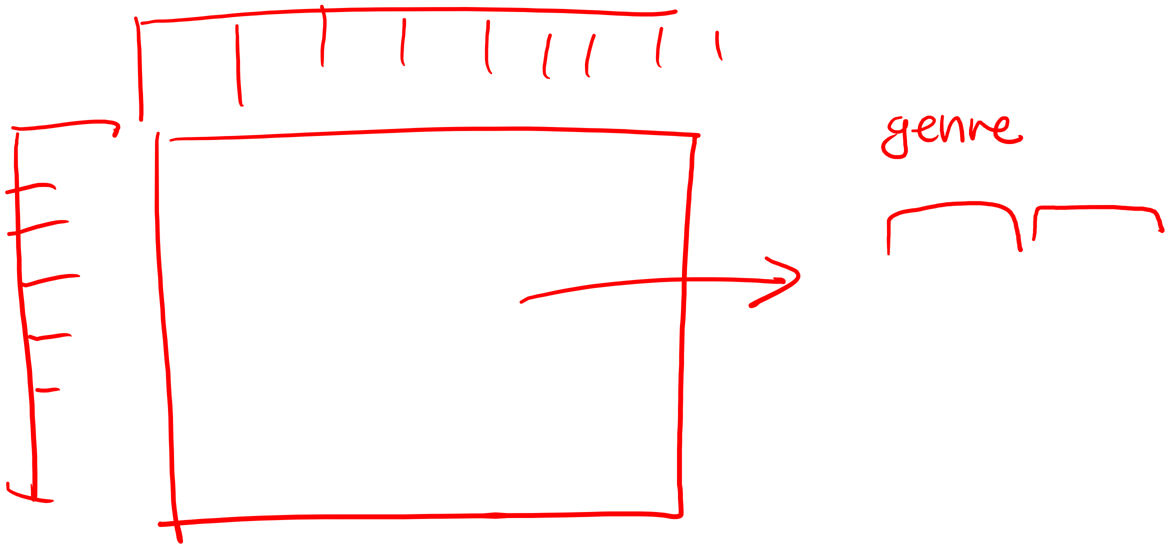
Matrix Factorization

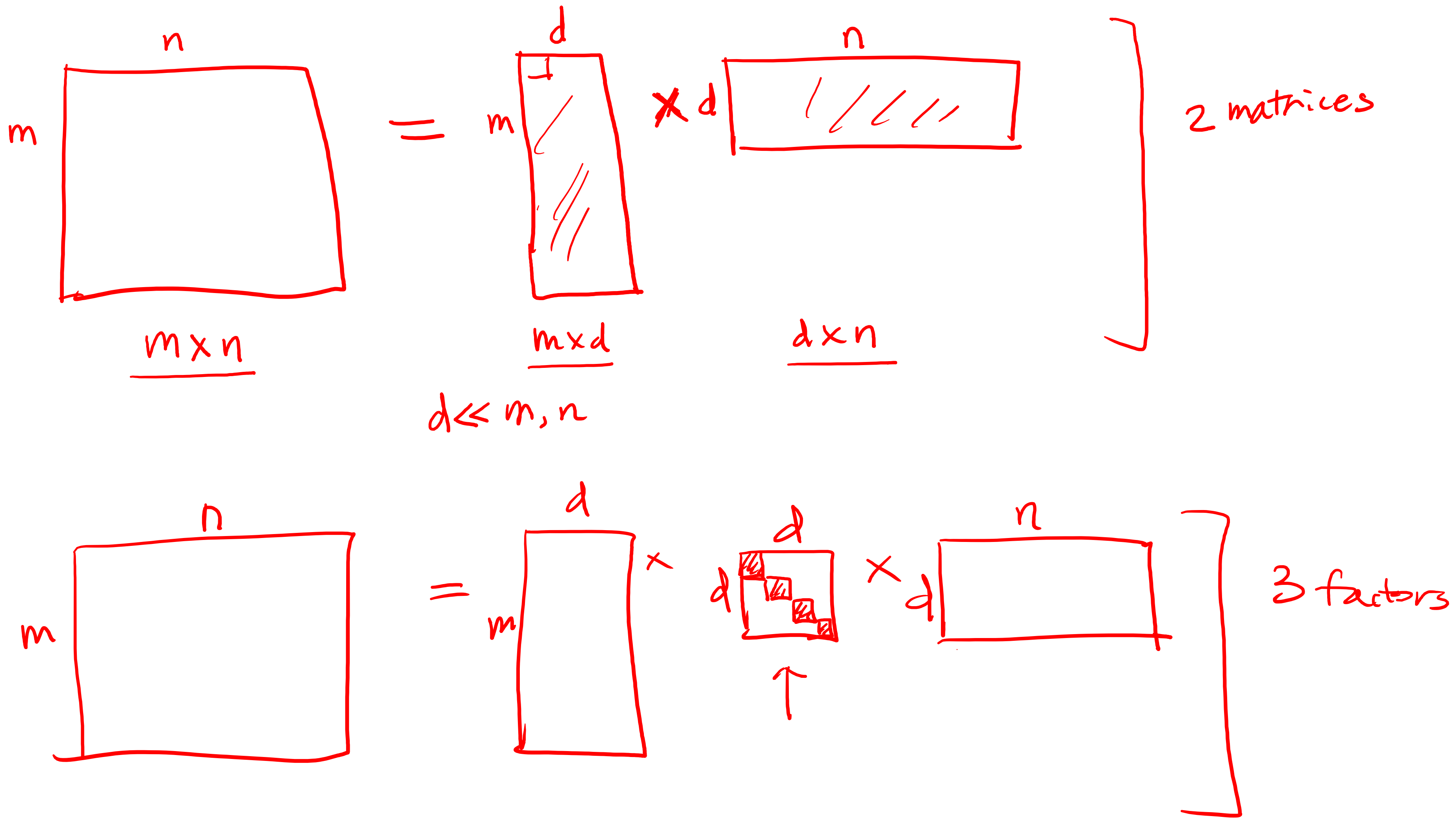
Motivation

Collaborative Filtering

