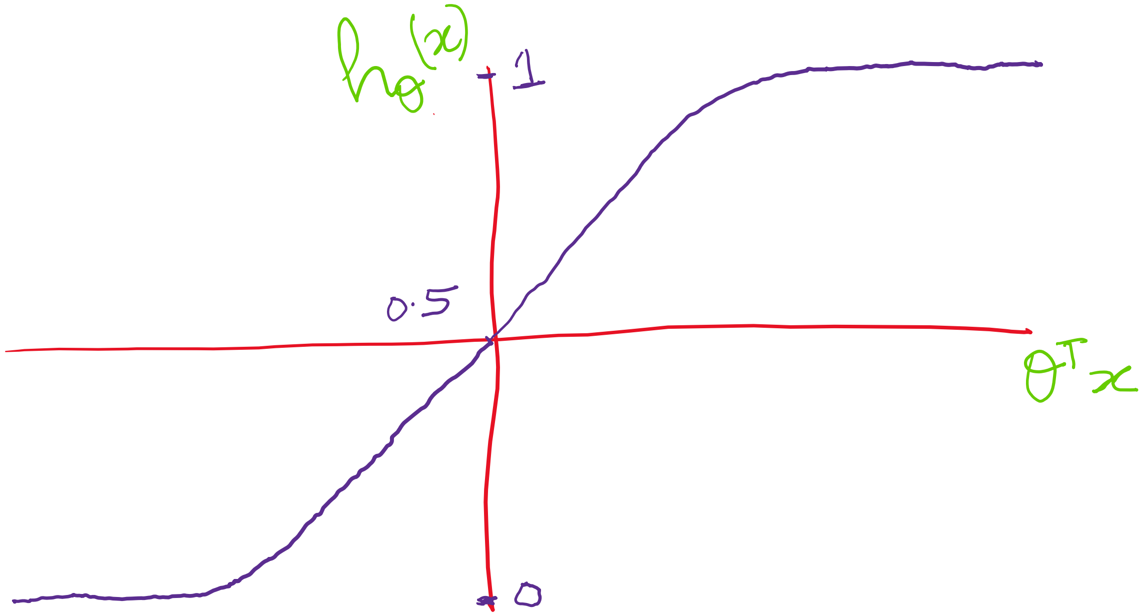
DSL 810 (Data Driven Design)

Instructor: Dr Jay Dhariwal, Dept of Design, IIT Delhi

This tutorial is to work out a classification example by hand. We would be working on a popular classification algorithm, logistic regression.

Let's start with some slides from Andrew Ng.



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

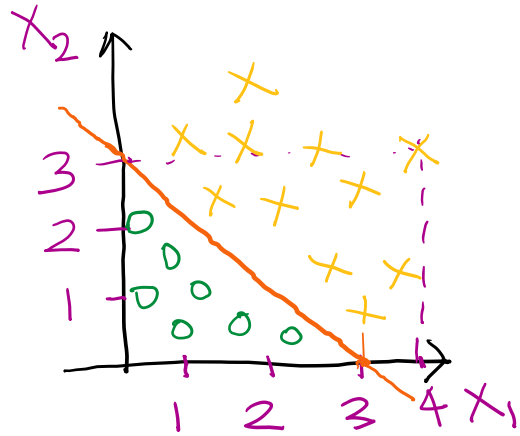
$h_\theta(x)$ = probability that $y=1$, given x, parametrized by θ.

$$\theta^T x = \underbrace{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots}_{\text{familiar from multiple linear regression}}$$

or $$\boxed{h_\theta(x) = p = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\dots)}}}$$

Decision Boundary

Example 1 :



In this case, $h_\theta(x) = \dfrac{1}{1+e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$ Eqn. (1)

$y=1$ is predicted if $-3 + x_1 + x_2 \geq 0$

Let's say $x_1 = 4$, $x_2 = 3$,

$\therefore h_\theta(x) = \dfrac{1}{1+e^{-(-3+4+3)}} = \dfrac{1}{1+e^{-4}}$

or $h_\theta(x) = p = 0.982$.

Similarly, if $x_1 = 1$, $x_2 = 1$,

$h_\theta(x) = \dfrac{1}{1+e^{-(-3+1+1)}} = \dfrac{1}{1+e^{1}}$

$h_\theta(x) = p = 0.27$ so $y=0$ is predicted.

Given a dataset of predictors and response for binary classification, how do we choose the parameters (βs)?

From eqn. 1,

$$h_\theta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

if we consider 2 predictors x_1 & x_2.

$h_\theta(x)$ = probability that x_1 & x_2 belong to class 0 or 1.

$$p(y=1|x:\theta) = h_\theta(x)$$

$$p(y=0|x:\theta) = 1-h_\theta(x)$$

or $$p(y|x:\theta) = [h_\theta(x)]^y\,[1-h_\theta(x)]^{(1-y)}$$

Suppose we have m independent training samples

Method of maximum likelihood can be used to estimate parameters (βs).

$$L(y_1, y_2, \ldots y_m, \beta) = \prod_{i=1}^{m} p(y_i|x_i:\theta)$$

$$L = \prod_{i=1}^{m} [h_\theta(x_i)]^{y_i}\,[1-h_\theta(x_i)]^{(1-y_i)}$$

Taking log on both sides,

$$\ln L(y_1, y_2 \ldots \beta) = \sum_{i=1}^{m}\left[y_i \ln[h_\theta(x_i)] + (1-y_i)\ln[1-h_\theta(x_i)]\right]$$

For optimizing βs, we use the method of Gradient Descent to minimize the cost function $\ln L(y, \beta)$.

$$\min \underbrace{\ln L(y, \beta)}_{} \doteq J(\beta)$$

Repeat {

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j}\left[\ln L(y, \beta)\right]$$

