Question 2: Weighted Linear Regression

Data

In the current problem we are given m values of attributes such that for each i^{th} value of the attribute ($\alpha^i \epsilon R^1$) we have $Y^i \epsilon R^1$. We define X^i such that for each i^{th} sample $X^i = < 1, \alpha^i >$ to accomodate intercept term where $X \epsilon R^{mx(1+1)}$.

In [1]:

```
import sys
import numpy as np
sys.path.append('../')
from lib.ml import linear
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
#%matplotlib
```

executed in 449ms, finished 18:30:03 2018-02-12

loading training and testing data

In [2]:

```
X = (np.loadtxt(open('weightedX.csv'), delimiter=",")).reshape(-1, 1)
X = np.hstack((np.ones(X.shape),X))
Y = (np.loadtxt(open('weightedY.csv'), delimiter=",")).reshape(-1, 1)
```

executed in 12ms, finished 18:30:03 2018-02-12

Functions to plot model estimates

Plotting linear estimates using normal equations of unwighted and weighted linear model

In [3]:

```
def _plot_model(X,Y,theta,X_os=None,X_range=None,tau=None,flag=True):

    if flag:
        plt.figure(5)
        plt.title("Model Using normal equation",fontsize=10,y=1.08)
        Y_predicted = theta[0,1]*X[:,1]+theta[0,0]
        plt.plot(X[:,1],Y,'b.',label='Actual Data')
        plt.plot(X[:,1],Y_predicted,'r-',label='model')
        plt.legend(bbox_to_anchor=(0.6, 0.2), loc=2, borderaxespad=0.)
        plt.xlabel(r'$X$')
        plt.ylabel(r'$Y$')
        plt.draw()
    else:
        plt.figure(1)
        plt.title(r'Model for $\tau$ = %f'%(tau),fontsize=10,y=1.08)
        X_values = np.linspace(X_range[0],X_range[1],10)
        Y_values = X_values*theta[0,0,1]+theta[0,0,0]
        count = X_range.shape[0]
        for i in range(2,count):
            X_temp = np.linspace(X_range[i-1],X_range[i],10)
            Y_temp = X_temp*theta[i-1,0,1]+theta[i-1,0,0]
            X_values = np.hstack((X_values,X_temp))
            Y_values = np.hstack((Y_values,Y_temp))
        Y_os = X_os[:,1]*theta[:,0,1]+theta[:,0,0]
        plt.plot(X[:,1],Y,'b.',label='Actual Data')
        plt.plot(X_values,Y_values,'r-',label='model')
        plt.plot(X_os[:,1],Y_os,'go',label='Observed Points')
        plt.legend(bbox_to_anchor=(0.6, 0.25), loc=2, borderaxespad=0.)
        plt.xlabel(r'$X$')
        plt.ylabel(r'$Y$')
        plt.draw()
```

executed in 148ms, finished 18:30:03 2018-02-12

Evaluating θ for unweighted linear model

Normal equation for θ in unweighted linear regression:

Error function $J(\theta) = (X\theta - Y)^T(X\theta - Y)$ now $\bigtriangledown_\theta J = 2X^TX\theta - 2X^TY$. For optimum value of $J(\theta)$ put $\bigtriangledown_\theta J = 0$ i.e.

