

Three Types of Recommendations

There are three methods that you have now implemented for making recommendations. These are the three most recognized methods in industry:

1. Knowledge Based Recommendations

Knowledge based recommendations frequently are implemented using filters, and are extremely common amongst luxury based goods. Filters that you might see when purchasing items like cars or homes are examples of knowledge based recommendations. In knowledge based recommendations, users provide information about the types of recommendations they would like back.

2. Collaborative Filtering Based Recommendations

Collaborative filtering uses the connections between users and items to make recommendations. Even the content based recommendation you just implemented used some collaborative filtering techniques, as you were not treating items and users independent from one another. In this lesson, you used neighborhood based collaborative filtering to find users who were alike and then recommend new movies based on these similar users.

Even in the content based recommendation, you were using collaborative filtering. You were finding items that were similar and making recommendations of new items based on the highest ratings of a user. Because you were still using the user ratings of an item, this was an example of a blend between content and collaborative filtering based techniques.

3. Content Based Recommendations

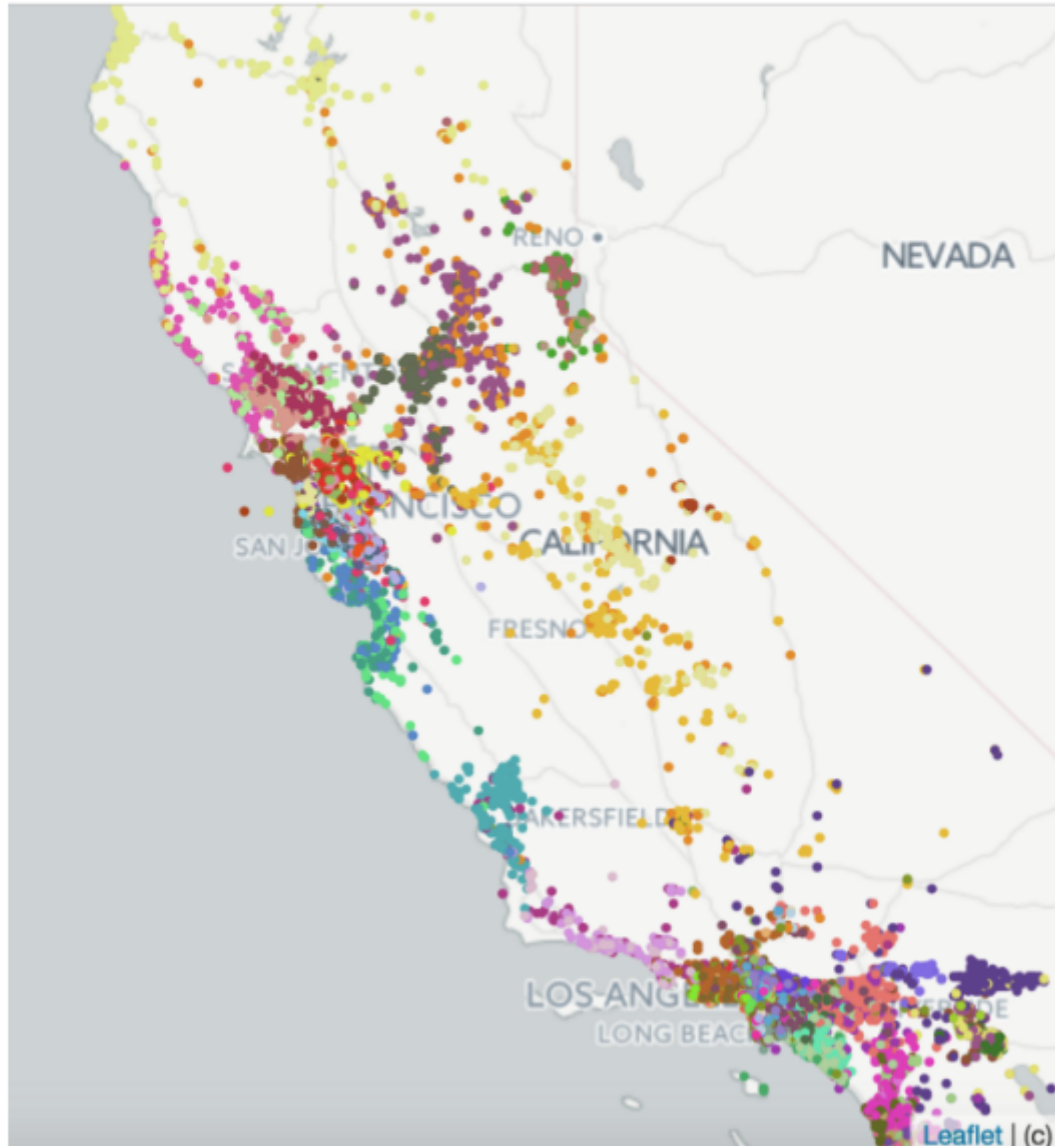
In the previous notebook, you created a matrix of similarities between items (movies) based only on the content related to those movies (year and genre). The similarity matrix that was used, was completely created using only the items (movies). There was no information used about the users implemented. For any movie, you would be able to determine the most related additional movies based only on the genre and the year of the movie. This is the premise of how a completely content based recommendation would be made.

