SOFT COMPUTING : ISC0630E

ASSIGNMENT 3 : IIT2016067(Maanas Vohra)

Question Description

Using the data set of two examination results design a predictor using logistic regression for predicting whether a student can get an admission in the institution. Use regularizer to further tune the parameters. Use 70 % data for training and rest 30% data for testing your predictor and calculate the efficiency of the predictor/hypothesis.

Hints: 1. You can pre process the data for convenience

2. You must use Python program for evaluating parameters using batch gradient descent algorithm (GDA). No function should be used for GDA.

Using the data set of two quality test results of a microchip product, design a predictor using logistic regression which will predict the acceptance or rejection of the microchip given the two test results. Use regularizer to further tune the parameters. Use 70 % data for training and rest 30% data for testing your predictor and calculate the efficiency of the predictor/hypothesis.

Hints: 1. You can pre process the data for convenience

2. You must use Python program for evaluating parameters using batch gradient descent algorithm (GDA). No function should be used for GDA.

Introduction

This is used for the classification problem. This is just like the regression problem, except that the values y we now want to predict take on only a small number of discrete values. Given $x^{(i)}$ as the training sample features, the corresponding $y^{(i)}$ is also called the label for the training example.

The i^{th} training sample is represented as ($x^{(i)}$, $y^{(i)}$).

We are given a dataset comprising of marks obtained by various students in two subjects and a binary result of 0 if the student was not admitted based on his marks and 1 if the student was admitted.

We are given a dataset comprising of chips and a binary result of 0 if the chip was not accepted based on his marks and 1 if the chip was accepted.

Hypothesis Function

$h\theta(x) = g(X\Theta)$ $X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}. \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

The hypothesis considers both Θ and x as (n + 1) size vectors.

(1) X = m X (n + 1) matrix, containing the training samples features in the rows, where each $x^{(i)}$ is a (n + 1) X 1 matrix.

(2) m = number of training samples

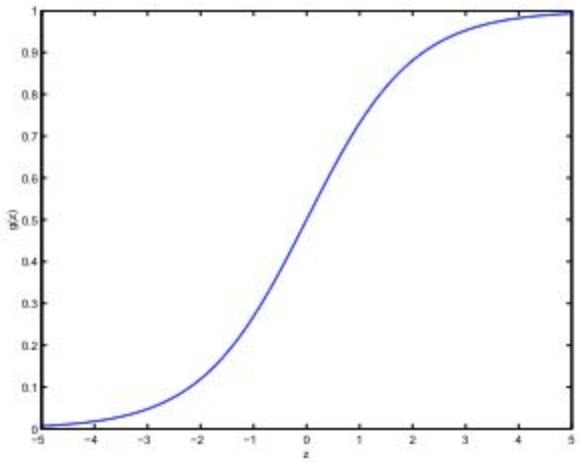
(3) Θ_i = the parameters

(4) x_0 = 1 for every sample.

(5) n = features count, not including x_0 for every training sample

(6) Y contains all the m target values from the training samples, so it's m X 1 matrix

The hypothesis function here is known as the logistic or the sigmoid function. The plot of the logistic function v/s the input is shown below:



$$h(x) = \begin{cases} > 0.5, & \text{if } \theta^T x > 0 \\ < 0.5, & \text{if } \theta^T x < 0 \end{cases}$$

If the weighted sum of inputs is greater than zero, the predicted class is 1 and vice-versa. So the decision boundary separating both the classes can be found by setting the weighted sum of inputs to 0.

The derivative of the sigmoid function is :

$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$

Cost Function

The cost function for a single training sample can be found as

$$
cost = \begin{cases} -log(h(x), & \text{if } y = 1 \\ -log(1-h(x)), & \text{if } y = 0 \end{cases}
$$

Using this, we get

$$
cost(h(x), y) = -ylog(h(x)) - (1-y)log(1-h(x))
$$

Using regularization and for m samples, each having (n + 1) features we get

$$
J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.
$$

This represents the required cost function.

