CS/ECE/ME532 Assignment 6

1. Suppose $\boldsymbol{A}$ is an n-by-n symmetric, *rank-one* matrix with right singular vectors $\boldsymbol{v}_k, k = 1, 2, \ldots, n$. Show that the power method converges to $\boldsymbol{v}_1$ and determine the number of iterations needed to converge within 1% of $\boldsymbol{v}_1$ if your initial vector $\boldsymbol{b}_0 = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

SOLUTION: Since $\boldsymbol{A}$ is rank one and symmetric, it's singular value (or eigen) decomposition can be written as $\boldsymbol{A} = \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T$. The first iteration is

$$\boldsymbol{b}_1 = \frac{\boldsymbol{A}\boldsymbol{b}_0}{||\boldsymbol{A}\boldsymbol{b}_0||_2} = \frac{\lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T \boldsymbol{b}_0}{||\lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T \boldsymbol{b}_0||_2}$$

Now note that $||\lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^T \boldsymbol{b}_0||_2 = |\lambda_1 \boldsymbol{v}_1^T \boldsymbol{b}_0|$ so

$$\boldsymbol{b}_1 = \boldsymbol{v}_1 \text{sign}\{\boldsymbol{v}_1^T \boldsymbol{b}_0\}$$

Thus in one iteration the power method converges to the correct singular vector. The sign doesn't matter, since if $\boldsymbol{v}_1$ is a singular vector, then $-\boldsymbol{v}_1$ is also a singular vector.

2. Use the starter script available for download on the course page. The script loads the 1000 three-dimensional data points in `sdata.csv` into a 1000-by-3 matrix `X`. The second section of the code displays the data using a scatter plot format. We wish to approximate the data using a subspace. Use the rotate tool to view the data cloud from different perspectives.

a) Does the data appear to lie in a low-dimensional subspace? Why or why not? Remember the definition of a subspace.

b) What could you do to the data so that it lies (approximately) in a low-dimensional subspace?

c) The third section of the code removes the mean (average) value of the 1000 data points. Use the rotate tool to inspect the scatterplot of the data with the mean removed. Does the mean-removed data appear to lie in a low-dimensional subspace?

d) The fourth section of the code finds the SVD of the mean-removed data. If `a` is a unit-norm vector representing the best one-dimensional subspace for the mean-removed data, complete the line of code to define `a` in terms of the SVD matrices. The fifth section displays the one-dimensional subspace with the mean-removed

data. Note that the length of the vector representing the subspace is scaled by the root-mean-squared value of the data for display purposes. Use the rotate tool to inspect the relationship between the subspace and the data, and comment on how well a one-dimensional subspace captures the data.

e) Let $\boldsymbol{x}_{zi}, i = 1, 2, \ldots, 1000$ be the individual mean-removed data points and $\boldsymbol{a}$ the unit-norm vector representing the best one-dimensional subspace for the data. Thus, $\boldsymbol{x}_{zi} \approx \boldsymbol{a} w_i$. Find w_i in terms of the SVD matrices $\boldsymbol{U}$, $\boldsymbol{S}$, and $\boldsymbol{V}$.

f) Now write the original data $\boldsymbol{x}_i, i = 1, 2, \ldots, 1000$ as $\boldsymbol{x}_i \approx \boldsymbol{a} w_i + \boldsymbol{b}$. What is $\boldsymbol{b}$?

g) Let $\boldsymbol{E}$ be the difference between $\boldsymbol{X}$ and the rank-one approximation. Find a mathematical expression for $||\boldsymbol{E}||_F^2$ in terms of the singular values of the mean-removed data $\boldsymbol{X}_z$.

h) Now try a rank-two approximation. Use the SVD to find an orthonormal basis for the best plane containing the mean-removed data. Display the mean-removed data and the bases for the plane.

