EXERCICE DE PROGRAMMATION 2

Logistic Regression

GAO BIN

30/11/2016

Introduction

Dans ce TP, on va définir une fonction de deux notes pour évoluer un étudiant sera admis dans une université.

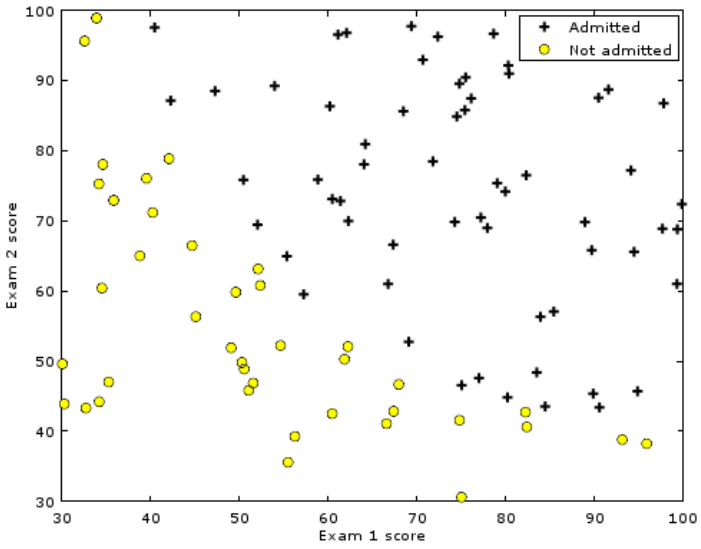
1. Logistic Regression

1.1 Visualizing the data

Les données des examens sont stockées dans le fichier ex2data1.txt, et les données sont sous la forme de deux valeurs réelles.

```
data = load('ex2data1.txt');
X = data(:, [1, 2]); y = data(:, 3);
```

Dans la première partie de la ex2.m, le code chargera les données et l'affichera sur un tracé bidimensionnel en appelant la fonction plotData, en prenant comme paramètres X et Y et nomme les axes, donc on peut compléter le code dans plotData afin qu'il affiche une figure comme la figure 1, où les axes sont les deux scores d'examen, et les exemples positifs et négatifs sont montrés avec des marqueurs différents.
On peut obtenir suivant :



1.2 Implementation

1.2.1 Warmup exercise : sigmoid function

On défini la fonction g est la fonction sigmoide :

```
12
13 g = 1./(1+exp(-z));
14
```

Pour vérifier la fonction, on essaye de tester quelques valeurs en appelant sigmoid (x) à la ligne de commande octave. Pour les grandes valeurs positives de x, le sigmoïde devrait être proche de 1, tandis que pour les grandes valeurs négatives, le sigmoïde devrait être proche de 0 :

```
>> sigmoid(15)
ans =  1.00000
>> sigmoid(1)
ans =  0.73106
>> sigmoid(0)
ans =  0.50000
>> sigmoid(-8)
ans =   3.3535e-004
>> sigmoid(-10)
ans =   4.5398e-005
>> sigmoid(-1)
ans =  0.26894
```

Donc on a bien résultat de tester.

1.2.2 Cost function and gradient

Complétez le code dans costFunction.m pour retourner le coût et le dégradé. On a cost fonction et gradient suivant :

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

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

On complète le code dans Octave :

```
23 J = 1 / m * sum(-y' * log(sigmoid(theta'*X')') -(1 - y)' * log(1 - sigmoid(theta'*X'))');
24 grad = 1 / m .* (sigmoid(theta' * X') - y') * X;
```

Dans le programme ex2.m, on initialise les paramètres initial theta, X et y, et puis on rentre dans la fonction Cost function. On peut obtenir le coût est de 0,693147.

```
Program paused. Press enter to continue.
Cost at initial theta (zeros): 0.693147
```

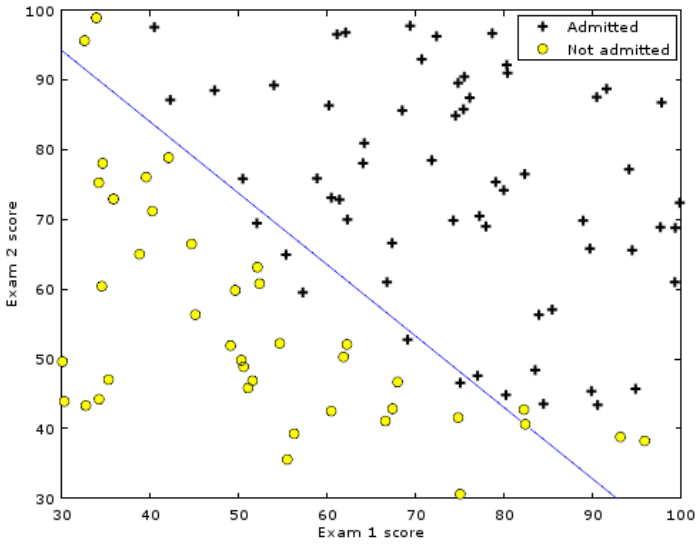
1.2.3 Learning parameters using fminunc

On passe à fminunc, on a terminé correctement la fonction costFunction, fminunc converge sur les bons paramètres d'optimisation et renvoie les valeurs finales du coût et θ. Une fois fminunc terminée, ex2.m appellera votre fonction costFunction en

utilisant les paramètres optimaux de θ, on peut trouver le minimum de la fonction coût avec les variables X et y fixées, donc on a :

```
Program paused. Press enter to continue.
Cost at theta found by fminunc: 0.203498
```

