Predicting ratings and usefulness of restaurant reviews using classification and regression methods

November 22, 2017

Abstract

In this paper we implement four machine learning methods for two datasets, provided in Aalto University's course "CS-E3210 Machine Learning: Basic Principles". Two of the implemented methods (Multinomial Naive Bayes and Neural Network) solve a classification problem, where the task is to predict whether a Yelp review was deemed helpful by other users. The other two implemented methods (Least Squares Regression and Ridge Regression) solve a regression problem, where the task is to predict the amount of votes for a Yelp review. Both of the datasets have a bag-of-words as input parameters.

For each task, we compare the used methods and their results, as well as a simple benchmark, produced by a trivial baseline method. We found that the methods we implemented are more accurate than the baseline methods.

1 Introduction

In document classification, a bag-of-words model is used to define each document simply as frequencies of its words. This means that for each document we form a vector $\mathbf{X}$ that consists of N elements which each represent the amount of times a particular word appears in the document. We can conduct supervised learning on these vectors $\mathbf{X}$ if we obtain a training set that contains some output value(s) for each document. The output could be anything, for example the type of the document or the main topic of a news article. The goal is then to try and predict the output values of unknown documents after learning some model parameters on a training set.

In this paper, we approach two different problems concerning a dataset of 5000 Yelp restaurant reviews. In the classification problem, the aim is to predict whether a restaurant review is positive (4-5 stars) or negative (1-3 stars). In the regression problem, the aim is to predict the amount of "useful"-votes a review has received. For classification, our algorithms of choice are Multinomial Naive Bayes and Neural Network and for the regression task we implement Least Squares Regression and Ridge Regression. We train our algorithms on the training set of 5000 reviews, where true output labels are provided. After this we evaluate and compare our methods on an additional test set of 1000 reviews. The aim of this study is to determine which of the methods in question produce the most accurate results and see if there is any advantage in using the more complex methods (Neural Network and Ridge Regression) over the simpler ones.

In a broad sense the research of document classification methods is important, because we are facing an ever-growing amount of literature/documents especially on the Internet. It is useful to have a computer algorithm to sort the documents instead of handling them manually, especially when the amount of documents is large.

2 Methods

2.1 Classification

2.1.1 Multinomial Naïve Bayes

Naïve Bayes is a group of relatively simple methods for classification that rely on the use of Bayes rule. Prior probability estimates p_{ri} of observing each class i are often based on their frequencies in the training data. The maximum likelihood estimates for class-conditional feature probabilities are then calculated by counting the appearance or absence of a feature in each data vector. In a document-classification setting a bag-of-words model can be used, in which each feature is simply a word and a data vector contains values denoting the frequency of each word in a single document. In a Bernoulli Naive Bayes model word frequencies are either just 0 or 1 depending on if the word is present in a document or not.

Multinomial Naïve Bayes is an expansion to Bernoulli Naïve Bayes in that data vectors now consist of actual word frequencies in a document rather than just binary values. [MN98] found Multinomial Naive Bayes to almost always improve prediction accuracy in text classification when compared to Naive Bayes model (denoted as multi-variate Bernoulli model in their article). We chose Multinomial Naive Bayes for our task, because it is relatively easy to implement, but should still produce sensible results.

2.1.2 Neural Network

A neural network is a computational approach for finding patterns in complex (big) data. Neural networks consist of multiple artificial neurons, loosely mimicking the way human brains work. Due to their ability to learn patterns from a data set, they are ideal for solving our classification problem. [HN09]

There are multiple ways to structure neural networks, but the one we implemented is a feed-forward neural network using backpropagation. Even though this is one of the simplest neural networks that can be constructed, it is still reported to be rather effective for tasks that are similar to the one at hand.

A basic neural network consists of $2+N$ layers: one input layer, one output layer, and N *hidden layers*, which are positioned in between the input and output layers. All layers consist of one or more artificial neurons, or *nodes*, which process input data using an *activation function*.

