Search or jump to... / Pull requests Issues Marketplace Explore

Junhyuck-Woo / CAU-MachineLearning Private

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 5 Wiki Security Insights Settings

Branch: master CAU-MachineLearning / Assignment05 / Assignment05.ipynb

Find file Copy path

Junhyuck-Woo Plotting the Training error and classifier 72afb13 4 minutes ago

1 contributor

1 lines (1 sloc) 229 KB

Raw Blame History

Information

Writer : *JunhyuckWoo*
Std.ID : 20145337
*Assignment*05 : *Logisticregression for a binary classification*
Deadline : *Apr* 23, 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/Assignment05/data.txt"
data    = np.genfromtxt(path, delimiter=',')

x       = data[:, 0]
y       = data[:, 1]
label   = data[:, 2]

x_label0    = x[label == 0]
x_label1    = x[label == 1]

y_label0    = y[label == 0]
y_label1    = y[label == 1]

m = label.size
```

Logistic regression

$\hat{h} = \sigma(z)$
$z = \theta_0 + \theta_1 x + \theta_2 y$, where $\theta_0, \theta_1, \theta_2 \in \mathbb{R}$
$\sigma(z) = \frac{1}{1+\exp(-z)}$
$\sigma'(z) = \sigma(z)(1-\sigma(z))$

InitialState
$\theta_0 = 0$
$\theta_1 = 0.1$
$\theta_2 = -0.1$

In [0]:
```
theta0 = 0; theta0_old = 0
theta1 = 0.1; theta1_old = 0
theta2 = -0.1; theta2_old = 0
theta0_history = [theta0]
theta1_history = [theta1]
theta2_history = [theta2]

z = theta0 + theta1*x + theta2*y
sigma = 1/(1 + np.exp(-z))
sigma_ = sigma * (1-sigma)
```

Objective Function

$J(\theta_0, \theta_1, \theta_2) = \frac{1}{m}\sum_{i=1}^{m}(-l^{(i)}\log(\sigma(z^{(i)})) - (1-l^{(i)})\log(1-\sigma(z^{(i)})))$

In [0]:
```
j = np.sum(-label*np.log(sigma) - (1-label)*np.log(1-sigma)) / m
j_old = 0
j_history = [j]
```

Training Process - Gradient Descent

$\alpha = 0.0002$
$\theta_0^{(t+1)} = \theta_0^{(t)} - \alpha\frac{1}{m}\sum_{i=1}^{m}(\sigma(z^{(i)}) - l^{(i)})$
$\theta_1^{(t+1)} = \theta_1^{(t)} - \alpha\frac{1}{m}\sum_{i=1}^{m}(\sigma(z^{(i)}) - l^{(i)})x^{(i)}$
$\theta_2^{(t+1)} = \theta_2^{(t)} - \alpha\frac{1}{m}\sum_{i=1}^{m}(\sigma(z^{(i)}) - l^{(i)})y^{(i)}$

In [0]:
```
alpha = 0.0002

# Check the number of iteration
iteration = 1

# Set the two condition because it spent too much time for converge
while  (iteration < 1048576):
    # Calculate the theta
    theta0_old = theta0
    theta1_old = theta1
    theta2_old = theta2
    theta0 = theta0 - alpha*np.sum(sigma-label)/m
    theta1 = theta1 - alpha*np.sum((sigma-label)*x)/m
    theta2 = theta2 - alpha*np.sum((sigma-label)*y)/m

    # Update the j
    j_old = j
    z = theta0 + theta1*x + theta2*y
    sigma = 1/(1 + np.exp(-z))
    j = np.sum(-label*np.log(sigma) - (1-label)*np.log(1-sigma)) / m

    # Record the history of parameter
```

```
    theta0_history.append(theta0)
    theta1_history.append(theta1)
    theta2_history.append(theta2)
    j_history.append(j)

    iteration = iteration +1
```

Check the Convergence

In [7]:
```
# 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')

# Theta 2
print("Updated Theta1: "+ str(theta2))
print("Old Theta1: " + str(theta2_old))
print("Diff: " + str(theta2 - theta2_old) + '\n')

