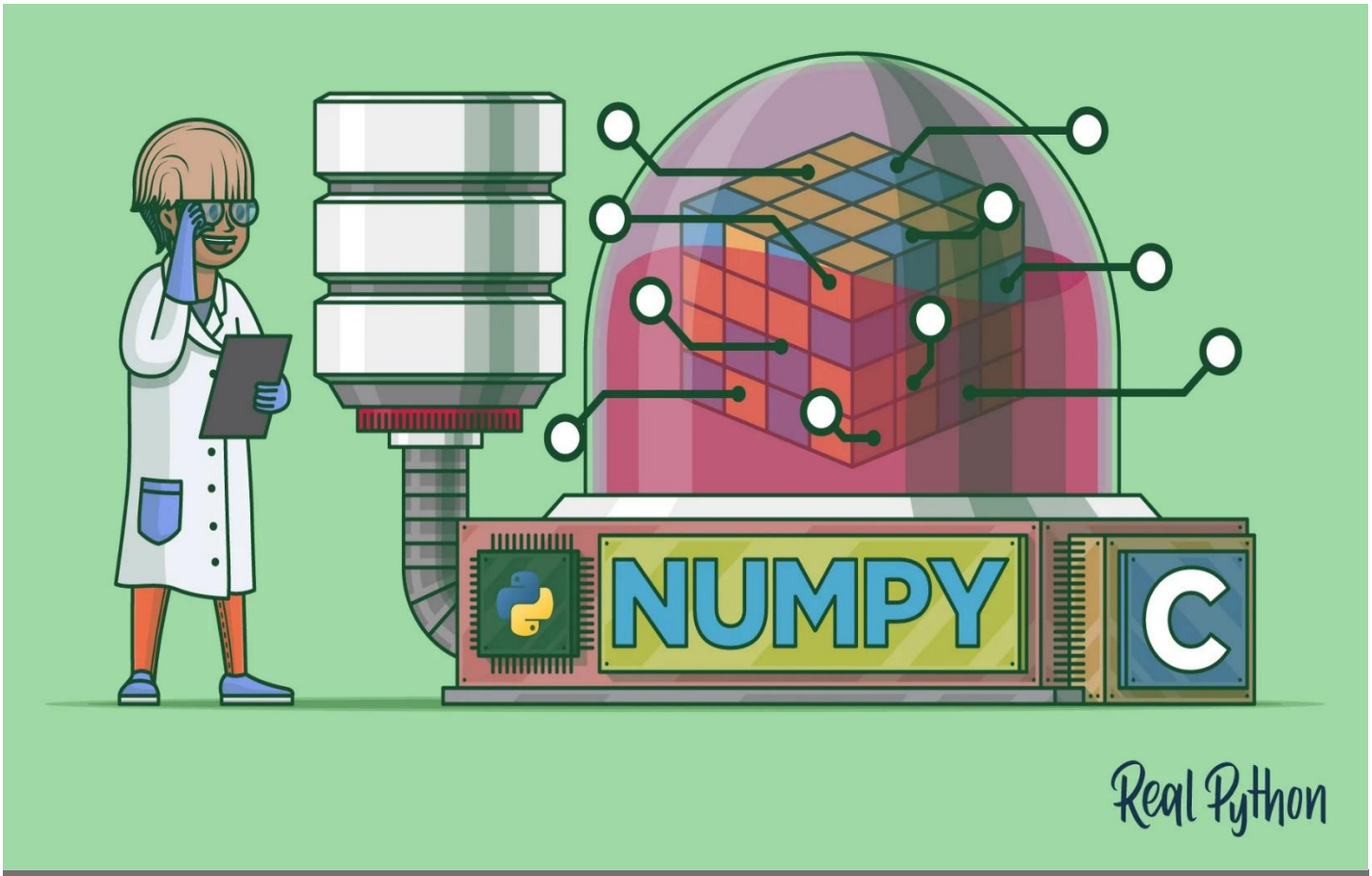


دوره جامع پایتون در یادگیری ماشین

قسمت اول، توابع پرکاربرد

Numpy[1]!

