Darrell Nelson II

Data Analytics (IST - 707)

05/28/2019

HW08: Lie Detection

Introduction

Lie detection is a major theme in "psychology and law," which is one of the main areas of applied psychology. It is not difficult to understand why it is important to know whether someone is lying or telling the truth in police investigations, court trials, border control interviews, intelligence interviews etc. To aid lie detection, psychologists and practitioners have developed numerous lie detection tools that range from observing behavior, analyzing speech, and measuring peripheral physiological responses to recording brain activity.

Humans ever since the beginning of time humans have tried to sharpen their own senses to detect when someone else is lying. Many claim to be experts and provide social clues and signs as to when someone is lying (e.g. if they look to the left) however, even the most self-established experts are still no better than random guessers based on probability. That means that if you toss a coin in the air you will be as likely to detect deception as the truth no matter how much you train. And while it is true that a very few people are better at detecting deception than others, they are barely above chance. In fact, those that are really good are only correct somewhere around 60% of the time; that means that 40% of the time they are wrong, and you would not like them on jury duty if you are on trial.

For humans there is no magic bullet. Detecting lies is still a mystery that has not been solved despite heavy study. Mainly, because the person detecting the lies does not have enough background information on the behaviors of the accused liar to make a valid deduction. In my estimation, the best person to detect if another is lying is either their mom or spouse. Can machines pick up the mantle that we can't possibly carry? Can machines solve deception and rid us of using juries and subsequently freeing all of humanity of the pain of jury duty?! Read on to find out.

Analysis and Models

About the Data

The dataset used is a comma separated values file (CSV) that consists of 3 columns and 93 rows. The first column labeled 'lie' categorizes each row as either containing a true customer review or a fake one. The second column labeled 'sentiment' categorizes each row as either a positive review or a negative one. The final column labeled 'review' is text data that contains the actual review for each row/sample of data.

The dataset was then imported into Rstudio and a corpus was created for the text in the 3rd column. The corpus had 92 documents, each one corresponding to a different review. The corpus was then turned into a document matrix where every column represented a different word that was found in the entire

corpus. The values under each word was their frequency in that row of text. It was this final document term matrix that was used in the analysis below.

Classification Models

Naive-Bayes

A classification algorithm used for supervised machine learning is Naïve-Bayes. This technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. It uses joint and conditional probabilities to draw relationships between the inputs and outputs. A Naïve-Bayes classifier estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label. While this assumption is somewhat violated in this case, proper sentence structure is predicated on the words coming before and after in that sentence as well as in sentences above and below. For example, it would be considered an error to change verb tense in the middle of a paragraph.

Support Vector Machine

A support vector machine (SVM) is a classification technique that can be used to run supervised machine learning also. This model has its roots in statistical learning theory and has shown promising empirical results in many practical applications. SVMs also work well with high-dimensional data and avoid the curse of [the] dimensionality problem. Another unique aspect of this approach is that it represents the decision boundary using a subset of the training examples, known as the support vectors.[1] The decision boundary is chosen by maximizing the distance between the datapoints within a cluster and the boundary itself without sacrificing accuracy. This increases the chance of properly labeling unknown test data they may reside near the boundaries. Two different types of test boundaries were chosen for this dataset: 1) polynomial and 2) linear. The cost parameter (cost of constraints violation) was held constant.

Results

A generic naivebayes algortithm from the e1071 R package was implemented to predict which reviews were true or fake. The table below shows the actual labels (columns) vs the predicted values (rows):

```
> NB_e1071table
                LIE
NB_e1071_Pred    f  t
             f 17 26
             t 29 20
```

The overall training accuracy of this model was 40%, which is even less than the 50/50 chance given if the outcome of the review was guessed (either false or true). In order for a model to be considered descent it has to at least be able to produce higher odds of accuracy than random chance. The accuracy, precision, and recall for this model were all below the probability of random chance as well.

[1] (Tan, Steinbach, & Kumar, 2006)

Precision in this context is the proportion of true positive identifications that were actually correct for a particular class label. Precision answers the question: Out of all of the predicted labels for this class, what percentage of them were actually correct.

Recall in this context is the proportion of actual positives for a given class that were identified correctly. Recall answers the question: Out of all the correct labels for a particular class, how many of them were properly predicted?

