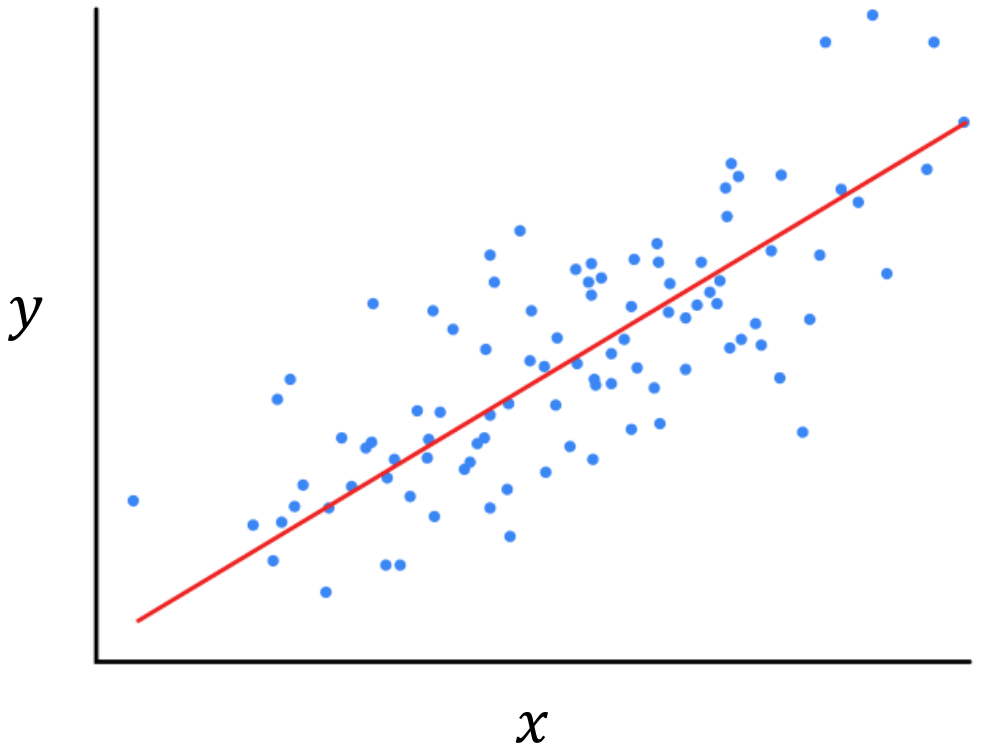
Introduction to Computer Programming Lecture 8.3:

Curve Fitting

Hemma Philamore

Department of Engineering Mathematics

Linear Regression



$$y = mx + c$$

m = gradient
c = y intercept

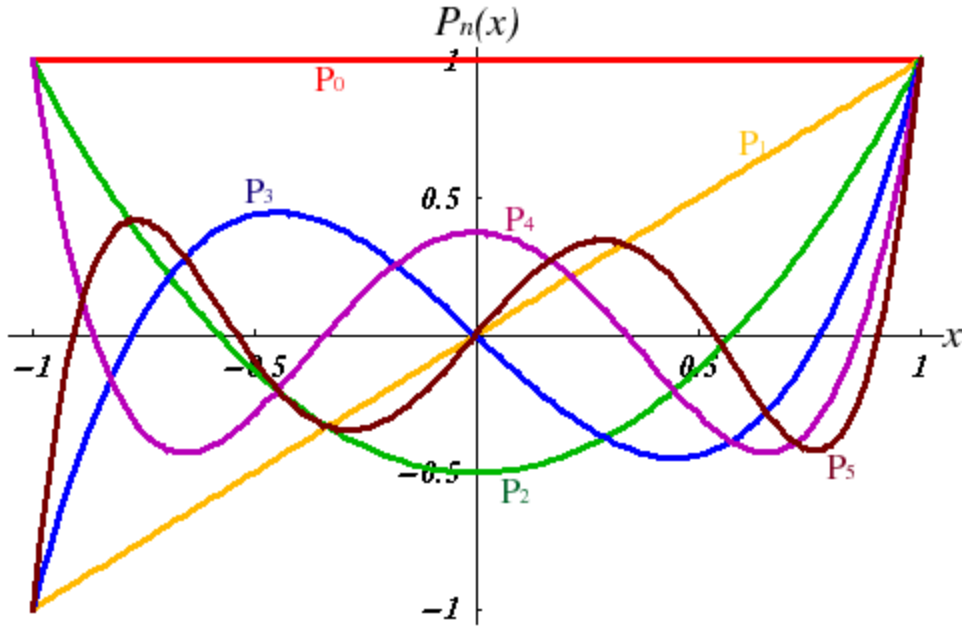
Fitting a Polynomial Function

Polynomial function: Only non-negative powers of x

1st order polynomial function (linear function) $y = \mathbf{a}x^1 + \mathbf{b}x^0$

