

- Will
- Contact Us
- My Contributions
- Log Out

# ML:Recommender Systems

## From Coursera

### Contents

- 1 Problem Formulation
- 2 Content Based Recommendations
- 3 Collaborative Filtering
- 4 Collaborative Filtering Algorithm
- 5 Vectorization: Low Rank Matrix Factorization
- 6 Implementation Detail: Mean Normalization

## Problem Formulation

Recommendation is currently a very popular application of machine learning.

Say we are trying to recommend movies to customers. We can use the following definitions

$n_u$  = number of users

$n_m$  = number of movies

$r(i, j) = 1$  if user  $j$  has rated movie  $i$

$y(i, j)$  = rating given by user  $j$  to movie  $i$  (defined only if  $r(i, j) = 1$ )

## Content Based Recommendations

We can introduce two features,  $x_1$  and  $x_2$  which represents how much romance or how much action a movie may have (on a scale of 0 – 1).

One approach is that we could do linear regression for every single user. For each user  $j$ , learn a parameter  $\theta^{(j)} \in \mathbb{R}^3$ . Predict user  $j$  as rating movie  $i$  with  $(\theta^{(j)})^T x^{(i)}$  stars.

$\theta^{(j)}$  = parameter vector for user  $j$

$x^{(i)}$  = feature vector for movie  $i$

For user  $j$ , movie  $i$ , predicted rating:  $(\theta^{(j)})^T x^{(i)}$

$m^{(j)}$  = number of movies rated by user  $j$

To learn  $\theta^{(j)}$ , we do the following

$$\min_{\theta^{(j)}} = \frac{1}{2} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

This is our familiar linear regression. The base of the first summation is choosing all  $i$  such that  $r(i, j) = 1$ .

To get the parameters for all our users, we do the following:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

We can apply our linear regression gradient descent update using the above cost function.

The only real difference is that we **eliminate the constant**  $\frac{1}{m}$ .

## Collaborative Filtering

It can be very difficult to find features such as "amount of romance" or "amount of action" in a movie. To figure this out, we can use *feature finders*.

We can let the users tell us how much they like the different genres, providing their parameter vector immediately for us.

To infer the features from given parameters, we use the squared error function with regularization over all the users:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

You can also **randomly guess** the values for theta to guess the features repeatedly. You will actually converge to a good set of features.

## Collaborative Filtering Algorithm

To speed things up, we can simultaneously minimize our features and our parameters:

$$J(x, \theta) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

It looks very complicated, but we've only combined the cost function for theta and the cost function for x.

Because the algorithm can learn them itself, the bias units where  $x_0 = 1$  have been removed, therefore  $x \in \mathbb{R}^n$  and  $\theta \in \mathbb{R}^n$ .

These are the steps in the algorithm:

1. Initialize  $x^{(i)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$  to small random values. This serves to break symmetry and ensures that the algorithm learns features  $x^{(i)}, \dots, x^{(n_m)}$  that are different from each other.
2. Minimize  $J(x^{(i)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$  using gradient descent (or an advanced optimization algorithm).  
E.g. for every  $j = 1, \dots, n_u, i = 1, \dots, n_m$ 

$$x_k^{(i)} := x_k^{(i)} - \alpha \left( \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left( \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$
3. For a user with parameters  $\theta$  and a movie with (learned) features  $x$ , predict a star rating of  $\theta^T x$ .

## Vectorization: Low Rank Matrix Factorization

## Implementation Detail: Mean Normalization

Next: Large Scale Machine Learning Back to Index: Main

Retrieved from "https://share.coursera.org/wiki/index.php?title=ML:Recommender\_Systems&oldid=26291"

Category: ML:Lecture Notes

- 
- 
- 

Privacy policy  
About Coursera  
Disclaimers