COMP7630 – Web Intelligence and its Applications

Singular Value Decomposition

Valentino Santucci

(valentino.santucci@unistrapg.it)

Singular Value Decomposition (SVD)

- Eigendecomposition does not work with rectangular matrices (and also with non-diagonalizable matrices)

- SVD generalizes EigenDecomposition on rectangular matrices (to be precise: on any kind of matrix)

- It is also possible to describe PCA by using SVD, though we will not do it.
    - The trick is to work directly with data-table X and not with X^TX as seen before.
    - ScikitLearn uses the SVD implementation, though it is mathematically equivalent to what we have seen!

Singular Value Decomposition (SVD)

- Another way to factorize a matrix into singular vectors and single values.
- Every real matrix (even not a square) has a SVD.
- Singular value decomposition

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T$$

- $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ $\quad \boldsymbol{U} \in \mathbb{R}^{m \times m}$ $\quad \boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ $\quad \boldsymbol{V} \in \mathbb{R}^{n \times n}$
- $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices.
- $\boldsymbol{\Sigma}$ is diagonal (but not necessarily square) and the elements along the diagonal are the singular values.
- $\boldsymbol{U}$ is left-singular vector and $\boldsymbol{V}$ is right-singular vector.

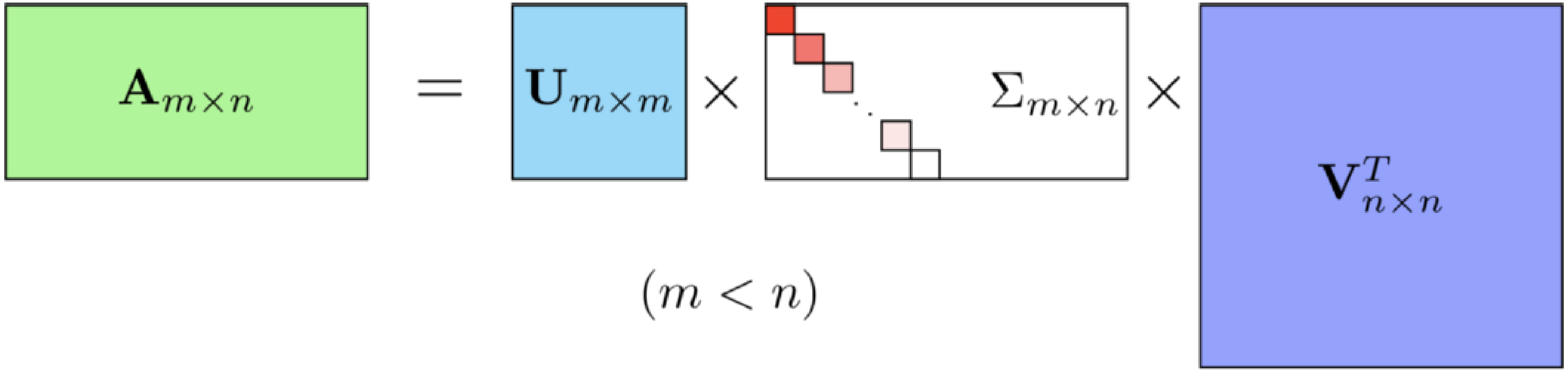
Some properties of the SVD

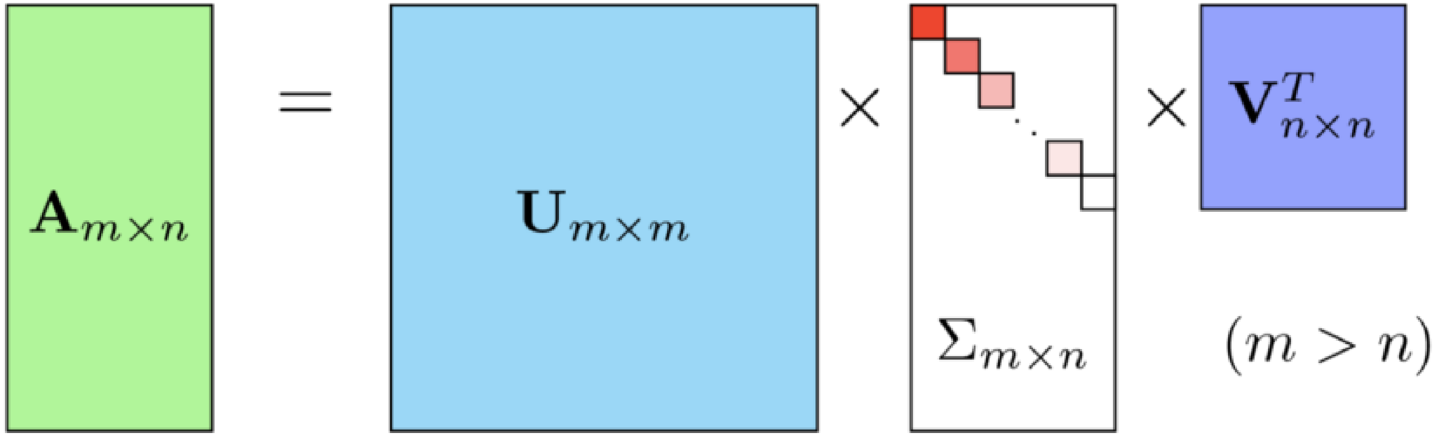
- U columns are the eigenvectors of A^TA (which is symmetric by definition)
- They are called "left singular vectors" of A

