UNIVERSITY OF CALIFORNIA, SAN DIEGO

Adaptive KNN Recommender

Chen, Zexi
Luo, Shihao
Sun, Jingzhe

December 4, 2017

1 INTRODUCTION

The main goal of this project is to build a collaborative filter recommender system by implementing adaptive KNN algorithm.

1.1 DATA SET

In order to make sure our model works universally. More than one data set are chosen. First data set is the reviews of businesses from Google Local,which were used in assignment 1. The other data set is the meta information and reviews of Video Games from Amazon.

The attributes in Google local review are:

- businessID: The hashed ID of each unique business (eg.B660702032)
- userID: The hashed ID of each unique user (eg.U342536218)
- categories: A list of categories this business belongs to (eg.[Donut Shop, Dessert Shop, Bakery])
- ratings: The star rating of the user's review (eg.4.0)

The attributes in Amazon video game meta information are:

- asin: The hashed ID of each unique video game (eg.B00005O0I2)
- categories: A list of categories this video game belongs to (eg.['Video Games', 'PC', 'Games'])
- price: The unit price of each unique video game (eg.23.88)
- salesRank: The rank of each unique game in all kinds of Video Games (eg. 22579)

For reviews of video games from Amazon data set, only two attributes will be made use of

- asin: The hashed ID of each unique video game (eg.B00005O0I2)

- reviewerID: The hashed ID of each unique reviewer (eg. A3EFSLEMHNPP6A)

We built an adjacency matrix based on the (user, item) pairs extracted from reviews. If the user reviewed this item, then the corresponding value in this matrix is 1, else is 0.

1.2 MOTIVATION

Based on features extracted from detailed information of Video Games(meta), such as price, platform in categories and salesRank, those feature-based models âĂŞ logistic regression and SVM can be used to predict whether a particular video game is visited by a user or not. A part of the adjacency matrix extracted from Amazon data set can be used as labels to train the model and another part can be used to calculate the accuracy of prediction as test set.

Since some features included in meta may not be that related to the visit labels, performance of those models will not be very good. Thus, the collaborative filter based on the data of reviews of video games from Amazon data set which computing similarities between items can also be applied.

2 PREDICTIVE TASK

This task requires a binary output for each user-business pair, the model we have learning related to it are logistic regression, SVM etc. Thus, there are several ways to solve it.

For convenience, we define I, U to represent all items and users in training set, S to be the data in training set, and

$$I_u = \{i \in I : (u, i) \in S\}$$

$$U_i = \{u \in U : (u, i) \in S\}$$

2.1 EVALUATION APPROACH

To evaluate performances of those models, we choose to compute the accuracy of prediction of each model. First, extract visited (user, item) pairs from reviews of video games from Amazon data set. Then, by choosing those random generated (user, item) pairs which are not in visited pairs, we generate a negative labeled train set whose length is exactly equal to the positive labeled train set extracted directly from reviews. We combine these two train sets as a final train set. Test set will also be generated with the same method.

2.2 BASELINE: MOST POPULAR

The naive visit prediction provided in baseline just returns True if the business in question is popular, using a threshold of the 50th percentile of popularity (totalVisits/2).

This only depends on the number of times a certain business showed up in training data without any bias on different user. This is for sure not a good classifier since different user may have different preference and the training data maybe just a small portion of the whole data.

2.3 BASELINE: LOGISTIC REGRESSION

Logistic regression is a method to convert a real valued expression into a probability and then train a classifier for prediction. The loss that need to be optimized is:

$$l_\theta(y|X) = \sum_i -\log(1+e^{-X_i\cdot\theta}) + \sum_{y_i=0} -X_i\cdot\theta - \lambda\|\theta\|_2^2$$

$X_i \cdot \theta$ should be maximized when label is positive and minimized when label is negative. Thus, by calculating the log-likelihood, computing the gradient and solving the problem with gradient ascent, we could achieve an appropriate model. The derivative is the following:

$$\frac{\partial l}{\partial \theta_k} = \sum_i X_{ik}(1 - \sigma(X_i \cdot \theta)) + \sum_{y_i=0} -X_{ik} - 2\lambda\theta_k$$

Considering the attributes of Amazon video game, features selected for the logistic regression include the platforms, the price and the sales ranking of the video game.

$$visited = \Theta \times [1, VGA, PC, Linux, Mac, Nintendo, PlayStation, SonyPSP, Wii, Xbox, price/10, salesrank]$$

2.4 BASELINE: SVM

Since the logistic regressors didnâĂŹt optimize the number of mistakes, and we want to minimize the number of misclassification, and SVM will fix the issue.

$$\arg\min_\theta \sum_i \delta(y_i(X_i \cdot \theta - \alpha) \leq 0)$$

Compared to logistic regression, SVM is trying to find a perfect classifier and the soft-margin formulation is used to figure out the classifier that maximizes the distance to the nearest points. In practical, features selected for the SVM is exactly same as that in logistic regression.

