**Recommender Systems**

**A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy.**

***If a completely new user visits an e-commerce site****, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario?*

One possible solution could be **to recommend the best selling products**, i.e. the products which are high in demand. Another possible solution could be to ***recommend the products which would bring the maximum profit to the business.***

If we can recommend a few items to a customer based on their needs and interests, **it will create a positive impact on the user experience and lead to frequent visits.** Hence, businesses nowadays are building smart and intelligent recommendation engines by studying the past behavior of their users.

#### Relationships

#### User-Product Relationship

Some users have an affinity or preference towards specific products that they need.

Eg: A cricket player has a preference for cricket related items thus the e-commerce website will build a user product relation of player->cricket.

#### Product-Product Relationship

When we look within products, some products are similar in nature. Looks or description wise they have similarity between each other.

eg: Books of the same genre, Music from the same genre, Dishes from the same cuisine, News Articles from a particular event.

#### User-User Relationship

Some customers have similar taste with respect to a particular product or service.

eg: Mutual Friends, Similar Backgrounds, Similar Age

These relationships can provide systems with tremendous insight as well as an understanding of customers. The insight understood can be monetized as relevant suggestions often lead to purchase.

#### Data

#### User Behaviour Data

Useful information about the engagement of the user on the product. It can be collected from rating, clicks, purchase history.

#### User Demographic Data

User information related to the user’s personal information such as age, education, income, location.

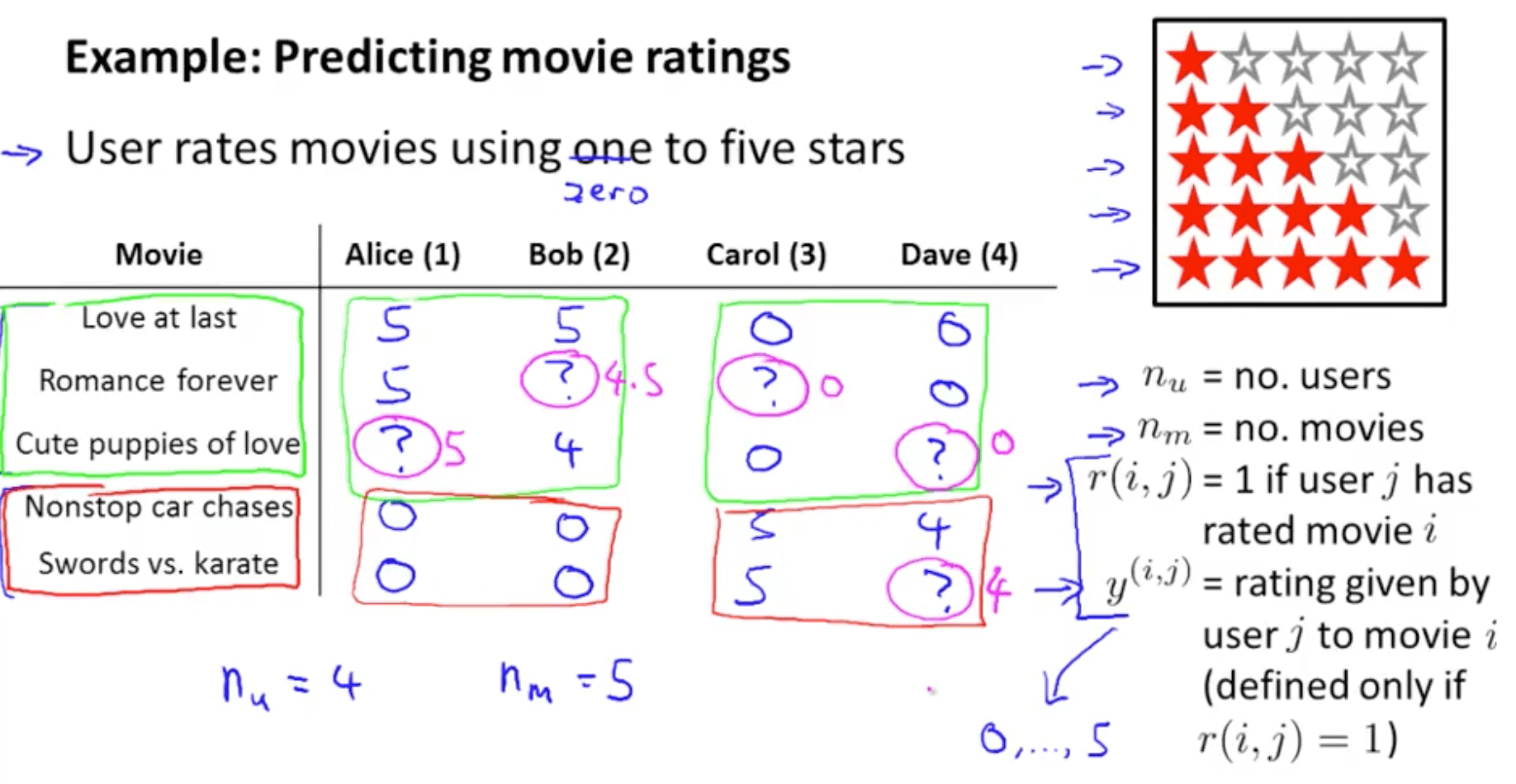
#### Product Attribute Data

Information related to product’s attribute such as genre in case of books, cast in case of movies, cuisine in case of food.

