Recommender Systems

Varun B Patil

Sri Jayachamarajendra College of Engineering, Mysore

2012

Recommender Systems. . . what?

Recommender systems are. . .

Information filtering systems that seek to predict the 'rating' or 'preference' that a user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item or the user's social environment.

Where are they used ?

Mostly in e-commerce...

Where are they used ?

Mostly in e-commerce...

- Netflix suggests movies based on user's previous behavior, user's movie preferences, movie genre and peer ratings.

Where are they used ?

Mostly in e-commerce. . .

- Netflix suggests movies based on user's previous behavior, user's movie preferences, movie genre and peer ratings.
- Amazon suggests books based on user's reading habits, user's book genre preferences.

Where are they used ?

Mostly in e-commerce...

- Netflix suggests movies based on user's previous behavior, user's movie preferences, movie genre and peer ratings.
- Amazon suggests books based on user's reading habits, user's book genre preferences.
- Flipkart suggest electronics and books based on user's previous purchases and based on what other user's with similar preferences bought.

Where are they used ?

Mostly in e-commerce...

- Netflix suggests movies based on user's previous behavior, user's movie preferences, movie genre and peer ratings.
- Amazon suggests books based on user's reading habits, user's book genre preferences.
- Flipkart suggest electronics and books based on user's previous purchases and based on what other user's with similar preferences bought.
- Other examples include Pandora, youtube, Hulu and many many more...

Where are they used ?

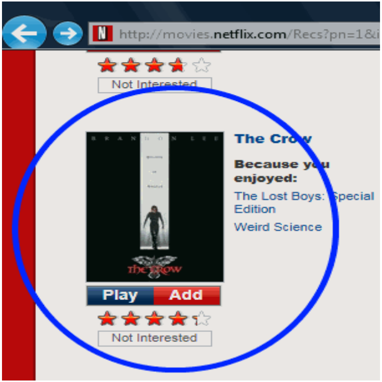
Mostly in e-commerce...

- Netflix suggests movies based on user's previous behavior, user's movie preferences, movie genre and peer ratings.
- Amazon suggests books based on user's reading habits, user's book genre preferences.
- Flipkart suggest electronics and books based on user's previous purchases and based on what other user's with similar preferences bought.
- Other examples include Pandora, youtube, Hulu and many many more...

This is how they make more MONEY !!!

Images to prove the point. . .

Movie recommendations on Netflix

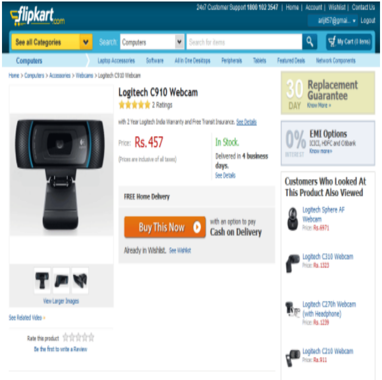


Images to prove the point. . .

Book recommendations on Amazon

Images to prove the point. . .

Recommendations on Flipkart



What is Collaborative filtering?

Collaborative filtering is. . .

A machine learning algorithm that predicts movie ratings for new users (speaking in the context of the movie recommendations system) based on some (possibly none) movies in the database that they have already rated and also based on movie ratings by other users in the systems social environment. These predicted ratings can then be used to recommend top rated movies to the new user.

Why Collaborative filtering?

Advantages of Collaborative Filtering over other algorithms. . .

- Allows us to predict movie ratings for new users even when they have not rated any movies. The most intuitive prediction in this case would be to predict the average rating for each movie.

Why Collaborative filtering?

Advantages of Collaborative Filtering over other algorithms...

- Allows us to predict movie ratings for new users even when they have not rated any movies. The most intuitive prediction in this case would be to predict the average rating for each movie.
- Allows us to predict ratings for new movies even when no user has rated those movies. So you don't need someone to watch every new film and rate it when it is first released.

What is needed ?

The one and only input required is. . .

- A matrix 'Y' storing the current ratings for each movie by each user. 'Y' may contain empty cells where a user has not rated a movie.

What is needed ?

The one and only input required is. . .

- A matrix 'Y' storing the current ratings for each movie by each user. 'Y' may contain empty cells where a user has not rated a movie.

users

$Y =$ movies

4	3	3	?
4	1	?	1
2	5	4	1
2	?	5	3

movie ratings
on scale 1 - 5

n_m X n_u matrix

n_m = number of movies in DB

n_u = number of users in DB

What the algorithm needs to learn ?

Two matrices. . .

- Matrix 'X', the feature vectors for all movies.

What the algorithm needs to learn ?

