ols_from_scratch

July 30, 2020

[162]:
```
import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_boston

import statsmodels.api as sm
```

1 Bivariate Linear Regression with Ordinary Least Squares

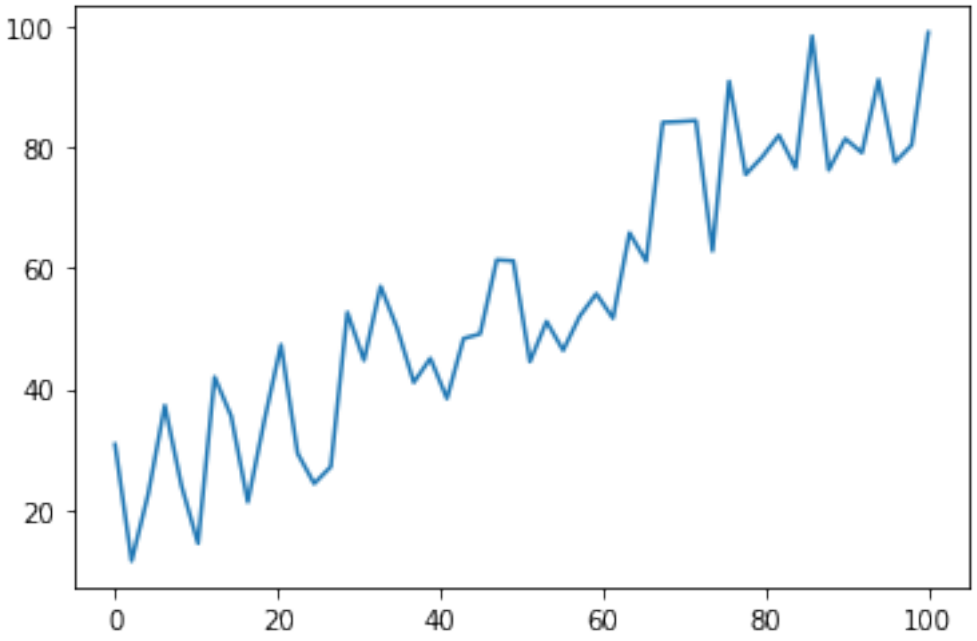
1.1 Create data

[176]:
```
np.random.seed(888)

X = np.linspace(0, 100, 50)
epsilon = np.random.uniform(0, 30, size=(50,))
y = 5 + .75*x  + epsilon

plt.plot(x, y)
```

[176]:
```
[<matplotlib.lines.Line2D at 0x7f2eadca6be0>]
```



1.2 Bivariate Ordinary Least Squares from Minimization

In Ordinary Least Squares, we aim to minimize error or the distance squared between the actual value of y and the predicted value of $\hat{y}$. We will look at solving for β_0 and β_1 as well as $\hat{\beta_0}$ and $\hat{\beta_1}$ We set up the minimization problem as:

$$\min_{\beta_0,\beta_1} \mathbb{E}\big[(Y - \beta_0 - \beta_1 X)^2\big]$$

Which has the empirical analogue of:

$$\min_{\hat{\beta}_0,\hat{\beta}_1} \sum_{i=1}^{n} (y_i - \hat{\beta_0} - \hat{\beta_1} x_i)^2$$

Now taking first order conditions for the expectation we get:

$$\mathbb{E}\big[-2(Y - \beta_0 - \beta_1 X)\big] = 0$$

$$\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X]$$

And β_1 is

$$\mathbb{E}[XY] - \beta_1 \mathbb{E}[X^2]$$

$$\beta_1 = \frac{\mathbb{E}[XY] - \mathbb{E}[Y]\mathbb{E}[X]}{\mathbb{E}[X^2] - \mathbb{E}[X]^2} = \frac{cov(Y,X)}{\mathbb{V}[X]}$$

We take the partial derivatives with respect to be $\hat{\beta_0}$ and $\hat{\beta_1}$ and set them to 0.

$$\frac{\delta W}{\delta \hat{\beta_0}} = \sum_{i=1}^{n} -2(y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$

$$\frac{\delta W}{\delta \hat{\beta_1}} = \sum_{i=1}^{n} -2x_i(y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$

Solving for the equations we end up with the following:

$$\hat{\beta_1} = \frac{\frac{1}{n}\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{n}\sum_{i=1}^{m}(x_i - \bar{x})^2}$$

$$\hat{\beta_0} = \bar{y} - \hat{\beta_1}\bar{x}$$

---

[177]:
```
mean_x = np.mean(X)
mean_y = np.mean(y)

m = len(X)

numer = 0
denom = 0

for i in range(m):
    numer += (X[i] - mean_x) * (y[i] - mean_y)
    denom += (X[i] - mean_x) ** 2

b1 = numer/denom
b0 = mean_y - (b1 * mean_x)

print(b0, b1)
```

```
20.069334201206736 0.7112836798882212
```

Let's check against using Statsmodel built in OLS

[178]:
```
X = sm.add_constant(X)
coeffs_lm = sm.regression.linear_model.OLS(y, X).fit().params
print(coeffs_lm)
```

```
[20.0693342   0.71128368]
```

2 Multivariate Ordinary Least Squares from Linear Algebra

There is a way to derive OLS using linear algebra and matrices. This is perfect for computation and is also very easy on the notation. Our ultimate equation is:

$$\beta = (X^T X)^{-1} X^T y$$

However, let us look at the derivation. We can represent our linear equation as

$$y = X\beta + \epsilon$$

where y is a vector of all dependent values, X is a matrix of all independent observation values, β is a vector of all coefficients, and *epsilon* is a vector of all error terms - the stochastic component. Let us rearrange in terms of the error:

$$e = y - X\hat{\beta}$$

Thus the sum of squared residuals (RSS) is $e^T e$. Substituting in the previous equation we get:

$$y^T y - 2\hat{\beta}^T X^T y + \hat{\beta}^T X^T X \hat{\beta}$$

Now to minimize the RSS, we take its partial derivative in respect to $\hat{\beta}$

$$\frac{\delta e^T e}{\hat{\beta}} = -2X^T y + 2X^T X\hat{\beta} = 0$$

Solving we get the first equation.

[179]:
```
df = load_boston()
X = df.data
y = df.target

feature_names = df.feature_names
print(feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

[180]:
```
int = np.ones(shape=y.shape)[...,None]
X = np.concatenate((int, X), 1)
```

[181]:
```
coeffs = np.linalg.inv(X.transpose().dot(X)).dot(X.transpose()).dot(y)
```

[182]:
```
feature_names = np.insert(df.feature_names, 0, 'INT')
results = pd.DataFrame({'coeffs':coeffs}, index=feature_names)
print(results)
```

```
            coeffs
INT      36.459488
CRIM     -0.108011
ZN        0.046420
INDUS     0.020559
```