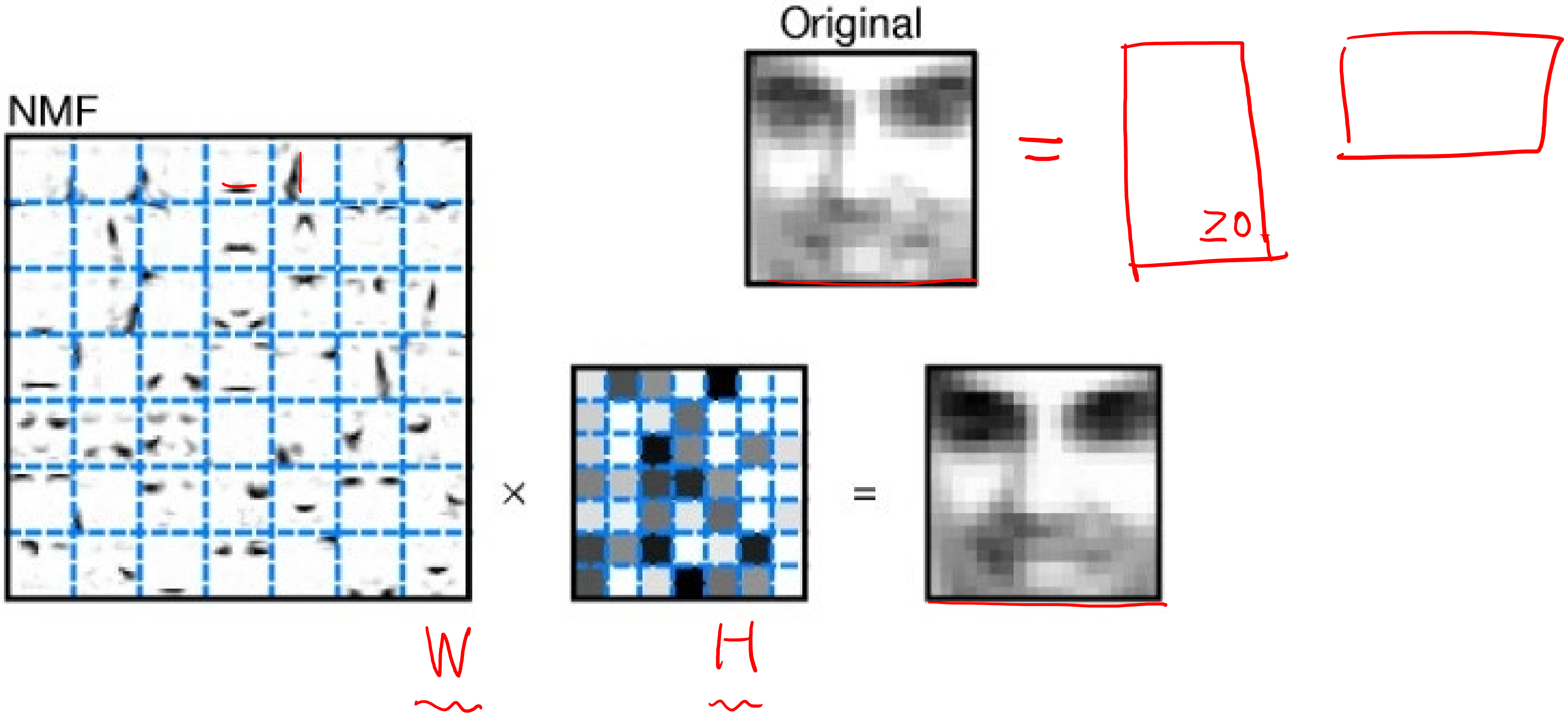
- Neighborhood methods
- Latent factor models



What is Matrix Factorization?