Datalk.ir
Created by : Ali Arabshahi
Contact Us : Linkedin.com/in/mrAliArabshahi

نامپای چیست!

نامپای یه کتابخونه مربوط به جبر خطی هست و این که چرا این قدر مهمه و اسمش رو در آینده به عنوان یک دیتا ساینتیست زیاد می شنوید اینه که خیلی از کتابخونه های معروف و مدل هایی که ازشون استفاده می کنین به عنوان ورودی نیاز با یه مفهومی به اسم آرایه دارن و چیزی که نامپای به شما تحویل میده یک آرایه است. 😉

ما در این جلسه بررسی مفاهیم اساسی این کتابخانه می پردازیم اما قبل از هر چیز نیاز به نصب این کتابخانه داریم:

نصب کتابخانه نامپای

من همیشه به دوستان پیشنهاد می کنم از محیط جوپیتر برای استفاده از زبان پایتون استفاده کنن و معمولا کتابخونه های اساسی رو به صورت پیشفرض در خودش نصب می کنه اما اگر نیاز به نصب این کتابخونه داشتین به ترمینال مربوطه برین (اناکوندا) و دستور زیر رو تایپ کنین. (معمولا نصب در محیط های مختلف برنامه نویسی متفاوت هست برای همین پیشنهاد می کنم یه سرچی هم تو گوگل کنین با این کلمات کلیدی، آموزش نصب کتابخانه و ....)

```
conda install numpy or pip install numpy
```

اگر هم از اناکوندا استفاده نمی کنین بهتون پیشنهاد می کند لینک زیر رو هم یه بررسی کنین:

```
http://docs.scipy.org/doc/numpy-1.10.1/user/install.html
```

استفاده از نامپای

بعد از نصب نامپای نیاز به فرواخونی اون در برنامه مون داریم .دستور زیر این کار رو برامون انجام میده

```
import numpy as np
```

نامپای داخل خودش توابع زیادی داره و ما در این دوره سعی می کنیم به کاربردی ترین اون ها بپردازیم.

بیاین یکم راجع به آرایه ها صحبت کنیم 😀

آرایه ها در نامپای

آرایه ها یه چیزی تو مایه های همون لیست های خودمون هستن با این تفاوت که از جنس بردارن و می تونین کارهای محاسباتی رو راحت تر بر روی کل مقادیری که داخلشون وجود دارن، اعمال کنین .معمولا دو تا چیز خیلی معروف با خودشون میارن .**بردار ها** که بهشون آرایه های یک بعدی گفته میشه و ماتریس ها که به صورت آرایه های دو بعدی هستن .برای این که بتونیم با داده هامون کار بکنیم و کلی کارای جالب روشون انجام بدین استفاده کردن از مفاهیمی که مربوط به جبر خطی و فضای برداری هست خیلی به کمکمون می یاد.

تبدیل لیست به آرایه توسط نامپای

لیست زیر رو در نظر بگیرین.

```
my_list = [1,2,3]
my_list
```

```
[1, 2, 3]
```

حالا توسط کتابخونه نامپای اون رو تبدیل به آرایه می کنیم .تفاوتشون رو می بینین؟

```
np.array(my_list)
```

```
array([1, 2, 3])
```

در مثال بالا با یک لیست مواجه بودیم و خروجی اون تبدیل شد به یک بردار. حالا بیاین با چند تا لیست (لیستی از لیست ها)سر و کار داشته باشیم .چرا؟ 🤷 در دنیای واقعی هر لیست می تونه مشابه یک فیچر (ستون) از داده های ما باشه مثلا یک لیست اسم افراد، یک لیست سن افراد و ...

```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
my_matrix
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

حالا توسط کتابخونه نامپای اونا رو تبدیل به آرایه دو بعدی یا ماتریس می کنیم .این ماتریس می تونه تجمیعی از همه داده های ما باشه.

