Linear Regression based on Guassian model

input data : Slump test for concrete (http://archive.ics.uci.edu/ml/datasets/Condition+Based+Maintenance+of+Naval+Propulsion+Plants# (http://archive.ics.uci.edu/ml/datasets/Condition+Based+Maintenance+of+Naval+Propulsion+Plants#))

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from numpy.linalg import inv
from numpy.linalg import matrix_rank
from sklearn.metrics import explained_variance_score
```

Read input data:

In [2]:
```
ip_data = pd.read_csv("slump_test.data")
ip_data = ip_data.set_index("No")
```

Divide the input data into two sets as samples for input and output random vectors :

Output random vector represents last three columns and input is the remaining columns

In [3]:
```
x_size = 7
y_size = 3
total_vars = x_size+y_size
input_dist = ip_data.iloc[:, 0:x_size]
output_dist = ip_data.iloc[:, x_size:total_vars]

X_train, X_test, Y_train, Y_test = train_test_split(input_dist.values, output_di
st.values, test_size=0.2, random_state=1)
```

Get the parameters for the conditional Gaussian p(Y/X=x):

In [4]:
```
#First get the parameters for joint gaussian using method of moments
mu_Y = Y_train.mean(axis=0)
mu_X = X_train.mean(axis=0)
input_cov_matrix = np.cov(np.transpose(np.concatenate([X_train,Y_train], axis=1)
))
cov_XX = input_cov_matrix[0:x_size, 0:x_size]
cov_XY = input_cov_matrix[0:x_size, x_size:total_vars]
cov_YX = input_cov_matrix[x_size:total_vars, 0:x_size]
cov_YY = input_cov_matrix[x_size:total_vars, x_size:total_vars]
```

Covariance parameter for p(Y/X=x):

In [5]:
```
cov_Y_x = cov_YY - np.dot(np.dot(cov_YX, inv(cov_XX)), cov_XY)
```

Mean parameter for p(Y/X=x):

In [6]:
```
def estimate_Y_given_x_mean(x):
    '''
    When sample x is given , find the estimated mean of y/x with formula
    mu(y/x) = mu(y) + Cov(y,X) * Cov(X,X) * (x - mu(X))
    '''
    global cov_XX, cov_YX, mu_Y, mu_X
    return (mu_Y + np.dot(np.dot(cov_YX, inv(cov_XX)), (x-mu_X)))
```

Estimate parameters for test data using the p(Y/X=x):

In [7]:
```
Y_predicted = np.apply_along_axis(estimate_Y_given_x_mean, 1, X_test)
for i in range(Y_test.shape[1]):
    print("Explained variance for Y{0} : {1}".format(i,explained_variance_score(
Y_test[:,i], Y_predicted[:,i])))
```

```
Explained variance for Y0 : 0.3635419587805233
Explained variance for Y1 : 0.5255239777932958
Explained variance for Y2 : 0.8784326202684619
```