Matrix Factorization in Recommender Systems

2018.9.16

References

Matrix Factorization Techniques for Recommender Systems IEEE Computer Society Computer 2009

Probabilistic Matrix Factorization NIPS 2008

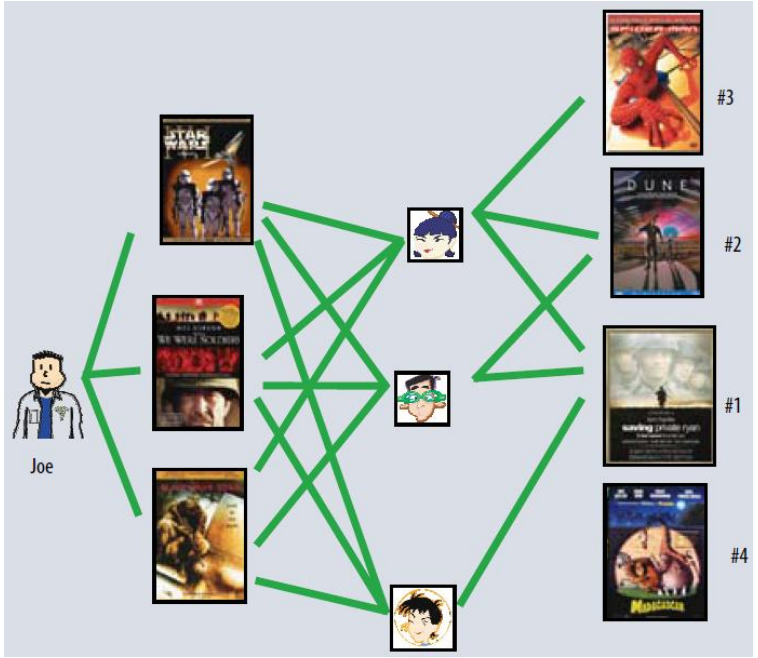
Local Low Rank Matrix Approximation ICML 2013

Learning the parts of objects by nonnegative matrix factorization Nature 1999

Recommender system strategies

Content filtering

Creates a profile for each user or product to characterize its nature



Collaborative filtering

Analyze relationships between users and interdependencies among products to identify new user-item associations

Neighborhood methods

Neighborhood methods are centered on computing the relationships between items or, alternatively, between users.

Latent factor models

Explain the ratings by characterizing both items and users on, say, 20 to 100 factors inferred from the ratings patterns

A basic matrix factorization model

singular value decomposition(SVD)

	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	∅	4	5	5
Cindy	2	2	4	∅
David	3	∅	5	2

Table 1. A fragment of a rating matrix for a movie recommender system.

- the high portion of missing values caused by sparseness
- carelessly addressing only the relatively few known entries is highly prone to overfitting

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(|| q_i ||^2 + || p_u ||^2)$$

SGD

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u$$

- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

ALS

The first is when the system can use parallelization

The second case is for systems centered on implicit data.

Because the training set cannot be considered sparse, looping over each single training case—as gradient descent does—would not be practical. ALS can efficiently handle such cases.

Adding biases

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

global average, item bias, user bias, and user item interaction.

$$\min_{p^*, q^*, b^*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (|| p_u ||^2 + || q_i ||^2 + b_u^2 + b_i^2)$$

Additional input sources

Often a system must deal with the cold start problem, wherein many users supply very few ratings, making it difficult to reach general conclusions on their taste.

Recommender systems can use **implicit feedback** to gain insight into user preferences

N(u) denotes the set of items for which user *u* expressed an implicit preference.

A user who showed a preference for items in N(u) is characterized by the vector

$$\sum_{i \in N(u)} x_i$$

Another information source is known user attributes, for example, demographics.

Each attribute to describe a user through the set of user-associated attributes

$$\sum_{a \in A(u)} y_a$$

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left[p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a \right]$$

Temporal dynamics

The system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

An item's popularity might change over time.

