Multilinear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

In [10]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [20]:

```
columns = ['Homework','Midterm','Final']
data = pd.DataFrame({
    "Homework": [95,70,80,100,70],
    "Midterm": [90,60,80,80,85],
    "Final": [93,66,85,60,90]

},index=['Alice','Bob','Clare','David','Eve'])
data.head()
```

Out[20]:

	Homework	Midterm	Final
Alice	95	90	93
Bob	70	60	66
Clare	80	80	85
David	100	80	60
Eve	70	85	90

In [21]:

```
print(np.corrcoef(data['Homework'],data['Final'])[0,1])
```

```
-0.1771490677357476
```

In [22]:

```
print(np.corrcoef(data['Midterm'],data['Final'])[0,1])
```

```
0.6700743886411277
```

In [23]:

```
theta1 = 0.2
theta2 = 0.8
theta0 = 93 - 0.2*95 - 0.8*90
print(theta0)
```

```
2.0
```

In [12]:

```
#MSE
x1 = data.loc['Alice',['Homework','Midterm']].values
print(x1)
```

```
[95 90]
```

In [24]:
```
y1 = data.loc['Alice',['Final']].values
print(y1)
```

```
[93]
```

In [25]:
```
theta = np.array([2.0,0.2,0.8])
print(theta)
```

```
[2.  0.2 0.8]
```

In [26]:
```
prediction = 2.0 + 0.2*x1[0] + 0.8*x1[1]
print(prediction)
```

```
93.0
```

In [27]:
```
squared_error = (prediction - y1)**2
print(squared_error)
```

```
[0.]
```

In [28]:
```
def get_squared_error(data,name,theta):
    x = data.loc[name,['Homework','Midterm']].values
    y = data.loc[name,['Final']].values
    prediction = theta[0] + theta[1]*x[0] + theta[2]*x[1]
    squared_error = (prediction-y)**2
    return squared_error
get_squared_error(data,"Bob",theta)
```

Out[28]:
```
array([4.])
```

In [29]:
```
all_errors = [get_squared_error(data,name,theta)for name in data.index]
print(all_errors)
```

```
[array([0.]), array([4.]), array([9.]), array([676.]), array([36.])]
```

In [30]:
```
mse = np.mean(all_errors)
print('MSE: ',mse)
print("RMSE: ",np.sqrt(mse))
```

```
MSE:  145.0
RMSE:  12.041594578792296
```

Multilinear Regression: Normal Equation

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}.$$

In [34]:
```
m,n = data.shape
X = np.hstack([np.ones([m,1]),data[['Homework','Midterm']].values])
print(X)
```

```
[[  1.  95.  90.]
 [  1.  70.  60.]
 [  1.  80.  80.]
 [  1. 100.  80.]
 [  1.  70.  85.]]
```

In [32]:
```
y = data[['Final']].values
print(y)
```

```
[[93]
 [66]
 [85]
 [60]
 [90]]
```

In [35]:
```
num = X.T.dot(x)
theta_opt = np.linalg.inv(num).dot(X.T.dot(y))
print(theta_opt)
```

```
[[35.        ]
 [-0.71627907]
 [ 1.30697674]]
```

In [38]:
```
all_errors = [get_squared_error(data,name,theta_opt)for name in data.index]
mse = np.mean(all_errors)
rmse = np.sqrt(mse)
print("MSE: ",mse)
print("RMSE: ",rmse)
```

```
MSE:  36.82790697674422
RMSE:  6.068600083770903
```

Multilinear Regression: Gradient Descent

$$\hat{\theta} \leftarrow \hat{\theta} - r \cdot \frac{\partial J(\hat{\theta})}{\partial \theta}.$$

- The partial derivative of the cost function is given by

$$\frac{\partial J(\hat{\theta})}{\partial \theta} = \frac{2}{m} \cdot \mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y}).$$

- **Verify the formula of partial derivative asuuming there is one input feature.**
- End iteration if certain stop criteria is reached, such as:
  - Value of $\hat{\theta}$ becomes stable.
  - Certain iteration amount is reached.

In [42]:
```
m,n=data.shape
theta_hat = np.random.rand(n,1)
print(theta_hat)
```

```
[[0.76272655]
 [0.59038968]
 [0.14812814]]
```

In [47]:
```
X2 = np.hstack([np.ones([m,1]),data[['Homework','Midterm']].values/100])
y2 = y
print(X2)
```

```
[[1.   0.95 0.9 ]
 [1.   0.7  0.6 ]
 [1.   0.8  0.8 ]
 [1.   1.   0.8 ]
 [1.   0.7  0.85]]
```

In [48]:
```
num_iter = 6000
r = 0.05
MSEs = []
for iter in range(num_iter):
    gradient = (X2.T).dot(X2.dot(theta_hat)-y2)*2/m
    theta_hat -= r*gradient
    MSE = 1/m*(X2.dot(theta_hat)-y2).T.dot(X2.dot(theta_hat)-y2)
    MSEs.append(MSE[0,0])

print(theta_hat)
```

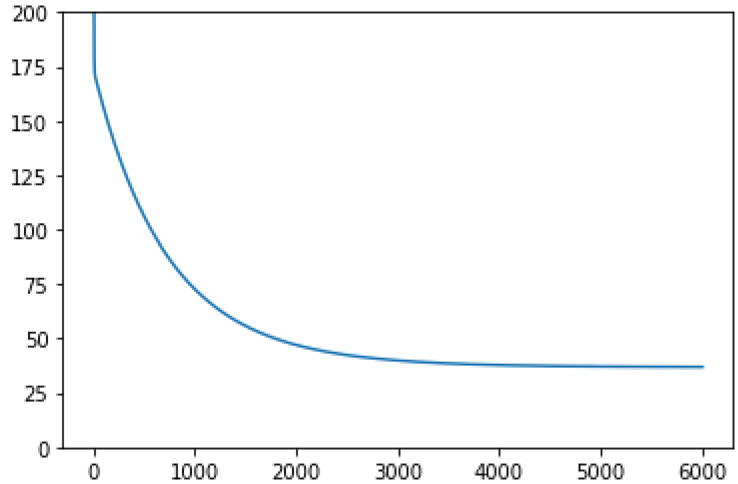
```
[[ 36.31043372]
 [-69.94711093]
 [127.28166283]]
```

In [49]:
```
MSE
```

Out[49]:
```
array([[36.92103962]])
```

In [50]:
```
plt.plot(range(num_iter),MSEs)
plt.ylim(0,200)
```

Out[50]: (0, 200)



In [51]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()
model_lr.fit(data[['Homework','Midterm']],data['Final'])
```

Out[51]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fa
lse)
```

In [52]:
```
model_lr.intercept_,model_lr.coef_
```

Out[52]:
```
(35.00000000000004, array([-0.71627907,  1.30697674]))
```