}

simultaneously update all β_j's.

$$\frac{\partial J(\beta)}{\partial \beta_j} = \sum_{i=1}^{m}\left[\frac{y_i}{h_\theta(x_i)}\frac{\partial(h_\theta(x_i))}{\partial \beta_j} - \frac{(1-y_i)}{(1-h_\theta(x_i))}\frac{\partial(1-h_\theta(x_i))}{\partial \beta_j}\right]$$

$$= \sum_{i=1}^{m}\left[\left(\frac{y_i}{h_\theta(x_i)} - \frac{(1-y_i)}{1-h_\theta(x_i)}\right)\frac{\partial h_\theta(x_i)}{\partial \beta_j}\right] \quad (2)$$

$h_\theta(x_i)$ is sigmoid function.

$$h_\theta(x_i) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

$$\frac{\partial h_\theta(x_i)}{\partial \beta_j} = \underbrace{h_\theta(x_i)\left(1-h_\theta(x_i)\right)}, \left[\frac{\partial(\beta_0+\beta_1x_1+\beta_2x_2)}{\partial \beta_j}\right] \quad 3$$

Since for a sigmoid function $f(x)$, $f'(x) = f(x)[1-f(x)]$

From eqn. 2 & eqn. 3,

$$\frac{\partial J}{\partial \beta_j} = (-1)\sum_{i=1}^{m}\left[y_i(1-h_\theta(x_i)) - (1-y_i)h_\theta(x_i)\right]\frac{\partial(\beta_0+\beta_1X_1+\beta_2X_2)}{\partial \beta_j}$$

$$= (-1)\sum_{i=1}^{m}\left[y_i - \cancel{y_i h_\theta(x_i)} - h_\theta(x_i) + \cancel{y_i h_\theta(x_i)}\right]\frac{\partial(\beta_0+\beta_1X_1+\beta_2X_2)}{\partial \beta_j}$$

Now
$$\left.\begin{aligned}\frac{\partial J}{\partial \beta_0} &= (-1)(y_i - h_\theta(x_i))(1)\\ \frac{\partial J}{\partial \beta_1} &= (-1)(y_i - h_\theta(x_i))(x_1)\\ \frac{\partial J}{\partial \beta_2} &= (-1)(y_i - h_\theta(x_i))x_2\end{aligned}\right\} \text{eqn. } 4$$

Using $\frac{\partial J}{\partial \beta_j}$ in gradient descent,

eqn: 5

$$\beta_0 = (\beta_0)_{old} + \alpha \left[h_\theta(x_i) - y_i \right]$$

$$\beta_1 = (\beta_1)_{old} + \alpha \left[h_\theta(x_i) - y_i \right] \left[x_1 \right]$$

$$\beta_2 = (\beta_2)_{old} + \alpha \left[h_\theta(x_i) - y_i \right] \left[x_2 \right]$$

α is the learning rate.

Iterations are run until β_0, β_1 & β_2 converge.

Let's take examples now to compute the MATLAB output from logistic regression (through the Classification Learner App or through the *mnrfit* function) using the math developed above.

Example1: Scores of two exams have been given for various applicants and we need to classify whether they could be admitted to the university or not. Class 1 if they can be admitted and Class 0 if they can't be admitted.

The data for this example has been taken from here and has also been uploaded on the course website. Further discussion on this dataset can be found at this link.

MATLAB output: ***logreg_marks.mlx*** file has been uploaded with the data.

```
marks=table2array(marks);

x1=marks(:,1);
x2=marks(:,2);
x=[x1 x2];
y=marks(:,3);


% changing the class =0 to class =1 and vice versa.
for i=1:100

    if y(i) == 0
        y(i)=1
    else y(i) = 0
    end


end
```

```
% Performing logistic regression using MATLAB function mnrfit
% https://in.mathworks.com/help/stats/mnrfit.html
ycat=categorical(y);
[B,dev,stats] = mnrfit(x,ycat);
B
```

```
B = 3×1
  -25.1613
    0.2062
    0.2015
```

β_0, β_1, β_2

The model is: $y = \dfrac{1}{1+e^{-(-25.16+0.206x_1+0.2015x_2)}}$

For predictions, we can take values of x1 and x2 to get the probabilities and if probability > 0.5 then y = 1 else y = 0.

For e.g.: x1 = 60, x2 = 50,

$$p = \frac{1}{1+e^{-(-25.16+0.206\times 60+0.2015\times 50)}} = 0.06$$

Since $p \le 0.5$ so $y = 0$.

stats.p

```
p-value = 1.0e-04 *
    0.1430
    0.1736
    0.3422
```

p-values for $\beta_0, \beta_1, \beta_2$ show statistical significance.

stats.se

standard error (se)

```
    5.7986
    0.0480
    0.0486
```

The below MATLAB code implements the math developed initially.

```
%code for implementing gradient descent.

numrows=20000000; % number of iterations = 20 million

alpha=0.001;
```

α = learning rate

β_0 β_1 β_2

```
b0= zeros(numrows,1);
b1= zeros(numrows,1);
b2= zeros(numrows,1);


db0_avg= zeros(numrows,1);
db1_avg= zeros(numrows,1);
db2_avg= zeros(numrows,1);


datarows=length(x1);

yhat= zeros(datarows,1);
res= zeros(datarows,1);
db1= zeros(datarows,1);
db2= zeros(datarows,1);


b0(1)=0;
b1(1)=0;
b2(1)=0;
```

yhat = $h_\theta(x_i)$

```
for j=2:numrows
    for i=1:datarows
        yhat(i)=1/(1+exp(-1*(b0(j-1)+b1(j-1)*x1(i)+b2(j-1)*x2(i))));
        res(i)=yhat(i)-y(i);
        db1(i)=res(i)*x1(i);
        db2(i)=res(i)*x2(i);
    end
```

