Empirical Study of Matrix Factorization Methods for Collaborative Filtering

Evgeny Kharitonov

Moscow Institute of Physics and Technology
eugene.kharitonov@gmail.com

Abstract. Matrix factorization methods have proved to be very efficient in collaborative filtering tasks. Regularized empirical risk minimization with squared error loss function and L_2-regularization and optimization performed via stochastic gradient descent (SGD) is one of the most widely used approaches.

The aim of the paper is to experimentally compare some modifications of this approach. Namely, we compare Huber's, smooth ϵ-insensitive and squared error loss functions. Moreover, we investigate a possibility to improve the results by applying a more sophisticated optimization technique — stochastic meta-descent (SMD) instead of SGD.

Keywords: collaborative filtering, matrix factorization, loss functions.

1 Introduction

Recommender systems are widely used in online shops and services such as Amazon, Netflix, Google Video and Google News. Collaborative Filtering (CF) is one of successful approaches to build such systems. During the Netflix Prize Challenge the matrix factorization methods attracted a lot of attention and proved to be very efficient.

A considerable portion of practical problems in the field deals with explicit user feedback: users watch movies and explicitly rate them afterwards. Our task is to recommend some new items to the user, which he or she hasn't seen yet.

The general idea behind factor models implies that it is possible to find user- and item-vectors which characterize the user's tastes and items in such a way that their dot product approximates the user's rating of an item. After that, the items with highest predicted user's rating are recommended to him.

Usually these vectors are found via regularized empirical risk minimization with squared error loss and L_2-regularization. The stochastic gradient descent (SGD) has become the most popular technique for this optimization.

The question we address is the following: *Is there a better choice of a loss function and an optimization algorithm, which can result in better recommendation performance?*

In order to answer the question we compare Huber's [1] and smooth ϵ-insensitive [2] loss functions in comparison with squared error loss.

Stochastic meta-descent (SMD) [3] is a more elaborated algorithm which can substitute SGD in this combination.

S.O. Kuznetsov et al. (Eds.): PReMI 2011, LNCS 6744, pp. 358–363, 2011.

2 Model

Let us denote a rating matrix as X $(n \times r)$, with rows and columns corresponding to the users and items, respectively. Elements of X are real numbers or a special sign $*$ which is used if the value of corresponding rating is unknown.

We associate vectors from $\mathbb{R}^k$ with i-th user and j-th item: u_i and m_j, correspondingly.

Our model is based on the idea that it is possible to find such u_i and m_j that the product $u_i \cdot m_j^T$ represents the i-th user rating of the j-th item:

$$\forall i, j : X_{ij} \neq * \quad X_{ij} \approx u_i \cdot m_j^T$$

It is convenient to formalize the task as a problem of discrepancy minimization between X and its approximation $U \times M^T$ in the cells with known values. In order to do this we introduce a nonnegative loss function L. This function penalizes divergence between predictions and actual values.

More formally, our task is to find U and M, which will minimize divergence for the available data:

$$(U, M) = argmin \frac{1}{T} \sum_{i,j:X_{ij} \neq *} L(X_{ij}, u_i \cdot m_j^T),$$

where T is the number of observed cells in X.

In order to avoid overfitting, a special term is introduced in the target function. Usually it penalizes complex models. We use a L_2 regularization term:

$$(\hat{U}, \hat{M}) = argmin \left(\frac{1}{T} \sum_{i,j:x_{ij} \neq *} L(x_{ij}, u_i \cdot m_j^T) + \lambda_u \sum_i u_i \cdot u_i^T + \lambda_m \sum_j m_j \cdot m_j^T \right)$$

, where $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ is the loss function.

3 Loss Functions

We consider the following loss functions (they are described in [4]) $L(x, \hat{x})$:

squared error

$$L(x, \hat{x}) = \frac{1}{2} (x - \hat{x})^2$$

smooth ϵ-insensitive loss [2]

$$L(x, \hat{x}) = log \left(1 + e^{x - \hat{x} - \epsilon}\right) + log \left(1 + e^{\hat{x} - x - \epsilon}\right)$$

Huber's robust loss [1]

$$L(x, \hat{x}) = \begin{cases} \frac{1}{2} (x - \hat{x})^2, & |x - \hat{x}| \leq \sigma \\ \sigma \left(|x - \hat{x}| - \frac{1}{2}\sigma\right), & otherwise \end{cases}$$

Huber's loss function is not as sensitive to outliers as squared error loss function. In addition to robustness to outliers, ϵ-insensitive loss is insensitive to an additive noise with amplitude up to ϵ.

