Machine Learning for Mechanical Engineering

Homework 5

Due Monday, 04/30/2018, 9:30 AM

by Prof. Seungchul Lee
Industrial AI Lab
POSTECH

- For your handwritten solution, hand in your hardcopy at class.
- For your code, email your .ipynb file to (`iai.postech@gmail.com`)
- When you send an e-mail, write down [Machine Learning HW5] on the title
- And please write your NAME on your .ipynb files. ex) 김지원_20182315_HW5.ipynb
- Do not submit a printed version of your code. It will not be graded.

Problem 1

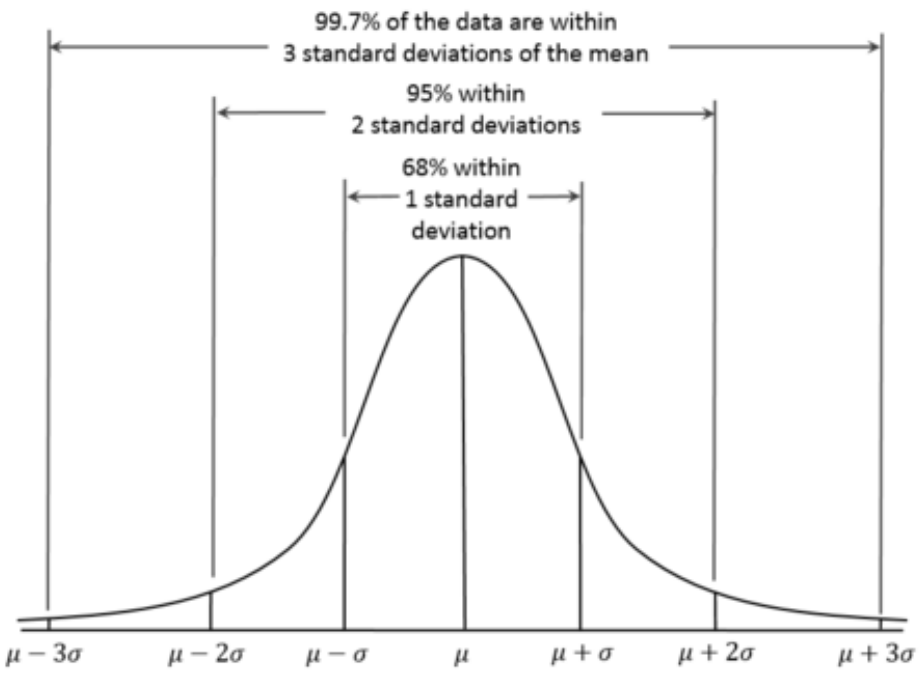
Find the correlation coefficient between random variables X and $Y = aX + b$. Explain the meaning of a correlation coefficient with the example of random variables X and $Y = aX + b$.

Problem 2

The probability of values that lie within a band $[\mu - 2\sigma, \mu + 2\sigma]$ of Gaussian distribution $N(\mu, \sigma^2)$ is 95.45%. In mathematical expression

$$P(\mu - 2\sigma \leq x \leq \mu + 2\sigma) \approx 0.9545$$

We want to numerically show it with random numbers in python. Note that this method is known as the Monte Carlo method.



1) Generate a random samples using `randn` command in python.

In [1]:

```
## your code here
```

2) Count the number of samples which fall into a band $[\mu - 2\sigma, \mu + 2\sigma]$.

In [2]:

```
## your code here
```

3) Compute the ratio of samples that fall into a band to the total number of samples.

In [3]:

```
## your code here
```

4) Plot the graph to show how the ratio changes as the number of samples (i.e., sample size) increases.

