**（1）决策边界：**不是训练数据集的属性，而是假设本身以及其参数的属性。只需要给定了参数theta，就可以的到相应的决策边界。（即：用训练集来拟合参数theta）。

**得到决策边界的流程：**

1）使用的激活函数为

2）函数模型：

3）假设预测的条件为： ，

即：求得theta参数后，将测试数据入，得到的概率与0.5作比较。判断y=0或者y=1；（实现分类）

4）绘制决策边界：

①二维特征变量：

1）模型为，（即：Logistic线性回归模型）；

2）决策边界函数：，纵坐标、横坐标。

②多维特征变量：

1）模型为，（即：Logistic非线性回归模型）；

2）决策边界函数： ；

直接带入不同的x1、x2…..计算对应的数值，并存放在数组z(i,j)中。

在matlab中使用等高线函数绘制相应的图像（即：决策边界）

matlab代码：

contour(u, v, z, [0, 0], 'LineWidth', 2)

**（2）Logistic 回归：**

**思想：（二元分类）**

1. 代价函数→计算出梯度→梯度下降法→theta
2. 通过求解参数，得到线性模型 。
3. 利用sigmoid函数，得到逻辑函数，将模型的输出值控制在： ，得到的我们将其理解为概率。
4. 模型→决策边界→带入预测数据→判断线性模型>或<0，以此来对数据进行分类。

（测试数据的时候，需要在测试数据矩阵的最左边添加一列全1元素）

**思想：（多元分类：one-vs-all）**

1、代价函数（与二元分类一样，需要正则化）→计算出梯度→梯度下降法→theta（n\*（m+1）矩阵；n：输出矩阵y的行数；m：特征变量数量）；

2、theta矩阵：对应着n个分类器；m+1：+1是增加偏置变量

3、预测数据 + sigmoid函数 = 得到该数据属于第i类的概率，使用max函数得到n个分类器中概率最大的一个，即：判断预测数据属于哪一类。

示例代码：（预测数据的分类）

% 输出矩阵：sigmoid( X\*transpose(all\_theta))，有n列

% X：测试数据

% all\_theta：n\*（m+1）维矩阵

% max（）：得到这个矩阵每行最大的数

% b：每行最大的数据；p：每行最大数据所在的位置

[b,p] = max( sigmoid( X\*transpose(all\_theta)),[],2 );

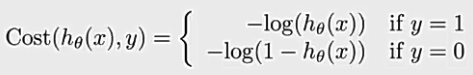
1. 代价函数：（**最终目标：得到，代价函数为凹函数，即求解最小值** ）



其中，****为sigmoid函数（S型函数）。



令， （该模型并不适合）



**（取该模型合适，注意，此处的log是以e为底——手写中习惯写成ln）**

将上述的两个式子整理成一个式子，得到，

 ，带入下式

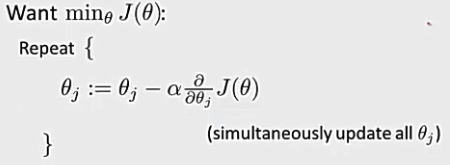


得，最终的代价函数为：



1. 求解参数theta：

**（1）使用梯度下降法：**与线性回归使用的方法一致。



程序中具体实现为：

**法一**： ，注意此处的

**法二：** 

**（2）高级优化：**使用fminunc( )函数求解。（此法较为简单，计算速度快）

（详见：(4)正则化部分2）theta求解）

**（3）过拟合：**

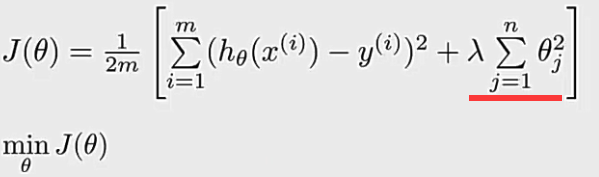
1. 当特征数量过多，且训练的样本数量较少时就会出现过拟合现象。
2. 解决方案：法一：人工去除无关的特征变量

法二：**正则化：**保留所有的特征变量，但减小其量级（或者theta的大小）

**（4）正则化：（不适合在特征数量少的模型中使用）**

1. 一般模型的代价函数：

**（注意：**正则化只涉及 ，没有涉及**，即：无需正则化偏差项）**



Logistic回归**正则化**的代价函数为：



其中，1）：正则化项；**注意：不计算**：也就是输入层的偏置单元。

2） ：为正则化参数（也称惩罚系数）。

一般令，取值一般从0.01开始，并以3倍速度增加，直至取到合适值为止。不宜过大（欠拟合）、过小（过拟合）。

在 中增加 ，为使 最小化，必然要减小 的值，这便使得在不减少特征变量数量的同时，通过减小来抑制过拟合。

附：代价函数+梯度函数的求解代码：

**theta\_1** = [0;theta(2:end)]; % 先令theta\_0 = 0，不参与正则化

**J** = -1 \* sum( y .\* log( sigmoid(X\*theta) ) + (1 - y ) .\* log( (1 - sigmoid(X\*theta)) ) ) / m + lambda/(2\*m) \* theta\_1' \* theta\_1 ;