4 Optimization

Both optimization techniques we describe follow the same principle: given rating X_{ij}, they perform gradient descent in factors u_i and m_k. Instead of minimizing previously discussed target function, they minimize its stochastic approximation:

$$F_{ij} = L(x_{ij}, u_i \cdot m_j^T) + \lambda_u u_i \cdot u_i^T + \lambda_m m_j \cdot m_j^T$$

4.1 SGD

SGD was described by Simon Funk[1] and modified in [5].

Input: X, λ_u, λ_m, optimization steps η_u, η_m, number of iterations K
Output: U, M^T
Initialize U and M^T with small random values;
$i \leftarrow 0$
while $i < K$ **do**
  foreach $i, j \ : \ x_{ij} \neq *$ **do**
    foreach $l \in 1 : k$ **do**
      $u_{il} \leftarrow u_{il} - \eta_u \cdot \frac{\partial F_{ij}}{\partial u_{il}}$
      $m_{jl} \leftarrow m_{jl} - \eta_m \cdot \frac{\partial F_{ij}}{\partial m_{jl}}$
    end
  end
  $i \leftarrow i + 1$
end

Algorithm 1. SGD scheme

4.2 SMD

Stochastic Meta-Descent (SMD) is an online optimization method described in [3]. It can be considered as an improvement of SGD, since it can be reduced to SGD via proper parameter setting. The main difference is that each optimized coordinate has its own step size which is adjusted simultaneously with regular optimization.

The application of this method requires an ability to perform fast multiplication of Hessian of F_{ij} by an arbitrary vector. This can be achieved in our case since the Hessian can be represented as a sum of a diagonal, a block-diagonal and a rank-1 matrixes.

We omit the description of the method and its adaptation to the optimization problem due to lack of space.

[1] Netflix Update: Try This at Home, `http://sifter.org/~simon/journal/20061211.html`

4.3 Meta-optimization

Each combination of loss function and optimization technique has several parameters which are to be determined before model training, e.g. optimization rates η_u, η_m, regularization coefficients λ_u, λ_m, parameters of loss functions such as ϵ and σ. However, there is no commonly adopted approach for their selection in literature. In paper [5] a grid search is used to find optimal parameters. Another approach is to use derivative-free optimization to adjust the parameters on a small subset of available data. Similarly to [6] we use Nelder-Mead method for tuning these parameters.

The initial values of parameters in our experiments are presented in Table 1; the number of iterations used for SGD, SMD and Nelder-Mead was equal to 25.

Table 1. Initial parameters

η_u	η_m	λ_u	λ_m	σ	ϵ	μ (SMD)	ν (SMD)
0.01	0.01	0.02	0.02	6.7	0.03	0.08	0.9

5 Evaluation

The authors of [7] thoroughly discuss the main evaluation criteria used in collaborative filtering. We use three of them: RMSE (Rooted Mean Square Error), MAE (Mean Absolute Error) and Precision-N.

The first two criteria characterize how good we approximate X via product of U and M^T:

$$RMSE = \left(\frac{1}{T} \sum_{i,j:x_{ij} \neq *} \left(x_{ij} - u_i \cdot m_j^T \right)^2 \right)^{\frac{1}{2}} \qquad MAE = \frac{1}{T} \sum_{i,j:x_{ij} \neq *} \left| x_{ij} - u_i \cdot m_j^T \right|$$