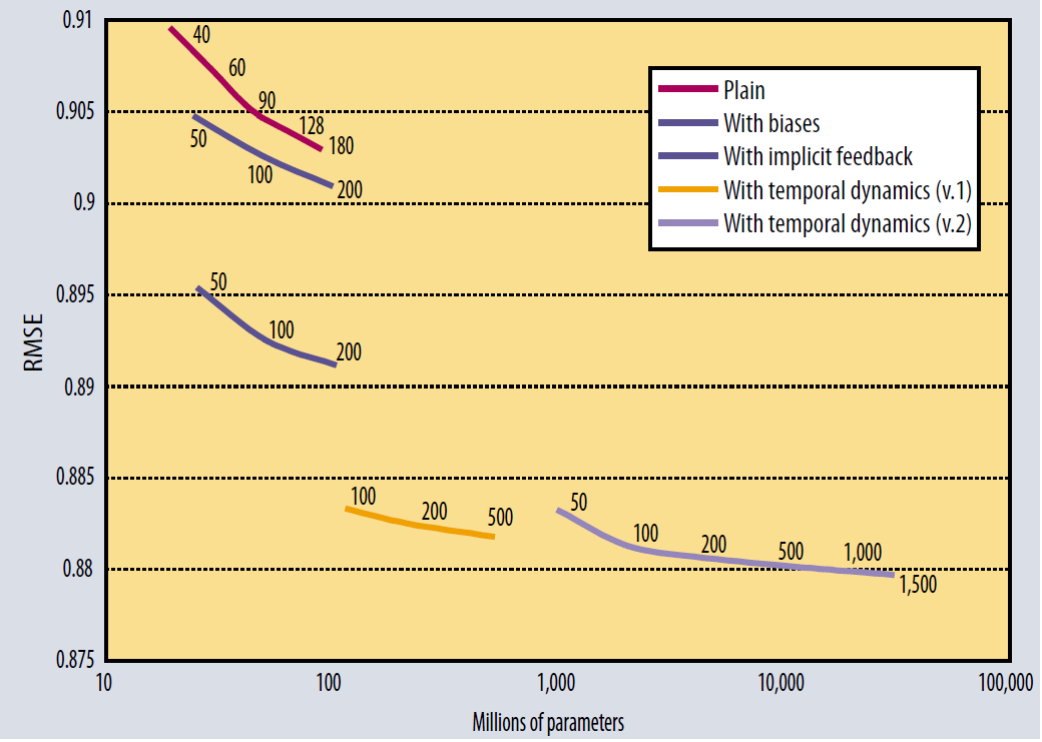
A natural drift in a user's rating scale

Temporal dynamics go beyond this; they also affect user preferences and therefore the interaction between users and items

Inputs with varying confidence levels

In several setups, not all observed ratings deserve the same weight or confidence.

$$\min_{p^*,q^*,b^*} \sum_{(u,i)\in\kappa} c_{ui}(r_{ui} - \mu - b_u - b - p_u^T q_i)^2 + \lambda\,(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

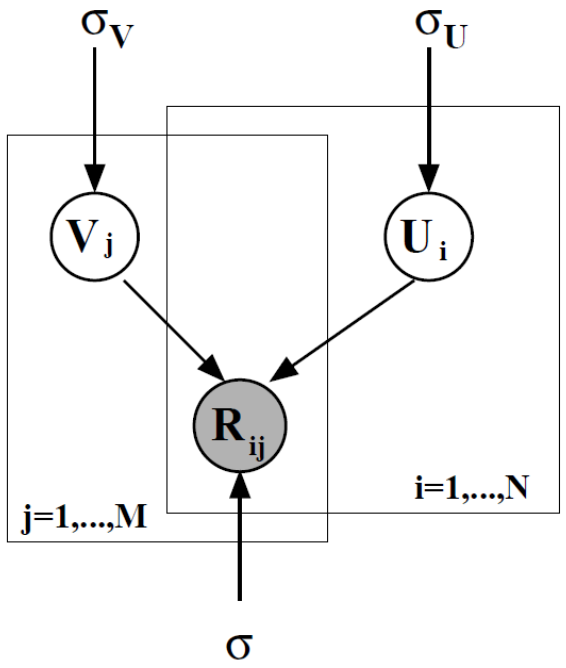


Probabilistic Matrix Factorization

NIPS 2008

$$\hat{R} = U^T V \qquad U \in R^{D \times N} \quad V \in R^{D \times M}$$

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|U_i^T V_j, \sigma^2) \right]^{I_{ij}}$$



