**Recommender Systems**

# Problem Formulation

[Let's say you let](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [your users rate different movies,](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [using a 1 to 5 star rating.](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [So, users may, you know,](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [something one, two, three, four or five stars.](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [In order to make this example](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [just a little bit nicer, I'm](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [going to allow 0 to](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [5 stars as well,](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation) [because that just makes some of the math come out just nicer.](https://www.coursera.org/learn/machine-learning/lecture/Rhg6r/problem-formulation)

Some notation

* nu - Number of users (called ?nu occasionally as we can't subscript in superscript)
* nm- Number of movies
* r(i, j) - 1 if user j has rated movie i (i.e. bitmap)
* y(i,j) - rating given by user j to move i (defined only if r(i,j) = 1)

y(i, j) would be a number from zero to five, depending on the star rating, zero to five stars that user gave that particular movie.

# Content Based Recommendations

We could treat each rating for each user as a separate linear regression problem.

 If we have features like x1=0.9,x2=0 then each film can be recommended by a feature vector

* Add an extra feature which is x0 = 1 for each film
* So for each film we have a [3 x 1] vector, which for film number 1 ("Love at Last") would be  
  
* i.e. for our dataset we have
  + {x1, x2, x3, x4, x5}
    - Where each of these is a [3x1] vector with an x0 = 1 and then a romance and an action score
* To be consistent with our notation, n is going to be the number of features NOT counting the x0 term, so n = 2
* For each user j we could learn a parameter vector
* Then predict that user j will rate movie i with
  + (θj)*T*xi= stars where (θj)
  + [More generally, theta (j) would be an R (n+1),](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations) [where n is the number of features not counting the set term.](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations)
  + inner product of parameter vector and features

[So, all we're doing here is we're applying a different copy of this linear](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations) [regression for each user,](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations).

Problem Formulation

r(i,j) = 1, if user j has rated the movie I (otherwise 0)

y(i,j) = rating by user j on the movie i (if defined)

(θj) = parameter vector for user j

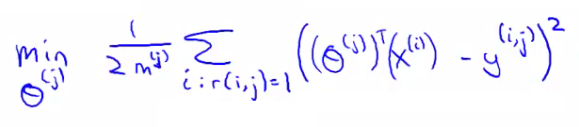
xi  = feature vector for movie i

For user j , movie i predicted rating = (θj)*T*xi

We should also add one final piece of notation

* mj, - Number of movies rated by the user (j)

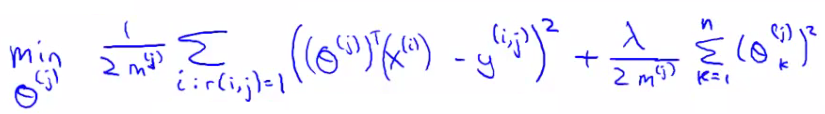
**To learn (θj):-**[So what we can do is just choose a parameter vector theta j so](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations) [that the predicted values here are as close as possible to the values that we](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations) [observed in our training sets and the values we observed in our data.](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations)

 Create some parameters which give values as close as those seen in the data when applied  


[So you'll be summing over all the movies that user j has rated.](https://www.coursera.org/learn/machine-learning/lecture/uG59z/content-based-recommendations)

 Sum over all values of i (all movies the user has used) when r(i,j) = 1 (i.e. all the films that the user has rated)

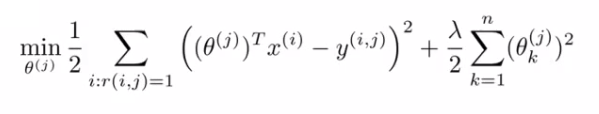
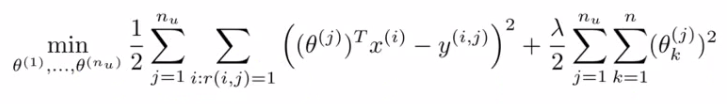
 This is just like linear regression with least-squared error

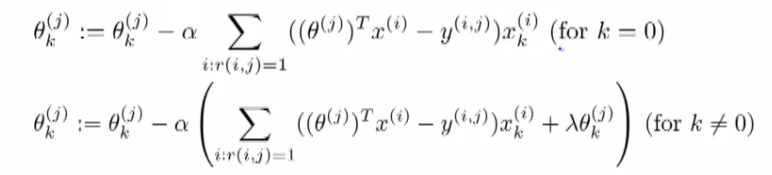
 We can also add a regularization term to make our equation look as follows  


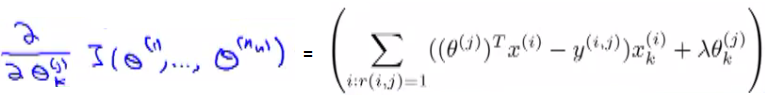
* The regularization term goes from k=1 through to m, so (θj) ends up being an n+1 feature vector
  + Don't regularize over the bias terms (0)

