Rapport du Projet Libre

Xiao WANG (xiao.wang@polytechnique.edu, xiao.wang@telecom-paristech.fr)

June, 2014

Abstract

The project allows me to first have a general view on recent development in recommendation systems on both industrial and academic fields. It then guides me to get involved in a live research project, especially on rank recommendatons with k-order statistic loss.

The first part of this rapport is a sythesis of papers, which lays a theoretical foundation of following research project. In the second part, the paper Learning to Rank Recommendations with the k-Order Statistic Loss written by Jason Weston et al. presented in every detail. In the third part, I'll explain my research work based on the paper, using parameter comparison and ....

1 Introduction

1.1 State of art in recommendation systems

This part summaries the paper[1] of Gediminas Adomavicius et. al. presenting a survy of recommendation systems and possible entensions.

Recommender systems emerged as an independent research area in the mid-1990s when researchers started focusing on the rating structure. In its most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user.

More formally, the recommendation problem can be formulated as follows: Let U be the set of all users and let D be the set of all possible items that can be recommended. Both U and D can be very large, raning in hundreds of thousands or even millions. Let f be the utility function that measures the uesfulness of item d to user u, i.e., $f : U \times D \to R$, where R is a totally ordered set.

The utility of an item is usually represented by a rating. Once the unknown ratings are estimated, actual recommendations of an item to a uesr are made by selecting the highest rating or the N best ratings among all the estimated ratings for that user.

Recommender systems are usually classifed into the following categories:

- Content-based recommendations: the user will be recommended items similar to the ones the uesr preferred in the past.

- Collaborative recommendations: the user will be recommended items that people with similar tastes and preferences liked in the past.
- Hybrid approaches: these methods combine collaborative and content-based methods.

1.2 Large-Scale machine learning

Many machine learning algorithms works efficiently for small set of data. Problems arise when we try to learn from a large-scale data. There are challenges in using algorithms that balance computing efficiency and accuracy, data storage, accelerate computing etc. Even if storage demande is satisfied, the computing time largely limits the performance of statistical machine learning methods.

In an era when data sizes grow faster than the speed of processors, different methods are proposed.

1.2.1 Stochastic Gradient Descent (SGD)

The computational complexity of learning algorithm becomes the critical limiting factor when one envisions very large datasets.

Algorithms such as stochastic gradient descent show amazing performance[2]. According to the paper of Léon Bottou, compared to gradient descent algorithm(GD), SGD reduces the time per iteration from n to 1, reduce the iterations to accuracy ρ from $log\frac{1}{\rho}$ to $\frac{1}{\rho}$, reduce the time to accuracy ρ from $nlog\frac{1}{\rho}$ to $\frac{1}{\rho}$. As a result, when the limiting factor is the computing time rather than the number of examples, the stochastic learning algorithm performs asymptotically better.

1.2.2 Multi-threading and MapReduce

1.3 Effectiveness of Recommendations

In most of the recommender systems literature, the performance evaluation of recommendation algorithms is usually done in terms of coverage and accuracy metrics.

- Coverage measures the percentage of items for which a recommender system is capable of making predictions
- Accuracy mesaures cans be either statistical or decision-support.

  Statistical accuracy metrics mainly compare the estimated ratings against the actual ratings R in the $User \times Item$ matrix.

  Decision support measure determine how well a recommender system can make predictions of high-relevance items. They include classical IR measures of **precision** (the percentage of truly “high” ratings among those that were predicted to be "high" by the recommender system), **recall** (the percentage of correctly predicted "high" ratings among all the ratings known to be "high"), **F-measure** (a harmonic mean of precision and recall), and **Receiver Operating Characteristic** (ROC) measure demonstrating the trade-off between true positive and false positive rates in recommender systems.

Limitatiions One limitation is that these measures are typically performed on test data that the user choose to rate. However, items that users choose to rate are likely to constitute a skewed sample. in other words, the empirical evaluation results typically only show how accurate the system is on items the user decided to rate, whereas the ability of the system to properly evaluate a random item is not tested.

In the ideal case, controlled experiments should be conducted with users in the recommender systems settings, which are however expensive and time-consuming.