PRML8.1

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 \mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 \mathbf{I}).$$

$$\ln p(U, V|R, \sigma^2, \sigma_V^2, \sigma_U^2) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} (R_{ij} - U_i^T V_j)^2 - \frac{1}{2\sigma_U^2} \sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^{M} V_j^T V_j$$
$$- \frac{1}{2} \left(\left(\sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} \right) \ln \sigma^2 + ND \ln \sigma_U^2 + MD \ln \sigma_V^2 \right) + C, \qquad (3)$$

$$\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} V_j^T V_j$$
$$-\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\right)\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C, \qquad (3)$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij}-U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\| V_j \|_{Fro}^2$$

$$\lambda_U = \sigma^2/\sigma_U^2 \qquad \lambda_V = \sigma^2/\sigma_V^2$$

the dot product between user- and movie-specific feature vectors is passed through the logistic function

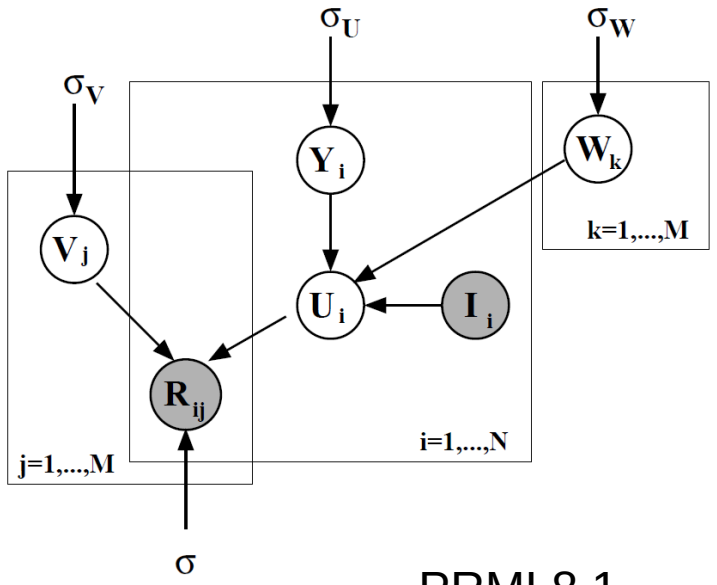
$$g(x) = 1/(1+\exp(-x))$$

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|g(U_i^T V_j),\sigma^2)\right]^{I_{ij}}$$

$$t(x) = (x-1)/(K-1)$$

Constrained PMF

Once a PMF model has been fitted, users with very few ratings will have feature vectors that are close to the prior mean, or the average user, so the predicted ratings for those users will be close to the movie average ratings.



PRML8.1

latent similarity constraint matrix $W \in R^{D \times M}$

A column in W captures the effect of a user having rated a particular movie has on the prior mean of the user's feature vector.

users that have seen the same (or similar) movies will have similar prior distributions for their feature vectors

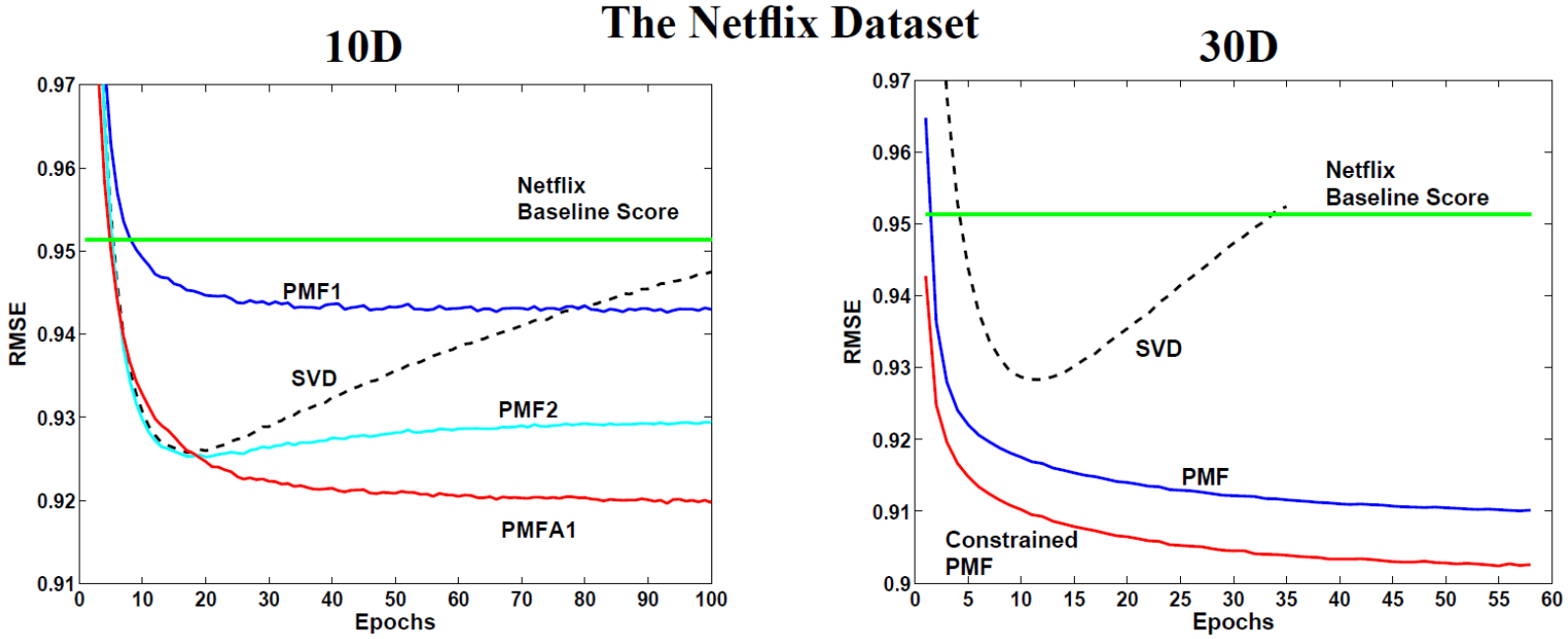
$$U_i = Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}$$