2.5 BPR

Instead of using the feature-based models, it is an alternate way to use a collaborative filter. The section below will talking about a model called Bayesian Personalized Ranking (BPR).

3 ONE CLASS RECOMMENDER

Instead of predict directly if a user will visit a business, one class recommender tried to rank the businesses in training set for each user by maximizing a positive loss:

$$max(\ln\sigma(x_{ui} - x_{uj}) - \lambda_\Theta ||\Theta||^2)$$

where $(u, i, j) \in D_s$ is a random triple that $i \in I_u and j \notin I_u$, λ_Θ is the regularizer for parameters.

In order to maximized the positive loss, we need to use stochastic gradient descent and apply many triples (might be overfitting). The stochastic gradient descent formula is:

$$\Theta \leftarrow \Theta + \alpha\left(\frac{e^{-x_{uij}}}{1 + e^{-x_{uij}}} \cdot \frac{\partial}{\partial\Theta} x_{uij} + \lambda_\Theta \Theta\right)$$

With the update rule above the bpr procedure can be represented by following pseudocode.

Algorithm 1 BPR(D_s, Θ)

```
initialize $\Theta$
while not converge do
    draw $(u, i, j)$ from $D_s$
    $\Theta \leftarrow \Theta + \alpha\left(\frac{e^{-x_{uij}}}{1+e^{-x_{uij}}} \cdot \frac{\partial}{\partial\Theta} x_{uij} + \lambda_\Theta \Theta\right)$
end while
return $\Theta$
```

There are two major way to represent x_{uij} [1].

- BPR-MF: It is a good idea to represent the adjacency matrix to a product of two low-rank matrices. SVD was a solution to decomposition, while a better way is to draw samples and train these two matrices. There are several libraries implemented with this method, so instead, we worked on the other approach.

- BPR-KNN: It is also a good way to find the correlation matrix over all items, based on which we can cluster them in groups, and that can be use to rank items for a specific user.

3.1 BPR-MF

By using the matrix factorization method, x_{ui} can be represented by product of two matrices $\gamma_u \times \gamma_i$ that $\gamma_u \in R^{|U| \times K}$ and $\gamma_i \in R^{|I| \times K}$, where K is the dimension of latent factors, in which case, the partial derivative:

$$\frac{\partial}{\partial\theta} x_{uij} = \begin{cases} (\gamma_i - \gamma_j) & \theta = \gamma_u \\ \gamma_u & \theta = \gamma_i \\ -\gamma_u & \theta = \gamma_j \\ 0 & else \end{cases}$$

and α is the learning rate that we need to adjust. While ranking, for each user u, $\gamma_u(u) \times \gamma_i$ will return a list of scores with size $|I|$, where each entry in result represent the score for an item. Then a simple sorting can complete the ranking task.

As mentioned above, this predictor will return a list of scores that indicate preference on all businesses for each user. The last step for prediction visit/unvisit task is selecting a threshold among scores to separate them(eg. median will predict 1 for half of items and 0 for the other half).

The threshold we used is based on the training data for each user. Find the minimum of all scores of items the user has visited in training set, and minus a offset:

$$threshold_u = min\left(S_{u,i} \text{ for } i \in I_u\right) - \text{offset}$$

where the offset we chose is 0.5.

This classifier can be a good predictor since it only consider how different a user would treat a business from another. And of course it also depends on other parameters $K, \lambda_u, \lambda_i, \lambda_j \alpha$, threshold, convergence condition etc.

3.2 BPR-KNN

Instead of matrix decomposition, x_{ui} can also be represented by a likelihood that u will visit i (ie. summing the correlation between i with all item u has visited)

$$x_{ui} = \sum_{l \in I_u} c_{il}$$

where $C : I \times I$ is the symmetric item-correlation matrix, or, use K highest correlation in I_u.

$$x_{ui} = \sum_{l \in I_u} \text{K highest } c_{il}$$

Thus, for adaptive-KNN, $\Theta = C$. For convenience, the diagonal of correlation matrix are all 0. A common representation of C can be computed by item-cosine similarity.

$$c_{ij} = \frac{|U_i| \cap |U_j|}{\sqrt{|U_i| \cdot |U_j|}}$$

While a better approach is to train their correlation by the algorithm mentioned above. Again the partial derivative of x_{uij} will be:

$$\frac{\partial}{\partial\theta} x_{uij} = \begin{cases} +1 & \theta \in \{c_{il}, c_{li}\} \wedge l \in I_u \wedge i \neq l \\ -1 & \theta \in \{c_{jl}, c_{lj}\} \wedge l \in I_u \wedge j \neq l \\ 0 & else \end{cases}$$

While ranking, for each user u and item i, the score can be computed by summing up K-highest c_{il} such that $l \in I_u$, then a list of scores with size $|I|$ can be computed. A sorting method can complete the ranking task.