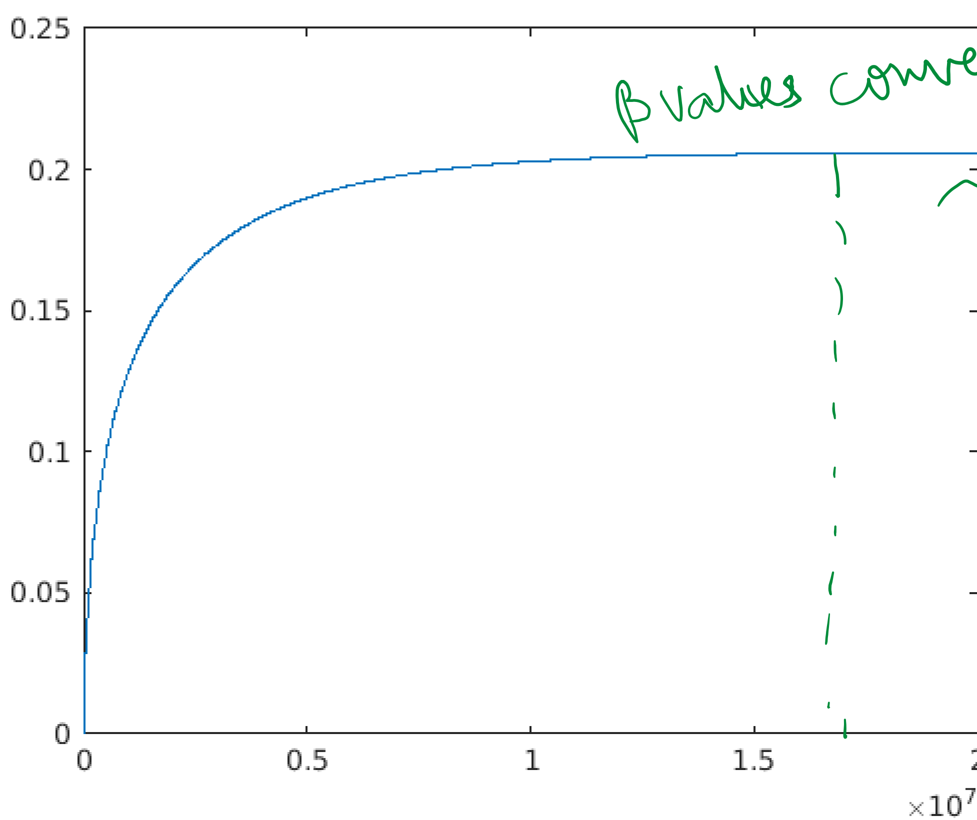
— eqn. (4)

```
    temp1=0;
    temp2=0;
    temp3=0;
    for i=1:datarows
        temp1=temp1+res(i);
        temp2=temp2+db1(i);
        temp3=temp3+db2(i);
    end
```

```
    db0_avg(j-1)=temp1/datarows;
    db1_avg(j-1)=temp2/datarows;
    db2_avg(j-1)=temp3/datarows;


    b0(j)=b0(j-1)-db0_avg(j-1)*alpha;
    b1(j)=b1(j-1)-db1_avg(j-1)*alpha;
    b2(j)=b2(j-1)-db2_avg(j-1)*alpha;


end

plot(b1)
```

i → every data point
j → iteration no.

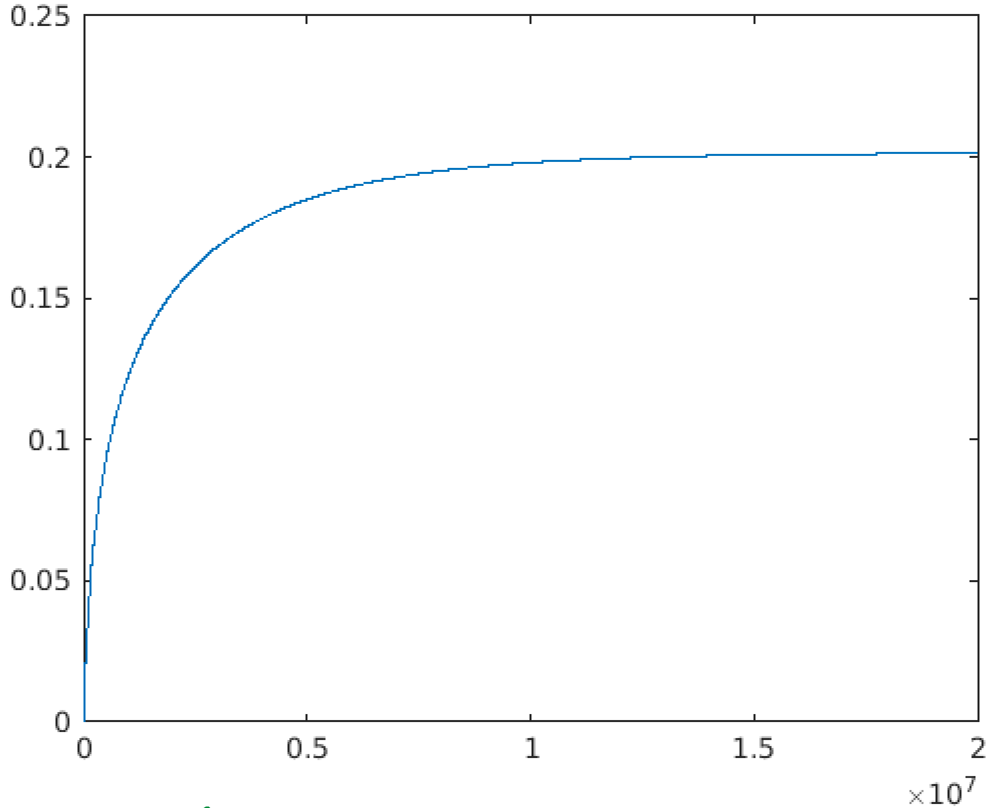
eqn. 5



β values converge at ~18 million iterations.

```
b1(20000000)
ans = 0.2061
plot(b2)
```

