DATA643 Discussion 2: Spark in Recommendation Engine

Yun Mai

June 22, 2017

Problem

After watching the following talk and summarize the most important or interesting points. The first half will cover some of the mathematical techniques covered in this unit's reading and the second half some of the data management challenges in an industrial-scale recommendation system.

https://www.youtube.com/watch?v=3LBgiFch4_g

Summary of the Interesting Points of the Talk

1. Methods Could be Used to Find Recommendations

The talk of Chris Johnson was about the algorithmic music recommendation at Spotify and the evolution of Hadoop. In the talk, the presenter review two collaborative filtering models, how they scale up the models to handle gigantic data, and how they solve the problem when they encounter the input/output bottleneck at Hadoop.

Spotify provides the recommendation for on-demand music streaming service such as personalized recommendation (Discover), artist radio, and related artists etc. First, the presenter introduced several ways to find recommendations and gave some examples. Two methods worked well for smaller catalog: manually curation adopted by Songa, or manually tagging catalog with attributes adopted by Pandora. However, they don't scale well. Analyzing music related text and audio content adopted by Echonest is the third way to find recommendations. Spotify has been using collaborative filtering and this is the main topic of this talk.

2. Collaborative Filtering and Matrix Facterization

Collaborative filtering is to look at and analyze what users are listening to, find the relationships, and then recommend music based on the relationships. The idea is to find the common songs and difference between users and recommend the different songs to similar users.

Collaborative filtering could be based on explicit matrix factorization or implicit matrix factorization.

Netflex recommendation model is an example based on explicit matrix factorization. Users rated some subset of the movies on 1 to 5 star scale. The model is built to predict how the users going to rate the movies that they haven't rated and recommended the movies with high predicted rating to a user. The idea is to approximate the dot product of the low-dimensional user vector and the low-dimensional item vector to the original user-item rating matrix. Mathematically, minimum RMSE (root mean square error) indicate the doc product closely approximate the original matrix.

For Spotify, the recommendation is built on implicit matrix factorization. Implicitly infer whether a user like a song or not based on what they're listening. The value will be 1 if the user stream to a track or 0 if the user did not so there will be a binary matrix. RMSE is used for evaluation but it will be weighted by the times the user listened to the track. Alternative least squares is used to solve the problem. When the song vector is fixed, the problem becomes weighted Ridge regression, the user vector could be solved. Then fix the user vector and solve for the song vector. After alternative back and forth between song vector and user vector until convergence, the least squares regression is solved.

3. Scale Up Challenges

Then the presenter talked about how they scale up with Hadoop and Spark. To do implicit matrix factorization with Hadoop, they first do so-called full-gratify to block off the rating matrix by a lot of blocks so that each block only represent a subset of users and songs. Then in the map phase, only items associated, if user vector is fix, will be taken followed by aggregating a lot of terms. In the reduced phase, all terms will be sum up and solve for the optimal user vectors.

As they distribute cache to send the vectors of the block, Hadoop incurs input/output problem because for an iterative algorithm need to continue reading and writing from disk. They use Spark to solve the problem. What Spark do is to load the rating matrix in the memory, cache it and joins thing where the rating is cached so avoid rereading the data from disk for every iteration.

At first, they tried take all item vectors transpose themselves and take the dot product (YtY). Then YtY were broadcasted so every partition would get a full set of item vectors. Then they grouped all the ratings by users and solved for the optimal user vectors. But it was not efficient as this shuffled data around each iteration and sent a full copy of all the item vectors to every single worker.

To get around the problem, they tried a different way to do the computation. The blocked matrixes were cached and partitioned to different workers, but never shuffled around. YtY would be broadcasted but each partition got a copy of the item vectors it needed. The intermediate terms were calculated and shuffled around to group by the user. Then the terms were aggregated for solving the optimal user vectors.

The third method, which is the so-called half-gratify method, was introduced to avoid the data shuffling in the second method. The difference was that block contained K users and all of the rating for those users. This way, they avoided shuffling the intermediate terms to group by the user and aggregating all those terms. And the presenter mentioned that alternative least square was available in a Spark package MLlib.

Comparing the alternative least squares running times, Spark under half-gratify method had the best performance and Spark under full-gratify method performed less well, and Hadoop took the longest time.

I like the second half part of this talk as it shows me how data scientists deal with the industrial-scale data management challenge in real life. Also, I get some concepts on how open-source cluster computing framework could be used in building recommendation models. This will be helpful in my future work when I need to use Hadoop or Spark.