Matrix Factorization

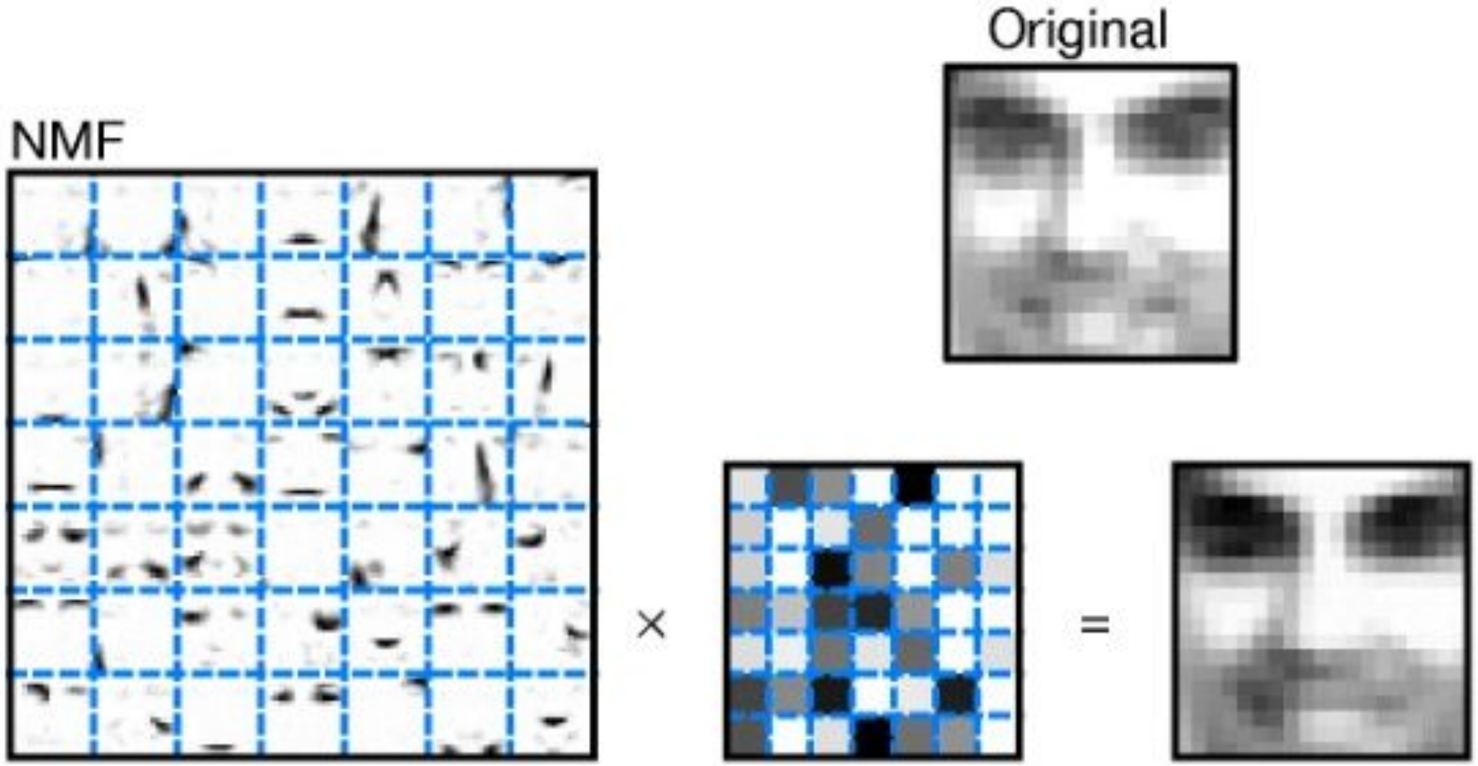
Motivation

Collaborative Filtering