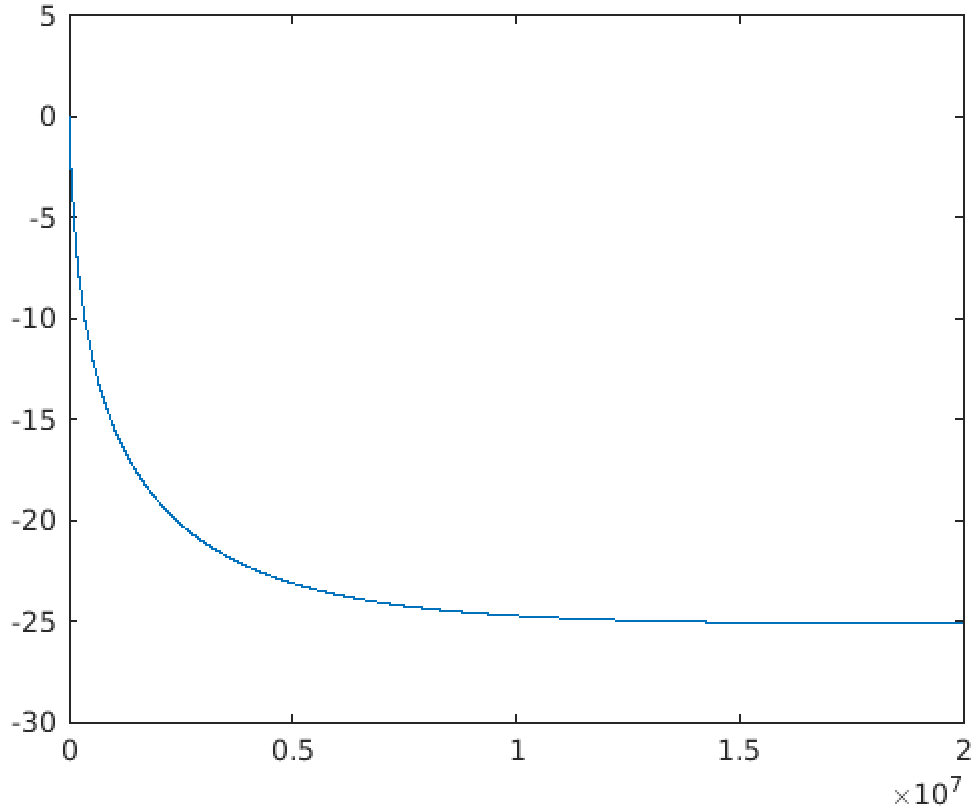
→ β_1 converged value matches MATLAB mvrfit output of β_1 above.




```
b2(20000000)
```

ans = 0.2013 ✓ β_2

```
plot(b0)
```




```
b0(20000000)
ans = -25.1401
```

✓ β_0

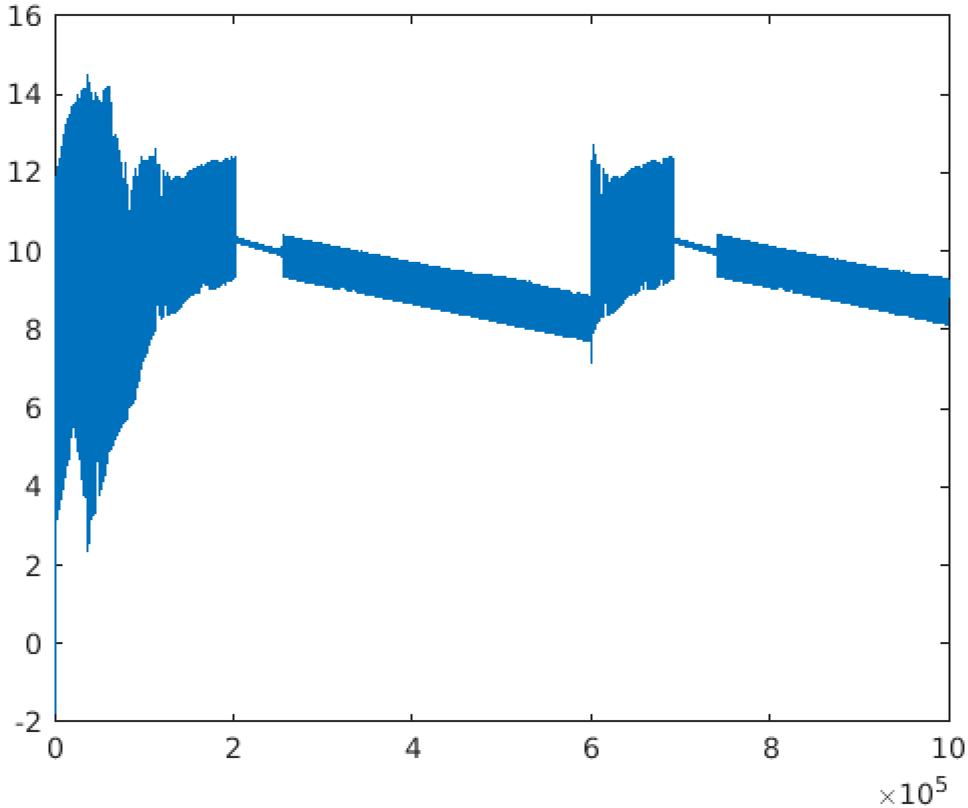
```
If we had chosen a different alpha and different number of iterations then the
output would have looked like:

numrows=1000000;
alpha=0.2;
```