Often blended techniques of all three types are used in practice to provide the the best recommendation for a particular circumstance.

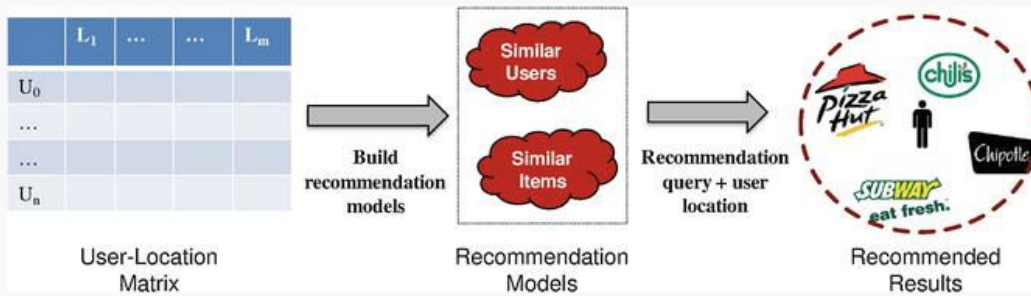
There are still more advanced techniques that are related to the methods that you learned about here, and they will most likely fall in one of the three buckets below.

- [AirBnB uses embeddings in their recommendation, which you can read more about here.](#)

What did Embeddings Learn?



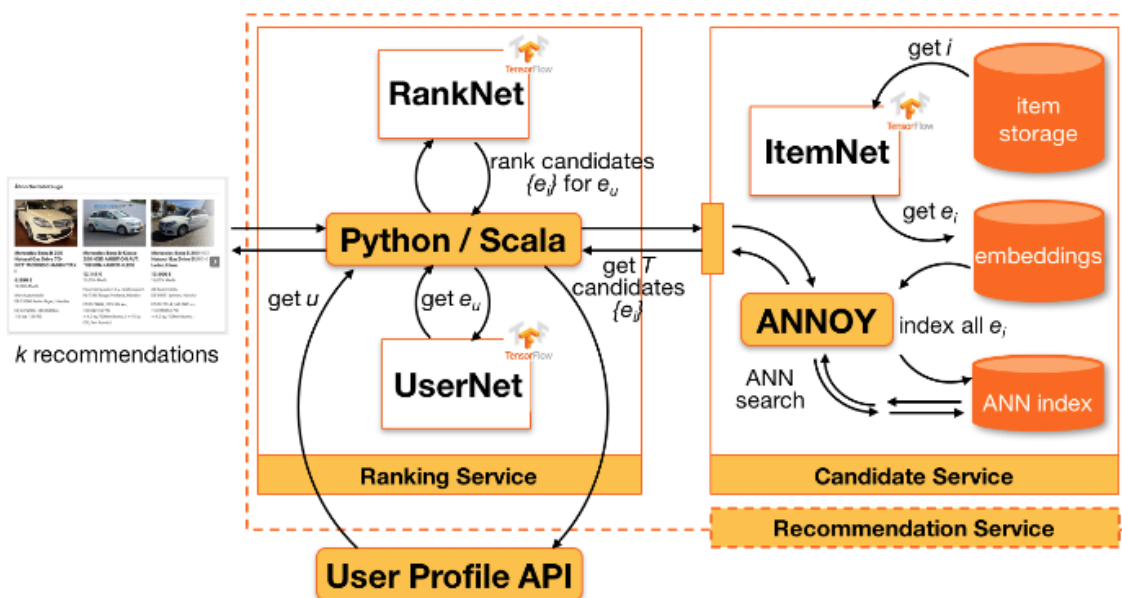
- As our smart phones become more addicting every day, it is easy to see why location based recommendations will be more and more popular. You can read more about these types of recommendations [here](#).



Location-Based Recommendation Systems, Fig. 1

Example of collaborative filtering-based location recommendation

- Many companies are also exploring deep learning use cases in recommendation systems.



recap

Types of Recommendations

In this lesson, you worked with the MovieTweatings data to apply each of the three methods of recommendations:

1. Knowledge Based Recommendations
2. Collaborative Filtering Based Recommendations
3. Content Based Recommendations

Within Collaborative Filtering, there are two main branches:

1. Model Based Collaborative Filtering
2. Neighborhood Based Collaborative Filtering

In this lesson, you implemented Neighborhood Based Collaborative Filtering. In the next lesson, you will implement Model Based Collaborative Filtering.

