Non-negative Matrix factorization

DSTA

1 Review of Spectral Analysis/SVD

Decompose the data matrix and interpret its 'first' eigenvalues as concepts/topics for user and activity classification:

$$M = U\Sigma V^T$$

. . .

$U_{(m\times r)}$ is column-orthonormal: $u_i \cdot u_j^T = 0$

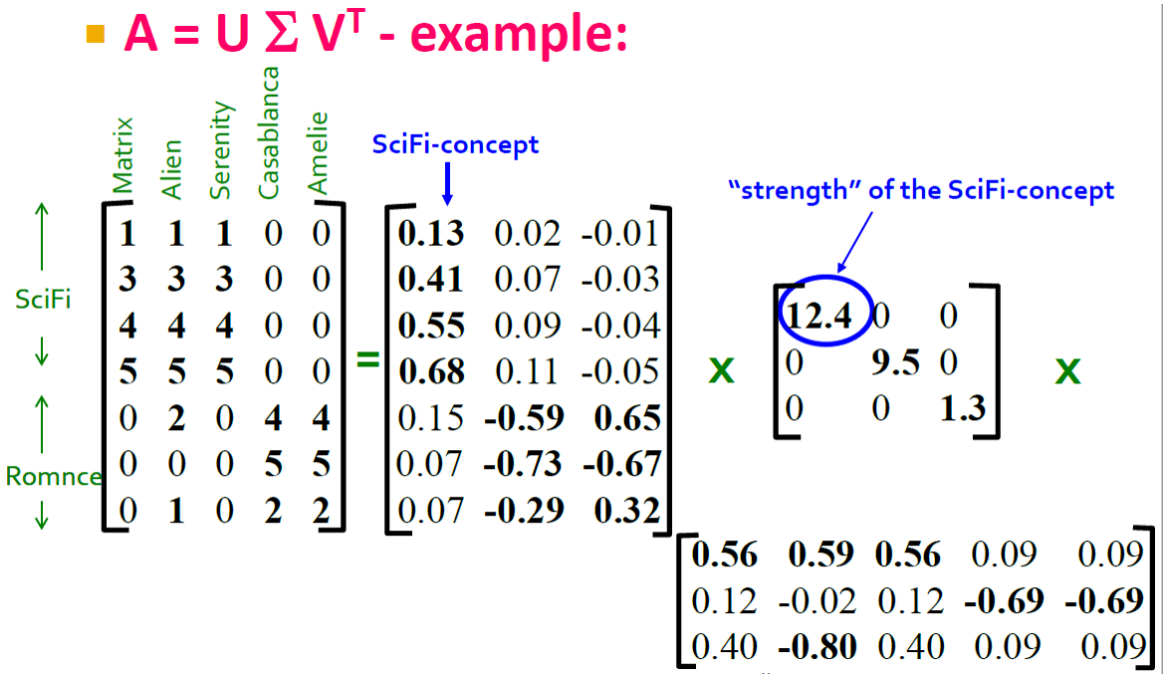
$V^T_{(r\times n)}$ is row-orthonormal: $v_i^T \cdot v_j = 0$

$\Sigma_{(r\times r)}$ is diagonal, σ_{ij} are the singular values

dimension $\boldsymbol{r}$ will depend on the no. of singular values found

---

1.1 Hurdle: interpretation of negative values



With negative values we cannot distinguish between i) lack of information, ii) lack of interest or ii) outright *repulsion.*

A non-negative decomposition of the activity matrix would be *interpretable:*

$$A_{(n\times d)} = P_{(n\times r)} \cdot Q_{(r\times d)}$$

- A: activity
- P: user participation to a topic
- Q: quantity of the topic in product

. . .

user/product profiling and reccommender sys. would be half-done already!

2 Non-negative decomposition

2.1 The numerical problem

Istance: a non-negative matrix V

Solution: non-negative matrix factors W and H s.t.

$$V \approx W \cdot H$$

with $w_{ij}, h_{rs} \geq 0$

2.2 Notation

$$A = B \cdot C$$

Let $\mathbf{a}_i$ be the i-th column of A. It can be expressed as

. . .

$$\mathbf{a}_i = B \cdot \mathbf{c}_i$$

each col. of the result is seen as a linear combination of the cols. of B, with $\mathbf{c}_i$ supplying the *weights:*

. . .

$$\mathbf{a}_i = B \cdot \mathbf{c}_i = c_{1,i}\mathbf{b}_1 + c_{2,i}\mathbf{b}_2 + \cdots + c_{n,i}\mathbf{b}_n$$

2.3 Interpretation

Let $\mathbf{v}_i$ be the i-th column of V.