- Neighborhood methods
- Latent factor models

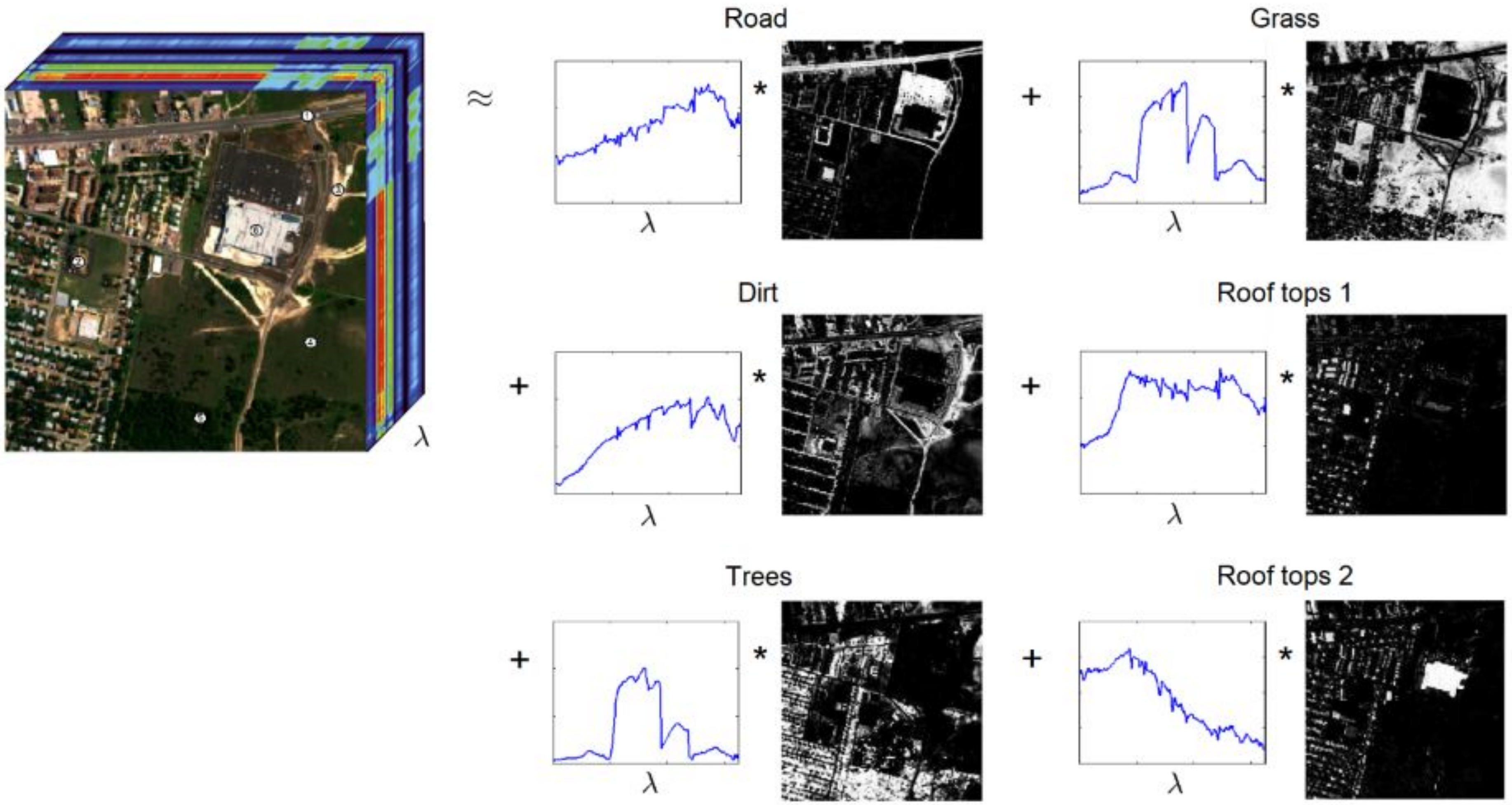
What is Matrix Factorization?

