Machine Learning, Spring 2019
EN 2105
MWF 9-9:50am

Chao Lan
EN 4084A
clan@uwyo.edu

Matrix Completion and Low-Rank Matrix Factorization

Let $M \in \mathbb{R}^{n\times p}$ be a data matrix, e.g., M is a user-movie rating matrix with M_{ij} being the rating of user i on movie j. In practice, M can be incomplete and it is desirable to recover its missing values, e.g., to predict user ratings for better movie recommendations. This is the matrix completion problem and is widely applied in recommender systems. The traditional approach is statistical imputation. A modern approach is low-rank matrix factorization.[1]

MF assumes M has a low rank k ($k \ll \min\{n,p\}$) and thus can be factorized into two factors $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times p}$, i.e.

$$M = UV \tag{1}$$

In the user-movie-rate problem, U can be interpreted as the latent feature matrix of users (each row is a user and each column is a latent feature), and V can be interpreted as the latent feature matrix of movies (each column is a latent feature and each row is a movie).

MF finds U, V that minimize the completion errors on observed elements in M. Let $W\in\mathbb{R}^{n\times p}$ be an indicator matrix defined as W_{ij}=1 if rating M_{ij} is observed and W_{ij}=0 otherwise. The objective function of MF is

$$\begin{aligned} J(U,V) &= \sum\nolimits_{i=1}^{n}\sum\nolimits_{j=1}^{p} W_{ij}\cdot(U_{i:}V_{:j}-M_{ij})^2 + \lambda_1\sum_{i,a}U_{ia}^2 + \lambda_2\sum_{b,j}U_{bj}^2 \\ &= \sum_{i=1}^{n} ||\ (X_{i:}-U_{i:}V)\cdot[W_{i:}]\ ||^2 + \lambda_1\sum_{i=1}^{n}||U_{i:}||^2 + \lambda_2||V||^2 \\ &= \sum_{j=1}^{p} ||\ [W_{:j}]\cdot(X_{:j}-UV_{:j})\ ||^2 + \lambda_1||U||^2 + \lambda_2\sum_{j=1}^{p}||V_{:j}||^2 \\ &= ||W\circ(X-UV)||^2 + \lambda_1||U||^2 + \lambda_2||V||^2, \end{aligned} \tag{2}$$

where $[W_{i:}]$ is a $p\times p$ diagonal matrix with $W_{i:}$ on its diagonal and $[W_{:j}]$ is an $n\times n$ diagonal matrix with $W_{:j}$ on its diagonal; all norms are F-norm and the last two (regularization) terms are used to reduce model complexity.

We can minimize $J(U,V)$ by alternate least square.

Step 1: randomly initialize U and V.

Step 2: fix V and optimize $U_{i:}$. Let $\Sigma_{[i:]} = [W_{i:}][W_{i:}]^T$. First observe that

$$\begin{aligned} J(U_{i:}) &= \sum_{i=1}^{n} ||\ (X_{i:}-U_{i:}V)\cdot[W_{i:}]\ ||^2 + \lambda_1\sum_{i=1}^{n}||U_{i:}||^2 + \lambda_2||V||^2 \\ &= \sum_{i=1}^{n} X_{i:}\Sigma_{[i:]}X_{i:}^T - 2X_{i:}\Sigma_{[i:]}V^TU_{i:}^T + U_{i:}V\Sigma_{[i:]}V^TU_{i:}^T + \lambda_1\sum_{i=1}^{n}U_{i:}U_{i:}^T + \lambda_2||V||^2. \end{aligned} \tag{3}$$

[1] Koren and Volinsky. Matrix factorization techniques for recommender systems. IEEE Computer, 2009.

This is a quadratic function of $U_{i:}$. Applying the critical point method, we have

$$\frac{\partial J}{\partial U_{i:}} = -2X_{i:}\Sigma_{[i:]}V^T + 2U_{i:}V\Sigma_{[i:]}V^T + 2\lambda_1 U_{i:}. \tag{4}$$

Setting the above to zero and solving for $U_{i:}$, we have a closed-form solution for every i

$$U_{i:} = \left(X_{i:}\Sigma_{[i:]}V^T\right)\left(V\Sigma_{[i:]}V^T + \lambda_1 I\right)^{-1}. \tag{5}$$

Step 3: fix U and optimize $V_{:j}$. Let $\Sigma_{[:j]} = [W_{:j}]^T[W_{:j}]$. Observe that

$$\begin{aligned} J(V_{:j}) &= \sum_{j=1}^{p} || \ [W_{:j}] \cdot (X_{:j} - UV_{:j}) \ ||^2 + \lambda_1 ||U||^2 + \lambda_2 \sum_{j=1}^{p} ||V_{:j}||^2 \\ &= \sum_{i=1}^{n} X_{:j}^T \Sigma_{[:j]} X_{:j} - 2X_{:j}^T \Sigma_{[:j]} UV_{:j} + V_{:j}^T U^T \Sigma_{[:j]} UV_{:j} + \lambda_1 ||U||^2 + \lambda_2 \sum_{j=1}^{p} V_{:j}^T V_{:j}. \end{aligned} \tag{6}$$

Applying the critical point method, we have

$$\frac{\partial J}{\partial V_{:j}} = -2U^T\Sigma_{[:j]}X_{:j} + 2U^T\Sigma_{[i:]}UV_{:j} + 2\lambda_2 V_{:j}. \tag{7}$$

Setting the above to zero and solving for $V_{:j}$, we have a closed-form solution for every j

$$V_{:j} = \left(U^T\Sigma_{[:j]}U + \lambda_2 I\right)^{-1}\left(U^T\Sigma_{[:j]}X_{:j}\right). \tag{8}$$

Step 4: repeat step 2-3 until convergence.

Another way to optimize $J(U, V)$ is using alternate gradient descent.

Step 1: randomly initialize U and V.

Step 2: compute gradient $\frac{\partial J}{\partial U_{i:}}$ and update for every i for only one time

$$U_{i:} = U_{i:} - \eta \frac{\partial J}{\partial U_{i:}}. \tag{9}$$

Step 3: compute gradient $\frac{\partial J}{\partial V_{:j}}$ and update for every j for only one time

$$V_{:j} = V_{:j} - \eta \frac{\partial J}{\partial V_{:j}}. \tag{10}$$

Step 4: repeat step 2-3 until convergence