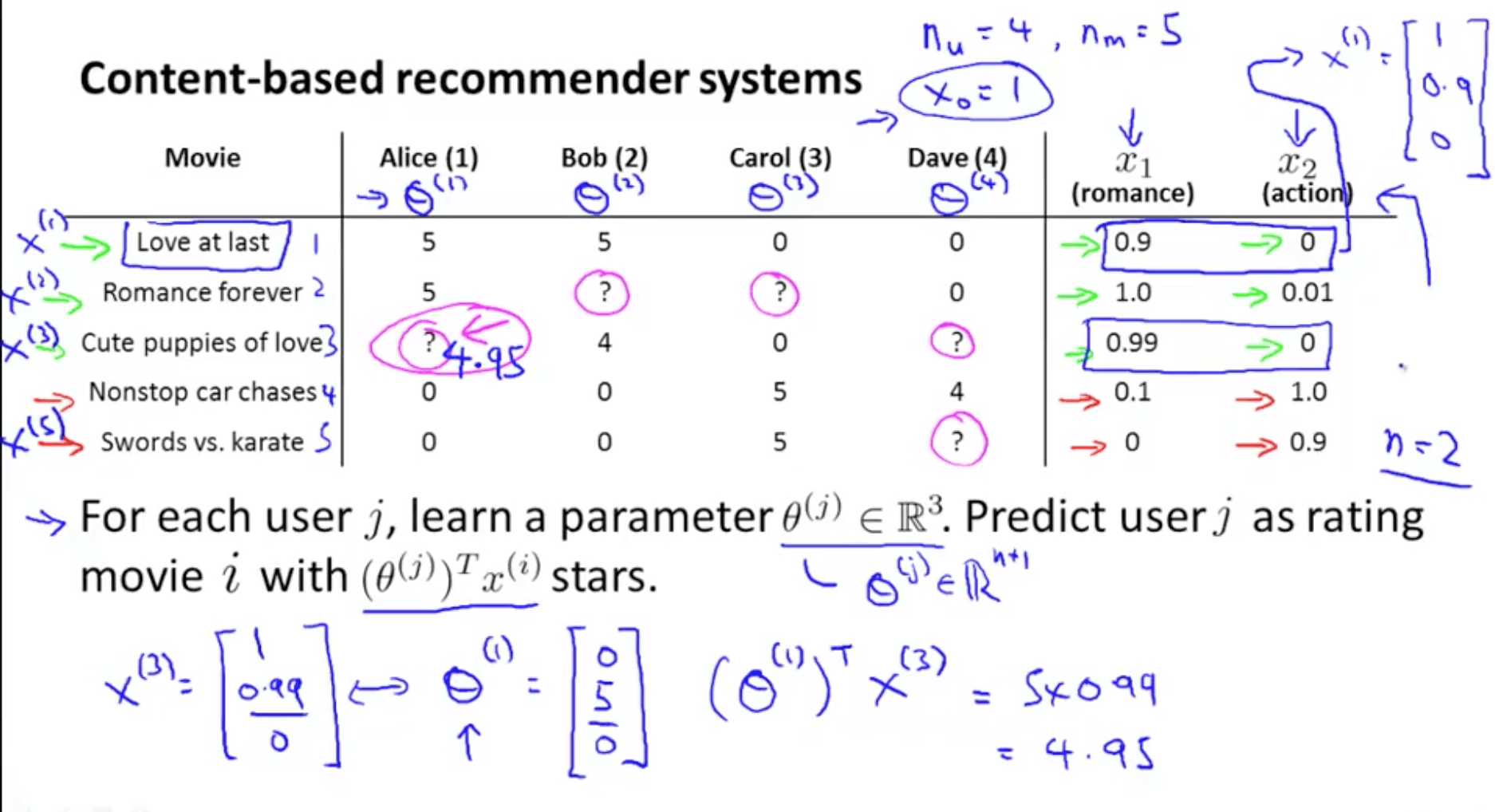
**1. Predicting Movie Ratings**

I would like to give full credits to the respective authors as these are my personal python notebooks taken from deep learning courses from Andrew Ng, Data School and Udemy :) This is a simple python notebook hosted generously through Github Pages that is on my main personal notes repository on <https://github.com/ritchieng/ritchieng.github.io>. They are meant for my personal review but I have open-source my repository of personal notes as a lot of people found it useful.

