Matrix Factorization for Collaborative Filtering Mathematical Details

Brendan Kolisnik

January 12, 2023

Abstract

This document contains some of the mathematical details behind popular optimization methods for Matrix Factorization in the collaborative filtering setting.

1 Introduction

This document assumes some familiarity with recommender systems and collaborative filtering. The specific model we will be focusing on is the matrix factorization (mf) model. This model was a popular choice for recommendation systems and a variant of mf was used to win the Netflix Movie Recommendation Challenge in 2009. Currently, there are many deep learning approaches that often outperform mf on recommendation challenges. Neural approaches can be seen as a more sophisticated version of this model. Readers should be familiar with the trace of a matrix (the sum of the diagonal elements for a square matrix) and the Frobenius Norm. The Frobenius Norm of a matrix is

$$\|X\|_F^2 := \text{Tr}(X^T X) \tag{1}$$

The mf model update equations are also similar to multivariate regression where there are k targets that must be predicted.

$$\underset{n\times k}{\text{Y}} = \underset{n\times p}{X} \times \underset{p\times k}{\beta} + \underset{n\times k}{\varepsilon} \tag{2}$$

There the loss function we aim to optimize is mean squared error. However, the argument that optimizes the mean squared error are the same as that which optimizes the sum of squared error. We can write the cost function as shown below.

$$\begin{aligned}\mathcal{L}(W) &= \|XW - Y\|_F^2 \\ &= \text{Tr}((XW - Y)^T (XW - Y))\end{aligned} \tag{3}$$

Let $E = XW - Y$ and let $\varepsilon_j \in \mathbb{R}^n$ be the column vector for target variable j error. Then the cost function can be seen as the sum of square errors summed over each target variable.

$$\begin{aligned}\mathcal{L}(W) &= \text{Tr}(E^T E) \\ &= \sum_{j=1}^{k} \varepsilon_j^T \varepsilon_j \quad = \sum_{j=1}^{k}\sum_{i=1}^{n} \varepsilon_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{k} \varepsilon_{ij}^2\end{aligned} \tag{4}$$

This cost formulation is very similar to what we will optimize for mf. To find a closed form solution for the argument that minimizes this cost function in terms of W is possible but it will require us to take derivatives of scalar valued functions with respect to matrices. That is exactly what we will do in the next section.

2 Matrix Factorization

2.1 Setup

The mf model attempts to find latent vectors for each user and item where the number of elements in these vectors is a hyper-parameter to be decided by the data scientist. Given a ratings matrix $R \in \mathbb{R}^{n \times p}$ for a dataset with n users and p items we will find a matrix $U \in \mathbb{R}^{k \times n}$ for the users and a matrix $V \in \mathbb{R}^{k \times p}$ for the items. Our task is to optimize the following:

$$\min_{U,V} \|R - U^T V\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2) \tag{5}$$

We will call this expression we are minimizing our cost function $\mathcal{L}$

2.2 SGD

It is possible that not every element of R is filled. Often this matrix is mostly sparse with NaN values since no rating has been observed for many user-item pairs. In this case we can optimize over only the ratings we have seen using SGD. Here κ is the set of user-item pairs that have been observed, $u_i \in \mathbb{R}^k$ is user latent vector and $v_j \in \mathbb{R}^k$ is the item latent vector.

$$\min_{U,V} \sum_{(i,j) \in \kappa} (r_{ij} - u_i^T v_j)^2 + \lambda(\|u_i\|^2 + \|v_j\|^2) \tag{6}$$

Additionally, let $e_i j = r_{ij} - u_i^T v_j$, let $s =$ and let $L = (r_{ij} - u_i^T v_j)^2 + \lambda(\|u_i\|^2 + \|v_j\|^2)$. Then we can write the partial derivatives as the following:

$$\begin{aligned} \frac{\partial L}{\partial u_i} &= -2e_{ij} v_j + 2\lambda u_i \\ \frac{\partial L}{\partial v_j} &= -2e_{ij} u_i + 2\lambda v_j \end{aligned} \tag{7}$$

Finally, we can write the Stochastic Gradient Descent updates as:

$$\begin{aligned} u_i &\leftarrow u_i - \alpha \frac{\partial L}{\partial u_i} \\ v_j &\leftarrow v_j - \alpha \frac{\partial L}{\partial v_j} \end{aligned} \tag{8}$$

2.3 Alternating Least Squares

What if the ratings matrix is complete? In that case we can use an optimization method that converges much faster than SGD. Let $\hat{R} = U^T V$ be our predicted values for each rating in the matrix. Let $\mathcal{L} = \|R - U^T V\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$. Recalling our trace property lets define a new equation to be optimized.