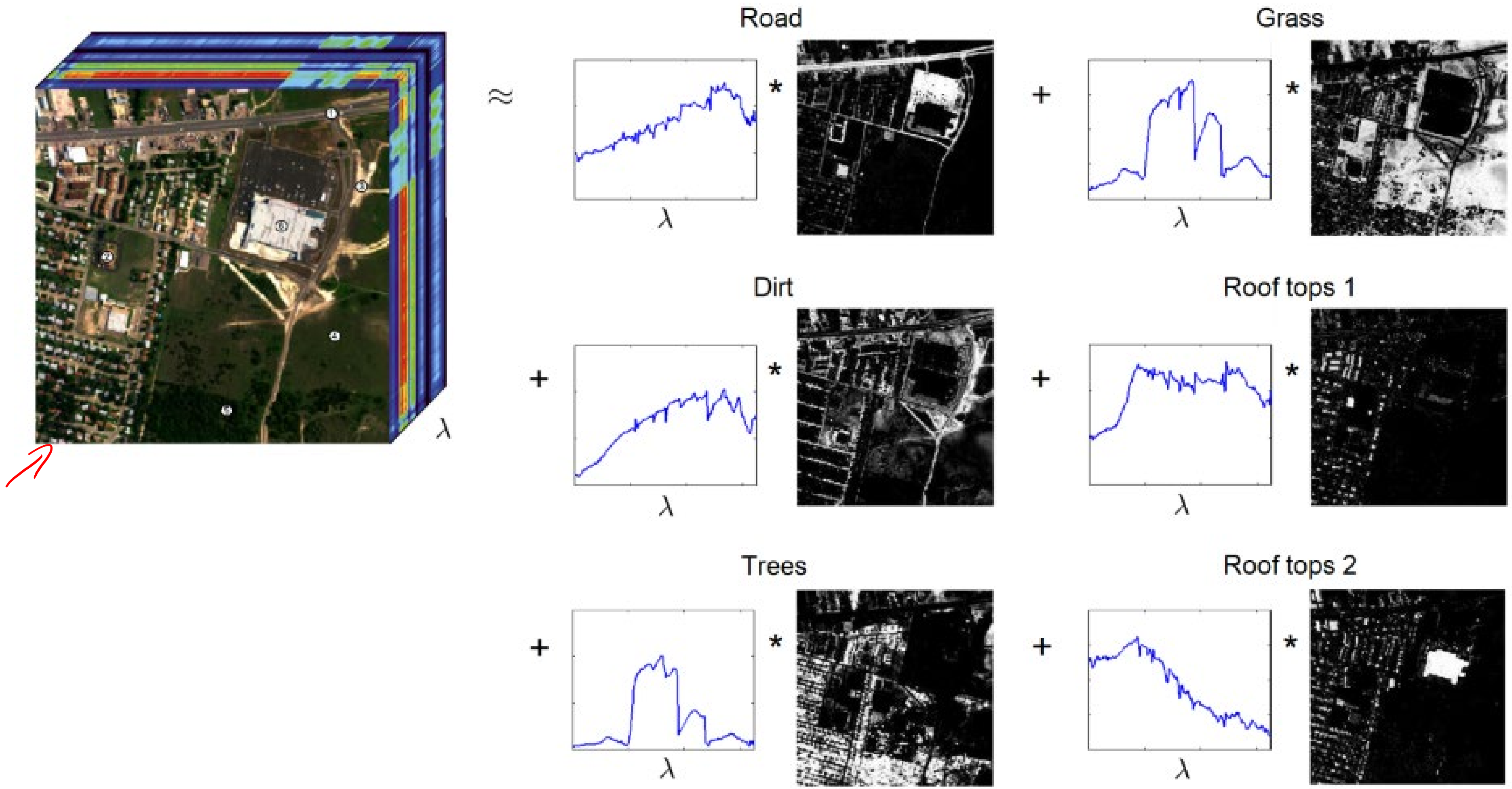


Applications of MF



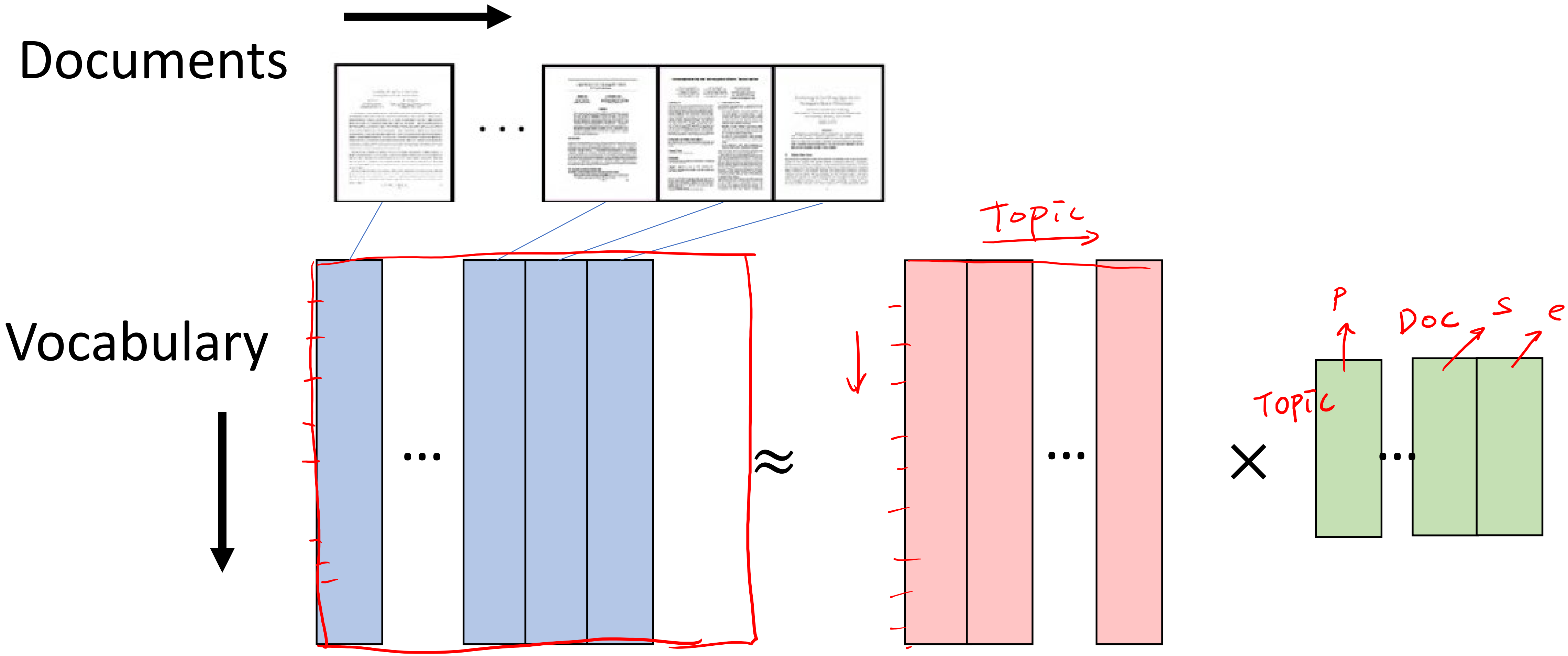
Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)

Applications of MF



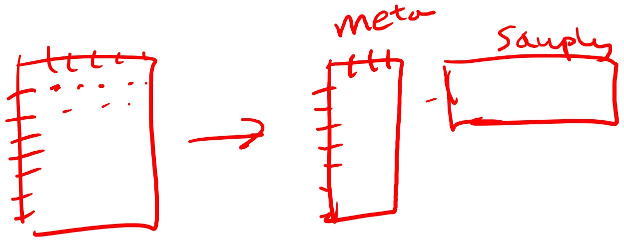
Gillis, N.: Learning with nonnegative matrix factorizations.
SIAM News 52(5), 1–3 (2019)

Applications of MF



Applications of MF

- Audio signal separation
- Analytic Chemistry
- Gene expression analysis
- Recommender systems



Matrix Factorization methods

- Singular Value Decomposition

- Non-negative Matrix Factorization

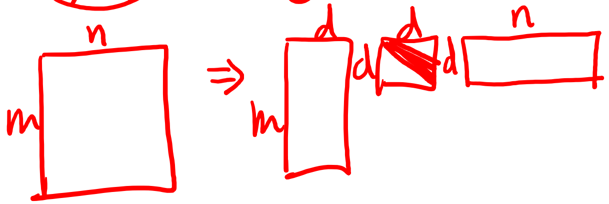
- Approximation methods

Singular Value Decomposition

$m \times n$

$m \neq n$

$\leftarrow A = \vec{U} \Sigma \vec{V}^T$



$$A = \begin{pmatrix} 3 & 2 \\ 2 & 3 \end{pmatrix} - \lambda I$$

$$\begin{vmatrix} 3-\lambda & 2 \\ 2 & 3-\lambda \end{vmatrix} = 0$$

$ad - bc$

$$(3-\lambda)^2 - 2^2 = 0$$

$$\lambda = 5, 1$$

Eigen value decomposition

$$Av = \lambda I v$$

$$v(A - \lambda I) = 0$$

$\to 0$

$V = \emptyset$

$$|A - \lambda I| = 0$$

$\lambda = 5$

$$\begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix} V_1 = 0 \qquad V_1 \propto \begin{pmatrix} 1 \\ 1 \end{pmatrix} \to V_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$\lambda = 1$

