Exercise

11

TUM Department of Informatics

Supervised by Prof. Dr. Stephan Günnemann
Informatics 3 - Professorship of Data Mining and Analytics

Submitted by Marcel Bruckner (03674122)
Julian Hohenadel (03673879)
Kevin Bein (03707775)

Submission date Munich, January 19, 2020

Dimensionality Reduction and Matrix Factorization

Problem 1:

Leslie: Votes 3 for Alien, 4 for Titanic.

Originial space: $[0, 3, 0, 0, 4]$

To obtain the wanted concept space a projection from original space to concept space is needed.

(script page 67)

$$P = A * V$$

$$P = [0, 4, 0, 0, 4] * \begin{bmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{bmatrix}$$

$$P = [1.74, 2.84]$$

This means Leslie will most likely favor the Titanic and Casablanca movies (score: 2.84) over the sci-fi genre (score: 1.74). She already rated Titanic, which means she has already seen it, so Casablanca would be a good recommendation.

Problem 2:

Integrating out z (script page 31):

$x_i \sim \mathcal{N}(\mu, WW^T + \sigma^2 I)$

x has a gaussian distribution with mean μ and covariance $WW^T + \sigma^2 I = WW^T + \Phi$.

$y = Ax$ is a linear transformation $\implies$ y has still a gaussian distribution.

y now has a mean of $A\mu$ and a covariance of $AWW^TA^T + A\Phi A^T$.

(Because the covariance after a linear transformation with A is $cov(AZ) = Acov(Z)A^T$.)

If μ_{ML}, W_{ML}, Φ_{ML} represent the max. likelihood solution before the transformation, $A\mu_{ML}$, AW_{ML}, $A\Phi_{ML}A^T$ will represent the max. likelihood solution after a transformation with A.

A is orthogonal $\implies$ $AA^T = I$, which means $A\Phi A^T = A\sigma^2 I A^T = \sigma^2 I AA^T = \sigma^2 II = \sigma^2 I^2 = \sigma^2 I$

Which means the form of the model is preserved.

Problem 3:

a)

Transformation is only scaling done with the identity.

$\implies$ 70 % is kept.

b)

R is an orthogonal matrix with only applies rotation, but doesn't change the form of the model.

$\implies$ 70 % is kept.

c)

This combines a) and b). Everything is scaled by 5 and some dimensions are mirrored at the oirigin.

$\implies$ 70 % is kept.

d)

Scaling is not uniform and therefore we cannot tell without additional information.

$\implies$ 70 % is kept.

e)

Adding the mean μ to every entry only shifts the points but PCA centers them again, so this has no effect.

$\implies$ 70 % is kept.

f)

$rank(A) = 5 \leq rank(Y_6) \leq K = 5$ and therefore all information is captured by the principal components.

$\implies$ 100 % is kept.

Problem 4:

a)

$$X = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 1 & -2 \\ 4 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}$$

$$\implies \text{mean vector} = \frac{1}{N}[4+2+4-2, 3+1-1+1, 2-2+2+2] = [2,1,1]$$

$$\implies \widetilde{X} = X - \text{mean} = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & -3 \\ 2 & -2 & 1 \\ -4 & 0 & 1 \end{bmatrix}$$

$$Var(X_1) = \frac{1}{4}\sum_{i=1}^{4}(x_{i1} - \bar{x_1})^2 = \frac{1}{4}(4+0+4+16) = 6$$

$$Var(X_2) = \frac{1}{4}\sum_{i=1}^{4}(x_{i2} - \bar{x_2})^2 = \frac{1}{4}(4+0+4+0) = 2$$

$$Var(X_3) = \frac{1}{4}\sum_{i=1}^{4}(x_{i3} - \bar{x_3})^2 = \frac{1}{4}(1+9+1+1) = 3$$

$$Cov(X_1X_2) = \frac{1}{4}\sum_{i=1}^{4}(x_{i1} - \bar{x_1})(x_{i2} - \bar{x_2}) = \frac{1}{4}(4+0-4+0) = 0$$

$$Cov(X_2X_3) = \frac{1}{4}\sum_{i=1}^{4}(x_{i2} - \bar{x_2})(x_{i3} - \bar{x_3}) = \frac{1}{4}(2+0-2+0) = 0$$

$$Cov(X_3X_1) = \frac{1}{4}\sum_{i=1}^{4}(x_{i3} - \bar{x_3})(x_{i1} - \bar{x_1}) = \frac{1}{4}(2+0+2-4) = 0$$

$$\Sigma_X = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Eigendecomposition: $\Sigma_X = \Gamma\Lambda\Gamma^T$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y = \widetilde{X} * \Gamma = \widetilde{X}$$

b)

Truncate Γ: 2 is the lowest eigenvalue so it will get "dropped".

