華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Lizhao Liu

Supervisor:
Mingkui Tan

Student ID:
201730683109

Grade:
Undergraduate

November 18, 2019

Recommender System Based on Matrix Factorization

***Abstract*—In this report, we solve the Recommendation System problem based on Matrix Factorization (MF).**
We perform experiments on three aspects:
1. Hidden dimension's impact.
2. Loss function's impact.
3. Regularizer's impact.

I. Introduction

MATRIX Factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices Usually, the user-item interaction matrix is very sparse e.g. a lot of zeros or mising values in it. The adavantage of MF is that it transforms a sparse matrix into two lower dimensionality matrices which are regarded as User Embeddings (UE) and Item Embeddings (IE) respectively. MF is wildly used in academic and industry community, which make it very important for us to fully explore it. We will explore it under different settings.

II. Methods and Theory

In this part, we define Matrix Factorization algorithm.

A. Matrix Factorization

Given a rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, with sparse ratings from m users to n items. MF factorizes $\mathbf{R}$ into P and Q, where $P \in \mathbb{R}^{m \times h}$, $Q \in \mathbb{R}^{n \times h}$ and h is the hidden dimension which is a hyper-parameter. In this way, rating matrix can be expressed as $\mathbf{R} = PQ^T$. Usually, $h \ll min(m, n)$. The Space complexity is $O(m * n)$ for $\mathbf{R}$ and $O((m + n) * h)$ for P and Q, where $O((m + n) * h) \ll O(m * n)$. So MF can be used for space saving strategy in some scenarios. Beyond that, the factorized P and Q can be used for unseen user-item pair prediction, which is the core of MF.

III. Experiments

A. Dataset

We conduct all the experiments on MovieLens-100k dataset, which contains 10,000 comments from 943 users out of 1682 movies. Each comment's form is $(user_{id}, item_{id}, rating)$, where $rating \in \{1, 2, 3, 4, 5\}$. We randomly split the dataset into three part: 70,000 training examples, 10,000 validation examples and 20,000 test examples.

B. Implementation

We implement the Matrix Factorization using python and mainly rely on the numpy package. The initializer for P and Q is Xavier uniform initializer, which initial matrix M, but bounded to interval $[-a * b, a * b]$, where n is $M \in \mathbb{R}^{a \times b}$. We use Adam optimizer with leraning rate 0.001, set batch size to 256 and epoch to 200. We use MAE as evalutation metric. If not sepecifficly stated, we use $h = 10$, MSE as loss function and l_2 $Regularizer$ as regularizer and penalty cofficient 0.1 in regularizer.

C. Hidden dimension's impact

In this section, we explore the influence of hidden dimension in MF. We validate the hidden dimension in MF in a range, e.g. 1, 5, 10, 15, 20, 25, 30.

From Fig 1 we can see that, bigger h tends to have overfitting problem. As presented in Table I, $d = 10$ gets the best performance in validation set. Surprisingly, $h = 1$ still get MAE about 0.7522 in validation set, which is not very far from the best performance e.g. 0.7259, but $h = 1$ can be very time-efficiency and space-saving.

From Fig 2 and Fig 3 we can see that, bigger h converges faster.

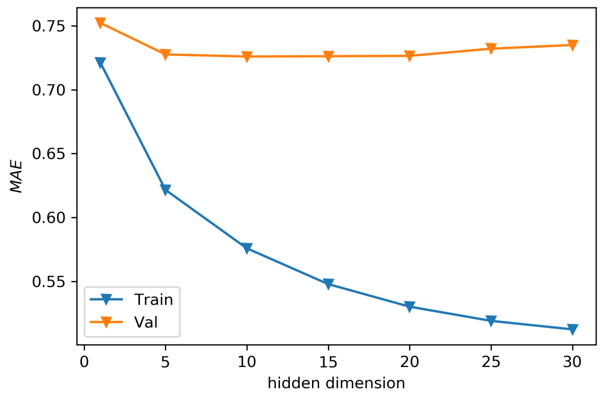

Fig. 1. MF MAE under different h.

