ENGG1003 - Thursday Week 9

Random numbers from normal distributions —aka random numbers from Gaussian distributions

Steve Weller

University of Newcastle

6 May 2021

Last compiled: May 6, 2021 12:53pm +10:00

Lecture overview

1. normal distribution
   - also known as "bell curve", or Gaussian distribution
   - today: draw random samples from "standard" normal distribution
2. compute probabilities using normal distribution
   - uses numerical integration

1) Normal distribution

- intoduced *uniformly* distributed random numbers in week 4
- today: will focus on "standard" normal, extend next week to general form
- goal today is to random samples from a standard normal (Gaussian) distribution, and compute probability of number falling in specified range
- aka Gaussian random numbers
- widely appear in engineering

Standard normal distribution

- Straight into it, generate $100,000$ random numbers generated using `normal` function in numpy's random library
- standard: mean = 0, std = 1

filename.py

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
x = np.random.normal(0.0, 1.0, size=100000)

plt.hist(x, 10)
plt.show()
```

- code commentary
- normal random numbers aka Gaussian distribution
- general form of call to normal()
- explain hist()
- live demo
- nothing much to see in plot of numbers themselves "noise"

Histogram

- interpret histogram
- bins, counts, examples
- call hist to return bins—too hard?

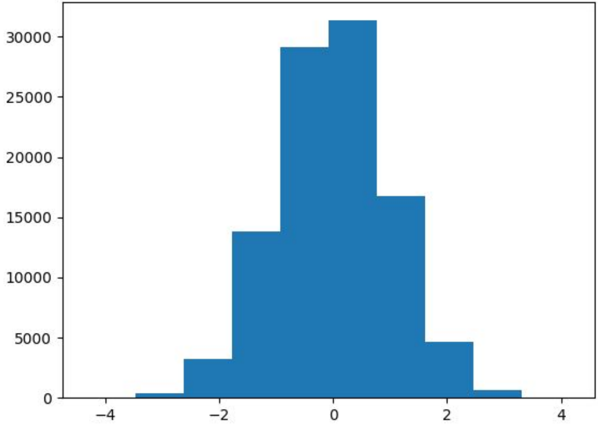
- A histogram is a graph showing frequency distributions
- It is a graph showing the number of observations within each given interval.
- To visualize the data set we can draw a histogram with the data we collected
- We will use the Python module Matplotlib to draw a histogram

Python code

histdemo.py

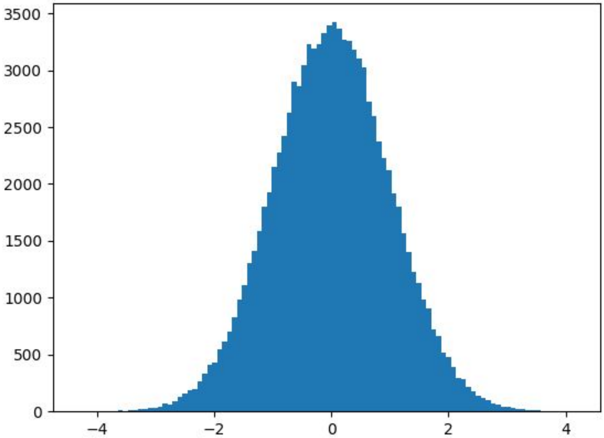
```python
# histdemo
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
d = np.random.normal(0.0, 1.0, size=100000)

x = np.linspace(-5,5,num=1000)
f = 1/(np.sqrt(2 * np.pi)) * np.exp(-x**2 / 2)

plt.hist(d, 100, density=True)
plt.plot(x, f, color='r', linewidth=3)
#plt.hist(d, 100)

#plt.plot(d, 'o')
plt.show()
```

Histogram: 10 bins



- eg: XXX samples in range $[XXX, XXX]$

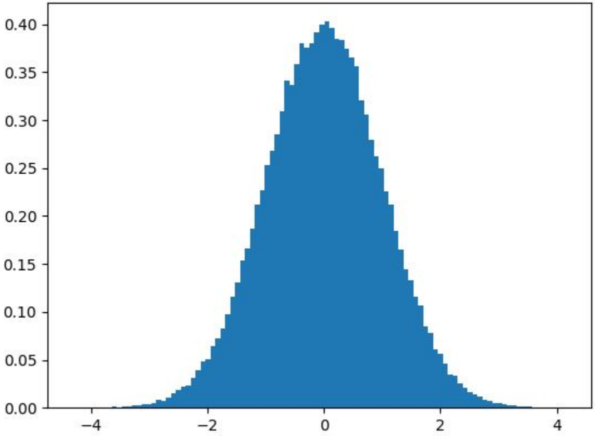
Histogram: 100 bins



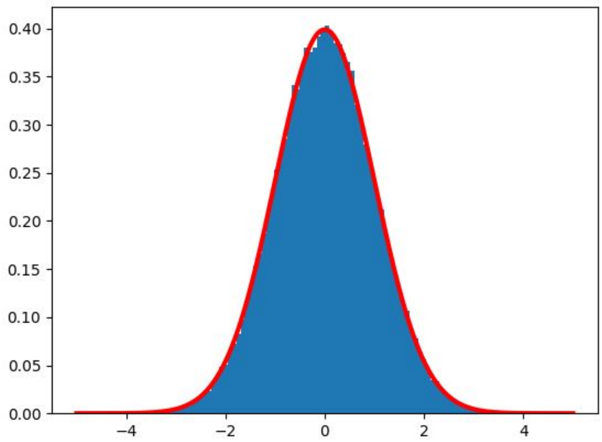
- identical data set as for 10 bins

Normalized histogram (area 1): 100 bins

