Information

Writer : *Junhyuck Woo*
Std.ID : 20145337
*Assignment*03 : *Visualization of Gradient Descent algorithm based on Linear Regression problem*
Deadline : *Apr* 11, 2020

Library

In [0]:
```
import matplotlib.pyplot as plt; import numpy as np
```

Data

In [0]:
```
path="/content/drive/My Drive/Spring|2020/Machine_Learning/CAU-MachineLearning/Assignment03/data.csv"
data = np.genfromtxt(path, delimiter=',')
x_data = data[:, 0]
y_data = data[:, 1]

m = x_data.size
```

Linear Model

$h_\theta(x) = \theta_0 + \theta_1 x$
$\theta_0 = -30$
$\theta_1 = -30$

In [0]:
```
theta0 = -30; theta0_old = 0
theta1 = -30; theta1_old = 0
theta0_history = [theta0]
theta1_history = [theta1]
h = theta0 + theta1*x_data
```

Objective Function

$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

In [0]:
```
j = np.sum((h - y_data)**2) / (2*m)
j_old = 0
j_history = [j]
```

Training Process - Gradient Descent

$\theta_0^{(t+1)} := \theta_0^{(t)} - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$
$\theta_1^{(t+1)} := \theta_1^{(t)} - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$
$\alpha = 0.004$

In [0]:
```
alpha = 0.004

# Check the number of iteration
iteration = 1

while  (abs(j - j_old) + abs(theta0 - theta0_old) + abs(theta1 - theta1_old)) != 0:
    # Calculate the theta
    theta0_old = theta0
    theta1_old = theta1
    theta0 = theta0 - alpha*np.sum(h-y_data)/m
    theta1 = theta1 - alpha*np.sum((h-y_data)*x_data)/m

    # Update the j, h
    j_old = j
    h = theta0 + theta1*x_data
    j = round(np.sum((h - y_data)**2) / (2*m), 1)

    # Record the history of parameter
    theta0_history.append(theta0)
    theta1_history.append(theta1)
    j_history.append(j)

    iteration = iteration +1
```

Check the Convergence

In [8]:
```
# Iteration
print("# Iteration: " + str(iteration) + '\n')

# Theta 0
print("Updated Theta0: "+ str(theta0))
print("Old Theta0: " + str(theta0_old))
print("Diff: " + str(theta0 - theta0_old) + '\n')

# Theta 1
print("Updated Theta1: "+ str(theta1))
print("Old Theta1: " + str(theta1_old))
print("Diff: " + str(theta1 - theta1_old) + '\n')

# J, Energy Value
print("Updated J: "+ str(j))
print("Old J: " + str(j_old))
print("Diff: " + str(j - j_old) + '\n')
```

```
# Iteration: 8132

Updated Theta0: 24.907393293947564
Old Theta0: 24.907393293947564
Diff: 0.0

Updated Theta1: 9.934635539221347
Old Theta1: 9.934635539221347
Diff: 0.0

Updated J: 27.5
Old J: 27.5
Diff: 0.0
```

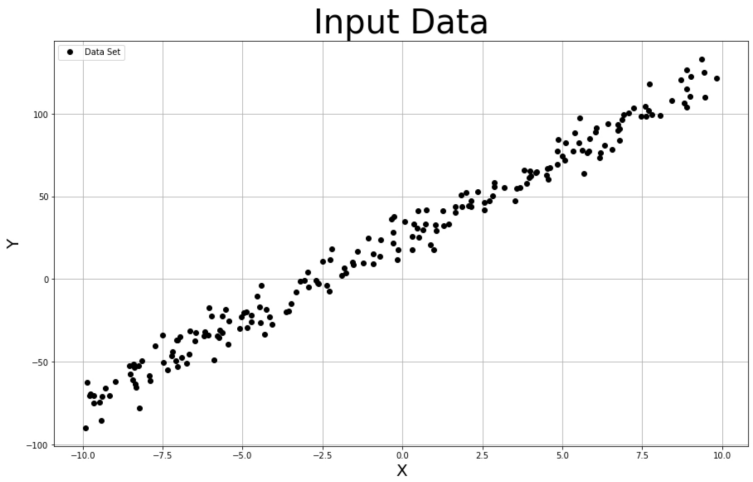
Generate Energy Surface

In [0]:
```
x = np.linspace(-300,300,601)
y = np.linspace(-300,300,601)

t0_grid,t1_grid = np.meshgrid(x, y)
energy_grid = np.zeros((601, 601))

for row in x:
    for col in y:
            energy_grid[int(row)+300][int(col)+300] = np.sum((row/10 + col*x_data/10 - y_data)**2)
 / (2*m)
```

1. Input Points

In [10]:
```
plt.figure(figsize=(15,9))
plt.plot(x_data, y_data, 'ro', color='black', label='Data Set')
plt.title('Input Data', fontsize=40)
plt.grid()
plt.legend()
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
```