The prediction visit/unvisit task can also be accomplished by selecting a threshold among scores to separate them.

The threshold we used is based on the training data for each user, or use a global percentile cutoff for all users (eg. 30%).
The result of this classifier would vary a lot with parameters $K, \lambda_i, \lambda_j, \alpha$, threshold etc.

3.3 Strengths and Weaknesses

Adaptive KNN of collaborative filtering are very popular recently. Meanwhile, it also has some flaws.

Strengths of Adaptive KNN:

- Accurate prediction for users that have personalized preference.
- Better solution for cold start problem: recommender system can work once a user start viewing.
- Convincing recommendation because history-based prediction.
- Latent factors are easier to be extracted than features.

Weaknesses of Adaptive KNN:

- Slow, very slow, time-consuming.
- Works bad for loose adjacency matrix.
- Space consuming while number of items getting large.
- Malicious review may lead to a wrong direction.

3.4 Issues

There were some issues need to be discussed during building this model:

- There are lots of tricks can help save time (eg. use numpy array instead of list, user MapReduce instead of loop).
- While optimizing, the mini-batch gradient descent[2] was used. It is hard to choose an appropriate batch size and epochs.
- While initializing the correlation matrix, we have no idea what it should like, so the initial value may be very strange from it should be.
- The hardest part is adjusting parameters λ_i, λ_j and K, those are the main factors affecting the final result.
- As mentioned above, this model is very time-consuming while training, for which reason the parameters were adjusted under small epochs. Thus, the real accuracy with large epochs should be higher.

4 Literature

4.1 Similar data sets

Data sets used to make predictions in this assignment are reviews of businesses from Google Local, detailed information of Video Games(meta) and reviews of Video Games from Amazon. There are many similar datasets have been studied such as Netflix Prize, which is an anonymized version of their movie rating dataset; it consists of 100 million ratings, done by 480,000 users. Furthermore, Jester dataset contains 4.1 million continuous ratings (-10.00 to +10.00) of 100 jokes from 73,421 users.

4.2 State-Of-The-Art

For recommender system, there are many popular models. For feature-based approach, we can use:
Naive Bayes, which is simple to compute probability just by counting.

Table 5.1: Accuracy of each model

Model	Accuracy	comment
Most Popular	0.63491	
SVM	0.74286	
Logistic Regression	0.783712	
BPR-MF	0.792077	
BPR-KNN	0.74814	K = 5
BPR-KNN	0.73155	K = 10

SVM: fixes the âĂIJdouble countingâĂİ and try to find a hyperplane that minimizes the number of misclassification.
Logistic Regression: optimizes the classification error rather than the likelihood.
Collaborative filtering: make recommendation based on past user-item interaction, which has good performance for users and items with enough data. It, however, does not naturally handle new users and new items. Collaborative filtering is very popular among big companies. Amazon and Google news are all using it.

5 Conclusion

5.1 Result Comparison

The test set was generated from Amazon video game data set, half of which have positive label(visit) and the other half have negative label(unvisit). The accuracy of each method are shown in Table 5.1.

The accuracy of BPR-MF of assignment 1 on Kaggle is above 0.85, this time the only difference is the epochs. Similarly, with appropriate parameters, the accuracy of BPR-KNN should be much better.

5.2 Parameter discussion

Theoretically the parameters will vary between different data sets for this model, so only parameter of this specific data set will be discussed.

K It is indicating the size of group after clustering. A larger K will result in low accuracy, while a small K won't make any sense. The recommended K for this data set is approximately between [5,20].

λ_i, λ_j This pair of parameters guarantees the model to be converging and prevent it from underdamping or overdamping. The reasonable value are about 0.03 and 0.003 respectively.

LEARNING RATE 0.1 should be a good learning rate(converging and not slowly).

EPOCHS With appropriate parameters above, a larger epochs will result in a better accuracy, but again, it is very time-consuming. Thus, try not to use epochs over 100.

BATCH SIZE Large batch size will take too much time, while small batch size may cause some bias. Hence, 500 is a acceptable batch size.

5.3 Summary

Since the goal of our team is to build a library, the parameters will not be main issue of this project although there are some default parameters.
This item-based adaptive KNN is part of collaborative filtering. This model is very popular recent years(Amazon is using collaborative filtering). It will have high accuracy only with appropriate parameters. Otherwise, it still hold similar results as other feature-based models(eg. SVM). There is still some details need to be optimized, but overall it is successfully implemented.

References

[1] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme *BPR: Bayesian Personalized Ranking from Implicit Feedback*. 2009.

[2] Mu Li, Tong Zhang, Yuqiang Chen, Alexander J. Smola *Efficient Mini-batch Training for Stochastic Optimization* 2014.

Appendix A: First

GitHub Link to view code or view `https://github.com/JingzheSun/Recommender_System_Adaptive_KNN`