Octave/Matlab Tutorial

1 Basic Operations

xor(x,y):x和y中有且仅有一个值为真，则返回真。

```
PS1('>> ');
```
取消前面的提示

如果你想分配一个变量，但是不希望在屏幕上显示结果，你可以在命令后加一个分号。

```
>>a=pi;
>>disp(sprintf('2 decimals:%0.2f',a))
#sprintf命令生成一个字符串，'%0.2f'意味着将a放在这里并显示a值的小数点后两位数字。
```

format long:默认情况下是字符串。显示出的小数点位有点多。

format short:是默认的输出格式，只是打印小数位数的第一位。

A=[1 2;3 4;5 6]会产生一个三行两列的矩阵A，；的作用就是换行到下一行。

```
v=1:0.1:2
\\可以得到一个行向量，从1开始，增量为0.1，增加到2
```

```
>>ones(2,3)
#也可以用来生成矩阵，结果为一个两行三列的矩阵，不过矩阵中的所有元素都为1
```

ans = 1 1 1 1 1 1

```
>>c=2*ones(2,3)
#当我想生成一个元素为2的两行三列的矩阵
>>w=randn(1,3)
#则生成一行三列的元素均为随机值的行向量
>>hist(w)
#绘制直方图的命令
>>eye(4)
#生成一个4行4列单位阵
```

2 Moving Data Around

```
>>size(A)
#打印出矩阵A的行列
>>size(A,1)
#打印出矩阵A的行数
>>size(A,2)
#打印出矩阵A的列数
>>length(A)
#打印出A最大维度的大小
>>pwd
#得到octave的安装位置
>>cd ''
#把路径改为
>>load featuresX.dat
#我将加载了featuresX文件
>>who
#显示出在当前空间的所有变量
>>featuresX
#显示出所有数据
>>size(featuresX)
#代表是一个几行几列的矩阵
>>whos
#列出了变量名字还列出了变量的维度，还显示出需要占用多少内存空间以及数据类型是什么。
>>clear featuresX
#清除featuresX
>>v=priceY(1:10)
#表示将向量Y的前十个元素存入v中
>>save hello.mat v;
#会将变量v存成一个叫hello.mat的文件
>>save hello.txt v -ascii
#存成一个文本文档或者将数据的ascii码存成文本文档
>>A(3,2)
```

```
#打印出A的三行两列的元素
>>A(2,:)
#打印出第二行的所有元素，：表示该行或该列的所有元素
>>A(:,2)
#打印出A矩阵第二列的所有元素
>>A([1 3],:)
#取A矩阵第一个索引值为1或3的元素
>>A(:,2)
#返回第二列
>>A(:,2)=[10;11;12]
#取A矩阵第二列并给辅助10,11,12
>>A=[A,[100; 101; 102]];
#在A矩阵的右边附加了一个新的列矩阵
>>A(:)
#把A中的所有元素放入一个单独的列向量
>>C=[A; B]
#把分号后面的东西放在下面
\section{Computing on Data}
\paragraph{}
\begin{minted}{octave}
>>A*C
>>A.*B
#对应位元素相乘
>>A.^2
#对矩阵A中的每一个元素平方
>>1./V
#得到V中每一个元素的倒数
>>log(V)
#对每一个元素进行求对数运算
>>exp(V)
#以e为底，以V中元素为幂的运算。
>>abs(V)
#对V中的每一个元素求绝对值
>>V+ones(length(V).1)
#将V的各元素都加上这些1.
>>V+1
#把V中的每一个元素都加上1
>>B=A'
#求A的转置
>>val=max(a)
#返回a中的最大值
>>[val,ind]=max(a)
#将返回a矩阵中的最大值存入val，以及对该值对应的索引，对应的索引存入ind
>>B=max(A)
#A是一个矩阵的话，这样做就是对每一列求最大值
>>a=[1 15 2 0.5]
```

```
>>a<3
#将进行逐元素的运算，所以元素小于3的返回1，否则返回0
>>find(a<3)
#将告诉我a中的哪些元素是小于3的
>>A=majic(3)
#majic将返回一个矩阵，成为魔方阵或幻方，它们所有的行和列和对角线加起来都等于相同的值。
>>[r,c]=find(A>=7)
#将找出所有A矩阵中大于等于7的元素
```