If V is an activity m., $\mathbf{v}_i$ represent the *consumption* of i

. . .

$$v_i \approx W \cdot h_i$$

Consumption of i is given by a linear combination of the cols. of W, with h_i supplying the weights.

Each $\mathbf{w}_i$ is interpretable as a pattern (or mask)

[Lee & Seung, Nature, 1999]: "Learning the parts of objects by non-negative matrix factorization."

$$\mathbf{v}_i \approx \mathbf{w}_1 \cdot h_{1,i} + \ldots \mathbf{w}_r \cdot h_{1,r}$$

W can be regarded as containing a basis that is optimized for the linear approximation of the data in V.

. . .

Since relatively few basis vectors are used to represent many data vectors, good approximation can only be achieved if the basis vectors discover structure that is latent in the data.

2.4 Norm notation

Frobenius' element-wise norm: $||A_{m\times n}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{\sum_{i,j} a_{ij}^2}$

. . .

Notation for error:

$||X - Y||_F^2 = ||X - Y||^2 = \sum_{i,j}(x_{ij} - y_{ij})^2$

3 NMF as error-minimization

3.1 Computational problem

Input: $V_{n\times m}$

Minimize $||V - WH||^2$

subject to $W, H \geq 0$.

. . .

- choose the new dimension r s.t. $(n+m)r < nm$;
- calculate $W_{n\times r}$ and $H_{r\times m}$.

3.2 Information-theoretic view

If the input matrix can be (somehow) normalised then we see the search for the perfect non-negative decomposition in terms of minimizing *divergence:*

$D_I(X||Y) = \Sigma_{i,j}(x_{ij} \cdot \log(\frac{x_{ij}}{y_{ij}}) - x_{ij} + y_{ij}))$

. . .

Minimize $D_I(V||WH)$

subject to $W, H \geq 0$.

Recommended version for sparse counting data.

The Kullback-Leibler divergence, D_{KL}, may also be used.

3.3 Gradient descent KO

> Although [error func.] are convex in W only or H only, they are not convex in both variables together.
>
> Therefore it is unrealistic to expect an algorithm to solve [the problem] in the sense of finding global minima.

> However, there are many techniques from numerical optimization for finding local minima.
>
> Gradient descent is perhaps the simplest technique to implement, but convergence can be slow.

4 Lee-Seung's Method

4.1 Iterated error balancing

1. start from random W and H
2. compute the error
3. update W and H with the **multiplicative update** rule:

$$W_{ia} \leftarrow W_{ia} \sum_{\mu} \frac{V_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}}$$

$$H_{a\mu} \leftarrow H_{a\mu} \sum_i W_{ia} \frac{V_{i\mu}}{(WH)_{i\mu}}$$

4.2 Multiplicative update

Classical Gradient descent: we *move around* by adding/subtracting some quantity

NMF: we *move around* by multiplying by a *local* error measure

$\frac{v_{i\mu}}{(wh)_{i\mu}}$

. . .

$$W_{ia} \leftarrow W_{ia} \sum_{\mu} \frac{V_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}}$$

$$H_{a\mu} \leftarrow H_{a\mu} \sum_i W_{ia} \frac{V_{i\mu}}{(WH)_{i\mu}}$$

$$W_{ia} \leftarrow W_{ia} \sum_{\mu} \frac{V_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$$

$$W_{ia} \leftarrow \frac{W_{ia}}{\sum_j W_{ja}}$$

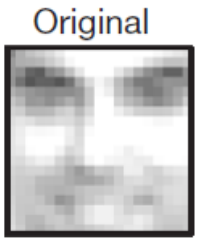
$$H_{a\mu} \leftarrow H_{a\mu} \sum_i W_{ia} \frac{V_{i\mu}}{(WH)_{i\mu}}$$

- through iteration, the $\frac{v_{i\mu}}{(wh)_{i\mu}}$ factors vanish and we stop.

- the update rules maintain non-negativity and force $\mathbf{w}_i$ to sum to 1.

