Indian Institute of Technology (Indian School of Mines), Dhanbad

Project: Logistic Regression

Submitted by: -

Sambhavesh Haralalka

16JE002482

Section-II

INDEX

Contents | Page

Contents	Page
1) Introduction	1
2) PowerPoint Presentation	4
3) Codes	
a) Two Classes	24
b) Three Classes	28
4) Result	33

Introduction

In this project I have tried to implement two data sets namely:

1) Diffuse large b-cell lymphomas (DLBCL) and follicular lymphomas which is a two class data set.
2) Acute myelogenous leukemia (AML), acute lympboblastic leukemia (ALL) and mixed-lineage leukemia (MLL) which is three class data set.

I have used Logistic Regression Classifier to implement the above two data sets and used two optimization functions namely fminunc and fminsearch and noted the accuracy for both functions while changing the number of features.

fminunc

Find minimum of unconstrained multivariable function

Nonlinear programming solver.

Finds the minimum of a problem specified by

$$\min_{x} f(x)$$

where $f(x)$ is a function that returns a scalar.

x is a vector or a matrix; see Matrix Arguments.

Syntax

```
x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
x = fminunc(problem)
[x,fval] = fminunc( __ )
[x,fval,exitflag,output] = fminunc( __ )
[x,fval,exitflag,output,grad,hessian] = fminunc( __ )
```

Description

`x = fminunc(fun,x0)` starts at the point `x0` and attempts to find a local minimum `x` of the function described in `fun`. The point `x0` can be a scalar, vector, or matrix. example

`x = fminunc(fun,x0,options)` minimizes `fun` with the optimization options specified in `options`. Use `optimoptions` to set these options. example

`x = fminunc(problem)` finds the minimum for `problem`, where `problem` is a structure described in Input Arguments. Create the `problem` structure by exporting a problem from Optimization app, as described in Exporting Your Work. example

`[x,fval] = fminunc( __ )`, for any syntax, returns the value of the objective function `fun` at the solution `x`. example

`[x,fval,exitflag,output] = fminunc( __ )` additionally returns a value `exitflag` that describes the exit condition of `fminunc`, and a structure `output` with information about the optimization process. example

fminsearch

Find minimum of unconstrained multivariable function using derivative-free method

Nonlinear programming solver. Searches for the minimum of a problem specified by

$$\min_{x} f(x)$$

$f(x)$ is a function that returns a scalar, and x is a vector or a matrix.

Syntax

```
x = fminsearch(fun,x0)
x = fminsearch(fun,x0,options)
x = fminsearch(problem)
[x,fval] = fminsearch( __ )
[x,fval,exitflag] = fminsearch( __ )
[x,fval,exitflag,output] = fminsearch( __ )
```

Description

`x = fminsearch(fun,x0)` starts at the point `x0` and attempts to find a local minimum `x` of the function described in `fun`. example

`x = fminsearch(fun,x0,options)` minimizes with the optimization options specified in the structure `options`. Use `optimset` to set these options. example

`x = fminsearch(problem)` finds the minimum for `problem`, where `problem` is a structure.

`[x,fval] = fminsearch( __ )`, for any previous input syntax, returns in `fval` the value of the objective function `fun` at the solution `x`. example

`[x,fval,exitflag] = fminsearch( __ )` additionally returns a value `exitflag` that describes the exit condition.

`[x,fval,exitflag,output] = fminsearch( __ )` additionally returns a structure `output` with information about the optimization process. example

Logistic Regression

Classification

- Email:Spam/Not Spam?
- Online Transactions: Fraudulent (Yes / No)

$y \in \{0, 1\}$ 0:“Negative Class”

1:“Positive Class”

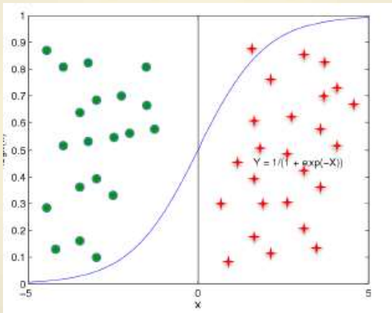
Logistic Regression: $0 \leq h_\theta(x) \leq 1$