2nd order polynomial function $y = \mathbf{c}x^2 + \mathbf{d}x^1 + \mathbf{e}x^0$

3rd order polynomial function $= \mathbf{f}x^3 + \mathbf{g}x^2 + \mathbf{h}x^1 + \mathbf{i}x^0$



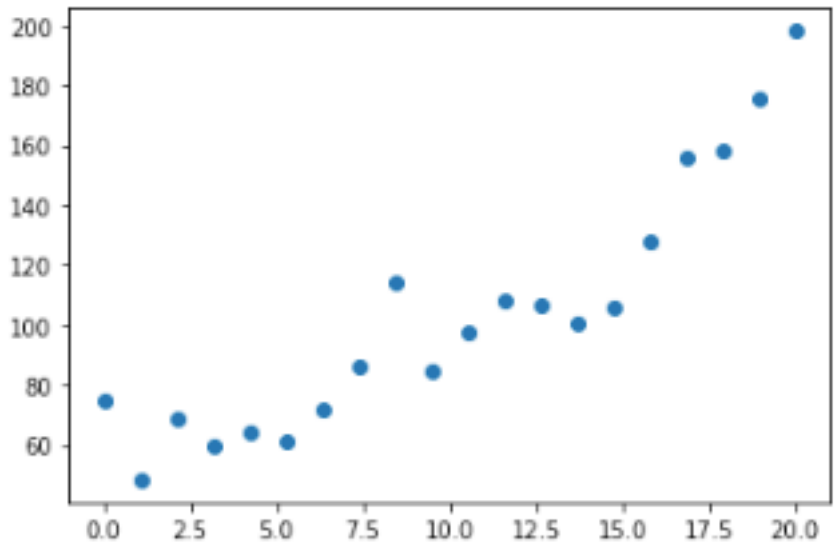
Fitting a Polynomial Function

Import required modules

Two lists (not all elements shown

Scatter plot

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

x = [0.0000000000000000e+00, 1.0526315789473683e+00, 2.1052631578947367e+00,
y = [7.4451929472406007e+01, 4.8348357924118289e+01, 6.8733054363407788e+01,

plt.scatter(x,y)
```



Fitting a Polynomial Function

Convert list to array for numpy operations

3 coefficients, 3 variables

3 coefficients, 1 data structure, 3 elements

```
x = np.array(x)
y = np.array(y)

a, b, c = np.polyfit(x, y, 2)
print(a, b, c)

coeffs = np.polyfit(x, y, 2)
print(coeffs)
```

polyfit: coefficients of polynomial

Convert list to array for numpy operations

3 coefficients, 3 variables

3 coefficients, 1 data structure, 3 elements

```
x = np.array(x)
y = np.array(y)

a, b, c = np.polyfit(x, y, 2)
print(a, b, c)

coeffs = np.polyfit(x, y, 2)
print(coeffs)
```

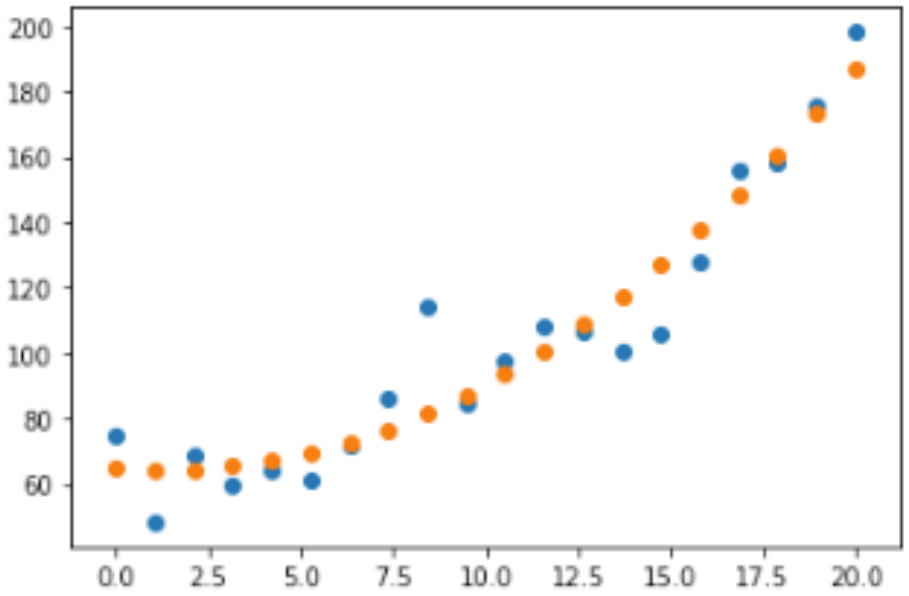
poly1d: generate fitted data

More efficient than typing polynomial equation

Array to fit values to

