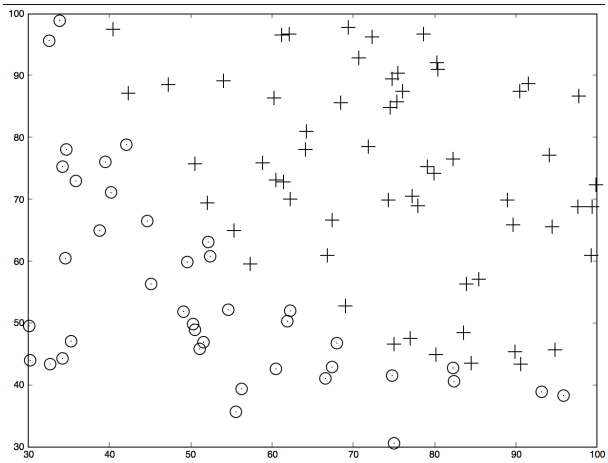
1 Logistic Regression

1.1 visualizing the data

代码：

```
pos = find(y==1); neg = find(y==0);
plot(X(pos,1),X(pos,2),'k+','LineWidth',2,'MarkerSize',7);
plot(X(neg,1),X(neg,2),'ko','MarkerFaceColor','y','MarkerSize',7);
```



1.2 implementation

1.2.1 sigmoid function

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

method(it should perform the sigmoid function on every element):

```
function g = sigmoid(z)
g = zeros(size(z));  %initialize
g = 1./(ones(size(z))+exp(-z));
end
```

1.2.1 Cost function and gradient

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

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

method:

```
function [J,grad] = costFunction(theta,X,y)
m = length(y);
J = 0;
grad = zeros(size(theta));
% -ylog() - (1-y)log
```

```
x = x' ;
h = sigmoid(theta' * x);  % size(1,20),while y:size(20,1)
y_ = y';
temp_matrix = (-y_) .* log(h) - (ones(size(theta'))-y_) .* log(ones(size(theta')-h)) ;
J = sum(temp)/m ;

for i=1:m,
grad(i) = sum((h - y_) .* x(i,:))/m;
end;
```

1.2.3 Learning parameters using fminunc

fminunc is an optimization solver that finds the minimum of an unconstrained function.

Code :

```
%set options for fminuc
options = optimset('GradObj','on','MaxIter',400);
[theta,cost] = fminunc(@(t)(costFunction(t,X,y)),initial_theta,options)
```

1.2.4 Evaluating logistic regression

use the model to predict whether a student can be permitted given his/her grades

Code:

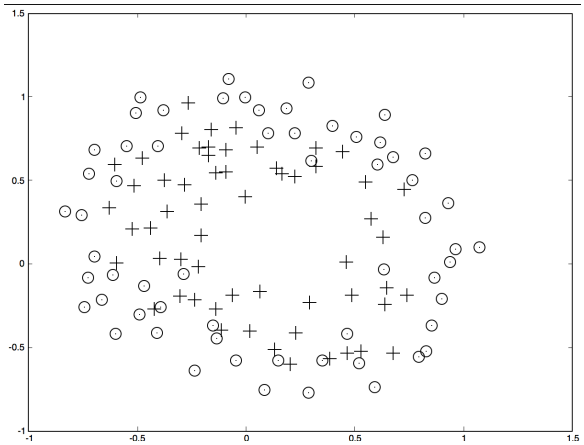
```
function p = predict(theta,X)
m = size(X,1);
p = zeros(m,1); %initialize

v = sigmoid(X*theta);
for i=1:m
  if v(i,1)>=0.5
      p(i,1)=1
  else p(i,1)=0;
end
```

建立在sigmoid上函数的预估模型，**如果sigmoid（）>=0.5,则估计值为1；否则为0**

2 Regularized logistic regression

2.1 visualizing the data



2.2 feature mapping

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

As a result of this mapping, our vector of two features (the scores on two QA tests) has been transformed into a 28-dimensional vector. A logistic regression classifier trained on this higher-dimension feature vector will have a more complex decision boundary and will appear nonlinear when drawn in our 2-dimensional plot.

While the feature mapping allows us to build a more expressive classifier, it also more susceptible to overfitting. In the next parts of the exercise, you will implement regularized logistic regression to fit the data and also see for yourself how regularization can help combat the overfitting problem.

code:

```
function out = mapFeature(X1,X2)
degree = 6;
out ones(size(X1(:,1)));
for i=1:degree
     for j=0:i
          out(:,end+1) = (X1.^(i-j))*(X2.^j);
     end
end
end
```

2.3 Cost function and gradient

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

Note that you should not regularize the parameter θ_0. In **Octave/MATLAB**, recall that indexing starts from 1, hence, you should not be regularizing the `theta(1)` parameter (which corresponds to θ_0) in the code. The gradient of the cost function is a vector where the j^{th} element is defined as follows:

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

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

这里要注意的是要将theta(1)情况特殊化处理（其中J和grad（i）是原cost function中）

```
J = J + lameda(sum(theta .^ 2)-theta(1)^2))
grad(1) = grad(1)
for i=2:m
   grad(i) = grad(i) + lameda*theta(i)/m
end
```