- V^T rows are the eigenvectors of AA^T
- They are called "right singular vectors" of A

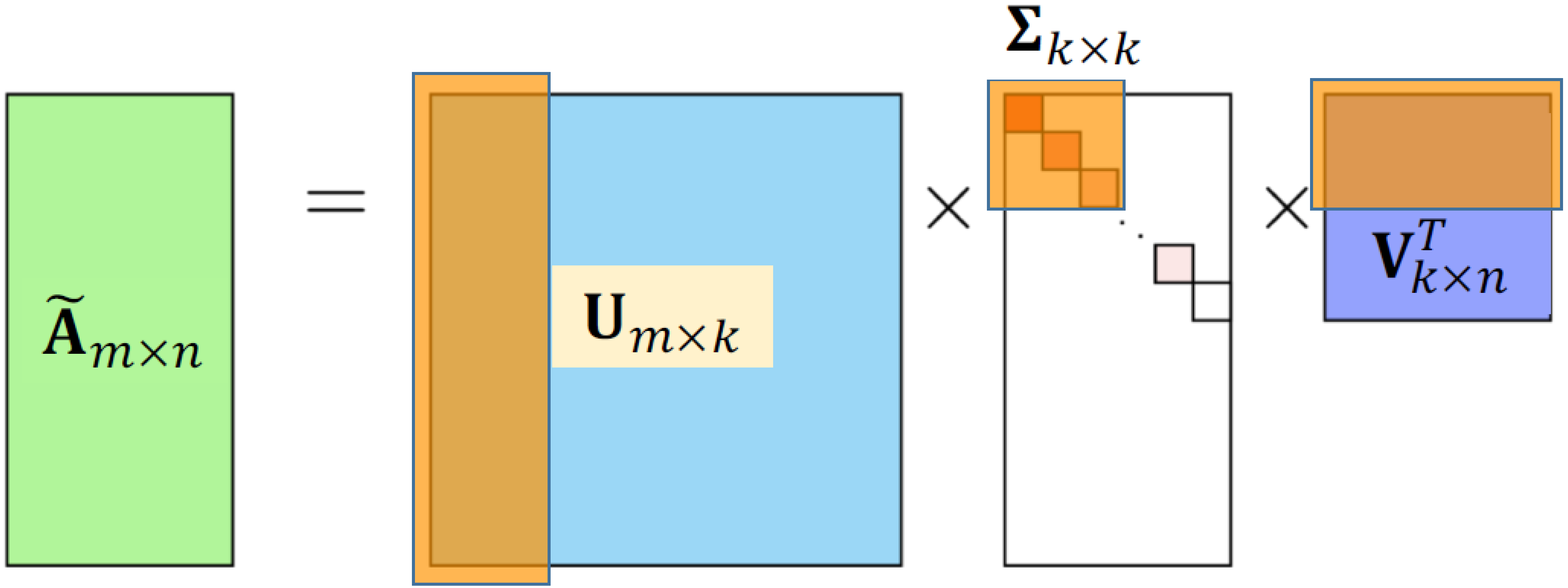
- The non-zero diagonal values in Σ are the square-root of the eigenvalues of both A^TA and AA^T (which are equal)
- They are called "singular values" of A

Singular Value Decomposition (SVD)





Truncated SVD



$$\boldsymbol{A} \approx \widetilde{\boldsymbol{A}}_{\boldsymbol{k}} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k{}^T$$

Truncated SVD

- Keep only the k largest singular values.

$$\boldsymbol{A} \approx \widetilde{\boldsymbol{A}}_{\boldsymbol{k}} = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k{}^T$$

- $\widetilde{\boldsymbol{A}}_{\boldsymbol{k}} \in \mathbb{R}^{m \times n}$ $\boldsymbol{U}_k \in \mathbb{R}^{m \times k}$ $\boldsymbol{\Sigma}_{\boldsymbol{k}} \in \mathbb{R}^{k \times k}$ $\boldsymbol{V}_k \in \mathbb{R}^{k \times n}$

- It can be shown that this gives the minimum value for the Frobenius norm of $\left\|\boldsymbol{A} - \widetilde{\boldsymbol{A}}_k\right\|_F$