**grad** = ( X' \* (sigmoid(X\*theta) - y ) )/ m + lambda/m \* theta\_1 ;

1. 求解theta的方法：

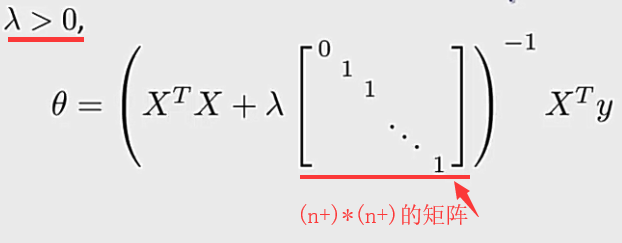
**（1）使用梯度下降法：（注意，j=1时，梯度函数不同）**







由于（事实上，约为0.9几），该式将 的值通过每一次的更新不断的减小，以达到缩小theta的目的。

**（2）使用正规方程法：**

**（3）使用matlab库函数fminunc：（适用于梯度下降求解方法）**

**具体代码如下：**

% 初始化theta参数为0，

initial\_theta = zeros(size(X, 2), 1);

% Set regularization parameter lambda to 1 (you should vary this)即：取值

lambda = 1;

% Set Options，'GradObj'：梯度下降的工程 ；'on'：打开 ；'MaxIter'：迭代次数400

options = optimset('GradObj', 'on', 'MaxIter', 400);

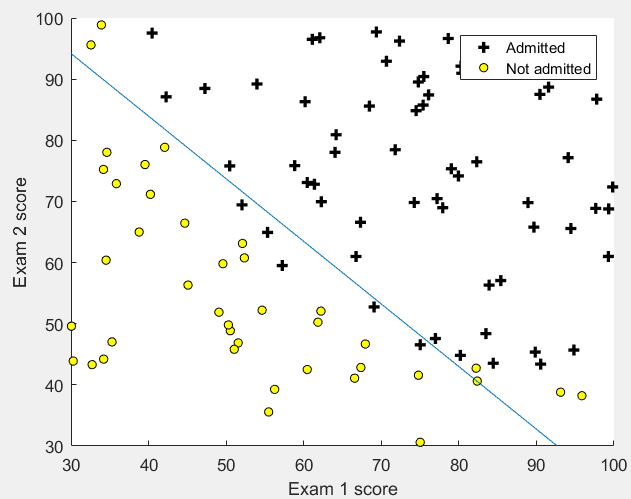
% Optimize，costFunctionReg：自己编写的代价函数（返回值：代价函数数值 + 梯度 ）@(t)：用于调用costFunctionReg

[theta, J, exit\_flag] = ...

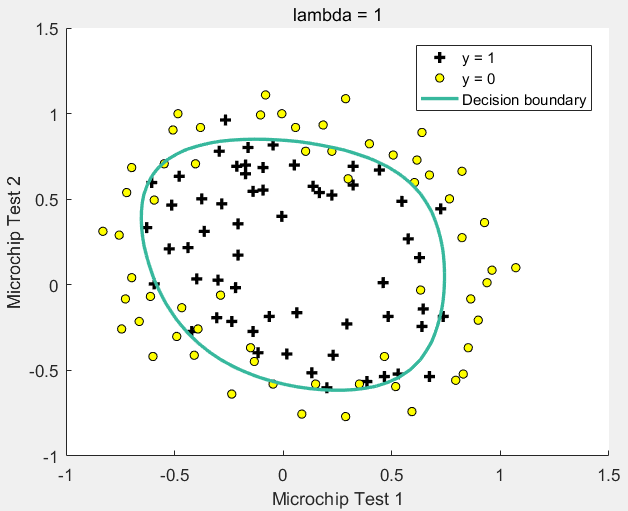
fminunc(@(t)(costFunctionReg(t, X, y, lambda)), initial\_theta, options);

附录：logistic回归的应用图形：（下列均属于二元分类）

1. Logistic线性回归**（适用于简单的分类模型）**



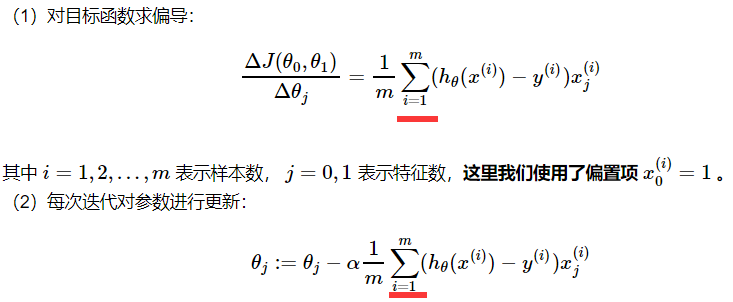
1. 正则化logistic线性回归**（适用于复杂的分类模型，需增加特征变量数目）**

