A Comparison: Classifying News with Naive Bayes and Softmax Regression

Text classification is among the dominant aspects in Digital Humanities applying computing techniques to analyze and categorize large volumes of data, thanks to the proliferation of digital archives ranging from publications, social media, and many other up-to-date digital formats that can convey information. Topic modeling, sentiment analysis, stylometry, and genre classification are all among the common applications, while news classification is a very concrete application that can be used to categorize the political ideology in current affairs reports, analyze the theme of historical newspaper and articles, as well as detect the trends on social media.

The algorithms behind these applications are diverse, including naive Bayes, softmax regression, support vector machines, and k-nearest neighbors along with many others. Among them, the naive Bayes probabilistic classifier is often used in text classification applications and experiments because of its simplicity and effectiveness which basic idea is to use the joint probabilities of words and categories to estimate the probabilities of categories given a document. They are simple probabilistic classifiers based on Bayes' theorem with strong independence assumed between features given a class, while “the independence assumptions on which naive Bayes classifiers are based almost never hold for natural data sets”. (Lewis 11). Softmax regression as a deep learning algorithm and neural network models the relationship between features and the probabilities of belonging to each label using the softmax function. Since deep learning algorithms allegedly have better performance for text classification tasks such as sentiment analysis (Sunarya et al. 77), this paper aims to compare the accuracy of naive Bayes and softmax regression from this perspective.

Introduce the Dataset

For this classification task, the training dataset is a BBC news archive consisting of 1490 pieces of news labeled in 5 classes, namely business, tech, politics, sports, and entertainment. The test data is another BBC news archive consisting of 2225 pieces of news labeled in the same way.

Multinomial Naïve Bayes Classifier

The core logic of Naive Bayes is that the probability of event A and event B happening together can be obtained by multiplying the respective probabilities of each happening. In mathematical terms, the Bayes' Theorem is expressed as:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Where P(A|B) is the probability of A happening given the feature/condition B, and likewise P(B|A) is the probability of B happening given the feature/condition A. While P(A) and P(B) are the independent probabilities of A and B. “A” can be seen as the class that we aim to predict, and “B” is the sample, or features that can be extracted from the sample, therefore P(A|B) can be translated as the probability of B belonging to the class “A” when we have a sample” B” with certain observed features.

In the “Multinomial_Naive bayes_Classifier”, the training dataset is used to calculate two kinds of probabilities in general. Firstly, the prior probabilities of a given piece of news belonging to each class. For example, if there are 100 pieces in total, and 20 of them are political news, then P (political news) = 0.2. Secondly and more importantly, the probability of a piece of news with a certain feature word belonging to each class. For example, if there are altogether 100,000 words in the class labeled as “political news”, and the feature word “war” showed up 200 times in this

particular class, then it can be expressed as P (political | war) = 0.02. In the script, these probabilities are built into a 2-dimension matrix with the rows representing each class, and the columns representing each feature word, consisting of feature word counts in each class:

```
word_list = word_dict[cls]
    for word in prob_count.keys():
        if word in word_list:
            prob_count[word] += word_list.count(word)
    for word in prob_count.keys():
        prob_matrix.loc[word, cls] = prob_count[word] / (news_nums * features_nums + features_nums)
```

They are calculated by the "get_probability(training_set, feature_words)" function fed with the crucial matrix of word counts and classes.

Given these probabilities as training results, the prediction result of a given piece of news is based on the comparison among probabilities of belonging to each class, and each of these probabilities is calculated by firstly multiplying the prior probability with the feature word probability, and the feature word counts in the given news, and then summarizing theses probabilities for each feature word present in the given news. This is realized via the function " predict_with_content". It is obvious that each feature here is assumed as being independent from each other, because the probabilities of an event are calculated via the multiplication of probabilities of assumedly independent other events.

Softmax Regression Classifier

The similarity between Softmax and Naive Bayes is that they both require a 2D dimension of feature word presences (as columns) in each piece of news (as rows) to keep track of the words (or more important words in terms of frequencies, when there are limits set in the script to exclude the

more obscure ones) that show up in particular classes. However, Softmax works differently in that it does not count word frequencies, it only records their occurrences and positions in the matrix, and the matrix would be like this: [[0,0,0,0,1,0,0,1,…], [0,1,0,1,0,0,0,0,…] …], where 1 indicates the presence of a feature word in one class, and 0 indicates none:

```
X_train = [[1 if word in element[0] else 0 for word in feature_words] for element in training_set]
…
X_test = [[1 if word in element[0] else 0 for word in feature_words] for element in test_data]
```

This training matrix is built inside the function "train_test_extract(training_set, test_data, feature_words)" of the "Soft_Max_Classifier" script.

Then the algorithm feeds the softmax function with the training matrix of size m*n where m is the number of samples and n is the number of features (feature words), and applies the gradient descent algorithm to optimize the predictions. The Softmax regression involves learning a set of parameters (weights and biases) to map the input features to the output classes, and gradient descent could update the parameters by iterations to minimize the discrepancies between the outputs (predicted results) and actual classes. The appropriate set of parameters can accurately capture the pattern of word occurrences in a specific news class through multi-dimensional computing with a training matrix representing the positions and layouts of specific words in certain classes. This is realized through the function "softmax(self. X)" and the training function "fit_BGD(self, X, y, alpha=0.01, reg=0.1, max_iter=1000, epsilon=1e-10)" and "softmax((self, X))", Where X can be the training matrix of features*number of pieces news, and y is the training matrix of class*number of pieces of news.

Results and reflections

For the same dataset, the accuracy of Naive Bayes is around 87%, while Softmax regression is around 97%. One of the reasons for the latter to predict more accurately may lie in that Naïve Bayes assumes that the features/conditions are independent of each other, and the Softmax does not hold this assumption. When the features are relative to each other, which is true in the case of news where there are prevalent topic patterns and collocates of words, Naïve Bayes functions may not yield the most accurate prediction, while Sodtmax can always find the most optimized set of parameters to map the training data into a predictive pattern through iterations and adjustions.

However, the result perhaps can also be attributed to the size of the dataset. In a comparative study of logistic regression and naive Bayes, the experiments find that naive Bayes perform better with smaller datasets, and sometimes even beat the logic regression algorithm (Ng and Jordan). In this case, the BBC news training file may be large enough for the naive Bayes to be left behind by softmax regression. To test this finding, the BBC training file is reduced to only 200 pieces of news, and the accuracy rate of the naive Bayes classifier does rise from the previous 87% to 89%, while the softmax regression also increases from 97% to 99%. This result somehow validates the contention that naive Bayes performs better with smaller sample size.

Conclusion

Through the comparison of these two algorithms for news classification, the finding is that softmax regression demonstrates higher accuracy compared to naive Bayes. While naive Bayes may perform relatively better with smaller datasets, softmax regression is steadier and more effective

in most cases. The choice between the two algorithms may depend on factors such as dataset size and complexity of relationships between features and classes.

Notes:

The training data are from https://www.kaggle.com/c/learn-ai-bbc/data, and the test data are from https://www.kaggle.com/competitions/learn-ai-bbc/data?select=BBC+News+Test.csv .

References

Lewis, D. D. “Naive(Bayes)at Forty: The Independence Assumption in Information Retrieval”. *Springer Verlag*, 1998, p.11.

Ng, A. Y., and M. I. Jordan. “On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes.” *Advances in Neural Information Processing Systems*, Jan. 2002. https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

Sunary, P.O. Abas et al., “Comparison of Accuracy between Convolutional Neural Networks and Naïve Bayes Classifiers in Sentiment Analysis on Twitter.” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, 2019.