A feedforward neural network is simply a neural network where processed input data is always output to the next layer of nodes. In other words, there are no cycles in a feedforward neural network.

Backpropagation is a method for teaching a neural network the desired output. It works by feeding the gradient of a loss function back to the network, which uses it to adjust the activation functions' weights to minimize the error. By doing this automatically in the training phase, the neural network optimally learns possible hidden patterns in the provided dataset.

2.2 Regression

2.2.1 Linear Least Squares

Linear Least Squares regression is a simple method that aims to predict an output y from input variable(s) x. The equation implied by the model is written as $y = a + bx + \epsilon$, where a is the intercept or "bias" term and b is a weight parameter for x. LS-regression can also have multiple input variables and it is then called Multivariate Linear Regression. In a case of multiple inputs the model equation can be written as $y = b_0 + b_1x_1 + b_2x_2...b_kx_k + \epsilon$ or in vectorized format $\mathbf{Y} = \mathbf{BX} + \epsilon$. In our case of a bag-of-words model, each variable x_k represents the frequency of a single word. The least squares solution for weight parameters is then found as $\mathbf{B} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$. After solving the values for $\mathbf{B}$ on the training data, we can predict output labels for test data by calculating $\mathbf{Y} = \mathbf{BX}$

2.2.2 Ridge Regression

Ridge Regression, also known as Tikhonov regularization, is a method for solving regression models, which produce a vector **r** of result estimates when given an input matrix **X**. It is commonly used for non-orthogonal, or "ill-conditioned", problems, which are characterized by the eigenvalues of **X'X** being much smaller than 1. The Ridge Regression consists of a loss function, which is the Linear Least Squares function, and a regularization term that is given by the L_2-norm (i.e., $\Gamma = \alpha I$, where Γ is the regularization term, $\alpha \in [0.0, 1.0]$, and I is an identity matrix). [MS75]

3 Experiments

3.1 Classification

3.1.1 Multinomial Naive Bayes

As a starting point, we implemented the MNB algorithm from scratch, using maximum likelihood estimates as stated in equations (5) and (6) of [MN98]. These equations also contain a Laplacean prior to take into account cases where a document might have a frequency of 0 for some words. This would produce a probability estimate of 0 without the prior, which is not desirable.

Figure 1 shows the obtained weights and figure 2 shows the absolute values of the same weights in an ascending order. From figure 2 we can see that after the six largest weights, there is a slight gap after which the values start to descend almost linearly. As one experiment, we ran the same algorithm selecting only the six most important features, which were ”disappointed”, ”never”, ”delicious”, ”definitely”, ”pretty” and ”bad”. Most of these, with the exception being ”pretty”, seem relatively strong words, so it makes sense that the algorithm places a large weight on them. ”Pretty” is slightly controversial, because it could be used for example in describing the outlook of a restaurant or in phrases like ”pretty good” and ”pretty bad”. The resulting error rates after and before feature selection can be seen in table 1.

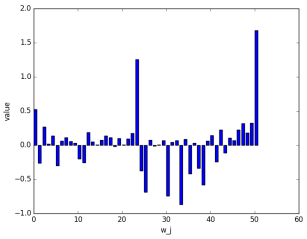

Figure 1: Values of weights ordered by index

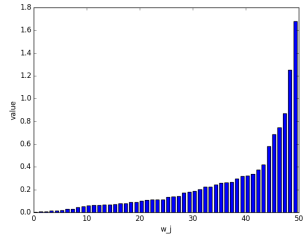

Figure 2: Absolute values of weights sorted in an ascending order

3.1.2 Neural Network

The neural network was implemented from scratch using Python with libraries NumPy and Pandas. Our implementation is a feedforward neural network that uses gradient descent as the backpropagation function. It consists of one hidden layer and it outputs a float in the range of [0, 1]. The float represents the deemed probability of the related review's classification: output of 1 means that the probability for 1 is 100%, output of 0.25 means that the probability for 0 is 75%, and so on.