Cette valeur thêta sera utiliser pour tracer la frontière de décision sur les données d'examen.



Au-dessus du droite, les élèves sont considéré comme admis, mais en dessous, ils ne sont pas considéré.

1.2.4 Evaluating logistic regression

Après avoir appris les paramètres, on peut utiliser le modèle pour prédire si un étudiant particulier sera admis, on complète le code de la fonction predict :

```
18 p= sigmoid(X*theta) >= 0.5;
```

Pour un étudiant avec un score d'examen 1 de 45 et un score d'examen 2 de 85, vous devriez vous attendre à voir une probabilité d'admission de 0,776298. On a résultat suivant :

```
124 prob = sigmoid([1 45 85] * theta);
125 fprintf(['For a student with scores 45 and 85, we predict an admission '
126          'probability of %f\n\n'], prob);
```

```
Program paused. Press enter to continue.
For a student with scores 45 and 85, we predict an admission probability of 0.776289
```

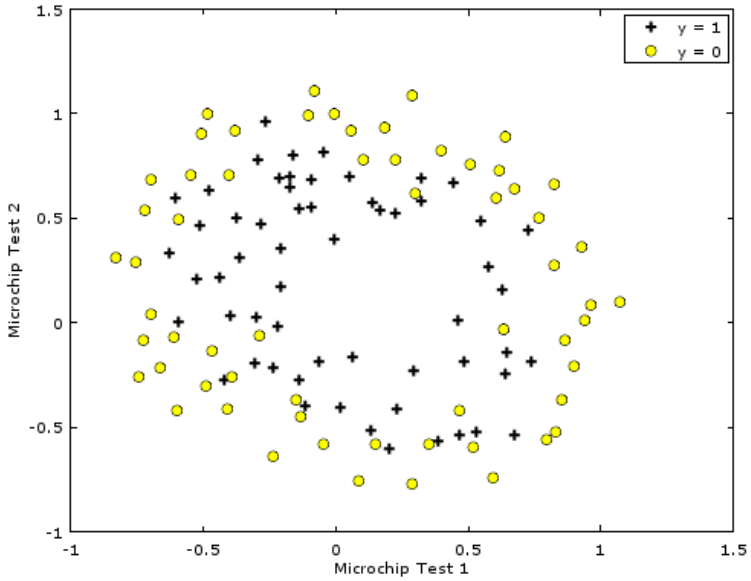
2. Regression logistic regression

Dans cette partie de l'exercice, on met en œuvre une régression logistique régulée

pour prédire si les puces reçues d'une usine de fabrication passent l'assurance de la qualité, on a les données de deux tests et s'il a été accepté ou non.

2.1 Visualizing the data

Dans la figure, les axes sont les deux scores de test, et les exemples positifs (y = 1, accepté) et négatif (y = 0, non accepté).



Les pièces acceptées sont représentées par des croix noirs et les pièces refusées par des ronds jaunes. Dans le cas présent, on ne peut pas séparer les données par une ligne droite, la forme à adopter ici est une ligne courbe fermée. C'est pourquoi on ne peut pas utiliser la régression logistique comme au premier exemple.

2.2 Feature mapping

Pour adapter les données, on doit créer plus de fonctionnalités à partir de chaque point de données.

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

2.3 Cost function and gradient

On a :

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

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

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

Donc on complète le code :

```
20 h = sigmoid(X*theta);
21 J = 1/m*sum(-y.*log(h)-(1-y).*log(1-h))+0.5*lambda/m*(theta(2:end)'*theta(2:end));
22 n = size(X,2);
23 grad(1) = 1/m*dot(h-y,X(:,1));
24 for i = 2:n
25     grad(i) = 1/m*dot(h-y,X(:,i))+lambda /m * theta(i);
26
```

On appelle la fonction de costFunctionReg en utilisant la valeur initiale de θ, on peut voir que le coût est d'environ 0.693147.

```
命令窗口
Cost at initial theta (zeros): 0.693147

Program paused. Press enter to continue.
```

2.4 Plotting the decision boundary

On peut obtenir suivant et on peut prédire en fonction des tests si la puce électronique sera acceptée ou non acceptée :