However, high-quality experiments are necessary in order to truly understand the benefits and limitations of the proposed recommendation techniques. This is not part of my project, but keep the limitations in mind will help build better recommender systems in the future.

1.4 Brief introduction of paper k-Order Statistic Loss

In this part, I'd like to profile brievely the paper[3] on k-Order Statistic Loss.

It might be abrupt to dive into technical details of this paper without putting it in a more global view on recommender systems.

1.4.1 Model-based collaborative recommendation

There have been many model-based collaborative recommendation approaches in the literature. including K-means clustering and Gibbs sampling. Most recently a significant amount of research has been done in trying to model the recommendation process using more complex probabilistic model. For example, probabilistic latent semantic analysis and a combination of multinomial mixture and aspect models using generative semantics of Latent Dirichelet Allocation.

The essential idea in this paper is founded on a latent semantic analysis model, assuming that user's utility pattern can be approximated by a combination of multinomial factors.

2 Deep study of the paper

In this part of report, I'd like to explain the paper[3] on k-Order Statistic Loss written by Jason Weston et al.

Given an observation matrix $X_{|Users| \times |Items|}$. X is a sparse matrix of users' notations over a set of items, say, films. There might be millions of users and millions of items, making manual prediction impossible. The object of a recommendation system is to predict notations on unrated items.

2.1 LSI(Latent structure index)

One important and basic assumption is that there are latent patterns over users' choices.

However large the user-set is, we can approximate a choice by a combination of choices from a set. That's why we can use a matrix of lower dimension $V_{m\times|Items|}$ to approximate the original one X.

$V_{k\cdot}$where $k \in 0, 1, ..., m-1$ is a factor of preferences over all the items. $V_{\cdot i}$ is the likeliness on item i over all patterns.

So in a very natural way, for a user u, given an item d, the his likeliness for d can be defined as $f_d(u) = \frac{1}{|D_u|}\sum_{i\in D_u} V_i^T \cdot V_d$, where D_u denotes the items already rated by user u.

2.2 Loss functions

The general definition of loss function is as the following: $\sum_{u=1}^{|Users|} L(f(u), D_u)$.

This definition suppose that all users are independent.

2.2.1 AUC loss function

AUC denotes to Area Under the Curve.

$$L_{AUC}(f(u), D_u) = \sum_{d\in D_u} \sum_{\bar{d}\in D\setminus D_u} max(0, 1 - f_d(u) + f_{\bar{d}}(u)). \tag{1}$$

Here the hinge function $max(0, 1-f_d(u)+f_{\bar{d}}(u))$ is used to measure the correctness of the model: whether rated items and unrated items are clearly separated by a distance at least 1.

We can prove that 1 can be replaced by other constants and it does not influence the performance of the recommandation system, neither the speed of convergence nor the correctness of the system.

2.2.2 WARP loss function

WARP is an improvement of AUC loss function. Since AUC loss function treat each item as equal wherever their ranking, WARP is one of the new algorithms that propose to focus on the top or bottom ranked items.

A rank function is defined as the following :

$$rank_d(u) = \sum_{\bar{d}\in D_u} I(f_d(u) \leq 1 + f_{\bar{d}}(u)) \tag{2}$$

WARP loss function is defined as

$$L_{WARP}(f(u), D_u) = \sum_{d \in D_u} \Phi(rank_d(u)) \tag{3}$$

, where $\Phi(\eta)$ converts the rank of a positive item d to a weight.

Φ should be increasing with *rank* function value. For a positive rated item, in a well-perfomed system, its weight should be small. In other words, worse rank leads to larger numeric rank value thus more loss to the objective loss function.

Then should Φ be concave, linear or convext?

Choosing $\Phi(\eta) = C_1\eta$ pays no attention to items on any part of the list. Because change on the top of the list causes the same loso reduce to change on other part of the list.

Choosing $\Phi(\eta)$ concave, for example $\Phi(\eta) = \sum_{i=1}^{\eta} \frac{1}{i}$, pays attention to top ranked items. Because a change on top of the list leads to more change to final loss function than change on bottom of the list.