 If you do this you get a reasonable value

 To make this a little bit clearer you can get rid of the mj term (it's just a constant so shouldn't make any difference to minimization)

* So to learn (θj)   
  
* But for our recommender system we want to learn parameters for *all* users, so we add an extra summation term to this which means we determine the minimum (θj) value for every user  
  
* When you do this as a function of each (θj) parameter vector you get the parameters for each user
  + So this is our optimization objective -> J(θ1, ..., θnu)

 In order to do the minimization we have the following gradient descent   


* Slightly different to our previous gradient descent implementations
  + k = 0 and k != 0 versions
  + We can define the middle term above as
* 
  + Difference from linear regression
    - No 1/m terms (got rid of the 1/m term)
    - Otherwise very similar

 This approach is called content-based approach because we assume we have features regarding the content which will help us identify things that make them appealing to a user

# Collaborative Filtering

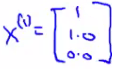
[A](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [recommender system that's called collaborative filtering.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering)

This will be an algorithm that can start to learn for itself what features to use.

[Let's change the problem](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [a bit and suppose that](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [we have a data set where](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [we do not know the values of these features.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [So we're given the data set](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [of movies and of](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [how the users rated them, but we](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [have no idea how romantic each](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [movie is and we have no](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [idea how action packed each](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [movie is so I've replaced all of these things with question marks.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [Let's say we've gone to each of our users, and each of our users has told has told us](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [how much they like the](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [romantic movies and how much](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [they like action packed movies.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering)

 But what we're really asking is

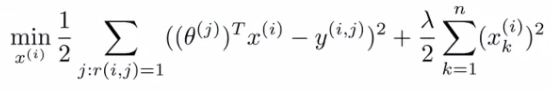
 "What feature vector should x1 be so that

* + (θ1)*T*x1is about 5(They like this feature)
  + (θ2)*T*x1is about 5
  + (θ3)*T*x1is about 0(They do not like this feature)
  + (θ4)*T*x1is about 0
* From this we can guess that x1may be  
   where x0=1.
* Using that same approach we should then be able to determine the remaining feature vectors for the other films

**Formalizing the collaborative filtering problem**

 We can more formally describe the approach as follows

*  Given (θ1, ..., θnu) (i.e. given the parameter vectors for each users' preferences)
* We must minimize an optimization function which tries to identify the best parameter vector associated with a film



[So, just to](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [summarize what this term does](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [is it tries to choose features](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [XI so that for](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [all the users J that](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [have rated that movie, the](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [algorithm also predicts a](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [value for how that user would have rated that movie](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [that is not too far, in](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [the squared error sense, from the](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [actual value that the user had rated that movie.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [So that's the squared error term.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [As usual, we can](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [also add this sort of](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [regularization term to prevent](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering) [the features from becoming too big.](https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering)

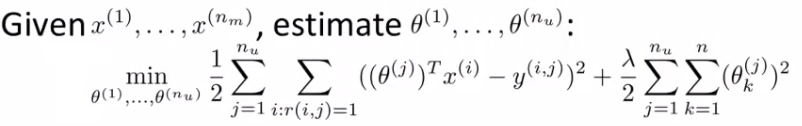
 So we're summing over all the indices j for where we have data for movie (i )we're minimizing this squared error

 Like before, the above equation gives us a way to learn the features for one film

 We want to learn all the features for *all* the films - so we need an additional summation term

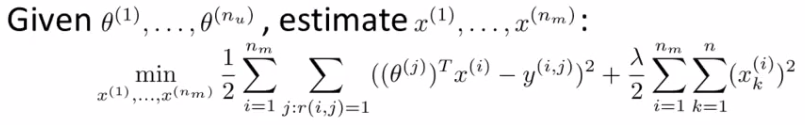
# Collaborative Filtering Algorithm

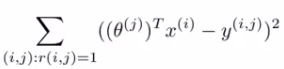
If we're given the film's features, we can use that to work out the users' preference



If we're given the users' preferences, we can use them to work out the film's features [The first summation is sum](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [over all users J and](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [then sum over all movies rated by that user.So, this is really summing over all](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [pairs IJ, that correspond](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [to a movie that was rated by a user.](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [Sum over J says, for every](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [user, the sum of](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [all the movies rated by that user.](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm)

[This summation down here, just does things in the opposite order.](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [This says for every movie](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [I, sum over all](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [the users J that have](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [rated that movie and so, you](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [know these summations, both of these](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [are just summations over all pairs](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [ij for which](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [r of i J is equal to 1.](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [It's just something over all the](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm) [user movie pairs for which you have a rating.](https://www.coursera.org/learn/machine-learning/lecture/f26nH/collaborative-filtering-algorithm)