$$p(W|\sigma_W) = \prod_{k=1}^{M} \mathcal{N}(W_k|0, \sigma_W^2 \mathbf{I})$$

$$p(R|Y, V, W, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|g([Y_i + \frac{\sum_{k=1}^{M} I_{ik} W_k}{\sum_{k=1}^{M} I_{ik}}]^T V_j), \sigma^2) \right]^{I_{ij}}$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - g\left(\left[Y_i + \frac{\sum_{k=1}^{M} I_{ik}W_k}{\sum_{k=1}^{M} I_{ik}}\right]^T V_j\right)\right)^2$$
$$+\frac{\lambda_Y}{2}\sum_{i=1}^{N} \| Y_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2 + \frac{\lambda_W}{2}\sum_{k=1}^{M} \| W_k \|_{Fro}^2$$

The Netflix Dataset



$[n]$ $\{1,\ldots,n\}$ $M \in \mathbb{R}^{n_1 \times n_2}$ $\hat{M} = UV^T$ $U \in \mathbb{R}^{n_1 \times r}$ $V \in \mathbb{R}^{n_2 \times r}$

$$\mathtt{A} \stackrel{\text{def}}{=} \{(a_1, b_1), \ldots, (a_m, b_m)\} \subseteq [n_1] \times [n_2] \qquad \{M_{a,b} : (a,b) \in \mathtt{A}\}$$

$\mathcal{T} : [n_1] \times [n_2] \to \mathbb{R}^{n_1 \times n_2}$ Mappings from matrix indices to a matrix space

$$\Pi_{\mathtt{A}} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2} \qquad [\Pi_{\mathtt{A}}(M)]_{a,b} \stackrel{\text{def}}{=} \begin{cases} M_{a,b} & (a,b) \in \mathtt{A} \\ 0 & \text{otherwise} \end{cases}$$

$$[A \odot B]_{i,j} = A_{i,j} B_{i,j} \qquad \|X\|_F \stackrel{\text{def}}{=} \sqrt{\textstyle\sum_i \sum_j X_{i,j}^2}$$

$$\|X\|_\infty \stackrel{\text{def}}{=} \sup_{i,j} |X_{i,j}| \qquad \|X\|_* \stackrel{\text{def}}{=} \textstyle\sum_{i=1}^{r} \sigma_i(X)$$

Two popular approaches for constructing a low-rank approximation

Incomplete SVD

$$\hat{M} = UV^T$$

$$(U, V) = \arg\min_{U,V} \sum_{(a,b)\in \mathtt{A}} ([UV^T]_{a,b} - M_{a,b})^2$$

$$\hat{M} = \arg\min_{X} \|\Pi_{\mathtt{A}}(X - M)\|_F \ \text{ s.t. } \operatorname{rank}(X) = r$$

Compressed Sensing

$$\hat{M} = \arg\min_{X} \|X\|_* \quad \text{s.t.} \quad \|\Pi_{\mathtt{A}}(X - M)\|_F < \epsilon$$

Do not need to constrain the rank of M in advance, while being convex, may not necessarily scale up easily to large matrices.

