NumPy 中的矩阵和向量

numpy的 `ndarray` 类用于表示矩阵和向量。 要在numpy中构造矩阵，我们在列表中列出
的行， 并将该列表传递给numpy数组构造函数。

例如，构造与矩阵对应的numpy数组

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 0 \end{bmatrix}$$

我们会这样做

```
A = np.array([[1,-1,2],[3,2,0]])
```

向量只是具有单列的数组。 例如，构建向量

$$\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

我们会这样做

```
v = np.array([[2],[1],[3]])
```

更方便的方法是转置相应的行向量。 例如，为了使上面的矢量，我们可以改为转置行向

$$\begin{bmatrix} 2 & 1 & 3 \end{bmatrix}$$

这个代码是

```
v = np.transpose(np.array([[2,1,3]]))
```

numpy重载数组索引和切片符号以访问矩阵的各个部分。 例如，要打印矩阵A中的右下
目，我们会这样做

要切出A矩阵中的第二列，我们会这样做

第一个切片选择A中的所有行，而第二个切片仅选择每行中的中间条目。

要进行矩阵乘法或矩阵向量乘法，我们使用np.dot()方法。

```
w = np.dot(A,v)
```

用numpy求解方程组

线性代数中比较常见的问题之一是求解矩阵向量方程。 这是一个例子。 我们寻找解决方
向量x

$A\ \mathbf{x} = \mathbf{b}$

当

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 3 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -3 \\ 5 \\ -2 \end{bmatrix}$$

我们首先构建A和b的数组。

```
A = np.array([[2,1,-2],[3,0,1],[1,1,-1]])
b = np.transpose(np.array([[-3,5,-2]]))
```

为了解决这个系统

```
x = np.linalg.solve(A,b)
```

应用：多元线性回归

在多元回归问题中，我们寻找一个能够将输入数据点映射到结果值的函数。 每个数据点*征向量（x1，x2，…，xm）*，由两个或多个捕获输入的各种特征的数据值组成。为了表有输入数据以及输出值的向量，我们设置了输入矩阵X和输出向量 y：

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

在简单的最小二乘线性回归模型中，我们寻找向量**β**，使得乘积Xβ最接近结果向量 y。

一旦我们构建了**β**向量，我们就可以使用它将输入数据映射到预测结果。 给定表单中的输向量

$$\mathbf{x} = \begin{bmatrix} 1 & x_1 & x_2 & \cdots & x_m \end{bmatrix}$$

我们可以计算预测结果值

$$\hat{y} = \mathbf{x} \cdot \boldsymbol{\beta} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

计算β向量的公式是

$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$

在我们的下一个示例程序中，我将使用numpy构造适当的矩阵和向量并求解**β**向量。一旦解决了**β**，我们将使用它来预测我们最初从输入数据集中遗漏的一些测试数据点。

假设我们在numpy中构造了输入矩阵X和结果向量**y**，下面的代码将计算**β**向量：

```
Xty = np.dot(Xt,y)
beta = np.linalg.solve(XtX,Xty)
```

β，因为等式是：

$$\mathbf{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

在数学上等价于方程组：

$$(X^T X)\,\mathbf{\beta} = X^T \mathbf{y}$$

我将用于此示例的数据集是Windsor房价数据集，其中包含有关安大略省温莎市区房屋销信息。 输入变量涵盖了可能对房价产生影响的一系列因素，例如批量大小，卧室数量以种设施的存在。此处提供具有完整数据集的CSV文件。我从这个网站下载了数据集，站提供了大量涵盖大量主题的数据集。

这里现在是示例程序的源代码。

```
import csv
import numpy as np

def readData():
    X = []
    y = []
    with open('Housing.csv') as f:
        rdr = csv.reader(f)
        # Skip the header row
        next(rdr)
        # Read X and y
        for line in rdr:
            xline = [1.0]
            for s in line[:-1]:
                xline.append(float(s))
            X.append(xline)
            y.append(float(line[-1]))
    return (X,y)

X0,y0 = readData()
# Convert all but the last 10 rows of the raw data to numpy arrays
d = len(X0)-10
X = np.array(X0[:d])
y = np.transpose(np.array([y0[:d]]))

# Compute beta
Xt = np.transpose(X)
XtX = np.dot(Xt,X)
Xty = np.dot(Xt,y)
beta = np.linalg.solve(XtX,Xty)
```

```
for data,actual in zip(X0[d:],y0[d:]):
    x = np.array([data])
    prediction = np.dot(x,beta)
                                    ion[0,0])+' actual = '+str(actual))
```

原始数据集包含500多个条目 为了测试线性回归模型所做预测的准确性，我们使用除最后
个数据条目之外的所有数据条目来构建回归模型并计算**β**。一旦我们构建了**β**向量，我们
它来预测最后10个输入值，然后将预测的房价与数据集中的实际房价进行比较。

以下是该计划产生的产出：

```
[[ -4.14106096e+03]
 [  3.55197583e+00]
 [  1.66328263e+03]
 [  1.45465644e+04]
 [  6.77755381e+03]
 [  6.58750520e+03]
 [  4.44683380e+03]
 [  5.60834856e+03]
 [  1.27979572e+04]
 [  1.24091640e+04]
 [  4.19931185e+03]
 [  9.42215457e+03]]
prediction = 97360.6550969 actual = 82500.0
prediction = 71774.1659014 actual = 83000.0
prediction = 92359.0891976 actual = 84000.0
prediction = 77748.2742379 actual = 85000.0
prediction = 91015.5903066 actual = 85000.0
prediction = 97545.1179047 actual = 91500.0
prediction = 97360.6550969 actual = 94000.0
prediction = 106006.800756 actual = 103000.0
prediction = 92451.6931269 actual = 105000.0
prediction = 73458.2949381 actual = 105000.0
```

总体而言，预测并不是非常好，但是一些预测有点接近正确。从这些数据中做出更好的
将成为机器学习冬季学期教程的主题。

文章出处

由NumPy中文文档翻译，原作者为 劳伦斯大学，翻译至：
http://www2.lawrence.edu/fast/GREGGJ/Python/numpy/numpyLA.html。

NumPy 中文网

NumPy 数据分析

昵称 邮箱 网址(http://)

评论系统公测中，欢迎体验！

表情

1 评论

Anonymous Firefox 69.0 Windows 7
2019-10-16

d

Anonymous Firefox 69.0 Windows 7
2019-10-16

@Anonymous , sdasdasdas

Anonymous Firefox 69.0 Windows 7
2019-10-16

@Anonymous , asdasdasd