CS 182: Introduction to Machine Learning, Fall 2021
Homework 5

(Due on Tuesday, Nov. 23 at 11:59pm (CST))

Notice:

- Please submit your assignments via Gradescope. The entry code is KYJ626.
- Please make sure you select your answer to the corresponding question when submitting your assignments.
- Each person has a total of five days to be late without penalty for all the homeworks. Each late delivery less than one day will be counted as one day.

1. [30 points] *(PCA)* For the covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2 & \sigma^2\rho & 0 \\ \sigma^2\rho & \sigma^2 & \sigma^2\rho \\ 0 & \sigma^2\rho & \sigma^2 \end{pmatrix},$$

where $\sigma > 0$ and $-\frac{1}{\sqrt{2}} < \rho < \frac{1}{\sqrt{2}}$.

(a) Find the principal components of the covariance matrix $\mathbf{\Sigma}$.

(b) Compute $\sum_{i=1}^{3} \mathsf{Var}(z_i)$, where $\mathsf{Var}(z_i) = \mathbf{w}_i^T \mathbf{\Sigma} \mathbf{w}_i$ and $\mathbf{w}_i$ is the i-th principal component of $\mathbf{\Sigma}$.

Solution:

(a) Let $|\lambda \mathbf{I} - \mathbf{\Sigma}| = 0$. Then we have

$$\begin{vmatrix} \lambda - \sigma^2 & -\sigma^2\rho & 0 \\ -\sigma^2\rho & \lambda - \sigma^2 & -\sigma^2\rho \\ 0 & -\sigma^2\rho & \lambda - \sigma^2 \end{vmatrix} = 0,$$

that is

$$\left(\lambda - \sigma^2\right) - 2\left(\sigma^2\rho\right)^2\left(\lambda - \sigma^2\right) = 0.$$

We can get the eigenvalues

$$\begin{aligned} \lambda_1 &= \sigma^2, \\ \lambda_2 &= \sigma^2(1 + \sqrt{2}\rho), \\ \lambda_3 &= \sigma^2(1 - \sqrt{2}\rho). \end{aligned}$$

When $\rho = 0$, we have $\lambda_1 = \lambda_2 = \lambda_3$, that means any orthonormal basis can be the principal components.
When $\rho \neq 0$, we can solve the principal components as

$$\mathbf{w}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}, \mathbf{w}_2 = \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{2}}{2} \\ \frac{1}{2} \end{bmatrix}, \mathbf{w}_3 = \begin{bmatrix} \frac{1}{2} \\ -\frac{\sqrt{2}}{2} \\ \frac{1}{2} \end{bmatrix}.$$

When $\rho > 0$, the principal components are $\mathbf{w}_2, \mathbf{w}_1, \mathbf{w}_3$, and when $\rho < 0$, the principal components are $\mathbf{w}_3, \mathbf{w}_1, \mathbf{w}_2$.

(b) Observe

$$\begin{aligned}\sum_{i=1}^{3} \mathsf{Var}(z_i) &= \sum_{i=1}^{3} \mathbf{w}_i^T \mathbf{\Sigma} \mathbf{w}_i \\ &= \sum_{i=1}^{3} \lambda_i \mathbf{w}_i \mathbf{w}_i^T \\ &= \sum_{i=1}^{3} \lambda_i \\ &= 3\sigma^2.\end{aligned}$$

2. [40 points] (Eigenface) It has been shown that PCA can be solved by SVD optimally. In this question, you will do PCA on Yale B dataset. The algorithm is described in Algorithm 1.

Algorithm 1 PCA via SVD

Load the images as a matrix $\mathbf{X}$, where each column of $\mathbf{X}$ is a vectorized image matrix.
Compute the mean of the data $\boldsymbol{\mu}$.
Compute the SVD of the data matrix $\mathbf{X} - \boldsymbol{\mu}\mathbf{1}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$.
Sort the singular values in descending order, choose the first d singular values and the corresponding d left singular vectors $\mathbf{U}_d$.
Find the d principal components $\mathbf{Y} = \mathbf{U}_d^T(\mathbf{X} - \boldsymbol{\mu}\mathbf{1}^T)$.

The principal bases $\mathbf{U}$ estimated by PCA are also known as the eigenfaces in the computer vision literature. The first eigenface is the left singular vector corresponding to the largest singular value, the second eigenface is the left singular vector corresponding to the second largest singular value and so on.