Logistic Regression

Hypothesis Representation

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

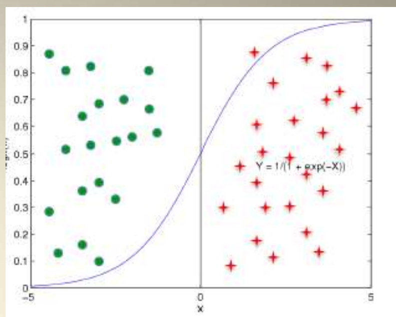
$$h_\theta(x) = \frac{1}{1+e^{-(\theta^T X)}}$$



- Outcome
  - We predict the *probability* of the outcome occurring

Logistic regression

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



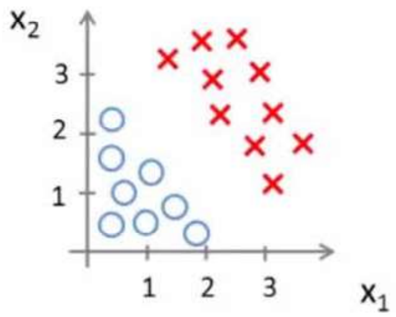
Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

Whenever $\Theta^T X > 0$

predict "$y = 0$" if $h_\theta(x) < 0.5$

Whenever $\Theta^T X < 0$

Decision Boundary

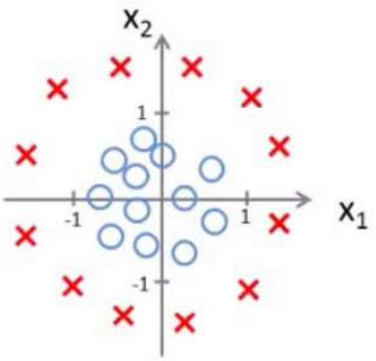


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- Let ϴ0 = -3, ϴ1 = 1, ϴ2 = 1

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

Cost function

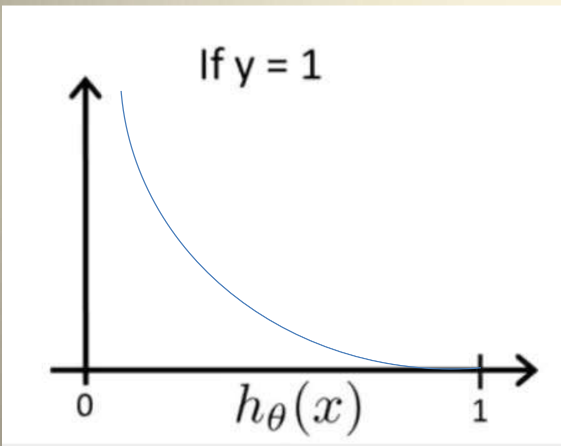
Linear regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

$$\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Non Convex

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

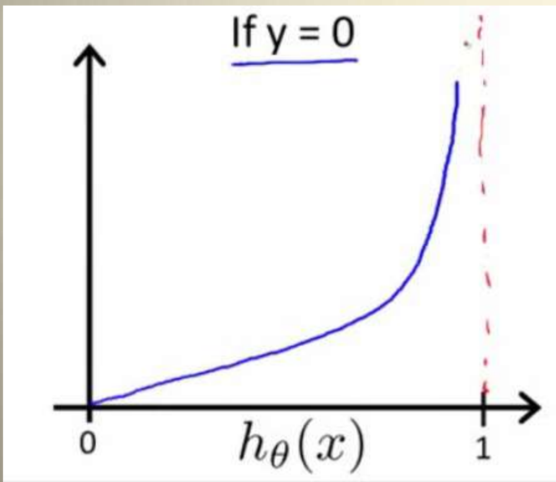


Cost $= 0$ if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
$Cost \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x;\theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

Logistic regression cost function

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right] \end{aligned}$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

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

$\}$ (simultaneously update all θ_j)

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$] ;

gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$] ;

gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$] ;