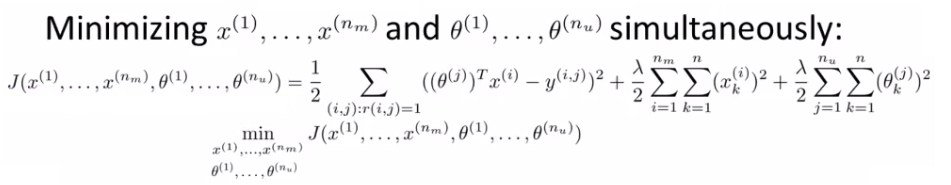


The squared error term is the same as the squared error term in the two individual objectives above  


* So it's summing over every movie rated by every users
* Note the ":" means, "for which"
  + Sum over all pairs (i,j) for which r(i,j) is equal to 1

There's a more efficient algorithm which can solve θ and x simultaneously

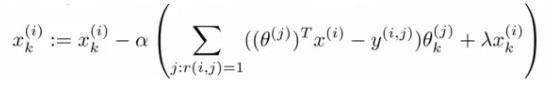
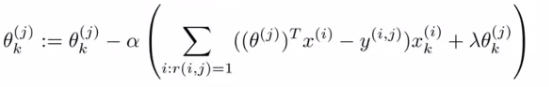
 Define a new optimization objective which is a function of x and θ



When we're learning the features this way

* Previously had a convention that we have an x0 = 1 term
* When we're using this kind of approach we have no x0,
  + So now our vectors (both x and θ) are n-dimensional (not n+1)
* We do this because we are now learning all the features so if the system needs a feature always = 1 then the algorithm can learn one

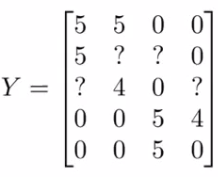
**Algorithm Structure**

* **1)** Initialize θ1, ..., θnuand x1, ..., xnm to small random values
  + A bit like neural networks - initialize all parameters to small random numbers
* **2)** Minimize cost function (J(x1, ..., xnm, θ1, ...,θnu) using gradient descent
  + We find that the update rules look like this  
      
    
  + Where the top term is the partial derivative of the cost function with respect to xki while the bottom is the partial derivative of the cost function with respect to θki
  + So here we regularize EVERY parameters (no longer x0parameter) so no special case update rule
* **3)** Having minimized the values, given a user (user j) with parameters θ and movie (movie i) with learned features x, we predict a start rating of (θj)*T*xi
  + This is the collaborative filtering algorithm, which should give pretty good predictions for how users like new movies

# Vectorization: Low Rank Matrix Factorization

We start by working out another way of writing out our predictions

* So take all ratings by all users in our example above and group into a matrix Y



* 5 movies
* 4 users
* Get a [5 x 4] matrix

The elements of this matrix of the (i, j) element of this matrix is really

what we were previously writing as y superscript i, j.

It's the rating given to movie i by user j.

# Given [Y] there's another way of writing out all the predicted ratings

# C:\Books\Machine Learning\Machine_learning_complete\Machine_learning_complete\16_Recommender_Systems_files\Image [22].png

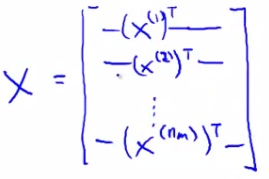
 So here 5 = Theta(1)’\*x(1) where x1= user1 and Theta(1) = 1st user’s rating for 1st movie

For ? = Theta(2)’\*x(2) where x2=user 2 and Theta(2) = 2st user’s rating for 2nd movie

* + With this matrix of predictive ratings
  + We determine the (i,j) entry for EVERY movie

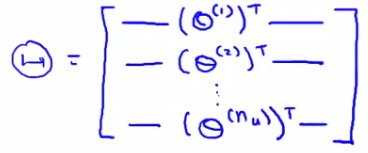
 We can define another matrix X

* Just like matrix we had for linear regression
* Take all the features for each movie and stack them in rows



*  Think of each movie as one example

 Also define a matrix Θ





* + Take each per user parameter vector and stack in rows

 Given our new matrices X and θ

* We can have a vectorized way of computing the prediction range matrix by doing X \* θ*T*

 We can given this algorithm another name - **low rank matrix factorization**

*  This comes from the property that the X \* θ*T*calculation has a property in linear algebra that we create a **low rank** matrix

**Recommending new movies to a user**

Find movies j which is similar to i, which you can recommend

 If we have two movies xi and xj

* We want to minimize ||xi - xj||
  + i.e. the distance between those two movies

 Provides a good indicator of how similar two films are in the sense of user perception

# Implementational Detail: Mean Normalization

Read from the written note.