CSE255-A Assignment 2 Report

Gao, Yansong
PID: A53100210
yag037@ucsd.edu

Jiang, Tong
PID:A53098335
toj002@ucsd.edu

Zhao, Ruiwen
PID: A53091644
ruz073@ucsd.edu

ABSTRACT

In this assignment, we use Amazon movie reviews data to predict the rating of a given movie. Here we mainly use two models to build the predictor – linear regression and latent factor. When choosing the features of the linear regression, we use natural language processing techniques to analyze the reviews text to get whether the reviews is positive or negative. When applying the latent factor model, we try to figure out the rating baseline over all the movies and the rating offset for every specific user and movie.

Keywords

data mining, natural language processing, latent factor

1. EXPLANATORY ANALYSIS

In assignment 2, we use Amazon movie reviews from http://snap.stanford.edu/data/web-Movies.html.

This dataset consists 7,911,684 movie reviews from Amazon. The reviews span a period of 15 years, from Aug 1997 to October 2012. The total number of movies covered by this dataset is 16,341. The total number of users covered by this dataset is 889,176, and 16,341 of these users had writen more than 50 reviews.

The dataset is organized in the following format. "product/productId" indicates the Id of a movie on Amazon. "review/userId" indicates the Id of a user on Amazon. "review/profileName" indicates the user name of the writer of the review. "review/helpfulness" indicates how many users have rated this review and how many users considered this review was helpful to them. "review/score: 5.0" indicates the rating given by the user to this movie, which is also what we want to predict. "review/time" shows when the review was written in unix time. "review/summary" and "review/text" gives us the summary and content of the review.

2. PREDICTIVE TASK

Our task is to predict rating based on user, item, and review text. We will use MSE to indicate the performance of our model. The baseline we compared to our model is to simply predict a global average rating on every movie.

Considered that this task is rating prediction task, which means that we need to give a concrete number of the rating at last, linear regression is a perfect model that fits the requirement. On the other hand, we can also take this task as a recommendation task, in which the rating indicates to what extent a user will like a movie. So we can also use a latent factor model to build this recommendation system.

For the linear regression model, we will build our feature based on the context of the review, which has been given directly by the dataset. For the latent factor model, we don't need any concrete feature but only the user ID, the product Id and the rating given by the user to the product, which have also been given directly by the dataset.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

3. MODEL DESCRIPTION

In this assignment, we use two different models to do the prediction: linear regression and latent factor. In linear regression, we use natural language processing techniques to analysis the sentiment of each review text.

3.1 Sentiment Analysis

The basic idea of sentiment analysis is to break the sentence into words, and see if each word is in the sentiment dictionary. Our dictionary is combined with the dictionary downloaded from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html, and the dictionary we created from the training data.

3.1.1 Build Dictionary

Since the dictionary downloaded from web contains only general positive and negative words, but no "movie-ish" words, we need to find out these movie-ish words on our own.

To build our own dictionary, we chose 252977 positive reviews and 46637 negative reviews (Each review with rating higher or equal to 4.0 are regarded as positive reviews, and those with rating lower or equal to 2.0 are regarded as negative reviews). We break each review into a bag of words, and label them "positive" or "negative" according to the sentiment polarity of the review they belong to. Then we use nltk.NaiveBayesClassifier to train these features.

By using Naive Bayes Classifier, we can calculate if a word is more likely to be a positive word or a negative one by calculating the following ratio:

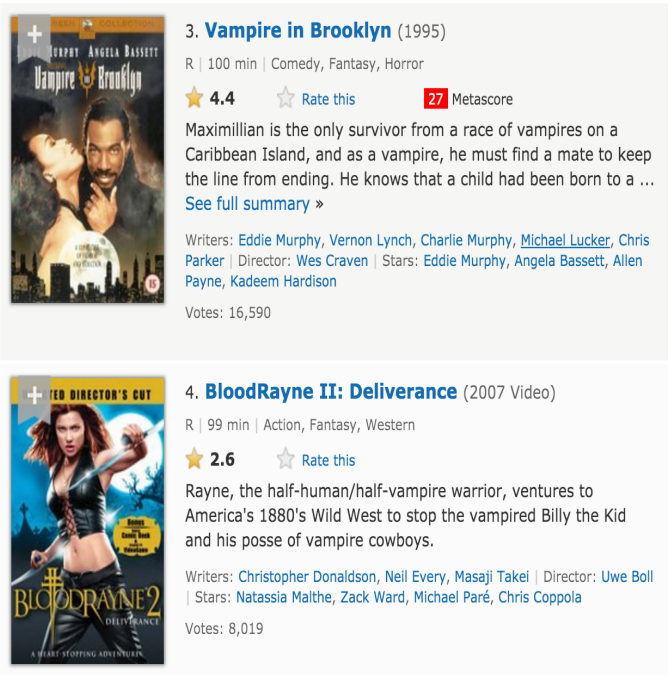

Figure 1: Two movies with keywords "Dhampir"

