CIS520 Report: ML Crackers

Francine Leech, Ziyin Qu, Chen Xiang

December 12, 2016

1 Introduction

The rise in web and mobile based social networking has opened a stream of continuous text data. These data reflect the sentiment of individuals and masses of people. Understanding the sentiment of users is relevant on the most basic level, understanding how people are responding to a stimuli, and extending to human machine interaction systems. The ambiguity of language and emotional expression creates an interesting machine learning problem. We try to tackle one aspect of this problem by classifying tweets into two categories, joy and sadness.

2 Preliminary Methods

When starting the project, we thought that we should first experiment with different classification methods on image data like train_color, train_img_prob because they were smaller datasets. We made an assumption that to some extent, the image data may contain some information about people's sentiments. For example, lighter and bright colors represent joy, darker colors represent sadness.

Because the image data alwats contains a lot of features, so the method we use for image data is PCA and Gaussian Mixture Model. With PCA, we can reduce the dimensions of features, with GMM, if it works, we may cluster the trainning data into two clusters, joy and sadness and thus get the trained classifier.

For the trainning, we tried three datasets for PCA and GMM, that is train_color, train_img_prob and train_cnn_feat. However, the results are not satisfying. The table below demonstrates the trainning accuracies with PCA and GMM on the three datasets.

	train_color	train_img_prob	train_cnn_feat
PCA and GMM	59%	58%	59%

We did not use cross validation for tunning the parameters becasue the trainning process will take a large amount of time and instead we used 4000 data to train and 500 hold out to test. We trained the image data in this way. We trained two GMM respectively for joy and sadness and each GMM has 5 clusters. For the PCA, we tried different numbers principle components for the three dataset but the results did not improve much.

We think the reason why PCA and GMM did not work on image data may because the image data itself does not contain enough information for this method to get a classifier, unlike other classical clustering problems like human face recognition or male and female recognition. The accuracy is not enough to beat the baseline 1, so we have to try different methods.

Dimensionality Reduction

We tried to reduce the dimensionality of the $word_train$ data using Principal Component Analysis to determine if we could classify the data with a PC-ed version of the training data using a Gaussian Mixture Model. We used a 'holdout' of 20% of the train data to use as testing because cross validation with PCA took too long.

$$GraphwithPCA/GMMerror/accuracy$$

We found that even with

3 Main Methods

3.1 Naive Bayes

For supervised learning, Naive Bayes method is a very good generative method on classifying texts, like spam classification problem. Actually the words_train data set uses bag of words model where counts of words matters and position of words does not matter. There are lots of advantages for Naive Bayes model, it can be a dependable baselien for text classification, it trains fast. Although the assumption of Naive Bayes which is the Conditional Independence Assumption may not be true, it may still work well on text classification problem.

To use the Naive Bayes model on words_train dataset, we use the built-in matlab function fitNaiveBayes. For trainning data, we sue 9-fold cross validation error to estimate the test error for Naive Bayes model, which means we randomly choose 4000 data to train and 500 data to test. For the built-in function fitNaiveBayes, there are some parameters for us to choose, for the distribution parameter, because we are using the bag-of-words model, we use the multinomial distribution in the fitNaiveBayes function.

The Naive Bayes model actually works really well. We got around 0.8 cross validation accuracy on trainning data. And we got 0.7962 accuracy for the test data. We successfully beat the baseline1 using a simple Naive Bayes model with multinomial distribution.

But there are still problems with the simple Naive Bayes model. For example, the dataset matrix for words is very sparse, and for each observation there are many words did not show up. Naive Bayes model for text assumes that there is no information in words that are not observed and this may cause overfitting. We can solve this by smoothing the Naive Bayes model.

3.2 GentleBoost

The second main method we utilized was an ensemble method.