In [4]:

```
## your code here
```

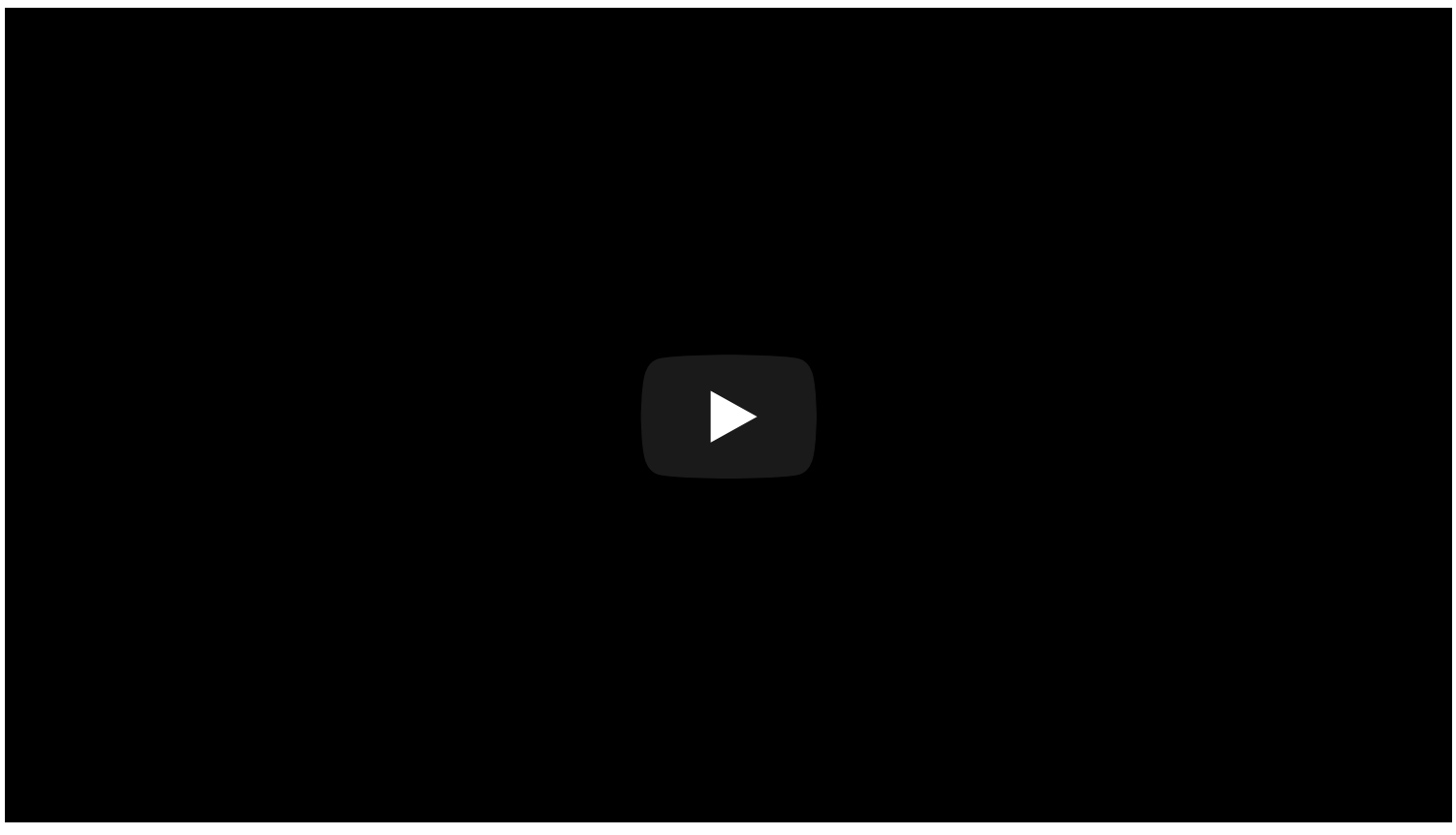
Problem 3

Monty Hall Problem (http://en.wikipedia.org/wiki/Monty_Hall_problem) with Monte Carlo Simulations

Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

In [5]:

```
%%html
<center><iframe width="560" height="315" src="https://www.youtube.com/embed/HH
-676dOCFs?rel=0"
frameborder="0" allowfullscreen></iframe></center>
```



Solve this problem with Monte Carlo simulation and print out the winning rate about 'Staying' and 'Switching'. You can utilize the below functions

In [6]:

```
import numpy as np
from random import shuffle, choice
```

In [7]:

```
np.random.randint(3)
```

Out[7]:

```
1
```

`shuffle(list)` : randomly shuffle the positioin of elements in list

In [8]:

```
doors = ['car', 'goat', 'goat']
print(doors)
shuffle(doors)
print(doors)
```

```
['car', 'goat', 'goat']
['car', 'goat', 'goat']
```

`choice(list)`: choice random element in list

In [9]:

```
choice([0, 1, 2])
```

Out[9]:

```
0
```

In [10]:

```
## your code here
```

Sample output

```
Start simulation ...

Winning rate
Staying: 33.396100%
Switching: 66.603900%
```

Problem 4

Download this data reg_pca_data (https://www.dropbox.com/sh/n56f2y2vo9t876l/AADExG_yeB-druiUAQgSXo-za?dl=1)

In [11]:

```
from six.moves import cPickle

x_reg = cPickle.load(open('./image_files/x_reg.pkl', 'rb'))
y_reg = cPickle.load(open('./image_files/y_reg.pkl', 'rb'))
```

1) Write a python code to conduct a linear regression (least square).