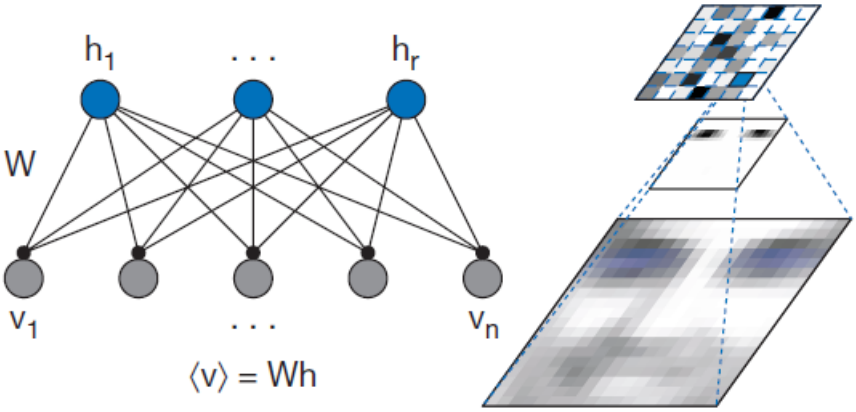
5 Interpretability of NMF

5.1 The 19x19 mugshots



5.2

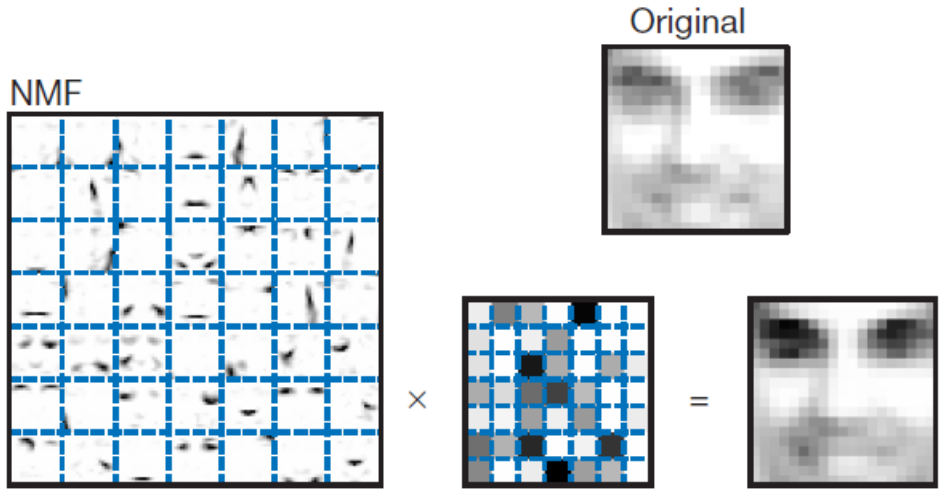
A probabilistic hidden-variables model:



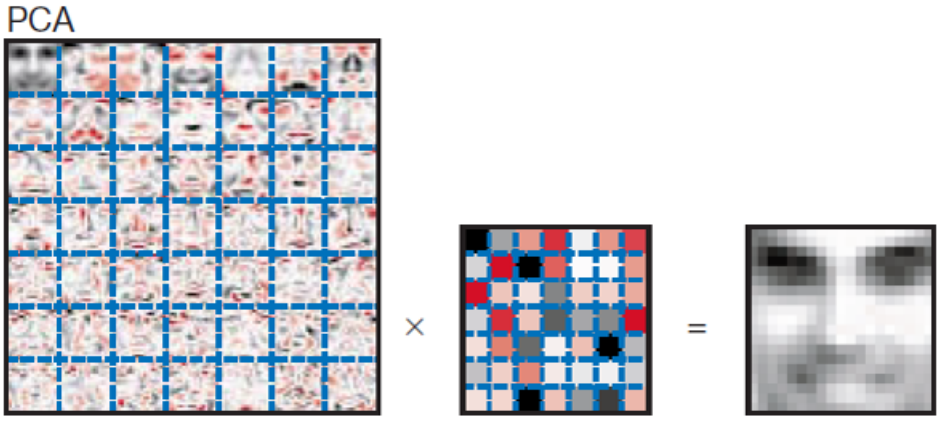
Cols. of W are *bases* that are combined to form the rec.

