Análisis de Datos y Aprendizaje Máquina con Tensorflow 2.0: Pre-procesamiento de Datos para Aprendizaje Máquina

2019/09/30

Revisión de Probabilidad

Objetivo: Conocer las herramientas de numpy para estadística.

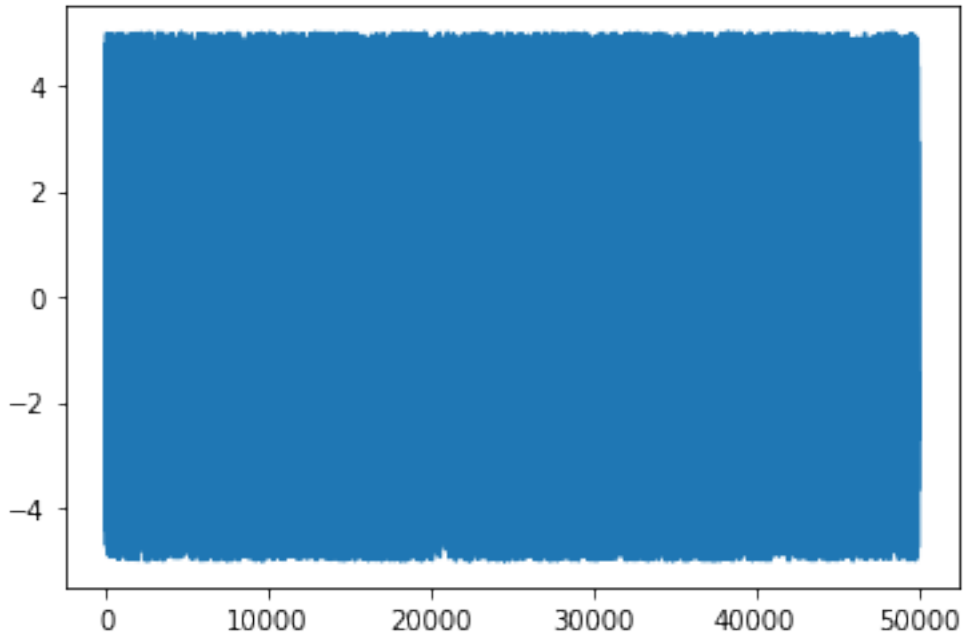
```
In [2]: import numpy as np
        x=np.random.rand(1,5)
        x
```

```
Out[2]: array([[0.46388743, 0.77432407, 0.45674028, 0.02040987, 0.39911621]])
```

```
In [3]: import random
        y=random.sample(range(-5, 5), 5)
        y
```

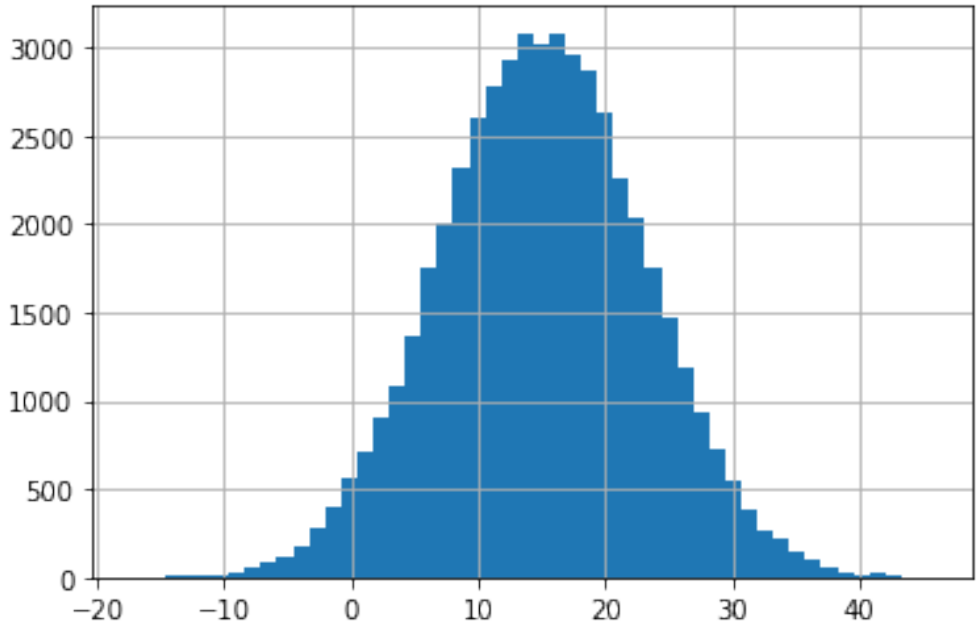
```
Out[3]: [4, -2, -1, 3, 0]
```

```
In [15]: import matplotlib.pyplot as plt

         c=np.random.uniform(-5, 5, 50000)

         fig = plt.figure()
         ax = fig.add_subplot(1, 1, 1)


         plt.plot(c, "-")
         plt.show()
```