In [12]:

```
## your code here
```

2) Write a python code to perform PCA. Since `mean(x) = mean(y) = 0`, normalization step (mean subtraction and rescaling) can be skipped.

In [13]:

```
## your code here
```

3) Comment on why the results are different as shown in Figure 6?

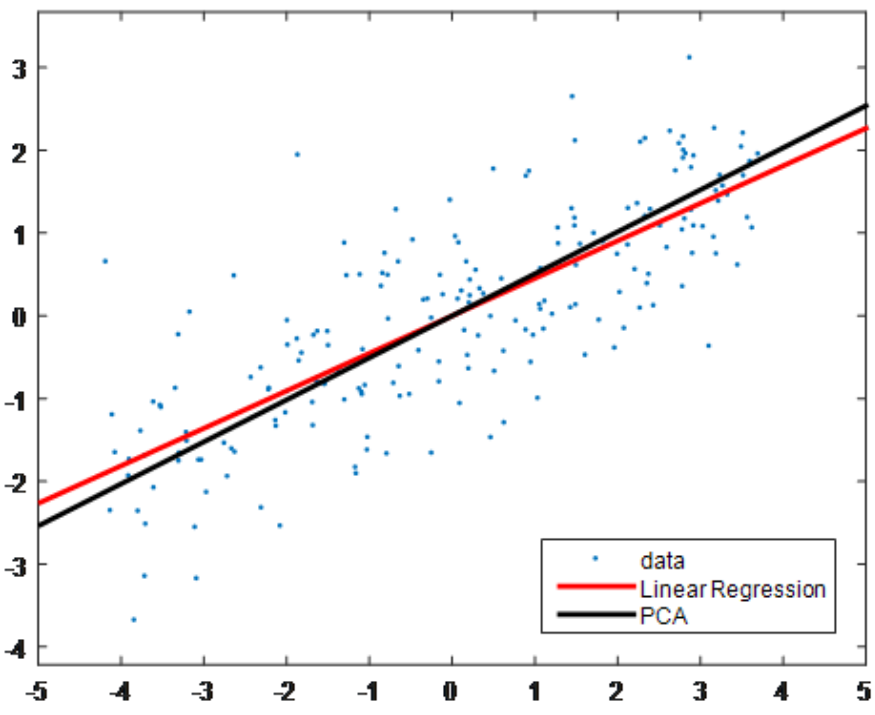


Figure 6

In [14]:

```
## your code here
```

Problem 5

We want to find a major axis in a different way from what we learned in PCA class. Load PCA_data (https://www.dropbox.com/sh/dk7ls0emvskeq2g/AADP7BEmqSZc72g-g6NpPX9Sa?dl=1).

1) Plot the principal component (unit vector u_1) with the largest variance using PCA (i.e., eigen-analysis).

In [15]:

```
from six.moves import cPickle

X = cPickle.load(open('./image_files/X.pkl', 'rb'))
```

In [16]:

```
## your code here
```

2) Create an arbitrary unit vector u. Then, multiply the sample variance matrix to u. After each multiplication, normalize u to make it an unit vector and observe (and plot) the direction of vector u. Compare it with the principal vector u_1 computed in 1). Are they pointing at the same (or opposite) direction? If the answer is yes, explain why they are parallel. (parallel = point either at the same or opposite direction)

In [17]:

```
## your code here
```

Problem 6

1) Solove the problem with Fisher Discriminant Analysis and plot the projection line.