Similarly choosing $\Phi(\eta)$ to be convexe, for example $\Phi(\eta) = \sum_{i=1}^{\eta} i^2$ pays attention to bottem ranked items, since a change on bottom of the list has more influence than change on other part of the list.

In the original paper, rank function is defined differently as $rank_d(u) = \sum_{\bar{d} \in D_u} I(f_d(u) \geq 1 + f_{\bar{d}}(u))$**, which to me seems to be an error.** Let me explain it from two aspects, from a semantic point of vue and from the consistence with other parts of the same paper.

a) In the first aspect, in common sense, higher rank means smaller numercial rank value. For example, an item ranked 1st means the best item according to a certain criteria. Ainsi, from another point of vue, top ranked items shoud have small numeric rank values.

b) In the second aspect, definition (1) is more consistent with other content in this paper. In at least two points.

Firstly, it's more logic that a higher rank of a well rated item contributes to less loss than the contrary. Given a well rated item, in a well functioning system, its score should be much higher than unrated items, thus have small value in eq (1) and low loss in eq (2).

Secondly, in Algorithm 2, we see an approximate algorithm to get eq (1).

```
pick a positive item d using Algorithm 1.
Set N = 0
**repeat**
    Pick a random item $\bar{d} \in D \backslash D_u$,
    N = N + 1
**Until** $f_{\bar{d}}(u) > f_d(u) - 1$ **or** $N \geq |D \backslash D_u|$
```

After N tri, we got a $\bar{d}$ which satisfies the condition $f_{\bar{d}}(u) > f_d(u) - 1$. The authors suppose that in the set $D \backslash D_u$, in every N values, there is a $\bar{d}$ value that satisfies the above condition. Thus, there are $\frac{|D \backslash D_u|}{N}$ values that satisfy

$f_{\bar{d}}(u) > f_d(u) - 1$. In other words, $rank_d(u) = \sum_{\bar{d} \in D_u} I(f_{\bar{d}}(u) > f_d(u) - 1) = \frac{|D \backslash D_u|}{N}$. This makes sense.

2.2.3 Precision at k

k-Order Statistic loss means we only care about the position at the top k of the ranked list, and don't care about the position of others. This is an essential concept in this paper.

A new definition of loss function is given as

$$L_{k-OS}(f(u), D_u) = \frac{1}{Z} \sum_{i=1}^{|D_u|} P(\frac{1}{|D_u|}) \Phi(rank_{D_{u_{oi}}}(f(u))) \tag{4}$$

where o is the vector of indices indicating the order of the positive examples in the ranked list:

$$f_{D_{u_{o1}}}(u) > f_{D_{u_{o2}}}(u) > ... > f_{D_{u_{os}}}(u) \tag{5}$$

$Z = \sum_i P(\frac{1}{|D_u|})$ is normalization factor. $P(\frac{j}{100})$ is the weight assigned to the j^{th} percentile of the ordered positive items.

Different choices of P result in different loss functions. For example, $P(j) = C_2$ for all j and any positive constant C_2 results in the original WARP or AUC formulations.

$P(i) > P(j)$ for $i < j$ results in paying more attention to positive items that are on the top of the ranked list.

Conversely, $P(i) < P(j)$ for $i < j$ should focus more on improving the worst ranked positives in the user's rating set.

In the majority of experiments conducted by the authors, $P(i) = 1$ if $j = \frac{k}{N}$ and 0 otherwise. That is simply they always select the positive in the k^{th} position in the list. And in this case, $L_{k-OS}(f(u), D_u) = \sum_{i=1}^{|D_u|} \Phi(rank_{D_{u_{o1}}}(f(u)))$.

2.3 The algorithms

In this section, the most important algorthms introduced in the paper are explained in detail.

2.3.1 K-os algorithm for picking a positive item

The complexity of this algorithm is $O(max(K * O(f_d(u)), KlogK))$. Since K is usually set to a small number such as 5, the most costly part is to compute $f_{d_i}(u)$ for each i.

