Data Mining

Tamás Budavári - budavari@jhu.edu
Class 2

- Probability Density Function (PDF)
- Cumulative Density Function (CDF)
- Moments

Probability Density Function

- PDF a.k.a. Probability Distribution Density Function
- Probabiliy of x between a and b for any (a, b) is

$$P_{ab} = \int_a^b p(x)\, dx$$

- Always

$$\int_{-\infty}^{\infty} p(x)\, dx = 1$$

- Example 1: uniform distribution on (a, b)

$$U(x; a, b) = \frac{\mathbf{1}_{ab}(x)}{b-a}$$, where $\mathbf{1}_{ab}(x)$ is 1 between a and b, but 0 otherwise

- Example 2: Gaussian or normal distribution

$$G\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Example 3: Log-normal

Gauss on Money!

- Even the formula

Cummulative Distribution Function

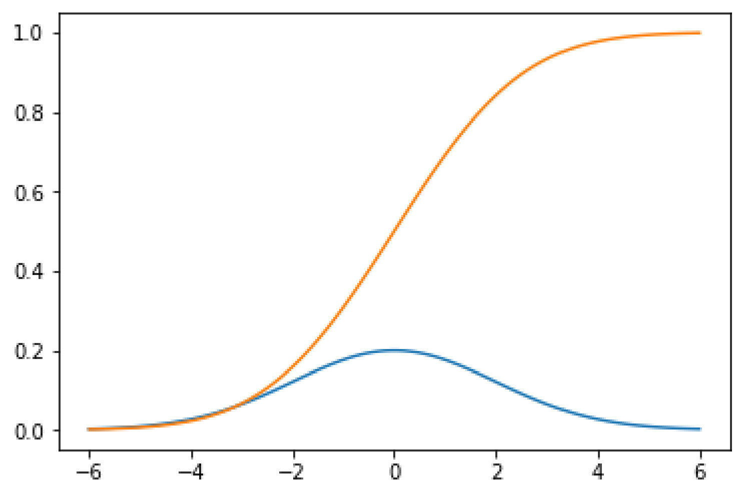
- Integral up to given x: prob of being less than x

$$\mathrm{CDF}(x) = \int_{-\infty}^{x} p(t)\, dt$$

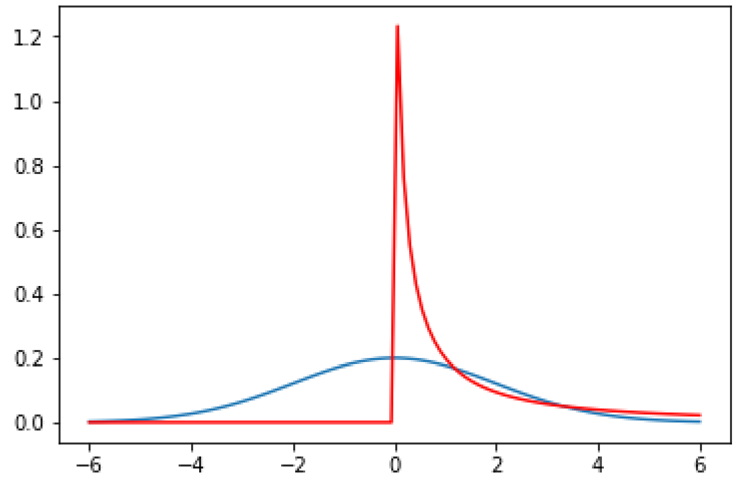
In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
from scipy.stats import norm as gaussian
x = np.linspace(-6,6,100)
mu, sig = 0, 2
plot(x, gaussian.pdf(x,mu,sig));
plot(x, gaussian.cdf(x,mu,sig));
```



In [3]:
```
from scipy.stats import lognorm
plot(x, gaussian.pdf(x,0,sig));
plot(x, lognorm.pdf(x,sig), color='r');
```



Characterization of PDFs

- Expectation value of X

$$\mu = \mathbb{E}[X] = \int_{-\infty}^{\infty} x\, p(x)\, dx$$

- Expectation value of any $f(X)$

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)\, p(x)\, dx$$

- Moments

$$\mathbb{E}[X^k]$$

- Central moments

$$\mathbb{E}\left[(X-\mu)^k\right]$$

- Variance

$$\mathbb{Var}[X] = \mathbb{E}\left[(X-\mu)^2\right]$$

- Standard deviation

$$\sigma = \sqrt{\mathbb{Var}[X]}$$

- Normalized moments

$$\mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^k\right]$$

- Skewness

> 3rd normalized moment (k=3)

- Kurtosis

> 4th normalized moment (k=4)

