Homework 1: Sentiment Analysis

Joshua Wallace
Department of Astrophysical Sciences
Princeton University
joshuajw@princeton.edu

Abstract

1 Introduction

Classification is important in all fields of science. Some examples specific to astrophysics: classifying celestial objects based simply on the emission of light from the object, and often do so in an approximate way without having detailed observations of defining characteristics; classifying whether a transient dip in a star's brightness is due to a planet or another star crossing between us and the star; and classifying galaxies based on images. Less grandiose but more useful, in this work I try to understand useful classifiers and parameters for sentiment classification of a given set of reviews. Information from sentiment analysis of reviews can give companies valueable insights as to which of their products are most successful, as well as allow them to identify bad products and situations and focus resources towards mitigating fallout from the situation producing the negative reviews.

In this work, I investigate various classifiers and associated tools (e.g., feature selection) to figure out the best and most accurate classifiers. Specifically, I look at X classifiers in this way and that way. Bag-of-words. Maybe speed. Feature selection.

2 Related Work

Some of the earliest work of sentiment analysis of online data was done by Turney [7], Pang, Lee, and Vaithyanathan [3], among others. Pang, Lee, and Vaithyanathan highlight earlier "knowledge-based" sentiment classifiers that incorporated pre-selected sets of words or linguistics-based models rather than, in their words, the "completely prior-knowledge-free supervised machine learning methods" they employ. They used Naïve Bayes (NB), Maximum Entropy, and Support Vector Machines (SVMs) classifiers to great effect, getting accuracies between 77%–83%, depending on how they defined their feature set. Turney, although strongly machine learning-oriented, did not have a "completely prior-knowledge-free" method, but built a classifier based on the mutual information between phrases and the words "excellent" and "poor". He achieved an average accuracy of 74%.

Sentiment analysis has exploded in the decade and a half since these initial works. In fact, sentiment analysis is itself a business. An informal listing of such businesses can be found on, e.g., the Quora post "Which companies are doing real time sentiment analysis on social media?" [5]. A Google search for "why does sentiment analysis matter for my business" finds many articles from many business-related sites about sentiment analysis and why it is useful for a business. Much progress has been made in the field, and the top-performing fully automated sentiment analyses of 2017 are far more sophisticated than those of 2002. As just one (outdated) example, a 2008 paper from several Google associates [2] describes pulling aspects of a particular location or service from a comment and then rating individual aspects in an automated way. As both sentiment analyses get more sophisticated and a greater volume and variety of reviews and information get poured into the Internet, sentiment analysis is sure to become increasingly used in the future.

3 Methods

3.1 The Data and Preprocessing

The data consisted of 3,000 online reviews, of which 2,400 were taken as a training set and 600 were taken as a test set. These reviews were given without explanation of source, but a personal perusal of the reviews revealed that they cover a variety of reviewable things, from cell phones to movies to restaurants. The data were preprocessed using a code kindly provided by the professors. This code tokenized the reviews as well as removed stop words, with the help of the `nltk` Python library. It then determined a vocabulary list based on words that had a frequency across the entire training set above a certain threshold. I tried different values for for this threshold, between 3 and 6 inclusive, to determine a best value to use. The code then converted each review into a bag-of-words representation based on the vocabulary list. The length of the vocabularly list for the threshold values of 3, 4, 5, and 6 are respectively 812, 640, 541, and 462. These are the initial lengths of the feature vectors; feature selection will shorten these.

3.2 The Real Meat of this Work: Classifying

This work makes use of classifiers from the `nltk` [1] and `sklearn` [4] Python packages. I use the following classifiers:

- NB from `nltk`, which employs a Bernoulli estimator
- NB from `sklearn`, employing three different estimators (Gaussian, Bernoulli, and multinomial)
- Random Forest (RF) with 150 trees and an entropy-based splitter (I was particularly interested in this one since a recent astrophysics study [6] used a RF classifier to classify 450,000 of a certain type of binary star and I need to do something very similar for my PhD research.)

This work put some emphasis on feature selection. Several feature selection algorithms were used:

- A simple "by-hand" one, where I tried to cull the features myself
- Variance threshold
- Select K-best, using three different scoring functions (chi2, f_classif, mutual_info_classif)
- Recursive feature elimination, using a Support Vector Classifiers estimator

Each combination of classifier and feature selection was evaluated using on the 600 reviews withheld for testing, as discussed above and based on quantities such as accuracy (number of correct classifications over total number of reviews), precision, recall, and F_1 score.

