

# m2-l2-p2

February 3, 2024

## 1 M2-L2 Problem 2 (5 points)

In this problem we will perform least-squares regression using sklearn's built-in tools.

First, you will generate a standard linear least squares regression model with `LinearRegression`.

Next, you will use stochastic gradient descent to train another model with `SGDRegressor`.

Run this cell to perform the required imports and load the data:

```
[1]: import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor

def plot_data_with_regression(x_data, y_data, x_reg, y_reg, title=""):
    plt.figure()

    plt.scatter(x_data.flatten(), y_data.flatten(), label="Data", c="black")
    plt.plot(x_reg.flatten(), y_reg.flatten(), label="Fit")

    plt.legend()
    plt.xlabel(r"$x_1$")
    plt.ylabel(r"$y$")
    plt.xlim(-2,2)
    plt.ylim(-2,2)
    plt.title(title)
    plt.show()

x = np.array([-1.52362349, -1.60576489, -1.34827768, -1.45340266, -1.42652973,
↪ -1.20489448, -1.34662672, -0.77989399, -1.02479866, -1.18388425, -0.63601103,
↪ -1.06184469, -0.81853545, -0.98253754, -0.41599291, -0.39373095, -0.49828026,
↪ -0.50820654, -0.33545205, -0.17745781, -0.11232523, -0.1065807, -0.28129177,
↪ -0.27996129, -0.30355288, 0.32030413, -0.10195465, 0.44533511, 0.43274828,
↪ 0.35343658, 0.10612364, 0.22594603, 0.72121097, 0.81574553, 0.50008731, 0.
↪ 61311114, 0.65468014, 0.60571755, 0.67438719, 0.67226667, 0.82554208, 0.
↪ 99744723, 1.04085817, 1.26282573, 1.14517384, 1.22017733, 1.28481911, 1.
↪ 29034024, 1.66304968, 1.49217068])
```

```

y = np.array([ 1.65517515,  1.33249684,  1.38328432,  1.1531808 ,  0.89478436,0.
↪66783182,  0.59841012,  0.64060691,  0.73930904,  0.91751879,0.52952715,  0.
↪30183071,  0.31052792,  0.3684916 ,  0.6584406 , 0.72407787,  0.83710857,  0.
↪74432213,  0.80478856,  0.6755436 ,0.47643214,  0.50833171,  0.7731494 ,  0.
↪60172717,  0.39123432, 0.69418038,  0.59625493,  0.90711162,  1.01760262,  0.
↪98348453, 0.78208567,  0.76030704,  0.55105897,  0.7830408 ,  0.86284938, 0.
↪88284709,  0.35630757,  0.7237893 ,  0.66325283,  0.23944077,-0.04822037,  0.
↪16059611, -0.14460941, -0.38449511, -0.54652704, -0.85011695, -0.84597649,↪
↪-0.90989068, -1.06840649, -1.77916364])
X = np.vstack([x*x*x, x*x, x, np.ones_like(x)]).T

xreg = np.linspace(-2,2)
Xreg = np.vstack([xreg*xreg*xreg, xreg*xreg, xreg, np.ones_like(xreg)]).T

```

## 1.1 Least Squares Regression

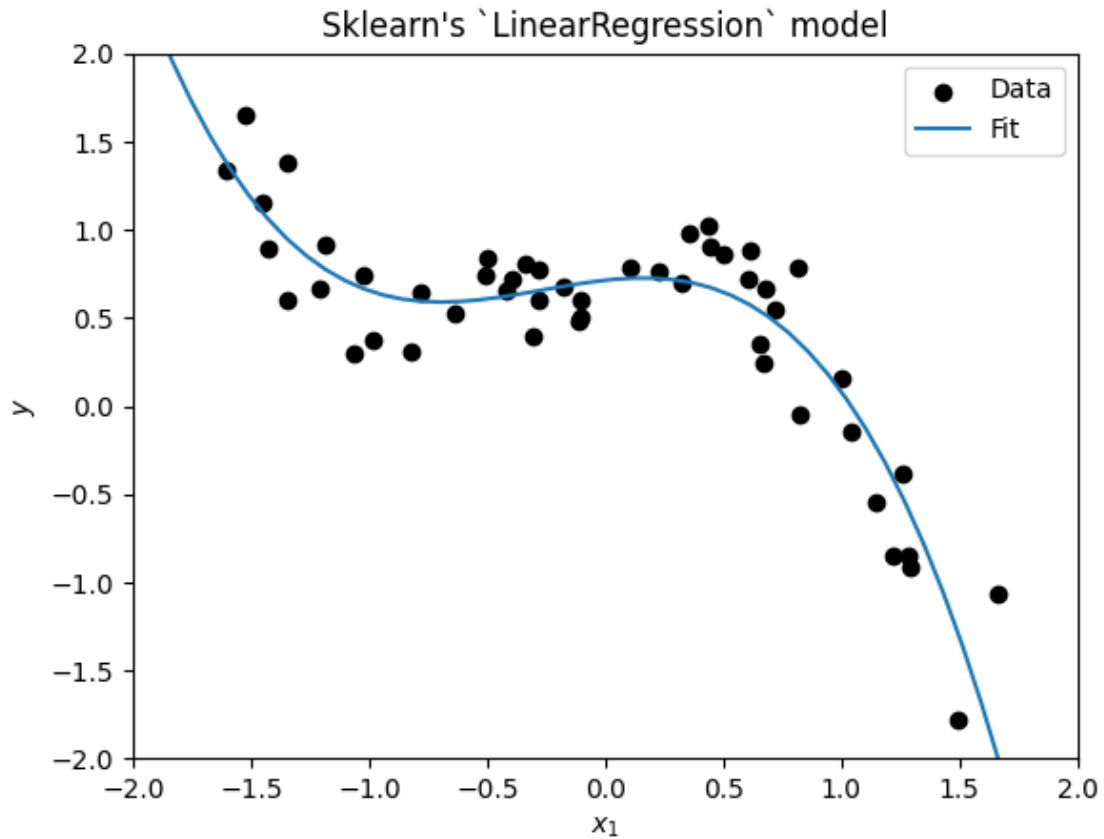
We have provided a demonstration of least squares regression using sklearn:

```

[2]: model = LinearRegression()
model.fit(X,y)

yreg = model.predict(Xreg)
plot_data_with_regression(x, y, xreg, yreg, "Sklearn's `LinearRegression`↪
↪model")

```



## 1.2 Using SGD

Now use stochastic gradient descent to solve the same problem and make a similar plot, but for a `SGDRegressor` model instead of `LinearRegression`:

```
[3]: # YOUR CODE GOES HERE
model = SGDRegressor()
model.fit(X,y)

yreg = model.predict(Xreg)
plot_data_with_regression(x, y, xreg, yreg, "Sklearn's `SGDRegressor` model")
```

