DATA 612: Discussion 2

Derek G. Nokes

2019-06-20

Introduction

Watch the following talk and summarize what you found to be 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.

Summary

The most salient points in the presentation (for me) are summarized as follows:

- Explicit and implicit matrix factorization are both conceptually and mathematically very similar
- There are many ways to solve the explicit and implicit matrix factorization problems
- Matrix factorization scales well, but complexity increases significantly as the size of data moves beyond what can be held in memory simultaneously
- Implicit matrix factorization has enormous potential for developing lower dimensional representations of high dimensional data generated by modern web-based applications (e-commerce, entertainment platforms, etc.)
- It can take many iterations to find the right implementation approach to scale up recommendation systems operating on big data

Mathematical Techniques

The presentation covers both explicit and implicit matrix factorization techniques:

Explicit Matrix Factorization

We can approximate a very high dimensional user u by item i ratings matrix by the product of two lower dimensional latent factor matrices (X and Y). Essentially, this allows us to create a unique (latent factor) representation characterizing the preferences of each user.

We formulate the problem as follows:

$$\min_{x,y} \sum_{u,i} \left(r_{ui} - x_u^T y_i - \beta_u - \beta_i \right)^2 + \lambda \left(\sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right)$$

where

r_{ui} is the user u's rating for item i

x_u is the user u's latent factor vector

y_i is the item i's latent factor vector

β_u is the bias for user u

β_i is the bias for item i

λ is the regularization parameter

We minimize the root mean squared error (RMSE) to solve for the latent factors.

Implicit Matrix Factorization

Rather than using explicit ratings, with implicit matrix factorization, binary labels are used where 1 indicates that a user u has shown an implicit interest in an item i (e.g., the track was streamed) and 0 indicates that a user u has not shown any interest in an item i (e.g., the track was *not* streamed).

$$\min_{x,y} \sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i - \beta_u - \beta_i \right)^2 + \lambda \left(\sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right)$$

where

p_{ui} is 1 if the user u provides an implicit indication of interest in item i (e.g. the user streamed the track), otherwise it is 0

c_{ui} is $1 + \alpha r_{ui}$ and represents a confidence based on the number of indications (e.g., the number of times a user streamed a track)

x_u is the user u's latent factor vector

y_i is the item i's latent factor vector

β_u is the bias for user u

β_i is the bias for item i

λ is the regularization parameter

Minimization is done using a *weighted* root mean squared error (RMSE) where the weights are a function of the total streams.

The latent factors associated with the implicit matrix factorization can be solved using *Alternating Least Squares (ALS)*:

$$y_i = \left(X^T X + X^T \left(C^i - I \right) X + \lambda I \right)^{-1} X^T C^i p(i)$$

In this formulation, we alternate between solving for user latent factors and solving for item latent factors by fixing items or fixing users and using closed-form least squares regression until convergence is reached:

Fixing the latent item vector, we solve for the latent user vector x_u:

$$x_u = \left(Y^T C^u Y + \lambda I\right)^{-1} Y^T C^u p(u)$$

Then, fixing the latent user vector, we solve for the latent item vector x_i:

$$y_i = \left(X^T C^i Y + \lambda I\right)^{-1} X^T C^i p(i)$$

Interestingly, we only need the ratings a user actually streamed because the zeros are factored out in the $X^T X$

Once we have the latent factor vectors, user u's predicted implicit rating $\hat{r}_{ui}$ of item i is the dot project of two latent factor vectors, x_u and y_i:

$$\hat{r}_{ui} = x_u^T \, y_i = \sum_k x_{uk} y_{ik}$$

Data management Challenges in an Industrial-Scale Recommendation System

The presentation covers two main approaches to scale the recommendation system:

Hadoop Implementation

The production system implemented implicit matrix factorization with Hadoop. This implementation worked well, but there was a significant IO bottleneck.

Spark Implementation

The Spark implementation was intended to improve the system by removing this bottleneck. The first attempt to improve the system using Spark did not cache ratings data, did a lot of unnecessary data shuffling for each iteration, and sent an unnecessary full copy of the user/item vectors to all workers. The second attempt cached the ratings data and did not do any data shuffling, and required a lot less local memory than the first attempt, but had more IO overhead.

Software

This discussion post was created using base R [R Core Team, 2018] and the R markdown [Allaire et al., 2018], tint [Eddelbuettel, 2018], kableExtra [Zhu, 2018], and tidyverse [Wickham, 2017] libraries.

Python Implementation Links

The following URLs provide Python implementations for some of the methods used at Spotify:

https://github.com/MrChrisJohnson/implicit-mf

https://github.com/MrChrisJohnson/logistic-mf

https://github.com/MrChrisJohnson/deep-mf

https://github.com/MrChrisJohnson/annoy

Presentation Links

The following URLs provide fascinating details about features of the Spotify recommender system:

[2014-01-13] https://www.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify

[2014-05-09] https://www.slideshare.net/MrChrisJohnson/collaborative-filtering-with-spark

[2014-06-27] https://www.slideshare.net/MrChrisJohnson/music-recommendations-at-scale-with-spark

[2015-01-06] https://www.slideshare.net/MrChrisJohnson/scala-data-pipelines-for-music-recommendations

[2015-09-25] https://www.slideshare.net/MrChrisJohnson/interactive-recommender-systems-with-netflix-and-spotify

[2015-09-25] https://www.slideshare.net/erikbern/approximate-nearest-neighbor-methods-and-vector-models-ny

[2015-11-16] https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly

Key Papers

The following URLs provide details of matrix factorization on implicit data:

http://yifanhu.net/PUB/cf.pdf

http://stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf

https://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf

References

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, and Winston Chang. *rmarkdown: Dynamic Documents for R*, 2018. URL https://CRAN.R-project.org/package=rmarkdown. R package version 1.10.

Dirk Eddelbuettel. *tint: 'tint' is not 'Tufte'*, 2018. URL https://CRAN.R-project.org/package=tint. R package version 0.1.0.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL https://www.R-project.org/.

Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.2.1.

Hao Zhu. *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*, 2018. http://haozhu233.github.io/kableExtra/, https://github.com/haozhu233/kableExtra.