Machine Learning: Homework #10

Due on January 15, 2018 at 09:59am

Professor Dr. Stephan Guennemann

Marc Meissner - 03691673

Problem 1

In this exercise, we use proof by induction to show that the linear projection onto an M - dimensional subspace that maximizes the variance of the projected data is defined by the M eigenvectors of the data covariance matrix S, given by

corresponding to the M largest eigenvalues. In Section 12.1 in Bishop this result was proven for the case of $M = 1$. Now suppose the result holds for some general value of M and show that it consequently holds for dimensionality $M + 1$. To do this, first set the derivative of the variance of the projected data with respect to a vector u_{M+1} defining the new direction in data space equal to zero. This should be done subject to the constraints that u_{M+1} be orthogonal to the existing vectors $u_1, ..., u_M$, and also that it be normalized to unit length. Use Lagrange multipliers to enforce these constraints. Then make use of the orthonormality properties of the vectors $u_1, ..., u_M$ to show that the new vector u_{M+1} is an eigenvector of S. Finally, show that the variance is maximized if the eigenvector is chosen to be the one corresponding to eigenvector $_{M+1}$ where the eigenvalues have been ordered in decreasing value.

Solution
Formulate lagrangian:

$$u_{M+1}^T S u_{M+1} + \sum_{i=1}^{M+1} \lambda_i \cdot (\delta_{i,M+1} - u_i^T u_{M+1})$$

where $\delta_{i,M+1} = 1$ only holds for $i = M + 1$.
Derivation with respect to λ_i and λ_{M+1}, respectively:

$$\frac{\partial}{\partial \lambda_i} u_{M+1}^T S u_{M+1} + \sum_{i=1}^{M+1} \lambda_i \cdot (\delta_{i,M+1} - u_i^T u_{M+1}) = 0$$
$$u_i^T u_{M+1} = 0$$

$$\frac{\partial}{\partial \lambda_{M+1}} u_{M+1}^T S u_{M+1} + \sum_{i=1}^{M+1} \lambda_i \cdot (\delta_{i,M+1} - u_i^T u_{M+1}) = 0$$
$$u_{M+1}^T u_{M+1} = 1$$

Derivation with respect to u_{M+1}:

$$\frac{\partial}{\partial u_{M+1}} u_{M+1}^T S u_{M+1} + \sum_{i=1}^{M+1} \lambda_i \cdot (\delta_{i,M+1} - u_i^T u_{M+1}) = 0$$
$$2 S u_{M+1} - 2\lambda_{M+1} u_{M+1} = \sum_{i=1}^{M} \lambda_i u_i$$

Problem 1 continued on next page...

Multiplying with u_{M+1}^T:

$$2u_{M+1}^T S u_{M+1} - 2\lambda_{M+1} u_{M+1}^T u_{M+1} = \sum_{i=1}^{M} \lambda_i u_{M+1}^T u_i$$

$$2u_{M+1}^T S u_{M+1} - 2\lambda_{M+1} = 0$$

$$u_{M+1}^T S u_{M+1} = \lambda_{M+1}$$

Since u_{M+1} is orthogonal to all other eigenvectors u_i, it is an eigenvector of S. The result above shows that the value of $u_{M+1}^T S u_{M+1}$ corresponds to λ_{M+1}, which is the next biggest eigenvalue.

Problem 2

Consider the latent space distribution

$$p(z) = N(z|0, I)$$

and a conditional distribution for the observed variable $x \in R^d$,

$$p(x|z) = N(x|Wz + \mu, \Phi)$$

where Φ is an arbitrary symmetric, positive-definite noise covariance variable. Now suppose that we make a nonsingular linear transformation of the data variables $y = Ax$ where A is a non-singular dxd matrix. If μ_{ML} , W_{ML} , and Φ_{ML} represent the maximum likelihood solution corresponding to the original untransformed data, show that $A\mu_{ML}$, AW_{ML} , and $A\Phi_{ML}AT$ will represent the corresponding maximum likelihood solution for the transformed data set. Finally, show that the form of the model is preserved if A is orthogonal and Φ is proportional to the unit matrix so $\Phi = \sigma^2 I$ (i.e. probabilistic PCA). The transformed Φ matrix remains proportional to the unit matrix, and hence probabilistic PCA is covariant under a rotation of the axes of data space, as is the case for conventional PCA.