Applications of MF



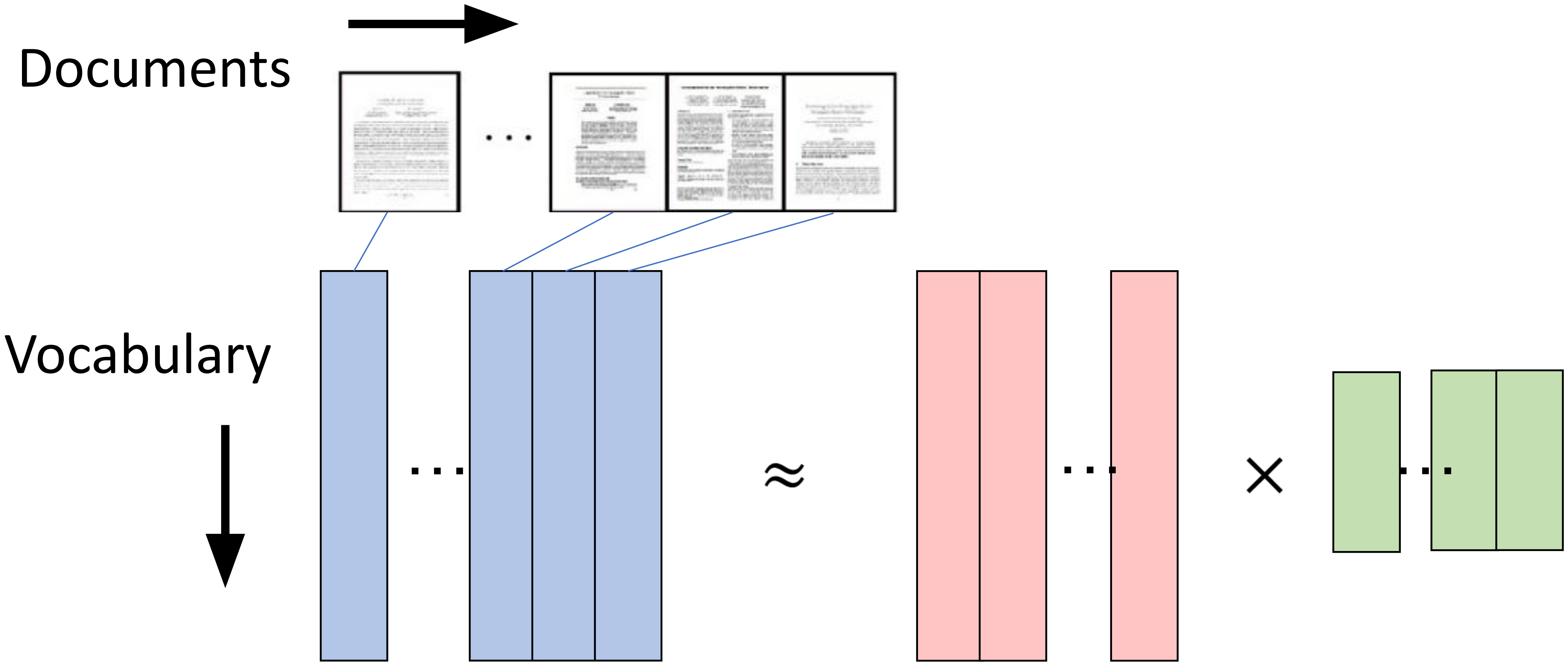
Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization.
Nature 401, 788–791 (1999)

