MATH 482

Matrix Factorisation Project

Code Available: https://github.com/danielbraithwt/MATH-482

Daniel Braithwaite

June 2, 2017

1 Introduction

To dicuss 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 Netflicks challenge the problem was to build a system to recomend movies to users. We have this very large martix R with the rows corosponding to a user and a column corosponding 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 recomendations we would like to predict the ratings which a user might give a movie which they havent watched.

2 Matrix Factorization Solutions

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

The first soluton we consider is that R (an uxm matrix) is actually the product of two smaller matricies U and M. Where U (a uxk matrix) represents the users in some latent feature space and M (a mxk matrix) represents the movies in the latent feature space. We consider $M_{i,j}$ to be the ammount 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 $\hat{R}_{i,j} = row(U,i)^T \cdot row(M,j)$. Now the problem becomes how do we learn these matricies U and M.

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

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

This optimization problem can be solved with gradient decent

2.2 Peformance

Using some randomly generated data we will fix the number of latent factors and run a grid search over the learning rate and training time. The grid we will use is $\eta \in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ and $n \in \{5000, 10000, 15000\}$, 10-Fold Cross Validation will be used to compare the different configurations. For latent factors 3,5,8 we get the folowing configurations which give the best peformance. However there is alot of overfitting occouring in our models, as demonstraited by there being a very low training loss and a disproportiantly high test loss, this indicates a definit need for some regularazation. In practice the standard method used with this sort of matrix factorizeation is to regularize on the matricies U and M. Making our optimization problem the folowing

$$\underset{U,M}{\arg\min} \; \Big[\sum_{(i,j)\in G} (R_{i,j} - row(U,i)^T \cdot row(M,j))^2\Big] + \lambda(\|U\|_2 + \|M\|_2)$$

However now we have another paramater to add to our grid search, our regularizer term λ

latent factors	η	λ	n

2.3 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.

2.3.1 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_θ which takes a row of R to some latent feature space and f_ϕ 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)

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

2.3.2 Single Neural Network

This approach is very similar to the one just presented, how ever insted now we only have one neural network f_ψ, 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.

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

3 Factorization Method Comparason

We wish to compare the peformance of these methods against each other and identify the tradeoffs between them. We will experement with the folowing three cases

1. $R = A \cdot B$
2. $R = f(A) \cdot g(B)$
3. The MovieLens 100K data set, which is some real world data.

For each of the cases we wish to compare there best peformance so we will peform a grid search across the hyperparamaters. For simplisicity we will fix the number of latent features.

We would expect similar peformance for (1) and expect to notice the neural network solutions peforming better for (2) as the matrix decomposition cant handle this case.

3.1 $R = A \cdot B$

3.2 $R = f(A) \cdot g(B)$

3.3 MovieLens 100K

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.

3.4 Conclusions