```
np.array(my_matrix)
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

و به این صورت یک سری لیست رو بردیم در فضای برداری و حالا می تونیم کلی علمیات محاسباتی رو به راحتی روی دیتا هامون انجام بدیم.

تابع های پر کاربرد نامپای

در ادامه به چند تا از مهم ترین تابع هایی که در داخل نامپای وجود دارن می پردازیم :

arange

این تابع لیست از اعداد بین دو مقدار تعیین شده رو به ما تحویل می ده .مثلا اعداد بین 0 و 10

```
np.arange(0,10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

به صورت پیشفرض یه دونه یه دونه اضافه می شه که می تونیم اون رو هم تعیین کنیم .همان طور که در مثال پایین مشاهده می کنید .خروجی تابع زیر (اعداد بین 0 تا 11 شامل 11 نمی شود) با گام 2 می باشد. یعنی همان اعداد زوج خودمون😎

```
np.arange(0,11,2)
```

```
array([ 0,  2,  4,  6,  8, 10])
```

ساخت آرایه های صفر و یک

در پایین نحوه ساخت آرایه های صفر و یا یک را مشاهده می کنین که در دنیای جبر خطی هر کدوم می تونن کاربرد های باحالی داشته باشن.

```
np.zeros(3)
```

```
array([0., 0., 0.])
```

اعداد داخل پرانتز به ترتیب بیانگر 5 سطر و 5 ستون می باشند .

```
np.zeros((5,5))
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

و در پایین هم بردار و یا آرایه یک رو مشاهده می کنیم. کاربردش چیه؟ خیلی ساده! هر چی ضرب در ۱ میشه خودش و با این ترفند ساده می شه با کلی از مسائل جبر خطی سر و کله زد.

```
np.ones(3)
```

```
array([1., 1., 1.])
```

```
np.ones((3,3))
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

linspace

این تابع تعداد تعیین شده ای رو (پارامتر سوم که در پایین 3 تعریف شده است) را در یک بازه مشخص که در مثال پایین بین 0 و 10است را به ما تحویل می دهد.

```
np.linspace(0,10,3)
```

```
array([ 0.,  5., 10.])
```

```
np.linspace(0,10,50)
```

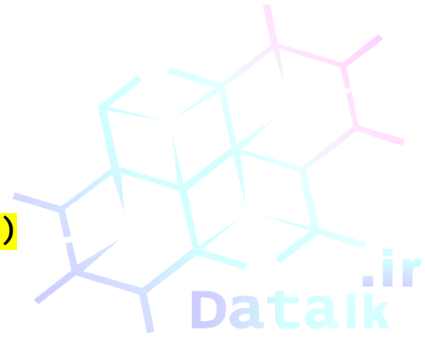
```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
        2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
        3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
        4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
        5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
        6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
        7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
        8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
        9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

eye

تعریف ماتریس یکه توسط تابع زیر صورت می گیره .همان طور که احتمالا نمی دونین !هر ماتریسی ضرب در ماتریس یکه یا همانی ، تبدیل به خودش می شه .یکی از بی خاصیت ترین مفاهیم در جبر خطی و در عین حال پرکاربردترین.

```
np.eye(4)
```

```
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```



Random

زمانی که می‌خوایم یک سری داده به صورت رندوم تولید کنیم از این تابع استفاده می کنیم که درون خودش تابع های دیگه ای رو جای داده.

rand

این تابع ، ماتریس یا برداری از اعداد تصادفی با توزیع یکنواخت(حتما سعی‌کنید کم کم با مفاهیم علم آمار به خصوص توزیع های مختلف آشنا بشین) بین اعداد صفر و یک رو تولید می کنه.