3.3 Naïve Bayes in Detail

Naïve Bayes may be a simple classifier, but to me it is special because it was the first classifier I formally learned. Before that, machine learning was nothing but magic to me, but after I learned Naïve Bayes I saw how formal, understandable, and simple machine learning was. Naïve Bayes is named such because it makes use of Bayes' theorem and also makes the "naïve" assumption that all the features are independent. For the specific application of sentiment analysis, counter-examples of the independence assumption are easily thought of: the existence of the word "great" in a review can have a very different classification interpretation depending on whether the word "not" occurs before it. Despite this, Naïve Bayes still enjoys prominence as a classifier because of its simplicity.

The following description of the Naïve Bayes classifier greatly benefitted from review of [8]. Wikipedia may not be the most reliable source, but the math speaks for itself. For a feature vector representation of some input $\vec{x}$ and a set of classifications $C_{1..k}$, the classifier's job is to calculate

$p(C_k|\vec{x})$ and from this to decide which C_k is to be assigned as the classification for the input data. For the Naïve Bayes classifier, Bayes' theorem is invoked,

$$p(C_k|\vec{x}) = \frac{p(\vec{x}|C_k)p(C_k)}{p(\vec{x})}. \tag{1}$$

Since the denominator $p(\vec{x})$ is independent of C_k, it does not matter in this analysis and is thus ignored.

Using the chain rule,

$$p(C_k, \vec{x}) = p(C_k, x_1, ..., x_n) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k, x_1, ..., x_{i-1}). \tag{2}$$

Note that $i = 1$ in Equation (2) as it is written would give an x_0 in the product, a value that doesn't exist. This should be interpreted as meaning no values of x go into the conditional probability inside the product for $i = 1$. Here is where the assumption of mutual independence among features comes in. To be precise, all that is needed to be assumed is *conditional independence* among the x_i given C_k. If this is the case, then we can write $p(x_i|C_k, x_1, ..., x_{i-1}) = p(x_i|C_k)$. In this case, Equation (2) becomes

$$p(C_k, \vec{x}) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k) = p(C_k)p(x_1, ..., x_n|C_k). \tag{3}$$

This can be plugged in to Equation (1) to obtain

$$p(C_k|\vec{x}) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k), \tag{4}$$

where I have ignored the evidence term. Thus, if one can establish $p(C_k)$ and all the $p(x_i|C_k)$, then it is possible to calculate probabilities for each C_k for any given $\vec{x}$. The typical approach is then to select the C_k with the largest probability as the classification for the input data.

The question now becomes how to evaluate the various $p(x_i|C_k)$ from a given set of training data. There are different models that can be used to fit the probabilities given the training data, but here I will focus on the Bernoulli distribution, since it lends itself well to how the data are expressed in this work (vectors of zeroes and ones, zero meaning the given feature is not present in the data and one meaning the given feature is present in the data) as well as the binary nature of the classification. The Bernoulli distribution is defined entirely by the probability p that outcome will be 1 (positive review); the probability that the outcome will be 0 (negative review) is $1 - p$. Thus, to fit the Bernoulli distribution to data all we need is to calculate p. A simple way to do this is, for a particular feature (word) and a particular set of training data (reviews), to simply calculate $p(\text{word}|C_k)$, i.e., the fraction of reviews of a given sentiment that contain the word. However, simply doing this makes your trained model particularly vulnerable to insufficient training data. For instance, if in your training data you never see a certain word occur in a negative review, then from Equation (4) the probability that any review that contains this word is negative will be 0. It will not always be accurate to have the probability of a given classification shot down to 0 based on a single word, so there should be some sort of "correction" to prevent this in the Naïve Bayes. One such correction is to use "add-one smoothing". This correction takes the fraction described above used to calculate $p(\text{word}|C_k)$ and adds 1 to the numerator and a number K to the denominator equal to the total number of classes (for our analysis, $K = 2$). This ensures that there will never be $p = 0$ for any classification given a feature.

4 Results

4.1 Naive Bayes

I first used the Naive Bayes classifier from the `nltk` package [1]. I tried different minimum number of words to be counted as vocab. For 3 I got training: 86.4% accurate and test 78.8% accurate, for 4 I got training: 85% accurate and test 78.3% accurate, for 5 I got training: 84.1% accurate and test: 77% accurate, and for 6 82.8% accurate for training and 77.3% accurate for training.