```
yfit2 = np.poly1d(coeffs)(x)

yfit2 = a*x**2 + b*x + c

yfit2 = coeffs[0]*x**2 + coeffs[1]*x + coeffs[2]
```

```
plt.scatter(x,y)
plt.scatter(x, yfit2)
plt.show()
```



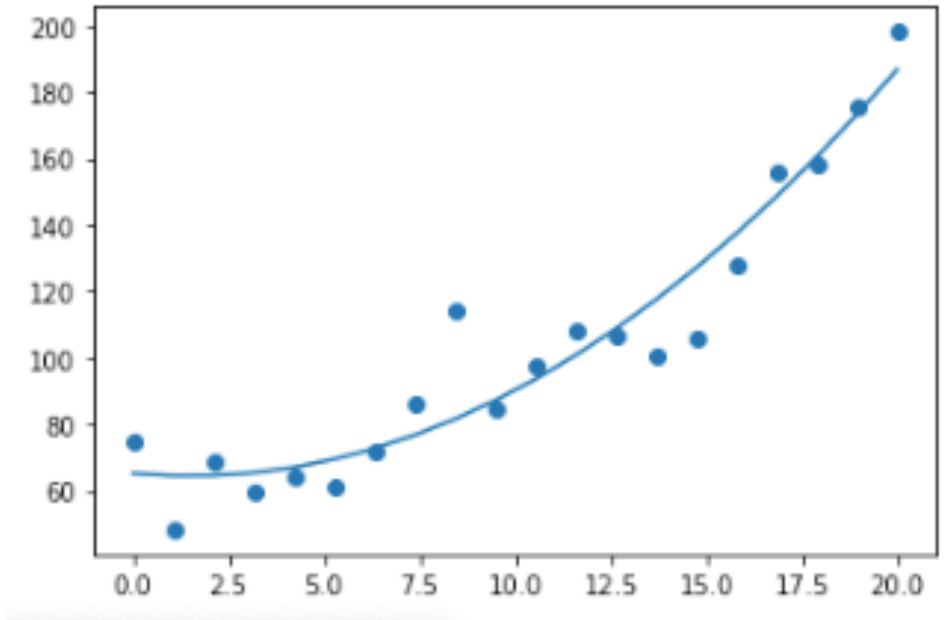
Monotonic x data

```
tmp = sorted(zip(x, y))

x = [t[0] for t in tmp]
y = [t[1] for t in tmp]

x = np.array(x)
y = np.array(y)
```

Sorts monotonically based on first variable

Split list of tuples using list comprehension

```
plt.scatter(x,y)
plt.plot(x, yfit2)
plt.show()
```



Importing data

	A	B	C	D	E	F	G	H	I	J	K	L
1	0	0.19	0.381	0.571	0.762	0.952	1.142	1.333	1.523	1.714	1.904	2.09
2	4.401	7.323	6.319	5.96	5.193	6.06	6.807	4.506	3.917	2.218	1.674	0.4
3												
4												

signal_data.csv

