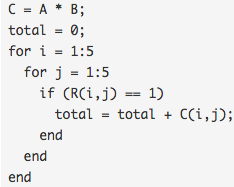
* Suppose you run a bookstore, + have ratings (1-5 stars) of books. Your collaborative filtering algorithm has learned a parameter vector θ(j) for user j, + a feature vector x(i) for each book. You would like to compute the training error/average squared error of your system's predictions on all ratings you’ve gotten from your users. Which of these are correct ways of doing so?
* For this problem, let m = total number of ratings gotten from users 
* In which of the following situations will a collaborative filtering system be the most appropriate learning algorithm (compared to linear or logistic regression)?
* **You run an online bookstore + collect ratings of many users. You want to use this to ID what books are "similar" to each other (i.e., if a user likes a certain book, what are other books that she might also like?)**
* **You own a clothing store that sells many styles + brands of jeans. You’ve collected reviews of the different styles + brands from frequent shoppers + want to use these reviews to offer those shoppers discounts on the jeans you think they are most likely to purchase**
* You run a movie empire + want to build a movie recommendation system based on collaborative filtering. There were 3 popular review sites (A, B, C) which users to go to rate movies, + you’ve just acquired all 3 companies that run these sites. You'd like to merge the 3 companies' datasets together to build a single/unified system. On site A, users rank a movie 1-5 stars. On site B, users rank on a scale of 1-10, + decimal values (e.g., 7.5) are allowed. On site C, ratings are from 1-100. You also have enough info to ID users/movies on 1 site w/ users/movies on a different site. Which of the following statements is true?
* **You can merge the 3 datasets into 1, but you should first normalize each dataset's ratings (say rescale each dataset's ratings to a 0-1 range).**
* **You can merge the 3 datasets into 1, but 1st normalize each separately by subtracting the mean + then dividing by (max - min) where max + min = (5-1) or (10-1) or (100-1) for the 3 sites respectively.**
* Which of the following are true of collaborative filtering systems?
* **Even if each user has rated only a small fraction of all products (so r(i,j)=0 for the vast majority of (i,j) pairs), you can still build a recommender system by using collaborative filtering.**
* **For collaborative filtering, it is possible to use one of the advanced optimization algorithms (L-BFGS/conjugate gradient/etc.) to solve for both the x(i)'s and θ(j)'s simultaneously.**
* If you have a dataset of users ratings' on some products, you can use these to predict one user's preferences on products he has not rated.
* When using gradient descent to train a collaborative filtering system, it is okay to initialize all the parameters (x(i) + Ө(j) to zero)
* To use collaborative filtering, you need to manually design a feature vector for every item (e.g., movie) in your dataset, that describes that item's most important properties.
* If you have a dataset of user ratings' on some products, you can use these to predict a user's preferences on products he has not rated.
* Recall that the cost function for the content-based recommendation system is



Suppose there is only 1 user + he has rated every movie in the training set. This implies that n(u) = 1 + r(i, j) = 1 for every (i, j). In this case, the cost function J is equivalent to the one used for regularized linear regression.

* You have 2 matrices A + B, where A is 5x3 + B is 3x5. Their product is C = AB, a 5x5 matrix. Furthermore, you have a 5x5 matrix r where every entry is 0 or 1. You want to find the sum of all elements C(i, j) for which the corresponding r(i, j) is 1, + ignore all elements C(i, j) where r(i, j) = 0. One way to do so is the following code:



* Which of the following pieces of Octave code will also correctly compute this total?
* **total = sum(sum((A \* B) .\* R))**
* **C = A \* B; total = sum(sum(C(R == 1)));**
* **C = (A \* B) .\* R; total = sum(C(:));**