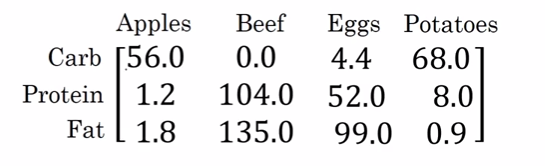
**2019.1.20总结**

**Python广播**

例题，Calories from carb,protein,fat in 100g of different foods:



代码：

**import numpy as np**

**A=np.array( [[56.0,0.0,4.4,68.0],**

**[1.2,104.0,52.0,8.0],**

**[1.8,135.0,99.0,0.9]])**#array()函数加入数值后生成了矩阵

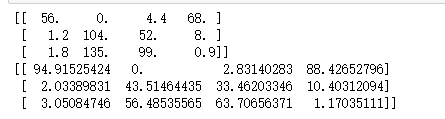
**print(A)**

**cal=A.sum(axis=0)**#求和，axis=0表示竖直相加，axis=1表示水平相加

**percentage = 100\*A/(cal.reshape(1,4))**”””reshape()函数将cal变成1x4的矩阵（其实这里可以不用，因为cal已经是了，但是为了养成习惯，便于改变，加上为好）reshape()函数是数组对象中的用法，用于改变数组的形状，确保矩阵或向量是需要的维度。”””

**print(percentage)**

Generated result(生成的结果)：



 在python中计算矩阵与常数的加法，python会自动把常数复制补全成对应的矩阵

 同理复制补全

复制m次

mxn

fu

**m**xn

1xn

 同理复制补全

mx**n**

复制m次

Mx1

mxn

**（+，-，\*，/）都同理python可复制补全成对应矩阵**

**Python/numpy vectors(numpy 向量)：**

生成矩阵代码：

**import numpy as np**

**a=np.random.randn(5)**

**print(a)**

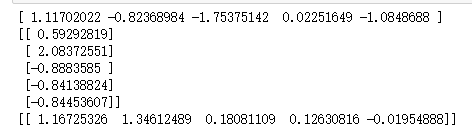
**a=np.random.randn(5,1)**

**print(a)**

**a=np.random.randn(1,5)**

**print(a)**

a=np.random.randn(5)这条语句生成的a的维度是（5，）。它既不是行向量也不是列向量，我们把a叫做rank 1 array。这种定义会带来一些问题。例如我们对a进行转置，还是会得到a本身。所以，如果我们要定义（5，1）的列向量或者（1，5）的行向量，最好使用下来标准语句，a=np.random.randn(5,1)；a=np.random.randn(1,5) 避免使用rank 1 array。



**assert(a.shape ==(5,1))**# 使用assert语句对向量或数组的维度进行判断, assert会对内嵌语句进行判断，即判断a的维度是不是（5，1）的。如果不是，则程序在此处停止。使用assert语句也是一种很好的习惯，能够帮助我们及时检查、发现语句是否正确。

**a.reshape((5,1))**# 使用reshape函数对数组设定所需的维度

logisitic损失函数解释:

= when 

Interpret 

If y=1  If y=0 

由上面可得：

由于log函数的单调性，可以对上式P(y|x)进行log处理：

我们希望上述概率P(y|x)越大越好，对上式加上负号，则转化成了单个样本的Loss function，越小越好，也就得到了我们之前介绍的逻辑回归的Loss function形式。

如果对于所有m个训练样本，假设样本之间是独立同分布的（iid），我们希望总的概率越大越好：

p（labels in training set）= 同样引入log函数，加上负号，将上式转化为Cost function：

log p(labels in training set)=log



把上式中的“-”去掉，因为我们求最小化成本，不需要直接用最大似然估计，为了方便，对成本函数加上1/m进行适当缩放

Cost function:

Maimun likelihood estamation(最大似然估计)即求出一组参数，使这个式子取最大值



= 

**第三周：Neural Network**

[i]表示层数 (i)表示样本个数

Neural Network Representation(表示法)

**一个样本**

