Weighted Alternating Least Squares (WALS) for Movie Recommendations

Drew Hodun

SCPD

Introduction

Collaborative Filtering is a common and powerful way for building feedback-based recommender systems. Low-rank matrix factorization is a common way of solving this problem. There are various methods of computing the low-rank representations, including ‘Alternating Least Squares’ (ALS) and the weighted version (WALS).

Purpose: Explore and compare the WALS (Weighted Alternating Least Squares) approach to collaborative filtering, in particular for explicit-preference datasets as opposed to implicit-preference datasets where WALS is more common. We’ll use the MovieLens Movie recommendation dataset.

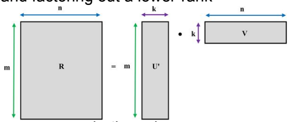
Theory

$$L = \sum_{m,n}(R_{mn} - U_m^T \cdot V_n)^2 + \lambda \sum_m ||U_m||^2 + \lambda \sum_n ||V_n||^2$$

Low-Rank Matrix Factorization

-Collaborative filtering through low-rank matrix factorization is a way of taking a sparse matrix of users and ratings, assuming a certain number of latent factors **(k)**, and factoring out a lower-rank representation of all the users and items.

-Can be roughly interpreted as ‘genres’ in Movielens dataset.



Alternating Least Squares (ALS)

Optimization:

1. When either user-factors or item-factors is held constant, Loss becomes quadratic and we can then optimize, alternating rows and columns
2. Set row constant
3. Set derivative to 0 and solve
4. Repeat for constant column

$$\frac{\partial L}{\partial U_m} = \sum_n (R_{mn} - U_m^T \cdot V_n)V_n^T + 2\lambda U_m^T$$

$$0 = -(R_m - U_m^T V^T)Y + \lambda U_m^T$$

$$U_m^T(V^T V + \lambda I) = R_m Y$$

$$U_m^T = R_m Y (Y^T Y + \lambda I)^{-1}$$

Now to approximate a rating, a fairly simple low-rank calculation:

$$R_{mn} = U_m^T \cdot V_n$$

Weighted Alternating Least Squares (WALS)

Use a weight vector which can be linearly or exponentially scaled to normalize row and/or column frequencies. In the case of explicit dataset like MovieLens, we will linearly weight columns (movies) to normalize signal, helping to boost movies that are less reviewed.

$$L^w = W \circ \sum_{m,n}(R_{mn} - U_m^T \cdot V_n)^2$$

$w_{mn} = \omega_0 + f(c_m)$ unobserved weight + function of observed weight

$c_m = \sum_{m,n} if\, R_{mn} > 0$ sum of number of non-zero entries for each column (reviews per movie)

$f(c_m) = \frac{\omega_k}{c_m}$ linearly (explicit) scaling - scale down reviews of often-reviewed movies

$f = (\frac{1}{c_m})^e$ exponential(implicit) scaling

Methods

Data Acquisition:

- 100k Rating Movielens Dataset
- 1,000 users, 1,700 movies
- Very clean dataset (less noisy)

Pre-Processing:

- Analyzed for Rating / User / Movie Distributions
- Removed from test set any ratings for movies / users not present in training set

Results

Figure 1. Train/Test Error for WALS vs ALS

- WALS results in RMSE of 0.94073 vs 0.97062 for ALS (3.1% improvement)
- Used Google’s Vizier-Backed Cloud Hyper-Parameter tuning to aggressively tune WALS Hyper-params to make it more accurate than more-obvious ALS
- Found interesting condition where either Col Factor Weight and Regularization would always reach maximum edge of hyper-parameter space where unobserved weight reached minimum edge of space, Test RMSE remained small and factors produced were reasonable
- Overall improvement was good, though would hope for larger boost
- Further examination of 100k rating dataset reveals much smaller dispersion (variance to mean ratio) between review per movie than the 20M rating dataset
- Would take a much larger amount of compute to complete hyper-param search on 20M dataset

Figure 2. Reviews per Movie - Distributions on 100k and 20M datasets

stddev	mean	min	percentile1	percentile25	median	percentile75	percentile90	percentile99	max
80.384	59.453	1	1	6	27	80	169	378	583
3,085.818	747.841	1	1	3	18	205	1,306	14,396	67,310

-100k, Dispersion index = 0.1k
-20M, Dispersion index = 12.7k

Discussion

- WALS is used to linearly weight down movies more commonly reviewed
- Decreased loss by 3.1% based on standard ALS
- Could Potentially see greater impact with less normalized / clean datasets
- WALS provides a lot of flexibility to model user preference / confidence and clean dataset, i.e. you could weigh users based on their reliability, if two people are using same account, etc. and still retain ease-of-use of Collaborative Filtering
- Next Steps: try WALS on noisier dataset and try weighing users, not just movies
- Next Steps:try exponential WALS on implicit dataset (click stream), as WALS is more commonly applied on these problems