Similarity Metrics

In order to implement Neighborhood Based Collaborative Filtering, you were introduced to and applied a few techniques to assess how similar or distant two users were from one another:

1. Pearson's correlation coefficient
2. Spearman's correlation coefficient

3. Kendall's Tau
4. Euclidean Distance
5. Manhattan Distance

Types of Ratings

We took a quick look at different types of ratings:

1. Did the user interact with an item or not.
2. Did the user like an item or not.
3. More granular scales 1-7, 1-10, etc.

It is important to understand what the data might be used for, and what type of granularity might be important for a particular case. One of the main considerations is whether you want to have neutrality available, in which case an odd number of possible values in your scale will provide a value in the middle. Another common question is, how many levels do you really need to understand how much a user likes a particular product? Again, this is largely up to individual preference and specific use cases.

Business Cases For Recommendations

Finally, you looked at the four ideas needed for businesses to implement successful recommendations to drive revenue, which include:

1. Relevance
2. Novelty
3. Serendipity
4. Increased Diversity











At the end of this lesson, you will have gained a ton of skills to build upon or to start creating your own recommendations in practice.

Next Lesson

In the upcoming lesson, we will take a closer look at model based collaborative filtering, different methods for dealing with the cold start problem, and how to assess how well our model is performing. Then as a final touch, you will have the opportunity to deploy your recommendations to the web!

Latent Factors

When performing SVD, we create a matrix of users by items (or customers by movies in our specific example), with user ratings for each item scattered throughout the matrix. An example is shown in the image below.

		Items					
							
Users		3	4	8	9	2	2
		5	4	8	6	10	4
		5	4	9	8	9	6
		4	8	10	8	2	6

You can see that this matrix doesn't have any specific information about the users or items. Rather, it just holds the ratings that each user gave to each item. Using SVD on this matrix, we can find **latent features** related to the movies and customers. This is amazing because the dataset doesn't contain any information about the customers or movies!

SVD Closed Form Solution

What Is A Closed Form Solution?

A closed form solution is one where you can directly find the solution values (unlike iterative solutions, which are commonly used in practice). There isn't an iterative approach to solving a particular equation. One of the most popular examples of a closed form solution is the solution for multiple linear regression. That is if we want to find an estimate for

β

β in the following situation:

$$y = X\beta$$

$$y = X\beta$$

We can find it by computing the **Best Linear Unbiased Estimate (BLUE)**. It can be found **in closed form** using the equation:

$$\hat{\beta} = (X'X)^{-1}X'y$$

$$\beta$$

$$\wedge$$

$$=(X$$

$$,$$

$$X)$$

$$-$$

$$X$$

$$,$$

$$y$$

where **X** is a matrix of explanatory inputs and **y** is a response vector.

Another common example of a closed form solution is the quadratic equation. If we want to find **x** that solves:

$$ax^2 + bx + c = 0$$

$$ax$$

2

$$+bx+c=0$$

We can find these values using the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x =$$

$$2a$$

$$-b \pm$$

$$b$$

2

$$-4ac$$

Each of these is an example of a closed form solution, because in each case we have an equation that allows us to solve directly for our values of interest.

Closed Form Solutions for SVD

It turns out there is a closed form solution for Singular Value Decomposition that can be used to identify each of the matrices of interest (

U, Σ, V

U, Σ, V). The most straightforward explanation of this closed form solution can be found at [this MIT link](#).

As put in the paper -

"Calculating the SVD consists of finding the eigenvalues and eigenvectors of

AA'

AA

,

and

$A'A$

A

,

A . The eigenvectors of

$A'A$

A

,

A make up the columns of

V

V , the eigenvectors of

AA'

AA

,

make up the columns of

U

U . Also, the singular values in

Σ

Σ are square roots of eigenvalues from

AA'

AA

,

or

$$A'A$$

$$A$$

,

A . The singular values are the diagonal entries of the

σ

σ matrix and are arranged in descending order. The singular values are always real numbers. If the matrix

$$A$$

A is a real matrix, then

$$U$$

U and

$$V$$

V are also real."

Again, you can see a fully worked example of the closed form solution at the [MIT Link here](#).

A More Common Approach

The main issue with the closed form solution (especially for us) is that it doesn't actually work when we have missing data. Instead, Simon Funk (and then many followers) came up with other solutions for finding our matrices of interest in these cases using **gradient descent**.