```
np.random.rand(2)
```

```
array([0.3559501 , 0.59749345])
```

آقای نامپای خان ! من ماتریسی می خوام که 5 سطر و 5 ستون داشته باشه و مقادیر اون اعداد تصادفی باشن با توزیع یکنواخت و بین 0 و 1:

```
np.random.rand(5,5)
```

```
array([[0.41881215, 0.27456824, 0.60430626, 0.81696813, 0.4738828 ],
       [0.29701508, 0.14773108, 0.42029791, 0.29759995, 0.6627741 ],
       [0.28666253, 0.9205843 , 0.46420464, 0.24500328, 0.33719412],
       [0.84781644, 0.81808906, 0.50982746, 0.84902031, 0.05064031],
       [0.76569843, 0.39343513, 0.5370932 , 0.79101072, 0.27166449]])
```

به همین راحتی 😊

randn

مشابه تابع بالاست با این تفاوت که توزیع اون ها از نوع توزیع نرمال هست.

```
np.random.randn(2)
```

```
array([1.43347188, 2.16689865])
```

```
np.random.randn(5,5)
```

```
array([[ 0.50996179,  0.20075232,  1.07247229,  0.77345163,  0.29741362],
       [-0.22304086,  0.92869599, -1.72093038, -0.46961579, -0.24515472],
       [-0.65731575,  1.05090805, -0.54166518,  1.10907127,  0.31295193],
       [-0.73624926,  0.89441369,  0.810255  , -0.31651652,  0.42331168],
       [ 1.25739985,  0.6640432 ,  0.5403786 , -1.31059665,  0.59521381]])
```

randint

یه عدد صحیح رندوم بین بازه تعیین شده(1 و 100) و به تعداد مشخص شده (در مثال اولی چون تعیین نشده به طور پیش فرض یکی و در مثال دومی 3 تا) به ما تحویل می ده

```
np.random.randint(1,100)
```

```
53
```

```
np.random.randint(1,100,10)
```

```
array([19, 30, 55, 98, 70, 78, 27, 63, 65, 91])
```

توابعی برای پی بردن به ویژگی آرایه ها در نامپای

بردارهای زیر رو تولید می کنیم.

```
arr = np.arange(25)
ranarr = np.random.randint(0,50,10)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

```
ranarr
```

```
array([48,  8,  4, 36, 42, 16, 33, 11, 41, 18])
```

Reshape

ما یک بردار داریم و توسط این تابع اون رو از بردار(آرایه تک بعدی) به یک ماتریس (آرایه دو بعدی)تبدیل می کنیم

```
arr.reshape(5,5)
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

یه نکته خیلی مهم اینه که اگر تعداد اعداد بردار اولیه تون 25 تاست باید ماتریسی که تعیین می کنین هم شامل 25 مقدار باشه(5*5=25)

max,min,argmax,argmin

برای پیدا کردن کمترین ، بیشترین و ایندکس یا جایگاه های آن ها در آرایه ها به کار میان

```
ranarr
```

```
array([10, 12, 41, 17, 49,  2, 46,  3, 19, 39])
```

```
ranarr.max()
```

```
48
```

```
ranarr.argmax()
```

```
0
```

```
ranarr.min()
```

```
4
```

```
ranarr.argmin()
```

```
2
```

Shape

شکل آرایه ها رو به ما نشون میده، یعنی این که چه تعداد سطر و چه تعداد ستون داریم.

```
arr.shape
```

```
(25,)
```

```
arr.reshape(1,25)
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
        16, 17, 18, 19, 20, 21, 22, 23, 24]])
```

```
arr.reshape(1,25).shape
```

```
(1, 25)
```

```
arr.reshape(25,1)
```

```
array([[ 0],
       [ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
```

```
       [10],
       [11],
       [12],
       [13],
       [14],
       [15],
       [16],
       [17],
       [18],
       [19],
       [20],
       [21],
       [22],
       [23],
       [24]])
```

```
arr.reshape(25,1).shape
```

```
(25, 1)
```

dtype

برای مشخص کردن جنس داده هایی که داخل آرایه ها وجود دارن که در بردار زیر عدد صحیح می باشند!

```
arr.dtype
```

```
dtype('int64')
```

تا جلسه دوم، خدا نگهدار

\\\😎///