Numpy 使用教程

一、实验介绍

1.1 实验内容

如果你使用 Python 语言进行科学计算，那么一定会接触到 Numpy。Numpy 是支持 Python 语言的数值计算扩充库，其拥有强大的高维度数组处理与矩阵运算能力。除此之外，Numpy 还内建了大量的函数，方便你快速构建数学模型。

1.2 实验知识点

- Numpy 数学函数
- Numpy 代数运算

1.3 实验环境

- python2.7
- Xfce 终端
- ipython 终端

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有 Python 基础，并对使用 Numpy 进行科学计算感兴趣的用户。

二、数学函数

使用 python 自带的运算符，你可以完成数学中的加减乘除，以及取余、取整，幂次计算等。导入自带的 math 模块之后，里面又包含绝对值、阶乘、开平方等一些常用的数学函数。不过，这些函数仍然相对基础。如果要完成更加复杂一些的数学计算，就会显得捉襟见肘了。

numpy 为我们提供了更多的数学函数，以帮助我们更好地完成一些数值计算。下面就依次来看一看。

2.1 三角函数

首先, 看一看 numpy 提供的三角函数功能。这些方法有：

1. `numpy.sin(x)` ：三角正弦。
2. `numpy.cos(x)` ：三角余弦。
3. `numpy.tan(x)` ：三角正切。
4. `numpy.arcsin(x)` ：三角反正弦。
5. `numpy.arccos(x)` ：三角反余弦。
6. `numpy.arctan(x)` ：三角反正切。
7. `numpy.hypot(x1,x2)` ：直角三角形求斜边。
8. `numpy.degrees(x)` ：弧度转换为度。
9. `numpy.radians(x)` ：度转换为弧度。
10. `numpy.deg2rad(x)` ：度转换为弧度。
11. `numpy.rad2deg(x)` ：弧度转换为度。

比如，我们可以用上面提到的 `numpy.rad2deg(x)` 将弧度转换为度。

```
import numpy as np

np.rad2deg(np.pi)
```

```
In [2]: np.rad2deg(np.pi)
Out[2]: 180.0
```

这些函数非常简单，就不再一一举例了。

2.2 双曲函数

在数学中，双曲函数是一类与常见的三角函数类似的函数。双曲函数经常出现于某些重要的线性微分方程的解中，使用 numpy 计算它们的方法为：

1. `numpy.sinh(x)` ：双曲正弦。
2. `numpy.cosh(x)` ：双曲余弦。
3. `numpy.tanh(x)` ：双曲正切。
4. `numpy.arcsinh(x)` ：反双曲正弦。
5. `numpy.arccosh(x)` ：反双曲余弦。
6. `numpy.arctanh(x)` ：反双曲正切。

2.3 数值修约

数值修约, 又称数字修约, 是指在进行具体的数字运算前, 按照一定的规则确定一致的位数, 然后舍去某些数字后面多余的尾数的过程[via. 维基百科]。比如, 我们常听到的「4 舍 5 入」就属于数值修约中的一种。

1. `numpy.around(a)` ：平均到给定的小数位数。
2. `numpy.round_(a)` ：将数组舍入到给定的小数位数。
3. `numpy.rint(x)` ：修约到最接近的整数。
4. `numpy.fix(x, y)` ：向 0 舍入到最接近的整数。
5. `numpy.floor(x)` ：返回输入的底部(标量 x 的底部是最大的整数 i)。
6. `numpy.ceil(x)` ：返回输入的上限(标量 x 的底部是最小的整数 i).
7. `numpy.trunc(x)` ：返回输入的截断值。

随机选择几个浮点数，看一看上面方法的区别。

```
>>> import numpy as np

>>> a = np.array([1.21, 2.53, 3.86])
>>> a
array([ 1.21,  2.53,  3.86])

>>> np.around(a)
array([ 1.,  3.,  4.])

>>> np.round_(a)
array([ 1.,  3.,  4.])

>>> np.rint(a)
array([ 1.,  3.,  4.])

>>> np.fix(a)
array([ 1.,  2.,  3.])

>>> np.floor(a)
array([ 1.,  2.,  3.])

>>> np.ceil(a)
array([ 2.,  3.,  4.])

>>> np.trunc(a)
array([ 1.,  2.,  3.])
```