So all of this is to say, people don't really use the closed form solution for SVD, and therefore, we aren't going to spend a lot of time on it either. The link above is all you need to know. Now, we are going to look at the main way that the matrices in SVD are estimated, as this is what is used for estimating values in FunkSVD.

Additional Resources

Below are some additional resources in case you are looking for others that go beyond what was shown in the simplified MIT paper.

- [Stanford Discussion on SVD](#)
- [Why are Singular Values Always Positive on StackExchange](#)
- [An additional resource for SVD in Python](#)
- [Using Missing Values to Improve Recommendations in SVD](#)

Funk SVD Practice









In the notebook on the next page, you will be writing the code to implement Funk SVD.





Before you dive in, let's do a practice run here.

First, consider we have a user-item matrix that looks like the matrix below, where we want to make an update of \mathbf{U} and \mathbf{V} matrices based on the 4 highlighted.

				
				9
	5		8	
		4		
			10	8

Also consider we have the following \mathbf{U} and \mathbf{V} matrices:

Latent Factors				Movies					
Users		Factor 1	Factor 2	Factor 3					
		0.8	1.2	-0.2	Factor 1	-1.8	1.2	-0.2	-1.2
		1.5	-0.8	1.4	Factor 2	0.5	-0.9	0.4	1.1
		0.4	1.1	-1.2	Factor 3	1.4	1.1	-1.2	-0.2
		1.5	-0.6	0.4					

Latent Factors		Factor 1	Factor 2	Factor 3
		0.8	1.2	-0.2
		1.5	-0.8	1.4
		0.4	1.1	-1.2
		1.5	-0.6	0.4

First, use the quiz below to identify what your current prediction would be for the 4 rating (highlighted in the first image above) based on the current values in the **U** and **V** matrices. Notice, this should be the prediction for the third user on the movie AI.

QUESTION 1 OF 2





The predicted value for the **third user** on the movie **AI** based on the current U and V matrices.

- 1.8
- -1.8
- 2.4
- 2.1

ENVIAR

Next, let's look at updating the value 0.4 as shown in the **U** matrix below.

Latent Factors

Users		Factor 1	Factor 2	Factor 3
		0.8	1.2	-0.2
		1.5	-0.8	1.4
		0.4	1.1	-1.2
		1.5	-0.6	0.4

The Cold Start Problem

El problema del arranque en frío es el problema de que los nuevos usuarios y los nuevos elementos de una plataforma no tienen ninguna clasificación. Debido a que estos usuarios y elementos no tienen ninguna clasificación, es imposible utilizar métodos de filtrado colaborativo para hacer recomendaciones.

Por lo tanto, los métodos que usó en la lección anterior como (recomendadores basados en rango y contenido) son la única forma de comenzar a hacer recomendaciones para estas personas.

Conclusion

In this lesson, you got your hands on some of the most important ideas associated with recommendation systems:

Recommender Validation

You looked at methods for validating your recommendations (when possible) using offline methods. In these cases, you could split your data into training and testing data. Frequently this split is based on time, where events earlier in time are in the training data, and events later in time are in a testing dataset.

We also quickly introduced the idea of being able to see how well your recommendation engine works by simply throwing it out into the world to directly see the impact.

Matrix Factorization with SVD

Next, we looked at matrix factorization as a technique for making recommendations. Traditional singular value decomposition a technique can be used when your matrices have no missing values. In this decomposition technique, a user-item (**A**) can be decomposed as follows:

$$A = U \Sigma V^T$$

$$A = U \Sigma V$$

$$T$$

Where

- U
- U gives information about how users are related to latent features.
- Σ
- Σ gives information about how much latent features matter towards recreating the user-item matrix.
- V^T
- V
- T
- gives information about how much each movie is related to latent features.

Since this traditional decomposition doesn't actually work when our matrices have missing values, we looked at another method for decomposing matrices.

FunkSVD

FunkSVD was a new method that you found to be useful for matrices with missing values. With this matrix factorization you decomposed a user-item (**A**) as follows:

$$A = UV^T$$

$$A = UV$$

$$T$$

Where

- U
- U gives information about how users are related to latent features.
- V^T
- V
- T
- gives information about how much each movie is related to latent features.

You found that you could iterate to find the latent features in each of these matrices using gradient descent. You wrote a function to implement gradient descent to find the values within these two matrices.

Using this method, you were able to make a prediction for any user-movie pair in your dataset. You also could use it to test how well your predictions worked on a train-test split of the data. However, this method fell short with new users or movies.

The Cold Start Problem

Collaborative filtering using FunkSVD still wasn't helpful for new users and new movies. In order to recommend these items, you implemented content based and ranked based recommendations along with your FunkSVD implementation.

Author's Note

There are so many ways to make recommendations, and this course provides you a very strong mind and skill set to tackle building your own recommendation systems in practice.

