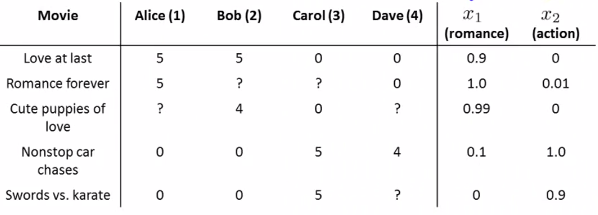
***Collaborative Filtering***

**I. COLLABORATIVE FILTERING**

* Another approach to building a recommender system is called **collaborative filtering** which has a very interesting property called **feature learning 🡪** algorithm that can start to learn for itself what features to use.
* Here was the data set that we had + we assumed that for each movie, someone told us how romantic a movie was or how much action there was in a movie.



* It can be very difficult + time consuming + expensive to actually try to get someone to watch each movie + tell you these things, + often you'll want even more features than just these 2.
* So where do you get these features from? So let's change the problem a bit and suppose that we have a data set where we do not know the values of these features. So we're given the data set of movies and of how the users rated them, but we have no idea how romantic each movie is and we have no idea how action packed each movie is so I've replaced all of these things with question marks. But now let's make a slightly different assumption. Let's say we've gone to each of our users, and each of our users has told has told us how much they like the romantic movies and how much they like action packed movies. So Alice has associated a current of theta 1. Bob theta 2. Carol theta 3. Dave theta 4. And let's say we also use this and that Alice tells us that she really likes romantic movies and so there's a five there which is the multiplier associated with X1 and lets say that Alice tells us she really doesn't like action movies and so there's a 0 there. And Bob tells us something similar so we have theta 2 over here. Whereas Carol tells us that she really likes action movies which is why there's a 5 there, that's the multiplier associated with X2, and remember there's also X0 equals 1 and let's say that Carol tells us she doesn't like romantic movies and so on, similarly for Dave. So let's assume that somehow we can go to users and each user J just tells us what is the value of theta J for them. And so basically specifies to us of how much they like different types of movies. If we can get these parameters theta from our users then it turns out that it becomes possible to try to infer what are the values of x1 and x2 for each movie. Let's look at an example. Let's look at movie 1. So that movie 1 has associated with it a feature vector x1. And you know this movie is called Love at last but let's ignore that. Let's pretend we don't know what this movie is about, so let's ignore the title of this movie. All we know is that Alice loved this move. Bob loved this movie. Carol and Dave hated this movie. So what can we infer? Well, we know from the feature vectors that Alice and Bob love romantic movies because they told us that there's a 5 here. Whereas Carol and Dave, we know that they hate romantic movies and that they love action movies. So because those are the parameter vectors that you know, uses 3 and 4, Carol and Dave, gave us. And so based on the fact that movie 1 is loved by Alice and Bob and hated by Carol and Dave, we might reasonably conclude that this is probably a romantic movie, it is probably not much of an action movie. this example is a little bit mathematically simplified but what we're really asking is what feature vector should X1 be so that theta 1 transpose x1 is approximately equal to 5, that's Alice's rating, and theta 2 transpose x1 is also approximately equal to 5, and theta 3 transpose x1 is approximately equal to 0, so this would be Carol's rating, and theta 4 transpose X1 is approximately equal to 0. And from this it looks like, you know, X1 equals one that's the intercept term, and then 1.0, 0.0, that makes sense given what we know of Alice, Bob, Carol, and Dave's preferences for movies and the way they rated this movie. And so more generally, we can go down this list and try to figure out what might be reasonable features for these other movies as well. Let's formalize this problem of learning the features XI. Let's say that our users have given us their preferences. So let's say that our users have come and, you know, told us these values for theta 1 through theta of NU and we want to learn the feature vector XI for movie number I. What we can do is therefore pose the following optimization problem. So we want to sum over all the indices J for which we have a rating for movie I because we're trying to learn the features for movie I that is this feature vector XI. So and then what we want to do is minimize this squared error, so we want to choose features XI, so that, you know, the predictive value of how user J rates movie I will be similar, will be not too far in the squared error sense of the actual value YIJ that we actually observe in the rating of user j on movie I.
* So, just to summarize what this term does is it tries to choose features XI so that for all the users J that have rated that movie, the algorithm also predicts a value for how that user would have rated that movie that is not too far, in the squared error sense, from the actual value that the user had rated that movie. So that's the squared error term. As usual, we can also add this sort of regularization term to prevent the features from becoming too big. So this is how we would learn the features for one specific movie but what we want to do is learn all the features for all the movies and so what I'm going to do is add this extra summation here so I'm going to sum over all Nm movies, N subscript m movies, and minimize this objective on top that sums of all movies. And if you do that, you end up with the following optimization problem. And if you minimize this, you have hopefully a reasonable set of features for all of your movies. So putting everything together, what we, the algorithm we talked about in the previous video and the algorithm that we just talked about in this video. In the previous video, what we showed was that you know, if you have a set of movie ratings, so if you have the data the rij's and then you have the yij's that will be the movie ratings. Then given features for your different movies we can learn these parameters theta. So if you knew the features, you can learn the parameters theta for your different users. And what we showed earlier in this video is that if your users are willing to give you parameters, then you can estimate features for the different movies. So this is kind of a chicken and egg problem. Which comes first? You know, do we want if we can get the thetas, we can know the Xs. If we have the Xs, we can learn the thetas. And what you can do is, and then this actually works, what you can do is in fact randomly guess some value of the thetas. Now based on your initial random guess for the thetas, you can then go ahead and use the procedure that we just talked about in order to learn features for your different movies. Now given some initial set of features for your movies you can then use this first method that we talked about in the previous video to try to get an even better estimate for your parameters theta. Now that you have a better setting of the parameters theta for your users, we can use that to maybe even get a better set of features and so on. We can sort of keep iterating, going back and forth and optimizing theta, x theta, x theta, nd this actually works and if you do this, this will actually cause your album to converge to a reasonable set of features for you movies and a reasonable set of parameters for your different users. So this is a basic collaborative filtering algorithm. This isn't actually the final algorithm that we're going to use. In the next video we are going to be able to improve on this algorithm and make it quite a bit more computationally efficient. But, hopefully this gives you a sense of how you can formulate a problem where you can simultaneously learn the parameters and simultaneously learn the features from the different movies. And for this problem, for the recommender system problem, this is possible only because each user rates multiple movies and hopefully each movie is rated by multiple users. And so you can do this back and forth process to estimate theta and x.
* So to summarize, in this video we've seen an initial collaborative filtering algorithm. The term collaborative filtering refers to the observation that when you run this algorithm with a large set of users, what all of these users are effectively doing are sort of collaboratively--or collaborating to get better movie ratings for everyone because with every user rating some subset with the movies, every user is helping the algorithm a little bit to learn better features, and then by helping-- by rating a few movies myself, I will be helping the system learn better features and then these features can be used by the system to make better movie predictions for everyone else. And so there is a sense of collaboration where every user is helping the system learn better features for the common good. This is this collaborative filtering. And, in the next video what we going to do is take the ideas that have worked out, and try to develop a better an even better algorithm, a slightly better technique for collaborative filtering.

**II. COLLABORATIVE FILTERING ALGORITHM**