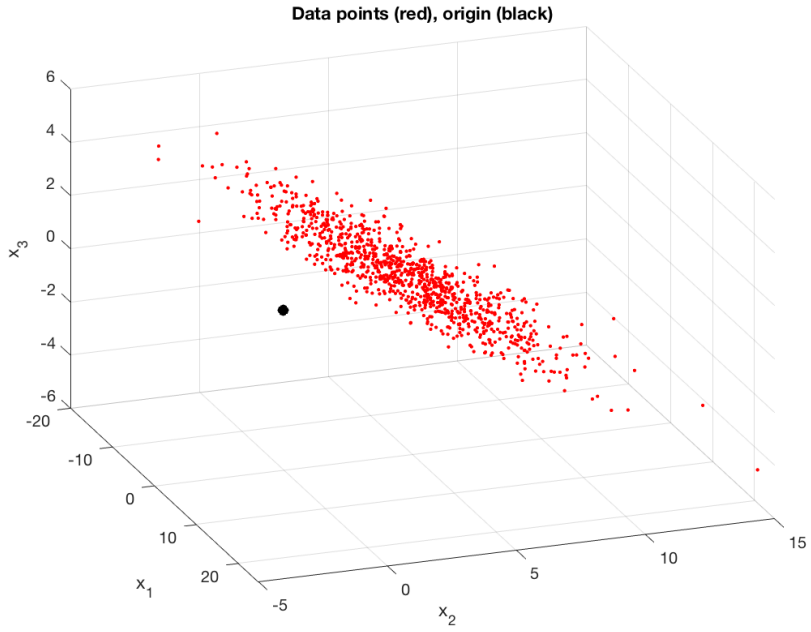
i) Your rank-two approximation for the original data is $\boldsymbol{x}_i \approx \boldsymbol{a}_1 w_{1i} + \boldsymbol{a}_2 w_{2i} + \boldsymbol{b}, i = 1, 2, \ldots, 1000$. Express $w_{2i}, i = 1, 2, \ldots, 1000$ in terms of the SVD of the mean-removed data matrix $\boldsymbol{X}_z$. Display a scatter plot of the original data (in red) and the rank-two approximations in blue. Does the rank-two approximation lie in a plane? Does that plane capture the dominant components of the data?

j) Let $\boldsymbol{E}$ be the difference between $\boldsymbol{X}$ and the rank-two approximation. Find a mathematical expression for $||\boldsymbol{E}||_F^2$ in terms of the singular values of the mean-removed data $\boldsymbol{X}_z$.

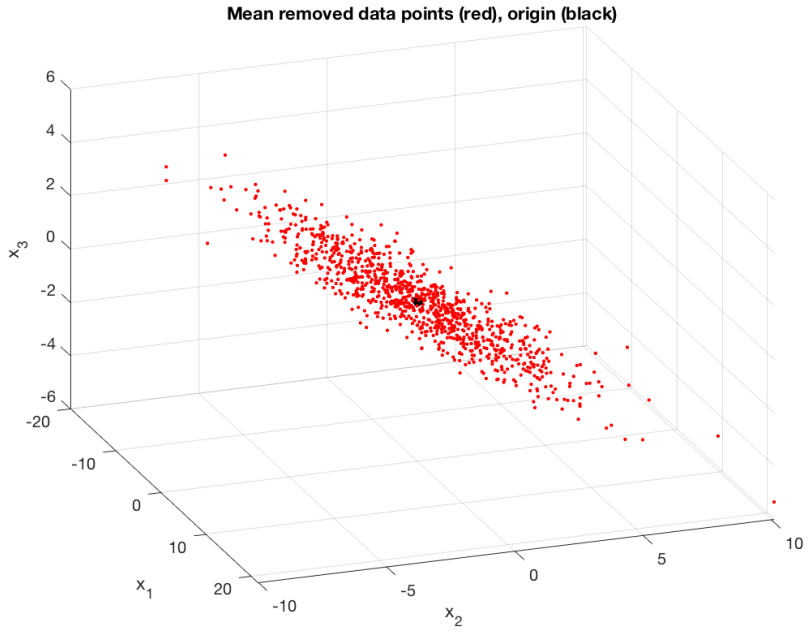
k) Find and compare the numerical values for $||\boldsymbol{E}||_F^2$ using both the rank-1 and rank-2 approximation.

SOLUTION:

a) The data appears to be concentrated along a line, or even more so in a plane, but the line/plane does not include the origin so it cannot be a subspace.



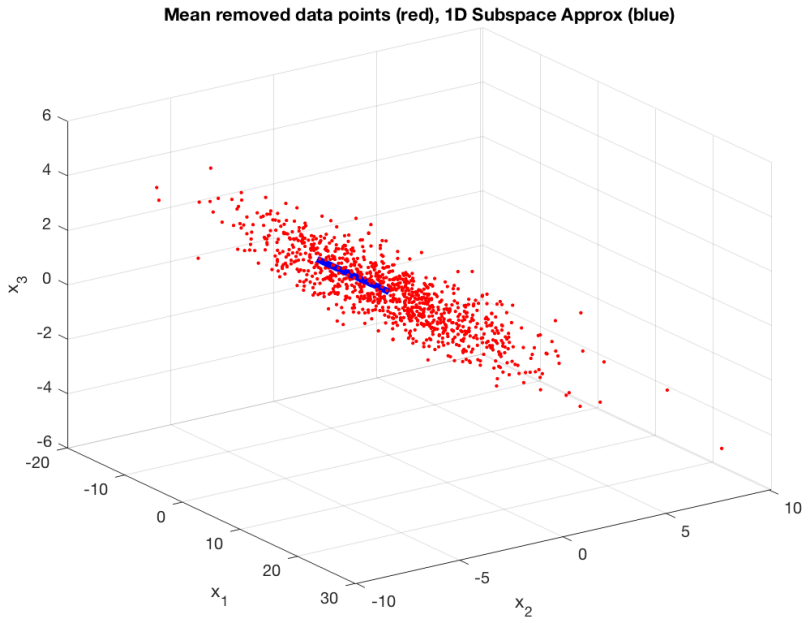
b) Remove the average value of the data. This will center the cloud on the origin and a line/plane approximation will then include the origin.