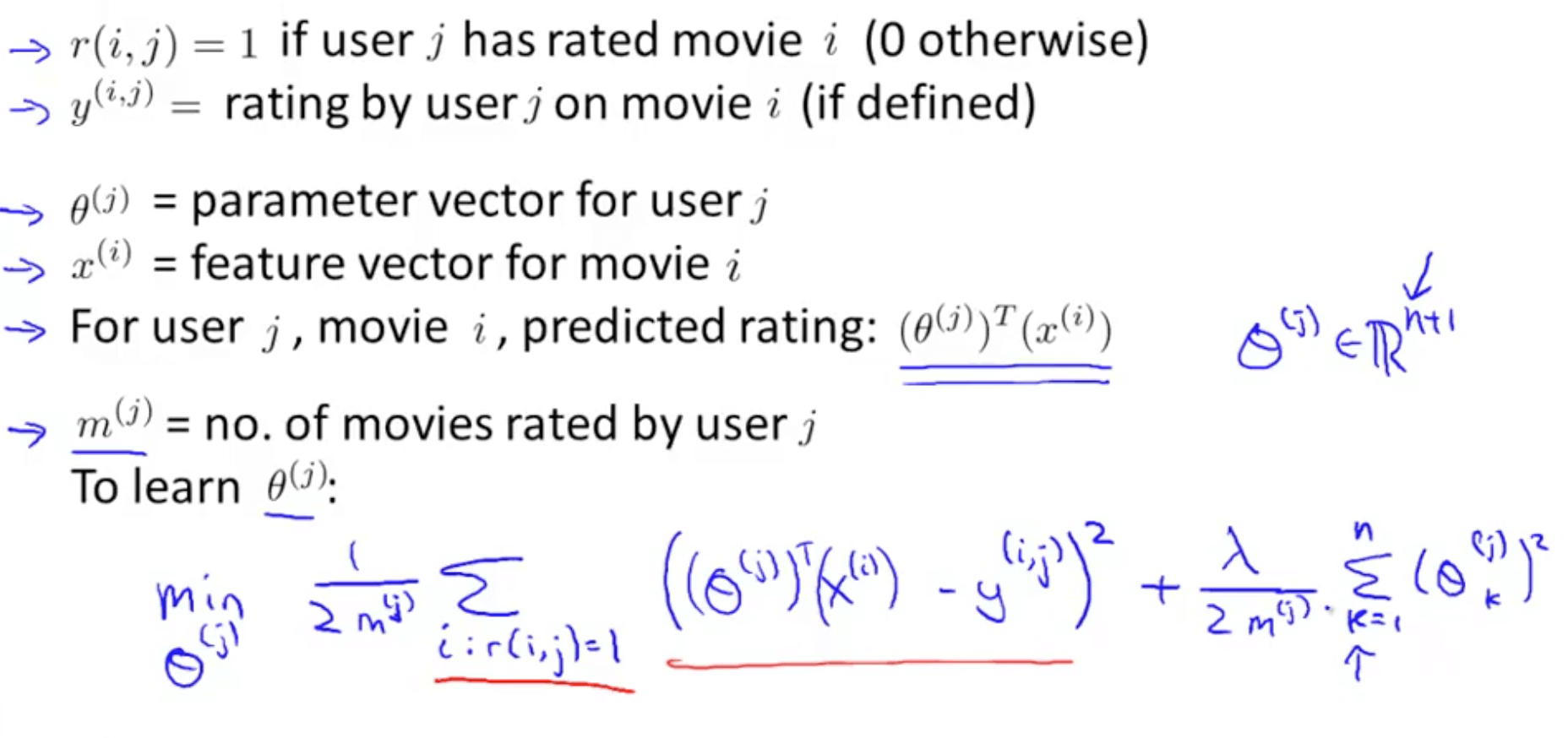
**1a. Problem Formulation**

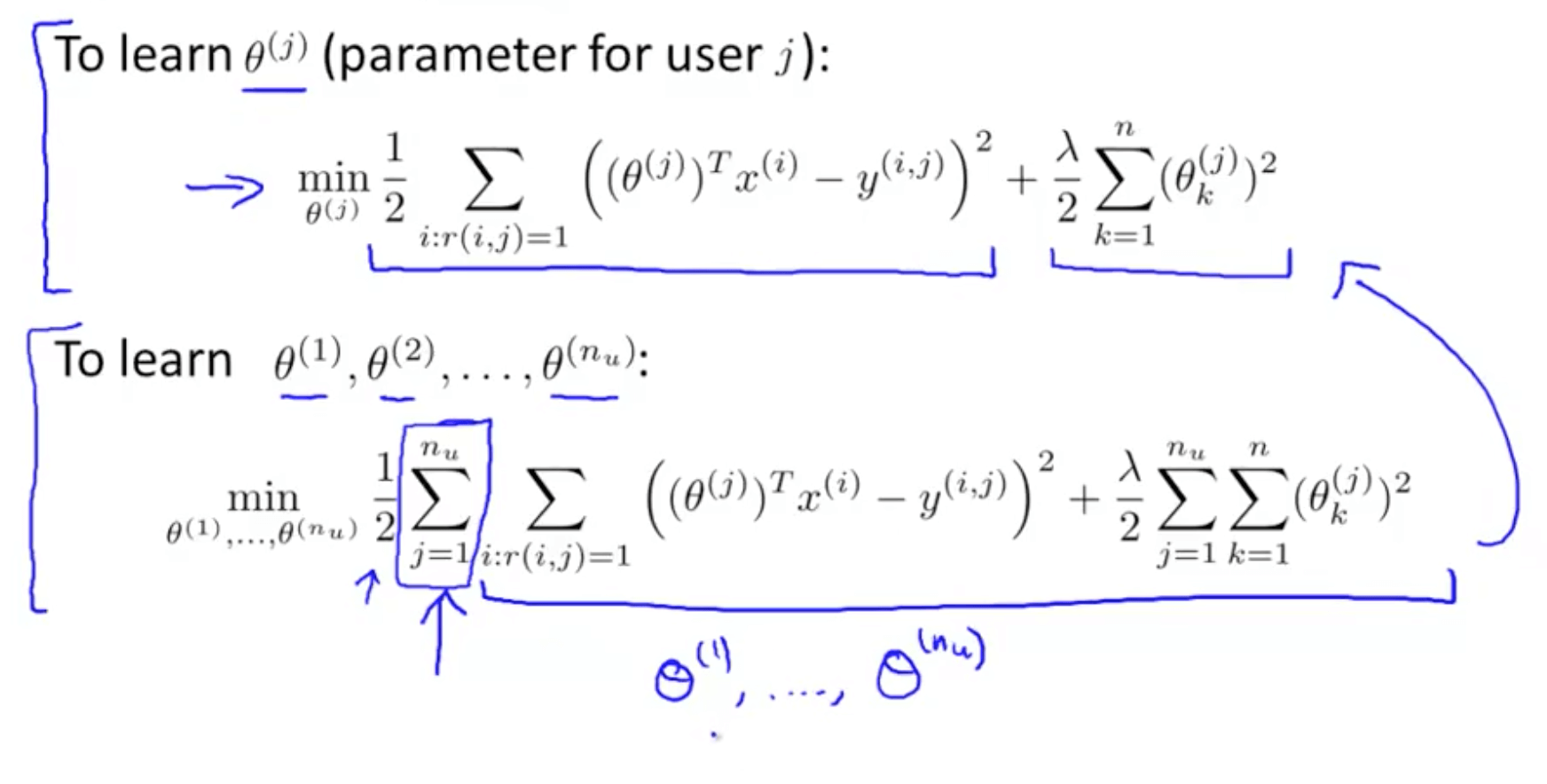