```
0.000,0.190,0.381,0.571,0.762,0.952,1.142,1.333,1.523,1.714,1.904,2.094,2.285,2.475,2.666,2.856,3.046,3.237,3.427,3.6
18,3.808,3.998,4.189,4.379,4.570,4.760,4.950,5.141,5.331,5.522,5.712,5.902,6.093,6.283,6.474,6.664,6.854,7.045,7.235,
7.426,7.616,7.806,7.997,8.187,8.378,8.568,8.758,8.949,9.139,9.330,9.520,9.710,9.901,10.091,10.282,10.472,10.662,10.85
3,11.043,11.234,11.424,11.614,11.805,11.995,12.186,12.376,12.566,12.757,12.947,13.138,13.328,13.518,13.709,13.899,14.
090,14.280,14.470,14.661,14.851,15.042,15.232,15.422,15.613,15.803,15.994,16.184,16.374,16.565,16.755,16.946,17.136,1
7.326,17.517,17.707,17.898,18.088,18.278,18.469,18.659,18.850
4.401,7.323,6.319,5.960,5.193,6.060,6.807,4.506,3.917,2.218,1.674,0.480,-0.570,0.142,-2.336,-3.390,-2.278,-4.094,-4.7
97,-2.316,-2.018,-4.518,-2.534,-1.862,-0.106,-0.564,-0.981,0.584,2.004,2.116,3.333,3.693,5.952,5.939,7.264,7.717,6.44
9,6.122,5.129,6.681,4.489,3.474,3.075,1.303,1.607,0.989,0.498,-2.272,-3.130,-2.539,-3.712,-3.295,-4.378,-4.168,-1.915
,-1.618,-1.228,-1.983,-1.436,0.718,-0.412,2.566,4.379,4.894,5.430,4.539,6.171,5.370,5.215,7.644,7.738,7.397,6.938,3.4
67,3.175,4.830,1.771,2.902,-0.909,-0.576,-1.188,-2.773,-2.411,-2.485,-2.608,-3.430,-4.350,-3.451,-3.857,-2.419,-0.703
,-1.570,1.403,0.429,0.664,3.588,3.451,4.389,4.552,5.074
```

Importing data

File in current directory

Open as read only

```
1  import csv
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  data_path = 'signal_data.csv'
6
7
8  with open(data_path, 'r') as f:
9      reader = csv.reader(f, delimiter=',')
10     data = np.array(list(reader)).astype(float)
11     print(data)
```

Specify delimiter

Convert imported data to numpy array

```
[[ 0.     0.19   0.381  0.571  0.762  0.952  1.142  1.333  1.523  1.714
   1.904  2.094  2.285  2.475  2.666  2.856  3.046  3.237  3.427  3.618
   3.808  3.998  4.189  4.379  4.57   4.76   4.95   5.141  5.331  5.522
   5.712  5.902  6.093  6.283  6.474  6.664  6.854  7.045  7.235  7.426
   7.616  7.806  7.997  8.187  8.378  8.568  8.758  8.949  9.139  9.33
   9.52   9.71   9.901 10.091 10.282 10.472 10.662 10.853 11.043 11.234
  11.424 11.614 11.805 11.995 12.186 12.376 12.566 12.757 12.947 13.138
  13.328 13.518 13.709 13.899 14.09  14.28  14.47  14.661 14.851 15.042
  15.232 15.422 15.613 15.803 15.994 16.184 16.374 16.565 16.755 16.946
  17.136 17.326 17.517 17.707 17.898 18.088 18.278 18.469 18.659 18.85 ]
 [ 4.401  7.323  6.319  5.96   5.193  6.06   6.807  4.506  3.917  2.218
   1.674  0.48  -0.57   0.142 -2.336 -3.39  -2.278 -4.094 -4.797 -2.316
  -2.018 -4.518 -2.534 -1.862 -0.106 -0.564 -0.981  0.584  2.004  2.116
   3.333  3.693  5.952  5.939  7.264  7.717  6.449  6.122  5.129  6.681
   4.489  3.474  3.075  1.303  1.607  0.989  0.498 -2.272 -3.13  -2.539
  -3.712 -3.295 -4.378 -4.168 -1.915 -1.618 -1.228 -1.983 -1.436  0.718
  -0.412  2.566  4.379  4.894  5.43   4.539  6.171  5.37   5.215  7.644
   7.738  7.397  6.938  3.467  3.175  4.83   1.771  2.902 -0.909 -0.576
  -1.188 -2.773 -2.411 -2.485 -2.608 -3.43  -4.35  -3.451 -3.857 -2.419
  -0.703 -1.57   1.403  0.429  0.664  3.588  3.451  4.389  4.552  5.074]]
```

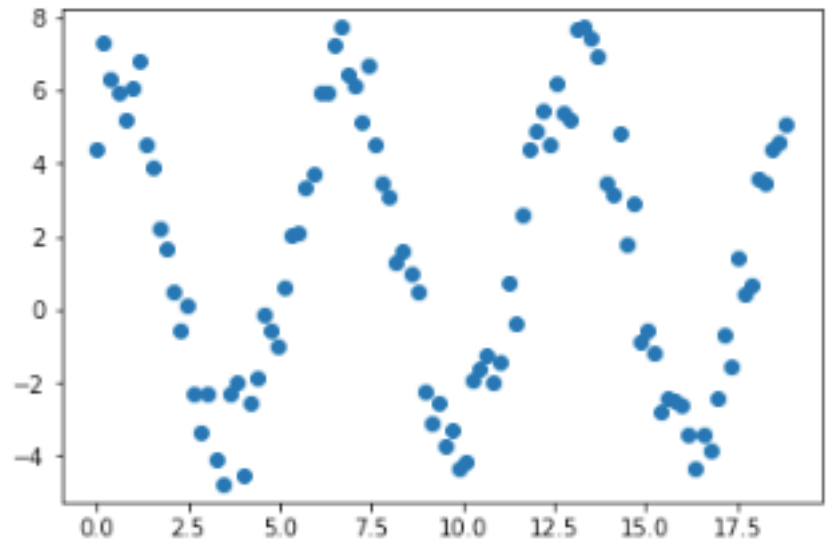
2 rows,
100 columns

Importing data

Open as read only

Select the data of interest