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

```
# Iteration: 1048576

Updated Theta0: -7.652334539261999
Old Theta0: -7.65233044856147
Diff: -4.090700529424396e-06

Updated Theta1: 0.06707489620293466
Old Theta1: 0.06707486438990407
Diff: 3.181303059873741e-08

Updated Theta1: 0.06058509437964219
Old Theta1: 0.060585062033159415
Diff: 3.234648277700147e-08

Updated J: 0.3123459385974633
Old J: 0.31234602227687963
Diff: -8.36794163405763e-08
```

Generate the grid

x = [30 : 0.5 : 100]
y = [30 : 0.5 : 100]

In [0]:
```
x_range = np.arange(30, 100.5, 0.5)
y_range = np.arange(30, 100.5, 0.5)
x_grid, y_grid = np.meshgrid(x_range, y_range)
classifier_colormap = 1/(1 + np.exp(-theta0 - theta1*x_grid - theta2*y_grid))
```

1. Training Data

In [12]:
```
plt.figure(figsize=(12,12))
plt.scatter(x_label0, y_label0, c='b', label='Label 0')
plt.scatter(x_label1, y_label1, c='r', label='Label 1')
plt.grid()
plt.legend()
plt.title('Training Data', fontsize=40)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.show()
```

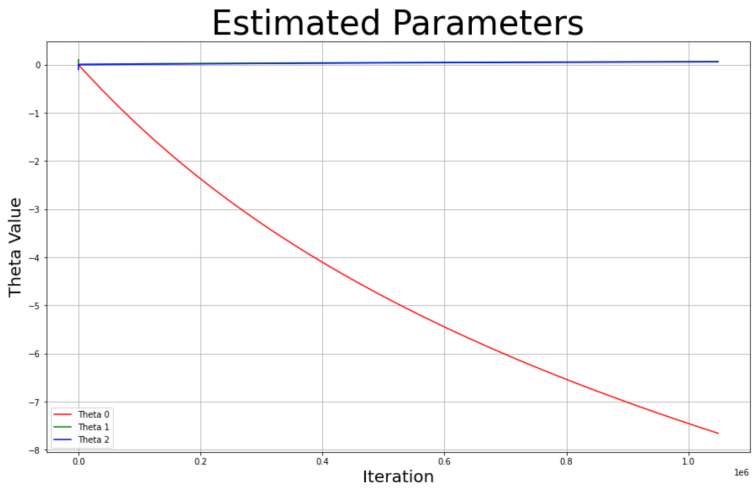
2. Estimated Parameters

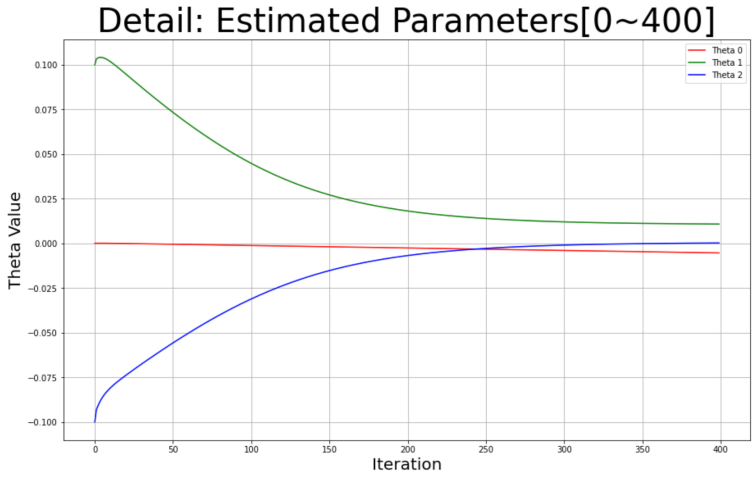
In [13]:
```
plt.figure(figsize=(15,9))
plt.plot(theta0_history, color='red', label='Theta 0')
plt.plot(theta1_history, color='green', label='Theta 1')
plt.plot(theta2_history, color='blue', label='Theta 2')
```

```
plt.grid()
plt.legend()
plt.title('Estimated Parameters', fontsize=40)
plt.xlabel('Iteration', fontsize=20)
plt.ylabel('Theta Value', fontsize=20)


plt.figure(figsize=(15,9))
plt.plot(theta0_history[0:400], color='red', label='Theta 0')
plt.plot(theta1_history[0:400], color='green', label='Theta 1')
plt.plot(theta2_history[0:400], color='blue', label='Theta 2')
plt.grid()
plt.legend()
plt.title('Detail: Estimated Parameters[0~400]', fontsize=40)
plt.xlabel('Iteration', fontsize=20)
plt.ylabel('Theta Value', fontsize=20)
```

