User Review Rating Prediction

Sammy Leong, Ashish Bhatia

Nov 15, 2012

Objective

We're given a set of user review data each of which is associated with a user rating. Our object is to build a model that is able to predict whether an unseen review is positive or negative.

Overview

To solve this problem, first we evaluated a number of learning algorithms such as Logistic Regression, Naive Bayes, and SVM on the original raw data as-is to get a sense of their feasibility, capabiilty, and speed. Then we developed our own pre-processing techniques to transform the original data to better emphasize sentimental features, which we evaluate with one of the learning algorithms.

Having developed a good pipeline for preprocessing the data, our next step will be to revisit the learning algorithms and tune them to achieve even better results.

Data Source

We gathered our user review data from the Yelp academic data set which contains user reviews of local businesses near Stanford University.

Each review contains a text sequence representing the user review along with a numerical rating that ranges from 1 to 5. The first thing we did was relabel each review as positive or negative. Reviews with ratings $1, 2$ are deemed negative, reviews with ratings $4, 5$ are deemed positive, and reviews with rating 3 are randomly labeled as positive or negative.

As it turns out, 80% of the reviews are positive which means the data is highly skewed to begin with. To address this, we reshuffle the data such that we have equal number of positive and negative reviews, and that positive and negative reviews are interleaved. We had to throw out a lot of positive reviews but we're still left with abdundant training examples.

Success Metric

Our goal is to find the learning algorithm along with data processing techniques that give us the highest generalization accuracy. To do this, we divided our data in half, trained with the first half, and then tested with the second half. The former gives us the training accuracy while the latter gives us the generalization accuracy.

First Attempt (Baseline)

For our first attempt we focused on trying out various learning algorithms on the original unprocessed data as-is to get a sense of their capability for our given problem. Another goal is to choose the fastest algorithm which gives reasonably good results and use it to iterate while we work on data cleaning and feature selection.

Results

Here is a table that shows the training and generalization accuracies for each learning algorithm, along with the time it took to run.

	Train	Test	Time
Matlab			
Naive Bayes (mn)	91.67	81.27	1.05
Liblinear			
Logistic Regression	97.39	81.67	1.58
L2-reg SVM (linear)	99.69	78.71	2.86
Libsvm			
C-SVM (linear)	98.99	78.86	786
C-SVM (radial)	70.26	69.68	356
C-SVM (sigmoid)	65.11	65.43	359
nu-SVM* (linear)	87.78	83.75	277
nu-SVM* (radial)	88.80	83.91	296
nu-SVM* (sigmoid)	84.78	81.49	291

* $nu = 0.5$

Discussion

Firstly, it's quite clear that liblinear runs significantly faster than libsvm. For this reason, we will iterate using liblinear when we evaluate the performance of various pre-processing techniques. In particular, we will use logistic regression because it has higher generalization accuracy than L2-regularized SVM with linear kernel.

Secondly, it appears that nu-SVM with $nu = 0.5$ in this case improved both the training and generalization accuracies significantly. In fact, the we were able to achieve the highest generalization accuracy using nu-SVM with the radial basis kernel. The only issue is that it takes a very long time to run. For this reason we will only revisit it after we've settled on a good pre-processing pipeline.

Second Attempt

For our second attempt, we focused on data processing and feature selection with the goal of reducing the noise in the data as much as possible. Here we only made use of logistic regression with 14000 training examples and 14000 testing examples. What follows are the data processing techniques that we used.

Spell Correction

The first thing we did was to apply spell checking on the reviews and replace mispelled words with suggested corrections. The idea here is that user reviews on the internet are often filled with misspelled words. One positive review may contain the word "good" while another may contain "gooodi". We want to treat both signals as representing positive reviews.

Stemming

After spell correction, we further consolidated words to their canonical forms by applying stemming. For example, one positive review may contain the word "perfect" while another may contain "perfection". Again, we want to treat both signals as representing positive reviews. Stemming converts both words to their root: "perfect".

Stopword Removal

In this step we removed stop words that are believed to add little relevence to reviews. Words like "the", "I", "was", etc. are stop words and thus removed. One caveat is that we did not remove the words "no" and "not" which will be explained in the bigram generation section that follows.

Bi-gram Generation

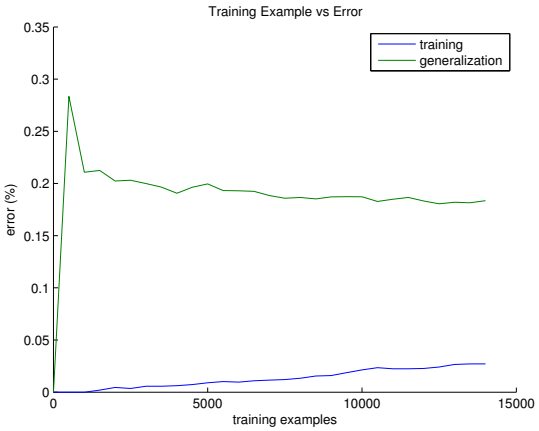
After spell correction, stemming, and stopword removal, we moved on to bi-gram generation. This step is very important because it allows us to capture semantics that are not captured in the uni-grams or worst have opposite meaning altogether. For example "really good" is a much stronger signal for positive reviews than "pretty good" or just "good". Similarly, "not good", if captured inidividually contains the word "good" which falsely indicates a positive review. We can fix that by adding the bigram "not-good" but now we have a problem where the review contains contradictory signals. To fix that, we remove the word "good" and keep only "not-good" which is a clear indication of a negative review.

Results

Here is a table that shows the training and generalization accuracies. The *Individual* section shows experiment results where we use each data processing technique individually. The *Incremental* section shows experiment results where we incrementally combine data processing techniques. Finally, the *Optimal* section shows experiment results where we combine only data processing techniques that we think will give us optimal results.

	Train	Test
Individual		
Baseline	97.3931	81.6759
Spell Correction	97.1862	82.3517
Stemming	95.5655	82.1586
Stopword Rem	97.1724	82.6759
Bi-grams	99.9241	84.4621
Incremental		
Baseline	97.3931	81.6759
+ Spell Correction	97.1862	82.3517
+ Stemming	95.5034	82.0000
+ Stopword Rem	95.3241	81.5586
+ Bi-grams	99.7379	83.2966
Optimal		
Spell + Bi-grams	99.9586	84.5103

Here is a graph that shows the number of training examples vs training/generalizat error for the case of using only spell correction and bi-grams.



Discussion

Individually, every data processing technique improved the generalization accuracy but all of them, with the exception of bi-grams, also reduced training accuracy. When combined incrementally, however, it appears that stemming and stopword removal actually reduces generalization accuracy.

From the individual and incremental experiment results, we gathered that the most promising data processing techniques are spell correction and bi-grams. Thus we combined only those two and indeed we achieved training and generalization accuracies that were superior to the best.

Looking at the training error vs generalization error, it is as expected that by adding more training examples, the training error (bias) goes up where as the generalization error (variance) goes down. However the trend suggests that as we add more training examples, training error will continue to go up where as the generalization error will likely flat off. Judging from this we will have to either explore other pre-processing techniques or other learning algorithms.

Future Work

For the rest of the semeter, we will focus on the following work, all of which have a common goal of further improving the generalization accuracy.

- We will take our processed data with spell correction and bigrams and retry some of the more expensive learning algorithms. In particular I believe that by tuning the nu-SVM algorithm we should be able to get substantially better results than what we currently have.
- After having settled on an algorithm, which I believe will be nu-SVM, we will play around with k-fold cross-verification to see whether we can further improve the results.
- Try out other data processing techniques???
- Try adding metadata features???