

Recommender Systems

Let's learn something!





- Let's learn how to build a recommender system with Spark and Python!
- There is no Consulting Project or Documentation Example for this section, because the ease-of-use of Spark doesn't lend itself towards being tested on!



- In case you are further interested in Recommender Systems than what we cover in this course, I suggest you take a look at:
 - Recommender Systems by Jannach and Zanker





- Fully developed and deployed recommendation systems can be complex and resource intensive.
- Keep this in mind as we continue, you'll usually want someone with previous experience implementing a production recommendation system!





- Since full recommender systems require a heavy linear algebra background we will try to provide only a high level overview in this lecture.
- Check out the book mentioned in the beginning of the lecture for a deeper look into this topic





 The two most common types of recommender systems are
Content-Based and Collaborative Filtering (CF).



 Collaborative filtering produces recommendations based on the knowledge of users' attitude to items, that is it uses the "wisdom of the crowd" to recommend items.





 Content-based recommender systems focus on the attributes of the items and give you recommendations based on the similarity between them.





• In general, Collaborative filtering (CF) is more commonly used than content-based systems because it usually gives better results and is relatively easy to understand (from an overall implementation perspective).





 The algorithm has the ability to do feature learning on its own, which means that it can start to learn for itself what features to use.



- These techniques aim to fill in the missing entries of a user-item association matrix.
- spark.ml currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries.





- spark.ml uses the alternating least squares (ALS) algorithm to learn these latent factors.
- Your data needs to be in a specific format to work with Spark's ALS Recommendation Algorithm!



 ALS is basically a Matrix Factorization approach to implement a recommendation algorithm you decompose your large user/item matrix into lower dimensional user factors and item factors.



- To fully understand this model you need to have a strong background in Linear Algebra
- Check out the various resource links for more detail on ALS and how it works



- The intuitive understanding of a recommender system is the following:
- Imagine we have 3 customers: 1,2,3.
- We also have some movies: A,B,C
- Customers 1 and 2 really enjoy movies A and B and rate them five out of five stars!
- #1 and #2 dislike movie C, and give it a one star rating.





- Now we have a new customer #3, who reports a 5 star review for movie A.
- What new movie should we recommend, B or C?
- Well, based off collaborative filtering we recommend movie B, because Users #1 and #2 also enjoyed that (and movie A)



Recommender Systems

- A content based system wouldn't need to take Users into account.
- It would just group movies together based off features (length, genre, actors, etc...)
- Often real recommendation systems have combinations of methods.

- For this course, we will leave further review of ALS mathematics up to the student and those resource links.
- Let's get started!