Machine Learning Assignment 10

Shivangi Aneja

15-January-2018

Problem 1

Assume, the result is true for any dimension M, we have
$u_M^T S u_M = \lambda_M$

For M+1 , we have following constraints
$u_{M+1}^T u_{M+1} = 1, u_{M+1}^T u_1 = 0, u_{M+1}^T u_2 = 0, ...u_{M+1}^T u_M = 0$

To make this constraint optimization, we have
$F(u_{M+1}) = u_{M+1} S u_{M+1} + \lambda_{M+1}(1 - u_{M+1}^T u_{M+1}) + \mu_1(u_{M+1}^T u_1) + \mu_2(u_{M+1}^T u_2) + ... + \mu_M(u_{M+1}^T u_M)$

Now ,
$\dfrac{F(u_{M+1})}{u_{M+1}} = 0$

$\Rightarrow 2Su_{M+1} - 2\lambda_{M+1} u_{M+1} + \mu_1(u_1) + \mu_2(u_2) + ... + \mu_M(u_M) = 0$

Now pre-multiply this equation with u_{M+1}^T
$2u_{M+1}^T S u_{M+1} - 2\lambda_{M+1} u_{M+1}^T u_{M+1} + \mu_1(u_{M+1}^T u_1) + \mu_2(u_{M+1}^T u_2) + ... + \mu_M(u_{M+1}^T u_M) = 0$

Substituting the constraints, we have
$2u_{M+1}^T S u_{M+1} - 2\lambda_{M+1} = 0$
$\Rightarrow \boxed{u_{M+1}^T S u_{M+1} = \lambda_{M+1}}$

Hence, variance is maximized if the eigenvector is chosen to be the one corresponding to eigenvector λ_{M+1} where the eigenvalues have been ordered in decreasing value

Problem 2

Given, $p(z) = \mathcal{N}(z|0, I)$
$p(x|z) = \mathcal{N}(x|Wz + \mu, \phi)$

We know, $p(x) = \mathcal{N}(x|\mu, WW^T + \phi)$

For the transformation $\boxed{y = Ax}$ we have,
$\boxed{Mean = E[Ax] = AE[x] = A\mu}$
$Variance = E[(Ax - E[Ax])(Ax - E[Ax])^T] = AE[(x - E[x])(x - E[x])^T]A^T =$
$\boxed{Variance = AWW^TA^T + A\phi A^T}$

$p(Ax) = \mathcal{N}(x|A\mu, AWW^TA^T + A\phi A^T)$
Given the MLE solutions for p(x) , seeing the distribution p(Ax) and P(x)we can derive MLE solutions for p(Ax) as follows

MLE Value	MLE for p(x)	MLE for p(Ax)
μ	μ_{ML}	$A\mu_{ML}$
W	W_{ML}	AW_{ML}
ϕ	ϕ_{ML}	$A\phi_{ML}A^T$

Given ϕ is proportional to unit matrix and A is orthogonal,
$AA^T = I$
$\phi = \sigma^2 I$
$\Rightarrow A\phi A^T = \sigma^2 I = A\sigma^2 A^T = \sigma^2 AA^T = \sigma^2 I$
The variance of the new transformed space p(Ax) is same as variance of old distribution p(x).
$\boxed{\phi = A\phi A^T = \sigma^2 I}$

Thus, probabilistic PCA is covariant under a rotation of the axes of data space

Problem 3

The mapping of movies to concept is

$$V = \begin{bmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{bmatrix}$$

The representation of Leslie is given as

$$q = \begin{bmatrix} 0 & 3 & 0 & 0 & 4 \end{bmatrix}$$

Representation of Leslie in Concept Space

$$A = qV = \begin{bmatrix} 1.74 & 2.84 \end{bmatrix}$$

As it is clear from the matrix that Leslie is more fond of Romantic Movies (2.84) and less likely to watch Sci-Fi movies(1.74)