filpup/filpud表示向上/向下翻转。

```
>>pinv(A)
#求A矩阵的逆矩阵
```

3 Plotting Data

```
>>t=[0:0.01:0.98];
>>t
>>y1=sin(2*pi*4*t);
#输入plot（t,y1),并回车，就可以绘制正弦图
```

如果我想要同时表现正弦和余弦曲线，我要做的就是：

```
>>plot(t,y1)
>>hold on
#功能是将新的图像绘制在旧的之上
>>plot(t,y2,'r')
#我要以不同的颜色绘制余弦函数，所以我在这里输入带引号的r绘制余弦函数，r表示所使用的颜色
>>xlable('time')
#来标记X轴即水平轴
>>ylable('value')
#来标记Y轴即垂直轴
>>legend('sin','cos')
#表示这两条曲线表示的内容
>>title('my plot')
#在图像的顶部显示这幅图的标题
>>print-dpng'myplot.png'
#png是一个图像文件格式，可以让你保存该图像
#octave也可以保存为很多其他格式，你可以键入help plot
#如果你想删掉这个图像，用close命令会让这个图像关掉
```

```
>>figure(1);plot(t,y1)
#将显示第一张图，绘制了变量ty1.
>>subplot(1,2,1)
#他将图像分为1*2的格子，也就是前两个参数，然后他使用第一个格子，也就是最后一个参数1的意思。
>>axis([0,5 1 -1 1])
#也就是设置了右边图的 x 轴和 y 轴的范围。具体而言,它将右图中的横轴的范围调整至 0.5
到 1,竖轴的范围为-1 到 1。
>>imagesc(A)
#它将绘制一个5*5的矩阵，一个5*5的彩色格图，不同的颜色对应A矩阵中的不同值。
>>imagesc(A),colorbar,colormap gray
#生成一个颜色图像，一个灰色分布图，并在右边也加入一个颜色条。
#这个颜色条显示不同深浅的颜色所对应的值。
>>imagesc(magic(15)),colorbar,colormap gray
#这将是一幅15*15的magic方阵值的图。
>>a=1,b=2,c=3
#它将输出所有这三个结果,即使用逗号连接函数调用
>>a=1; b=2;c=3;
#没有输出出任何东西
```

4 Control Statements:for,while,if statement

4.1 for循环

首先,我要将 v 值设为一个 10 行 1 列的零向量。

接着我要写一个 “for” 循环,让 i 等于 1 到 10,写出来就是 i = 1:10。我要设 v(i) 的值等于 2 的 i 次方,循环最后写上“end”。

向量 v 的值就是这样一个集合 2 的一次方、2 的二次方,依此类推。这就是我的 i 等于 1 到 10 的语句结构,让 i 遍历 1 到 10 的值

```
>>v=zeros(10,1)
>>for i=1:10,
>>v(i)=2^i;
>>end;
>>v
```

另外,你还可以通过设置你的 indices (索引) 等于 1 一直到 10,来做到这一点。这时indices 就是一个从 1 到 10 的序列。

你也可以写 i = indices,这实际上和我直接把 i 写到 1 到 10 是一样。你可以写 disp(i),也能得到一样的结果。所以 这就是一个 “for” 循环。

```
>>indices=1:10;
>>for i=indices,
>>disp(i);
>>end;
```

如果你对 “break” 和 “continue” 语句比较熟悉, Octave 里也有 “break” 和 “continue”语句,你也可以在 Octave 环境里使用那些循环语句。

4.2 while循环

```
>>i=1;
>>while i<=5,
>v(i)=100;
>i=i+1;
>end;
#这是什么意思呢:我让 i 取值从 1 开始,然后我要让 v(i) 等于 100,再让 i 递增 1,
#直到 i 大于 5 停止。
#现在来看一下结果,我现在已经取出了向量的前五个元素,把他们用 100 覆盖掉,
#这就是一个 while 循环的句法结构。
```