$2X^TX\theta - 2X^TY = 0$

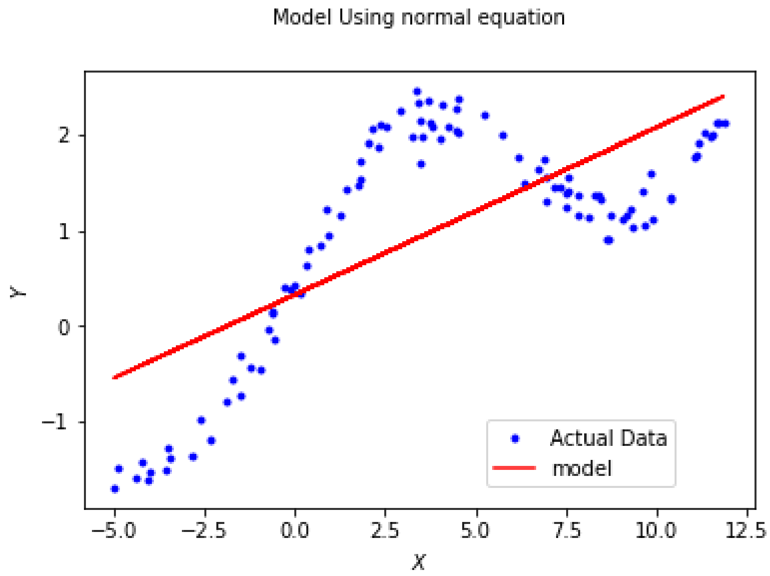
or

$\theta = (X^TX)^{-1}X^TY$

In [4]:

```
shp = X.shape[-1]
XXTinv = np.linalg.pinv(np.dot(X.T,X).reshape(shp,shp))
theta = np.dot(XXTinv,np.dot(X.T,Y)).reshape(1,-1)
_plot_model(X,Y,theta)
```

executed in 916ms, finished 18:30:04 2018-02-12



Evaluating θ for weighted linear model at given τ

Normal equation for θ in weighted linear regression:

Error function $J(\theta) = (X\theta - Y)^T W(X\theta - Y)$ where $W \varepsilon R^{mxm}$ such that

$W_{ij} = \begin{Bmatrix} e^{-S\frac{(X^i-X_o)^T(X^i-X_o)}{\tau}} & \forall i = j \\ 0 & o.w. \end{Bmatrix}$ τ is a scaling factor and X_o is the observed point.

now $\bigtriangledown_\theta J = 2X^T WX\theta - 2X^T WY$. For optimum value of $J(\theta)$ put $\bigtriangledown_\theta J = 0$ i.e.

$2X^T WX\theta - X^T WY = 0$

or

$\theta = (X^T WX)^{-1} X^T WY$

Generating observed points

In [5]:

```
X_b = np.linspace(min(X[:,1]),max(X[:,1]),15)
X_o = ((X_b[1:]+X_b[0:-1])/2).reshape(-1,1)
X_o = np.hstack((np.ones(X_o.shape),X_o))
```

executed in 9ms, finished 18:30:04 2018-02-12

Evaluating θ at all observed points (X_o) and plotting them for the respective range (X_b) at a given τ

In [6]:

```
def wlm(X,Y,X_os,X_b,tau):
    thetas = []
    for X_o in X_os:
        X_o = X_o.reshape(1,-1)
        W = np.identity(X.shape[0])
        W = W*np.nan_to_num(np.exp(-np.dot((X-X_o),(X-X_o).T)/(2*(tau)**2)))
        XTW = np.dot(X.T,W)
        XTWX = np.dot(XTW,X)
        shp = X.shape[-1]
        XTWXinv = np.linalg.pinv(XTWX).reshape(shp,shp)
        thetas.append(np.dot(XTWXinv,np.dot(XTW,Y)).reshape(1,-1))
    thetas = np.asarray(thetas)
    _plot_model(X,Y,thetas,X_os,X_b,tau,False)
```

