Ayan Deep Hazra ECE 532

Period 12 Activity

1. a) Given SVD, $X = U \Sigma V^T$

Thus

$$(X)^T = (U \Sigma V^T)^T$$

$$X^T = (V^T)^T (\Sigma)^T (U)^T$$

$$X^T = V \Sigma^T U^T$$

but since Σ is a diagonal matrix, we have,

$$X^T = V \Sigma U^T \text{ or } Z = V \Sigma U^T$$

b) The rows of Z are the columns of X as $Z = X^T$.

Since $X = U \Sigma V^T$ SVD, we know that the columns of U act as orthonormal basis for X. The first column of U acts as a rank 1 subspace to approximate columns of X.

→ Thus first column of U acts as rank-1 subspace to approximate rows of Z in terms of $U \Sigma V^T$.

2. a) Since X is n-by-p with $p<n$.

The least squares problem $\min_w \|y - Xw\|_2^2$ does not have unique solution if $\operatorname{rank}(X) < p$.

b) $\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2$

As $\left\| \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \right\|_2^2 = \|z_1\|_2^2 + \|z_2\|_2^2$,

we have

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2 =$$

$$\min_w \left\| \begin{bmatrix} y - Xw \\ \sqrt{\lambda} w \end{bmatrix} \right\|_2^2 = \min_w \left\| \begin{bmatrix} y \\ 0 \end{bmatrix} - \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} w \right\|_2^2$$

Thus $\hat{y} = \begin{bmatrix} y \\ 0 \end{bmatrix}$ & $\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix}$

c) We can write $\hat{X}$ as $\begin{bmatrix} & X & & \\ \sqrt{\lambda} & 0 & 0 & \cdots \\ 0 & \sqrt{\lambda} & 0 & \cdots \\ 0 & 0 & \sqrt{\lambda} & \\ 0 & \cdots & \cdots & \sqrt{\lambda} \end{bmatrix}$

We see that the rank of X is the same as the rank of $\hat{X}$ as the new rows appended to X do not change the relative independence of any rows or columns in X. Basically, if two columns/rows were independent before, they are independent now and if they were dependent before, they are still dependent.

Thus $\underline{\text{rank}(\hat{X}) = \text{rank}(X)}$

→ Thus the $\text{rank}(\hat{X}) < p$ condition from (a) holds for checking if LS problem has unique solution or not.

3.) $X^{+} = \lim_{\lambda \downarrow 0} (X^T X + \lambda I)^{-1} X^T$

we know, $X = U \Sigma V^T,$ $\qquad X^{n\times p} = U^{n\times p} \Sigma^{p\times p} V^{T\,p\times p}$

$X^T X = V \Sigma^2 V^T,$

$\lambda I = V \lambda I V^T$

a) $(X^T X + \lambda I)^{-1} X^T$

$= (V\Sigma^2 V^T + V\lambda I V^T)^{-1} V \Sigma U^T$

$= (V(\Sigma^2 + \lambda I)V^T)^{-1} V\Sigma U^T$

$= (V^T)^{-1}(\Sigma^2 + \lambda I)^{-1} \cancel{V^{-1}} \cancel{V} \Sigma U^T$

$= V(\Sigma^2 + \lambda I)^{-1} \Sigma U^T$

$$= V \begin{bmatrix} \frac{1}{\sigma_i^2 + \lambda} & 0 & 0 \\ 0 & \ddots & \\ 0 & & \ddots \end{bmatrix} \Sigma U^T$$

$$= V \begin{bmatrix} \frac{\sigma_i}{\sigma_i^2 + \lambda} & 0 & 0 \\ 0 & \ddots & \\ 0 & & \ddots \end{bmatrix} U^T$$