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} V_2 = 0 \to V_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$U = V$

$(\lambda_1, \lambda_2 \ldots)$

A : symmetric $\to V \Sigma V^T$ $\quad V^{+} = V^T$

! " $\quad A \to V \Sigma V^{-1}$ $\quad V \neq V^{-1}$

square $(n \times n)$ $\quad (m \times n)$

Singular Value Decomposition

Eigen : $A = V \Sigma V^T$ A : symmetric,

SVD : $A = U \Sigma V^T$

Power Iteration Method

init $v = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $v \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

for k

$v_{k+1} \leftarrow \dfrac{Av_k}{\|Av_k\|}$

$v_k \approx v_{k+1}$

$\downarrow$

v

$\rightarrow \underbrace{\lambda_1}_{\text{scalar}} = v^T A v^{(1)}$

$A^{(2)} = \boxed{A} - \lambda_1 \cdot \boxed{vv^T}$

$v^{(2)} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$

$v_{k+1}^{(2)} \leftarrow \dfrac{A^{(2)} v_k^{(2)}}{\|A^{(2)} v_k^{(2)}\|}$

$\downarrow$

$v^{(2)}$

λ_2

Singular Value Decomposition

$A = U \Sigma V^T$

X symmetric

X square shaped

Any matrix

→ Should not have null

generalization of eigenvalue decomposition

$A = U \Sigma V^T$

$A^T A = (U \Sigma V^T)^T U \Sigma V^T$

$= V \Sigma \underbrace{U^T U}_{\mathbb{1}} \Sigma V^T$

$V^T V \Rightarrow \mathbb{1}$

$$\boxed{A^T A = V \Sigma^2 V^T}$$

$\rightarrow V \quad \Sigma$

$$\boxed{A A^T = U \Sigma^2 U^T}$$

$\rightarrow U, \Sigma^2$

Singular Value Decomposition

$A =$

	M_1	2	3	4	5	6
U_1	1	2	3	4	3	3
2	1	2	3	4	3	3
3	3	3	3	1	4	5
4	3	3	3	1	4	5

$A \longrightarrow U \Sigma V^T$

```
u, sig, vt = np.linalg.svd(A3)
```

4,4

```
[[-0.44  0.55  0.71 -0.  ]
 [-0.44  0.55 -0.71  0.  ]
 [-0.55 -0.44 -0.   -0.71]
 [-0.55 -0.44  0.    0.71]]
```

4,1

```
[14.74  4.1   0.    0.  ]
```

λ_1 λ_2

6x6

```
[[-0.28 -0.34 -0.4  -0.32 -0.48 -0.55]
 [-0.38 -0.11  0.15  0.86 -0.06 -0.28]
 [-0.81  0.49 -0.06 -0.21  0.06  0.22]
 [ 0.05  0.15 -0.46  0.03  0.75 -0.45]
 [-0.33 -0.69  0.38 -0.25  0.45  0.07]
 [-0.06 -0.36 -0.67  0.24  0.02  0.6 ]]
```

```
Ap =np.matmul(u, np.matmul(np.diag(sig),vt[:4]))
np.around(Ap,3)
```

```
array([[1., 2., 3., 4., 3., 3.],
       [1., 2., 3., 4., 3., 3.],
       [3., 3., 3., 1., 4., 5.],
       [3., 3., 3., 1., 4., 5.]])
```

Singular Value Decomposition

Using np.linalg.svd

```
u, sig, vt = np.linalg.svd(A)
```

U

```
[[-0.44  0.55  0.71 -0.   ]
 [-0.44  0.55 -0.71  0.   ]
 [-0.55 -0.44 -0.   -0.71]
 [-0.55 -0.44  0.    0.71]]
```

Σ

```
[14.74  4.1   0.    0.  ]
```

V^T

```
[[-0.28 -0.34 -0.4  -0.32 -0.48 -0.55]
 [-0.38 -0.11  0.15  0.86 -0.06 -0.28]
 [-0.81  0.49 -0.06 -0.21  0.06  0.22]
 [ 0.05  0.15 -0.46  0.03  0.75 -0.45]
 [-0.33 -0.69  0.38 -0.25  0.45  0.07]
 [-0.06 -0.36 -0.67  0.24  0.02  0.6 ]]
```