**1b. Content Based Recommendations**

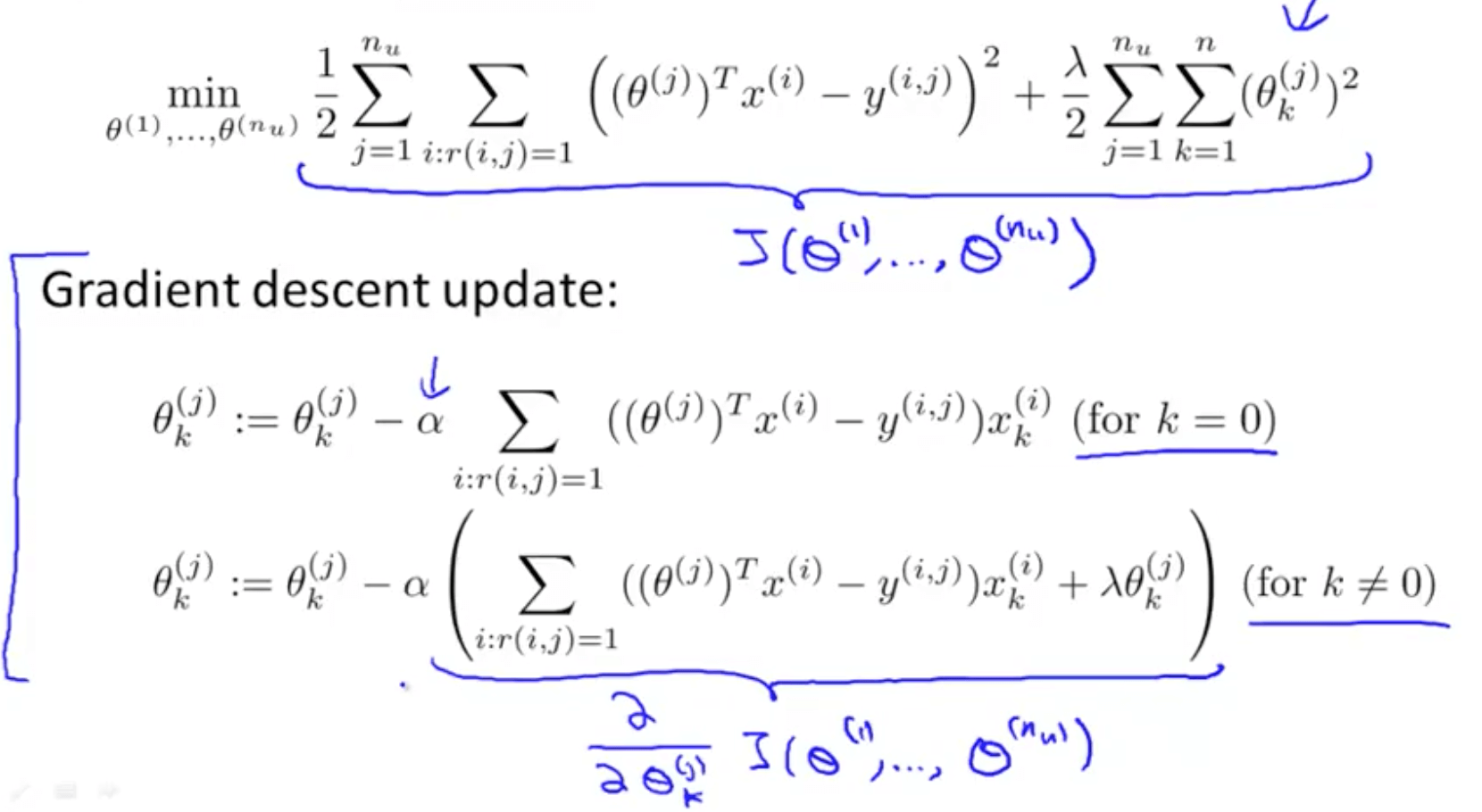
* How do we predict the missing values? 