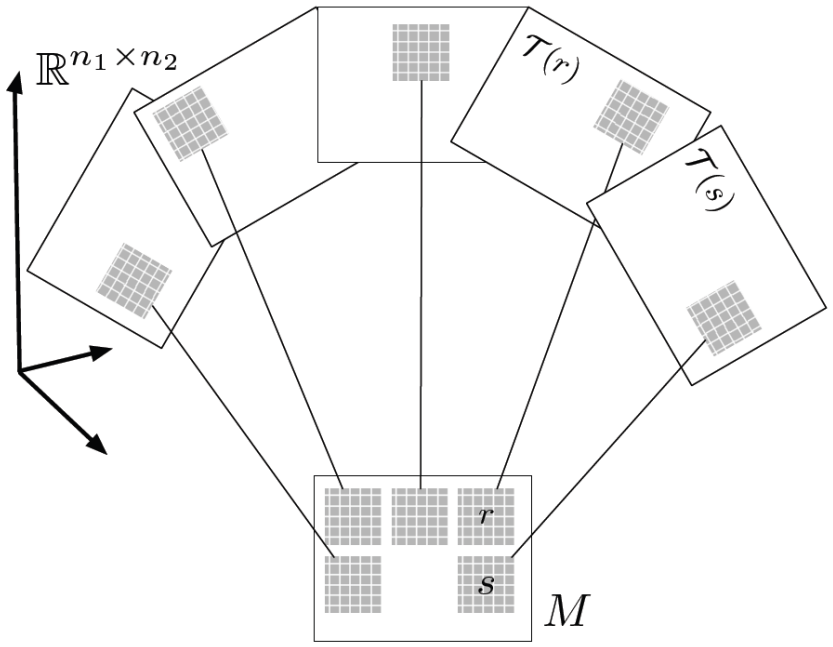
Local low-rank matrix approximation

Need to pose an assumption that there exists a metric structure over $[n_1] \times [n_2]$

$d((a,b),(a',b'))$ reflects the similarity between the rows a and a' and columns b and b'.

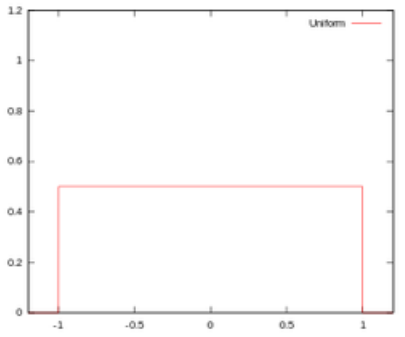
$$d(i,j) = \arccos\left(\frac{\langle u_i, u_j \rangle}{\|u_i\| \cdot \|u_j\|}\right)$$

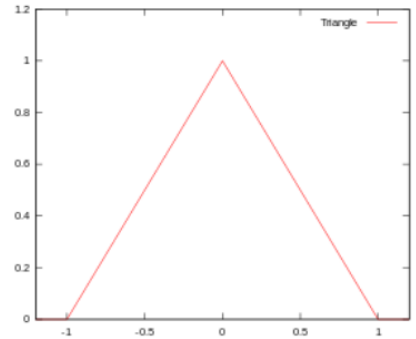
Assume that the model is characterized by multiple low-rank $[n_1] \times [n_2]$ matrices.

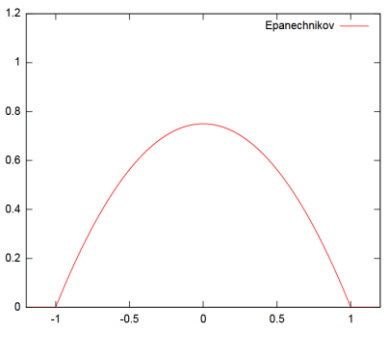


$$\mathcal{T} : [n_1] \times [n_2] \to \mathbb{R}^{n_1 \times n_2} \quad \text{where} \quad \mathcal{T}_{a,b}(a,b) = M_{a,b}$$

$$K_h(s_1, s_2) \qquad s_1, s_2 \in [n_1] \times [n_2]$$







Incomplete SVD

$$\hat{\mathcal{T}}(a,b) = \arg\min_{X} \|K_h^{(a,b)} \odot \Pi_{\mathtt{A}}(X - M)\|_F$$
$$\text{s.t.} \quad \text{rank}(X) = r \ .$$