```
In [4]: mean, var, skew, kurt = gaussian.stats(mu, sig, moments='mvsk');
        mu, sig, mean, var, skew, kurt
```

```
Out[4]: (0, 2, array(0.0), array(4.0), array(0.0), array(0.0))
```

```
In [5]: mean, var, skew, kurt = lognorm.stats(sig, moments='mvsk');
        sig, mean, var, skew, kurt
```

```
Out[5]: (2,
         array(7.38905609893065),
         array(2926.359837008584),
         array(414.359343300147),
         array(9220556.977307005))
```

Python by Examples

- tuple list function class for map lambda import
- numpy matplotlib

In [6]:
```python
# tuple
t = (1,2)
t = 100, 0.1
N, mu = t
N
```

Out[6]: 100

In [7]:
```python
# list
l = [1,2,3,4,5]
[l,l]
```

Out[7]: [[1, 2, 3, 4, 5], [1, 2, 3, 4, 5]]

In [8]:
```python
# function
def f(x,k=2):
    return x**k

f3 = f(3)
print (f3)
f(2), f(2,2), f(2,3), f(2,k=4), f3
```

```
9
```

Out[8]: (4, 4, 8, 16, 9)

In [9]:
```python
import math

# object-oriented programming
class Robot(object):

    def __init__(self, x=0, y=0, angle=0):
        self.x, self.y, self.angle = x, y, angle
        self.path = [(x,y)]

    def move(self, l):
        self.x += l* math.cos(self.angle)
        self.y += l* math.sin(self.angle)
        self.path.append( (self.x, self.y) )

    def left(self, a):
        self.angle += a

    def right(self, a):
        self.left(-a)
```

In [10]:
```python
r = Robot(100,0,np.pi/2)
r.move(10)
r.left(math.pi/4)
r.move(5)
r.path
```

Out[10]:
```
[(100, 0), (100.0, 10.0), (96.46446609406726, 13.535533905932738)]
```

In [11]:
```python
import sys
sys.stdout.write('asdf')
sys.stdout.write('fdasfsad')
```

```
asdffdasfsad
```

In [12]:
```
out = open('test.txt', 'w')
# loops
for i in range(10):
    out.write (str(i*i) + ' ')
print ('done')
```

```
done
```

In [13]:
```
# lambda expressions
g = lambda x: x*x
g(2)
```

Out[13]:
```
4
```

In [14]:
```
# using math functions and routines
import math

math.pi, math.sin(1.57)
```

Out[14]:
```
(3.141592653589793, 0.9999996829318346)
```

In [15]:
```
# same using numpy's methods
np.pi, np.sin(1.57), np.sin([0,np.pi,1.57])
```

Out[15]:
```
(3.141592653589793,
 0.99999968293183461,
 array([  0.00000000e+00,   1.22464680e-16,   9.99999683e-01]))
```

In [16]:
```
# numpy methods work also on arrays, e.g., elementwise
np.sin( [1.57, 3.14, np.pi] )
```

Out[16]:
```
array([  9.99999683e-01,   1.59265292e-03,   1.22464680e-16])
```

In [17]:
```
# arrays: vectors and matrices
import numpy as np
l = [1,2,3]
a = np.array([l,l], dtype=np.int32)
print (a.shape)
print (a.T)

b = a.T.dot(a)
b
```

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

Out[17]:
```
array([[ 2,  4,  6],
       [ 4,  8, 12],
       [ 6, 12, 18]], dtype=int32)
```

In [18]:
```
# slicing arrays
b[1:2,0:-1]
```

Out[18]:
```
array([[4, 8]], dtype=int32)
```

In [19]:
```
# componentwise operations
print (np.sin(l))

for s in map(math.sin, l):
    print (s)
```

```
[ 0.84147098  0.90929743  0.14112001]
0.8414709848078965
0.9092974268256817
0.1411200080598672
```

In [20]:
```
plt.plot([1,2,3,4,5],[5,4,5,2,4], 'ro-');
plt.savefig('test.png')
# change extension to .pdf to have it in that format
```