2.4 求和、求积、差分

下面这些方法用于数组内元素或数组间进行求和、求积以及进行差分。

1. `numpy.prod(a, axis, dtype, keepdims)`：返回指定轴上的数组元素的乘积。
2. `numpy.sum(a, axis, dtype, keepdims)`：返回指定轴上的数组元素的总和。
3. `numpy.nanprod(a, axis, dtype, keepdims)`：返回指定轴上的数组元素的乘积, 将 NaN 视作 1。
4. `numpy.nansum(a, axis, dtype, keepdims)`：返回指定轴上的数组元素的总和, 将 NaN 视作 0。
5. `numpy.cumprod(a, axis, dtype)`：返回沿给定轴的元素的累积乘积。

6. `numpy.cumsum(a, axis, dtype)`：返回沿给定轴的元素的累积总和。
7. `numpy.nancumprod(a, axis, dtype)`：返回沿给定轴的元素的累积乘积, 将 NaN 视作 1。
8. `numpy.nancumsum(a, axis, dtype)`：返回沿给定轴的元素的累积总和, 将 NaN 视作 0。
9. `numpy.diff(a, n, axis)`：计算沿指定轴的第 n 个离散差分。
10. `numpy.ediff1d(ary, to_end, to_begin)`：数组的连续元素之间的差异。
11. `numpy.gradient(f)`：返回 N 维数组的梯度。
12. `numpy.cross(a, b, axisa, axisb, axisc, axis)`：返回两个(数组）向量的叉积。
13. `numpy.trapz(y, x, dx, axis)`：使用复合梯形规则沿给定轴积分。

下面，我们选取几个举例测试一下：

```
>>> import numpy as np
>>> a=np.arange(5)
>>> a
array([0, 1, 2, 3, 4])

>>> np.prod(a) # 所有元素乘积
0

>>> np.sum(a) # 所有元素和
10

>>> np.nanprod(a) # 默认轴上所有元素乘积
0

>>> np.nansum(a) # 默认轴上所有元素和
10

>>> np.cumprod(a) # 默认轴上元素的累积乘积。
array([0, 0, 0, 0, 0])

>>> np.diff(a) # 默认轴上元素差分。
array([1, 1, 1, 1])
```

2.5 指数和对数

如果你需要进行指数或者对数求解，可以用到以下这些方法。

1. `numpy.exp(x)`：计算输入数组中所有元素的指数。
2. `numpy.expm1(x)`：对数组中的所有元素计算 exp(x）- 1.
3. `numpy.exp2(x)`：对于输入数组中的所有 p, 计算 2 ** p。
4. `numpy.log(x)`：计算自然对数。
5. `numpy.log10(x)`：计算常用对数。
6. `numpy.log2(x)`：计算二进制对数。
7. `numpy.log1p(x)`：`log(1 + x)`。
8. `numpy.logaddexp(x1, x2)`：`log2(2**x1 + 2**x2)`。
9. `numpy.logaddexp2(x1, x2)`：`log(exp(x1) + exp(x2))`。

2.6 算术运算

当然，numpy 也提供了一些用于算术运算的方法，使用起来会比 python 提供的运算符灵活一些，主要是可以直接针对数组。

1. `numpy.add(x1, x2)`：对应元素相加。
2. `numpy.reciprocal(x)`：求倒数 1/x。
3. `numpy.negative(x)`：求对应负数。
4. `numpy.multiply(x1, x2)`：求解乘法。
5. `numpy.divide(x1, x2)`：相除 x1/x2。
6. `numpy.power(x1, x2)`：类似于 x1^x2。
7. `numpy.subtract(x1, x2)`：减法。
8. `numpy.fmod(x1, x2)`：返回除法的元素余项。
9. `numpy.mod(x1, x2)`：返回余项。
10. `numpy.modf(x1)`：返回数组的小数和整数部分。
11. `numpy.remainder(x1, x2)`：返回除法余数。

```
>>> import numpy as np

>>> a1 = np.random.randint(0, 10, 5)
>>> a2 = np.random.randint(0, 10, 5)

>>> a1
array([3, 7, 8, 0, 0])

>>> a2
array([1, 8, 6, 4, 4])

>>> np.add(a1, a2)
array([ 4, 15, 14,  4,  4])

>>> np.reciprocal(a1)
array([0, 0, 0, , ])

>>> np.negative(a1)
array([-3, -7, -8,  0,  0])

>>> np.multiply(a1, a2)
array([ 3, 56, 48,  0,  0])

>>> np.divide(a1, a2)
array([3, 0, 1, 0, 0])

>>> np.power(a1, a2)
array([3,5764801,262144,0,0])

>>> np.subtract(a1, a2)
array([ 2, -1,  2, -4, -4])

>>> np.fmod(a1, a2)
array([0, 7, 2, 0, 0])

>>> np.mod(a1, a2)
array([0, 7, 2, 0, 0])

>>> np.modf(a1)
(array([ 0.,  0.,  0.,  0.,  0.]), array([ 3.,  7.,  8.,  0.,  0.]))

>>> np.remainder(a1, a2)
array([0, 7, 2, 0, 0])
>>>
```

