**Abstract:**

As science and technology progresses every day we are subjected to new information. This new found results and information need its separate work and research, unfortunately human mind has not evolved to carter to this abundance of knowledge. In today’s world there are so many options and fields that a user/person is not even aware of his/her entire options. To overcome this barrier, we have come up with recommendation systems. Unfortunately, recommendation is not an easy thing, problems associated with it are; different people have different priorities, difference in opinions, difference in preferences. It would be convenient if we could just group them altogether and then rank the most trending/often-opted choice and recommend it. However, such a system has no chance in the progressive world of today! We have designed few recommender systems that to some extent overcome this naïve approach and offer more solid recommendations.

**Introduction**:

The Internet and World Wide Web have brought us into a world of endless possibilities: interactive Web sites to experience, music to listen to, conversations to participate in, and every conceivable consumer item to order. But this world also is one of endless choice: how can we select from a huge universe of items of widely varying quality? So, what can we do? When people have to make a choice without any personal knowledge of the alternatives, a natural course of action is to rely on the experience and opinions of others. We seek recommendations from people who are familiar with the choices we face, who have been helpful in the past, whose perspectives we value, or who are recognized experts. We might turn to friends or colleagues, the owner of a neighborhood bookstore, movie reviews in a newspaper or magazine, or Consumers Union product ratings. And we may find the social process of meeting and conversing with people who share our interests as important as the recommendations we receive.

Recommendation, therefore, is a communicative act. Recommendation is based on the preferences of the recommender (and perhaps of the seeker and other individuals). A preference is an individual mental state concerning a subset of items from the universe of alternatives. Individuals form preferences based on their experience with the relevant items, such as listening to music, watching movies, tasting food, etc. For example, I might prefer vanilla or strawberry ice cream (among ice cream flavors), Bach, Mozart, and Haydn (among classical composers), and Joel Coen and Kevin Smith (among contemporary film directors). Of course, preferences can be more complicated: I might prefer one item over another (The Simpsons over The X-Files) or even think in terms of some scoring system (“on a scale of 1 to 10, Bob Dylan’s Highway 61 Revisited is a 10”).

Thus, the need of a computerized system to provide recommendations. A recommendation system is a computer program that helps a user discover products and content by predicting the user’s rating of each item and showing them the items that they would rate highly. Recommendation systems have several different uses. The most common use for a recommendation system is ranking products by how much a user would like them. If a user is browsing or searching for products, we want to show them the products they would like most first in the list. Recommendation systems can also be used to find out how similar different products are to each other. If products are very similar to each other, they might appeal to the same users.

Product similarity is especially useful in cases where we don’t know much about a particular user yet. We can recommend similar products, even if the user hasn’t entered any of their own product reviews yet. We can also use recommendation systems to figure out if two different users are similar to each other. If two users have similar preferences for products, we can assume they have similar interests. For example, a social network can use this information to suggest the two users should become friends.

**Related Work:**

There has been much progress made in this field too, the initial naive approach of, Popularity Based System, just arranging the choices in their ascending order had its flaws. Though it was Scalable it wasn’t able to provide personalization to its users. It may be used to recommend webpages in our search but when it comes to user preferences this approach fails. An attempt to improve the Popularity Based System is Classification based model, it is similar to Popularity Based model with a slight modification. It groups the user on the basis of ethnicity, gender, religion, culture etc. and then calculate the numbers. This slight modification enables somewhat the feature of personalization but it is not enough. Especially when the world is so inter-connected with different people from different time zones, different backgrounds are inter mingling with one and another. Another problem that it faces is that this approach is not scalable. Let’s say we’re gathering data for a book store. In order to apply this approach, we would have numerous variables like gender, religion, age, ethnicity, income... this extra data is useless for the organization and would take up huge loads of space. Hence making this system a burden rather than a help.