TABLE I
MTC MAE under different h. Best result are in bold.

h	1	5	10	15	20	25	30
MAE_{train}	0.7211	0.6216	0.5758	0.5478	0.5302	0.5191	**0.5124**
MAE_{val}	0.7522	0.7276	**0.7259**	0.7262	0.7265	0.7321	0.7349

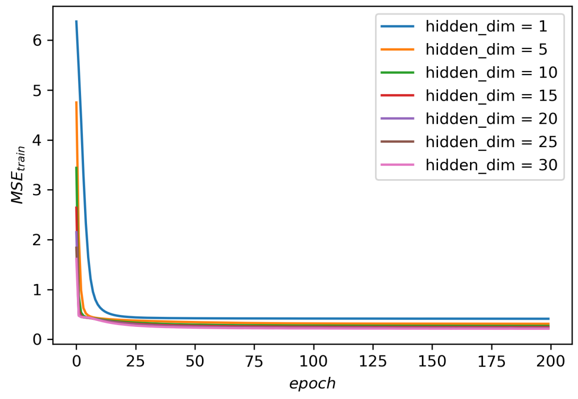

Fig. 2. MF MSE_{train} under different d and $epoch$.

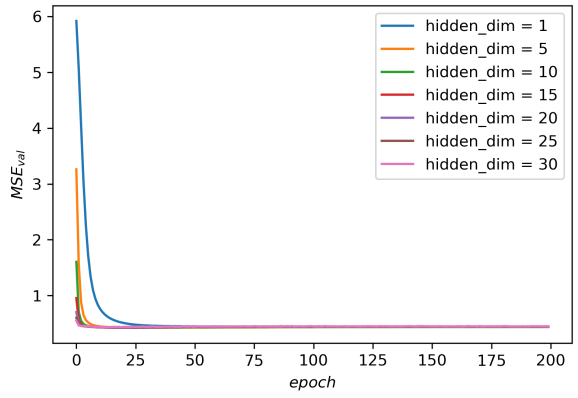

Fig. 3. MF MSE_{val} under different d and $epoch$.

D. Loss function's impact

In this section, we explore the impact of loss function in MF. In particular, we evaluate three loss functions:

1. $MAE(y, \hat{y}) = \frac{1}{m}\Sigma_{i=1}^{m}|y_i - \hat{y}_i|$

2. $MSE(y, \hat{y}) = \frac{1}{2m}\Sigma_{i=1}^{m}(y_i - \hat{y}_i)^2$

3. $HB(y, \hat{y}) = \frac{1}{m}\Sigma_{i=1}^{m}L_\delta(y_i, \hat{y}_i)$

where, $L_\delta(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \le \delta, \\ \delta\,|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$

We set $\delta = 1$.

Three loss functions are mean absolute error, mean squared error and huber loss respectively. The plot of $MAE(x)$, $MSE(s)$ and $HB(x)$ can be found in Fig 4. Huber Loss can been viewed as a combination of MSE and MAE.

From Fig 5, we can see that, HB gets the best performance on validation set. Intuitively, HB takes MAE's adavantage that robust to outliers and MSE's adavantage that smooth in low error region, so it's prone to perform well. The detailed result can be found in Table II.

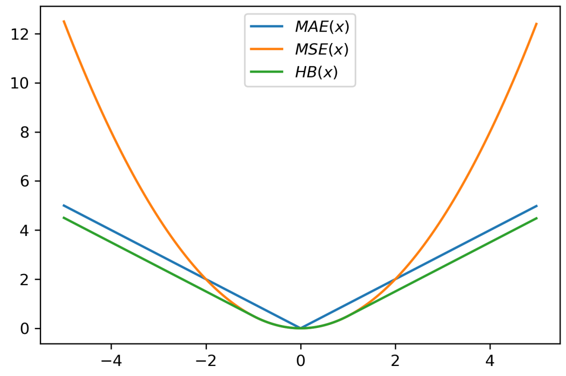

Fig. 4. MAE, MSE and HB plot.

TABLE II
MF'S PERFORMANCE UNDER DIFFERENT LOSS FUNCTION SETTINGS. BEST RESULT ARE IN BOLD.