$6 + 2 + 3 = 11 \implies \frac{11-2}{11} = \frac{9}{11}$ of the variance is preserved.

This means the corresponding eigenvector $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ will be dropped from Γ.

$$Y = \widetilde{X} * \Gamma$$

$$Y = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & -3 \\ 2 & -2 & 1 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 2 & 1 \\ 0 & -3 \\ 2 & 1 \\ -4 & 1 \end{bmatrix}$$

c)

x_5 needs to be the mean $[2, 1, 1]$.

Then $\widetilde{X} = \begin{bmatrix} 2 & 2 & 1 \\ 0 & 0 & -3 \\ 2 & -2 & 1 \\ -4 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

The variance changes only by a scaling factor because now you normalize with $\frac{1}{5}$ instead of $\frac{1}{4}$.

The covariance still yields 0 for each dimension pairing.

This means that again the y axis will get "dropped" by the truncation.

$$Y = \begin{bmatrix} 2 & 1 \\ 0 & -3 \\ 2 & 1 \\ -4 & 1 \\ 0 & 0 \end{bmatrix}$$

Problem 5:

exercise_11_notebook

January 16, 2020

1 Programming task 11: Dimensionality Reduction

[0]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

1.1 Exporting the results to PDF

Once you complete the assignments, export the entire notebook as PDF and attach it to your homework solutions. The best way of doing that is 1. Run all the cells of the notebook. 2. Export/download the notebook as PDF (File -> Download as -> PDF via LaTeX (.pdf)). 3. Concatenate your solutions for other tasks with the output of Step 2. On a Linux machine you can simply use `pdfunite`, there are similar tools for other platforms too. You can only upload a single PDF file to Moodle.

Make sure you are using `nbconvert` Version 5.5 or later by running `jupyter nbconvert --version`. Older versions clip lines that exceed page width, which makes your code harder to grade.

1.2 PCA

Given the data in the matrix X your tasks is to: * Calculate the covariance matrix Σ. * Calculate eigenvalues and eigenvectors of Σ. * Plot the original data X and the eigenvectors to a single diagram. What do you observe? Which eigenvector corresponds to the smallest eigenvalue? * Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace. * Transform all vectors in X in this new subspace by expressing all vectors in X in this new basis.

1.2.1 The given data X

[0]:
```
X = np.array([(-3,-2),(-2,-1),(-1,0),(0,1),
              (1,2),(2,3),(-2,-2),(-1,-1),
              (0,0),(1,1),(2,2), (-2,-3),
              (-1,-2),(0,-1),(1,0), (2,1),(3,2)])
```

1.2.2 Task 1: Calculate the covariance matrix Σ

[3]:
```
def get_covariance(X):
    """Calculates the covariance matrix of the input data.

    Parameters
    ----------
    X : array, shape [N, D]
        Data matrix.

    Returns
    -------
    Sigma : array, shape [D, D]
        Covariance matrix

    """
    return np.cov(X, rowvar=False)

print("X before transformation: ")
print(X.shape)
print(X)
print()

print("Covariance matrix: ")
print(get_covariance(X))
```

```
X before transformation:
(17, 2)
[[-3 -2]
 [-2 -1]
 [-1  0]
 [ 0  1]
 [ 1  2]
 [ 2  3]
 [-2 -2]
 [-1 -1]
 [ 0  0]
 [ 1  1]
 [ 2  2]
 [-2 -3]
 [-1 -2]
 [ 0 -1]
 [ 1  0]
 [ 2  1]
 [ 3  2]]

Covariance matrix:
[[3.    2.625]
```

```
 [2.625 3.   ]]
```