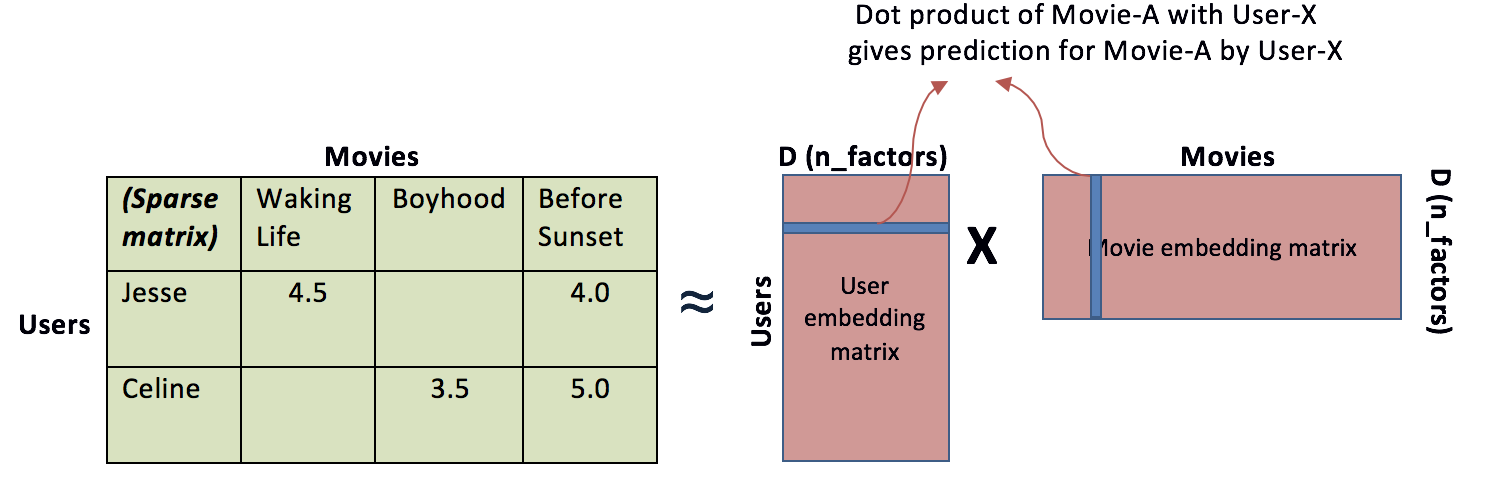
It is evident that just summing up the highest opted for cases aren’t always the options that the user is looking forward to. In order to improve the capabilities to personalize another approach is observed that is collaborative filtering. It works on the principle of “Birds of the feather flock together”, it takes into account similar users and then processes that information to come

up with a decent recommendation for its’ audience. Now the new problem is to process the similar users and how to process their information. For that I used two techniques “Nearest Neighbor” and “Matrices factorization”. The first technique takes in binary choices and computes the recommendation. However, if in case the choice is more diverse this fails. For example, it would be efficient if we were to see the users who have read a book or not and then recommend the confused user according to similar users, in this case the variable is yes or no it’s a binary choice hence it would work fine, but if we were asked to use the user ratings of the books then we would not be able to use this technique and would resort to “Matrices factorization”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Popularity  Based | Classification  Based | (Nearest Neighbor-based CF) | (Matrix Factorization based CF) |
| Personalized  Recommendations |  | Inserting image... |  |  |
| Uses Context  (Eg. time of day) |  | Inserting image... |  |  |
| User Features |  | Inserting image... |  |  |
| Item Features |  |  |  |  |
| Purchase History |  |  |  |  |
| Scalable |  |  |  |  |
| Can handle brand new Items? | Inserting image... |  | Inserting image... |  |

**Proposed Model:**

We proposed SVD recommender system, in order to carter the functionality to recommend on the basis of ratings rather then most viewed/searched. This not only gives us an additional functionality it also overcomes the barrier of scalability which was common in the previous recommender systems.