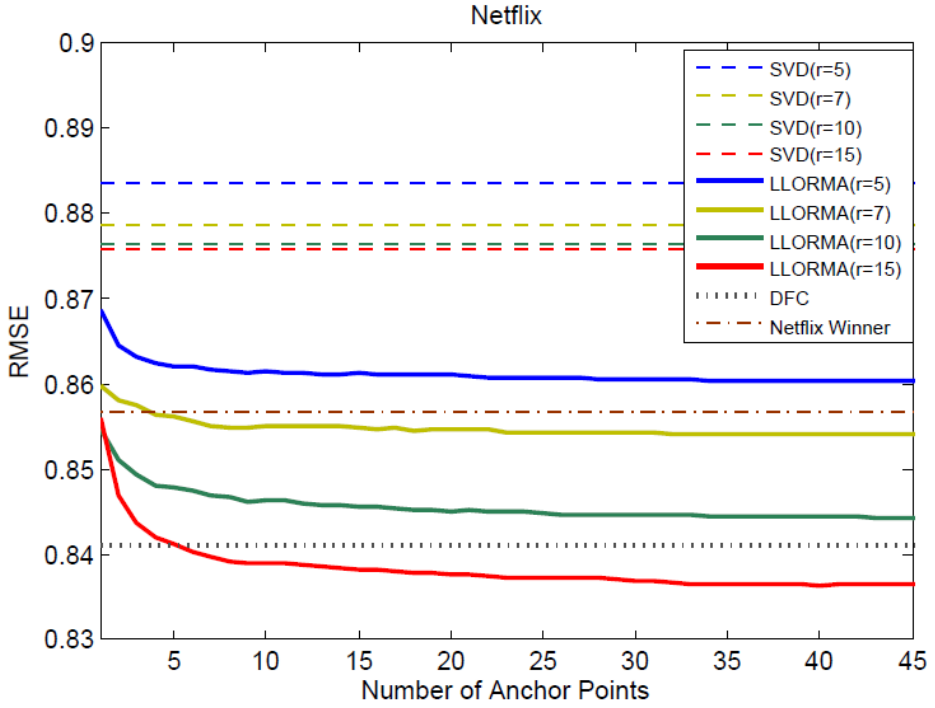
Compressed Sensing

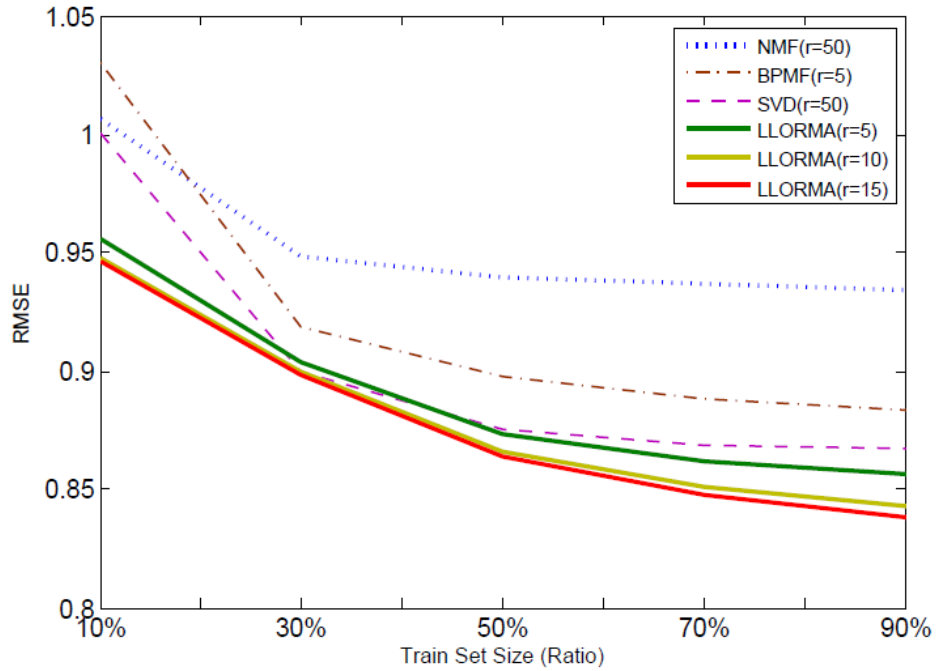
$$\hat{\mathcal{T}}(a,b) = \arg\min_{X} \|X\|_*$$
$$\text{s.t.} \quad \|K_h^{(a,b)} \odot \Pi_{\mathtt{A}}(X - M)\|_F < \epsilon$$

$$\hat{M}_{a,b} = \hat{\mathcal{T}}_{a,b}(a,b), \quad (a,b) \in [n_1] \times [n_2]$$