Distribución normal

```
In [5]: mu, sigma = 15, 8 # media y desviación estandar
        s = np.random.normal(mu, sigma, 50000)
        plt.hist(s, bins=50)
        plt.grid()
        plt.show()
```

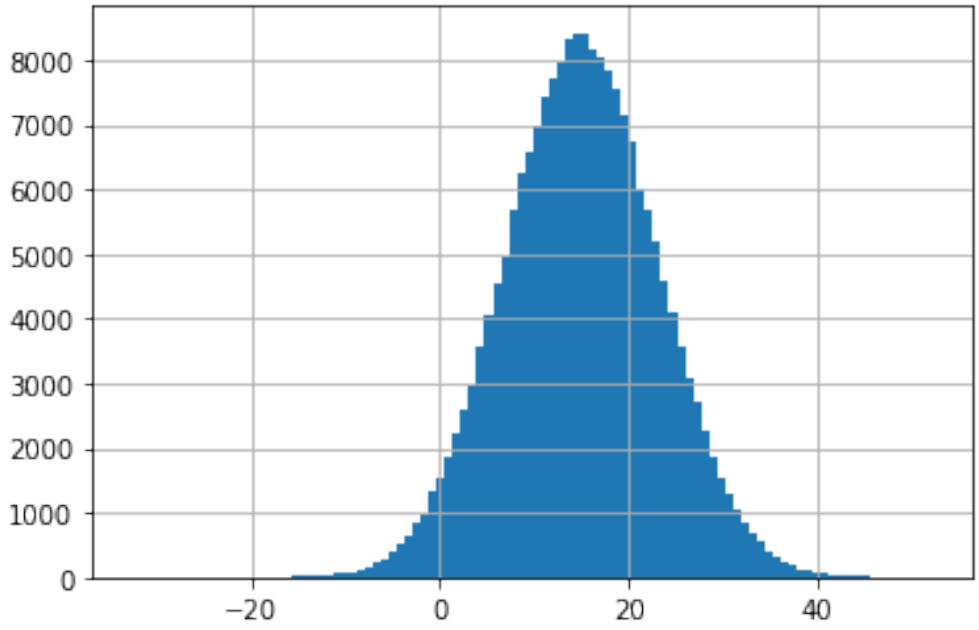


Graficar z con 200 mil muestras histograma de 100 barras

$$z = x + y$$

$$N(15, 8)N(0, 1)$$

```
In [6]: mu, sigma = 15, 8
        x = np.random.normal(mu, sigma, 200000)

        mu, sigma = 0, 1
        y = np.random.normal(mu, sigma, 200000)

        z = x+y
        plt.hist(z, bins=100)
        plt.grid()
        plt.show()
```