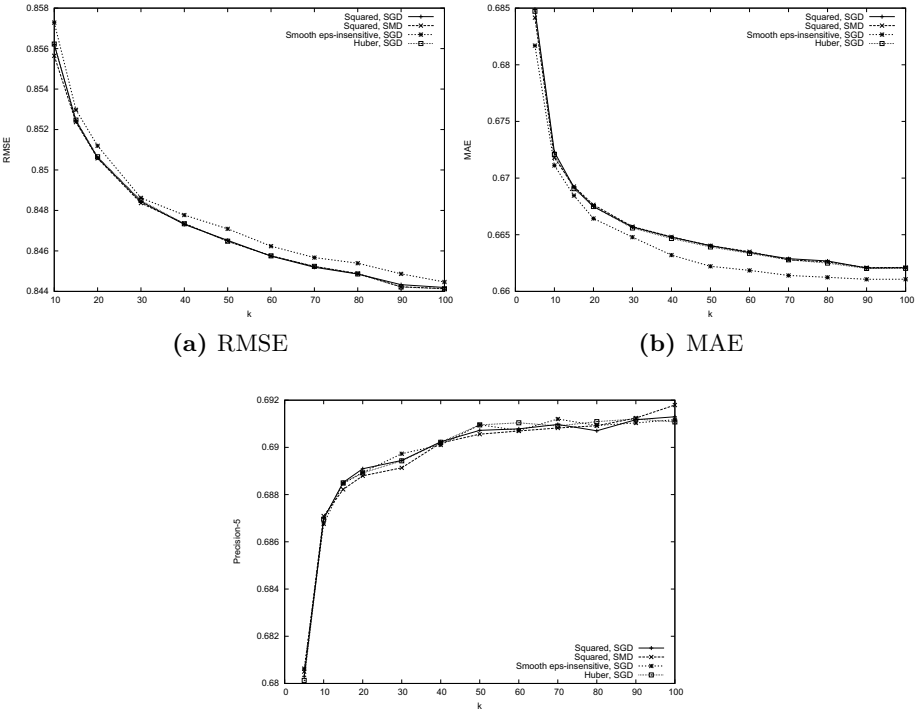
They have several drawbacks [7]. The main problem is that they evaluate the performance of approximation, not recommendations. For this reason, we also use Precision-N criterion.

A trained model can sort the items recommended to the user in the order of descending *predicted* rating.

For a particular user, Precision-N is a ratio of items with real rating above a predefined threshold t among N items with the highest predicted rating.

$$\text{Precision-N} = \text{Average} \left(\frac{1}{N} |\{\text{items with rating not less than } \ t \ \text{ among top-N}\}| \right)$$

In our experiments we used Precision-5 with $t = 4$.



(a) RMSE **(b)** MAE

(c) Precision-5

Fig. 1. Performance vs. k

6 Experiments

In the experiments we used an open dataset provided by MovieLens group . It contains about 1 million ratings of 3900 movies obtained from 6040 users.

The dataset was randomly split into 10 approximately equal parts. For each $k \in [1..10]$ the evaluation process consisted of two steps.

First of all, during the meta-optimization step the parameters that optimize target criterion (RMSE, MAE or Precision-5) were calculated for one part of data while learning on eight other parts were chosen. After that, these nine parts were used as training data and the evaluation was performed on the previously unused part.

This process was repeated until all the parts were used as test sets. The results were averaged and are depicted in Figure 1.

We ensured that for every combination tested we used the same learning, validation and test sets, initial approximations for U and M (for fixed k) and the number of meta-optimization and optimization iterations.

The source code is available on `http://github.com/n0mad/Demoniac`.

7 Discussion

As seen from figures, loss functions that are close to L_2 (Huber's and squared error loss) unsurprisingly result in a better RMSE performance. Similarly, smooth ϵ-insensitive loss (which approximates L_1 loss) achieves a better MAE performance.

More importantly, our experiment has shown that none of the considered modifications did significantly improve Precision-5 in comparison with the standard approach. Thus, taking into consideration simplicity of implementation and lack of additional parameters SGD with L_2 loss and regularization seems to be the best choice.

References

1. Huber, P.J.: Robust statistics. Wiley, New York (1981)
2. Dekel, O., Shalev-Shwartz, S., Singer, Y.: Smooth e-intensive regression by loss symmetrization. In: COLT, pp. 433–447 (2003)
3. Bray, M., Koller-meier, E., Muller, P., Van Gool, L., Schraudolph, N.N.: 3d hand tracking by rapid stochastic gradient descent using a skinning model. In: CVMP, pp. 59–68 (2004)
4. Karatzoglou, A., Smola, A., Weimer, M.: Collaborative filtering on a budget. In: AISTATS, vol. 9, pp. 389–396 (2010)
5. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research 10, 623–656 (2009)
6. Cohen, G., Ruch, P., Hilario, M.: Model selection for support vector classifiers via direct simplex search. In: FLAIRS Conference, pp. 431–435 (2005)
7. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22, 5–53 (2004)