There are many parameters that can be adjusted when using the neural network. For example, activation functions can be varied to gain different results. Some popular activation functions are tanh, rectifier, softmax, and sigmoid due to the relative ease of acquiring their derivatives by using the output values. We decided to use the tanh function in the hidden layer, and the softmax function in the output layer. Due to the laboriousness of changing the activation functions on the fly, we did not try to vary the aforementioned activation function setting while experimenting with the neural network.

Other parameters that can be adjusted when using the neural network include the amount of hidden layers in the network, the amount of nodes in each hidden layer, and the learning rate for the backpropagation. We decided to keep the amount of hidden layers at one due to the cumbersomeness of adding new hidden layers into the code, but we did try to adjust the amount of nodes in the hidden layer. After hours of (unfortunately undocumented) trial and error, we found that the network yields best results when the amount of nodes is kept in relatively low figures, e.g., 5. The learning rate was found to be optimal when kept at about 0.0001; larger figures seemed to cause oscillation in the weight adjustment, whereas smaller figures seemed to fixate on local minimas.

Finally, the number of iterations in the training phase can be adjusted. This is a critical parameter, as too many iterations could result in overfitting the model, whereas too few iterations result in underfitting the model. Most sources we consulted suggested that finding the correct parameters for a neural network is an art rather than exact science. Thus, we decided to visualize how the number of iterations impact the error rating in order to find an appropriate number. Figure 3 illustrates how the visualization shows that the error is at minimum after around 200 training iterations.

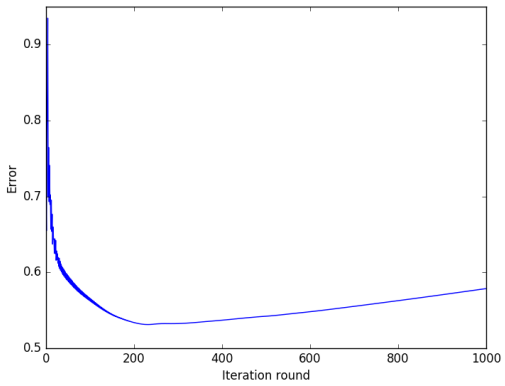

Figure 3: Prediction error as the function of iterations in the training phase

3.2 Regression

3.2.1 Linear Least Squares

Linear Least Squares was implemented from scratch using [Ale10] as a reference. The method placed overwhelmingly large weights on seven features and ignored the rest. These features were in a descending value of weight ”service”, ”menu”, ”price”, ”restaurant”, ”food”, ”staff” and ”taste”. All of these make sense, because they are words that describe the different aspects of a restaurant experience, which is important when voting whether a review is useful or not. Important words in classification such as ”delicious” and ”disappointed” may not tell anything about the quality of a review itself, while they tell a lot about whether the restaurant is good or bad. Again, similarly to the feature selection carried out in Chapter 3.1.1, we ran the LS-algorithm using only the seven largest weights.

One experiment to test whether there were any problems in the program code was to drop out two important features: ”service” and ”menu”. This produced weights as shown in figure 5. There is now more weight on most features but we can still discern five peaks higher than the rest, and the bias weight is now much larger. These changes at least confirm that the algorithm doesn't always produce near-zero weights on most features, which was suspicious at first.

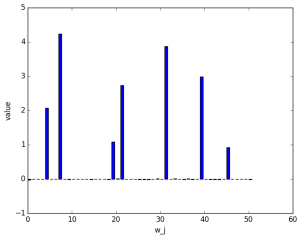

Figure 4: Values of weights ordered by index

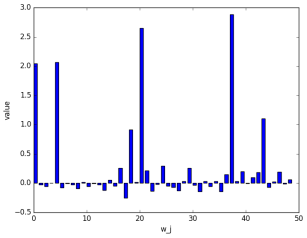

Figure 5: Values of weights ordered by index, ”service” and ”menu” features dropped

3.2.2 Ridge Regression

Our implementation of Ridge Regression uses the built-in Ridge function of the Python library scikit-learn. [rid]

The usage of the scikit-learn library is rather straightforward; the regression model is created, trained, and used with a total of three commands. There are some parameters that can be varied, but we tried to vary only *alpha*, which didn't seem to have effect on the output predictions at all.

