matrix factorization

Matrix Factorization takes into account both the similarities between other user's ratings and items.

$$\textbf{Carmen}\begin{bmatrix}\textbf{Alien} & \textbf{Bug's Life} & \textbf{Cars} & \textbf{Dark Night}\\ 5 & - & 1 & 3\end{bmatrix}$$

$$\textbf{Carmen}^{\mathbf{1}}\begin{bmatrix}\textbf{Scary} & \textbf{Kiddy}\\ 5 & 1\end{bmatrix} \rightarrow \textbf{Dark Knight}\begin{bmatrix}\textbf{Scary} & \textbf{Kiddy}\\ \frac{1}{2} & \frac{1}{2}\end{bmatrix}$$

Prediction from **Carmen's** rating of **Dark Knight**: $\mathbf{5}\times\frac{\mathbf{1}}{\mathbf{2}}+\mathbf{1}\times\frac{\mathbf{1}}{\mathbf{2}}=\mathbf{3}\rightarrow$ **inner product of the vectors**

$$\textbf{Carmen}^{\mathbf{1}}\begin{pmatrix}\textbf{Scary} & \textbf{Kiddy}\\ 5 & 1\end{pmatrix}\begin{pmatrix}\textbf{Dark Knight}\\ 1/2\\ 1/2\end{pmatrix}\begin{matrix}\textbf{Scary}\\ \textbf{Kiddy}\end{matrix} = \textbf{Carmen}^{\mathbf{1}}\begin{pmatrix}\textbf{Dark Knight}\\ \mathbf{3}\end{pmatrix}$$

The above logic implies predictions can be made for **Carmen's** rating of many different **movies** and equally many different **users**:

$$\begin{matrix}\textbf{Carmen}\\ \vdots\end{matrix}\begin{bmatrix}\textbf{Alien} & \textbf{Bug's Life} & \textbf{Cars} & \textbf{Dark Night} & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ 5 & - & 1 & 3 & 2 & - & 5 & 1 & 3 & 1 & 5 & -\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\end{bmatrix}$$

Matrix Factorization determines the **latent factors**

Determining **latent space**

- How to represent each user in the space
- How to represent each movie in the space

If the latent factors are ideal, few are required to recover ratings for accurate approximations of real ratings, regardless of having thousands of different movies to rate. If a new movie exists that a user has not seen prior, all that matters is the latent state of the movie and the latent state of the user; if these two attributes are known, predications can be accurately assigned:

$$\text{Rating for user } i \text{ on movie } j \approx \hat{R}_{ij} = \sum_{\text{latent factors } \ell} \text{user}_{u,\ell}\text{movie}_{\ell,m}$$

Prediction from **Leonore's** rating of **Dark Knight**: $\mathbf{3}\times\frac{\mathbf{1}}{\mathbf{2}}+2\times\frac{\mathbf{1}}{\mathbf{2}}=\mathbf{2.5}\rightarrow$ **vectors inner product**

$$\begin{matrix}\textbf{Carmen}^{\mathbf{1}}\\ \textbf{Leonore}^{\mathbf{1}}\end{matrix}\begin{pmatrix}\textbf{Scary} & \textbf{Kiddy}\\ 5 & 1\\ 3 & 2\end{pmatrix}\begin{pmatrix}\textbf{Dark Knight}\\ 1/2\\ 1/2\end{pmatrix}\begin{matrix}\textbf{Scary}\\ \textbf{Kiddy}\end{matrix} = \begin{matrix}\textbf{Carmen}^{\mathbf{1}}\\ \textbf{Leonore}^{\mathbf{1}}\end{matrix}\begin{pmatrix}\textbf{Dark Knight}\\ \mathbf{3}\\ \mathbf{2.5}\end{pmatrix}$$

Matrix Factorization applied to multiple movies and users:

Customer → Latent **Latent → Movies**

$$\begin{matrix} \\ \text{Carmen}^1 \\ \text{Leonore}^1 \\ \\ \text{Joeseph}^1 \end{matrix}\begin{pmatrix} \textbf{Scary} & \textbf{Kiddy} & \cdots & \textbf{Gore} \\ 5 & 1 & \cdots & 4 \\ 3 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 2 & 3 & \cdots & 1 \\ 4 & 5 & \cdots & 2 \end{pmatrix}\begin{pmatrix} \textbf{Dark Knight} & \textbf{Jurassic Park} & & \textbf{Zootopia} \\ 1/2 & 2/3 & \cdots\ 1/4 & 1/9 \\ 1/2 & 1/3 & \cdots\ 1/3 & 1/3 \\ \vdots & \vdots & \ddots\ \vdots & \vdots \\ 1/8 & 1/4 & \cdots\ 1/5 & 1/7 \end{pmatrix}\begin{matrix} \textbf{Scary} \\ \textbf{Kiddy} \\ \\ \textbf{Gore} \end{matrix}$$

$$= \begin{matrix} \\ \text{Carmen}^1 \\ \text{Leonore}^1 \\ \\ \text{Joeseph}^1 \end{matrix}\begin{pmatrix} \textbf{Dark Knight} & \textbf{Jurassic Park} & \textbf{Zootopia} \\ 3 & 3.7 & \cdots \\ 2.5 & 2.7 & \vdots \\ \vdots & \vdots & \ddots \\ \cdots & \cdots & \cdots \end{pmatrix}$$

The **first matrix** in the above illustration represents the **latent factors of users**. The **second matrix** represents the **latent factors of movies**. The **third matrix** represents the **inner product** of the latter:

Customer → Latent **Latent → Movies** **Customer → Movies**

$$\begin{pmatrix} 5 & 1 & \cdots & 4 \\ 3 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 2 & 3 & \cdots & 1 \\ 4 & 5 & \cdots & 2 \end{pmatrix}\boldsymbol{P}\begin{pmatrix} 1/2 & 2/3 & \cdots & 1/4 & 1/9 \\ 1/2 & 1/3 & \cdots & 1/3 & 1/3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1/8 & 1/4 & \cdots & 1/5 & 1/7 \end{pmatrix}\boldsymbol{Q^T}\begin{pmatrix} 3 & 3.7 & \cdots & \cdots \\ 2.5 & 2.7 & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}\boldsymbol{\hat{R}}$$

The above model: $P \times Q^T = \hat{R}$ with the **goal** of choosing $\boldsymbol{P}$ and $\boldsymbol{Q^T}$ such that $\boldsymbol{\hat{R}}$ is accurate.

$\boldsymbol{\hat{R}}$ is accurate when: $\boldsymbol{P}$ and $\boldsymbol{Q^T}$ are chosen to **minimize**:

$$\sum_{\text{users } U,\ \text{movies},m} \left(R_{u,m} - \hat{R}_{u,m}\right)^2 = \sum_{\text{users } U,\ \text{movies},m} \left(\text{error}_{u,m}\right)^2$$

The estimated rating should ideally be as close to the true rating as possible. The squared distance between the true ratings and the estimated ratings are minimized (**Sum of Squares Error**).

The **optimization problem** above is solved through **gradient descent** on the **errors**. The method begins by minimizing $\boldsymbol{P}$, then minimizing $\boldsymbol{Q}$, then reiterating until convergence.

Alternate minimization scheme:

$$\text{error}_{u,m}^2 = \left(R_{u,m} - \hat{R}_{u,m}\right)^2 = \left(r_{ij} - \sum_{\text{latent } \ell=1}^{L} \boldsymbol{p}_{u,\ell}\boldsymbol{q}_{\ell,m}\right)^2$$

$$\frac{d}{d\boldsymbol{p}_{u,\ell}}\text{error}_{u,m}^2 = -2\left(R_{u,m} - \hat{R}_{u,m}\right)\boldsymbol{q}_{\ell,m} = -2\ \text{error}_{u,m}\boldsymbol{q}_{\ell,m}$$