If we minimize the following function, we get the parameters to predict





We can use other minimization algorithms (other than gradient descent)

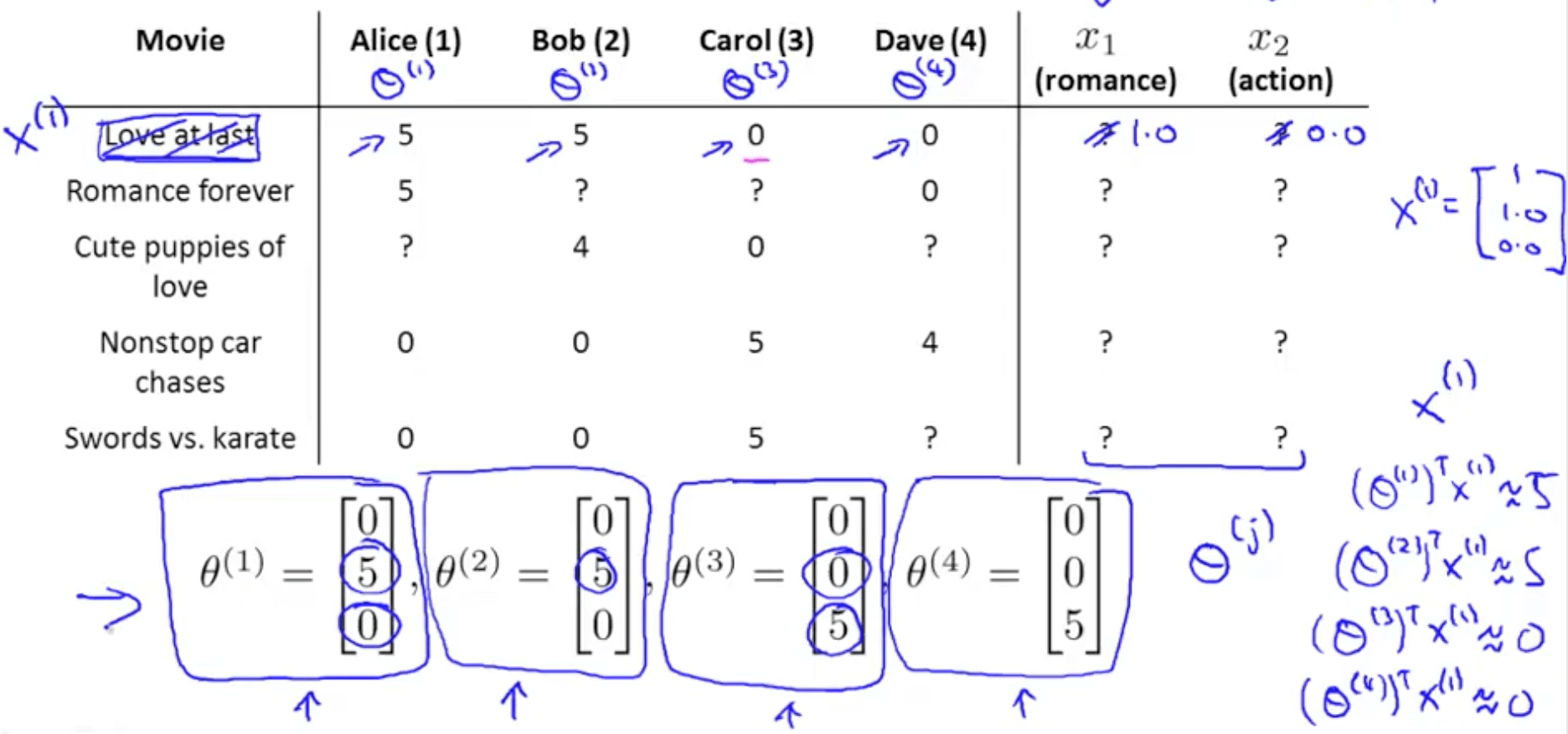


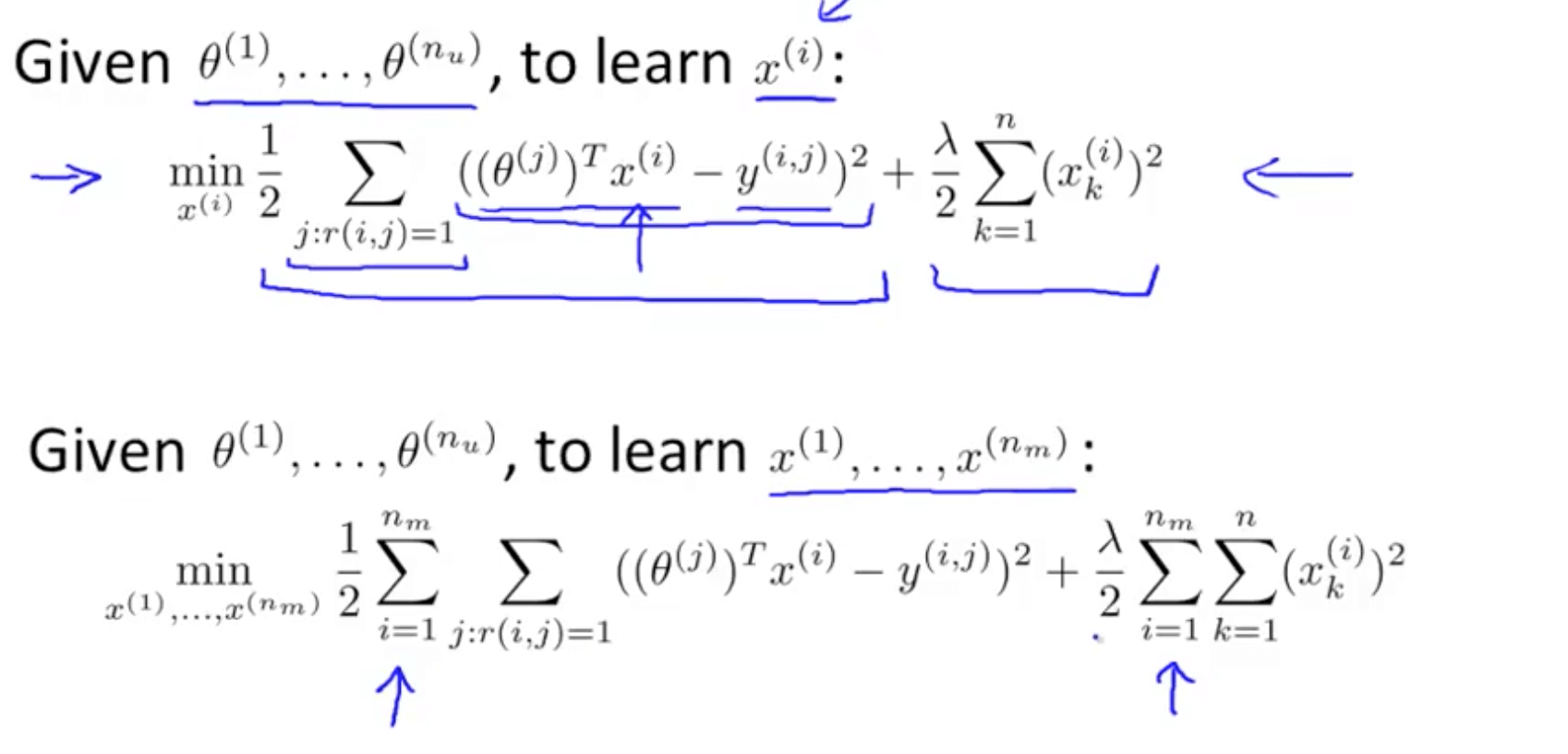
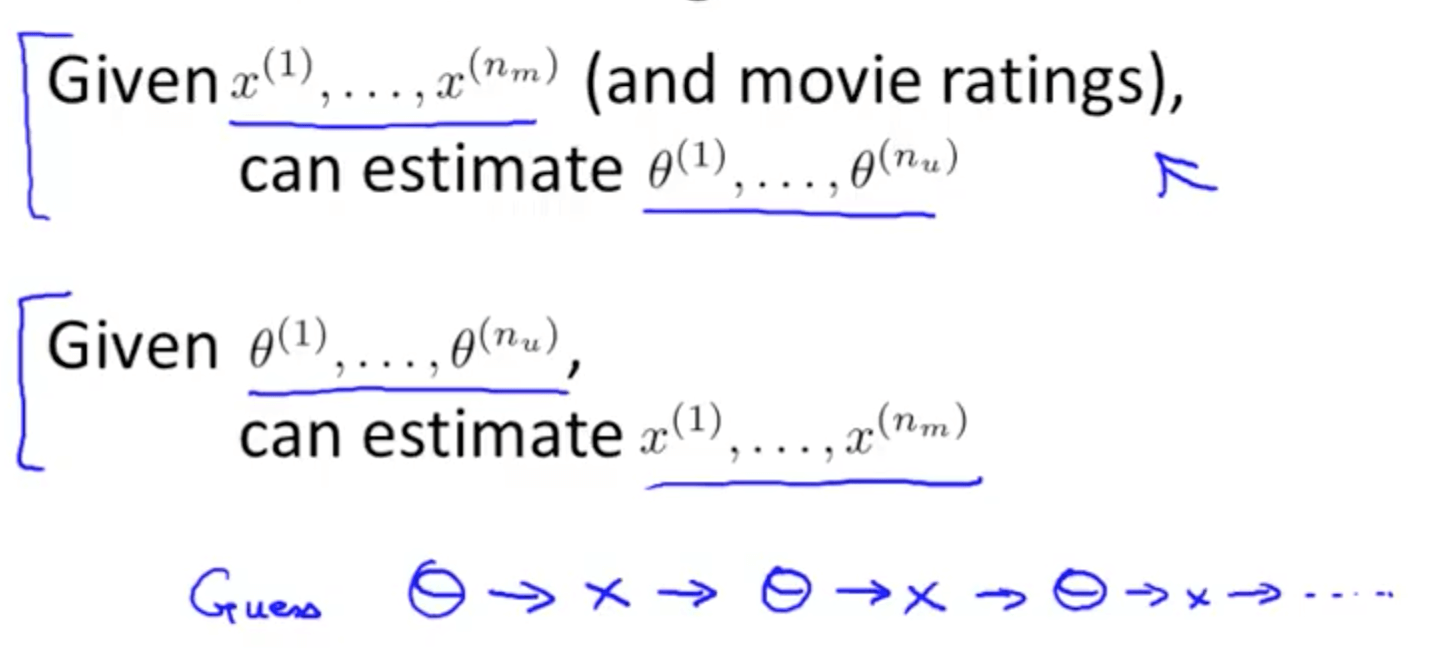
**2. Collaborative Filtering**