Algorithm 1 k-OS algorithm for picking a positive item

```
Given a probability distribution $P$ of drawing the $i^{th}$ position in a list of size $K$.
Pick a user u at random from the training set.
Pick $i = 1, ..., K$ positive items from items $d_i \in D_u$.
Compute $f_{d_i}(u)$ for each $i$.
Sort the scores by descending order, let $o(j)$ be the index into $d$ that is in position $j$ in the list.
Pick a position $k \in i, ..., K$ using the distribution $P$.
Perform a learning step using the positive item $d_{o(k)}$.
```

Algorithm 2 k-OS AUC Loss

```
Initialize model parameters (mean 0, std. deviation $\frac{1}{\sqrt{m}}$).
repeat
    Pick a user $u$ and a positive item $d$ using Algorithm 1.
    Pick a random item $\bar{d} \in D \backslash D_u$.
    If $f_{\bar{d}}(u) > f_d(u) - 1$ then:
        Make a gradient step to minimize max(0, $1 + f_{\bar{d}}(u) - f_d(u)$).
        Project weights to enforce constraints. e.g. if $||V_i|| > C$ then set $V_i \leftarrow (CV_i)/||V_i||$.
Until validation error does not improve.
```

2.3.2 K-os AUC Loss

From batch gradient descent algorithm to stochastic gradient algorithm In algorithm 2, we use stochastic gradient descend algorithm(SGD) to train the objective factor matrix V. The processus is quite simple and natural to get. Since the AUC loss function is written as eq (1). In order to get the minimum value of the loss function, we can use batch gradient to update $max(0, 1 - f_d(u) + f_{\bar{d}}(u))$, by looking into each $d \in D_u$ and each $\bar{d} \in D \backslash D_u$. If we denote n the number of items, the complexity of this processus is $O(n^2 * O(f_d(u)))$ which is obvious too costly. In SGD, we choose d and $\bar{d}$ at random, and make a gradient step to minimize $max(0, 1 - f_d(u) + f_{\bar{d}}(u))$.

Project weights It's interesting to see the step to enforce constraints. It's necessarily proposed because V_i might explose after several steps, which might bring some trouble in computing $f_d(u)$. In the experiment phrase, I see that different weight projection causes different influences to loss function.

For example, consider two weight projection algorithm:

- a) if $||V_p|| > C$ then set $V_p \leftarrow (CV_p)/||V_p||$.
- b) if $max(||V_d||) > C$, then set $V \leftarrow (CV)/max(||V_d||)$.

Both a and b solves the explosion problem of V.

Recall that $f_d(u) = \frac{1}{|D_u|}\sum_{i\in D_u} V_i^T V_d$, one possible side effect of weight projection is to influence the ordering of $f_d(u)$.

Using a), if $p \in D_u$, $f_p(u) \leftarrow f_p(u) \times (C/||V_p||)^2$, for $i \in D_u$and $i \neq p$, $f_i(u) \leftarrow$