(a) Apply NMF: $\mathbf{V} \approx \mathbf{MH}$ with rank $r = 10$ to the Training Set. Plot the mean face $\boldsymbol{\mu}$, the first basis element (the reshaped matrix of the first column of $\mathbf{M}$), the second basis element, the third basis element and the tenth basis element. (You can use sklearn.decomposition.NMF in Python or nnmf in Matlab to do NMF and you need to rescale your images to make them visible.)

(b) Apply PCA in Algorithm 1 with $d = 10$ to the Training Set. Plot the mean face $\boldsymbol{\mu}$, the first eigenface, the second eigenface, the third eigenface, the tenth eigenface and the first ten sorted singular values. (You can use numpy.linalg.svd in Python or svd in Matlab to do SVD and you need to rescale your images to make them visible.)

(c) Project the Test Set onto the face subspace given by PCA, that is $\mathbf{Y}_{\text{test}} = \mathbf{U}_d^T(\mathbf{X}_{\text{test}} - \boldsymbol{\mu}\mathbf{1}^T)$. Compute the projected faces, that is $\text{Proj}(\mathbf{X}_{\text{test}}) = \boldsymbol{\mu}\mathbf{1}^T + \mathbf{U}_d\mathbf{Y}_{\text{test}}$. Then choose one projected face for each individual and plot them. And show those projected faces for $d = 2$ again.

(Note: You need to export the code as a PDF file and then upload it onto Gradescope.)

Matrix Factorization

December 31, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import decomposition
import scipy.io as scio

plt.rcParams.update({'font.size': 8})

all_data = scio.loadmat("./data/all_data.mat")
X_train = all_data['data_train']
Y_train = all_data['Y_label_train']
X_test = all_data['data_test']
Y_test = all_data['Y_label_test']
```

[2]:
```
# (a)
mean = np.mean(X_train, 1)
model = decomposition.NMF(n_components = 10, init = 'nndsvd', max_iter = 10000)
W = model.fit_transform(X_train)

plt.figure(figsize = (15, 5))
plt.subplot(1, 5, 1)
plt.imshow(mean.reshape((42, 48)).T, cmap = 'gray')
plt.axis('off')
plt.title('mean face')
j = 2
titles = ['first', 'second', 'third', 'tenth']
for i in [0, 1, 2, 9]:
    plt.subplot(1, 5, j)
    plt.imshow(W[:, i].reshape((42, 48)).T, cmap = 'gray')
    plt.axis('off')
    plt.title('{} basis element'.format(titles[j - 2]))
    j = j + 1
plt.show()
```

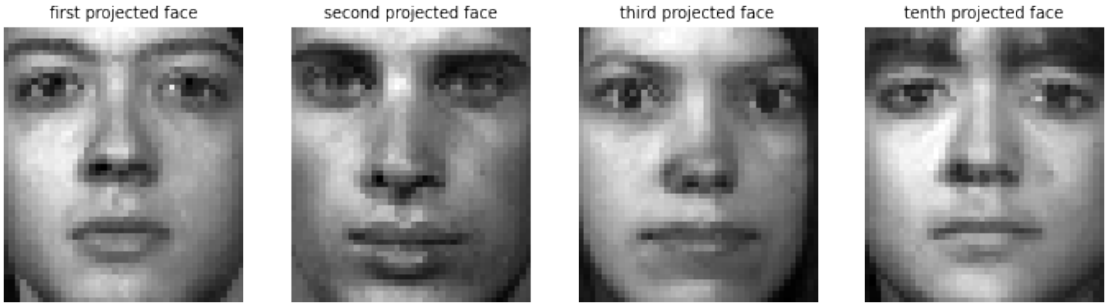
[3]:
```
#(b)
U, Sigma, V = np.linalg.svd(X_train - np.outer(mean, np.ones(40)))
print(Sigma[:10])