PRML6.3.1 Nadaraya-Watson

$$\hat{\hat{\mathcal{T}}}(s) = \sum_{i=1}^{q} \frac{K_h(s_i, s)}{\sum_{j=1}^{q} K_h(s_j, s)} \hat{\mathcal{T}}(s_i)$$





Algorithm 1 The LLORMA Algorithm

Input: $M \in \mathbb{R}^{n_1 \times n_2}, h_1, h_2, r, q$
for all $t = 1, ..., q$ **in parallel do**
  $(a_t, b_t) :=$ a randomly selected train element
  for $i = 1 \to n_1$ **do**
    $[K_{h_1}^{(a_t)}]_i := (1 - d_a(a_t, i)^2)\mathbf{1}_{\{d_a(a_t,i)<h\}}$
  end for
  for $j = 1 \to n_2$ **do**
    $[K_{h_2}^{(b_t)}]_j := (1 - d_b(b_t, j)^2)\mathbf{1}_{\{d_b(b_t,j)<h\}}$
  end for
  $(U^{(t)}, V^{(t)}) := \arg\min_{U,V} [\lambda_U \Omega(U) + \lambda_V \Omega(V)$
    $+ \sum_{(i,j)\in\mathtt{A}} [K_{h_1}^{(a_t)}]_i [K_{h_2}^{(b_t)}]_j \left([UV^T]_{i,j} - M_{i,j}\right)^2]$
end for
Output: $\hat{\mathcal{T}}(s_t) = U^{(t)} V^{(t)^T}, t = 1, \ldots, q$

NMF Nature 1999

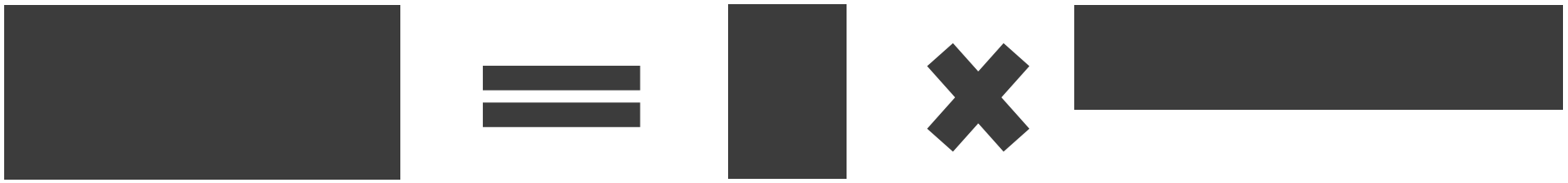
$$V_{m\times n} \approx W_{m\times k} \times H_{k\times n} = \hat{V}_{m\times n} \qquad W_{m\times k} \geqslant 0 \qquad H_{k\times n} \geqslant 0$$

$$\|A-B\|^2 = \sum_{i,j} (A_{i,j} - B_{i,j})^2$$

$$D(A \parallel B) = \sum_{i,j} \left(A_{i,j} log \frac{A_{i,j}}{B_{i,j}} - A_{i,j} + B_{i,j} \right)$$

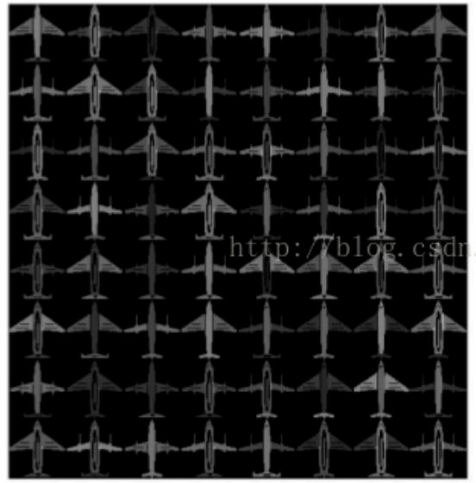
$$minimize \ \|V - WH\|^2$$
$$s.t. \ W \geqslant 0, H \geqslant 0$$

