Machine Learning

Homework 10 solution - Shyam Arumugaswamy

1.

Suppose that the result holds for projection spaces of dimensionality M. The M+1 dimensional principal subspace will be defined by the M principal eigenvectors u1, . . . , uM together with an additional direction vector uM+1 whose value we wish to determine. We must constrain uM+1 such that it cannot be linearly related to u1, . . . , uM (otherwise it will lie in the M-dimensional projection space instead of defining an M+1 independent direction). This can easily be achieved by requiring that uM+1 be orthogonal to u1, . . . , uM, and these constraints can be enforced using Lagrange multipliers η1, . . . , η_M.

Following the argument given in section 12.1.1 for u1 we see that the variance in the direction uM+1 is given by $u^T_{M+1} S u_{M+1}$.

maximize this using a Lagrange multiplier λ_{M+1} to enforce the normalization constraint $u^T_{M+1} u_{M+1} = 1$. Thus we seek a maximum of the function

$u^T_{M+1} S u_{M+1} + \lambda_{M+1} (1 - u^T_{M+1} u_{M+1}) + \sum_{i=1}^{M} \eta i \, uTM \, ui$ wrt uM+1

The stationary points occur when

$0 = 2SuM+1 - 2\lambda_{M+1} u_{M+1} + \sum_{i=1}^{M} \eta i \, ui$

Left multiplying with u^T_j and using the orthogonality constraints, we see that ηj = 0

for j = 1, . . . ,M. We therefore obtain

$S u_{M+1} = 2\lambda_{M+1} u_{M+1}$

and so uM+1 must be an eigenvector of S with eigenvalue uM+1. The variance in the direction uM+1 is given by $u^T_{M+1} S u_{M+1} = \lambda_{M+1}$ and so is maximized by choosing uM+1 to be the eigenvector having the largest eigenvalue amongst those not previously selected. Thus the result holds also for projection spaces of dimensionality M + 1, which completes the inductive step. Since we have already shown this result explicitly for M = 1 if follows that the result must hold for any M <= D.

2.

The log likelihood of the model is

$$L(\mu, W, \phi) = -\frac{N D}{2} \ln(2\pi) - \frac{N}{2} \ln |W W^T + \phi| - \frac{1}{2} \sum_{n=1}^{N} \{ (x_n - \mu)^T (W W^T + \phi)^{-1} (x_n - \mu) \}$$

If we consider log likelihood function for the transformed data set, we get

$$L_A(\mu, W, \phi) = -\frac{N D}{2} \ln(2\pi) - \frac{N}{2} \ln |W W^T + \phi| - \frac{1}{2} \sum_{n=1}^{N} \{ (A x_n - \mu)^T (W W^T + \phi)^{-1} (A x_n - \mu) \}$$

Solving for maximum likelihood estimator for μ , we get

$$\mu_A = \frac{1}{N} \sum_{n=1}^{N} A x_n = A x' = A \mu_{ML}$$

Back substituting into log likelihood function and using the definition of sample covariance matrix

$$L_A(\mu, W, \phi) = -\frac{N D}{2} \ln(2\pi) - \frac{N}{2} \ln |W W^T + \phi| - \frac{1}{2} \sum_{n=1}^{N} Tr \{ (W W^T + \phi)^{-1} A S A^T \}$$

We can cast the final term into same form as corresponding term in original log likelihood function if we define

$$\Phi_A = A \Phi^{-1} A^T \quad , \quad W_A = A W$$

Therefore

$$L_A(\mu_A, W_A, \phi_A) = -\frac{N D}{2} \ln(2\pi) - \frac{N}{2} \ln |W_A W_A^T + \phi_A| - \frac{1}{2} \sum_{n=1}^{N} \{ (x_n - \mu_A)^T (W_A W_A^T + \phi_A)^{-1} (x_n - \mu_A) \} - N \ln |A|$$

This takes the same form as the original log likelihood function apart from an additive constant $-\ln |A|$. Thus the maximum likelihood solution in the new variables for the transformed data set will be identical to that in the old variables.

In the case of probabilistic PCA the noise covariance ϕ is proportional to the unit matrix and takes the form $\sigma^2 I$. For this constraint to be preserved we require $AA^T = I$ so that A is an orthogonal matrix. This corresponds to a rotation of the coordinate system. For factor analysis ϕ is a diagonal matrix, and this property will be preserved if A is also diagonal since the product of diagonal matrices is again diagonal.

This corresponds to an independent re-scaling of the coordinate system.

3.

We can map [0, 3, 0, 0, 4] into “concept space” by multiplying it by V,

We get the representation of Leslie in concept space which is [1.74, 2.84].

Multiplying [1.74, 2.84] by V^T, we get [1.0092, 1.0092, 1.0092, 2.0164, 2.0164] which can be used to represent how well Leslie would like the other movies.

Programming assignment 10: Dimensionality Reduction¶

In [40]:

```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

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.

The given data X¶

In [41]:

```
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)])
```

Task 1: Calculate the covariance matrix Σ¶

In [42]:

```
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
    return np.cov(X.transpose())
```

Task 2: Calculate eigenvalues and eigenvectors of Σ.¶

In [43]:

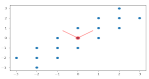
```
def get_eigen(S):
    """Calculates the eigenvalues and eigenvectors of the input matrix.

    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
    return np.linalg.eigh(S)
```

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

In [44]:

```
# 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:

[YOUR ANSWER]

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 [45]:

```
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
    return None
```

In [46]:

```
X_t = transform(X, U, L)
```

Task SVD¶

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 [47]:

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

In [48]:

```
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
    return None
```

In [49]:

```
M_t = reduce_to_one_dimension(M)
```