$\vdots$

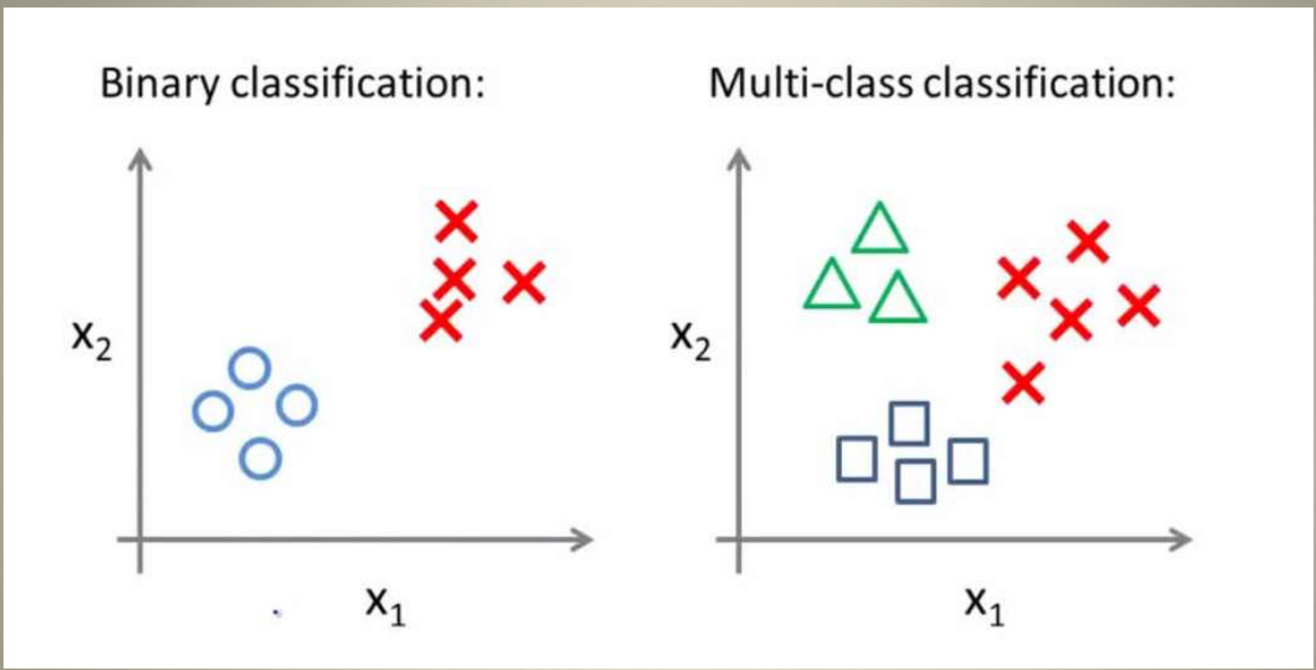
gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$] ;

Multiclass classification

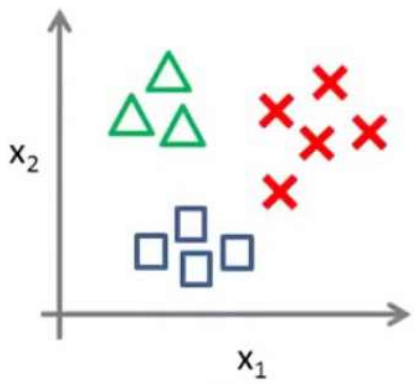
Email foldering/tagging: Work, Friends, Family, Hobby

