Dimensionality Reduction

Machine Learning - Prof. Dr. Stephan Günnemann

Leonardo Freiherr von Lerchenfeld

January 11, 2018

1 Problem 1

$$\begin{aligned}
cov(X) &= \frac{1}{N}\sum_{n=1}^{N}(x'_n - \bar{x})(x'_n - \bar{x})^T && (1)\\
with &\quad x'_n = U^T x_n && (2)\\
cov(x) &= U^T S U && (3)\\
Maximize &\quad U^T S U + \lambda(I - U^T U) && (4)\\
\frac{\partial}{\partial u_{m+1}}(\cdots) &= 0 && (5)\\
&= \begin{pmatrix} 0 \\ u_{m+1} \end{pmatrix} S \begin{pmatrix} 0 & u_{m+1} \end{pmatrix} - \lambda I \begin{pmatrix} 0 & 0 \\ 0 & u_{m+1}^2 \end{pmatrix} && (6)\\
S u_{m+1} &= \lambda_{m+1} u_{m+1} && (7)
\end{aligned}$$

2 Problem 2

$$\begin{aligned}
p(y|z) &= \mathcal{N}(y|Wz + \mu, \sigma^2 I) && (8)\\
Log\ L &= -\frac{N}{2}\left(d\ ln(2\pi) + ln|C| + tr(C^{-1}S)\right) && (9)\\
with &\quad S = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)(y_i - \mu)^T && (10)\\
\mu_{yML} &= \frac{1}{N}\sum_{i=1}^{N} y_i && (11)\\
&= \frac{1}{N}\sum_{i=1}^{N} A x_i && (12)\\
&= A\frac{1}{N}\sum_{i=1}^{N} x_i && (13)\\
&= A\mu_{xML} && (14)\\
\Phi_{yML} &= \sigma_y^2 I && (15)\\
&= \begin{pmatrix} (y_1 - \bar{y_1})^2 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & (y_D - \bar{y_D})^2 \end{pmatrix} && (16)\\
&= \begin{pmatrix} A(x_1 - \bar{x_1})^2 A^T & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & A(x_D - \bar{x_D})^2 A^T \end{pmatrix} && (17)\\
&= A\Phi_{xML}A^T && (18)\\
W_{yML} &= U_K(\Lambda_{yK} - \sigma_y^2)^{0.5} V && (19)\\
&= U_K(A\Lambda_{xK}A^T - A\sigma_x^2 A^T)^{0.5} V && (20)\\
&= AU_K(\Lambda_{xK} - \sigma_x^2)^{0.5} V && (21)\\
&= AW_{xML} && (22)
\end{aligned}$$

```
M=[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;0 3 0 0 4];
reddim=3;
len1=size(M,1);
len2=size(M,2);
[U,S,V] = svd(M);
% It works, check it out: U*S*V'
V=V(1:len2,1:reddim);
Vo=eye(reddim);
for i=1:reddim
    Vo(i,i)=max(V(i));
end
Uo=U(1:len1,1:reddim);
So=S(1:reddim,1:reddim);
Uo*So*Vo'
```

ans =

	Sci-Fi	**Romantic**	**Noise**
	0.9649	-0.1019	-0.0395
	2.8946	-0.3056	-0.1184
	3.8595	-0.4075	-0.1579
	4.8244	-0.5094	-0.1973
	0.2922	3.3654	-0.3815
	0.3653	4.2067	-0.4768
	0.1461	1.6827	-0.1907
Leslie	1.2199	1.8419	1.8490

That representation would predict Leslie's rating for Sci-Fi movies with

3 (≈ 1.2 (+ 1.8)) and romantic movies with 4 (≈ 1.8 (+ 1.8)).

However, be aware that the noise on Leslie`s preference is relatively high.

10_homework_dim_reduction

January 11, 2018

1 Programming assignment 10: Dimensionality Reduction

```
In [11]: import numpy as np
         import matplotlib.pyplot as plt

         %matplotlib inline
```

1.1 PCA Task

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.

1.1.1 The given data X

```
In [12]: 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)])
```

1.1.2 Task 1: Calculate the covariance matrix Σ

```
In [13]: 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
```

```
    """
    # TODO
    Sigma = np.cov(X,rowvar=False)
    return Sigma
```

1.1.3 Task 2: Calculate eigenvalues and eigenvectors of Σ.

```
In [14]: 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

             """
             # TODO
             [L,D] = np.linalg.eig(S)
             return L,D
```

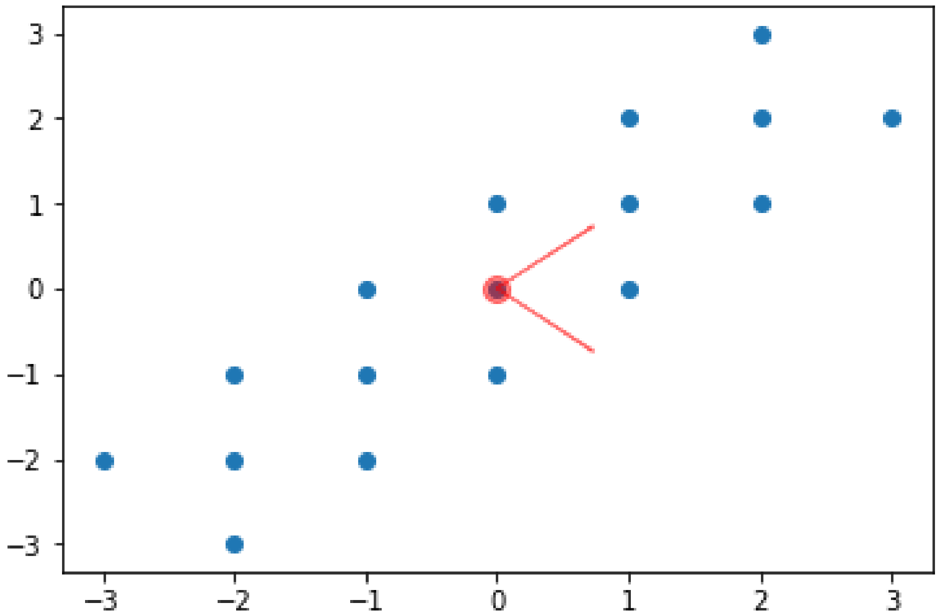
1.1.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

```
In [15]: # 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)

         plt.arrow(mean_d1, mean_d2, U[0, 0], U[0, 1], width=0.01, color='red', alp
         plt.arrow(mean_d1, mean_d2, U[1, 0], U[1, 1], width=0.01, color='red', alp
```




```
In [ ]: What do you observe in the above plot? Which eigenvector corresponds to the

        Write your answer here:

        The data has a high variance in the direction of the first eigenvector,
        which has the larger eigenvalue. The second eigenvector corresponds
        to the the second/smallest eigenvalue.
```

1.1.5 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.

```
In [ ]: def transform(X, U, L):
            """Transforms the data in the new subspace spanned by the eigenvector c

            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

        """
        # TODO
        i = np.argmin(L)
        U_new = np.delete(U, i, 1)
        X_t = np.matmul(X, U_new)
        return X_t
```

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

1.2 Task SVD

1.2.1 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.

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

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

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

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

        """
        # TODO
        return None
```

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