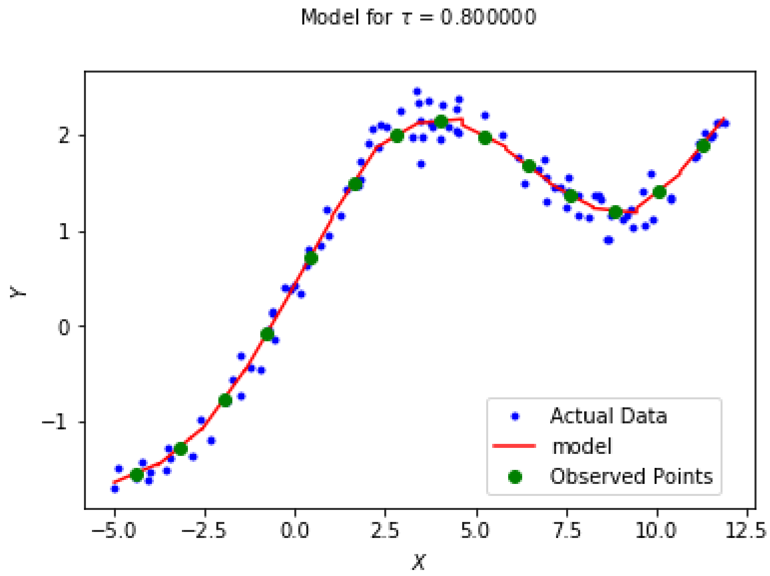
executed in 214ms, finished 18:30:04 2018-02-12