In [18]:

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

%matplotlib inline
```

In [19]:

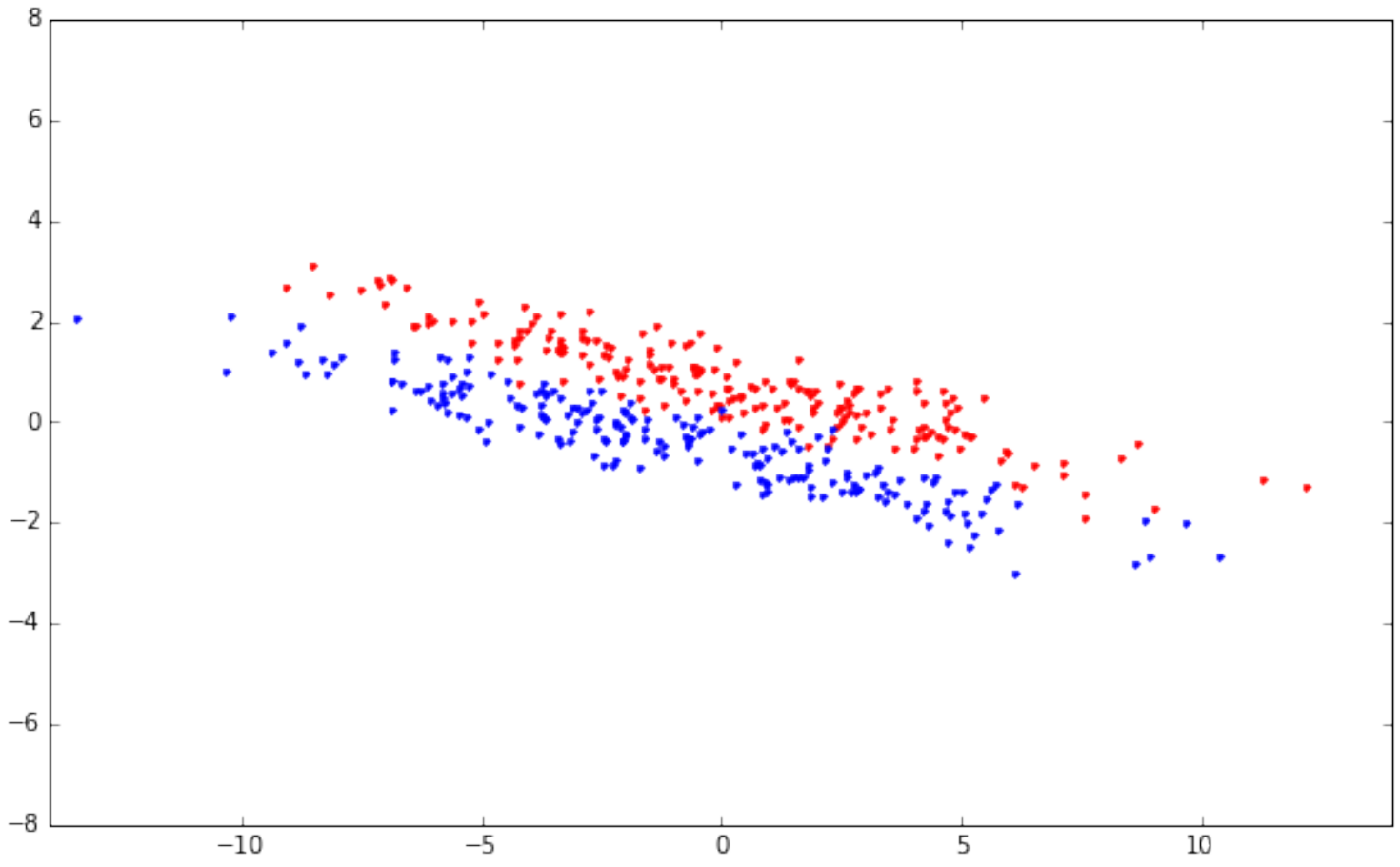
```
n0 = 200
n1 = 200

sigma = [[19, -4],
         [-4, 1]]

x0 = np.random.multivariate_normal([0.7,0.7], sigma, n0)        # data in clas
s 0
x1 = np.random.multivariate_normal([-0.5,-0.5], sigma, n1)      # data in clas
s 0

x0 = np.asmatrix(x0)
x1 = np.asmatrix(x1)

plt.figure(figsize = (10, 6))
plt.plot(x0[:,0],x0[:,1],'r.')
plt.plot(x1[:,0],x1[:,1],'b.')
plt.ylim([-8, 8])
plt.xlim([-14, 14])
plt.show()
```



In [20]:

```
#your code here
```

2) Solove the above problem with PCA and plot the projection line. In the case of PCA, you do not need to consider labels.(*i.e.* data class)

In [21]:

```
## your code here
```