Parameter Setting	Overall Accuracy	Precision on fake reviews	Recall on fake reviews	Precision on true reviews	Recall on true reviews
Generic NB	0.4021739	0.3953488	0.3695652	0.4081633	0.4347826
Poly SVM cost=100	0.53260870	1.00000000	0.06521739	0.51685393	1.00000000

As seen in the chart above, a SVM was used with a polynomial kernel and a cost setting of 100 to predict lies.

	LIE	
SVM_lie_pred	f	t
f	3	0
t	43	46

The SVM results showed increases in accuracy, precision, and recall however, this increase in performance must be taken with a grain of salt. The confusion matrix above reports that the SVM predicted that 89 of the 92 reviews were true. Neither of these models are robust enough to help in lie detection.

More data munging is needed to create a model with a better results. There are 1,736 unique words that were tested in the original dataset. For only 92 reviews, with reviews rarely consisting of over 100 words a piece this vast range in vocabulary is too much. In order to reduce the number of words in the matrix, all stopwords are going to be removed. Stopwords are words in language that are place holders and do not carry critical information on their own. Words such as: it, the, because, etc. Punctuations and number will also be removed as they are deemed to have little value in assisting the prediction of false and true customer reviews. Stemming is also enabled to minimize the amount of redundancy in the table. For example, three columns for 'help', 'helped', and 'helping' are unneeded and do nothing but dilute the weights of the words in the matrix. Instead, by counting the root word 'help' it allows for reviews to have higher counts of any individual word and creates a better distinction between reviews. Lastly, any words that were not removed but are insanely rare or widely used are also discarded. This new clean dataset has 1,128 unique words in it. A 35% reduction!

Re-running the models on this dataset resulted in a 26% increase in accuracy for the NB model. The reason for this increase is the model's improved ability to predict false reviews which went from 17 to 37. Increasing the Laplacian feature of this model from 0 to 10 to 100 did not affect the confusion matrix at all. An increase in the Laplacian numbers allows for smoother boundary functions.

	LIE	
NBc_e1071_Pred	f	t
f	37	22
t	9	24

The SVM performance with the newly cleaned data had no significant impact. Inspecting the confusion matrix revealed that the model just flip-flopped from predicting all answers to be true to predicting that all of the answers are false (What a lazy algorithm!).

	LIE	
SVMc_lie_pred	f	t
f	46	45
t	0	1

Increasing the cost function to 1,000 which tightens the SVM's class boundaries still produced a confusion matrix where the model predicted all reviews to be of one type. How about 10,000?

	LIE	
SVMc_lie_pred	f	t
f	39	0
t	7	46

Eureka! The SVM model now boasts an accuracy of 92%. It appears that a 100-fold increase in the cost parameter finally made the model give a variation of responses. It will be interesting to see how this model performs under test conditions, as high cost functions generally over-train the model.

Parameter Setting	Overall Accuracy	Precision on fake reviews	Recall on fake reviews	Precision on true reviews	Recall on true reviews
Generic NB	0.6630435	0.6271186	0.8043478	0.7272727	0.5217391
Poly SVM cost=100	0.51086957	0.50549451	1.00000000	1.00000000	0.02173913
Poly SVM cost = 10,000	0.9239130	1.0000000	0.8478261	0.8679245	1.0000000

The exact same models run with the sentiment labels yielded the following results. The NB algorithm on the unclean data operated poorly and only predicted positive reviews.

	SENT	
NBSent_e1071_Pred	n	p
n	0	0
p	46	46

What about the SVM?

	SENT	
SVM_sent_pred	n	p
n	3	0
p	43	46

As expected, why should I assume that the lazy algorithm would perform any better.

Increasing the cost to 10,000 produced valuable results. Again, this model must be verified by test data since there is a large chance of over-fitting by raising the cost function.

	SENT	
SVMSentc_pred	n	p
n	30	0
p	16	46

Parameter Setting	Overall Accuracy	Precision on negative reviews	Recall on negative reviews	Precision on positive reviews	Recall on positive reviews
Generic NB	0.5	NA	0.0	0.5	1.0
Poly SVM cost=100	0.53260870	1.00000000	0.06521739	0.51685393	1.00000000
Generic NB-clean data	0.5	NA	0.0	0.5	1.0
Poly SVM cost=100 clean data	0.51086957	1.00000000	0.02173913	0.50549451	1.00000000
Poly SVM cost = 10,000 clean data	0.8260870	1.0000000	0.6521739	0.7419355	1.0000000

Comparing the sentiment results to the lie detector results draws a clear conclusion. The task of lie detection is something a computer can have much more success with than comprehending sentiment.