Table 1: Sentiment words generated

refundbr	neg : pos	66.9 : 1.0
bolls	neg : pos	58.5 : 1.0
dhampir	neg : pos	52.4 : 1.0
crappiest	neg : pos	48.8 : 1.0
chucked	neg : pos	41.6 : 1.0
poorlywritten	neg : pos	41.6 : 1.0
kagan	neg : pos	40.1 : 1.0
whatsover	neg : pos	38.0 : 1.0
scientologists	neg : pos	38.0 : 1.0
bloodraynebr	neg : pos	38.0 : 1.0
stunkbr	neg : pos	38.0 : 1.0
craptastic	neg : pos	38.0 : 1.0
stinker	neg : pos	37.4 : 1.0

$$\frac{P(w \text{ is positive}|w)}{P(w \text{ is negative}|w)} = \frac{P(w \in r|\text{pos review } r)P(\text{pos})}{P(w \in r|\text{neg review } r)P(\text{neg})} \quad (1)$$

And then we pick out the words with this ratio higher than 5.0 or lower than 0.2 and add them to the corresponding sentiment dictionary. Some of the sentiment words we chose through Naive Bayes Classifier are as showed in Table 1. As we can find in Table 1, many words are just movie names or characters' names. For example, Dhampir is a creature half vampire half human, and this word has high neg/pos ratio, so we assume that movies with Dhampir characters are tend to be lame movies.

3.1.2 *Sentiment Analysis*

After building dictionary, we can easily know whether a word is positive or negative one. We calculate the number of positive words and negative words in a review, and add these two number as two features in our linear regression model.

3.2 Latent Factor Model

In the latent factor model, we considered the problem of rating prediction as a product recommendation task, in which the rating represents to what extent a user is likely to like a movie. Rating of 5 means that chances are very high that the user will love this movie. Rating of 1 means that the user probably doesn't like this movie at all.

We can build the latent factor model from the most basic form.

$$f(u,i) = \alpha \quad (2)$$

In this basic latent factor model, we simply predict every with the same rating. And the α will be given by the solution of the following equation.

$$\alpha = argmin_\alpha \sum_{u,i} (\alpha - R_{u,i})^2 \quad (3)$$

It is obvious that the solution of this equation is the global average of all the rating of the movies.

$$\alpha = \frac{\sum_{u,i} R_{u,i}}{N} \quad (4)$$

We will set this very basic model as our baseline, and we will try to improve the model to get a better MSE than the baseline.

To improve this basic model, we use a little bit more complex latent factor model as below.

$$f(u,i) = \alpha + \beta_u + \beta_i \quad (5)$$

In this model, we still include the global average as a baseline for every rating prediction. However, based on this baseline, we come up with two offsets for every pair of user and item(movie). The first offset is β_u, which describe that for a particular user, how much is the tendency that he or she will give a rating compared to the baseline. For example, if the global average is 4 for all the movies. And a user, named Bob,is a reviewer with a very high taste, and he is likely to give a lower rating than the average rating. So the offset for Bob β_{Bob} will be a negative number, like -0.5. Similarly, the definition of $beta_i$ is that how much is the tendency that a given movie will be rating compared to the baseline. For example, movies of the Harry Potter series are all very popular around the world, and their rating on the Amazon are all higher than the global average, so the β_{HP} will be a positive number, like 0.3.

To figure out all the α and β, we need to find the solution of the following equation.

$$argmin_{\alpha,\beta} \sum_{u,i} (\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda[\sum_u \beta_u^2 + \sum_i \beta_i^2] \quad (6)$$

Here we add a new parameter λ, which is to regularize all the other parameters. The solution of this model is not as intuitive as the first one. This model is jointly convex in β_i, β_u. It can be solved by iteratively removing the mean and solving for β.

$$\alpha = \frac{\sum_{u,i} R_{u,i} - \beta_u - \beta_i}{N} \quad (7)$$

Table 2: Movies with high β_i

Movie	β_i	Star
Gargoyles: Season 1	0.84	4.9
Seinfeld: Season 7	0.80	4.8
Community: Season 1	0.79	4.8
The Shield - The Complete First Season	0.79	4.5
The Elephant Man	0.78	4.7
The Sound of Music	0.77	4.8
Psycho	0.76	4.7

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - \alpha - \beta_i}{\lambda + |I_u|} \tag{8}$$

$$\beta_i = \frac{\sum_{i \in U_i} R_{u,i} - \alpha - \beta_u}{\lambda + |U_i|} \tag{9}$$

We trained the parameter of α and β on 500,000 reviews. And get the parameter that $\alpha = 4.01162494547$, which describes that the global average of rating is about 4 stars. The β for users varies from -2.04 to 1.06. The β for movies varies from -2.10 to 0.84.

To get a more straight impression, let's see some movies with a high β and their ratings. From table2, we could see that the movie with the highest β_i is called "Gargoyles: Season 1", whose rating is 4.9. From Amazon, we could see that it was released on December 7, 2004. 402 reviewers have given their reviews. 369 of these reviewers rated this TV serious 5 star, and all the others rated it 4 star. Thus, it reasonable that when a new user buy the DVD of this TV serious, he or she will give a very high rating to it. That's also why we get a very high β of this movie.