1.2.3 Task 2: Calculate eigenvalues and eigenvectors of Σ.

[4]:
```python
def get_eigen(S):
    """Calculates the eigenvalues and eigenvectors of the input matrix.

    Parameters
    ----------
    S : array, shape [D, D]
        Square symmetric positive definite matrix.

    Returns
    -------
    L : array, shape [D]
        Eigenvalues of S
    U : array, shape [D, D]
        Eigenvectors of S

    """
    return np.linalg.eig(S)
print("X before transformation: ")
print(X.shape)
print(X)
print()

print("Eigenvalues and Eigenvectors: ")
print(get_eigen(get_covariance(X)))
```

```
X before transformation:
(17, 2)
[[-3 -2]
 [-2 -1]
 [-1  0]
 [ 0  1]
 [ 1  2]
 [ 2  3]
 [-2 -2]
 [-1 -1]
 [ 0  0]
 [ 1  1]
 [ 2  2]
 [-2 -3]
 [-1 -2]
 [ 0 -1]
 [ 1  0]
 [ 2  1]
 [ 3  2]]
```

```
Eigenvalues and Eigenvectors:
(array([5.625, 0.375]), array([[ 0.70710678, -0.70710678],
       [ 0.70710678,  0.70710678]]))
```

1.2.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

Note that, in general if u_i is an eigenvector of the matrix M with eigenvalue λ_i then $\alpha \cdot u_i$ is also an eigenvector of M with the same eigenvalue λ_i, where α is an arbitrary scalar (including $\alpha = -1$).

Thus, the signs of the eigenvectors are arbitrary, and you can flip them without changing the meaning of the result. Only their direction matters. The particular result depends on the algorithm used to find them.

[5]:
```
# plot the original data
plt.scatter(X[:, 0], X[:, 1])

# plot the mean of the data
mean_d1, mean_d2 = X.mean(0)
plt.plot(mean_d1, mean_d2, 'o', markersize=10, color='red', alpha=0.5)

# calculate the covariance matrix
Sigma = get_covariance(X)
# calculate the eigenvector and eigenvalues of Sigma
L, U = get_eigen(Sigma)

print("L: ")
print(L)
print()

print("U: ")
print(U)
print()

print("U L U^{T}")
print((U.dot(np.diag(L)).dot(U.T)))
print()

plt.arrow(mean_d1, mean_d2, U[0, 0], U[1, 0], width=0.01, color='red', alpha=0.
 ↪5)
plt.arrow(mean_d1, mean_d2, U[0, 1], U[1, 1], width=0.01, color='red', alpha=0.
 ↪5);
```

```
L:
[5.625 0.375]

U:
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
```

```
U L U^{T}
[[3.    2.625]
 [2.625 3.   ]]
```

What do you observe in the above plot? Which eigenvector corresponds to the smallest eigenvalue?

Write your answer here:

[0.70710678, -0.70710678]

1.2.5 Task 4: Transform the data

Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace. Transform all vectors in X in this new subspace by expressing all vectors in X in this new basis.

[0]:
```
def transform(X, U, L):
    """Transforms the data in the new subspace spanned by the eigenvector␣
 ↪corresponding to the largest eigenvalue.

    Parameters
    ----------
    X : array, shape [N, D]
        Data matrix.
    L : array, shape [D]
        Eigenvalues of Sigma_X
```

```
    U : array, shape [D, D]
        Eigenvectors of Sigma_X

    Returns
    -------
    X_t : array, shape [N, 1]
        Transformed data

    """
    print("L before transformation: ")
    print(np.diag(L))
    print()

    print("U before transformation: ")
    print(U)
    print()

    idx = np.argsort(L)[::-1]
    U = U[:,idx]
    L = L[idx[:-1]]
    U = U[:, :-1]

    print("L after transformation: ")
    print(L)
    print()

    print("U after transformation: ")
    print(U)
    print()

    print("X before transformation: ")
    print(X.shape)
    print(X)
    print()

    print("X after transformation: ")
    print(X.dot(U).shape)
    print(X.dot(U))
    print()

    return X.dot(U)
```