$loss_{fn}$	MAE	MSE	HB
MAE_{val}	0.7293	0.7244	**0.7217**

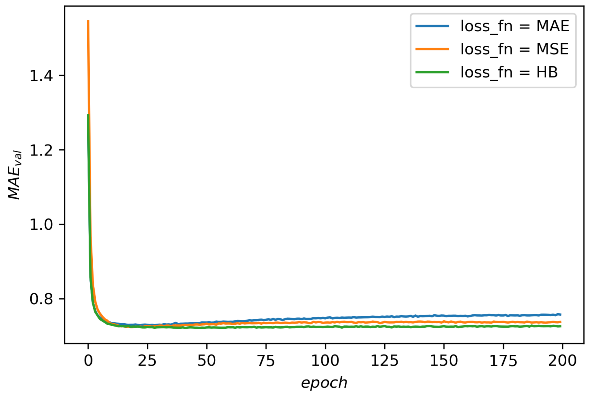

Fig. 5. MF MAE_{val} under different loss functions.

E. Regularizer's impact

In this section, we explore the impact of regularizer in MF. In particular, we evaluate three regularizers:

1. $L1(w) = \lambda * \Sigma_i|w_i|$

2. $L2(w) = \lambda * \Sigma_i w_i^2$

3. $L1\_L2(w) = \frac{L1(w)+L2(w)}{2}$

We set $\lambda = 0.01$ in all experiments.

$L1\_L2$ regularizer can be seen as a combination of $L1$ and $L2$ regularizer. The plot of three regularizers can be found in Fig 6.

From Fig 7, we can see that, $L2$ gets the best performance on validation set. $L1\_L2$ has sever overffiting problem than $L1$ and $L2$ after many epoches. However, $L1\_L2$ gets better performance than $L1$.

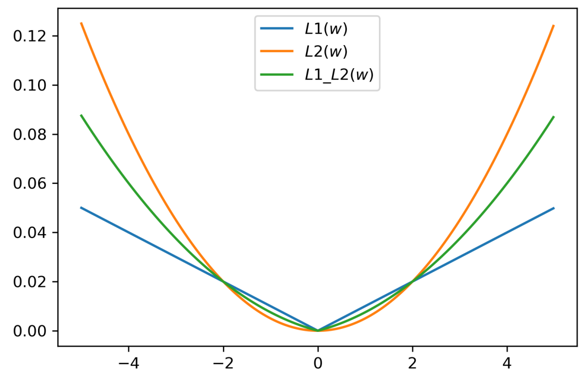

Fig. 6. $L1$, $L2$ and $L1\_L2$ plot.

TABLE III
MF'S PERFORMANCE UNDER DIFFERENT REGULARIZER SETTINGS. BEST RESULT ARE IN BOLD.

$Regularizer$	$L1$	$L2$	$L1\_L2$
MAE_{val}	0.7537	**0.7243**	0.7358

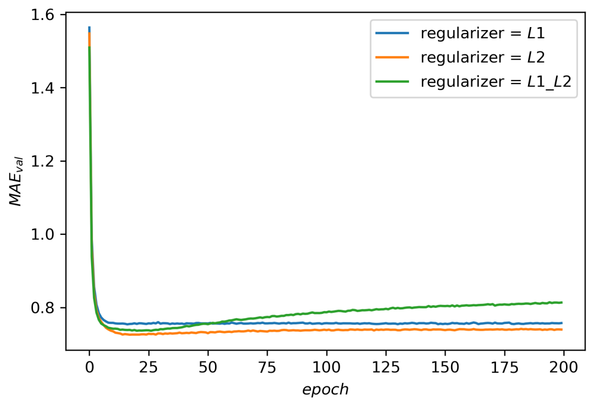

Fig. 7. MF MAE_{val} under different regularizer settings.

F. Final Performance

In this section, we report the final performance under best settings, e.g. $h = 10$, $loss_{fn} = HB$ and $regularizer = L2$. Best result can be found in Table IV.

TABLE IV
MF'S BEST PERFORMANCE UNDER BEST SETTINGS.

MAE_{train}	MAE_{val}	MAE_{test}
0.6342	0.7246	0.7294

IV. CONCLUSION

In this report, we explore Matrix Factorization algorithm. Specifically, we explore the hidden dimension h, loss function and regularizer in MF. We oberseve that smaller h causes under-fitting problem while bigger h causes over-fitting problem. Under the settings that using Huber Loss as loss function and L2 as regularizer, the best perfomance can be obtained.