Using sklearn.decomposition.TruncatedSVD

```
from sklearn.decomposition import TruncatedSVD
tsvd = TruncatedSVD(n_components=4).fit(A)
X = tsvd.transform(A)
U = np.matmul(A,np.matmul(tsvd.components_.T,np.diag(1/tsvd.singular_values_)))
Vt = tsvd.components_
S = tsvd.singular_values_
```

$A = U \Sigma V^T$

$A V \Sigma^{-1}$

$\lambda = 0$

U

```
[[ 4.40000000e-01  5.50000000e-01  2.25179981e+15  0.00000000e+00]
 [ 4.40000000e-01  5.50000000e-01  2.25179981e+15  0.00000000e+00]
 [ 5.50000000e-01 -4.40000000e-01  0.00000000e+00 -3.89422264e+33]
 [ 5.50000000e-01 -4.40000000e-01  0.00000000e+00 -3.89422264e+33]]
```

Σ

```
[14.74  4.1   0.    0.  ]
```

V^T

```
[[ 0.28  0.34  0.4   0.32  0.48  0.55]
 [-0.38 -0.11  0.15  0.86 -0.06 -0.28]
 [ 0.45 -0.86 -0.05  0.16  0.07  0.19]
 [-0.49 -0.35  0.64 -0.37  0.3  -0.04]]
```

Non-negative Matrix Factorization

$$\underset{\geq 0}{X} = \underset{\geq 0}{W}\,\underset{\geq 0}{H}$$

- Suitable for data with non-negative entries

- Solve optimization to get W and H

$$X \simeq WH$$

NMF design choices

- Latent dimension

  # genre
  # topic

  $d \ll m, n$

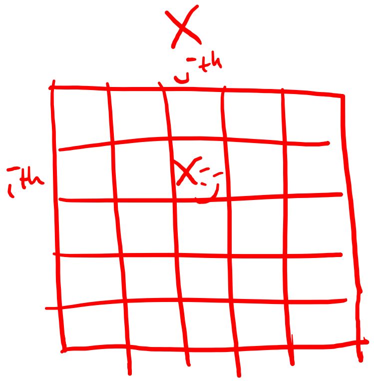
- Objective/Loss function
- Additional constraints

  $W^T W = \mathbb{1}$, $H^T W = \mathbb{1}$

  $X = WH$

- Regularization

Loss functions in NMF



$$\rightarrow W \times H$$

$$X_{ij} = W_i \times H_j + \varepsilon$$

$$X_{ij} \simeq W_i \times H_j$$

$X_{ij} \to$ r.V $\qquad \tilde{X}_{ij}$

Noise = error

$$\tilde{X} = WH + \varepsilon$$

$$\tilde{X} = WH \times \varepsilon$$

0.9 1 .01

$$\max \log \left(\prod_{ij} P(\tilde{X}_{ij} = W_i \cdot H_j) \right) \Longrightarrow \text{minimize } \underline{\text{loss}}$$

$$\prod P_{ij} \to \sum_{ij} \log P_{ij}$$

Loss functions in NMF

- Data entry as random variable $\tilde{X}_{ij}$
- Probability density and maximum likelihood

Loss functions in NMF

Loss functions in NMF

L2 Loss

$\tilde{X}_{ij} \sim N(\mu = W_i H_j, \sigma)$

$\log\left(\prod_{ij} P(\tilde{X}_{ij} = (WH)_{ij})\right) = \frac{1}{\sqrt{2\pi}\sigma}\left(e^{-\frac{1}{2\sigma^2}(X_{ij} - WH_{ij})^2}\right)$