Out[13]: Text(0, 0.5, 'Theta Value')

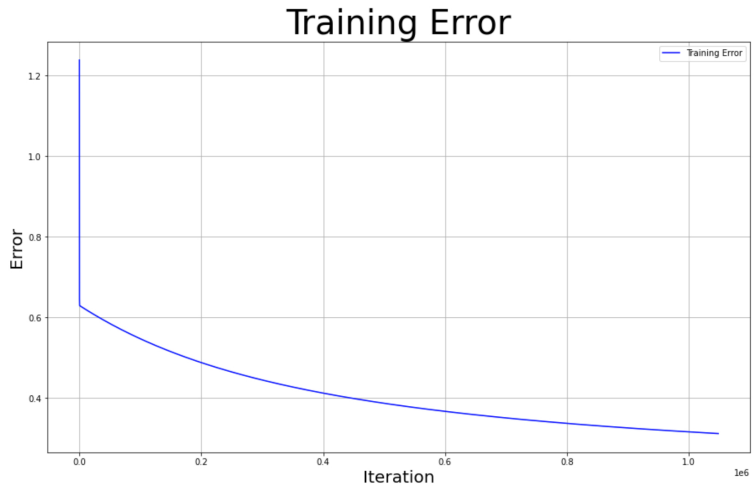




3. Training Error

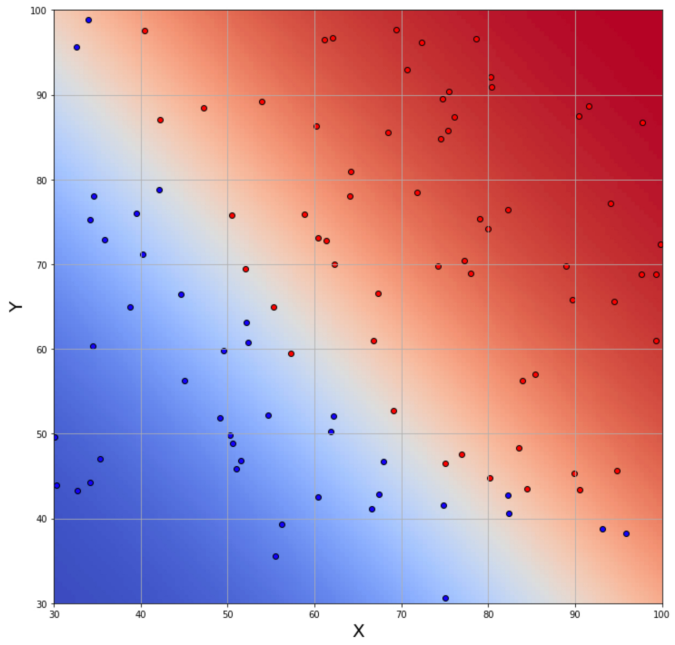
In [15]:
```
plt.figure(figsize=(15,9))
plt.plot(j_history, color='blue', label='Training Error')
plt.grid()
plt.legend()
plt.title('Training Error', fontsize=40)
plt.xlabel('Iteration', fontsize=20)
plt.ylabel('Error', fontsize=20)
```

Out[15]: Text(0, 0.5, 'Error')



4. Obtained Classifier

In [16]:
```
plt.figure(figsize=(12,12))
plt.pcolor(x_range, y_range, classifier_colormap, cmap='coolwarm')
plt.scatter(x_label0, y_label0, c='b', label='Label 0', edgecolors='black')
plt.scatter(x_label1, y_label1, c='r', label='Label 1', edgecolors='black')
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.grid()
plt.show()
```



© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About