Machine Learning Worksheet Solution 10

Julius Jankowski

1 Dimensionality Reduction

Problem 1:

First, we have to maximize the new variance

$$\max_{\mathbf{u}_{M+1}} \sum_{i=1}^{M+1} \mathbf{u}_i^T S \mathbf{u}_i = \max_{\mathbf{u}_{M+1}} \mathbf{u}_{M+1}^T S \mathbf{u}_{M+1} \tag{1.1}$$

under the constraints that the eigen base is orthogonal and the length is normalized to one:

$$\mathbf{u}_{M+1}^T \mathbf{u}_i = 0, \text{for } i < M+1. \tag{1.2}$$

$$\mathbf{u}_{M+1}^T \mathbf{u}_{M+1} = 1 \tag{1.3}$$

Hence the lagrange function looks like this:

$$L = \mathbf{u}_{M+1}^T S \mathbf{u}_{M+1} + \mathbf{u}_{M+1}^T \sum_{i=1}^{M} \lambda_i \cdot \mathbf{u}_i + \lambda_{M+1}(\mathbf{u}_{M+1}^T \mathbf{u}_{M+1} - 1) \tag{1.4}$$

The respective derivations are:

$$\frac{\delta L}{\delta \mathbf{u}_{M+1}} = 2S\mathbf{u}_{M+1} + 2\lambda_{M+1}\mathbf{u}_{M+1} + \sum_{i=1}^{M} \lambda_i \cdot \mathbf{u}_i \tag{1.5}$$

$$\frac{\delta L}{\delta \lambda_i} = \mathbf{u}_{M+1}^T \mathbf{u}_i \tag{1.6}$$

$$\frac{\delta L}{\delta \lambda_{M+1}} = \mathbf{u}_{M+1}^T \mathbf{u}_{M+1} - 1 \tag{1.7}$$

which have to be equal to zero. We can bring these equations together like this:

$$2S\mathbf{u}_{M+1} + 2\lambda_{M+1}\mathbf{u}_{M+1} = -\sum_{i=1}^{M} \lambda_i \cdot \mathbf{u}_i \mid \cdot \mathbf{u}_{M+1}^T \tag{1.8}$$

$$\implies 2\mathbf{u}_{M+1}^T S \mathbf{u}_{M+1} + 2\lambda_{M+1}\mathbf{u}_{M+1}^T \mathbf{u}_{M+1} = -\sum_{i=1}^{M} \lambda_i \cdot \mathbf{u}_{M+1}^T \mathbf{u}_i \tag{1.9}$$

$$\implies \mathbf{u}_{M+1}^T S \mathbf{u}_{M+1} + \lambda_{M+1} = 0 \tag{1.10}$$

We end up with a formulation for eigenvalue and eigenvector if $\mathbf{u}_{M+1}$ is an eigenvector and $-\lambda_{M+1}$ is the corresponding eigenvalue of S. When we look again at the original formulation of the optimization problem:

$$\max_{\mathbf{u}_{M+1}} \mathbf{u}_{M+1}^T S \mathbf{u}_{M+1} = \rho_{M+1} \tag{1.11}$$

where ρ_{M+1} is the M+1's eigenvalue, we see that we should take the highest eigenvalue.

Problem 2:

It is obvious that if we linearly transform a set of data, we transform the mean value in the same way. Hence we have the new mean:

$$\mu_y = \mathbf{A}\mu_x = \mathbf{AWz} + \mathbf{A}\mu \tag{1.12}$$

this allready shows the desired correlations for $\mathbf{A}\mu$ and $\mathbf{AW}$ as new parameters for the MLE solution. Furthermore, we demand that the resulting likelihood of the conditional distribution stays the same for similar points $\mathbf{x}$ (unless a constant factor c):

$$p(\mathbf{x}|\mathbf{z}) = c \cdot p(\mathbf{y}(\mathbf{x})|\mathbf{z}) \tag{1.13}$$

$$p(\mathbf{y}(\mathbf{x})|\mathbf{z}) = c_y \cdot exp(-0.5(\mathbf{Ax} - \mathbf{AWz} - \mathbf{A}\mu)^T \mathbf{\Phi}_y^{-1} (\mathbf{Ax} - \mathbf{AWz} - \mathbf{A}\mu) \tag{1.14}$$

$$p(\mathbf{y}(\mathbf{x})|\mathbf{z}) = c_y \cdot exp(-0.5(\mathbf{x} - \mathbf{Wz} - \mu)^T \mathbf{A}^T \mathbf{\Phi}_y^{-1} \mathbf{A} (\mathbf{x} - \mathbf{Wz} - \mu) \tag{1.15}$$

$$\implies \mathbf{A}^T \mathbf{\Phi}_y^{-1} \mathbf{A} = \mathbf{\Phi} \tag{1.16}$$

$$\implies \mathbf{\Phi}_y = \mathbf{A\Phi A}^T \tag{1.17}$$

The last thing to do is to show that the new variance matrix has still a diagonal form if $\mathbf{A}$ is a orthogonal matrix:

$$\mathbf{A\Phi A}^T = \mathbf{A}\sigma^2\mathbf{I}\mathbf{A}^T = \sigma^2\mathbf{AA}^T = \sigma^2\mathbf{I} \tag{1.18}$$