**2a. Introduction**

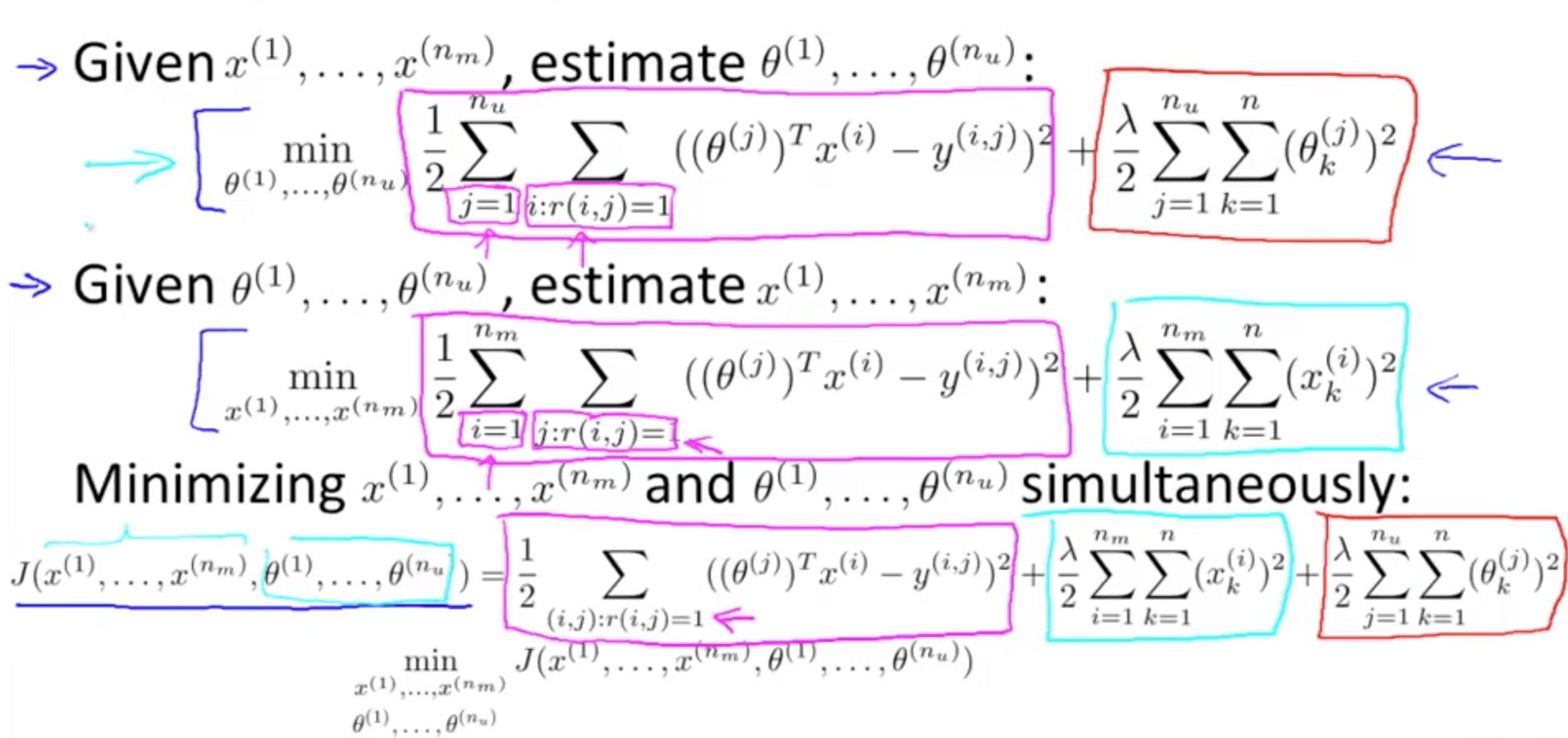
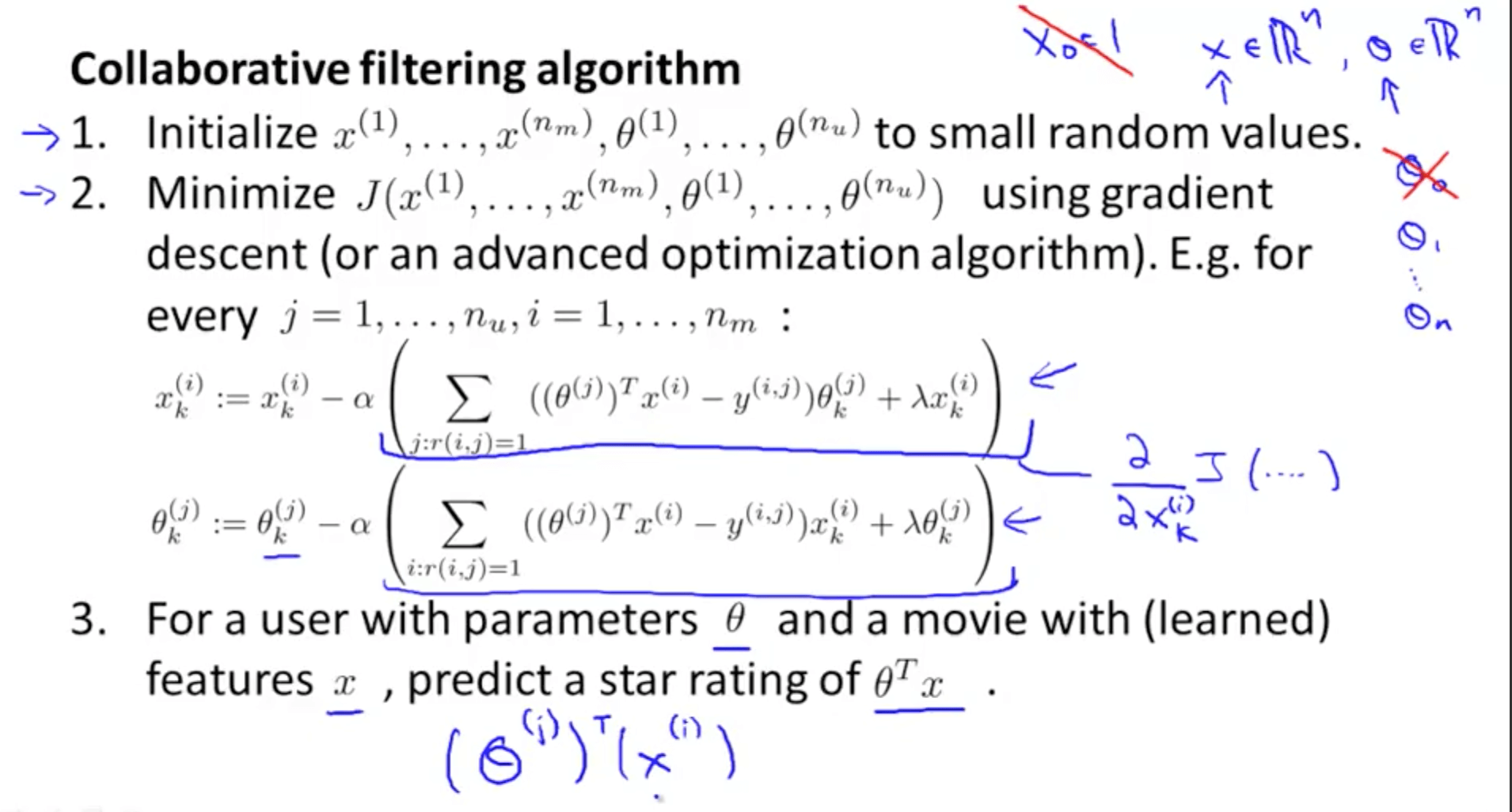
Here we will be learning about “Feature Learning”