2.4 Evaluation

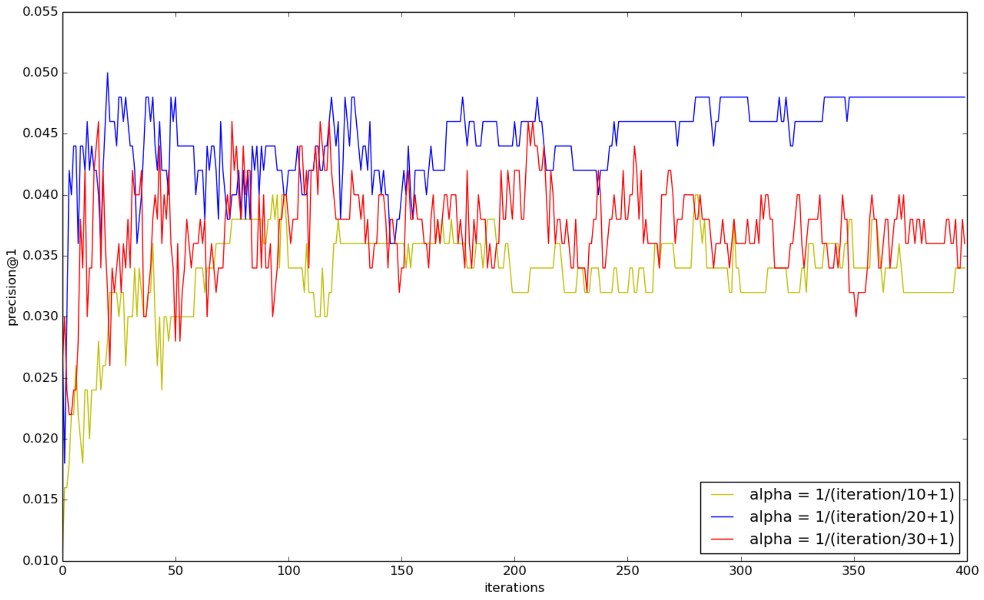
3 Parameter tests

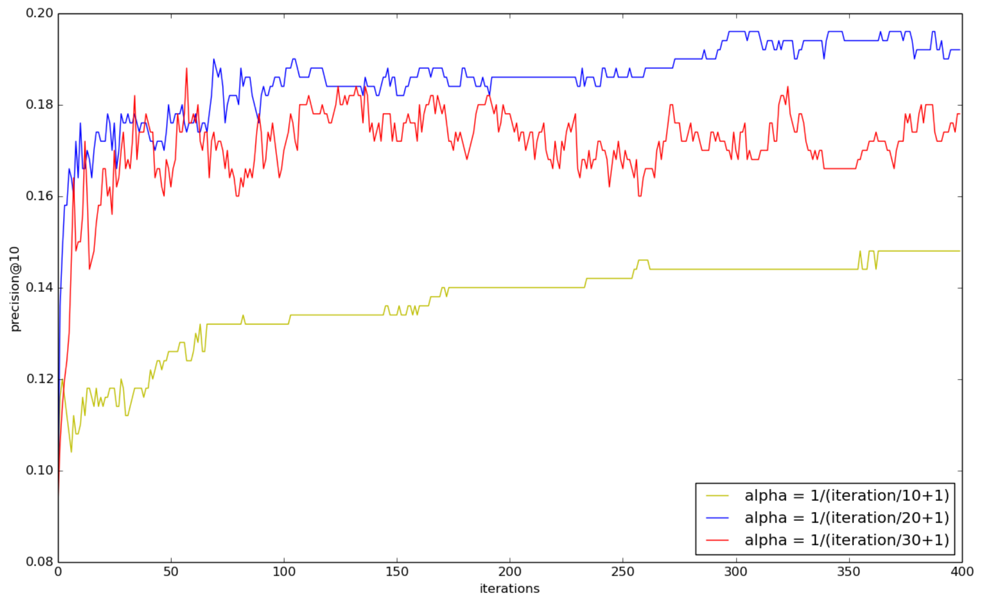
3.1 Learning speed α

α decides the speed of learning. When α is big, the function values (loss or mean-rank) changes quickly at first and can hardly reach the optimal value. When $\alpha$is too small, it performs better on converge period, but takes long time to reach the limit of convergence.

For my test, $\alpha = \frac{1}{(iterationNumber/n+1)}$ performs efficiently and reaches the highest number for precisions.

In the follwing figures we see that $\alpha = 20$ out performs $\alpha = 10$ and $\alpha = 30$.





4 Experimental Results

Datasets I run my algorithm for the following standard datasets for matrix completion problems, using AUC loss function, WARP loss function and k-os loss function.

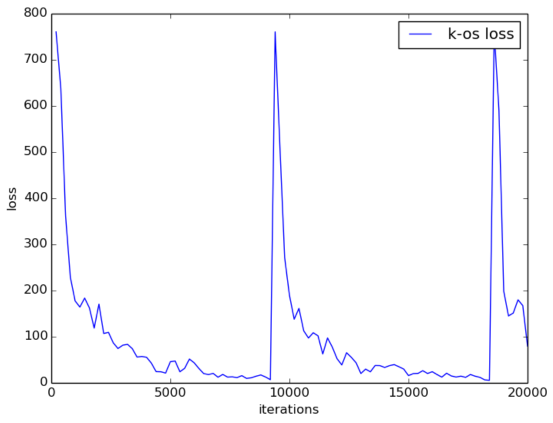
dataset	#ratings	n	m
MovieLens 100k	10^5	943	1682
MovieLen 1M	10^6	6040	3706
MovieLens 10M	10^7	69878	10677
Netflix	10^8	480189	17770

I implemented the algorithms in paper[3]. All results were obtained by implementation in Python2.7, numpy1.8.1 on a 2.4GHz Intel laptop.