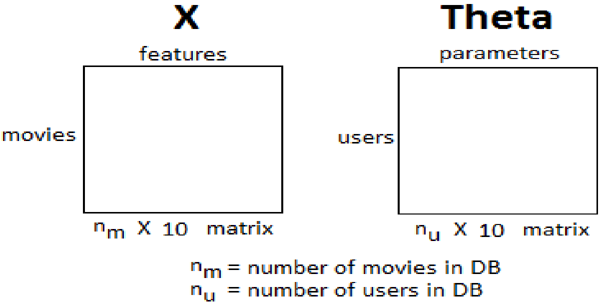
Two matrices. . .

- Matrix 'X', the feature vectors for all movies.
- Matrix 'Theta', the parameter vectors for all users.

What the algorithm needs to learn ?

Two matrices...

- Matrix 'X', the feature vectors for all movies.
- Matrix 'Theta', the parameter vectors for all users.



How predictions are calculated?

- Once the algorithm has learned matrices 'X' and 'Theta' defined previously, the matrix (X*Theta') gives us the matrix of movie rating predictions.

How predictions are calculated?

- Once the algorithm has learned matrices 'X' and 'Theta' defined previously, the matrix (X*Theta') gives us the matrix of movie rating predictions.
- Once we have the matrix (X*Theta'), we can recommend top rated movies for any user simply by extracting those movies with the highest ratings for that particular user.

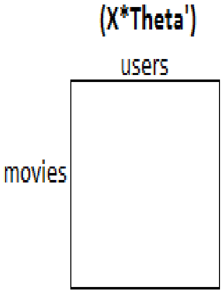
How predictions are calculated?

- Once the algorithm has learned matrices 'X' and 'Theta' defined previously, the matrix (X*Theta') gives us the matrix of movie rating predictions.
- Once we have the matrix (X*Theta'), we can recommend top rated movies for any user simply by extracting those movies with the highest ratings for that particular user.

Theta is a n_u X 10 matrix

X is a n_m X 10 matrix

$\therefore$ (X*Theta') is a n_m X n_u matrix



Learning 'X' and 'Theta' . . .

- 'X' and 'Theta' are two unknowns that need to be learnt.

Learning 'X' and 'Theta' . . .

- 'X' and 'Theta' are two unknowns that need to be learnt.
- One method would be to randomly initialize 'X' and find 'Theta' that minimizes the Cost Function(will be described soon). Then, use this 'Theta' to find 'X' that minimizes the Cost Function. Many iterations like this are carried out. But, this is sort of a chicken-and-egg problem.

Learning 'X' and 'Theta' . . .

- 'X' and 'Theta' are two unknowns that need to be learnt.
- One method would be to randomly initialize 'X' and find 'Theta' that minimizes the Cost Function(will be described soon). Then, use this 'Theta' to find 'X' that minimizes the Cost Function. Many iterations like this are carried out. But, this is sort of a chicken-and-egg problem.
- The solution to this is to randomly initialize both 'X' and 'Theta' and learn them simultaneously (this actually works !!!).

The notion of Cost Function. . .

Our ultimate goal is to learn 'X' and 'Theta' such that the predicted values in (X*Theta') are not very far from the real ratings in matrix 'Y'. Thus, we want to minimize the Cost Function 'J' which is nothing more than the sum of squared error between the actual and predicted ratings and is defined below :

The notion of Cost Function. . .

Our ultimate goal is to learn 'X' and 'Theta' such that the predicted values in (X*Theta') are not very far from the real ratings in matrix 'Y'. Thus, we want to minimize the Cost Function 'J' which is nothing more than the sum of squared error between the actual and predicted ratings and is defined below :

$$J = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

How to minimize the Cost Function ?

- In this regard it is better to use open-source function optimization libraries like fmincg (in Octave) because they are tried-and-tested and very fast and efficient (Why re-invent the wheel ?).

How to minimize the Cost Function ?

- In this regard it is better to use open-source function optimization libraries like fmincg (in Octave) because they are tried-and-tested and very fast and efficient (Why re-invent the wheel ?).
- Such libraries require than you define your own Cost Function 'J' and also provide it gradients for parameters in the function that you wish to minimize (in this case, gradients for 'X' and 'Theta'). These gradients are shown below :

How to minimize the Cost Function ?

- In this regard it is better to use open-source function optimization libraries like fmincg (in Octave) because they are tried-and-tested and very fast and efficient (Why re-invent the wheel ?).
- Such libraries require than you define your own Cost Function 'J' and also provide it gradients for parameters in the function that you wish to minimize (in this case, gradients for 'X' and 'Theta'). These gradients are shown below :

$$\text{X\_grad} = \frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)}$$

$$\boldsymbol{\theta}\text{\_grad} = \frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)}$$

How are gradients useful ?

Once fmincg knows the function to optimize and the gradients of parameters (in this case, the parameters are 'X' and 'Theta') it performs optimization in pretty much the same way as shown below albeit in a much more efficient way.