分析另外一个例子:

```
>>i=1;
>>while true,
>v(i)=999;
>i=i+1;
>if i==6,
>break;
>end;
>end;
>>
#这里我将向你展示如何使用 break 语句。比方说 v(i) = 999,然后让 i = i+1,
#当 i 等于 6的时候 break (停止循环),结束 (end)。
#当然这也是我们第一次使用一个 if 语句,所以我希望你们可以理解这个逻辑,
#让 i 等于 1 然后开始下面的增量循环,while 语句重复设置 v(i) 等于 999,不断让 i 增加,
#然后当i 达到 6,做一个中止循环的命令,尽管有 while 循环,语句也就此中止。
#所以最后的结果是取出向量 v 的前 5 个元素,并且把它们设置为 999。
#所以,这就是 if 语句和 while 语句的句法结构。并且要注意要有 end,
#上面的例子里第一个 end 结束的是 if 语句,第二个 end 结束的是 while 语句。
```

4.3 if-else语句

```
>>v(1)=2;
>>if v(1)==1.
>disp('The value is one');
>elseif v(1)==2.
>disp('The value is two');
>else
>disp('The value is not one or two.');
>end;
```

4.4 functions

我在桌面上存了一个预先定义的文件名为“squarethisnumber.m”,这就是在Octave 环境下定义的函数。

文件里:

$function \quad y = squareThisNumber(x)$

$y = x^2$

```
>>squarethisnumber(5)
ans=25
```

你可以使用 addpath 命令添加路径,添加路径将该目录添加到Octave搜索目录，这样即使跑到其他路径下，Octave依然会在函数所在目录下搜索函数。

Octave 的语法结构不一样,可以返回多个值.

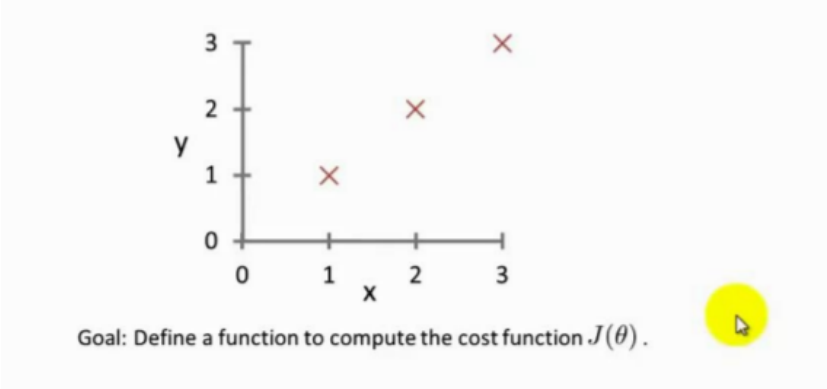
比如

```
#先定义函数\SquareAndCubeThisNumber(x)"
>>[a,b] = SquareAndCubeThisNumber(5)
#a就等于25，b就等于125
```

更复杂的例子

比方说,我有一个数据集,像这样,数据点为[1,1], [2,2], [3,3],我想做的事是定义一个 Octave 函数来计算代价函数 J(θ),就是计算不同 θ 值所对应的代价函数值 J。

首先让我们把数据放到 Octave 里,我把我的矩阵设置为 X = [1 1; 1 2; 1 3];



- 函数的定义

```
function J = costFunctionJ(X, y, theta)

% X is the "design matrix" containing our training examples.
% y is the class labels

m = size(X,1);          % number of training examples
predictions = X*theta;  % predictions of hypothesis on all m
examples
sqrErrors = (predictions-y).^2;  % squared errors


J = 1/(2*m) * sum(sqrErrors);
```