Benefits of ensemble methods

We used GentleBoost, a weak learning that was built by MATLAB under the fitensemble function. The method combines many weak learners into one high quality ensemble predictor. We chose this ensemble methods over the others offered by MATLAB, because it is preforms well with binary classification trees with many predictors (ensemble citation).

The input of the model was the $word_train$ data. We used a 10-fold cross validation method to observed how the model preformed, specified the use 300 learners, and the type of learner as 'tree'. The average cross validation error was 0.21. The algorithm classified joy and sadness well.

The method could have improved if we increased the number of learners, however it would have taken a very long time to train because the data is large. Initially we tried the method with the default number of learners, 100 trees, and found that the cross validation accuracy only improved slightly. This slight improvement with triple number of learners reveals that the data has some intricacies or patterns that the ensemble method cannot learn.

3.3 Support Vector Machine

Support vector machines (SVMs) proved to the most promising method to classify the data. We used the MATLAB function, fitcsvm, to train an SVM model for binary classification on the on the $word_train$ data.

We tried a simple SVM by specifying a linear kernel, and had a cross validation error was 0.2180. With a Gaussian or RBF kernel we had an error of 0.4373.

After experimenting with a variety of kernels, we found that the linear preformed the best. fitsvm allows you to make an assumption about the fraction of outliers in the data. While we could have gone through the raw tweets and looked through the data, we decided to experiment with 10%, 20%, and 30% and observed cross validation errors 0.2121, 0.2282, and 0.2131 respectively. Specifying the outlier percentage did not have an effect on our cross validation error, so we decided not to specify in our SVM final model.

Lastly we optimized our SVM by using MATLABs built in method to optimize a cross-validated SVM using Bayes Optimization (citation). The method originates from The Elements of Statistical Learning, Hastie, Tibshirani, and Friedman (2009). Paraphrasing from the MATLAB documentation, "the model begins with generating 10 base points for a "green" class, distributed as 2D independent normals with mean (1,0) and unite variance. It then generates 10 base points for a "class" that is also distributed as 2-D independent normals with mean (0,1) and unit variance. For each of the classes, it generate 100 random points by choosing a base point, b, of the respective color uniformly at random. It then generates an independent random point with 2-D normal distribution with mean b and variance I/5, where I is the 2-by-2 identity matrix. After 100 points for each of the colors has been generated, the point are classified using fitcsvm. The function bayesopt is used to optimize the parameters of the final SVM model with respect to cross validation." We submitted the method to the autograder, and it had an accuracy of 0.7991. The method was accurate enough to beat Baseline 1, but not Baseline 2. Similar to the other methods above, the optimized SVM may not preform well because the data was sparse and high dimensional, so the hyperplane could not separate data well.

4 Final Method

Ensembl: Sentiment Analysis 1 + Methods
(1) First, we use vader package in python to do sentimental analysis on each word shown on topwords list. Vader is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. The input is the every word in topwords list and the output is the probability of negative, positive and neutral emotion that this word may express.
The output often looks like this, when type "funny" we can see {'compound': 0.4404, 'neg': 0.0, 'neu': 0.0, 'pos': 1.0}, which means it is an extremely postive word.
(2) Second, we attach those scores to every word in topwords list. The advantage is that, we can use this preliminary method to choose the extremely positive and extremely negative sentences by assuming that there are no extreme positive words shown in negative sentences and vise versa.

Ensembl: Sentiment Analysis 2 + Methods
From the raw tweets we notice that there are some words begin with #, which may represent the topic this sentence belongs to. It is useful since some specific topics always express similar emotions, like #family

usually means positive. So instead of analyzing the sentiment of each word, we can use sentence as the input and in this way get the average emotion scores of each word.
(1) First, do sentimental analysis on each sentence in raw tweets. For all the words that appear in this sentence, attach the result score to this words.
(2) Second, for every word we can get the average emotion score based on all the sentences that they have shown.

5 Discussion

- NLP - difficulty with classification
    - Could have used deep learning too long, overfit, dataset small - articles

6 Works Cited

References