How are gradients useful ?

Once fmincg knows the function to optimize and the gradients of parameters (in this case, the parameters are 'X' and 'Theta') it performs optimization in pretty much the same way as shown below albeit in a much more efficient way.

$$X = X - \alpha(X\_grad)$$

$$\theta = \theta - \alpha(\theta\_grad)$$

where α is called " learning rate "

How are gradients useful ?

Once fmincg knows the function to optimize and the gradients of parameters (in this case, the parameters are 'X' and 'Theta') it performs optimization in pretty much the same way as shown below albeit in a much more efficient way.

$$X = X - \alpha(X\_grad)$$

$$\theta = \theta - \alpha(\theta\_grad)$$

where α is called " learning rate "

- Large 'alpha' - optimization may never converge.

How are gradients useful ?

Once fmincg knows the function to optimize and the gradients of parameters (in this case, the parameters are 'X' and 'Theta') it performs optimization in pretty much the same way as shown below albeit in a much more efficient way.

```
X = X - α(X_grad)

θ = θ - α(θ_grad)
```

where α is called " learning rate "

- Large 'alpha' - optimization may never converge.
- Small 'alpha' - takes too long to converge.

Some implementation intricacies. . .

Feature Scaling or normalization. . .

- We use randomly initialized movie feature vectors 'X'.

Some implementation intricacies. . .

Feature Scaling or normalization. . .

- We use randomly initialized movie feature vectors 'X'.
- Features with larger values have larger influence on the Cost Function and is undesirable.

Some implementation intricacies. . .

Feature Scaling or normalization. . .

- We use randomly initialized movie feature vectors 'X'.
- Features with larger values have larger influence on the Cost Function and is undesirable.
- We need to make sure that all features fall in the same range. This is achieved through feature scaling.

Some implementation intricacies...

Feature Scaling or normalization...

- We use randomly initialized movie feature vectors 'X'.
- Features with larger values have larger influence on the Cost Function and is undesirable.
- We need to make sure that all features fall in the same range. This is achieved through feature scaling.

$$X = \frac{X - (\text{mean})}{\text{variance}}$$

Some implementation intricacies. . .

Mean normalization of ratings. . .

- Suppose a new user does not rate any movie but still wants movie recommendations.

Some implementation intricacies. . .

Mean normalization of ratings. . .

- Suppose a new user does not rate any movie but still wants movie recommendations.
- fmincg will learn a parameter vector Theta of all zeros for that user. Thus, (X*Theta') will give zero rating for all movies for that user. This is undesirable and would be intuitive if we can predict average movie ratings for that user.

Some implementation intricacies. . .

Mean normalization of ratings. . .

- Suppose a new user does not rate any movie but still wants movie recommendations.
- fmincg will learn a parameter vector Theta of all zeros for that user. Thus, (X*Theta') will give zero rating for all movies for that user. This is undesirable and would be intuitive if we can predict average movie ratings for that user.
- So, we mean normalize the movie ratings database 'Y'. Doing this will intuitively predict average movie ratings for the special case.

Some implementation intricacies. . .

Mean normalization of ratings. . .

- Suppose a new user does not rate any movie but still wants movie recommendations.
- fmincg will learn a parameter vector Theta of all zeros for that user. Thus, (X*Theta') will give zero rating for all movies for that user. This is undesirable and would be intuitive if we can predict average movie ratings for that user.
- So, we mean normalize the movie ratings database 'Y'. Doing this will intuitively predict average movie ratings for the special case.

$$Y_i = Y_i - \text{mean}$$

for each rating Y_i in the database

Some implementation intricacies. . .

Regularization. . .

- Regularization is needed in order to prevent over-fitting of parameters, where the learnt parameters fit the training set very well(almost too perfectly), but fail to perform well on the test set.

Some implementation intricacies...

Regularization...

- Regularization is needed in order to prevent over-fitting of parameters, where the learnt parameters fit the training set very well(almost too perfectly), but fail to perform well on the test set.
- So, we add a regularization term to the Cost Function 'J' and the gradients as shown below. The parameter 'lambda' is called the regularization parameter.

Some implementation intricacies. . .

Regularization. . .

- Regularization is needed in order to prevent over-fitting of parameters, where the learnt parameters fit the training set very well(almost too perfectly), but fail to perform well on the test set.
- So, we add a regularization term to the Cost Function 'J' and the gradients as shown below. The parameter 'lambda' is called the regularization parameter.

$$\mathrm{J} = \mathrm{J} + \left(\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2\right) + \left(\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2\right)$$

$$\mathrm{X\_grad} = \mathrm{X\_grad} + \lambda x_k^{(i)}$$

$$\boldsymbol{\theta}\mathrm{\_grad} = \boldsymbol{\theta}\mathrm{\_grad} + \lambda\theta_k^{(j)}$$