Gradient Descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \qquad \text{for } j \geq 1$$

In the vectorized form it becomes,

$$\nabla J(\theta) = \tfrac{1}{m} \cdot X^T \cdot (g(X \cdot \theta) - \vec{y}) \; + \lambda/m * \theta \; (2{:}n)$$

Thus the gradient descent changes to :

$$\theta := \theta - \alpha \nabla J(\theta)$$

Where α is the learning rate.

The convergence arises when $J_t(\Theta) - J_{t+1}(\Theta) <=$ ERROR_THRESHOLD

RESULTS:

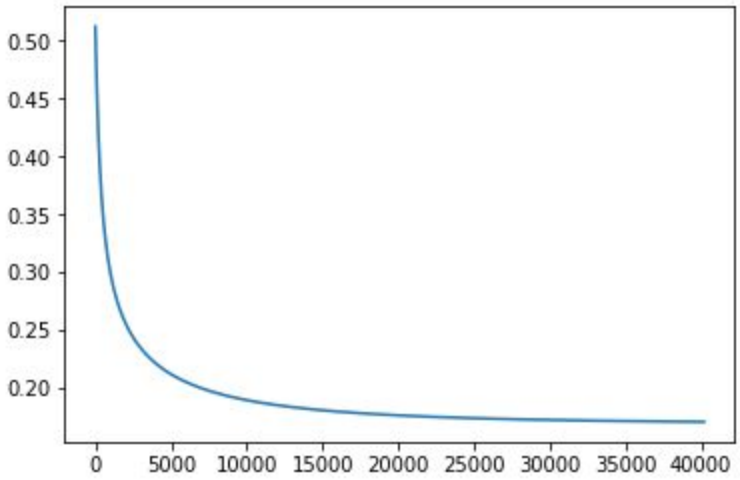
λ = 0.03, α = 0.01, ERROR_THRESHOLD = 0.0000001

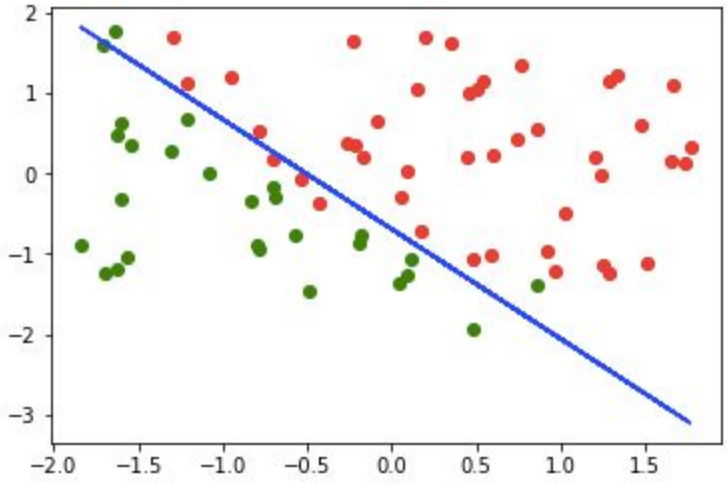
For dataset 1(exam):

Iterations : 40189

Final training error : 0.17080066004005828

Final theta values : [2.65149171 5.212354 3.8195268]





% accuracy of training data : 92.85714285714286

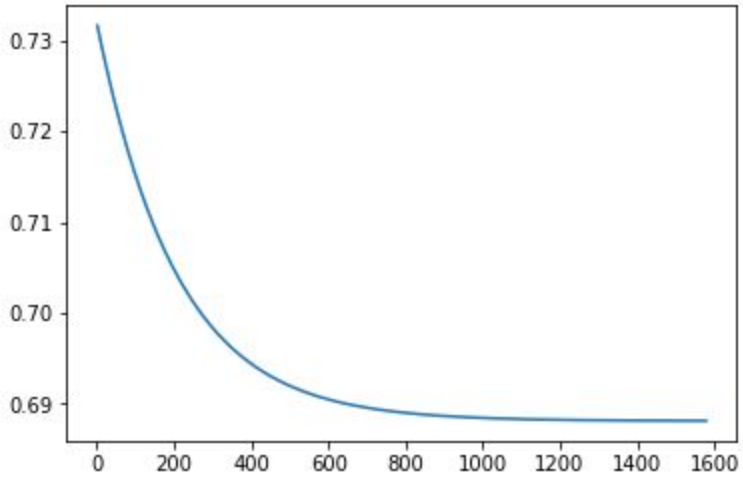
% accuracy of testing data : 86.66666666666667