```
plt.hist(x, 100, density=True)
```



Normalized histogram with PDF



red curve is *probability density function (PDF)*

Standard normal distribution

Standard normal probability density function:

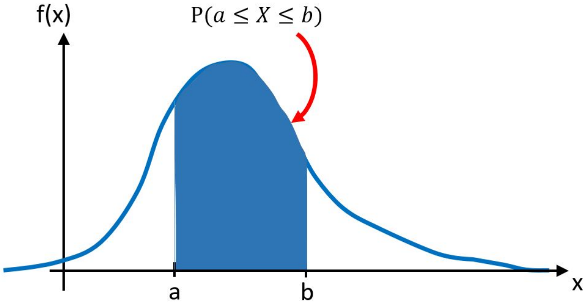
$$\boxed{f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}}$$

- we'll see a more general form of normal distribution next lecture
- standard normal is a special case: mean 0 and std 1
- reflections
- qualitative description of what a pdf is

Probability density functions

If X is a random number drawn from a distribution with PDF $f(x)$, probability X takes a value in interval $[a, b]$ is

$$\boxed{\mathrm{P}(a \leq X \leq b) = \int_a^b f(x)dx}$$



Properties of a PDF

To qualify as a PDF, function f must be non-negative, and must have the normalization property. Thsi means the entire area under the graph of f must be equal to 1

- area under $f(x)$ is 1
  - reason for the $1/\sqrt{2\pi}$ factor
- $f(x) \geq 0$ for all x, since probability can't be negative
- total area under pdf is 1, since X must take some value

$$\int^{\infty} f(x)dx = \mathrm{P}(-\infty \leq X \leq \infty) = 1$$

2) Integration

the story so far . . .

- PDF of random numbers following *standard normal distribution* is a "bell curve"

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

- probability of a random number drawn from standard normal distribution taking value in interval $[a, b]$ is

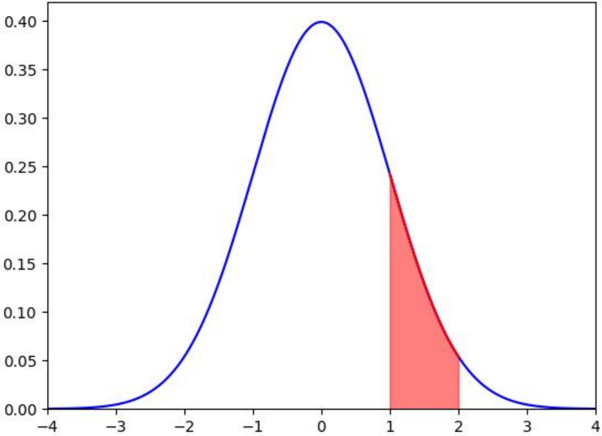
$$\mathrm{P}(a \leq X \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-x^2/2} dx$$

Example

- exact expression doesn't exist for $\int_a^b e^{-x^2/2}dx$
- need to use numerical integration

Example

- $a = 1$, $b = 2$
- calculated probability using `standardnormal.py` is 0.1359
  - uses trapezoidal method with 100 sub-intervals on $[1, 2]$



red shaded area is

$$\frac{1}{\sqrt{2\pi}}\int_1^2 e^{-x^2/2}dx \approx 0.1359$$

Python code: fraction of numbers in $[a, b]$

standardnormal.py

```
1  # standardnormal
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  def f(x):
6      return 1/(np.sqrt(2 * np.pi)) * np.exp(-x**2 / 2)
7
8  def trapezoidal(f, a, b, n):
9      h = (b-a)/n
10     f_sum = 0
11     for i in range(1, n, 1):
12         x = a + i*h
13         f_sum = f_sum + f(x)
14     return h*(0.5*f(a) + f_sum + 0.5*f(b))
```

- lines 5–6: PDF of standard normal distribution

Python code

`standardnormal.py`—continued

```
1 a = 1
2 b = 2
3 prob_ab = trapezoidal(f, a, b, 100)
4 print('Probability X in range [{},{}] is: {:.4f}'.format(a, b, prob_ab))
5
6 x = np.linspace(-4, 4, 1000)
7 xab = np.linspace(a, b, 100)
8
9 plt.plot(x, f(x), 'b')              # standard normal pdf
10 plt.plot(xab,f(xab),'r')
11 plt.fill_between(xab,f(xab),color='r',alpha=0.5) #alpha=transparency
12 plt.axis([-4, 4, 0, 0.42])
13 plt.show()
```

- line 3: approximate $\frac{1}{\sqrt{2\pi}} \int_1^2 e^{-x^2/2} dx$
- line 7: $1 \leq x \leq 2$ for red shaded area plot

Demo of standard normal generation

- generate 10^6 random numbers
- expect $10^6 \times 0.1359 = 135,900$ in range $[1, 2]$
- live demo
- results

Python code

standardnormaldemo.py

```
# standardnormaldemo
import numpy as np

N = 1000000
x = np.random.normal(0.0, 1.0, size=N)
a = 1
b = 2
num_ab = 0
for k in range(0, len(x)):
    if a <= x[k] <= b:
        num_ab += 1

print('{} standard normal random numbers'.format(N))
print('Fraction of random numbers in range [{},{}] = {:.4f}'.
    format(a, b, num_ab/N))
```

- fraction of random numbers in range $[1, 2] \approx 0.136$

Lecture summary

1. xxx
2. xxx
3. what's next