ENGG1003 - Monday Week 10

Normal distributions: extensions and applications
Curve-fitting

Steve Weller

University of Newcastle

10 May 2021

Last compiled: May 7, 2021 12:46pm +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
- normal aka Gaussian
- introduce mean μ and standard deviation σ
- shape of PDFs with μ and σ
- applications

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
```

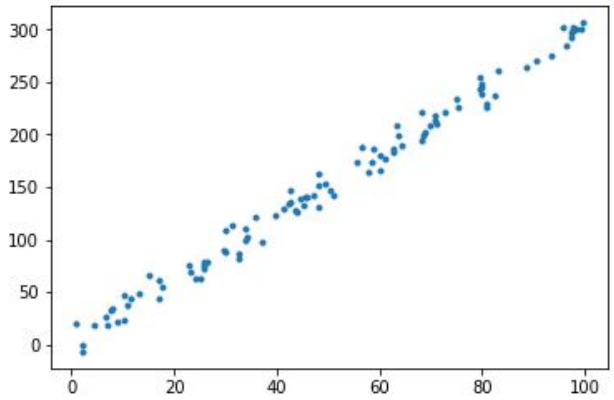
The full documentation for the `curve_fit` is available here, and we will look at a simple example here, which involves fitting a straight line to a dataset.

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)
```

Lecture summary

- Normal distributions
- Curve-fitting