Since these matrices are rank p

& $VU^T = \sum_{i=1}^{p} v_i u_i^T$

we can write,

$$V \begin{bmatrix} \frac{\sigma_i}{\sigma_i^2+\lambda} & 0 & 0 \\ 0 & \ddots & \\ 0 & & \ddots \end{bmatrix} U^T = \sum_{i=1}^{p} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

b) $$X^+ = \lim_{\lambda \to 0} (X^T X + \lambda I)^{-1} X^T$$

$$= \lim_{\lambda \to 0} \sum_{i=1}^{p} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

$$= \sum_{i=1}^{p} \frac{1}{\sigma_i} v_i u_i^T$$

we have,

$$(X^T X)^{-1} X^T = (V \Sigma^2 V^T)^{-1} (U \Sigma V^T)^T$$

$$= (V^T)^{-1} \Sigma^{-2} \cancel{V^{-1}} \cancel{V} \Sigma^T U^T$$

$$= V \Sigma^{-2+1} U^T$$

$$= V \Sigma^{-1} U^T = \sum_{i=1}^{p} \frac{1}{\sigma_i} v_i u_i^T$$

Thus, $X^{+} = (X^T X)^{-1} X^T$

c) from (b)

$X^{+} = (X^T X)^{-1} X^T$

since X is invertible X^T is also invertible by definition,

Thus,

$X^{+} = X^{-1} \cancel{(X^T)^{-1}} \cancel{X^T}$

Thus proven $X^{+} = X^{-1}$

d) we know, if X is rank p,

$$(X^TX + \lambda I)^{-1} X^T = \sum_{i=1}^{p} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

thus for rank $r < p$, we just take the first r singular values & the associated first r rows/columns from v_i & u_i^T

to get

$$(X^TX + \lambda I)^{-1} X^T = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

e) $$\lim_{\lambda \to 0} X^+ = \lim_{\lambda \to 0} \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

$$= \sum_{i=1}^{r} \lim_{\lambda \to 0} \frac{\sigma_i}{\sigma_i^2+\lambda} v_i u_i^T$$

$$= \sum_{i=1}^{r} \frac{1}{\sigma_i} v_i u_i^T$$

$$= v_i \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & & \\ \vdots & & \ddots & \\ 0 & & & \frac{1}{\sigma_r} \end{bmatrix} u_i^T$$

$$= V \Sigma_r^{-1} U^T$$

4. a) Yes the data appears to be close to a 1D subspace (it looks like a line).

The data is not zero mean, i.e. it is not setup so that the average point is at the origin.

b) A one-dimensional subspace is a reasonable fit to the data.

In positive x_2 & x_3 it does seem that there is some deviation from the line of best fit. The error is thus high in that region. The rest is reasonably well approximated.

c) The dominant feature no longer continues to be a good fit to the data. It infact aligns itself perpendicular to the previous feature.

→ The PCA calculates a new projection of your data set. And the new axes are based on the standard deviation of your variable. Data that is not normalized will have points with high standard deviations, by virtue of them being very far from the origin & having no counter weight to balance their effect.

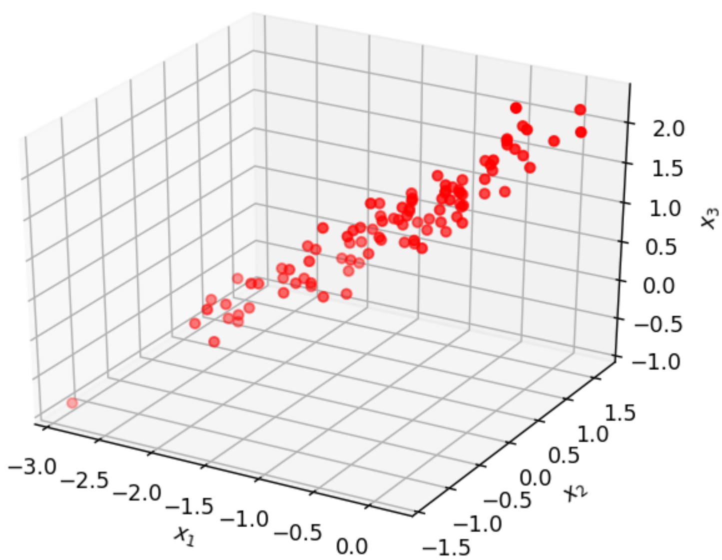
If we normalize all the datapoints, all variables have the same standard deviation, thus all variables have equal weights & PCA gives a good approximation of dominant feature.

