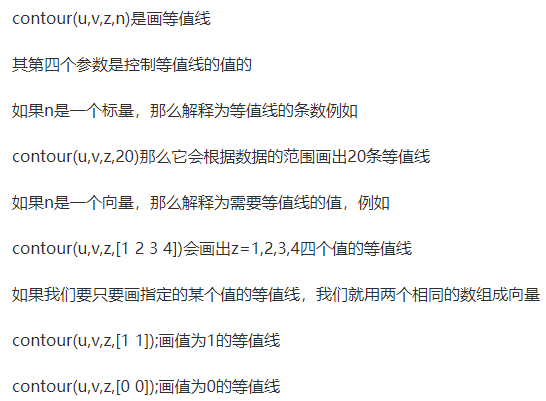
=====ex2.m

exp(x); 表示自然指数

log(x); 表示以e为底的对数，即自然对数

contour(u, v, z, [0, 0], 'LineWidth', 2); 绘制等高线函数，类似于上一次作业的surf函数的参数，x对应z的列索引，y对应z的行索引，所以如果x是行信息y是列信息，那z就得提前转置；[0,0]是指函数值z在0和0之间的等高线，只有一条



在plotDecisionBoundary.m文件中：

if size(X, 2) <= 3

% 这里是当点数小于等于2的时候直接利用直线公式进行计算，因为至少一个正/负样本

plot\_x = [min(X(:,2))-2, max(X(:,2))+2];

%取两端的两个点

plot\_y = (-1./theta(3)).\*(theta(2).\*plot\_x + theta(1));

% 这个公式对于每一个y，实际上是-(1/theta3)(theta2\*x+theta1)=y,变形就是theta1+theta2\*x+theta3\*y=0,所有就是一个直线公式

plot(plot\_x, plot\_y)

legend('Admitted', 'Not admitted', 'Decision Boundary')

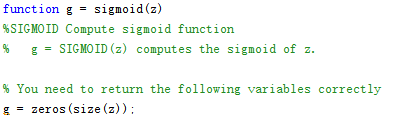
axis([30, 100, 30, 100]) %设置横纵坐标都是30到100；

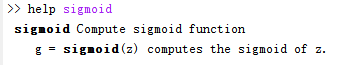
options = optimset('GradObj', 'on', 'MaxIter', 400); optimset(‘param1’,value1,’param2’,value2); 创建一个优化结构体，指定的参数用指定值，其他参数用默认值，后期用于传递给优化函数；GradObj设为on，告诉下面要调用的fminunc既要返回gradient又要返回cost，fminunc在优化的时候就可以利用梯度来进行优化，这里还设置最大迭代次数为400；

[theta, cost] = ...

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

这里，里面@(t)(costFunction(t,X,y)是定义了一个匿名函数，实际上@(t)(...)就是定义了一个没有名字的函数，说明这里的参数（就是自变量)只是t，可以参照[这里](https://stackoverflow.com/questions/43961765/matlab-meaning-of-tcostfunctiont-x-y-from-andrew-ngs-machine-learning)的解释，而这里这个匿名函数调用了costFunction函数；最外层fminunc的参数是（函数，参数初始值，其他参数的初始值）,可以参照MATLAB里的参考页，得到使函数最小的参数t的值，这里的返回形式是返回在t最小的情况下的t值，和依次得到的函数的返回值，costFunction函数返回数量为2大于1，后面的就不返回显示了；使用了fminunc就不需要自己去写代码获得最优值，学习率也不用自己设置，我们只需要给他提供一个计算。





函数名下面紧接着贴着的就是对函数名的注释，help命令会抽取这一部分进行解释输出。

====ex2\_reg.m

out(:, end+1) = (X1.^(i-j)).\*(X2.^j); end+1就可以动态添加新的维数数据；

**%**上一部分做的是线性分割theta1+theta2\*x1+theta3\*x2=0，变量指数都是1，这里就开始指数从1到6进行变化，也就是将原来的一次多项式特征映射到高次多项式特征中，以获得更多的特征，这里的特征就是x1和x2的不同指数的组合，原来单一的特征已经不能得出很好拟合数据的分割线，必须拿到更高维度的特征才能更好的拟合数据。

**%**

sprintf('lambda = %g', lambda)；格式化字符串