c) Yes, a line through the origin captures the majority of the variability in the data, and a plane even more.

d) `a = V(:,1)`

The line lines up with the major axis of the data point cloud. See plot.



e) $\boldsymbol{x}_{zi}$ is the ith row of the matrix $\boldsymbol{X}_z$. The rank-1 approximation to $\boldsymbol{X}_z$ is $\boldsymbol{X}_z \approx \boldsymbol{U}_1 * \sigma_1 * \boldsymbol{V}_1^T$ where $\boldsymbol{U}_1$ and $\boldsymbol{V}_1$ are the left and right singular vectors associated with the largest singular value σ_1. Thus, $w_i = [\boldsymbol{U}_1]_i \sigma_1$ where $[\boldsymbol{U}_1]_i$ is the ith entry in $\boldsymbol{U}_1$.

f) $\boldsymbol{b}$ is the mean that was removed from the original data. Note that $\boldsymbol{x}_i = \boldsymbol{x}_{zi} + \boldsymbol{b}$.

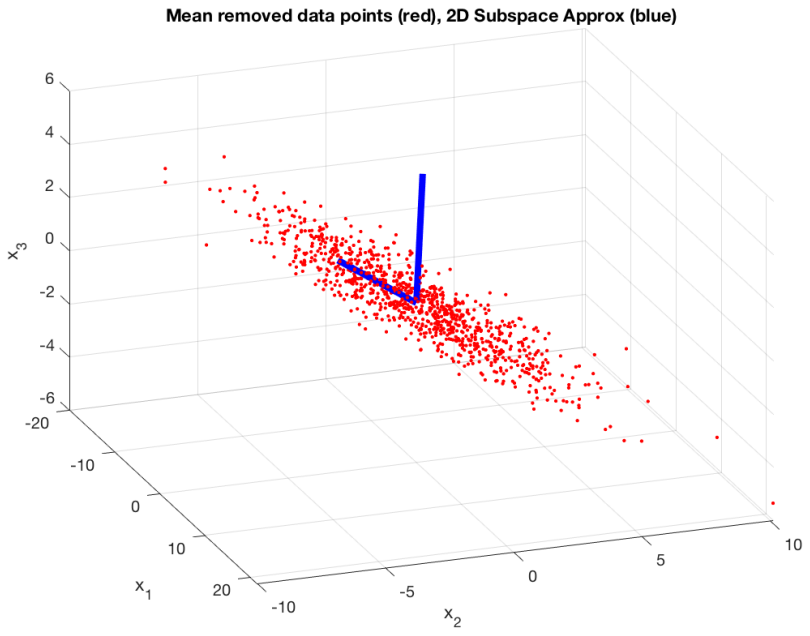
g) We have $\boldsymbol{X} = \sum_{i=1}^{3} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ and $\boldsymbol{X}_1 = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$, so $\boldsymbol{E} = \sum_{i=2}^{3} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$. Also, the Frobenius norm squared is the sum of the squares of the singular values, so $||\boldsymbol{E}||_F^2 = \sigma_2^2 + \sigma_3^2$.

h) Following the steps in the previous part using a rank-2 approximation gives $||\boldsymbol{E}||_F^2 = \sigma_3^2$. See plot.

i) We can write $\boldsymbol{X}_z \approx \boldsymbol{U}_1 * \sigma_1 * \boldsymbol{V}_1^T + \boldsymbol{U}_2 * \sigma_2 * \boldsymbol{V}_2^T$. Thus extracting the ith row of $\boldsymbol{X}_z$ and rewriting it is a column vector we have $\boldsymbol{x}_z i = \boldsymbol{V}_1 * \sigma_1 * [\boldsymbol{U}_1]_i + \boldsymbol{V}_2 * \sigma_2 * [\boldsymbol{U}_2]_i$ where $[\boldsymbol{U}_1]_i$ is the ith entry in $\boldsymbol{U}_1$ and $[\boldsymbol{U}_2]_i$ is the ith entry in $\boldsymbol{U}_2$. Hence identifying $\boldsymbol{a}_1 = \boldsymbol{V}_1$ and $\boldsymbol{a}_2 = \boldsymbol{V}_2$, we have $w_{1i} = [\boldsymbol{U}_1]_i \sigma_1$ and $w_{2i} = [\boldsymbol{U}_2]_i \sigma_2$.

j) From a previous activity we know that $||\boldsymbol{E}||_F^2 = \sigma_3^2$ where σ_3 is the smallest singular value of the 3-by-1000 matrix $\boldsymbol{X}_z$.

k) Rank-1 squared error is 626.69. Rank-2 absolute error is 152.95. Normalizing



these by $||\boldsymbol{X}_z||_F^2$ gives relative squared errors of 0.023 and 0.006, respectively.