In [1]:
```
# Enable interactive rotation of graph
%matplotlib notebook

import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Load data for activity
X = loadmat('PCA_Activity.mat')['X']
rows, cols = np.array(X.shape)
x, y, z = X

print('Rows of X = ',rows)
print('Cols of X = ',cols)
```

```
Rows of X =  3
Cols of X =  100
```

In [2]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x, y, z, c='r', marker='o')

ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$x_3$')

plt.show()
```

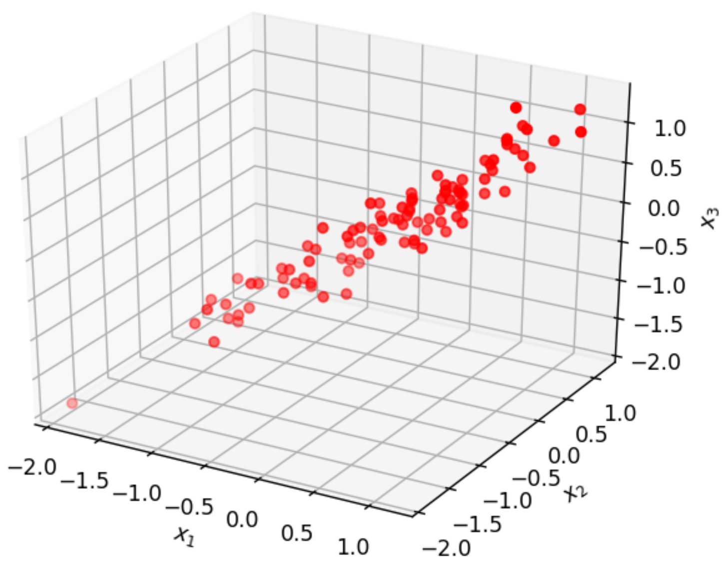
Figure 1



In [3]:
```
# Subtract mean
X_m = X - np.mean(X, 1).reshape((3,1))
x_m, y_m, z_m = X_m
```

In [4]:
```python
# display zero mean scatter plot

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x_m, y_m, z_m, c='r', marker='o')

ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$x_3$')

plt.show()
```

Figure 2

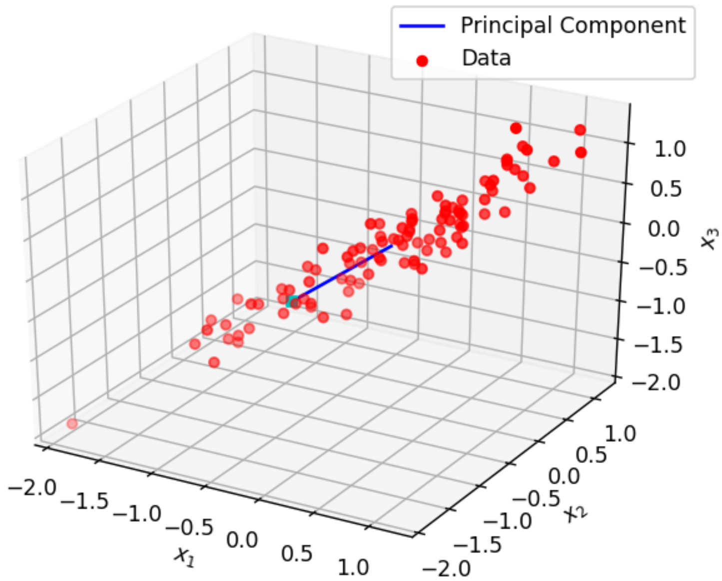


In [5]:
```python
# Use SVD to find first principal component

