ENGG1003 - Monday Week 10

Normal distributions: extensions and applications
Curve-fitting

Steve Weller

University of Newcastle

10 May 2021

Last compiled: May 7, 2021 5:36pm +10:00

Lecture overview

1. Normal distributions
   - extension of *standard* normal distribution (previous lecture)
   - applications

2. Curve-fitting

1) Normal distributions

- **quick recap** of standard normal PDF: equation, interpretation, how to generate & plot histogram
- reiterate importance of normal distribtion in applications
- but standard normal is inflexible

Recap: standard normal distribution

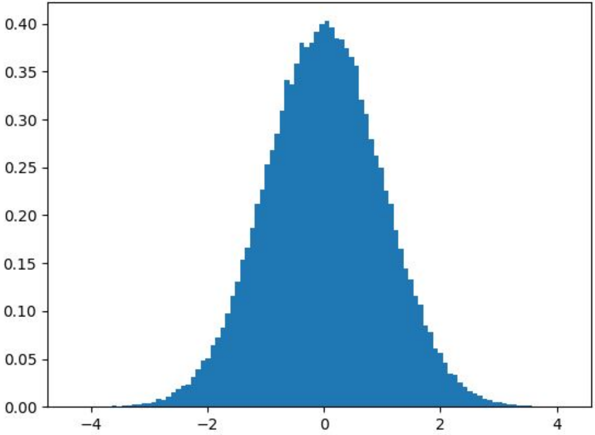
Standard normal probability density function:

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

- *standard* normal distribution is a special case of normal (Gaussian) distribution
- corresponds to parameters $\mathbf{0.0}$ and $\mathbf{1.0}$ in:

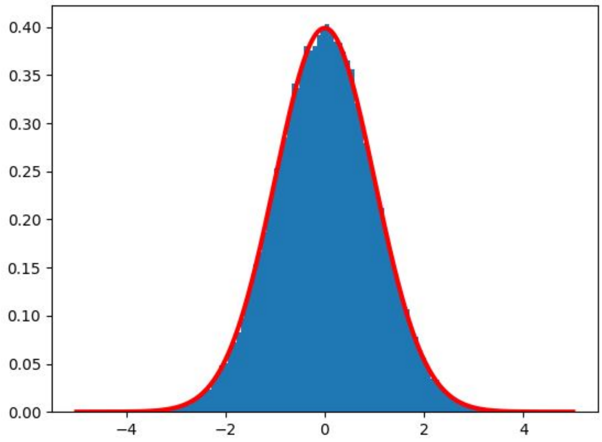
```
x = np.random.normal(0.0, 1.0, size=100000)
```

Normalized histogram (area 1), 100 bins



- same histogram, except total area of rectangles is normalized to be 1

Normalized histogram with PDF

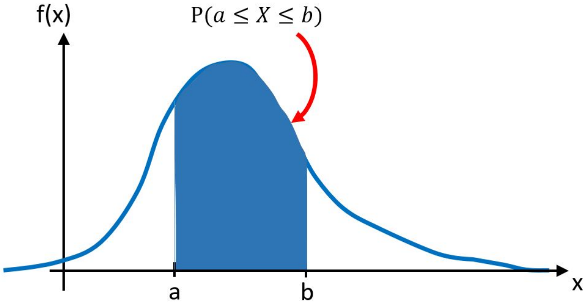


red curve is *probability density function (PDF)*

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

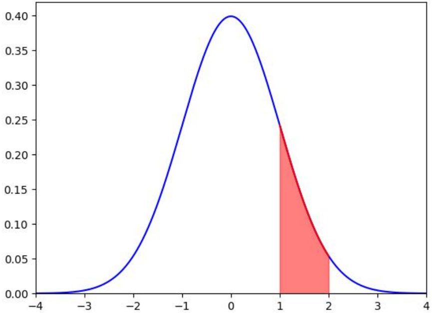
$$\boxed{\mathrm{P}(a \leq X \leq b) = \int_a^b f(x)dx}$$