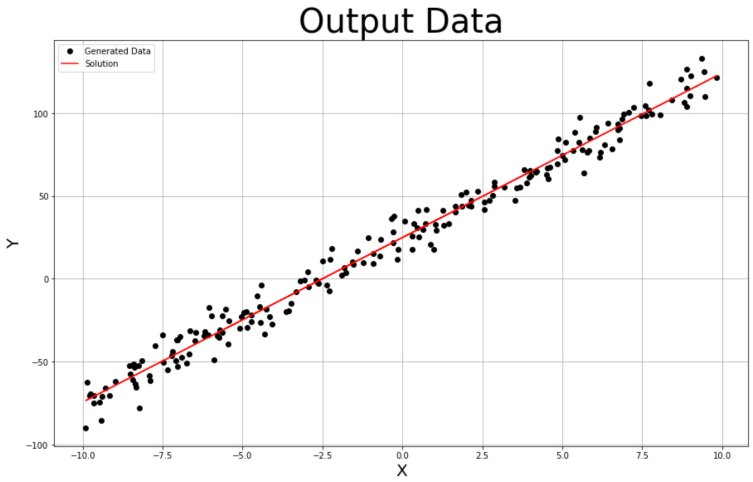
Out[10]: Text(0, 0.5, 'Y')



2. Output Result

In [11]:
```
plt.figure(figsize=(15,9))
plt.plot(x_data, y_data, 'ro', color='black', label='Generated Data')
plt.plot(x_data, h, color='red', label='Solution')
plt.title('Output Data', fontsize=40)
plt.grid()
plt.legend()
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
```

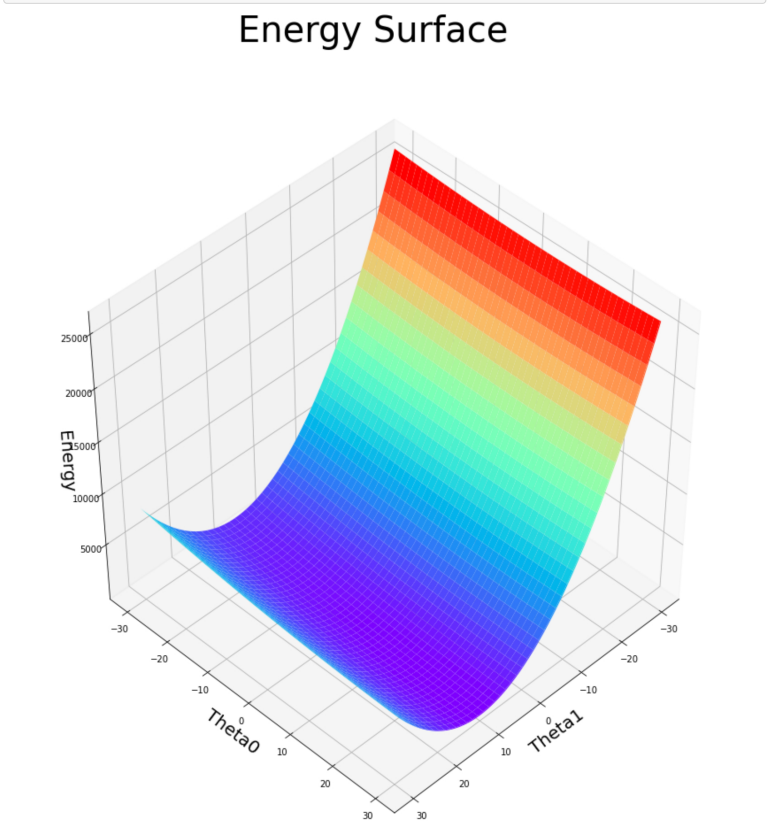
Out[11]: Text(0, 0.5, 'Y')



3. Energy Surface

In [12]:
```
fig=plt.figure(figsize=(15,15))
fig.suptitle('Energy Surface', fontsize=40)
ax = fig.add_subplot(projection='3d')
ax.plot_surface(t0_grid/10, t1_grid/10, energy_grid, cmap='rainbow')
ax.set_xlabel('Theta1', fontsize=20)
```

```
ax.set_ylabel('Theta0', fontsize=20)
ax.set_zlabel('Energy', fontsize=20)
ax.view_init(45, 45)
plt.show()
```



4.Gradient Descent Path on the Energy Surface

In [13]:
```
fig=plt.figure(figsize=(15,15))
fig.suptitle('Energy Surface', fontsize=40)
ax = fig.add_subplot(projection='3d')
ax.plot(theta1_history, theta0_history, j_history, color='black', marker="x", zorder=3, label='Ene
rgy', markersize=10)
ax.plot([-30], [-30], [j_history[0]], color='blue', marker="o", zorder=6, label='Initial Point', m
arkersize=10)
ax.plot_surface(t0_grid/10, t1_grid/10, energy_grid, cmap='rainbow')
ax.set_xlabel('Theta1', fontsize=20)
ax.set_ylabel('Theta0', fontsize=20)
ax.set_zlabel('Energy', fontsize=20)
ax.view_init(45, 45)
plt.legend()
plt.show()
```