

MATH 482  
Matrix Factorisation Project  
Code Available: <https://github.com/danielbraithwt/MATH-482>

Daniel Braithwaite

June 21, 2017

## 1 Introduction

To discuss the idea of matrix factorisation and methods to solve it first we must understand the motivation for wanting to solve such a problem. In the case of the Netflix challenge the problem was to build a system to recommend movies to users. We have this very large matrix  $R$  with the rows corresponding to a user and a column corresponding to a movie. The entry  $R_{i,j}$  is the rating that user  $i$  gave movie  $j$ , in practice we would find that a very small percentage of this matrix would be filled in. To make recommendations we would like to predict the ratings which a user might give a movie which they haven't watched.

## 2 Matrix Factorization Solutions

### 2.1 Solution 1: $R = U \cdot M$

The first solution we consider is that  $R$  (an  $u \times m$  matrix) is actually the product of two smaller matrices  $U$  and  $M$ . Where  $U$  (a  $u \times k$  matrix) represents the users in some latent feature space and  $M$  (a  $m \times k$  matrix) represents the movies in the latent feature space. We consider  $M_{i,j}$  to be the amount movie  $i$  has feature  $j$ , likewise we consider  $U_{i,j}$  to be how much user  $i$  is interested in movies with feature  $j$ . Then we can take the rating user  $i$  gives movie  $j$  to be  $\hat{R}_{i,j} = \text{row}(U, i)^T \cdot \text{row}(M, j)$ . Now the problem becomes how do we learn these matrices  $U$  and  $M$ .

We consider the following optimization problem, where  $G$  contains all pairs  $(i, j)$  for which we know  $R_{i,j}$

$$\arg \min_{U, M} \sum_{(i,j) \in G} (R_{i,j} - \text{row}(U, i)^T \cdot \text{row}(M, j))^2$$

This optimization problem can be solved with gradient descent. We generate a matrix  $R$  which is (20x23) by multiplying two randomly generated matrices

$U$  (20 x 15) and  $M$  (23 by 15). Initially the training set and test set are the same and we train over the entire data set. We train each situation 10 times each from random initial conditions.

latent factors	peformance (SSE)	95% CI
5	10.675	(10.648, 10.702)
10	0.767	(0.767, 0.767)
15	0.053	(0.039, 0.066)

Table 1: Table for latent features vs peformance, over entire data set

However in a real world situation we know that the problems which utilize matrix factorization usially involve factorizing sparse matrices, so what happens when we remove say half the entries. Now our data is split in two, half is our training data and the other half is our test data. The results below are generated by training with the same hyperparamaters as before

latent factors	training (SSE)	training 95% CI	test (SSE)	test 95% CI
5	0	0	0	0
10	0	0	0	0
15	0	0	0	0

Table 2: Table for latent features vs peformance, over partial data set

We observe that the test error is higher than the training error, and these differences are stastically significant which indicates that there is overfitting occouring.

One common approach to solve the overfitting that occurs in this type of matrix factorization is to use regularization on the matrices  $U$  and  $M$ . Making our optimization problem the folowing

$$\arg \min_{U, M} \left[ \sum_{(i,j) \in G} (R_{i,j} - \text{row}(U, i)^T \cdot \text{row}(M, j))^2 \right] + \lambda (\|U\|_2 + \|M\|_2)$$

latent factors	training (SSE)	training 95% CI	test (SSE)	test 95% CI
5	0	0	0	0
10	0	0	0	0
15	0	0	0	0

Table 3: Table for latent features vs performance, over partial data set with regulrization

While this does provide a reduction in overfitting it is sill a significant problem. The next concept we can implement is that of biases, our model that we

are learning is suppose to capture the interactions between the users and movies however we might find that in some cases a perticular user gives mostly low ratings or a perticular movie generally recieves low ratings, these propertys arnt interactions between the users and movies, prehaps the user is just harsh or the movie is simply bad. Biases capture this idea so the model can learn what is truely important.

latent factors	training (SSE)	training 95% CI	test (SSE)	test 95% CI
5	0	0	0	0
10	0	0	0	0
15	0	0	0	0

Table 4: Table for latent features vs peformance, over partial data set with regulirzation and biases

## 2.2 Solution 2: Using Neural Networks

In this section we present two similar solutions each using neural networks, only difference being whether we use two neural networks or one.

### 2.2.1 Two Neural Networks

In the same set up as before there is a matrix  $R$  with rows representing users and columns representing movies. Our aim is to optimize the folowing. Take two nerual networks  $f_\theta$  which takes a row of  $R$  to some latent feature space and  $f_\phi$  which takes columns of  $R$  to some feature space. Then we compute the ranking user  $i$  gives movie  $j$  by the folowing  $\hat{R}_{i,j} = f_\theta(user_i)^T \cdot f_\phi(movie_j)$ . Giving us the folowing optimization problem (where  $G$  is defined as before)

$$\arg \min_{\theta, \phi} \sum_{(i,j) \in G} (R_{i,j} - f_\theta(user_i)^T \cdot f_\phi(movie_j))^2$$

### 2.2.2 Single Neural Network

This approach is very similar to the one just presented, how ever insted now we only have one neural network  $f_\psi$ , which takes some row of  $R$  representing a user and some column of  $R$  representing a movie and outputs a rating. Making our approximation of ratings  $\hat{R}_{i,j} = f_\psi(user_i, movie_j)$ , and finally giving us the folowing optimization problem.

$$\arg \min_{\theta, \phi} \sum_{(i,j) \in G} (R_{i,j} - f_\psi(user_i, movie_j))^2$$

### **3 Factorization Method Comparason**

We wish to compare the peformance of these methods against each other and identify the tradeoffs between them. Testing our factorization methods on randomly generated data is a good way to develop an understadning in a small environment but really we want to see how these methods peform on real data. We will be using the MovieLens 100K data set, consiting of 943 users and 1682 movies where each user has rated atleast 20 movies.

#### **3.1 Conclusions**