Feature Learning: learning what features to use

Problem motivation It is inefficient and difficult to ask someone to watch each movie and inform us how romantic or action-packed the movie is 

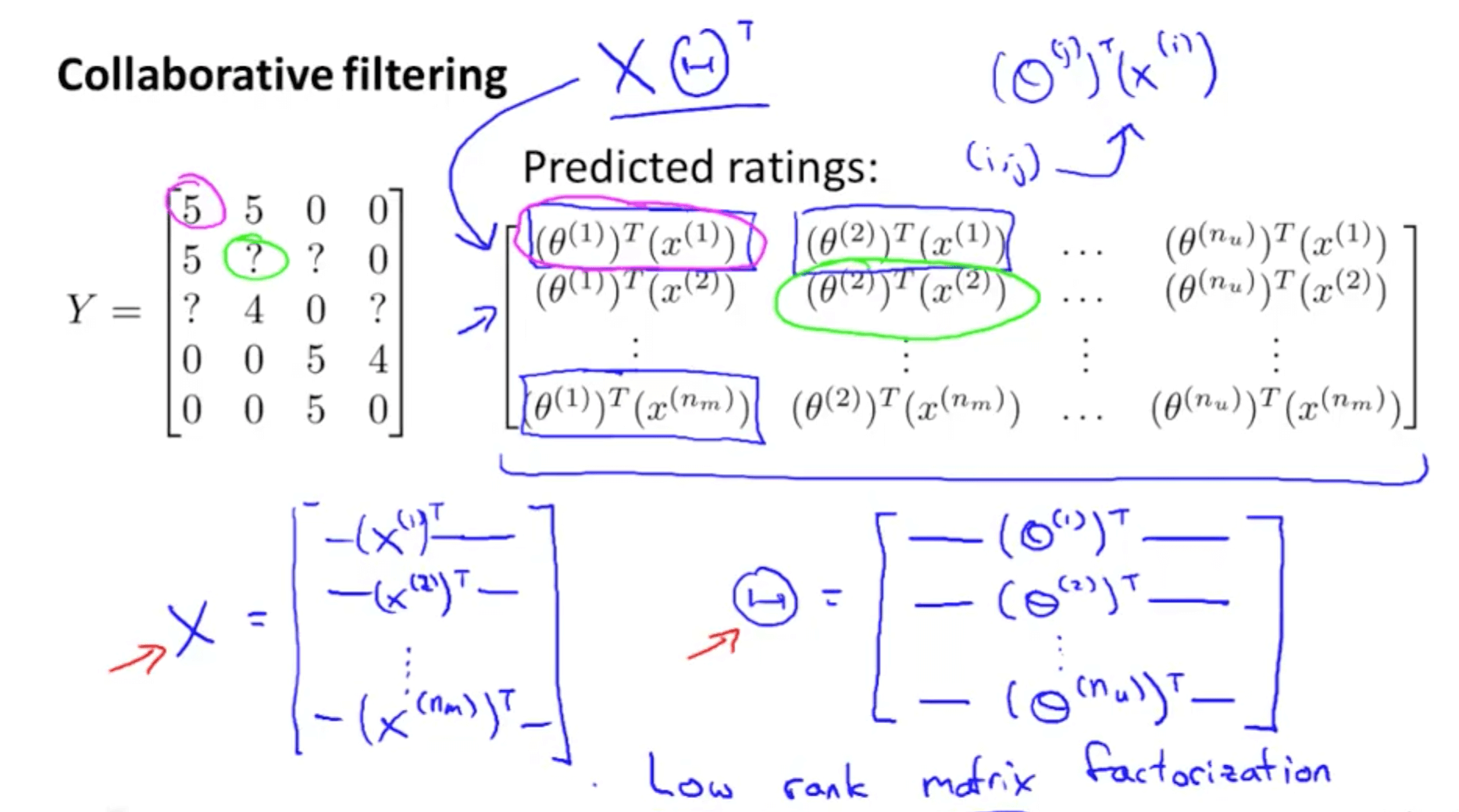
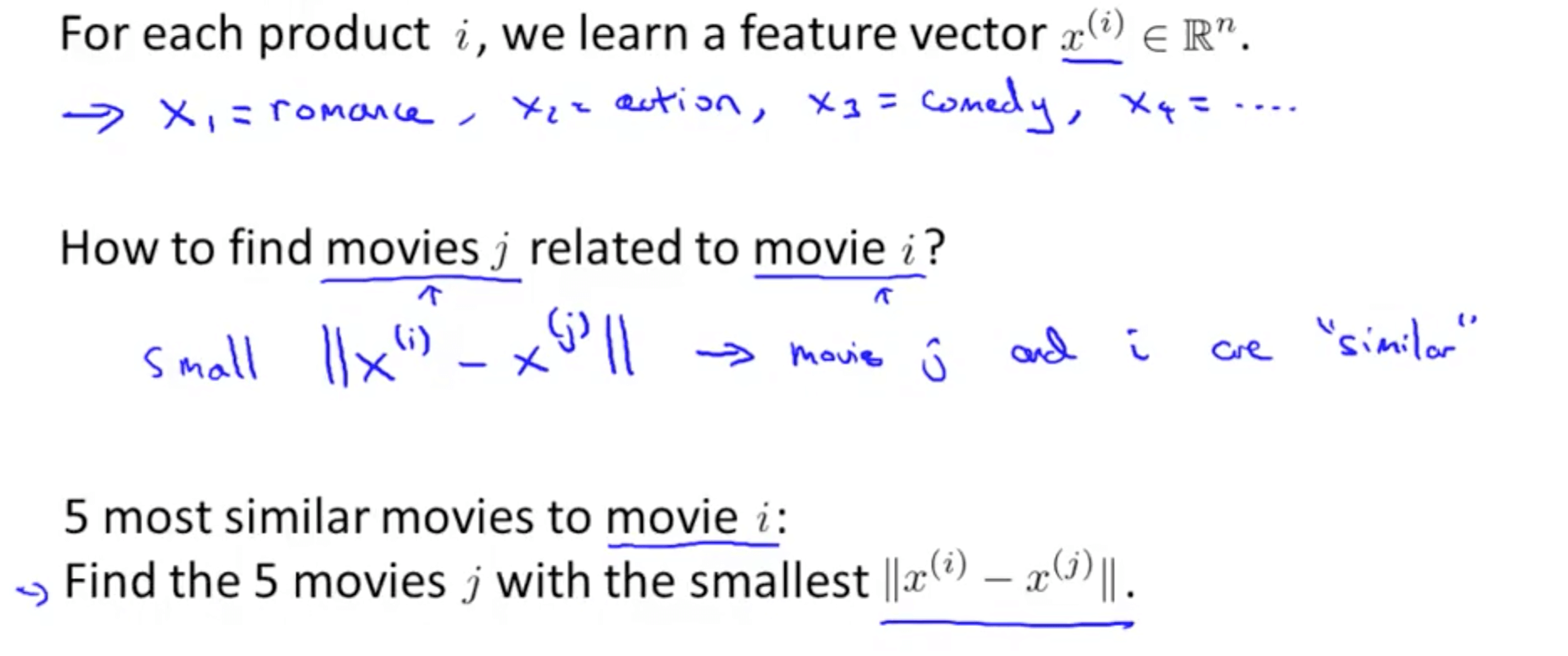
* + - Now we’ve no idea how each movie is romantic (x1) or action-packed (x2)
    - Let’s say that
      * Alice (θ1): likes romance
      * Bob (θ2): likes romance
      * Carol (θ3): likes action
      * Dave (θ4): likes action
    - We can discover x\_1 by making sure the following happens
      * θ1\_transpose \* x\_1 = 5
      * θ2\_transpose \* x\_1 = 5
      * θ3\_transpose \* x\_1 = 0
      * θ4\_transpose \* x\_1 = 0
* Optimization algorithm
  + This tries to choose features X\_i 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
  + Regularization term to prevent features from becoming too big 
  + We can guess θ, solve for x, then solve for θ and continue
    - There is a more efficient method to do this and we will be discussing this shortly 