Solution
Linear transformations on gaussian data can be seen as a linear transformation of the underlying gaussian distribution. Therefore:

$$A(Wz + \mu) = AWz + A\mu$$

and

$$A\Phi A^T$$

holds. For orthogonal A and Φ being proportional to the unit matrix, the covariance becomes:

$$A\Phi A^T = A\sigma^2 I A^T = \sigma^2 A A^T = \sigma^2 I$$

Therefore, the covariance remains unchanged.

Problem 3

Use the SVD shown below. Suppose a new user Leslie assigns rating 3 to Alien and rating 4 to Titanic, giving us a representation of Leslie in the 'original space' of [0, 3, 0, 0, 4]. Find the representation of Leslie in concept space. What does that representation predict about how well Leslie would like the other movies appearing in our example data?

Solution

The transposed data is derived by $P = A \cdot V$. Since we only need the transformation for Leslie, this is straightforward:

$$\begin{pmatrix} 0 & 3 & 0 & 0 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{pmatrix} = \begin{pmatrix} 1.74 & 2.84 \end{pmatrix}$$

One could interpret this in the following way: It is likely that Leslie is fond of science fiction and romantic movies alike, but she prefers romantic ones.

Problem 4

Load the notebook 10_*homework_dim_reduction.ipynb* from Piazza. Fill in the missing code and run the notebook. Convert the evaluated notebook to pdf and add it to the printout of your homework.

Solution

Attached.

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
```

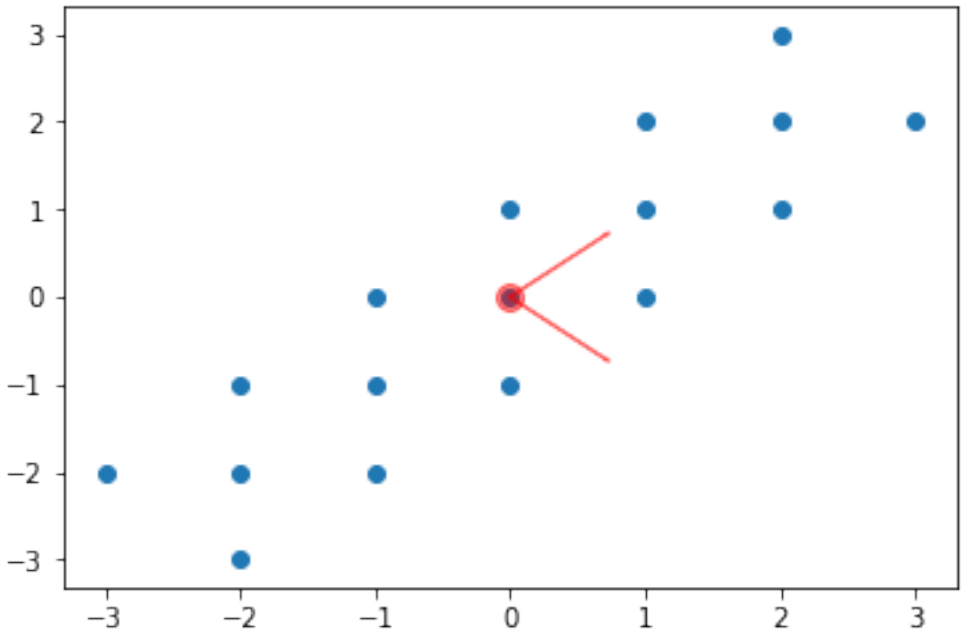
1.1.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

```
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)
```