I also tried to do a little bit of "feature selection" on my own. I identified some words I thought would have no impact on the sentiment of a review such as brand names (e.g., "samsung" and "bluetooth") and food items (e.g., "potato" and "taco"). I then ran the `nltk` Naive Bayes analysis on these. I chose these words because it seemed to me that these words were not positive or negative in themselves but rather were focused on the product being reviewed itself. This had a marginal effect on the percentages: some went up 0.1–0.2%, some went down 0.1–0.2%, some remain unchanged. This version of feature selection did not seem to offer any improvement (especially since it made some accuracies higher and some lower), and our feature set isn't so large as making it smaller by a few words to speed things up, so I decided to not use this "feature selection".

Implementing the `sklearn` [9] Gaussian Naive Bayes had worse results overall: for 3, 81.8% training and 71.7% test; for 4, 79.3% training and 71.8% test; for 5, 77.9% training and 70.5% test; for 6, training 76.5% and test 69.7% accurate.

Bernoulli and Multinomial do much better. Almost up to 80%!

Number of words to be included as vocab is: 3

GaussNB Training percentage for 3 : 0.818333333333 Gauss NB Test percentage for 3 : 0.716666666667

BernNB Training percentage for 3 : 0.863333333333 BernNB Test percentage for 3 : 0.793333333333

MultNB Training percentage for 3 : 0.862916666667 MultNB Test percentage for 3 : 0.791666666667 Number of words to be included as vocab is: 4

GaussNB Training percentage for 4 : 0.792916666667 Gauss NB Test percentage for 4 : 0.718333333333

BernNB Training percentage for 4 : 0.849583333333 BernNB Test percentage for 4 : 0.79

MultNB Training percentage for 4 : 0.852083333333 MultNB Test percentage for 4 : 0.786666666667 Number of words to be included as vocab is: 5

GaussNB Training percentage for 5 : 0.779166666667 Gauss NB Test percentage for 5 : 0.705

BernNB Training percentage for 5 : 0.840416666667 BernNB Test percentage for 5 : 0.773333333333

MultNB Training percentage for 5 : 0.8425 MultNB Test percentage for 5 : 0.77 Number of words to be included as vocab is: 6

GaussNB Training percentage for 6 : 0.765416666667 Gauss NB Test percentage for 6 : 0.696666666667

BernNB Training percentage for 6 : 0.828333333333 BernNB Test percentage for 6 : 0.773333333333

MultNB Training percentage for 6 : 0.828333333333 MultNB Test percentage for 6 : 0.766666666667

VarianceThreshold is a bust

Slightly better SelectKBest. Around k=310, 80% accuracy. However, fairly insensitive to value of k, and indeed there are several "local maxima", very small maxima though relative to the average. Accuracy starts to fall off for k¡=100 or so. This is chi2.

Using funderscoreclassif, also get 80% accuracy around k=310, but accuracy doesn't fall off so much. Also several "local maxima".

mutualunderscoreinfounderscoreclassif takes a much longer time, much longer! And lower accuracy.

RFEVC: optimal number of parameters (with 30-step resolution): 232, got 78% accuracy. I think. I don't know if the 78% corresponded to the 232 or something else.

5 Discussion and Conclusion

Acknowledgments

This work greatly benefitted from not only the guidance of the professors and instructors but particularly benefitted from the preprocessing code `preprocessSentences.py` provided by the professors.

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. ”O’Reilly Media, Inc.”, 2009.

[2] Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, volume 14, pages 339–348, 2008.

[3] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Quora. Which companies are doing real time sentiment analysis on social media? `https://www.quora.com/Which-companies-are-doing-real-time-sentiment-analysis-on-social-media`. Accessed: 24-Feb-2017.

[6] I Soszyński, M Pawlak, P Pietrukowicz, A Udalski, MK Szymański, Ł Wyrzykowski, K Ulaczyk, R Poleski, S Kozlowski, DM Skowron, et al. The ogle collection of variable stars. over 450 000 eclipsing and ellipsoidal binary systems toward the galactic bulge. *arXiv preprint arXiv:1701.03105*, 2017.

[7] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[8] Wikipedia. Naive bayes classifier — wikipedia, the free encyclopedia, 2017. Accessed 24-Feb-2017.

[9] Jun Zhu, Amr Ahmed, and Eric P Xing. Medlda: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1257–1264. ACM, 2009.