4 Results

4.1 Classification

To validate the methods' effectivenesses, we compared their prediction errors[1] to the prediction error that was produced by a simple baseline method, where all predictions are 0.5 (denoting that it would be equally possible that each review would be classified as 0 or 1). Whereas the baseline method produces an error of approximately 0.69, the MNB method predicts the correct classification with an error of approximately 0.66, which is an amiable improvement over the baseline method. The neural network predicts the correct classification with an error of approximately 0.60, which is an even better improvement.

Therefore, we can say that the tested predictive methods actually work, and especially the neural network is a very effective tool for predicting whether the comment was seen as helpful or not. Table 1 illustrates how the error rates of the baseline method, the Multinomial Naïve Bayes method, and the neural network compare to each other.

Method	E_{test}
NN	0.60
MNB	0.66
MNB (6 features)	0.68
Baseline	0.69

Table 1: LogLoss error in classification

4.2 Regression

To validate the methods' effectiveness, we compared their prediction errors[2] to the prediction error that was produced by a simple baseline method, where all predictions are the mode of the training dataset votes. The baseline method produces an error of 28.387, which means it does not yield even modestly accurate predictions. The studied methods perform drastically better; Ridge Regression is able to predict the votes with a very good accuracy, producing only an error of 0.101, whereas Least Squares surpasses even that with an error rate of 0.048.

Therefore, we can say that the tested predictive methods actually work, and especially the Least Squares method is a very effective tool for predicting whether the comment was seen as helpful or not. Table 2 illustrates how the error rates of the baseline method, the Least Squares method, and the Ridge Regression method compare to each other.

Method	E_{test}
LS (7 features)	0.047
LS	0.048
RR	0.101
Baseline	28.387

Table 2: MSE in regression

[1] LogLoss error, as computed by Kaggle https://inclass.kaggle.com/c/mlbp2016-classification.
[2] Mean Square Error, as computed by Kaggle https://inclass.kaggle.com/c/mlbp2016-regression

5 Discussion

The results of classification, as shown in table 1, are mostly what we expected. The results in regression are a bit more surprising as the Ridge Regression yields more inaccurate results than the slightly simpler Linear Regression. The difference between these methdods is that the Ridge Regression takes into account a generalization term in addition to the Linear Least Squares regression. In general, Ridge Regression should have an advantage over Linear Regression when there is multicollinearity present between the features [NCS]. In our case, changing the value of the regularization term did little to improve the accuracy, so it might be the case that there isn't much multicollinearity between the variables. We could have investigated this further by studying the multicollinearity between the features, but unfortunately we ran out of time at that point.

It would have been interesting to try a more advanced regression method and see if it would have provided even more accurate results than the simpler methods we implemented. Nevertheless, the prediction accuracy we obtained with LS and RR methods was already remarkably high.

In both tasks, there was a single method that produced superior results by a significant margin, the methods being Neural Network and Linear Regression. The study question was thus answered. In further studies, it would be a good idea to test the efficiency of these methods with other datasets that don't necessarily have to be text documents.

References

[Ale10] E. C. Alexopoulos. Introduction to multivariate regression analysis. *Hippokratia*, 14:23–28, 2010.

[HN09] Simon Haykin and Neural Network. A comprehensive foundation. *Neural Networks*, 3(2009), 2009.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. 1998.

[MS75] Donald W Marquardt and Ronald D Snee. Ridge regression in practice. *The American Statistician*, 29(1):3–20, 1975.

[NCS] NCSS: Ridge regression. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/mlc/2002/cotraining-11-02.pdf. Accessed: 2016-11-25.

[rid] scikit-learn: Ridge regression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge. Accessed: 2016-11-25.