```
1  import csv
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5
6  data_path = 'signal_data.csv'
7  with open(data_path, 'r') as f:
8      reader = csv.reader(f, delimiter=',')
9      data = np.array(list(reader)).astype(float)
10
11
12 x = data[0, :]
13 y = data[1, :]
14
15
16 # Plot the data
17 plt.scatter(x, y)
18 plt.show()
```



curve_fit: fit an arbitrary function

Define function with unknown constants

```
def fit_sin(x, a, b, c, d):        # function name and inputs
    y = a * np.sin(b*(x+c)) + d    # function
    return y                       # output
```

Function from scipy package

```
from scipy.optimize import curve_fit

opt, cov = curve_fit(fit_sin, x, y)


yfit = fit_sin(x, *opt)
```

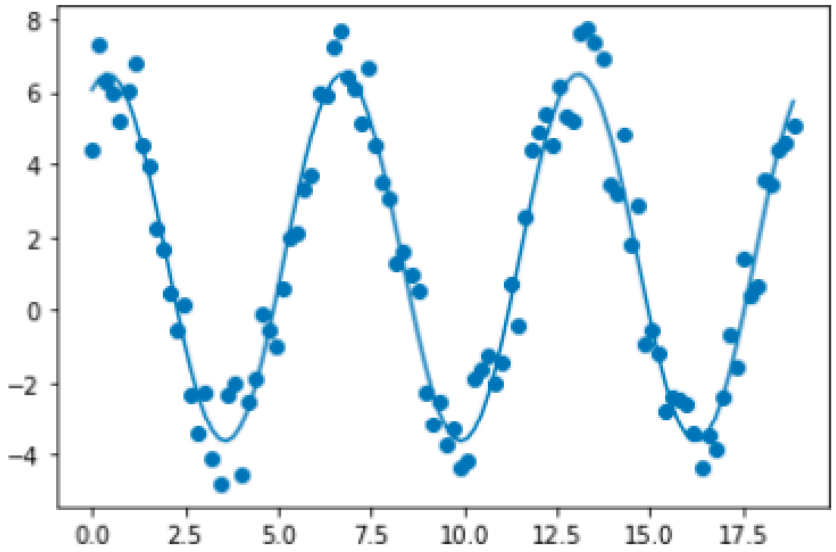
Two returned variables

* ‘unpacks’ the variable (to give 4 variabales in this case)

```
print(opt)
```

```
[5.07111945 0.99283092 1.15948693 1.43651514]
```

```
plt.scatter(x, y)
plt.plot(x, yfit)
plt.show()
```



curve_fit: multiple variables

X is a data structure with two elements x, y

```
def func(X, a, b):
    x,y = X
    z = a*x + b*y**2
    return z

x = np.linspace(0.1, 1.1, 101)
y = np.linspace(1.0, 2.0, 101)
a = 1
b = 2
z = func((x, y), a, b)
plt.scatter(x, z)
z = z + np.random.random(101)
plt.scatter(x, z)

opt, cov = curve_fit(func, (x, y), z)
zfit = func((x, y), *opt)
print(opt)
plt.plot(x, zfit)
```

Generate some randomized data

Fit a curve using

- Independent variables (x, y)
- Dependent variable z