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

Singular Value Decomposition

Singular Value Decomposition

Singular Value Decomposition

1	2	3	4		
1	2	3	4		
			1	4	5
			1	4	5

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

```
[[-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]]

[14.74  4.1   0.    0.  ]

[[-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)
```

```
[[-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]]

[14.74  4.1   0.    0.  ]

[[-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_
```

```
[[ 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]]

[14.74  4.1   0.    0.  ]

[[ 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

- Suitable for data with non-negative entries

- Solve optimization to get W and H

NMF design choices

- Latent dimension
- Objective/Loss function
- Additional constraints
- Regularization

Loss functions in NMF

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

Loss functions in NMF

L1 Loss

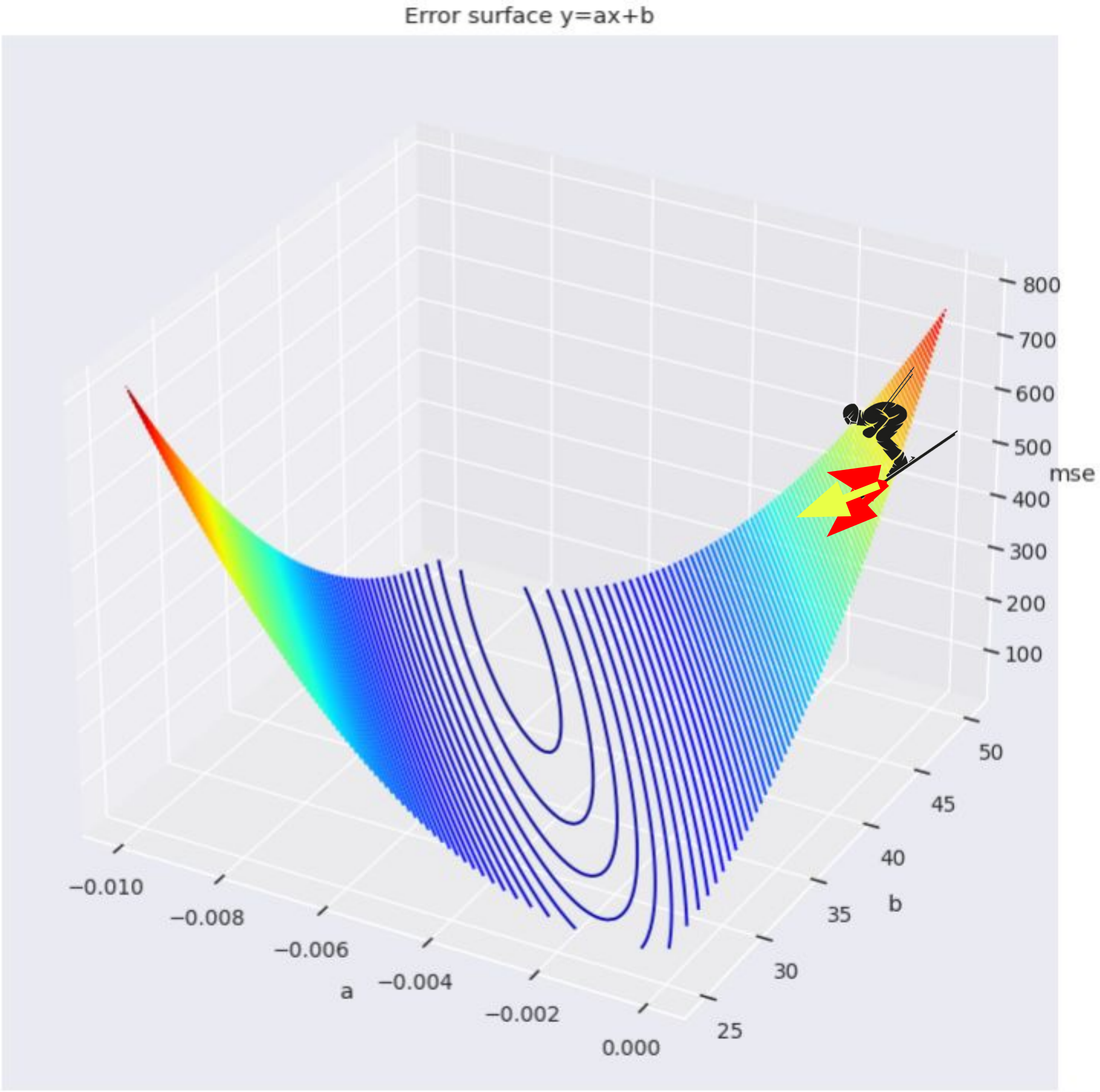
Loss functions in NMF

KL Loss

Loss functions in NMF

Itakura-Saito (IS) Loss

Optimization in NMF



Gradient Descent

Loss function

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

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

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

Optimization in NMF

Using NMF

sklearn.decomposition.NMF

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