[7]:
```
X_t = transform(X, U, L)
```

```
L before transformation:
[[5.625 0.   ]
 [0.    0.375]]
```

```
U before transformation:
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]

L after transformation:
[5.625]

U after transformation:
[[0.70710678]
 [0.70710678]]

X before transformation:
(17, 2)
[[-3 -2]
 [-2 -1]
 [-1  0]
 [ 0  1]
 [ 1  2]
 [ 2  3]
 [-2 -2]
 [-1 -1]
 [ 0  0]
 [ 1  1]
 [ 2  2]
 [-2 -3]
 [-1 -2]
 [ 0 -1]
 [ 1  0]
 [ 2  1]
 [ 3  2]]

X after transformation:
(17, 1)
[[-3.53553391]
 [-2.12132034]
 [-0.70710678]
 [ 0.70710678]
 [ 2.12132034]
 [ 3.53553391]
 [-2.82842712]
 [-1.41421356]
 [ 0.        ]
 [ 1.41421356]
 [ 2.82842712]
 [-3.53553391]
 [-2.12132034]
 [-0.70710678]
```

```
[ 0.70710678]
[ 2.12132034]
[ 3.53553391]]
```

1.3 SVD

1.3.1 Task 5: Given the matrix M find its SVD decomposition $M = U \cdot \Sigma \cdot V$ and reduce it to one dimension using the approach described in the lecture.

[0]:
```
M = np.array([[1, 2], [6, 3],[0, 2]])
```

[0]:
```
def reduce_to_one_dimension(M):
    """Reduces the input matrix to one dimension using its SVD decomposition.

    Parameters
    ----------
    M : array, shape [N, D]
        Input matrix.

    Returns
    -------
    M_t: array, shape [N, 1]
        Reduce matrix.

    """
    u, s, vh = np.linalg.svd(M, full_matrices=True)
    print("M before transformation: ")
    print(M.shape)
    print(M)
    print()

    print("U matrix before transformation: ")
    print(u.shape)
    print(u)
    print()

    print("Sigma matrix before transformation: ")
    print(s.shape)
    print(s)
    print()

    print("V matrix before transformation: ")
    print(vh.shape)
    print(vh)
    print()

    idx = np.argsort(s)[::-1]
```

```
    s = s[idx[0]]
    u = u[:,idx[0]]
    vh = vh[:,idx[0]]
    u = u[:,np.newaxis]
    vh = vh[:, np.newaxis]

    print("U matrix after transformation: ")
    print(u.shape)
    print(u)
    print()

    print("Sigma matrix after transformation: ")
    print(s.shape)
    print(s)
    print()

    print("V matrix after transformation: ")
    print(vh.shape)
    print(vh)
    print()

    result = (u * s).dot(vh.T)
    print("M after transformation: ")
    print(M.shape)
    print(result)
    print()
    print("Matrix rank of M after transformation: ")
    print(np.linalg.matrix_rank(result))
    return result
```

[10]:
```
M_t = reduce_to_one_dimension(M)
```

```
M before transformation:
(3, 2)
[[1 2]
 [6 3]
 [0 2]]

U matrix before transformation:
(3, 3)
[[-0.27073584  0.54578489 -0.79298232]
 [-0.95094914 -0.27969357  0.13216372]
 [-0.14965909  0.78986731  0.59473674]]

Sigma matrix before transformation:
(2,)
[7.02571561 2.15390813]
```

```
V matrix before transformation:
(2, 2)
[[-0.85065081 -0.52573111]
 [-0.52573111  0.85065081]]

U matrix after transformation:
(3, 1)
[[-0.27073584]
 [-0.95094914]
 [-0.14965909]]

Sigma matrix after transformation:
()
7.025715605900788

V matrix after transformation:
(2, 1)
[[-0.85065081]
 [-0.52573111]]

M after transformation:
(3, 2)
[[1.61803399 1.        ]
 [5.68328157 3.51246118]
 [0.89442719 0.5527864 ]]

Matrix rank of M after transformation:
1
```

Appendix

We confirm that the submitted solution is original work and was written by us without further assistance. Appropriate credit has been given where reference has been made to the work of others.

Munich, January 19, 2020, Signature Marcel Bruckner (03674122)

Munich, January 19, 2020, Signature Julian Hohenadel (03673879)

Munich, January 19, 2020, Signature Kevin Bein (03707775)