The first used randomized logistic regressions which creates an ensemble of lasso-regularized logistic regressions from bootstrap samples with random scaling of features. The importance of a feature is determined by the proportion of models where the feature has a non-zero weight. The other method, which we used in our final model, was using the feature ranking from random forests. This method calculates the average decrease in node impurity across all trees in the ensemble for each feature, often called the Gini importance. This method gives preference to features that are continuous or high cardinality, but because our features are mostly binary, this is not an issue.

Model/algorithm description

Support Vector Machines

Support Vector Machines (SVM) was chosen for several reasons

1. Linearly separable data due to mostly binary features

2. Ability to apply different kernel methods (RGB, Polynomial)

Kernels would allow the models to learn from non-linear data using RGB Kernels or create interaction terms using Polynomial kernels. Interaction terms were though to be useful because it may be that having relationships between words could be more useful as features than just words by themselves. The polynomial kernels were appended onto the initial, important features selected to create an additional 20 features.

Objective Function:

$$\begin{aligned} \underset{w,b,\zeta}{\text{minimize}} \quad & \frac{1}{2}w^T w + C\sum_{i=1}^{n}\zeta_i \\ \text{subject to} \quad & y_i(w^T\phi(x_i)+b) \geq 1-\zeta_i \qquad \zeta_i \geq 0, i = 1, ..., n \end{aligned} \tag{1}$$

Rearranging we come up with $\zeta = 1 - y_i(w^T\phi(x_i)+b)$ which is our *hingeloss*. This can be substituted in to get a simplified equation

$$\underset{w,b,\zeta}{\text{minimize}} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{n} l_{hinge}(y_i, w, x_i, b) \tag{2}$$

While the general Support Vector Classification has a large computation time *sklearn* that the $SGDClassifier$ was used in order to speed up computation of the SVM using Stochastic Gradient Descent with *Hinge Loss* to use Linear SVM.

Overall only a second degree polynomial had an improved performance for the SVM's cross validation, which was still poor in comparison to Random Forests.

Table 1: CV Results using 5Kfold Cross validation

Poly	**1**	**2**	**3**	**4**	**5**	**10**
Accuracy	0.7222	0.8113	0.7038	0.5327	0.5902	0.6569

Random Forests

Random Forests (RF) was investigated for several reasons

1. Handles noise within the data through averaging results of all randomly selected features within trees

2. Can use RF feature importance to remove irrelevant features and improve model performance

3. Reduced variance through voting process allows for a more stable model and less prone to overfitting

Objective Function:

$$\underset{G(\theta,Q)}{\text{minimize}} \;\; G(Q,\theta) = \frac{n_{left}}{N_m}H(Q_{left}(\theta)) + \frac{n_{right}}{N_m}H(Q_{right}(\theta)) \tag{3}$$

where $H(X_m)$ is the loss of the function and can be either selected as Gini, Entropy or Misclassification.n_{left}, n_{right} are the data points within the each branch. Q_{left} and Q_{right} are the subsets of data within each branch and N_m the total amount of points at the current split. The cross validation results plateaued around 94%, so we sought to further improve the model we had already established.

Model/algorithm selection and Predictor Evaluation

Improvements to the Random Forest Model

In our early exploration of the data, we constructed an ensemble method of logistic regression, gradient boosting and random forests only to find that the ensemble model did worse than the base random forest. We hypothesized that a possible explanation could be that all three models were incorrectly classifying the same set of points. To confirm this theory, we wanted to see if we could determine which points the random forest model was classifying with less uncertainty. In order to determine which samples the random forest was having difficulty with, we created many single trees using random forests in *sklearn* so that we could see how each tree would vote. By averaging over all of the votes, we could see how close the prediction was to zero and thus creating this notion of uncertainty for each sample. Below is a bar chart depicting where possible problems occured.