→ 1 million iterations

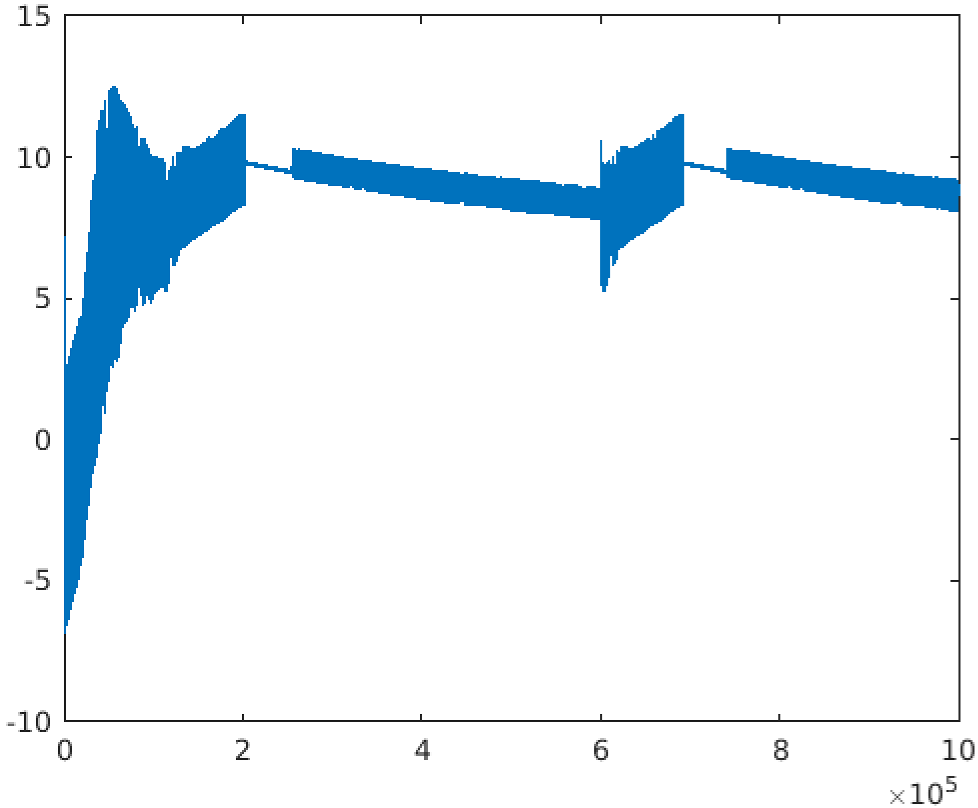
→ learning rate of 0.2.

```
b0= zeros(numrows,1);
b1= zeros(numrows,1);
b2= zeros(numrows,1);


db0_avg= zeros(numrows,1);
db1_avg= zeros(numrows,1);
db2_avg= zeros(numrows,1);


datarows=length(x1);
yhat= zeros(datarows,1);
res= zeros(datarows,1);
db1= zeros(datarows,1);
db2= zeros(datarows,1);
```

```
b0(1)=0;
b1(1)=0;
b2(1)=0;


for j=2:numrows
    for i=1:datarows
        yhat(i)=1/(1+exp(-1*(b0(j-1)+b1(j-1)*x1(i)+b2(j-1)*x2(i))));
        res(i)=yhat(i)-y(i);
        db1(i)=res(i)*x1(i);
        db2(i)=res(i)*x2(i);
    end

    temp1=0;
    temp2=0;
    temp3=0;
    for i=1:datarows
        temp1=temp1+res(i);
        temp2=temp2+db1(i);
        temp3=temp3+db2(i);
    end
    db0_avg(j-1)=temp1/datarows;
    db1_avg(j-1)=temp2/datarows;
    db2_avg(j-1)=temp3/datarows;


    b0(j)=b0(j-1)-db0_avg(j-1)*alpha;
    b1(j)=b1(j-1)-db1_avg(j-1)*alpha;
    b2(j)=b2(j-1)-db2_avg(j-1)*alpha;


end


plot(b1)
```



