Machine Learning Introduction

Exercise 1

Lior Yaacov

April, 2021

Importing Relevant Packages

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt

%matplotlib inline
```

Building the Model

Producing X

In [2]:
```
m,n = 50,30
```

In [3]:
```
X = np.random.randint(1,100,(m,n))
X.shape
```

Out[3]:
```
(50, 30)
```

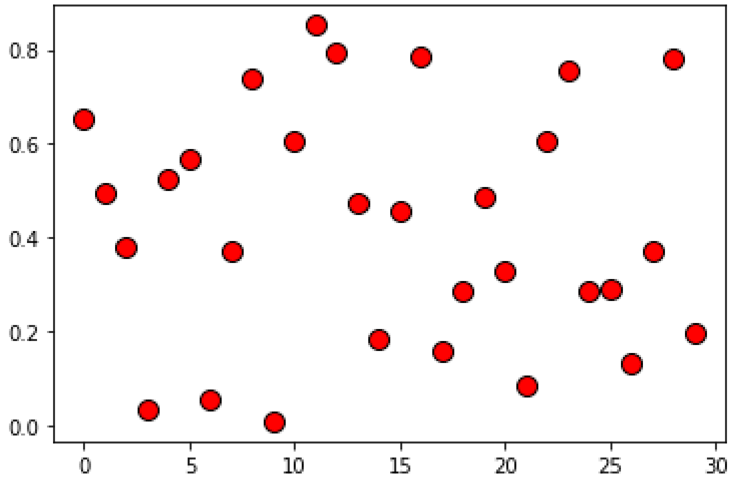
Determining $\beta$

In [4]:
```
initial_beta = np.random.rand(n)
initial_beta.shape
```

Out[4]:
```
(30,)
```

In [5]:
```
plt.plot(initial_beta, 'ro', ms=10, mec='k')
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x1f04b5182b0>]
```



Producing the Noise vector $\epsilon$

In [6]:
```
eps = np.random.randn(m)
```

```
In [7]:  eps.shape

Out[7]:  (50,)
```

Computing $y=X\beta+\epsilon$

```
In [8]:  Y = X.dot(initial_beta)+eps
```

```
In [9]:  Y.shape

Out[9]:  (50,)
```

Loss Function

```
In [10]: def loss(X,Y,beta):
             t = X.dot(beta)-Y
             J = t.T.dot(t)

             return J
```

Normal Equation

```
In [11]: def normal(X,Y):
             return np.dot(np.dot( np.linalg.inv(np.dot(X.T,X)) , X.T),Y)
```

Model Evaluation

Using Initial $\beta$ Parameters

```
In [12]: Y = X.dot(initial_beta)+eps
         print(loss(X,Y,initial_beta))

         55.31368554209154
```

```
In [13]: num=20
         sigma = np.arange(1,num+1,1)
         J = np.zeros(num)
```

$\sigma = 1$

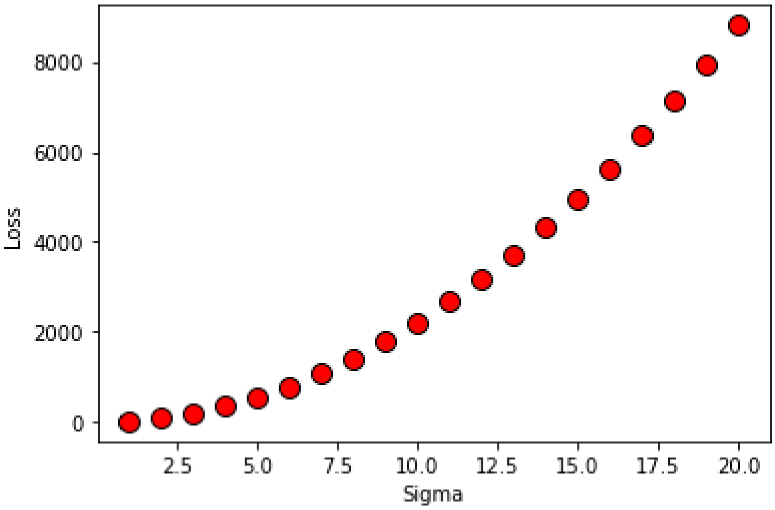
```
In [14]: J[0] = loss(X,Y,normal(X,Y))
```

$1 < \sigma < num$

```
In [15]: for i in range(0,num):
             new_eps = eps*sigma[i]
             Y = X.dot(initial_beta)+new_eps
             J[i] = loss(X,Y,normal(X,Y))
```

```
In [16]: plt.plot(sigma,J, 'ro', ms=10, mec='k')
         plt.xlabel('Sigma')
         plt.ylabel('Loss')

Out[16]: Text(0, 0.5, 'Loss')
```


8000
6000
4000
2000
0
Loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Sigma