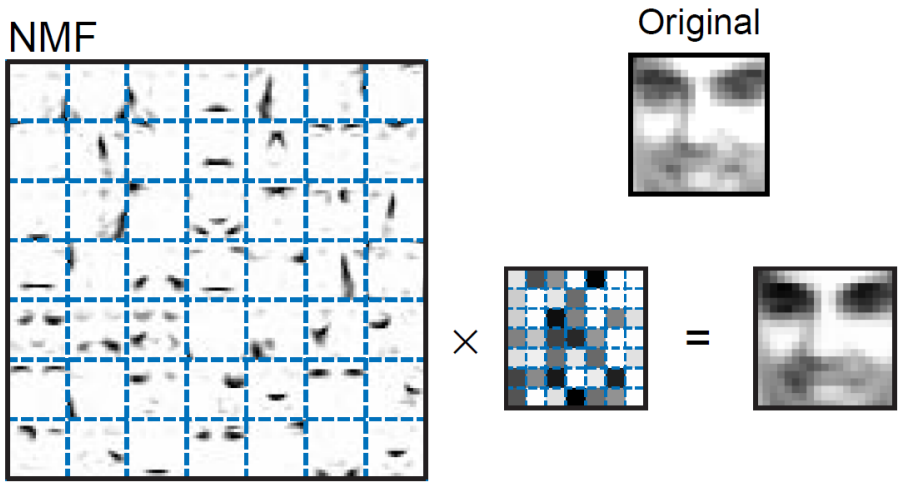
$$minimize \ D(V \parallel WH)$$
$$s.t. \ W \geqslant 0, H \geqslant 0$$

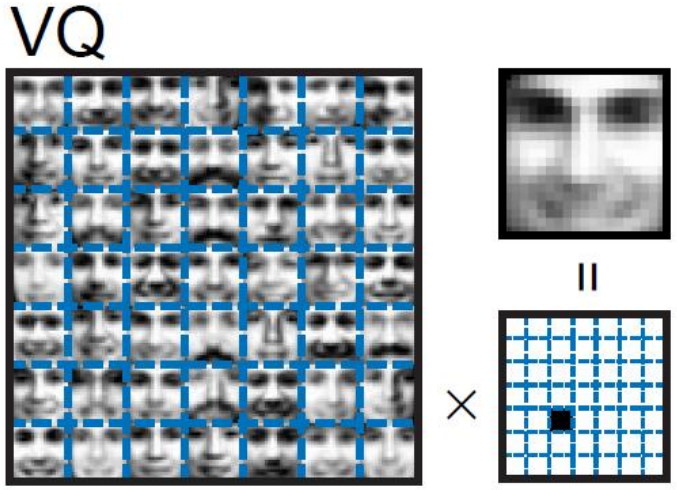


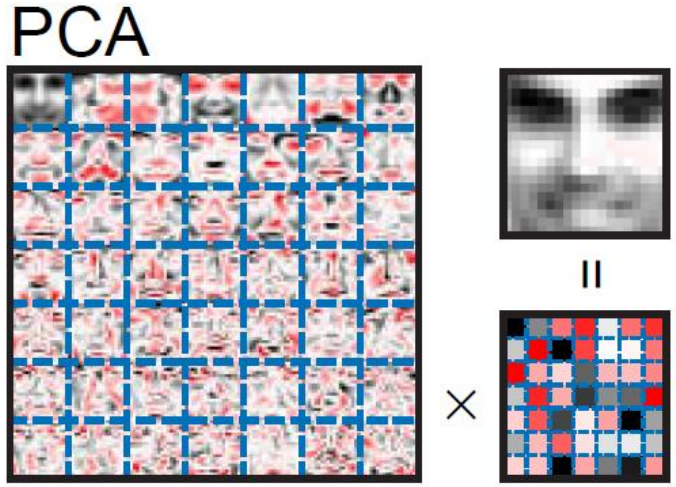
$$v \approx Wh$$

The NMF basis and encodings contain a large fraction of vanishing coefficients, so both the basis images and image encodings are sparse.
The basis images are sparse because they are non-global and contain Several versions of mouths, noses, and other facial parts.









矩阵分解有如下优点：

1. 能将高维的矩阵映射成两个低维矩阵的乘积，很好地解决了数据稀疏的问题；
2. 具体实现和求解都很简洁，预测的精度也比较好；
3. 模型的可扩展性也非常优秀，其基本思想也能广泛运用于各种场景中。

相对的，矩阵分解的缺点则有：

1. 可解释性很差，其隐空间中的维度无法与现实中的概念对应起来；
2. 训练速度慢，不过可以通过离线训练来弥补这个缺点；
3. 实际推荐场景中往往只关心topn结果的准确性，此时考察全局的均方差显然是不准确的。