Truncated SVD

- It is important to note that:
  - We reduce the number of columns in the "left-matrix" U
  - We reduce the number of rows in the "right-matrix" V
  - But $\tilde{A}$ has the same shape of A

- Anyway, $\tilde{A}$ has a smaller rank than A

(recall: the rank of a matrix is the maximum number of columns which are linearly independent to each other)

- Why this may be useful?
  - A can be interpreted as a data-matrix containing noise
  - $\tilde{A}$ is a denoised version of A
  - U_k rows are denoised/reduced representations of A rows
  - $V^T{}_k$ columns are denoised/reduced representations of A columns

Another persepective on SVD

$$U\Sigma V^T x = U\Sigma \begin{bmatrix} v_1^T x \\ \vdots \\ v_m^T x \end{bmatrix} = U \begin{bmatrix} \sigma_1 v_1^T x \\ \vdots \\ \sigma_m v_m^T x \end{bmatrix} = \sum_k u_k \sigma_k v_k^T x = \sum_k \sigma_k u_k v_k^T x.$$

Hence $U\Sigma V^T = \sum_k \sigma_k u_k v_k^T$.

- The SVD decomposition can be rewritten as a sum of rank-1 matrices: those obtained by the outer product between the *k*-th column of *U* and the *k*-th row of V^T, weighted by *k*-th singular value.
  - Recall: the outer product of two vectors returns a rank-1 matrix) and, obviously, multiplying by a scalar does not change the rank of a matrix.

- So, the Truncated SVD acts as removing the terms with smaller weights in the summation!!!
  - (If you are familiar with it, it is a sort of "discrete Fourier transform")

SVD in Python

```
In [7]: import numpy as np

In [8]: from scipy.linalg import svd

In [9]: X = np.array([ [1,2,1,2],
   ...:                [0,1,0,1],
   ...:                [1,0,1,0],
   ...:                [1,2,3,4] ])

In [10]: #SVD: X = U*s*V^T

In [11]: U, s, VT = svd(X)

In [12]: U
Out[12]:
array([[-0.47547615, -0.35956946,  0.69135054,  0.40824829],
       [-0.18137369, -0.54296386,  0.0750144 , -0.81649658],
       [-0.11272877,  0.72635827,  0.54132174, -0.40824829],
       [-0.85341563,  0.21978106, -0.47262887,  0.        ]])

In [13]: s
Out[13]: array([6.38105353e+00, 1.49005261e+00, 1.03048489e+00, 3.21362686e-16])

In [14]: VT
Out[14]:
array([[-0.22592203, -0.4449355 , -0.49340627, -0.71241974],
       [ 0.39365716, -0.55202122,  0.68865488, -0.2570235 ],
       [ 0.737559  ,  0.49729767, -0.17973513, -0.41999646],
       [-0.5       ,  0.5       ,  0.5       , -0.5       ]])

In [15]: X_recovered = U.dot(np.diag(s)).dot(VT)

In [16]: X_recovered
Out[16]:
array([[ 1.00000000e+00,  2.00000000e+00,  1.00000000e+00,
         2.00000000e+00],
       [-3.48525357e-17,  1.00000000e+00, -2.02744139e-16,
         1.00000000e+00],
       [ 1.00000000e+00, -1.87190742e-15,  1.00000000e+00,
        -1.48282707e-16],
       [ 1.00000000e+00,  2.00000000e+00,  3.00000000e+00,
         4.00000000e+00]])
```

Truncated SVD in Python

```
In [21]: for k in [1,2,3]:
    ...:     X_recovered = U[:,:k].dot(np.diag(s[:k])).dot(VT[:k,:])
    ...:     norm = np.linalg.norm(X-X_recovered)
    ...:     print('---')
    ...:     print(f'X{k} = \n{X_recovered}')
    ...:     print(f'norm(X,X{k}) = {norm}')
    ...:
---
X1 =
[[0.69 1.35 1.5  2.16]
 [0.26 0.51 0.57 0.82]
 [0.16 0.32 0.35 0.51]
 [1.23 2.42 2.69 3.88]]
norm(X,X1) = 1.811672121361281
---
X2 =
[[ 0.47  1.65  1.13  2.3 ]
 [-0.06  0.96  0.01  1.03]
 [ 0.59 -0.28  1.1   0.23]
 [ 1.36  2.24  2.91  3.8 ]]
norm(X,X2) = 1.0304848877206008
---
X3 =
[[ 1.00e+00  2.00e+00  1.00e+00  2.00e+00]
 [-1.66e-16  1.00e+00 -7.15e-17  1.00e+00]
 [ 1.00e+00 -1.81e-15  1.00e+00 -2.14e-16]
 [ 1.00e+00  2.00e+00  3.00e+00  4.00e+00]]
norm(X,X3) = 3.670575228292439e-15
```