Taking the top performing models for NB and SVM and running them on testing data for lie detection and sentiment analysis yields the below chart:

Parameter Setting	Overall Accuracy	Precision on fake reviews	Recall on fake reviews	Precision on true reviews	Recall on true reviews
Lie Detector: Generic NB clean data	0.54838710	1.00000000	0.06666667	0.53333333	1.00000000
Lie Detector: Poly SVM cost = 10,000 clean data	0.53260870	1.00000000	0.06521739	0.51685393	1.00000000
	0.45161290	0.43333333	1.00000000	1.00000000	0.05555556

NBc_e1071_Pred	f	t
f	1	0
t	14	16

SVMc_lie_pred	f	t
f	2	2
t	13	14

For Sentiment analysis it appears that those good results were from over-trained data. The testing confusion matrix reveals that the model predicted all of the reviews had positive sentiment.

NBSentc_e1071_Pred	n	p
n	0	0
p	13	18

The SVM didn't fair much better:

SVMc_sent_pred	n	p
n	13	17
p	0	1

Parameter Setting	Overall Accuracy	Precision on fake reviews	Recall on fake reviews	Precision on true reviews	Recall on true reviews
Sentiment Detector: Generic NB clean data	0.5806452	NA	0.0000000	0.5806452	1.0000000
Sentiment Detector: Poly SVM cost = 10,000 clean data	0.53260870	1.00000000	0.06521739	0.51685393	1.00000000

It appears to be very difficult to create a high accuracy model for either lie detection or sentiment analysis. The worse of the two is sentiment analysis.

Top 30 words based on Gain Ratio for lie detection:

```
> head(bich, n=30)
     cold       ice     drink     minut     enter      side       tea      wait       add    averag
0.2076351 0.2076351 0.1858327 0.1858327 0.1740352 0.1740352 0.1740352 0.1740352 0.1611196 0.1611196
      big      cafe      case    celebr    chocol     dirti     enjoy   everyon    expens    experi
0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196
      fri       got      help   horribl   kitchen     least     notic     offer  outstand      pack
0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196
```

It's hard to understand what these lies are about without context, but it appears that reviews about drinks were the most important in detecting if a review was a lie or not.

Top 30 words based on Gain Ratio for sentiment analysis:

```
> head(bich, n=30)
     amaz   terribl      took     alway      said     worst      best       eat     bland   consist
0.2486494 0.2486494 0.2283069 0.2180507 0.2076351 0.2076351 0.2022903 0.1969491 0.1858327 0.1858327
    drink     minut atmospher      feel      half   ingredi    review     salad     smile     sushi
0.1858327 0.1858327 0.1740352 0.1740352 0.1740352 0.1740352 0.1740352 0.1740352 0.1740352 0.1740352
     wait     arriv    averag    awesom     began    buffet      cafe    celebr     cheap    chocol
0.1740352 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196 0.1611196
```

For all the failure these models had in predicting sentiment analysis the most impactful words are words that could easily be labeled as good or bad. For example, words like "amaz" which is the root of amazing and amazed probably means a good review and a word like "terribl" which is the root of terrible is probably a bad review. Although, if you add a "not" in front of those words the reviews could take on a different tone entirely.

Conclusion

In conclusion, language is not as easy as it appears. There are a lot of nuances, tonality, and sarcasm that plays a huge role in determining if a person is lying or not. These slight of hands are even more

important with sentiment analysis where the user is determining the emotions of the text. There is so much of our language that we as humans quietly know and understand without realizing, which is what makes teaching it to computers so difficult.

Lie detection appears to be the easier task due to the semantic, discourse, and pragmatic levels of language. In those levels, to best understand the speaker the background information must be understood first. The way verb and noun phrases are tied together also convey a lot of information which this model was unable to capture. The semantic model built here simply counted the words in each review and used word count comparisons to predict if a customer is happy or sad. This is rarely the case. People express themselves differently and use different words to convey meaning. If the model used a sentiment calculator which associates a number with each word based on how negative or positive it is that would greatly increase the model's accuracy.

In all, the models used here were under-manned and under-armed. Language is a lot more complicated than 1s and 0s and in order to build the best predictive models more effort needs to be put into transforming sentences and words into corresponding semantic values. Luckily, there has been a lot of work done in this area to transform our language into simpler categorical and numerical values that a computer can understand. Although the models shown here leave a lot to be desired, hopefully they shed light on what is possible with ML.