Plotting for $\tau = 0.8$

In [7]:

```
wlm(X,Y,X_o,X_b,tau=0.8)
```

executed in 585ms, finished 18:30:05 2018-02-12

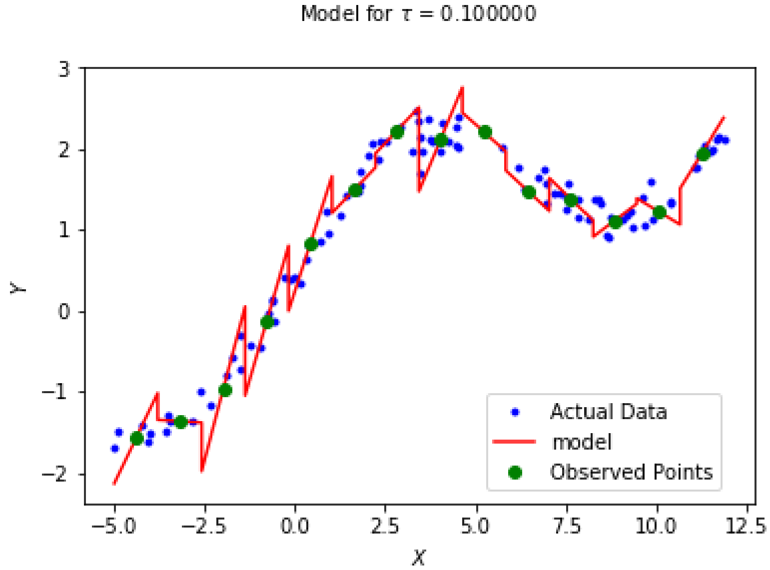


Plotting for $\tau = 0.1$

In [8]:

```
wlm(X,Y,X_o,X_b,tau=0.1)
```

executed in 408ms, finished 18:30:05 2018-02-12

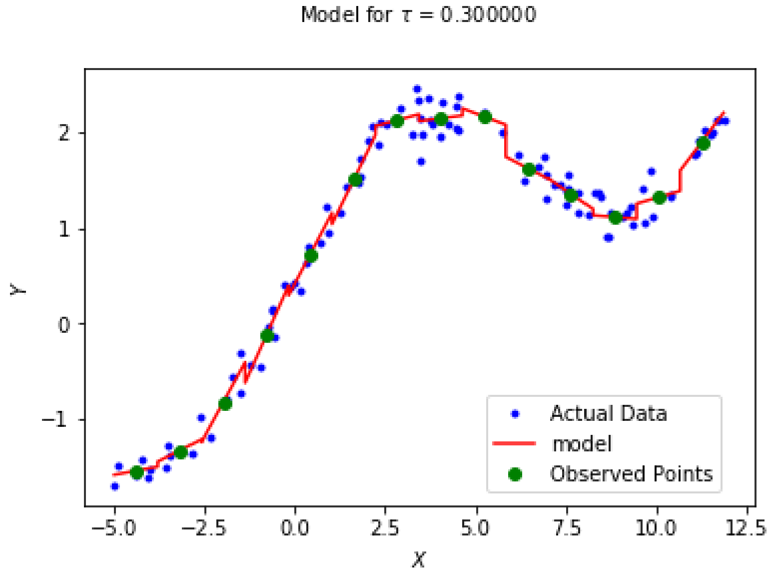


Plotting for $\tau = 0.3$

In [9]:

```
wlm(X,Y,X_o,X_b,tau=0.3)
```

executed in 507ms, finished 18:30:05 2018-02-12

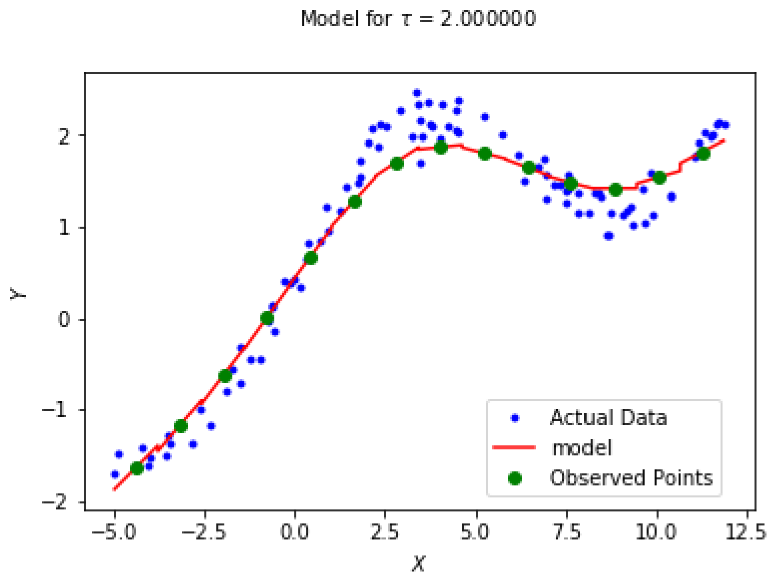


Plotting for $\tau = 2$

In [10]:

```
wlm(X,Y,X_o,X_b,tau=2)
```

executed in 468ms, finished 18:30:06 2018-02-12

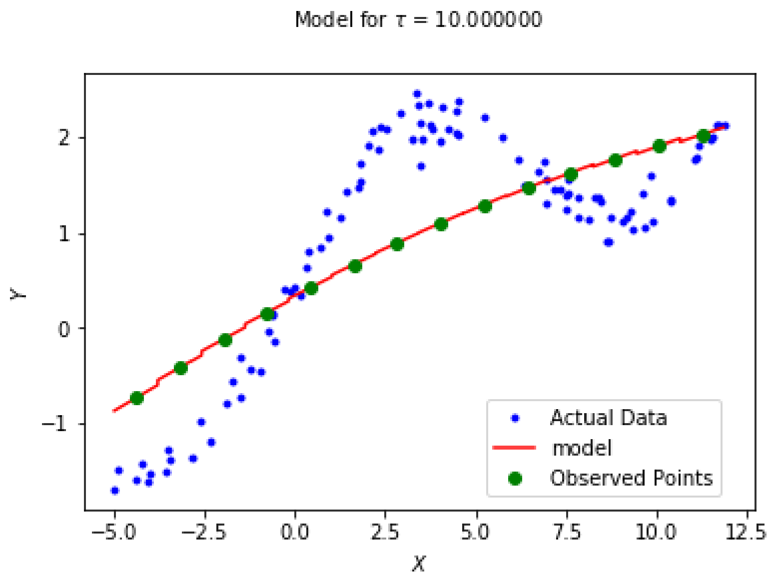


Plotting for $\tau = 10$

In [11]:

```
wlm(X,Y,X_o,X_b,tau=10)
```

executed in 401ms, finished 18:30:06 2018-02-12



Note: when τ is too small model seems to be overfiiting the data which is local to the observed point and when τ is too large model seems to be working like unweighted linear regression