The influence of $\mathbf{h}_a$ on $\mathbf{v}_i$ is represented by a connection **of strength** w_{ia}

5.3 W and H in a 7x7 montage



The *eigenfaces* might have negative values



5.4 The faces on Scikit-learn

Check a visual comparison of the methods on `64x64=4096` mugshots: 40 classes for 400 samples.

```
from sklearn.datasets import fetch_olivetti_faces

olivetti_faces = fetch_olivetti_faces()

olivetti_faces.images.shape
```

```
(400, 64, 64)
```

```
# ...
from sklearn import decomposition

from numpy.random import RandomState

faces, _ = fetch_olivetti_faces()
```

```
nmf_estimator = decomposition.NMF(solver='mu')

nmf_estimator.fit(faces)
```

6 Activity-matrix decomposition

6.1 A simple ratings matrix

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Figure 11.6: Ratings of movies by users

N=7, M=5.

Fix K=2 and run NMF:

```
@INPUT:
    R: a m. to be factorized, dim. N x M
    P: an initial m. of dim. N x K
    Q: an initial m. of dim. M x K
    K: the no. of latent features
```

```
    steps: the max no. of steps to perform the optimisation
    alpha: the learning rate
    beta: the regularization parameter

@OUTPUT:
    the final matrices P and Q
```

6.2 Direct implementation (1 run)

```
nP=
[[ 0.33104196  0.39332058]
 [ 1.08079793  1.08397306]
 [ 1.59267325  1.27929568]
 [ 1.87852789  1.72209575]
 [ 0.67146598  1.76523621]
 [ 1.04872774  2.10824903]
 [ 0.94419145  0.59698619]]
```

```
nQ.T=
[[ 1.27381876  1.3870236   1.67315614  0.9855609   0.81578369]
 [ 1.50953822  1.38352352  1.06501557  1.87281749  1.96189735]]
```

6.3 Analysis of the error, I

```
np.dot(nP, nQ.T) = [
        [ 1.01541991  1.00333129  0.97277743  1.06287968  1.04171324]
        [ 3.01303945  2.99879446  2.96279189  3.09527589  3.0083412 ]
        [ 3.95992279  3.97901104  4.02726085  3.9655638   3.80912366]
        [ 4.99247343  4.98812249  4.97712927  5.07657468  4.91104751]
        [ 3.52001748  3.37358497  3.00347147  3.96773585  4.01098322]
        [ 4.51837154  4.37142223  4.00000329  4.98195069  4.99170315]
        [ 2.10390225  2.13556026  2.21557931  2.04860435  1.94148161]
        ]
```

```
ratings =  [[1, 1, 1, 0, 0],
            [3, 3, 3, 0, 0],
            [4, 4, 4, 0, 0],
            [5, 5, 5, 0, 0],
            [0, 0, 0, 4, 4],
            [0, 0, 0, 5, 5],
            [0, 0, 0, 2, 2]
           ]
```

6.4 Analysis of the error, II

```
np.rint(np.dot(nP, nQ.T))= [
      [ 1.  1.  1.  1.  1.]
            [ 3.  3.  3.  3.  3.]
            [ 4.  4.  4.  4.  4.]
            [ 5.  5.  5.  5.  5.]
            [ 4.  4.  4.  4.  4.]
            [ 5.  5.  5.  5.  5.]
            [ 2.  2.  2.  2.  2.]
      ]
```

```
ratings =  [[1, 1, 1, 0, 0],
            [3, 3, 3, 0, 0],
            [4, 4, 4, 0, 0],
            [5, 5, 5, 0, 0],
            [0, 0, 0, 4, 4],
            [0, 0, 0, 5, 5],
            [0, 0, 0, 2, 2]]
```

try Scikit-learn on the same instance (right-click to save a local copy).

6.5 Analysis of the result

```
W(user x topic) = [
    [ 0.          0.82037571]
    [ 0.          2.46112713]
    [ 0.          3.28150284]
    [ 0.          4.10187855]
    [ 1.62445593  0.        ]
    [ 2.03056992  0.        ]
    [ 0.81222797  0.        ]
    ]
```

```
H(topic x film) =
[[ 0.          0.          0.          2.46236289  2.46236289]
 [ 1.21895369  1.21895369  1.21895369  0.          0.        ]]
```

. . .

W: users' committment to a topic.

H: films' pertinence to a specific topic (binary, why?)