Weighted Alternating Least-Squares for Low-Rank Completion

CME 258

February 11, 2018

Suppose that you want to recommend movies/tv shows/music based on users' viewing habits.

You have $A \in \mathbb{R}^{m \times n}$ where

- each row i corresponds to a user,
- each column j corresponds to an item (e.g. movie),

so that

$$A_{ij} = \begin{cases} \text{user } i\text{'s rating of movie } j & \text{if } i \text{ watched } j \\ \text{unknown} & \text{otherwise.} \end{cases}$$

Want to "complete" the matrix by determining the unknown entries (predict a user's ratings for movies they have not yet watched).

Two possible goals:

- Given a user's viewing history, suggest new movies to watch.
- Given a movie, determine similar movies (nearest neighbours).

Assume $A \approx UV^T$ for $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$ with $r \ll m, n$.

Arguably reasonable assumption:

- Can think of movies as being a "linear combination" of much fewer genres.
- Can think of users as being a "linear combination" of people who like certain genres.

Let $U = \begin{bmatrix} u_1^T \\ \vdots \\ u_m^T \end{bmatrix}$, $V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$, and $W_{ij} = \begin{cases} 1 & (i,j) \text{ observed} \\ 0 & \text{otherwise.} \end{cases}$

Then we solve

$$\min_{U,V} \frac{1}{2} \sum_{(i,j) \text{ observed}} (A_{i,j} - u_i^T v_j)^2 + \lambda \left(\sum_{i=1}^{m} \|u_i\|_2^2 + \sum_{j=1}^{n} \|v_j\|_2^2 \right)$$

$$= \min_{U,V} \frac{1}{2} \|W \circ (A - UV^T)\|_F^2 + \lambda \left(\|U\|_F^2 + \|V\|_F^2 \right),$$

where $(A \circ B)_{ij} = A_{ij} B_{ij}$.

We'll solve this problem via alternating minimization:

- Fix U and solve for V, then
- Fix V and solve for U.

When one factor is fixed, this becomes regular linear least-squares.

Notice that this is equivalent to solving

$$\min_{v_1,\ldots,v_n} \sum_{j=1}^{n} \frac{1}{2} \|W_{:,j} \circ (A_{:,j} - U v_j)\|_2^2 + \lambda \|v_j\|_2^2,$$

so we can solve for every row of V independently.

(Need to slice columns of A repeatedly...)

To get v_j, need to solve linear system

$$\left(U^T \mathrm{diag}(W_{:,j}) U + \lambda I\right) v_j = U^T A_{:,j}.$$

How can we perform this operation quickly?

Notice that

$$U^T \mathrm{diag}(W_{:,j}) U = \sum_{i:W_{i,j}=1} u_i u_i^T,$$

(syrk!).

Similarly,

$$U^T A_{:,j} = \sum_{i:W_{i,j}=1} u_i A_{i,j}.$$

Homework 2

We have provided you with a basic implementation of a weighted-alternating-least-squares solver in python. Your homework is to improve its performance!

Some ideas:

- Use cProfile to check which operations are taking up the most time.
- Use the correct sparse-matrix format depending on whether U or V is being updated.
- Find the best way to solve the linear system. Which factorization would you use? How would you call it?
- Avoid “todense()” calls.

This assignment is intentionally open-ended. Do your best to improve the performance as much as possible.

We've also provided you with a method to check the quality of your factorization: you can input a movie from the MovieLens dataset, and check which movies are considered similar according to the factorization.