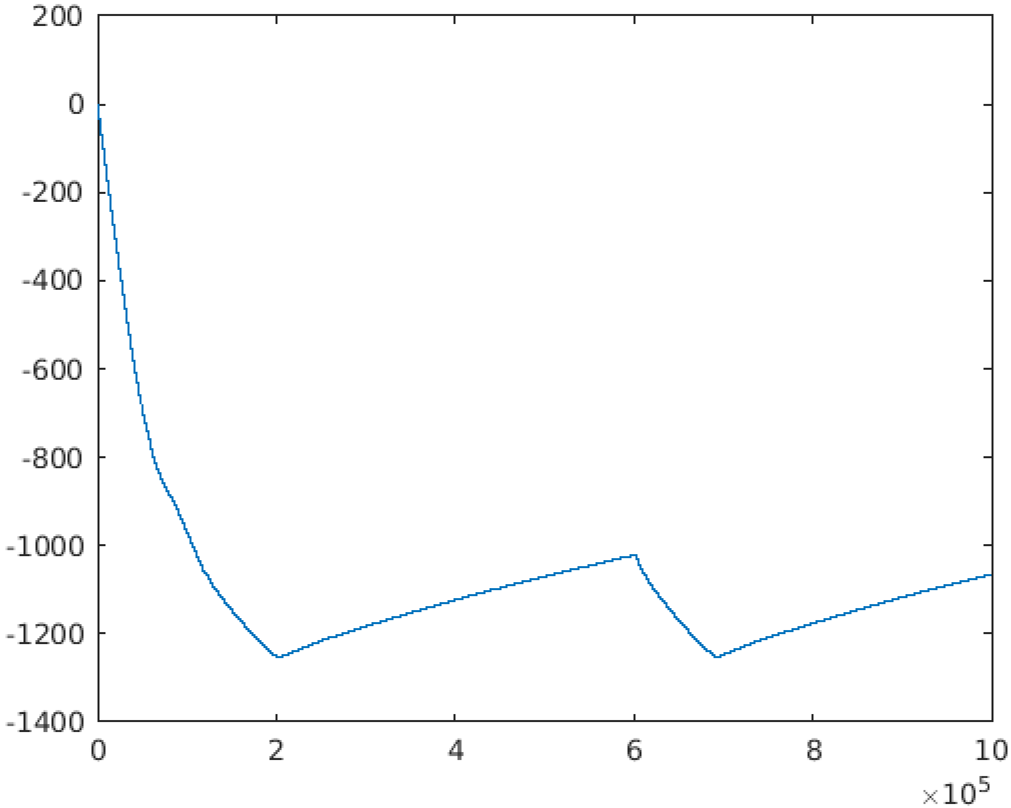

```
b1(1000000)
```
ans = 9.2762
```
plot(b2)
```




```
b2(1000000)
```

```
ans = 9.0696
```

```
plot(b0)
```




```
b0(1000000)
ans = -1.0648e+03
```

Likewise, we can perform binary classification for any number of predictors.

What about multi-class classification problems?

The multi-class classification problems can be broken into multiple binary classification problems.

Let's take an example. This code has been uploaded as ***logreg_fisheriris.mlx***.

```
%https://in.mathworks.com/help/stats/select-data-and-validation-for-classification-problem.html
fishertable = readtable('fisheriris.csv');

% https://in.mathworks.com/help/stats/mnrfit.html
load fisheriris

sp = categorical(species);
meas2=meas(:,[2]);


[B,dev,stats] = mnrfit(meas2,sp);
```

→ sepal width as predictor

class : setosa vs. virginica

versicolor vs. virginica

reference class.

```
B
B = 2×2
  -12.9973    5.8611
    4.0791   -2.0399
```

β_0, β_1

$$\ln\left(\frac{P_{setosa}}{P_{virginica}}\right) = -12.9973 + 4.0791X_1$$

$$\ln\left(\frac{P_{versicolor}}{P_{virginica}}\right) = 5.8611 - 2.0399X_1$$

```
stats.p
ans = 2×2
    0.0000    0.0035
    0.0000    0.0033
```

```
stats.se
ans = 2×2
    2.6883    2.0046
    0.8436    0.6933
```

```
%change species - sp to a number.
for i=1:150
    if sp(i) == "setosa"
        sp(i) = "2";
    end
    if sp(i) == "versicolor"
        sp(i) = "1";
    end
    if sp(i) == "virginica"
        sp(i) = "0";
    end
end
sp=double(sp);
for i=1:150
    if sp(i) == 4
        sp(i) = 2;
    end
    if sp(i) == 5
        sp(i) = 1;
    end
    if sp(i) == 6
        sp(i) = 0;
    end
end
```

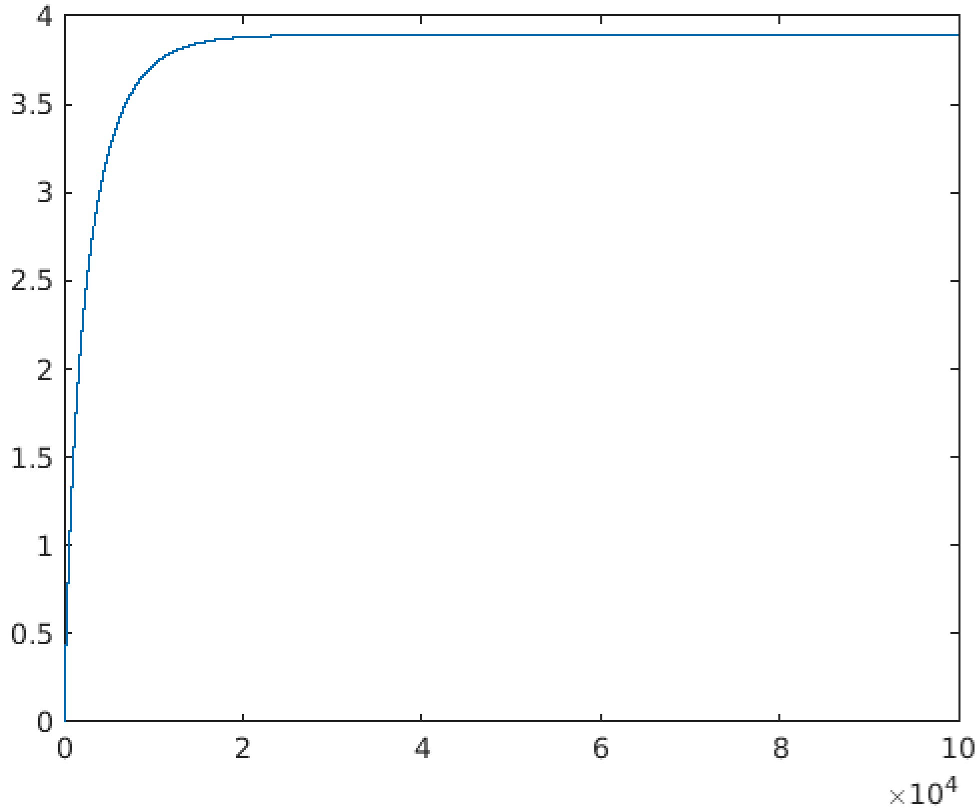
```
First we take the binary classification of setosa and virginica.

y=sp(:,1);
y = y( [51:end] , : ); % y value would be 1 or 0 only.
x=meas2(:,1);
x = x( [1:50,101:end] , : ); % once we take setosa = 1 class and virginica = 0
class %3.8, -12
%x = x( [51:end] , : );% once we take versicolor = 1 class and virginica = 0
class %-2,6
```