**2b. Collaborative Filtering Algorithm**

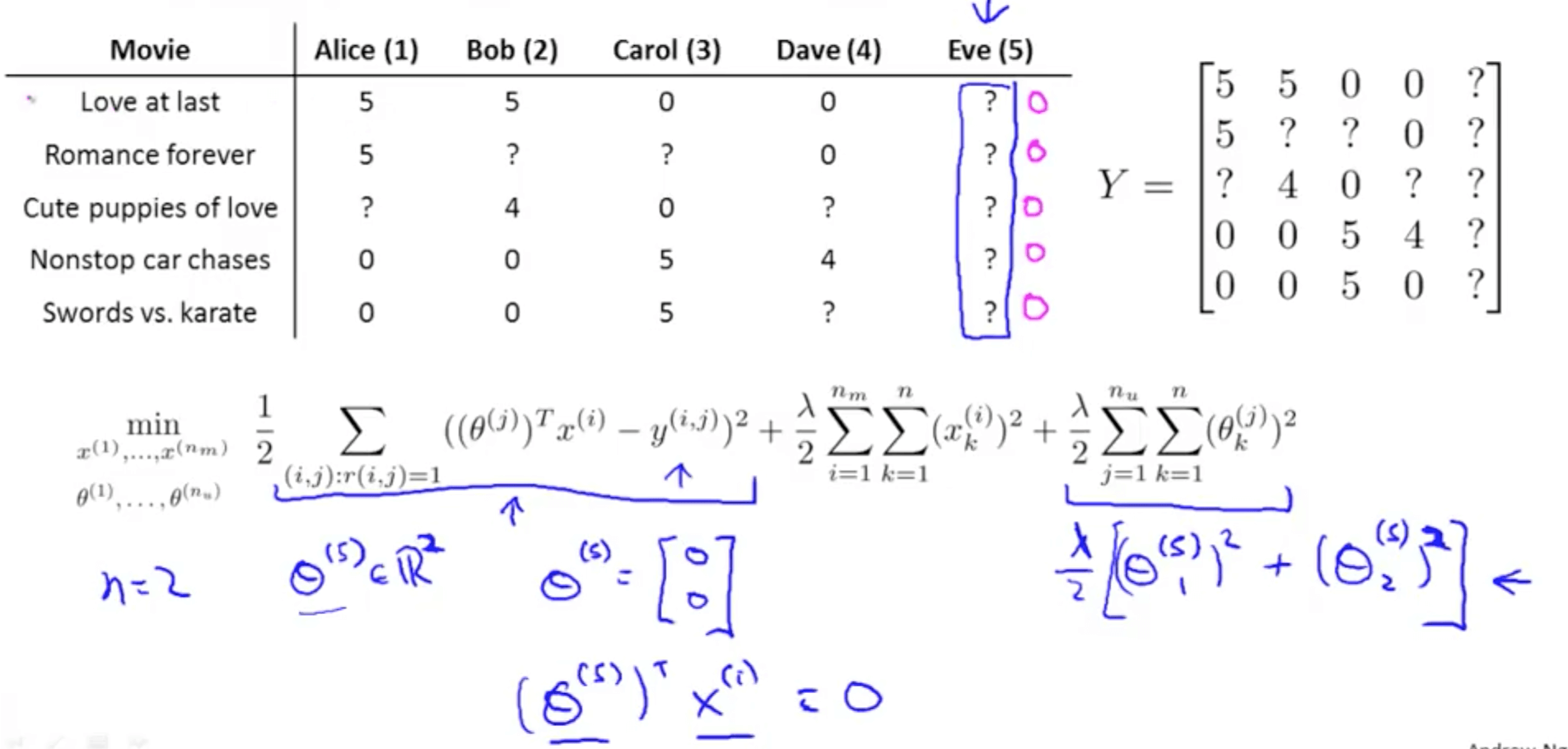
* Collaborative filtering optimization objective
  + We will take both of these optimization objectives and put them together
  + Now we get to minimize with respect to x and θ simultaneously 
* Collaborative filtering algorithm 

**3. Low Rank Matrix Factorization**

**3a. Vectorization: Low Rank Matrix Factorization**

* Low rank matrix factorization
  + Y matrix: all the predicted ratings
  + We conduct matrix factorization by decomposing the matrix into a product of matrices
    - X matrix: features of each movie stacked in rows
    - H matrix: parameters of each user stacked in rows
    - Product of X and H\_transpose equals Y matrix 
* Finding related movies 

**3b. Implementation Detail: Mean Normalization**

* Issue without mean normalization
  + And for even the Swords vs. Karate, someone rated it 5 stars
  + So some people do like some movies
  + It seems not useful to just predict that Eve is going to rate everything 0 stars
  + If we’re predicting that eve is going to rate everything 0 stars, we also don’t have any good way of recommending any movies to her, because you know all of these movies are getting exactly the same predicted rating for Eve so there’s no one movie with a higher predicted rating that we could recommend to her 
* We can conduct mean normalization to solve this issue
  + Our prediction would be the average of each movie 