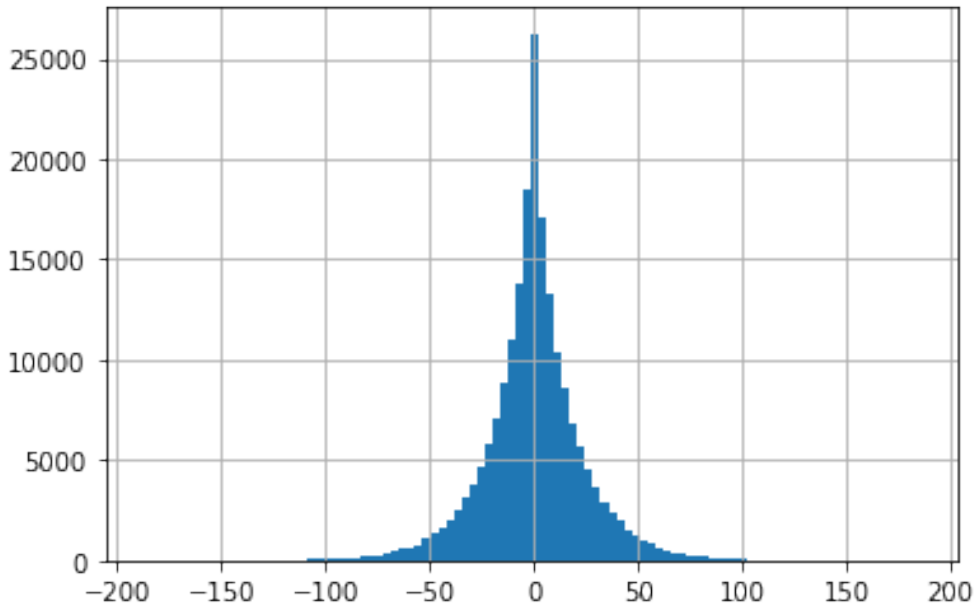


Graficar z con 200 mil muestras histograma de 100 barras

$$z = xy$$

$$N(0,2)N(10,6)$$

```
In [7]: mu, sigma = 0, 2
        x = np.random.normal(mu, sigma, 200000)

        mu, sigma = 10, 6
        y = np.random.normal(mu, sigma, 200000)

        z = x*y
        plt.hist(z, bins=100)
        plt.grid()
        plt.show()
```



Visualizar distribución con pandas

```
In [8]: mu, sigma = 0, 1
        s = np.random.normal(mu, sigma, (10000,4))

        import pandas as pd
        pd.DataFrame(s, columns = ['x1', 'x2', 'x3', 'x4']).tail(10)
```

Out[8]:

	x1	x2	x3	x4
9990	1.404036	-0.342709	0.112155	0.236394
9991	1.274652	-0.940055	-0.359976	-0.667419
9992	0.377774	0.248808	-0.119469	0.092184
9993	-0.372053	0.475924	-0.832724	-0.251068
9994	-0.395664	0.276921	-1.052459	0.821314
9995	-0.720800	0.695106	-0.562427	-0.585590
9996	-0.519405	0.625394	-1.231131	2.362969
9997	-1.363070	1.809466	-0.580014	0.869753
9998	-0.580251	-0.989288	-1.454165	-0.969680
9999	0.693807	0.291143	-0.144448	0.268554

Media

```
In [9]: m = np.mean(s, axis=0)
        m
```

```
Out[9]: array([ 0.01315903, -0.0216138 , -0.00331784, -0.00494405])
```

Desviación estándar

```
In [10]: d = np.std(s, axis=0)
         d

Out[10]: array([0.99460876, 1.00427593, 0.9849591 , 0.99220684])
```

Covarianza de las dos primeras columnas

```
In [11]: C = np.cov(s[:,0],s[:,1])
         C

Out[11]: array([[ 0.98934551, -0.01182995],
                [-0.01182995,  1.00867101]])
```

Coeficientes de correlación en las dos primeras columnas

```
In [12]: Co = np.corrcoef(s[:,0],s[:,1])
         Co

Out[12]: array([[ 1.        , -0.01184225],
                [-0.01184225,  1.        ]])
```

Arreglo con diferentes distribuciones

```
In [13]: x1 = np.random.normal(0, 1, 10000)
         x2 = np.random.uniform(0, 1, 10000)
         x3 = np.random.normal(3, 7, 10000)
         x4 = np.random.uniform(-2, 2, 10000)

In [14]: a = np.array([x1, x2, x3, x4])
         a.T # Transponiendo

Out[14]: array([[ 0.93239899,  0.41368919, 10.47231283,  0.42478918],
                [-1.57225651,  0.74483976,  1.25492919,  0.95844686],
                [ 0.07700187,  0.29732282, -0.54796522,  0.26998348],
                ...,
                [-0.42297886,  0.81414131, -1.31178767,  1.58362114],
                [ 0.9215658 ,  0.23208305, -1.63295614, -1.37348695],
                [-0.5108318 ,  0.81705609, -8.9710347 , -1.94914177]])
```