U,s,VT = np.linalg.svd(X_m,full_matrices=False)

# complete the next line of code to assign the first principal component to a
a = U[:,[0]]

print(a)
```

```
[[-0.58277194]
 [-0.57701087]
 [-0.57221964]]
```

In [6]:
```
# display zero mean scatter plot and first principal component

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x_m, y_m, z_m, c='r', marker='o', label='Data')

ax.scatter(a[0],a[1],a[2], c='c', marker='s')

ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$x_3$')

ax.plot([0,a[0]],[0,a[1]],[0,a[2]], c='b',label='Principal Component')

ax.legend()
plt.show()
```

Figure 3



Forward to next view

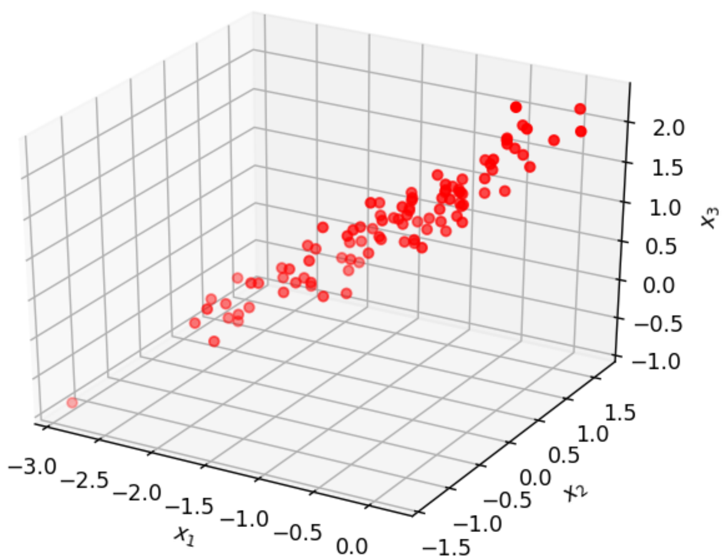
```
C:\Users\Ayan Deep Hazra\miniconda3\Lib\site-packages\numpy\lib\stride_tricks.py:116: VisibleDeprecatio
nWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples
-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  array = np.array(array, copy=False, subok=subok)
C:\Users\Ayan Deep Hazra\miniconda3\Lib\site-packages\numpy\core\_asarray.py:136: VisibleDeprecationWar
ning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or
ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dt
ype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order, subok=True)
C:\Users\Ayan Deep Hazra\miniconda3\Lib\site-packages\numpy\core\_asarray.py:83: VisibleDeprecationWarn
ing: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or n
darrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dty
pe=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
```

Same process but with mean removed

In [7]:
```python
# Subtract mean
X_m = X #- np.mean(X, 1).reshape((3,1))
x_m, y_m, z_m = X_m
```

In [8]:
```python
# display zero mean scatter plot

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x_m, y_m, z_m, c='r', marker='o')

ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$x_3$')

plt.show()
```

Figure 4



In [9]:
```python
# Use SVD to find first principal component

U,s,VT = np.linalg.svd(X_m,full_matrices=False)

# complete the next line of code to assign the first principal component to a
a = U[:,[0]]

print(a)
```

```
[[-0.57725541]
 [ 0.39008946]
 [ 0.71736072]]
```

In [10]:
```
# display zero mean scatter plot and first principal component

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x_m, y_m, z_m, c='r', marker='o', label='Data')

ax.scatter(a[0],a[1],a[2], c='c', marker='s')

ax.set_xlabel('$x_1$')
ax.set_ylabel('$x_2$')
ax.set_zlabel('$x_3$')

ax.plot([0,a[0]],[0,a[1]],[0,a[2]], c='b',label='Principal Component')

ax.legend()
plt.show()
```

Figure 5