Least-square Solutions Using Numpy

Francois Role
Paris-Descartes University

June 15, 2015

1 Numpy and Least-square solutions

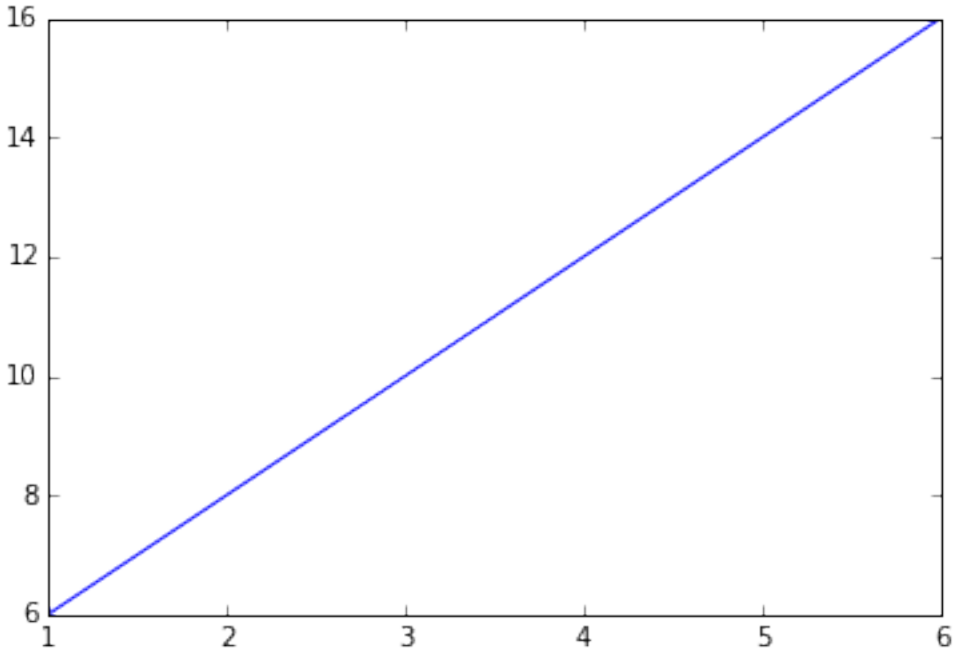
Assume we have the following line, and also assume we are not that clever and can't guess its slope and intercept...

```
In [103]: import matplotlib.pyplot as plt
          import numpy as np

          %matplotlib inline

          x=np.array([1., 2. , 3. , 4. ,5.,6])
          y=np.array([6.,8.,10.,12.,14.,16.])


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



The obvious solution is:

$$coeffs = X^{-1}y$$

where X is:

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \\ 6 & 1 \end{pmatrix}$$

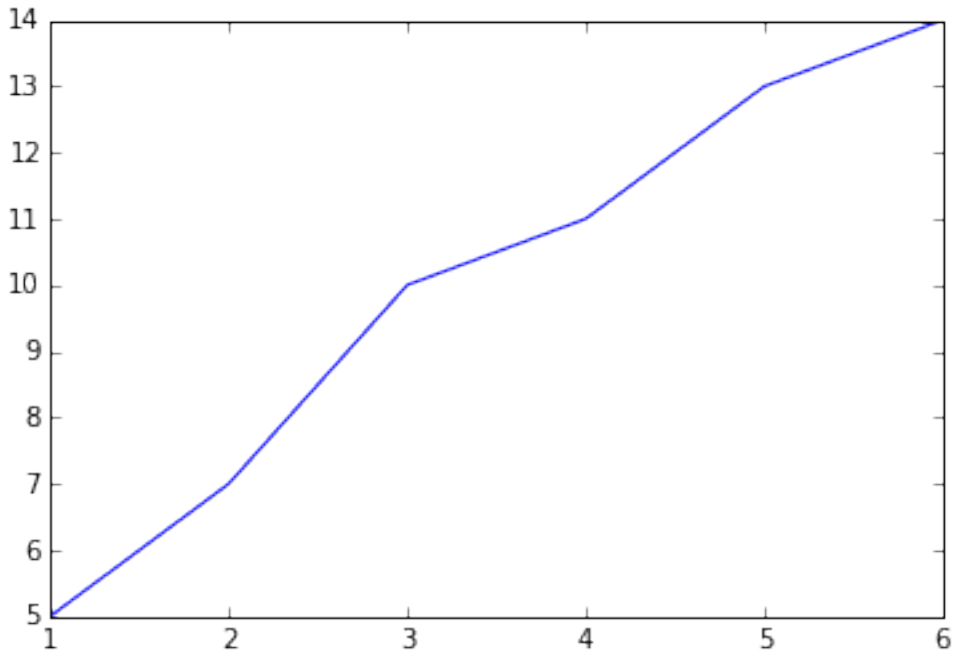
```
In [104]: X=np.mat(  np.vstack((x,np.ones(len(x)))).T)
          y=np.mat(y.reshape(6,1))
          X.I * y

Out[104]: matrix([[ 2.],
                  [ 4.]])
```

Now suppose we have:

```
In [105]: x=np.array([1., 2. , 3. , 4. ,5.,6])
          y=np.array([5.,7.,10.,11.,13.,14.])

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



How do we deal with that?

We can find anapproximate solutions.

1.1 Using covariation

$$slope = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

```
In [106]: x_bar=np.mean(x)
          y_bar=np.mean(y)

          covariation = np.sum( (x-x_bar) * (y - y_bar) )
          x_variation=np.sum((x-x_bar)**2)

          slope=covariation/x_variation
          intercept = y_bar - (slope * x_bar)
          print("Slope = {:.3f} and intercept = {:.3f}".format(slope,intercept) )
```

```
Slope = 1.829 and intercept = 3.600
```

1.2 Using normal equations

```
In [107]: X=np.mat(X) ; y=np.mat(y.reshape(6,1))
          slope_and_intercept = (X.T * X).I * X.T * y
          slope_and_intercept
```

```
Out[107]: matrix([[ 1.82857143],
                  [ 3.6       ]])
```

1.3 Finally, let Numpy do the job ...

```
In [108]: np.linalg.lstsq(X,y)[0]
```

```
Out[108]: matrix([[ 1.82857143],
                  [ 3.6       ]])
```