3. Consider the face emotion classification problem studied previously. Design and compare the performances of the classifiers proposed in **a** and **b**, below. In each case, divide the dataset into 8 equal sized subsets (e.g., examples $1-16$, $17-32$, etc). Use 6 sets of the data to estimate $\boldsymbol{w}$ for each choice of the *regularization parameter*, then select the best value for the regularization parameter by estimating the error on one of the two sets of data held out from training, and finally use the $\boldsymbol{w}$ corresponding to the best value of the regularization parameter to predict the labels of the remaining set of data that was held out. Compute the number of mistakes made on this hold-out set and divide that number by 16 (the size of the set) to estimate the error rate. Note there are $8 \times 7 = 56$ different choices of the two hold-out sets, so repeat this process 56 times and average the error rates across the 56 cases to obtain a final estimate.

a) Truncated SVD. Use the pseudo-inverse $\boldsymbol{V}\boldsymbol{\Sigma}_r^{-1}\boldsymbol{U}^T$, where $\boldsymbol{\Sigma}_r^{-1}$ is computed by inverting the r largest singular values. Hence the regularization parameter r takes values $r = 1, 2, \ldots, 9$.

b) Ridge Regression. Let $\widehat{\boldsymbol{w}}_\lambda = \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$, for the following values of the regularization parameter $\lambda = 0, 2^{-1}, 2^0, 2^1, 2^2, 2^3$, and 2^4. Show that $\widehat{\boldsymbol{w}}_\lambda$ can be computed using the SVD and use this fact in your code.

SOLUTION: Running the code below, we find an average error rate of `0.1116` for the truncated SVD and an average error rate of `0.0480` for ridge regression.

So ridge regression appears to result in a better classifier in this problem.

In [19]:
```
import numpy as np
import scipy.io as sio
data = sio.loadmat('face_emotion_data.mat')
X, y = data['X'], data['y']

err_sum = 0
for i in range(8):
    for j in range(8):
        if i == j: continue
        test_idx_1 = np.arange(i*16, (i+1)*16)
        test_idx_2 = np.arange(j*16, (j+1)*16)
        train_idx = np.setdiff1d(np.arange(128), test_idx_1)
        train_idx = np.setdiff1d(train_idx, test_idx_2)
        X_train, y_train = X[train_idx, :], y[train_idx, :]
        X_test_1, y_test_1 = X[test_idx_1, :], y[test_idx_1, :]
        X_test_2, y_test_2 = X[test_idx_2, :], y[test_idx_2, :]
        min_err, min_r, min_w = np.inf, -1, None
        for r in range(1,10):
            U, s, VT = np.linalg.svd(X_train)
            w = VT[:r, :].T@np.diag(1/s[:r])@U[:,:r].T@y_train
            err_ = np.mean(np.sign(X_test_1@w) != y_test_1)
            if err_< min_err:
                min_err, min_r, min_w = err_, r, w

        err_sum += np.mean(np.sign(X_test_2@min_w) != y_test_2)

print(err_sum/8/7)
```

```
0.11160714285714286
```

In [26]:
```
import numpy as np
import scipy.io as sio
data = sio.loadmat('face_emotion_data.mat')
X, y = data['X'], data['y']

err_sum = 0
for i in range(8):
    for j in range(8):
        if i == j: continue
        test_idx_1 = np.arange(i*16, (i+1)*16)
        test_idx_2 = np.arange(j*16, (j+1)*16)
        train_idx = np.setdiff1d(np.arange(128), test_idx_1)
        train_idx = np.setdiff1d(train_idx, test_idx_2)
        X_train, y_train = X[train_idx, :], y[train_idx, :]
        X_test_1, y_test_1 = X[test_idx_1, :], y[test_idx_1, :]
        X_test_2, y_test_2 = X[test_idx_2, :], y[test_idx_2, :]
        min_err, min_r, min_w = np.inf, -1, None
        for la in [0]+[2.**i for i in range(-1,5)]:
            U, s, VT = np.linalg.svd(X_train, full_matrices=False)
            w = VT.T@np.diag(s/(s**2+la))@U.T@y_train
            err_ = np.mean(np.sign(X_test_1@w) != y_test_1)
            if err_< min_err:
                min_err, min_r, min_w = err_, r, w
        err_sum += np.mean(np.sign(X_test_2@min_w) != y_test_2)


print(err_sum/8/7)
```

```
0.04799107142857143
```