Example

Use trapezoidal method to approximate $\mathrm{P}(1 \leq X \leq 2)$ when X is drawn from standard normal distribution

$$\mathrm{P}(1 \leq X \leq 2) = \frac{1}{\sqrt{2\pi}} \int_1^2 e^{-x^2/2} dx \approx \mathbf{0.1359}$$

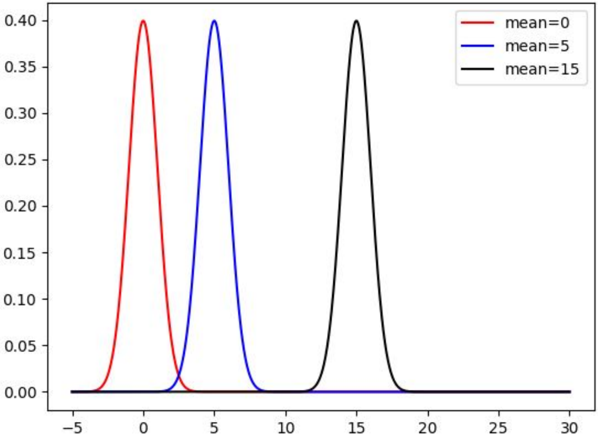


- reiterate importance of normal/Gaussian in applications
- but standard normal is inflexible
- now **experimentally observe** impact of first two parameters in `normal()` function call

Impact of mean

- shifts average (mean) value
- left-right shift of PDF
- image here: overlay PDFs for $\mu = 0, 5, 20$

Mean demo



- blah

Python code

meandemo.py

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

def f(x, mu, sigma):
    return 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu)**2 / (2 * sigma**2 ))

x = np.linspace(-5, 30, 1000)

plt.plot(x, f(x, 0, 1), color='r', label='mean=0')
plt.plot(x, f(x, 5, 1), 'b', label='mean=5')
plt.plot(x, f(x, 15, 1), 'k', label='mean=15')
plt.legend()
plt.show()
```

- blah

Impact of standard deviation σ

- shifts spread of PDF
- interpretation of "standard deviation"
- image here
- "most" of PDF within plus/minus 3 sigma of mean

"In statistics, the standard deviation is a measure of the amount of variation or dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the values are spread out over a wider range. Standard deviation may be abbreviated SD, and is most commonly represented in mathematical texts and

Impact of stedev

- XXX

Stddev demo

- blah

Normal PDF

$$\boxed{f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}$$

- mean μ
- standard deviation σ
- what are you expected to do with this PDF?
  1. call np.random.standard() to generate random numbers for specified μ and σ
  2. compute prob X in range $[a, b]$ using numerical integration (trapezoidal)

Standard normal as special case

Important special case: $\mu = 0$ and $\sigma = 1$

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

Key point: standard normal distribution has a mean of 0 and a standard deviation of 1

Application 1

- xxx

Application 2

- XXX

2) Curve-fitting

- straight line fitting
- low-order polynomials
- maybe fitting exponentials (?)
- `scipy.optimize.curve_fit`
- applications

In [1]:
```
import numpy as np
from scipy.optimize import curve_fit
```

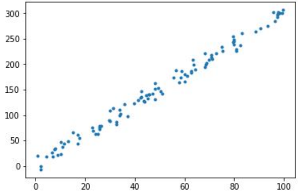
The full documentation for the `curve_fit` is available here, and we will look at a simple example here, which involves fitting a straight line to a dataset.

We first create a fake dataset with some random noise:

In [2]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
x = np.random.uniform(0., 100., 100)
y = 3. * x + 2. + np.random.normal(0., 10., 100)
plt.plot(x, y, '.')
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x1186f73c8>]
```



Let's now imagine that this is real data, and we want to determine the slope and intercept of the best-fit line to the data. We start off by defining a function representing the model:

In [4]:
```
def line(x, a, b):
    return a * x + b
```

The arguments to the function should be `x`, followed by the parameters. We can now call `curve_fit` to find the best-fit parameters using a least-squares fit:

In [5]:
```
popt, pcov = curve_fit(line, x, y)
```

- XXX

- XXX

- XXX

Lecture summary

- Normal distributions
- Curve-fitting