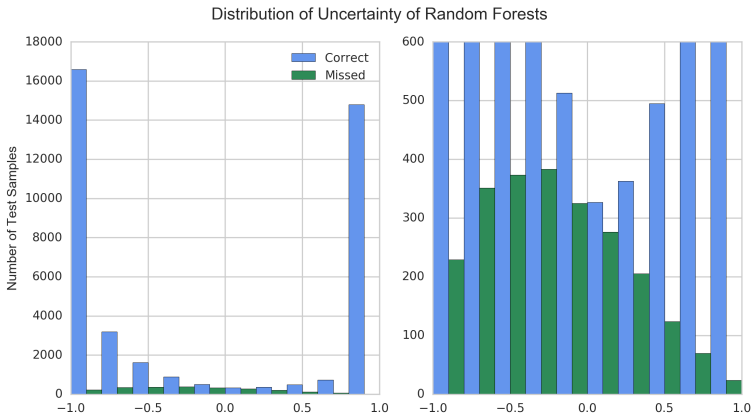

Figure 1: The y values are the predictions [-1,1] that are averaged over all 500 trees for each test sample

K-Nearest Neighbors

Although the random forest seemed to be the best classifier for this particular data set, we thought that perhaps another type of classifier would be more suited to classify points where the random forest was "uncertain". After training a second model, we decided to set a temporary uncertainty threshold (± 0.2) to determine whether the second model would outperform the random forests on uncertain test observations (inside the threshold). Using cross validation, we determined which model was the most accurate for the uncertain observations and found that K-nearest neighbors outperformed adaptive boosting and logistic

regression. By lowering the threshold, we found that there was a point where KNN and random forests had approximately the same performance, with random forests being more accurate when the threshold was widened and KNN being more accurate when the threshold was narrowed. Here we minimize the number of misclassifications with our KNN function f by choosing the correct number of nearest neighbors, K.

Objective Function:

$$\underset{K}{\text{argmin}} \quad \sum_{x \in X} \mathbb{1}[\hat{f}(K, x_i) \neq y_k] \tag{4}$$

To determine an appropriate uncertainty threshold we used cross validation and found that ± 0.12 seemed to yield the most favorable results, although there was significant uncertainty in this department.

K-Means

$MiniBatchKMeans$ was used from *sklearn* which uses the standard objective function for solving KMeans. $MiniBatchKMeans$ reduces running time by taking a small random sample from the data to update clusters instead of using all data points at each update. K-Means was chosen to try and find relationships between samples that might not be represented in our initial dataset, thus adding a relevant feature that would allow the random forest to make better decisions.

Objective Function:

$$\underset{u_j \in C}{\text{minimize}} \quad \sum_{i=0}^{n} ||x_i - u_j||^2 \tag{5}$$

where u_j is the mean for cluster C.

After adding this new feature, we trained both the random forest and K-nearest neighbor models on the entire training set (instead of using 5-fold cross validation) and submitted the results. Although the scores obtained by using 5-fold cross validation were comparable to the stand-alone random forest model, our combination of KNN and random forests yielded our best quiz score of 0.95188.

Figure 2: 5K-Fold Cross validation results

Other improvements

Lastly, we wanted to see if we could somehow tune the parameters in the default random forest to lift its performance on the data set. Our team ran an exhaustive grid search (with 5-fold cross validation at each iteration) to determine the optimal random forest parameters. We focused on tuning the main parameters: the maximum number of samples for each tree, the minimum number of samples to make a split and the minimum number of samples required to keep a leaf node. Unfortunately, our grid search only confirmed that the default parameters of the original random forest classifier yielded the best cross validation results.

Results of evaluation and analysis

Our scores on both the private and public test set outperformed our 5-fold cross validation error by about .005, likely because training on the full data set gave us 20% more observations to learn from. Our final score on the private test set was .95105 which was slightly below the score we achieved on the public test set of .95188 for the same model. Scoring below the public test set would suggest overfitting our model to the public test set by using feedback to tune or change our model. Most of our model evaluation was done through cross validation rather than using the public test set, making this unlikely. Additionally, the drop in performance of .0008 seems to be small enough to be caused by random variation. If our true accuracy is .951, then the standard deviation of our error rate would be .0012 on a random test set of 31709 observations and in 5-fold cross validation, the performance of the model varied by more than .002 between the best and worst folds. Thus, we believe the drop in performance is too small to suggest overfitting.

Conclusion

The results of the quiz scores and the final scores closely matched our cross validation results. For future improvements on the model we could look at adding more features, in particular the interaction terms that helped improve the SVM scores could also improve our RF scores.