Recommender systems with dimensionality reduction in Apache Spark - A trade-off between runtime and accuracy

Presenter: Mihai-Ionut Enache

29 November 2021

Apache Spark

- Open source analytics engine that can run on single machines or in a distributed environment
- Unified API → easy to scale from my laptop to a cluster
- Rich ML library: collaborative filtering, dimensionality reduction, clustering
- Has existed for a while + used in the industry → extensive documentation available
- Supports multiple languages: Python, Scala, Java etc.

Motivation

- Training a recommender system using collaborative filtering is expensive
- Latent factors one of the most used approaches
    - Tries to predict ratings using a combination of user and movies features
- Some movies are similar
    - Maybe too similar: e.g. redundant to have HP 1, 2 … 7 in the features set
- Question: can we trade accuracy for training time or even improve it through dimensionality reduction?

Dimensionality Reduction

- 2 common approaches:
  - Singular value decomposition
  - Principal component analysis
- Both available in Spark MLib
- But they are computationally expensive → questions:
  - If we reduce dimension how is the accuracy affected?
  - Can we reduce dimension and spend more resources on learning the recommender (e.g. increasing the number of iterations used to learn the parameters)?

Dimensionality Reduction (Continued)

- Another idea is to apply clustering for dimensionality reduction
- Few features per item → quick to compute distances → clustering can be efficient
  - Avoids expensive matrix multiplication
- Flexibility through selecting e.g. number of iterations in K-means
- Idea: use clusters of movies in the recommender system instead of individual movies
  - Explore trade-off between runtime and accuracy

Potential Extension: Bayesian Optimization

- Explore how Bayesian Optimization can help with recommender systems
- Many parameters to tune in a recommender system (number of latent factors, regularization parameter etc.)
- Questions:
  - Can BO help identify parameters for training the recommender system better than a random search?
  - How do we achieve this in Spark?

Plan

1. Install Apache Spark on my laptop
2. Identify a reasonable dataset
3. Ensure I can run baseline on the dataset
4. Experiment with the two standard methods of dimensionality reduction → identify which one provides a better trade-off
5. Reduce dimensionality through clustering → compare against method from step 4
6. If steps 4 and 5 done successfully:
    a. Explore how BO can be integrated in Spark in the context of recommender systems
    b. Try to find best parameters in steps 4 and 5 and in the baseline using BO
7. Write report explaining findings in steps 4-6, contrasting the benefits of each approach

Plan

1. ~~Install Apache Spark on my laptop~~ → Done
2. ~~Identify a reasonable dataset~~ → Done: 27 million ratings from 280k users for 58k movies
3. ~~Ensure I can run baseline on the dataset~~ → Done: baseline trained in < 2 min. 30 sec.
4. Experiment with the two standard methods of dimensionality reduction → identify which one provides a better trade-off
5. Reduce dimensionality through clustering → compare against method from step 4
6. If steps 4 and 5 done successfully:
    a. Explore how BO can be integrated in Spark in the context of recommender systems
    b. Try to find best parameters in steps 4 and 5 and in the baseline using BO
7. Write report explaining findings in steps 4-6, contrasting the benefits of each approach