Medical diagrams: Not ill, Cold, Flu

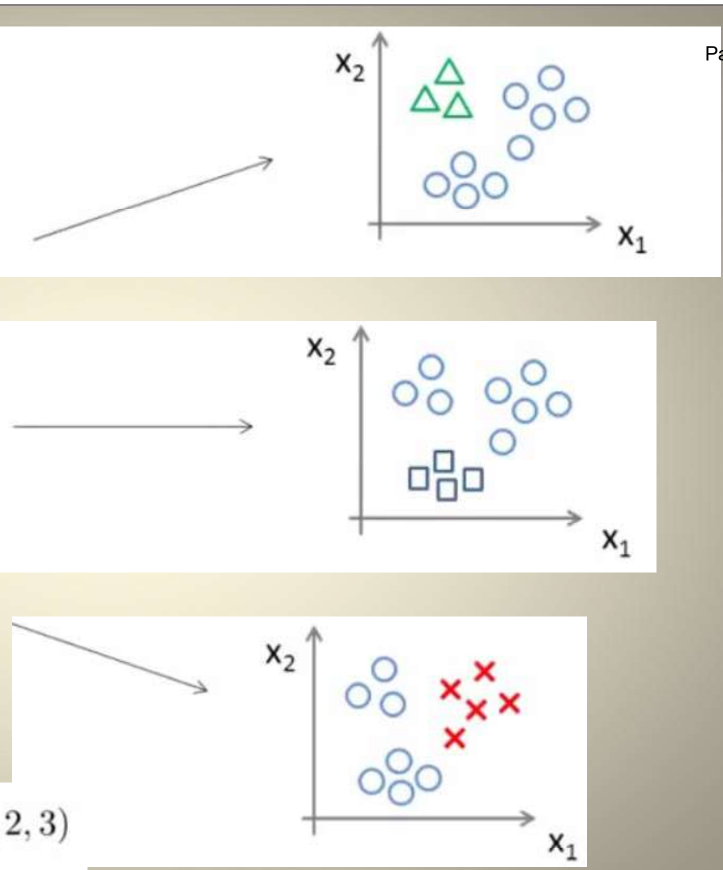
Weather: Sunny, Cloudy, Rain, Snow


Binary classification:
x_2
x_1
Multi-class classification:
x_2
x_1

One-vs-all (one-vs-rest):