4. RELATED LITERATURE

4.1 NLP model

We use natural language process techniques to analyze the sentiment of the review text.

Similarly, Zhuang's work in 2006 [5] does research on movie review mining. They use movie-related feature words to analysis movie review. Inspired by their work, we decide to dig into the review text, which we didn't do for assignment 1, and trying to figure out the sentiment behind the review text, in order to help us predict the rating.

Pang et al's work in 2008 [4] provides a good survey for us to explore the field of sentiment analysis. In order to get a more comprehensive idea of the algorithm, we explore the work of Hu in 2004 [1], and the work of Liu in 2005 [2]. Their work propose the idea of using sentiment word dictionary to analysis sentiment, and they provide a positive dictionary and a negative dictionary.

When implementing our own algorithm to analyze the sentiment of the review text, we adopted the rough idea in the paper, and we also used the sentiment dictionary mentioned in the paper.

4.2 Latent Factor model

We use the most simple but effective form of the latent factor model, which is mainly refer to what we have leaned from this course. We find that Professor McAuley [3] also did some research on this topic. His research is included in the paper below:

From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews

The dataset we used in this assignment is also created by Professor McAuley in his research work. According to his paper, the MSE of rating prediction of a standard latent factor model and some other non-stardard latent factor models is shown in table3.

We will compare the result of our model to these state-of-art models in the next section.

Table 3: MSE of the state-of-art latent factor models

Model	MSE
standard latent-factor	1.099
community at uniform rate	1.082
user at uniform rate	1.088
community at learned rate	1.082
user at learned rate	0.711

Table 4: Tune the weight of the mix model

NLP	LF	MSE
0.9	0.1	1.403
0.8	0.2	1.392
0.7	0.3	1.384
0.6	0.4	1.378
0.5	0.5	1.376
0.4	0.6	1.377
0.3	0.7	1.381
0.2	0.8	1.390
0.1	0.9	1.401

5. RESULT

5.1 Training models

We have used 550,000 reviews to train and test our models. We use the first 50,000 reviews as our test set. We use the following 500,000 reviews to train our NLP model and latent factor model.

We used 4 different models to test on the test dataset, besides the NLP model and the latent factor model, we also use the naive model mentioned in the section 3.2, and a mix model of NLP and latent factor model.

For the latent factor model, we have a parameter λ that need to be tuned according to equation (6). So we sample a validation set to tune the λ. It shows that we will get the best performance when $\lambda = 10$.

For the mixture model, we try to combine the NLP model and the latent factor model together. The basic idea is that these two models are two completely different models. Theoretically, each of these models should have a good performance on some kind of data. Although it is very hard for us to identify that what kind of data fits which model better, we can still find a way to combine these two models to make a better use of their advantages. That is to a weighted average of these two models. We also tuned the weight in a validation set, and the result is given in table 4. It shows that when we make it half and half, we get the best MSE.

5.2 Testing models

Table 5: Results of different models applied

Model	MSE
naive model	1.57
(nlp) NLP model	1.42
(lf) simple latent-factor	1.41
mix of (nlp) and (lf)	1.37

We can see in the Table 5 the result of the above models. When we simply predict the rating by a constant number, the global average rating, the MSE is 1.57. This will be our baseline. If our model can't beat the baseline, it will be trivial. For the model of NLP and the model of latent factor, although they are two completely different kind of models, the result of these two models are very closed, which are 1.42 and 1.41. At last, we can see that the mix of the previous two models can even improve the MSE to 1.37.

5.3 Analysis

The testing result shows that all the three models that we described in this paper have beaten the baseline, but not very much. The NLP model and the latent factor have a very similar result. This should be just a coincidence, because these two totally different models. The NLP model is based on the review context as the feature. On the other hand, the latent factor model used no features, just the rating itself.

The reason why sentiment analysis is not working very well is because we only use a simple model. We only consider the number of sentiment words, regardless their positions in a sentence. This over-simplified model may mis-classify many reviews. For example, the review "This movie is not bad at all." will be regarded as a negative review because it contains the word "bad". However, this review is obvious a positive one. A more complex sentiment analysis model should include syntax include. The reason why we didn't use this is because syntax analysis is too slow. It will take seconds to analysis one review, and intolerable long time to analysis all the reviews.

As we mentioned in the section 5.1, we expect that the mix model can make a better use of the advantage of each model. From the result, we can see that the mix model indeed improves the MSE on the testing set. It seems that our assumption that the NLP model and the latent factor give a good performance on different kinds of data is correct.

However, our models still work much worse than the state-of-art latent factor models.

6. REFERENCES

[1] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[2] B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

[3] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. International World Wide Web Conferences Steering Committee, 2013.

[4] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[5] L. Zhuang, F. Jing, and X.-Y. Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50. ACM, 2006.