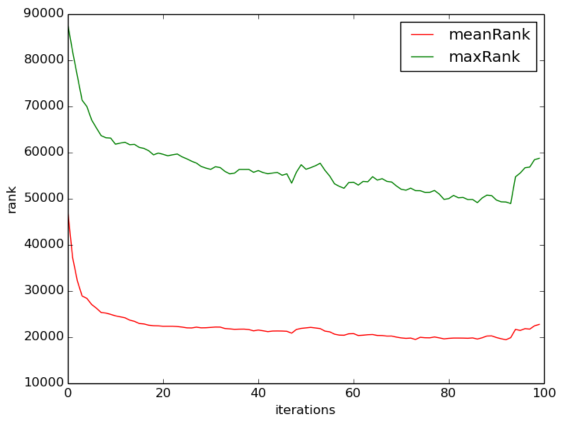
Test methods I randomly select 5 items for testing per user, and keep them apart from training.

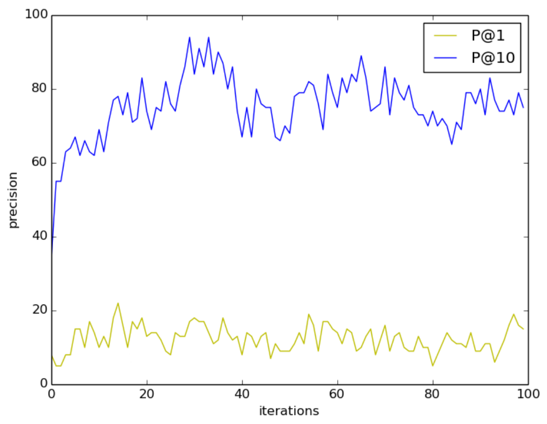
At prediction time for the set of test users, I then ranked all unrated items including the 5 test items, and observe where the 5 test items arein the ranked list of recommendations. Then the following metrics are evaluated: 1) mean rank, the position in the ranked list, averaged over all test items and users, mean maximum rank(the position of the lowest ranked item out of the 5 test items, i.e., the furtherest from the top, averaged over all test users), precision at 1 and 10 (p@1 and p@10), and recall at 1 and 10 (R@1 and R@10).

Sensitivity The generalization performance of the algorithm is relatively stable under diffferent choices of the configuration parameter. See figure 1. The k-os loss converges through iterations.



Mean-rank and maxRank





5 Some challenges

5.1 Scalability

Acknowledgement

I'd like to give the most sincere thanks Dr. Charanpal Dhanjal and Prof. Stephan Clémonçon. Thanks for guiding me into the realm of recommendation systems.

I see the conflicts between our expectations: Charanpal was expecting me as an independent researcher, but I could be stuck at a problem for days and months and was always waiting for help at every crossing. I was expecting a course-style and gradual guidance: from general theoretical introductions and industrial applications to classical algorithms to latest papers, but Charanpal lead me to final phrase from the beginning.

I see the conflicts between my desire for concentration and multiple classes to attend every day. I was not quite available during a period, busy with classes and all kinds of homework.

I see the conflicts between classes arangement and my project. It's better to have basic knowledge on machine learning at the beginning of the project. It'll surely save time.

But most importantly, I see how I grow from these difficulties.

References

[1] Gediminas Adomavicius, Alexander Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-art and possible extensions.

[2] Léon Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent.

[3] Jason Weston, Hector Yee, Ron J.Weiss, Learning to Rank Recommendations with the k-Order Statistic Loss, RecSys '13 Proceedings of the 7th ACM conference on Recommender systems