**It offers personalized recommendation.** Each user can tell his/her preferences to the machine in the form of ratings. The system then makes a user-item/item-user matrix and formulates similar pattern then it calculates the results and provide us with the recommendations. Use Matrix factorization approaches (Eg. Singular value Decomposition or SVD) to split the Rating Matrix into constituent User Matrix and Item Matrix with a minimum Sum of squared error (SSE)

**SVD:**

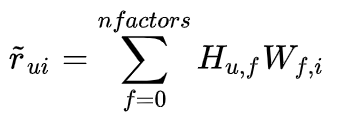
**A*nxp*= U*nxn* S*nxp* VT*pxp***

**Scalable.** The probability that two users who have bought 100 books each have a common book assuming that there is a catalog of million books is 0.01

Model learns to factorize rating matrix into user and movie representations, which allows the model to predict better personalized movie ratings for users

With matrix factorization, less-known movies can have rich latent representations as much as popular movies have, which improves recommender’s ability to recommend less-known movies

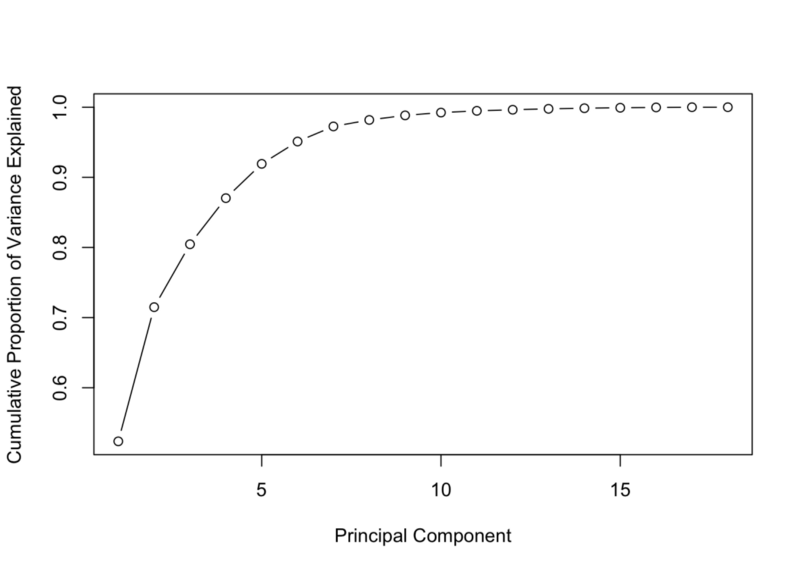
In the sparse user-item interaction matrix, the predicted rating user u will give item i is computed as:



where H is user matrix, W is item matrix

Rating of item i given by user u can be expressed as a dot product of the user latent vector and the item latent vector.

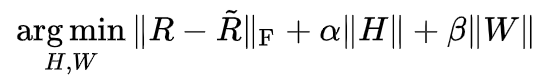
Notice in above formula, the number of latent factors can be tuned via cross-validation. Latent factors are the features in the lower dimension latent space projected from user-item interaction matrix. The idea behind matrix factorization is to use latent factors to represent user preferences or movie topics in a much lower dimension space. Matrix factorization is one of very effective dimension reduction techniques in machine learning.



Variance Explained By Components In PCA

Very much like the concept of components in PCA, the number of latent factors determines the amount of abstract information that we want to store in a lower dimension space. A matrix factorization with one latent factor is equivalent to a *most popular* or *top popular* recommender (e.g. recommends the items with the most interactions without any personalization). Increasing the number of latent factors will improve personalization, until the number of factors becomes too high, at which point the model starts to overfit. A common strategy to avoid overfitting is to add regularization terms to the objective function.

