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 [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
```

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

1.1.3 Task 2: Calculate eigenvalues and eigenvectors of Σ.

```
In [4]: 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
```

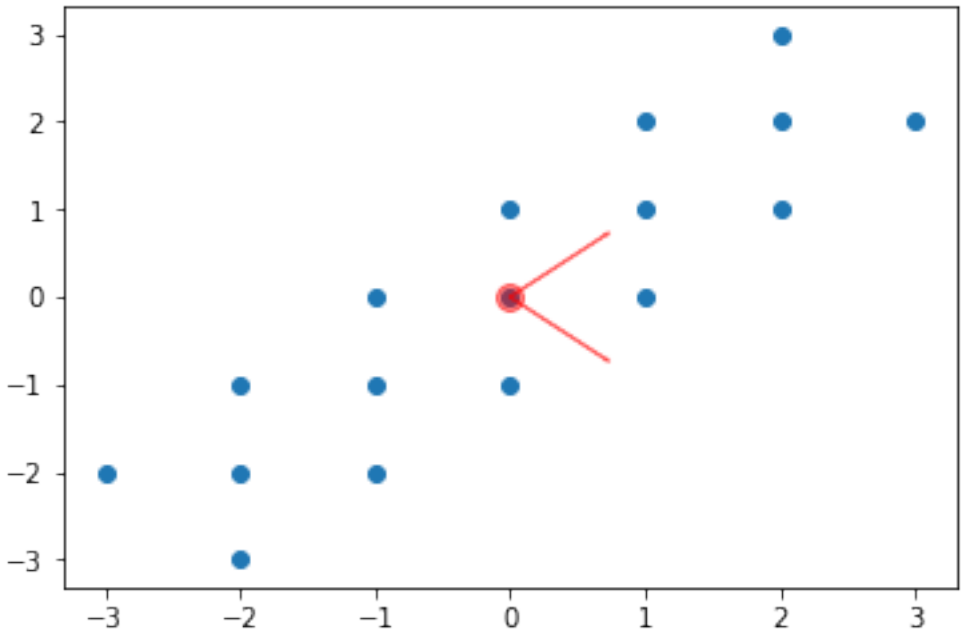
1.1.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

```
In [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)

        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:

Due to our 2-dimensional data, we have two eigenvectors. The larger one is pointing in the direction of the higher variance, while the lower one points in the direction of the lower variance (the dimension we want to get rid of in dimensionality reduction). Per definition, they are orthogonal to each other and span an alternative 2d space.

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 [6]: 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

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

```
In [7]: 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 [8]: M = np.array([[1, 2], [6, 3],[0, 2]])
```

```
In [9]: 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,full_matrices=False)
        M_t = U[:,0] * S[0]
        return M_t
```

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