2.7 矩阵和向量积

求解向量、矩阵、张量的点积等同样是 numpy 非常强大的地方。

1. `numpy.dot(a,b)` ：求解两个数组的点积。
2. `numpy.vdot(a,b)` ：求解两个向量的点积。
3. `numpy.inner(a,b)` ：求解两个数组的内积。
4. `numpy.outer(a,b)` ：求解两个向量的外积。
5. `numpy.matmul(a,b)` ：求解两个数组的矩阵乘积。
6. `numpy.tensordot(a,b)` ：求解张量点积。
7. `numpy.kron(a,b)` ：计算 Kronecker 乘积。

2.8 其他

除了上面这些归好类别的方法，numpy 中还有一些用于数学运算的方法，归纳如下：

1. `numpy.angle(z, deg)` ：返回复参数的角度。
2. `numpy.real(val)` ：返回数组元素的实部。
3. `numpy.imag(val)` ：返回数组元素的虚部。
4. `numpy.conj(x)` ：按元素方式返回共轭复数。
5. `numpy.convolve(a, v, mode)` ：返回线性卷积。
6. `numpy.sqrt(x)` ：平方根。
7. `numpy.cbrt(x)` ：立方根。
8. `numpy.square(x)` ：平方。
9. `numpy.absolute(x)` ：绝对值, 可求解复数。
10. `numpy.fabs(x)` ：绝对值。
11. `numpy.sign(x)` ：符号函数。
12. `numpy.maximum(x1, x2)` ：最大值。
13. `numpy.minimum(x1, x2)` ：最小值。
14. `numpy.nan_to_num(x)` ：用 0 替换 NaN。
15. `numpy.interp(x, xp, fp, left, right, period)` ：线性插值。

三、代数运算

上面，我们分为 8 个类别，介绍了 numpy 中常用到的数学函数。这些方法让复杂的计算过程表达更为简单。除此之外，numpy 中还包含一些代数运算的方法，尤其是涉及到矩阵的计算方法，求解特征值、特征向量、逆矩阵等，非常方便。

1. `numpy.linalg.cholesky(a)`：Cholesky 分解。
2. `numpy.linalg.qr(a ,mode)`：计算矩阵的 QR 因式分解。
3. `numpy.linalg.svd(a ,full_matrices,compute_uv)`：奇异值分解。
4. `numpy.linalg.eig(a)`：计算正方形数组的特征值和右特征向量。
5. `numpy.linalg.eigh(a, UPLO)`：返回 Hermitian 或对称矩阵的特征值和特征向量。
6. `numpy.linalg.eigvals(a)`：计算矩阵的特征值。
7. `numpy.linalg.eigvalsh(a, UPLO)`：计算 Hermitian 或真实对称矩阵的特征值。
8. `numpy.linalg.norm(x ,ord,axis,keepdims)`：计算矩阵或向量范数。
9. `numpy.linalg.cond(x ,p)`：计算矩阵的条件数。
10. `numpy.linalg.det(a)`：计算数组的行列式。
11. `numpy.linalg.matrix_rank(M ,tol)`：使用奇异值分解方法返回秩。
12. `numpy.linalg.slogdet(a)`：计算数组的行列式的符号和自然对数。
13. `numpy.trace(a ,offset,axis1,axis2,dtype,out)`：沿数组的对角线返回总和。
14. `numpy.linalg.solve(a,b)`：求解线性矩阵方程或线性标量方程组。
15. `numpy.linalg.tensorsolve(a,b ,axes)`：为 x 解出张量方程a x = b
16. `numpy.linalg.lstsq(a,b ,rcond)`：将最小二乘解返回到线性矩阵方程。
17. `numpy.linalg.inv(a)`：计算逆矩阵。
18. `numpy.linalg.pinv(a ,rcond)`：计算矩阵的（Moore-Penrose）伪逆。
19. `numpy.linalg.tensorinv(a ,ind)`：计算N维数组的逆。

四、实验总结

数学函数和代数运算方法是使用 numpy 进行数值计算中的利器，numpy 针对矩阵的高效率处理，往往可以达到事半功倍的效果。

上一节：Numpy 数组操作及随机抽样 (/courses/912/labs/3408/document)

下一节：Numpy 数组索引及其他用法 (/courses/912/labs/3430/document)