Problem 3:

In order to analyze the concept relation for Leslie, we have to transform her data by the "movie-to-concept" matrix $\mathbf{V}$:

$$\mathbf{y}_{Leslie} = \mathbf{x}_{Leslie} \cdot \mathbf{V} = [1.74\ 2.84] \tag{1.19}$$

which gives a hint that Leslie is more into romantic movies than into sci-fi movies. We could also transform this condensed information back to the original rating space:

$$\mathbf{x}_{Leslie,pred} = \mathbf{y}_{Leslie} \cdot \mathbf{V}^T = [1.01\ 1.01\ 1.01\ 2.02\ 2.02] \tag{1.20}$$

which is a prediction for possible future ratings of Leslie.

10_homework_dim_reduction

January 14, 2018

1 Programming assignment 10: Dimensionality Reduction

```
In [1]: 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 [2]: 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 [18]: 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
```

```
    """

    N, D = X.shape

    mean = np.zeros((D))
    S = np.zeros((D,D))

    for i in range(N):
        mean = X[i,:]
    mean = mean/N

    for i in range(N):
        S = S + np.outer((X[i,:]-mean),(X[i,:]-mean))
    S = S/N

    return S
```

1.1.3 Task 2: Calculate eigenvalues and eigenvectors of Σ.

```
In [22]: 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

    """

    L, U = np.linalg.eigh(S)

    return L, U
```

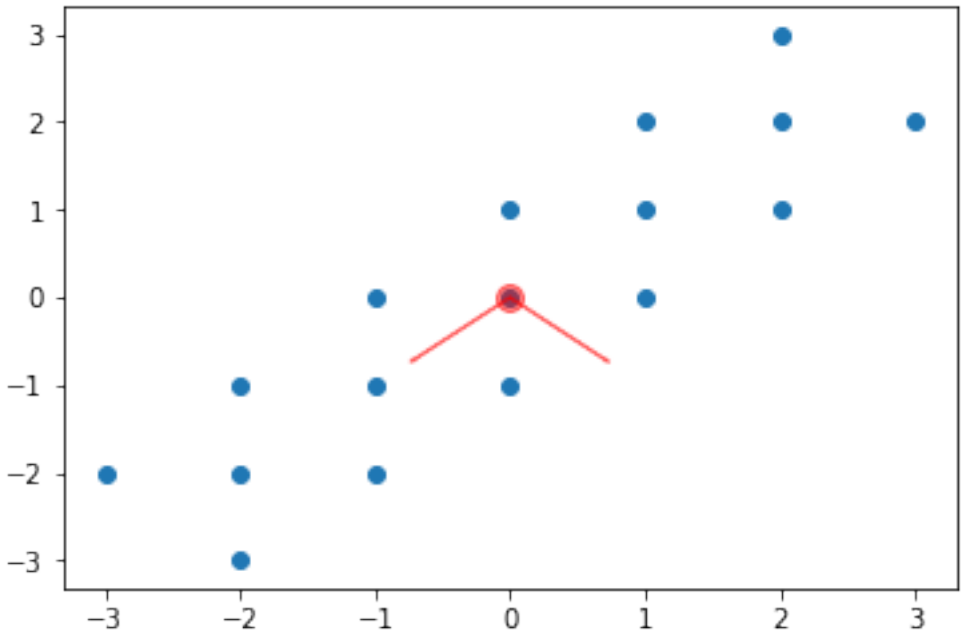
1.1.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

```
In [23]: # 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', alpha=0.5)
plt.arrow(mean_d1, mean_d2, U[1, 0], U[1, 1], width=0.01, color='red', alpha=0.5);
```



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

Write your answer here:

The plot shows the sampled data in blue, the mean point in red and the eigenvectors of the covariance matrix as red lines. The eigenvector which points to the lower left corner corresponds to the higher eigenvalue since the data points are more spread along this axis.

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

             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

        """

        N, D = X.shape

        min_idx = np.argmin(L)

        U_new = np.zeros((D,D-1))

        flag = False
        for i in range(D):
            if i == min_idx:
                flag = True
            else:
                if flag == False:
                    U_new[:,i] = U[:,i]
                else:
                    U_new[:,i-1] = U[:,i]

        X_t = np.inner(U_new.transpose(), X)

        return X_t
```

```
In [31]: 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 [34]: M = np.array([[1, 2], [6, 3],[0, 2]])
```

```
In [53]: 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.

        """

        N, D = M.shape

        U,S,V = np.linalg.svd(M)

        V_new = V[0:-1,:] # cut the transform matrix (allready sorted by svd)

        M_t = np.dot(M,V_new.transpose())

        return M_t
```

```
In [54]: M_t = reduce_to_one_dimension(M)
```