The objective of matrix factorization is to minimize the error between true rating and predicted rating:



where H is user matrix, W is item matrix

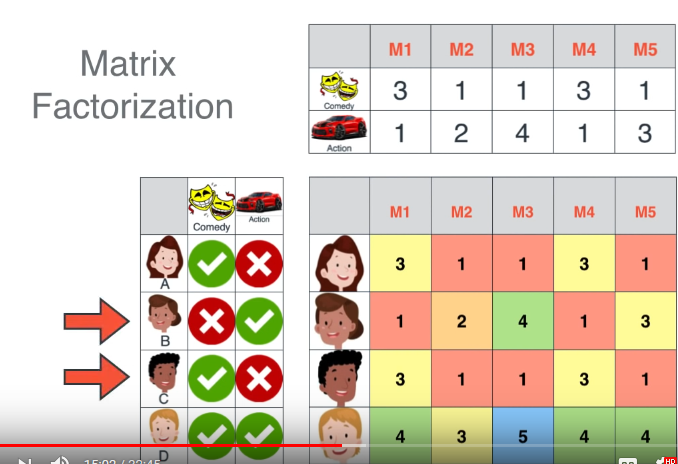
Once we have an objective function, we just need a training routine (eg, gradient descent) to complete the implementation of a matrix factorization algorithm. This implementation is actually called Funk SVD. It is named after Simon Funk, who he shared his findings with the research community during Netflix prize challenge in 2006.

**Limitation**: SVD system cannot carter to new items or new users as their ratings are not given. Also, it is not very scalable according to today's data. Researchers have demonstrated that the SVD-based algorithms can make the neighborhood formation process of CF systems highly scalable while producing better results in most of the cases [4, 9, 10]. Despite the good quality and excellent on-line performance SVD based algorithms suﬀer a serious drawback—the oﬀline SVD decomposition step is computationally very expensive. For an m×n user-item matrix, the SVD decomposition requires a run-time of O(m)3

**STEPS**:

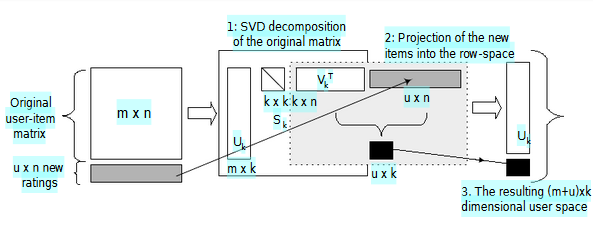
We create two methods for calculating SVD 1- computeSVD() and 2- computeEstimatedRatings().

In python we can easily compute the SVD using the sparsesvd library. Once we have calculated the SVD we can then calculate estimated ratings of the unknowns. Matrix factorization is simply a family of mathematical operations for matrices in linear algebra. To be specific, a matrix factorization is a factorization of a matrix into a product of matrices. In the case of collaborative filtering, matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. One matrix can be seen as the user matrix where rows represent users and columns are latent factors. The other matrix is the item matrix where rows are latent factors and columns represent items.



**Discussion:**

SVD has allowed us to save immense space, e.g., A matrix of 2000x1000 can now be saved in 100x1000 and 2000x100 I.e. from 2M it has shrunk down to only 300k entries. Also, it has allowed us to recommend on the basis of user preferences. It is personalized, it is scalable and it is more efficient then it’s older version. It’s generic, it can work on any dataset as long as it is factorizable.



Though it does solve the problem of scalability it brings in another challenge. The SVD recommendation system works in 2 steps the matrix formation and the recommendation. The recommendation is an easy simple task but the matrix formation on the other hand is tedious on one hand it takes O(1) to suggest a song/movie on the other it takes O(m)^3. SVD has a property that allows the model to be incrementally computed. This method was used by the LSI researchers [1, 2] to handle dynamic databases, where new terms and documents may arrive once the model is built. It was shown that a projection of additional terms and documents can potentially provide a fairly good approximation of the model. We extend this idea to build a system where we ﬁrst compute a suitably sized model and then use the projection method to build incrementally upon that. The resulting model is not a perfect SVD model as the space is not orthogonal, but the quality is expected to be good with potentially high-performance gain.

**Conclusion:**

We discussed how the world is in need of a system that would not just only provide recommendations but smart ones at that. Keeping in mind peoples’ preference their history and their likings. We managed to come up with a solution however it is far from acceptable as the world is progressing at an exponential rate. There is still a lot of room for improvement, we need to decrease our running time and further reduce the space taken to improve its scalability.