Class 1: △
Class 2: □
Class 3: ✕



$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

END

```matlab
clear; close all; clc

%% Load Data
data = load('data.txt');
X = data(:, 2:101); y = data(:, 1);
data_test = load('datatest.txt');
X_test = data_test(:, 2:101);  y_test = data_test(:, 1);
%% ============ Compute Cost and Gradient ============
[m, n] = size(X);
[m1, n1] = size(X_test);
X = [ones(m, 1) X];
X_test = [ones(m1, 1) X_test];
initial_theta = zeros(n + 1, 1);

% Compute and display initial cost and gradient
[cost, grad] = costFunction(initial_theta, X, y);

fprintf('Cost at initial theta (zeros): %f\n', cost);
% fprintf('Gradient at initial theta (zeros): \n');
% fprintf(' %f \n', grad);



%% ============= Optimizing using fminunc  =============
options = optimset('GradObj', 'on', 'MaxIter', 400,'TolFun', 01e-10);
[theta, cost] = ...
    fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);

fprintf('Cost at theta found by fminunc: %f\n', cost);
% fprintf('theta: \n');
% fprintf(' %f \n', theta);



%% ============== Predict and Accuracies ==============

p = predict(theta, X);
fprintf('Train Accuracy: %f\n', mean(double(p == y)) * 100);
ptest = predict(theta, X_test);
fprintf('Testing Accuracy: %f\n', mean(double(ptest == y_test)) * 100);
```

```
function [J, grad] = costFunction(theta, X, y)
%COSTFUNCTION Compute cost and gradient for logistic regression
%   J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the
%   parameter for logistic regression and the gradient of the cost
%   w.r.t. to the parameters.

% Initialize some useful values
m = length(y); % number of training examples

grad = zeros(size(theta));

h = sigmoid(X * theta);
J = -(1 / m) * sum( (y .* log(h)) + ((1 - y) .* log(1 - h)) );

for i = 1 : size(theta, 1)
    grad(i) = (1 / m) * sum( (h - y) .* X(:, i) );
end

end
```

```
function p = predict(theta, X)
%PREDICT Predict whether the label is 0 or 1 using learned logistic
%regression parameters theta
%   p = PREDICT(theta, X) computes the predictions for X using a
%   threshold at 0.5 (i.e., if sigmoid(theta'*x) >= 0.5, predict 1)

m = size(X, 1); % Number of training examples

p = round(sigmoid(X * theta));

end
```

```
function g = sigmoid(z)
%SIGMOID Compute sigmoid functoon
%   J = SIGMOID(z) computes the sigmoid of z.

g = 1 ./ (1 + exp(-z));


end
```

```
clear; close all; clc

%% Load Data
data = load('LeukemiaTraining.txt');
X = data(:, 2:101); y = data(:, 1);
data_test = load('LeukemiaTesting.txt');
X_test = data_test(:, 2:101);  y_test = data_test(:, 1);


s = size(y,1);
y1 = zeros(1,1);
for i = 1:s
    if(y(i)==0)
        y1 = [y1;1];
    else
        y1 = [y1;0];
    end
end
y1 = y1(2:s+1);

y2 = zeros(1,1);
for i = 1:s
    if(y(i)==1)
        y2 = [y2;1];
    else
        y2 = [y2;0];
    end
end
y2 = y2(2:s+1);
y3 = zeros(1,1);
for i = 1:s
    if(y(i)==2)
        y3 = [y3;1];
    else
        y3 = [y3;0];
    end
end
y3 = y3(2:s+1);


%% ============ Compute Cost and Gradient ============

[m, n] = size(X);
[m1, n1] = size(X_test);
X = [ones(m, 1) X];
X_test = [ones(m1, 1) X_test];
initial_theta = zeros(n + 1, 1);

% Compute and display initial cost and gradient
[cost1, grad1] = costFunction(initial_theta, X, y1);
```

```
[cost2, grad2] = costFunction(initial_theta, X, y2);
[cost3, grad3] = costFunction(initial_theta, X, y3);

 fprintf('Cost at initial theta(zeros) for class 0: %f\n', cost1);
 fprintf('Cost at initial theta(zeros) for class 1: %f\n', cost2);
 fprintf('Cost at initial theta(zeros) for class 2: %f\n', cost3);
% fprintf('Gradient at initial theta (zeros): \n');
% fprintf(' %f \n', grad);



%% ============= Optimizing using fminunc or fminsearch =============
 options = optimset('GradObj', 'on', 'MaxIter', 400, 'TolFun',1e-6, 'TolX',1e-7);

[theta1, cost1] = ...
    fminsearch(@(t)(costFunction(t, X, y1)), initial_theta, options);
[theta2, cost2] = ...
    fminsearch(@(t)(costFunction(t, X, y2)), initial_theta, options);
[theta3, cost3] = ...
    fminsearch(@(t)(costFunction(t, X, y3)), initial_theta, options);

% Print theta to screen
fprintf('Cost at theta found by fminunc for class 0: %f\n', cost1);
fprintf('Cost at theta found by fminunc for class 1: %f\n', cost2);
fprintf('Cost at theta found by fminunc for class 2: %f\n', cost3);
% fprintf('theta: \n');
% fprintf(' %f \n', theta);



%% ============== Predict and Accuracies ==============

p = predict(theta1,theta2,theta3, X);
fprintf('Train Accuracy: %f\n', mean(double(p == y)) * 100);
ptest = predict(theta1,theta2,theta3, X_test);
fprintf('Testing Accuracy: %f\n', mean(double(ptest == y_test)) * 100);
```

```
function [J, grad] = costFunction(theta, X, y)
%COSTFUNCTION Compute cost and gradient for logistic regression
%   J = COSTFUNCTION(theta, X, y) computes the cost of using theta as the
%   parameter for logistic regression and the gradient of the cost
%   w.r.t. to the parameters.


m = length(y);
grad = zeros(size(theta));

h = sigmoid(X * theta);
J = -(1 / m) * sum( (y .* log(h)) + ((1 - y) .* log(1 - h)) );

for i = 1 : size(theta, 1)
    grad(i) = (1 / m) * sum( (h - y) .* X(:, i) );
end

end
```

```
function p = predict(theta1,theta2,theta3, X)
%PREDICT Predict whether the label is 0 or 1 or 2 using learned logistic
%regression parameters theta


m = size(X, 1); % Number of training examples
H1 = zeros(1,1);
H2 = zeros(1,1);
H3 = zeros(1,1);
p = 10*ones(1,1);

H1 = sigmoid(X * theta1);
H2 = sigmoid(X * theta2);
H3 = sigmoid(X * theta3);



for i = 1:m

    if (H1(i)>=H2(i))
        if(H1(i)>= H3(i))
            p = [p;0];
        end
    end
    if (H2(i)>=H1(i))
        if(H2(i)>=H3(i))
            p = [p;1];
        end

    end
    if (H3(i)>=H2(i))
        if(H3(i)>=H1(i))
            p = [p;2];
        end
    end
end

p = p(2:m+1);


end
```

```
function g = sigmoid(z)
%SIGMOID Compute sigmoid functoon
%   J = SIGMOID(z) computes the sigmoid of z.

g = 1 ./ (1 + exp(-z));


end
```