$$L = \operatorname{Tr}((R - \hat{R})^T (R - \hat{R})) + \lambda \operatorname{Tr}(U^T U) + \lambda \operatorname{Tr}(V^T V) \tag{9}$$

We will solve this by fixing one of U or V and finding the closed form solution for the other. We will use some new notation and state several known properties of the trace and differentials.

$$\begin{aligned} A : B &= \operatorname{Tr}(A^T B) \\ \operatorname{Tr}(A^T B) &= \operatorname{Tr}(B^T A) = \operatorname{Tr}(A B^T) = Tr(B A^T) \\ \operatorname{Tr}(A + B) &= \operatorname{Tr}(A) + \operatorname{Tr}(B) \\ d \operatorname{Tr}(X) &= \operatorname{Tr}(dX) \\ d(X + Y) &= dX + dY \\ d(XY) &= (dX)Y + X(dY) \\ d(A : B) &= dA : B + A : dB \\ A : B dF &= B^T A : dF \\ A : dF B &= A B^T : dF \end{aligned} \tag{10}$$

There are also certain identification rules that allow one to convert from the differential form back into the derivative form. This is based on the definition of differentials.

$$dy = \mathrm{Tr}(AdX) \implies \frac{\partial y}{\partial X} = A \tag{11}$$

We may then begin to take the differential.

$$\begin{aligned} d\mathcal{L} &= d\,\mathrm{Tr}((R-\hat{R})^T(R-\hat{R})) + d\lambda\,\mathrm{Tr}(U^TU) + d\lambda\,\mathrm{Tr}(V^TV) \\ d\mathcal{L} &= d(R-\hat{R}) : (R-\hat{R}) + (R-\hat{R}) : d(R-\hat{R}) + \lambda dU : U + \lambda U : dU + \lambda dV : V + \lambda V : dV \end{aligned} \tag{12}$$

Holding V constant since we are optimizing for U leads to the following.

$$\begin{aligned} d\mathcal{L} &= -d\hat{R} : (R-\hat{R}) + (R-\hat{R}) : d\hat{R} + \lambda dU : U + \lambda U : dU \\ d\mathcal{L} &= -2(R-U^TV) : (dU^T)V + 2\lambda U^T : dU^T \\ d\mathcal{L} &= -2(R-U^TV)V^T : dU^T + 2\lambda U^T : dU^T \\ d\mathcal{L} &= (-2(R-U^TV)V^T + 2\lambda U^T) : dU^T \\ \frac{\partial \mathcal{L}}{\partial U^T} &= -2(R-U^TV)V^T + 2\lambda U^T \end{aligned} \tag{13}$$

We may now set the partial derivative equal to the 0 matrix and solve for the optimal U^T.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial U^T} &= \mathbf{0} \\ 2\lambda U^T &= 2(R-U^TV)V^T \\ \lambda U^T &= RV^T - U^TVV^T \\ \lambda U^T + U^TVV^T &= RV^T \\ U^T(\lambda\mathbf{I} + VV^T) &= RV^T \\ U^T &= RV^T(\lambda\mathbf{I} + VV^T)^{-1} \\ U &= (\lambda\mathbf{I} + VV^T)^{-1}VR^T \end{aligned} \tag{14}$$

After updating U we will hold U fixed and update V.

$$\begin{aligned} d\mathcal{L} &= -d\hat{R} : (R-\hat{R}) + (R-\hat{R}) : d\hat{R} + \lambda dV : V + \lambda V : dV \\ d\mathcal{L} &= -2(R-U^TV) : U^T(dV) + 2\lambda V : dV \\ d\mathcal{L} &= (-2U(R-U^TV) + 2\lambda V) : dV \\ \frac{\partial \mathcal{L}}{\partial V} &= -2U(R-U^TV) + 2\lambda V \end{aligned} \tag{15}$$

We may now set the partial derivative equal to the 0 matrix and solve for the optimal V.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial V} &= \mathbf{0} \\ 2\lambda V &= 2U(R-U^TV) \\ \lambda V &= UR - UU^TV \\ \lambda V + UU^TV &= UR \\ (\lambda\mathbf{I} + UU^T)V &= UR \\ V &= (\lambda\mathbf{I} + UU^T)^{-1}UR \end{aligned} \tag{16}$$

We can keep alternating the which matrix is fixed and which is free based on the number of iterations or until a certain cost function threshold has been reached.