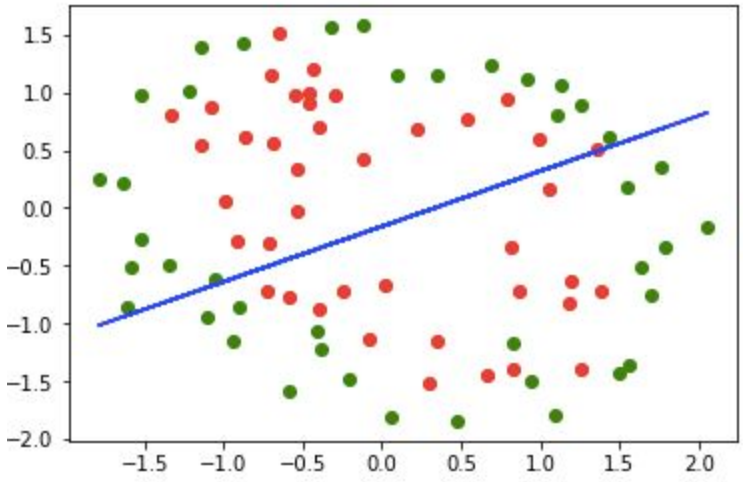
For dataset 2(chip):

(1) By taking only features with sum of power less than or equal to 1, i.e. $x_1^i x_2^j$ is a feature, then i + j <= 1, 0 <= i <= 1, 0 <= j <= 1 with both i, j being integers

Final error: 0.6881037376585071

Theta values: [0.0278978 -0.08352796 0.17462702]





% training accuracy : 53.65853658536586

% test accuracy : 47.22222222222222

The line doesn't divide the data properly, so we take the power limit upto 6

(2) By taking only features with sum of power less than or equal to 6, i.e. $x_1^i x_2^j$ is a feature, then i + j <= 6, 0 <= i <= 6, 0 <= j <= 6 with both i, j being integers

Final error : 0.28724040200133166

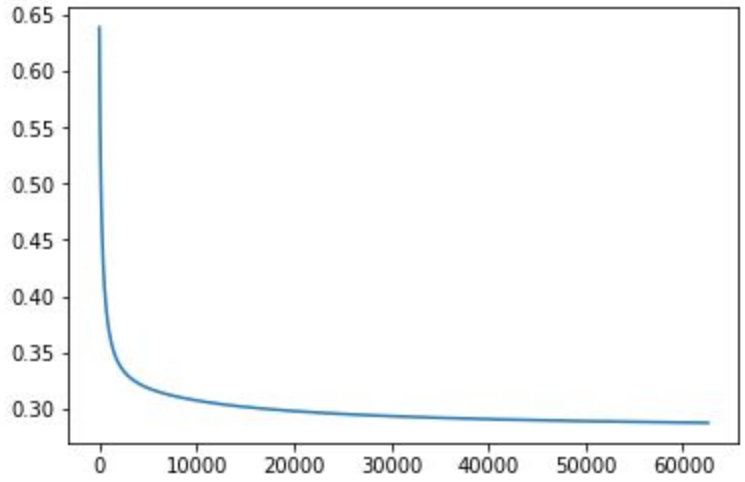
Final theta : [-0.9729242 1.66465509 0.43327726 -2.325464 -1.7867037 -1.35332996

0.96317464 0.95936676 0.48698166 0.56147169 -0.89611716 0.36958413

0.41194726 -0.76894898 -1.46232212 0.00285633 -1.30093424 -0.38460963

1.32374365 0.57522524 0.37050371 -0.88801954 -1.48614612 -1.48151435

1.01368465 -0.5456862 -0.85696933 -2.26971747]



% training accuracy: 87.8048780487805

% test accuracy: 77.77777777777779