Optimizer: fminsearch

<u>2 classes</u>
0: 45(training), 13(testing)
1: 14(training), 5(testing)
Initial cost (theta = 0) = 0.693147

Feat.	Iter	Train%	Test%	Cost
100	400	84.75	72.22	0.300123
100	800	89.83	77.78	0.227026
100	3200	100.00	88.89	0.003393
100	6400	100.00	89.89	0.000001
100	12800	100.00	89.89	0.000001

Feat.	Iter	Train%	Test%	Cost
200	400	83.05	66.67	0.403405
200	800	88.14	66.67	0.301087
200	3200	94.92	77.78	0.118487
200	6400	100.00	83.33	0.018970
200	12800	100.00	89.89	0.000001

Feat.	Iter	Train%	Test%	Cost
400	400	77.97	72.22	0.390960
400	800	77.97	72.22	0.390960
400	3200	93.22	77.78	0.235373
400	6400	100.00	83.33	0.116195
400	12800	100.00	88.89	0.020235

Feat.	Iter	Train%	Test%	Cost
1600	400	76.27	72.22	0.360138
1600	800	76.27	72.22	0.360138
1600	3200	76.27	72.22	0.360138
1600	6400	84.75	66.67	0.327079
1600	12800	93.22	77.78	0.223446

Feat.	Iter	Train%	Test%	Cost
5470	400	77.97	72.22	0.339391
5470	3200	77.97	72.22	0.339391
5470	6400	77.97	72.22	0.339391

<u>3 classes</u>

0: 20(training), 8(testing)
1: 20(training), 4(testing)
2: 17(training), 3(testing)
Initial cost (theta = 0) = 0.693147

Feat.	Iter	Train%	Test%
100	800	78.95	86.67
100	1600	87.72	80.00
100	3200	100.00	73.33
100	6400	100.00	80.00

Feat.	Iter	Train%	Test%
400	800	78.95	60.00
400	1600	92.98	80.00
400	3200	96.49	86.67
400	6400	98.25	86.67

Feat.	Iter	Train%	Test%
1600	800	70.18	73.33
1600	1600	70.18	73.33
1600	6400	100.00	80.00

Feat.	Iter	Train%	Test%
11226	400	87.72	86.67

Optimizer: fminunc

2 classes

0: 45(training), 13(testing)
1: 14(training), 5(testing)
Initial cost (theta = 0) = 0.693147
Tolerance = 01e-10

Feat.	Iter	Train%	Test%	Cost
100	200	100	94.44	2.2484e-10
200	200	100	88.89	4.4649e-10
400	200	100	88.89	4.7744e-10
1600	200	100	88.89	3.7502e-10

3 classes

0: 20(training), 8(testing)
1: 20(training), 4(testing)
2: 17(training), 3(testing)
Initial cost (theta = 0) = 0.693147
Tolerance = 01e-06

Feat.	Iter	Train%	Test%
100	400	100.00	93.33
200	400	100.00	100.00
400	400	100.00	93.33
1600	400	100.00	93.33