An Application of Bernoulli Naïve Bayes Classifier for High-dimensional Dataset Prediction

CS412 Final Project Report

Xin Wen(xinwen5), Qile Zhi(qilezhi2), Xintong Wu(xwu68)

May 3, 2017

1 Introduction

This algorithm aims to use existing data as training data to train Naive Bayes Classifier and predict unknown data. The training data includes train.txt, user.txt, movie.txt and test.txt. User.txt contains users' attributes such as age, gender, occupation. Movie.txt contains movies' attributes including year and genres (one movie can have multiple genres). All attributes may have N/A type, which means that data is unknown or missing. Train.txt is training data for the classifier. It contains two IDs of users and movies for identification and the rating from a specific user to a certain movie. Namely, user_id, movie_id, and rating. Test.txt is the data from predicting and it only consists of user_id and movie_id. This algorithm will first integrate the all the data to a DataFrame by matching attributes from user.txt and movie.txt to the train.txt and test.txt. Then preprocess the data by applying the One-Hot-Encoding method to convert categorical data into binary data. And finally, we use the processed data to train the Bernoulli Naive Bayes classifier, build training model and predict the rating for the test data.

2 Project Description

The core ideas of our new model are One-Hot-Encoding and Bernoulli Naive Bayes Classifier. The techniques we used this time are totally different from what we have tried in the midterm checkpoint.

2.1 Data Preprocessing

We using the pandas.DataFrame as the main data structure to store the data. Firstly, we concatenate the train and test datasets for efficiency and merge the data with user and movie files to create an integrated data set that contains all the attributes from different data set. Then, in order to binarize the data for constructing the model, we use one-hot-encoding to process Gender, Occupation, Genre, Age, and Year. Thus, We get a large table with binary values. In this case, we did not fill in the missing data since by testing, filling the missing data using mean value, median or mode of the attribute does not improve the accuracy.

For example, this is the sample data before one-hot-encoding process.

Table 1: Pre One-Hot-Encoding data

Gender	**Age**	**Occupation**
M	25	17
-1	25	17
F	18	4
M	-1	7
-1	-1	2

We convert the Gender column into three columns: Gender_M, Gender_F, and Gender_-1. For the column Gender_M, fill the boxes with 1 when the original entry is M; fill the others with 0 otherwise. Same to column Gender_F and Gender_-1.

Similarly, processing the Age and Occupation attributes through the same procedure can get the following table:

Table 2: Post One-Hot-Encoding data

Gender_M	**Gender_F**	**Gender_-1**	**Age_25**	**...**	**Occupation_4**	**Occupation_2**
1	0	0	1	...	0	0
0	0	1	1	...	0	0
0	1	0	0	...	1	0
1	0	0	0	...	0	0
0	0	1	0	...	0	1

Here we mainly use functions in the existing package pandas, like merge, concat, drop, and get_dummies to process the data sets.

2.2 Probabilistic Model

"Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the naive assumption of independence between every pair of features."(1) Given a class variable y and a dependent feature vector x_1 through x_n, Bayes theorem states the following relationship:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

Since $P(x_1, ..., x_n)$ is constant given the input, we can use the following classification rule:

$$P(y|x_1, ..., x_n) \propto P(y)\Pi_{i=1}^{n} P(x_i|y)$$

In this case, the Bernoulli Naive Bayes model can be expressed as:

$$P(x|y) = \Pi_{i=1}^{n} P_{k_i}^{x_i}(1 - P_{ki})^{1-x_i}$$

Where P_{ki} is the probability of class y generating the term w_i and x_i is a boolean expressing the occurrence or absence of the i'th term from the data provided(2).

2.3 Model Construction and Related Algorithm

We implement our own Bernoulli Naive Bayes model based on the description in sklearn website. We implement four functions fit, predict_log_proba, predict and _binarize_X.

We divide the whole data back into two sets, training data and testing data. We use train data and rating to construct the model.

This is a brief algorithm description of our model:

Algorithm 1 Bernoulli Naive Bayes Model Implementation

```
Data: Trainings set
Result: Prediction
TrainBernoulliNB()
 $X \leftarrow$ training data
 $y \leftarrow$ rating
 Fit($X, y$)

Fit($X, y$)
 Binarize $X$
 Group the training data by class
 Get the number of training samples
 Calculate the prior log probability for each class
 Count each word for each class and add smoothing
 Calculate the log probability of each word
 Get the number of documents in each class plus smoothing
```

2.4 Prediction

For every entry, we apply the trained model to predict the probability when rating equals to 1, 2, 3, 4, and 5, respectively. We select the rating with the highest probability as our prediction.

Last, we combine the Id with the predicted rating, and form a CSV file.

2.5 Work Distribution

We write the outline of the main code and implement One-Hot-Encoding for some attributes together. We separately tried a couple of models, including Naive Bayes, Decision Tree and

Random Forest. After comparison of the results via Kaggle, we decide to use Bernoulli Naive Bayes, and One-Hot-Encode the remaining attributes, except Ids.

In the implementation of Bernoulli Naive Bayes:
Xintong wrote the model-building part (fit function)
Qile wrote the prediction part (predict function)
Xin wrote the implementation of execution file (RatingPredictor)

We finished writing Final Report together.

3 Final Result

The output includes a list of transaction ID and predicted the rating for a user to a movie. The result finally achieved 23 out of 42 in Kaggle leadership board (Top 55%). The accuracy may increase if more data preprocessing method is applied to the data set.

Here is part of our sample output.

```
802553,5
802554,4
802555,5
802556,5
802557,5
802558,5
802559,5
802560,5
802561,5
802562,5
802563,5
802564,4
802565,4
802566,4
```

References

[1] Scikit Learn: Bernoulli Naive Bayes
http://scikit-learn.org/stable/modules/naive_bayes.html

[2] Naive Bayes classifier
https://en.wikipedia.org/wiki/Naive_Bayes_classifier