Here, $\beta_1 = a$

$\beta_0 = b$

$$\boxed{p = \frac{1}{1+e^{-(ax+b)}}}$$

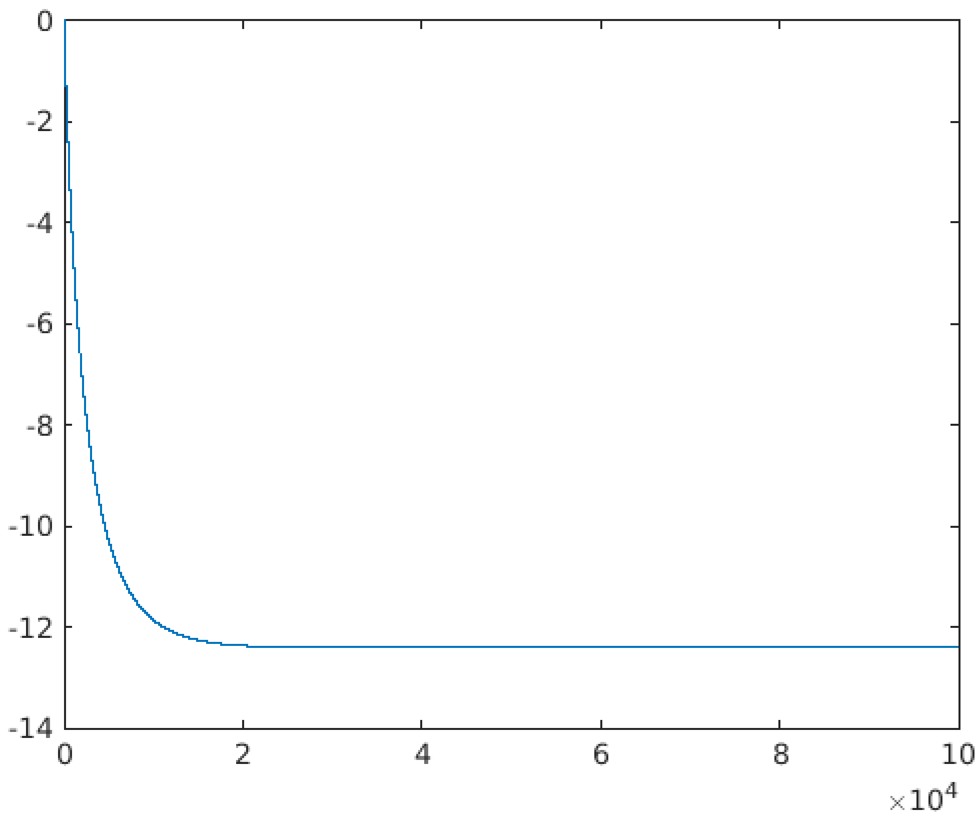
```
numrows=100000;
alpha=0.2;
datarows=length(y);
a= zeros(numrows,1);
b= zeros(numrows,1);
da_avg= zeros(numrows,1);
db_avg= zeros(numrows,1);
yhat= zeros(datarows,1);
res= zeros(datarows,1);
da= zeros(datarows,1);
a(1)=0;
b(1)=0;


for j=2:numrows
    for i=1:datarows
        yhat(i)=1/(1+exp(-1*(a(j-1)*x(i)+b(j-1))));
        res(i)=yhat(i)-y(i);
        da(i)=res(i)*x(i);
    end

    temp1=0;
    temp2=0;
    for i=1:datarows
        temp1=temp1+da(i);
        temp2=temp2+res(i);
    end
    da_avg(j-1)=temp1/datarows;
    db_avg(j-1)=temp2/datarows;

    a(j)=a(j-1)-da_avg(j-1)*alpha;
    b(j)=b(j-1)-db_avg(j-1)*alpha;


end
plot(a)
```




```
a(numrows)
```

