MATH 482

Matrix Factorisation Project

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

Daniel Braithwaite

May 24, 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 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 $R_{i,j}$

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

This optimization problem can be solved with gradient decent

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.

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)

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

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.

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