Regression/Chapter02 - HandsOn Exercise

In [ ]:
```
import pandas as pd
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
# Read X.txt
# It usually works without the "lineterminator='\n'" option
X=pd.read_csv("./HousingData/X.txt",sep='\t', lineterminator='\n', header=None )
X.head()
```

In [ ]:
```
# Read Y.txt
y=pd.read_csv("./HousingData/Y.txt", header=None )
y.head()
```

Least Squares: Linear Regression

Performing Linear Regression using Scikit-Learn is simple:

In [ ]:
```
# The input X should not have the X0=1 column in it, so, we drop it and create and use X2.
X2 = X.drop([0], axis=1)
X2.head()
```

In [ ]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X2, y)
lin_reg.intercept_, lin_reg.coef_ # Note how the intercept and the rest of the coefficient
s are returned.
```

The `LinearRegression` class is based on the `scipy.linalg.lstsq()` function (the name `linalg` stands for "**linear algebra**" and `lstsq()` stands for “**least squares**”), which you could call directly.

Now we can predict the house-price for the Query, Q=[5 3 2500], that is a house having 5 bedrooms, 3 bathrooms and 2500 sqft living area as:

In [ ]:
```
Q = pd.DataFrame([[5,3,2500]])
Q
```

In [ ]:
```
lin_reg.predict(Q)
```

The Normal Equation

To find the value of **Beta** that minimizes the cost or error function, we can use a *closed-form* solution - in other words, a mathematical equation that gives the result directly. This is called the *Normal Equation*, expresses as $Beta = (X^T X)^{-1} X^T y$.

We will use the `inv()` function from NumPy’s linear algebra module (`np.linalg`) to compute the inverse of a matrix, and the `dot()` method for matrix multiplication

In [ ]:
```
Beta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
Beta
```

If you need to use *pseudoinverse*, you can use `np.linalg.pinv()` as following:

In [ ]:
```
np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)
```

Now we use Beta for prediction.

- For a Query, Q=[1 5 3 2500], that is a house having 5 bedrooms, 3 bathrooms and 2500 sqft living area, for example,
  - we can predict the house price by: Q*Beta;
  - which should return 2.1384e+005 or, 213,840

Note: Q is in row form and Beta is in column form. Thus, we do `Q*Beta` instead of `Q.T*Beta` . And the '1' indicates X_0.

In [ ]:
```
Q= np.array([1,5,3, 2500])
Q
```

In [ ]:
```
Q.dot(Beta)
```

Additional Information

Now, assume the matrix X did not have *X0*=1 column inserted and we need to insert it:

In [ ]:
```
X.shape
```

In [ ]:
```
R_cnt=X.shape[0]
R_cnt
```

`np.c_` is array concatenate

In [ ]:
```
X1 = np.c_[np.ones((R_cnt,1)), X]
```

In [ ]:
```
X1
```