To predict how well Leslie would like other movies, we have

$$AV^T = \begin{bmatrix} 0.58 \times 1.74 & 0.58 \times 1.74 & 0.58 \times 1.74 & 0.71 \times 2.84 & 0.71 \times 2.84 \end{bmatrix}$$

$$A = \begin{bmatrix} 1.0092 & 1.0092 & 1.0092 & 2.0164 & 2.0164 \end{bmatrix}$$

The matrix A represents how likely is Leslie to watch the movies
Matrix, Alien, Star Wars = 1.0092
Titanic, Casablanca = 2.0164

Problem 4

January 13, 2018

1 Programming assignment 10: Dimensionality Reduction

```
In [20]: 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 [21]: 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 [22]: 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
        cov = np.cov(X.T)
        return cov
```

1.1.3 Task 2: Calculate eigenvalues and eigenvectors of Σ.

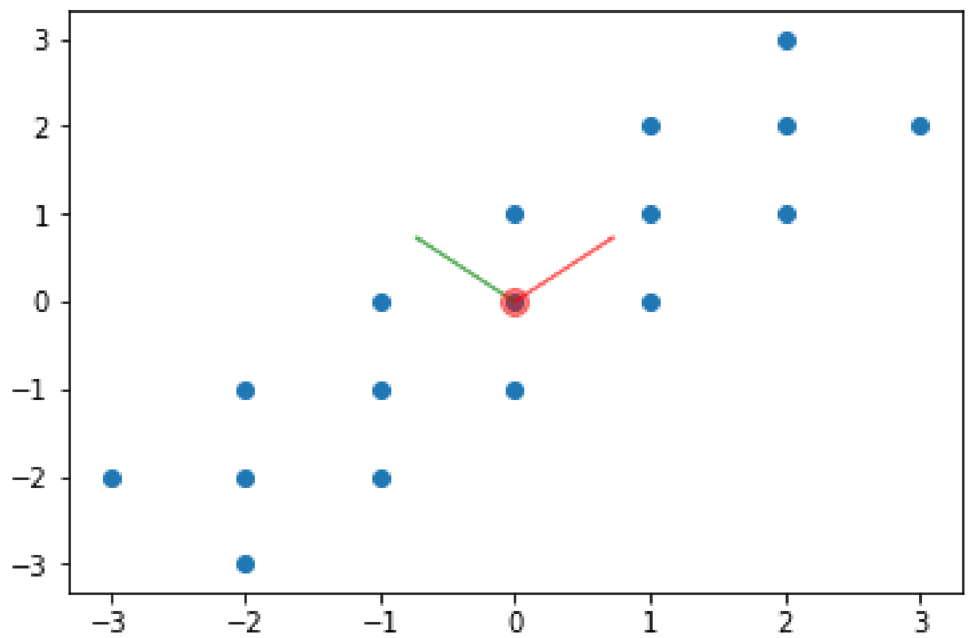
```
In [23]: def get_eigen(S):
             """Calculates the eigenvalues and eigenvectors of the input matrix.

             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, U = np.linalg.eig(S)
             return L,U.T
```

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

```
In [24]: # 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 U


         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='green', alpha=0.5);
```

```
[ 5.625  0.375]
[[ 0.70710678  0.70710678]
```

```
[-0.70710678  0.70710678]]
```



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

The Eigen Values are as follows:
Red Eigen Vector : 5.625
Green Eigen Vector : 0.375

Thus the smallest Eigen vector is [-0.70710678 0.70710678]. The larger Eigen vector is [0.70710678 0.70710678].

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 [26]: 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

    """
    index = np.argmax(L)
    eV = U[index]
    X_t = X.dot(eV)
    print X_t
    return X_t
```

```
In [12]: X_t = transform(X, U, L)
```

```
[-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]
```

1.2 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 [27]: M = np.array([[1, 2], [6, 3],[0, 2]])
```

```
In [28]: 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.

             """
             # TODO
             u,s,v = np.linalg.svd(M)

             index = np.argmax(s)
             eV = v[index]
```

```
        M_t = M.dot(eV)

        print M_t

        return M_t
```

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

```
[-1.90211303 -6.68109819 -1.05146222]
```