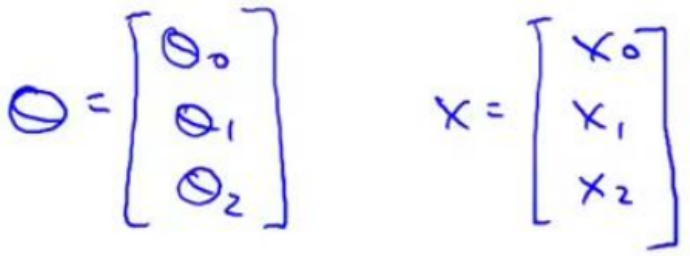
现在当我在 Octave 里运行时,我键入 j = costFunctionJ(X, y, theta),它就计算出 j 等于0,这是因为如果我的数据集 x 为 [1;2;3], y 也为 [1;2;3] 然后设置 $\theta 0$ 等于 $0, \theta 1$ 等于1,这给了我恰好 45 度的斜线,这条线是可以完美拟合我的数据集的。 而相反地,如果我设置 theta 等于[0; 0],那么这个假设就是 0 是所有的预测值,和刚才 一样,设置 $\theta 0 = 0, \theta 1$ 也等于 0,然后我计算的代价函数,结果是 2.333。实际上,他就等于 1 的平方,也就是第一个样本的平方误差,加上 2 的平方,加上 3 的平方,然后除以 2m,也就是训练样本数的两倍,这就是 2.33

4.5 Vectorization

这是一个常见的线性回归假设函数:

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

如果你想要计算$h_\theta(x)$,注意到右边是求和,那么你可以自己计算 j =0 到 j = n 的和。但换另一种方式来想想,把$h_\theta(x)$看作$\theta^T x$,那么你就可以写成两个向量的内积,其中 θ 就是$\theta_0\theta_1\theta_2$,如果你有两个特征量,如果 n =2,并且如果你把 x 看作$x_0x_1x_2$,这两种思考角度,会给你两种不同的实现方式。

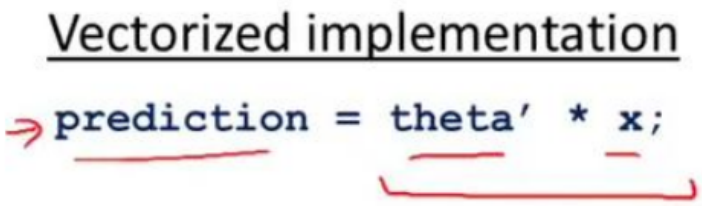


比如说，这是未向量化的代码实现方式:

```
prediction = 0.0;
for j = 1:n+1,
  prediction = prediction +
               theta(j) * x(j)
end;
```

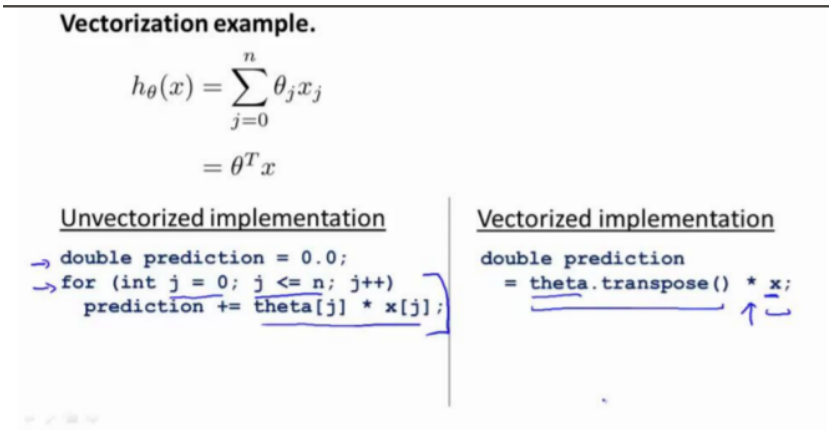
计算$h_\theta(x)$是未向量化的,我们可能首先要初始化变量 prediction 的值为 0.0,而这个变量 prediction 的最终结果就是$h_\theta(x)$,然后我要用一个 for 循环,j 取值 0 到 n+1,变量prediction 每次就通过自身加上 theta(j) 乘以 x(j) 更新值,这个就是算法的代码实现。

作为比较,接下来是向量化的代码实现:



你把 x 和 θ 看做向量,而你只需要令变量 prediction 等于 theta 转置乘以 x,你就可以这样计算。与其写所有这些 for 循环的代码,你只需要一行代码,这行代码就是利用Octave 的高度优化的数值,线性代数算法来计算两个向量 θ 以及 x 的内积,这样向量化的实现更简单,它运行起来也将更加高效。

这就是 Octave 所做的而向量化的方法,在其他编程语言中同样可以实现。让我们来看一个 C++ 的例子:



与此相反,使用较好的 C++ 数值线性代数库,你可以写出像右边这样的代码,因此取决于你的数值线性代数库的内容。你只需要在 C++ 中将两个向量相乘,根据你所使用的数值和线性代数库的使用细节的不同,你最终使用的代码表达方式可能会有些许不同,但是通过一个库来做内积,你可以得到一段更简单、更有效的代码。

现在,让我们来看一个更为复杂的例子,这是线性回归算法梯度下降的更新规则:

$$\begin{cases} \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \\ \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \\ \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \end{cases}$$
$$(n = 2)$$

我们用这条规则对 j 等于 0、1、2 等等的所有值,更新对象θ_j,我只是用θ_0、θ_1、θ_2来写方程,假设我们有两个特征量,所以 n 等于 2,这些都是我们需要对 θ_0、θ_1、θ_2 进行更新,这些都应该是同步更新,我们用一个向量化的代码实现,这里是和之前相同的三个方程,只不过写得小一点而已。

你可以想象实现这三个方程的方式之一,就是用一个 for 循环,就是让 j 等于 0、等于1、等于 2,来更新θ_j。

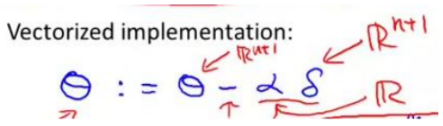
但让我们用向量化的方式来实现,看看我们是否能够有一个更简单的方法。基本上用三行代码或者一个 for 循环,一次实现这三个方程。

让我们来看看怎样能用这三步,并将它们压缩成一行向量化的代码来实现。做法如下:

我打算把 θ 看做一个向量,然后我用 θ - α 乘以某个别的向量 δ 来更新 θ 。

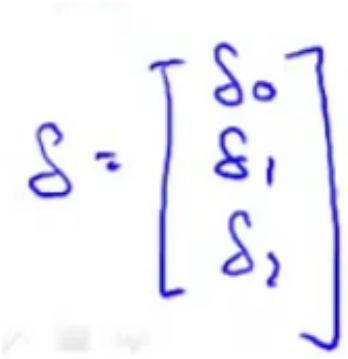
这里的 δ 等于 $\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$

让我解释一下是怎么回事:我要把 θ 看作一个向量,有一个 n+1 维向量,α 是一个实数,δ 在这里是一个向量。

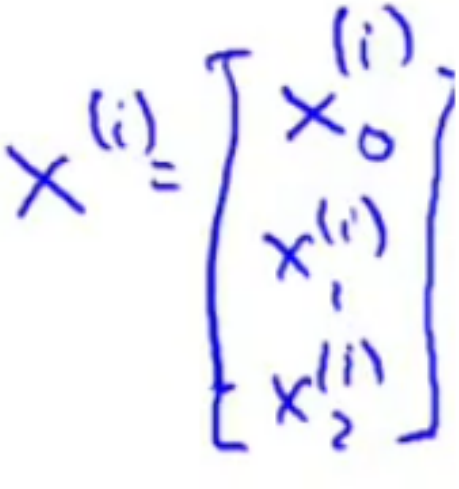


所以这个减法运算是一个向量减法,因为 α 乘以 δ 是一个向量,所以 θ 就是 θ - α^δ得到的向量。

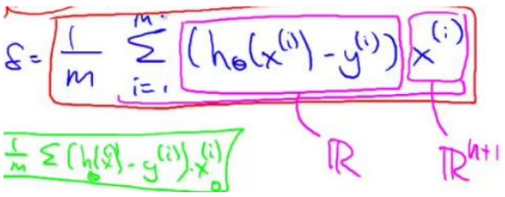
那么什么是向量 δ 呢？



$x^{(i)}$是一个向量



你就会得到这些不同的式子，然后作加和。



实际上,在以前的一个小测验,如果你要解这个方程,我们说过为了向量化这段代码,我们会令 u = 2v +5w 因此,我们说向量 u 等于 2 乘以向量 v 加上 5 乘以向量 w。用这个例子说明,如何对不同的向量进行相加,这里的求和是同样的道理。

$$u(j) = 2v(j) + 5w(j) \quad \text{(for all } j\text{)}$$
$$u = 2v + 5w$$