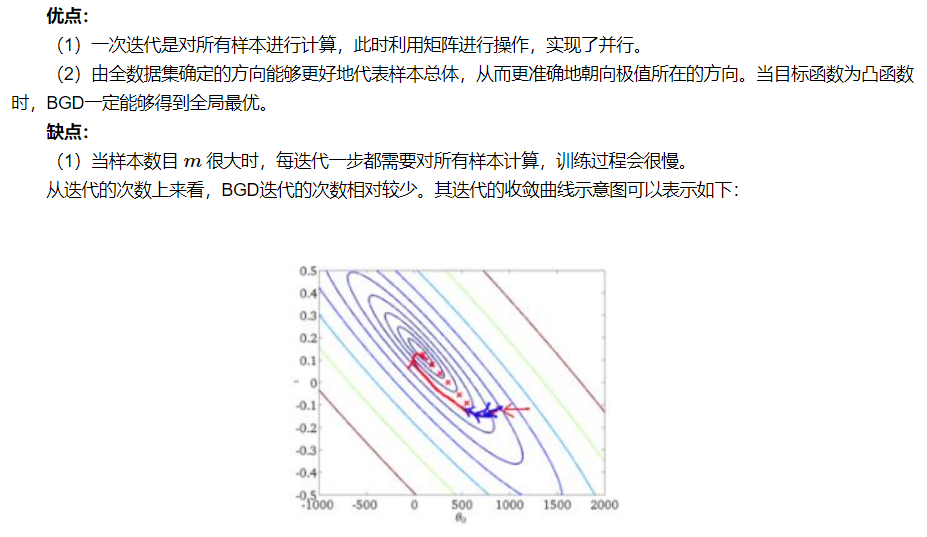


附：梯度下降分类：

1. 批量梯度下降：BGD，batch gradient descent

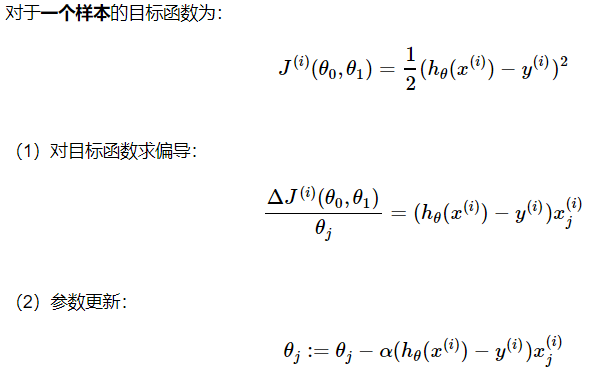
每次迭代，都需要使用所有的数据进行参数更新

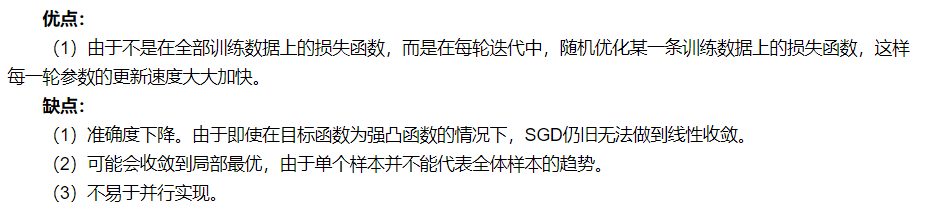


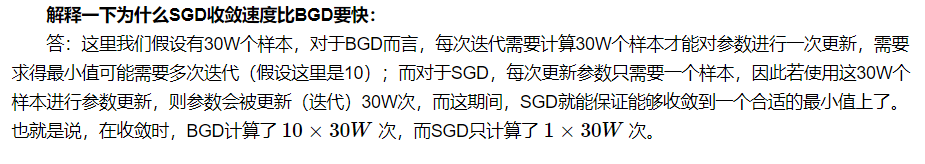


1. 随机梯度下降：SGD，stochastic gradient descent

每次使用一个数据进行梯度参数更新

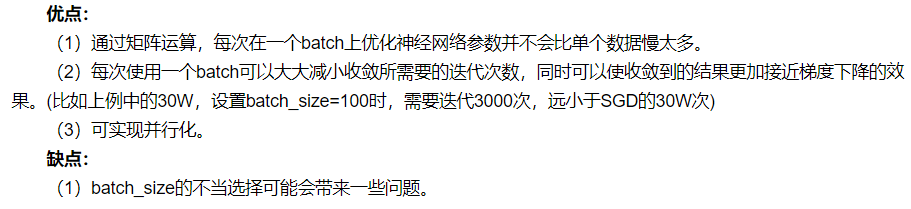






1. 小批量梯度下降：MBGD，Mini-batch gradient descent

每次迭代使用一定数量的数据，进行参数更新



三种梯度下降的收敛过程