$$\boldsymbol{p}_{u,\ell} := \boldsymbol{p}_{u,\ell} - \alpha\frac{d\ \text{error}_{u,m}^2}{d\boldsymbol{p}_{u,\ell}} = \boldsymbol{p}_{u,\ell} + 2\alpha\ \text{error}_{u,m}\boldsymbol{q}_{\ell,m}$$

$$\frac{d}{d\boldsymbol{q}_{u,\ell}}\text{error}_{u,m}^2 = -2\left(R_{u,m} - \hat{R}_{u,m}\right)\boldsymbol{p}_{\ell,m} = -2\ \text{error}_{u,m}\boldsymbol{p}_{\ell,m}$$

$$\boldsymbol{q}_{u,\ell} := \boldsymbol{q}_{u,\ell} - \alpha\frac{d\ \text{error}_{u,m}^2}{d\boldsymbol{q}_{u,\ell}} = \boldsymbol{q}_{u,\ell} + 2\alpha\ \text{error}_{u,m}\boldsymbol{p}_{\ell,m}$$

If the learning rate $\boldsymbol{\alpha}$ is set to be too **large**, gradient descent is likely to overshoot the minimum and oscillate back and forth. If the learning rate $\boldsymbol{\alpha}$ is set to be too **small**, gradient descent will be slow to coverage and might fail entirely. **Regularization** can also be added to the above illustration of **gradient descent**, adding an additional factor to the expression.

It is important to note that the above **minimization objective** does not necessarily provide the best possible salutation at a **global minimum**. The objective is not **convex** and thus, **local minimizers** may result when implementing but the technique works well in practice.

Questioning the above method:

Is **Matrix Factorization** illustrated above synonymous to setting the **unknowns** to **0** and performing least squares regression?

$$\begin{array}{c} \\ \textbf{Carmen}^{\mathbf{1}} \\ \textbf{Leonore}^{\mathbf{1}} \\ \\ \textbf{Joeseph}^{\mathbf{1}} \end{array} \begin{array}{c} \widehat{\boldsymbol{R}} \\ \begin{pmatrix} 3 & 3.7 & 2.7 & 2.9 \\ 2.5 & 2.7 & 4.1 & 1.3 \\ 2.1 & 0.9 & 0.5 & 2.8 \\ 4.6 & 2.6 & 4.2 & 0.1 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ \textbf{Carmen}^{\mathbf{1}} \\ \textbf{Leonore}^{\mathbf{1}} \\ \\ \textbf{Joeseph}^{\mathbf{1}} \end{array} \begin{array}{c} \boldsymbol{R} \\ \begin{pmatrix} 3 & 3.7 & ? & 2.9 \\ 2.5 & ? & 4.1 & ? \\ ? & 0.9 & 0.5 & 2.8 \\ ? & 2.6 & 4.2 & ? \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ \textbf{Carmen}^{\mathbf{1}} \\ \textbf{Leonore}^{\mathbf{1}} \\ \\ \textbf{Joeseph}^{\mathbf{1}} \end{array} \begin{array}{c} \widehat{\boldsymbol{R}} \\ \begin{pmatrix} 3 & 3.7 & 2.7 & 2.9 \\ 2.5 & 2.7 & 4.1 & 1.3 \\ 2.1 & 0.9 & 0.5 & 2.8 \\ 4.6 & 2.6 & 4.2 & 0.1 \end{pmatrix} \end{array} \quad \begin{array}{c} \\ \textbf{Carmen}^{\mathbf{1}} \\ \textbf{Leonore}^{\mathbf{1}} \\ \\ \textbf{Joeseph}^{\mathbf{1}} \end{array} \begin{array}{c} \boldsymbol{R} \\ \begin{pmatrix} 3 & 3.7 & \mathbf{0} & 2.9 \\ 2.5 & \mathbf{0} & 4.1 & ? \\ \mathbf{0} & 0.9 & 0.5 & 2.8 \\ \mathbf{0} & 2.6 & 4.2 & \mathbf{0} \end{pmatrix} \end{array}$$

The answer is **no**; if the **unknowns** are replaced with **zero** in **matrix factorization**, any missing votes will suffer a large penalty which becomes problematic when many votes are missing. Only the errors that can be calculated are considered.