ans = 3.8927

```
plot(b)
```



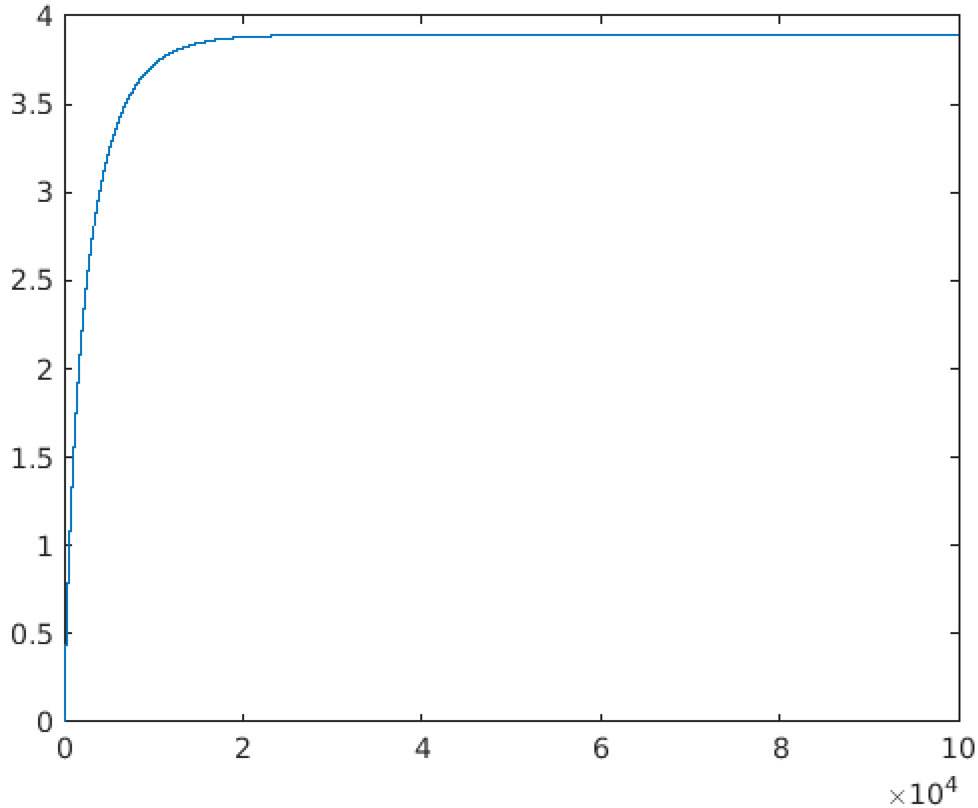

```
b(numrows)
ans = -12.4125
```

Next we consider the case of versicolor as class 1 and virginica as class 0.

```
y=sp(:,1);
y = y( [51:end] , : ); % y value would be 1 or 0 only.
x=meas2(:,1);
%x = x( [1:50,101:end] , : ); % once we take setosa = 1 class and viriginica = 0 
class %3.8, -12
x = x( [51:end] , : );% once we take versicolor = 1 class and viriginica = 0 
class %-2,6
```

```
numrows=100000;
alpha=0.2;
datarows=length(y);
a= zeros(numrows,1);
b= zeros(numrows,1);
da_avg= zeros(numrows,1);
db_avg= zeros(numrows,1);
yhat= zeros(datarows,1);
res= zeros(datarows,1);
```

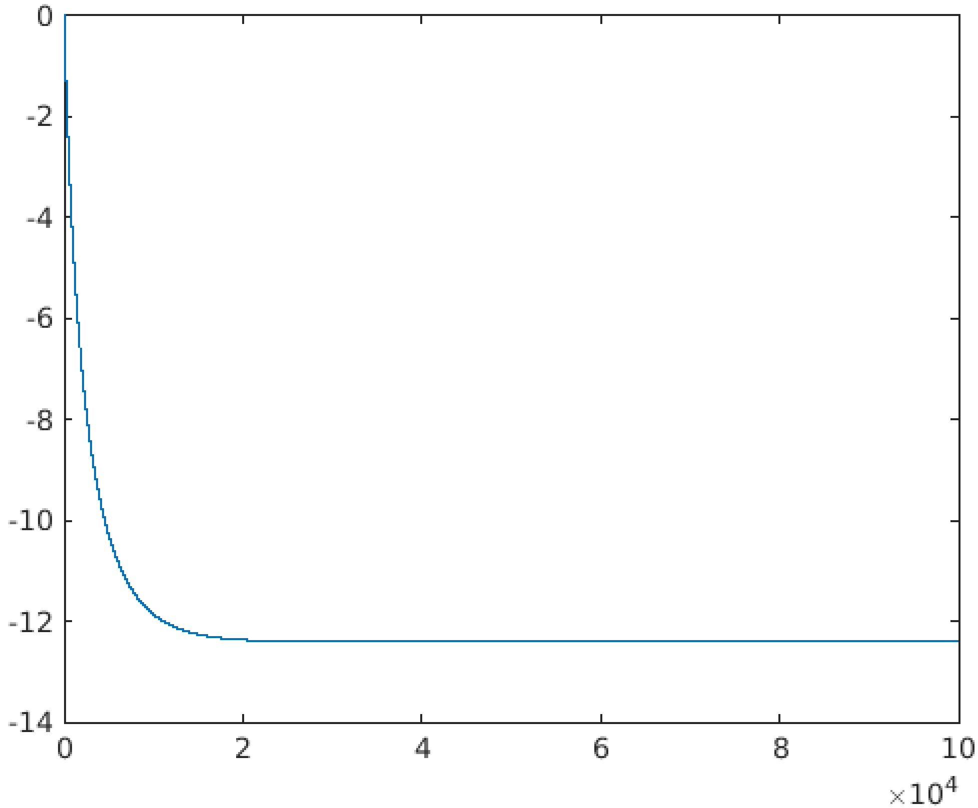
```
da= zeros(datarows,1);
a(1)=0;
b(1)=0;


for j=2:numrows
    for i=1:datarows
        yhat(i)=1/(1+exp(-1*(a(j-1)*x(i)+b(j-1))));
        res(i)=yhat(i)-y(i);
        da(i)=res(i)*x(i);
    end

    temp1=0;
    temp2=0;
    for i=1:datarows
        temp1=temp1+da(i);
        temp2=temp2+res(i);
    end
    da_avg(j-1)=temp1/datarows;
    db_avg(j-1)=temp2/datarows;

    a(j)=a(j-1)-da_avg(j-1)*alpha;
    b(j)=b(j-1)-db_avg(j-1)*alpha;


end
plot(a)
```




```
a(numrows)
```

```
ans = -2.0895
```

```
plot(b)
```




```
b(numrows)
```

```
ans = 6.0011
```

The results are very close for a multi-class classification problem with 3 classes and 1 predictor. Likewise, logistic regression can be implemented for any number of classes and any number of predictors.