$\log\left(\prod_{ij} \quad \right)$

$= \boxed{\sum_{ij} A\left(X_{ij} - (WH)_{ij}\right)^2 + C}$

$\Rightarrow$ minimize $\boxed{\sum_{ij}\left(X_{ij} - (WH)_{ij}\right)^2}$

L2

Loss functions in NMF

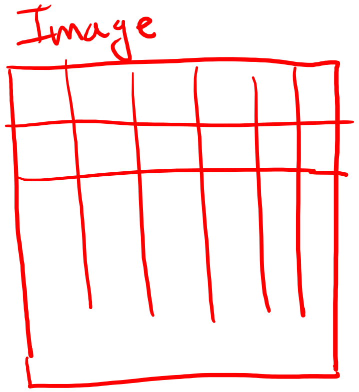
L1 Loss

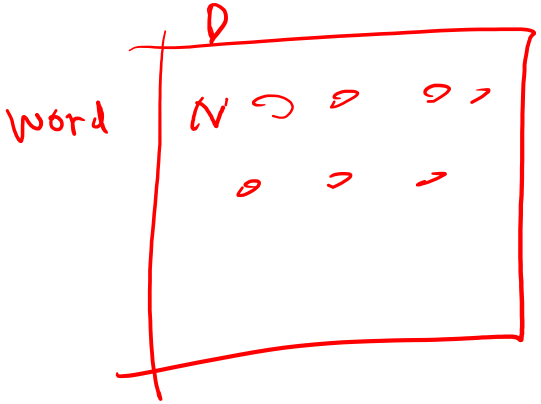
$\tilde{X}_{ij} \sim$ Laplace

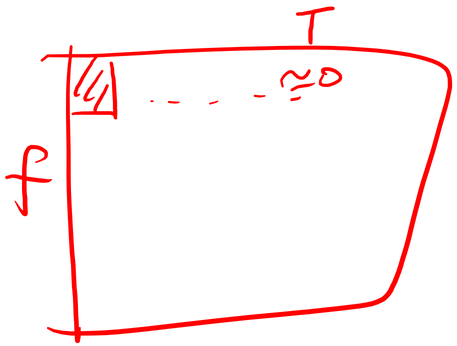
$$p = \frac{1}{2\sigma} e^{-\frac{1}{\sigma}\left|\tilde{X}_{ij} - (WH)_{ij}\right|}$$

$$\to \quad \min \sum_{ij} \left|X_{ij} - (WH)_{ij}\right|$$

$$\boxed{X_{ij} \neq 0} \pm \varepsilon \sim N(0,\sigma)$$







Loss functions in NMF

KL Loss

$\tilde{X}_{ij} \sim \text{Poisson}$

$$P(\tilde{X}_{ij} = (WH)_{ij}) = \frac{(WH)_{ij}^{k}}{k!} e^{-(WH)_{ij}}$$

$$\boxed{\mathcal{L}(X, WH) = \underbrace{\sum_{ij} X_{ij} \log \frac{X_{ij}}{(WH)_{ij}}}_{\text{KL divergence}} \underbrace{- X_{ij} + (WH)_{ij}}_{-\sum_{ij}}}$$

$$KL(P|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

$P(x) \rightsquigarrow \tilde{X}$

$Q(x) \rightarrow WH$

Loss functions in NMF

Itakura-Saito (IS) Loss

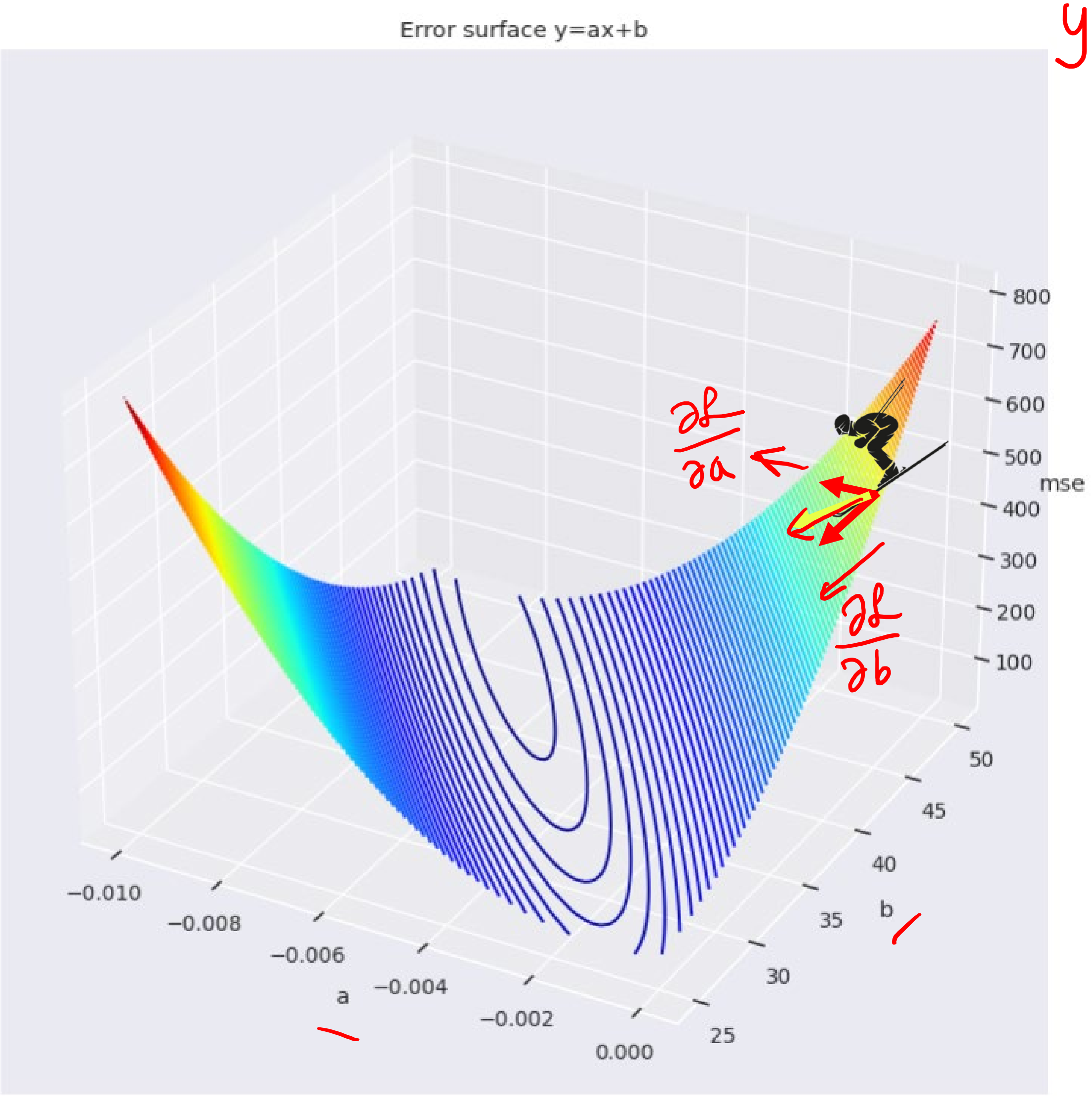
$$\tilde{X} = W \cdot H \cdot \underset{\sim}{N}$$