plt.figure(figsize = (15, 5))
plt.subplot(1, 5, 1)
plt.imshow(mean.reshape((42, 48)).T, cmap = 'gray')
plt.axis('off')
plt.title('mean face')
j = 2
titles = ['first', 'second', 'third', 'tenth']
for i in [0, 1, 2, 9]:
    plt.subplot(1, 5, j)
    plt.imshow(U[:, i].reshape((42, 48)).T, cmap = 'gray')
    plt.axis('off')
    plt.title('{} eigen face'.format(titles[j - 2]))
    j = j + 1
plt.show()
```

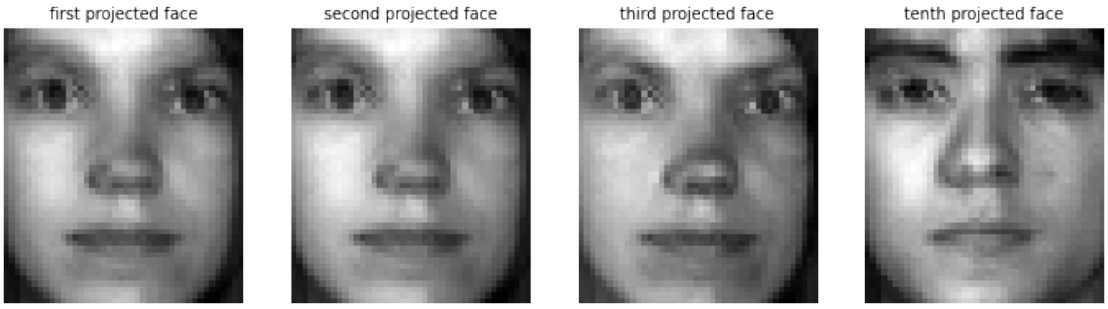
```
[38.86814203 21.26464132 15.17172002 11.42469483  9.2757139   8.12205094
  6.8112936   5.6962542   5.02487247  4.3993615 ]
```

[4]:
```
#(c)
#d = 10
plt.figure(figsize = (12, 5))
j = 1
titles = ['first', 'second', 'third', 'tenth']
```

```
for i in [0, 5, 10, 15]:
    plt.subplot(1, 4, j)
    plt.imshow((mean + np.dot(U[:,:10], np.dot(U[:,:10].T, X_test[:,i] - mean))).
 ↪reshape((42, 48)).T, cmap = 'gray')
    plt.axis('off')
    plt.title('{} projected face'.format(titles[j - 1]))
    j = j + 1
plt.show()
```



[5]:
```
#(d) = 2
plt.figure(figsize = (12, 5))
j = 1
titles = ['first', 'second', 'third', 'tenth']
for i in [0, 5, 10, 15]:
    plt.subplot(1, 4, j)
    plt.imshow((mean + np.dot(U[:,:2], np.dot(U[:,:2].T, X_test[:,i] - mean))).
 ↪reshape((42, 48)).T, cmap = 'gray')
    plt.axis('off')
    plt.title('{} projected face'.format(titles[j - 1]))
    j = j + 1
plt.show()
```



3. [30 points] *(Bayesian Regression)* Consider the likelihood

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}\left(y_i \mid \mathbf{w}^T \mathbf{x}_i, \alpha\right),$$

with prior on the parameters as

$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w} \mid \mathbf{0}, \beta \mathbf{I}\right),$$

where α and β are given constants. The posterior can be written as the product of the prior and the likelihood (up to a normalization constant) as follows

$$\arg\max_{\mathbf{w}} \ln p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) = \arg\max_{\mathbf{w}} \ln p(\mathbf{w}) + \ln p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}).$$

Show that maximizing the log posterior is equivalent to minimize a regularized loss function:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2 + \lambda \left\|\mathbf{w}\right\|_2^2,$$

and determine the corresponding λ expressed in terms of α and β.

Solution:

Observe that

$$\begin{aligned}
\arg\max_{\mathbf{w}} \ln p\left(\mathbf{w} \mid \mathbf{y}, \mathbf{X}\right) &= \arg\max_{\mathbf{w}} \ln p\left(\mathbf{w}\right) + \ln p\left(\mathbf{y} \mid \mathbf{X}, \mathbf{w}\right) \\
&= \arg\max_{\mathbf{w}} \ln \left(\frac{1}{\sqrt{(2\pi)^{\|\mathbf{w}\|_0 \, |\beta \mathbf{I}|}}} e^{-\frac{1}{2} \mathbf{w}^T \frac{1}{\beta} \mathbf{I} \mathbf{w}} \right) + \sum_{i=1}^{N} \ln \left(\frac{1}{\sqrt{2\pi\alpha}} e^{-\frac{\left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2}{2\alpha}} \right) \\
&= \arg\max_{\mathbf{w}} -\frac{1}{2} \mathbf{w}^T \frac{1}{\beta} \mathbf{I} \mathbf{w} - \sum_{i=1}^{N} \frac{\left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2}{2\alpha} \\
&= \arg\min_{\mathbf{w}} \alpha \mathbf{w}^T \mathbf{w} + \sum_{i=1}^{N} \beta \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2 \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2 + \lambda \left\|\mathbf{w}\right\|_2^2,
\end{aligned}$$

where $\lambda = \frac{\alpha}{\beta}$.