$$\mathcal{L}(X, WH) = \sum_{ij} \frac{X_{ij}}{(WH)_{ij}} - \log\left(\frac{X_{ij}}{(WH)_{ij}}\right) - 1$$

$$X \to \gamma X$$

$$(WH) \to \gamma (WH)$$

Optimization in NMF



$y \quad \hat{y} = (ax + b)$

Gradient Descent

Loss function

$$L = \frac{1}{2n}\sum_i^n (y_i - f(x_i))^2 = \frac{1}{2n}\sum_i^n (y_i - (ax_i + b))^2$$

$\Sigma\, \varepsilon^2$

Gradients

$$\nabla_a L = \frac{\partial L}{\partial a} = -\frac{1}{n}\sum_i^n (y_i - (ax_i + b))x_i$$

$$\nabla_b L = \frac{\partial L}{\partial b} = -\frac{1}{n}\sum_i^n (y_i - (ax_i + b))$$

Parameter(weight) update rule

$$a = a - \alpha \cdot \frac{\partial L}{\partial a}$$

$$\omega = \omega - \alpha \nabla_\omega L$$

$$b = b - \alpha \cdot \frac{\partial L}{\partial b}$$

Optimization in NMF

$$\underset{\sim}{L_2} = \underbrace{\sum_{ij} (X_{ij} - W_i H_j)^2}\ \underbrace{+ \underset{\sim}{\lambda} \|W\|_F^2 + \underset{\sim}{\lambda} \|H\|_F^2}$$

$$\frac{\partial L}{\partial W_i} = (W_i H_j - X_{ij}) H_j + \frac{\lambda}{n_i^W} W_i \qquad w \leftarrow w - \alpha \frac{\partial f}{\partial w}$$

$$\frac{\partial L}{\partial H_j} = (W_i H_j - X_{ij}) W_i + \frac{\lambda}{n_j^H} H_j$$

$$\boxed{\begin{aligned} W_i &\Leftarrow \left(1 - \frac{\lambda}{n_i^W} \cdot \alpha\right) W_i - (W_i H_j - X_{ij}) H_j \\ H_j &\Leftarrow \left(1 - \frac{\lambda}{n_j^H} \alpha\right) H_j - (W_i H_j - X_{ij}) W_i \end{aligned}}$$

Using NMF

sklearn.decomposition.NMF

L_2 KL IS

MU

class sklearn.decomposition.NMF(*n_components=None, *, init='warn', solver='cd', beta_loss='frobenius', tol=0.0001, max_iter=200, random_state=None, alpha='deprecated', alpha_W=0.0, alpha_H='same', l1_ratio=0.0, verbose=0, shuffle=False, regularization='deprecated'*) [source]

Non-Negative Matrix Factorization (NMF).

Find two non-negative matrices (W, H) whose product approximates the non- negative matrix X. This factorization can be used for example for dimensionality reduction, source separation or topic extraction.

The objective function is:

$$\begin{aligned}
& 0.5 * ||X - WH||_{loss}^2 \\
& + alpha_W * l1_{ratio} * n_features * ||vec(W)||_1 \\
& + alpha_H * l1_{ratio} * n_samples * ||vec(H)||_1 \\
& + 0.5 * alpha_W * (1 - l1_{ratio}) * n_features * ||W||_{Fro}^2 \\
& + 0.5 * alpha_H * (1 - l1_{ratio}) * n_samples * ||H||_{Fro}^2
\end{aligned}$$

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html