Computational methods

Assignment 2: Regression

Rasmus Hammar

Imports

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

# For displaying nice tables in quarto.
from IPython.display import Markdown
from tabulate import tabulate
```

Data

```
# Load data
data_WHO = np.genfromtxt(
    # "Computational_methods\\Lab_2\\Assignment2\\LifeExpectancyData.csv",
    "LifeExpectancyData.csv",
    delimiter=",",
    usecols=(3, 21),
    skip_header=1,
)
# Clean data (remove NANs)
data_clean = data_WHO[~np.isnan(data_WHO).any(axis=1)]
```

```
Markdown(tabulate(data_clean[:5], headers=["Life expectancy", "Schooling"]))
```

Table 1: Preview of life expectancy and years of schooling data from WHO.

Life expectancy	Schooling
65	10.1
59.9	10
59.9	9.9
59.5	9.8
59.2	9.5

Plot data:

```
plt.figure(1)
plt.scatter(data_clean[:, 1], data_clean[:, 0], c="blue", s=1)
plt.xlabel("Schooling (years)")
plt.ylabel("Life expectancy (years)")
plt.show()
```

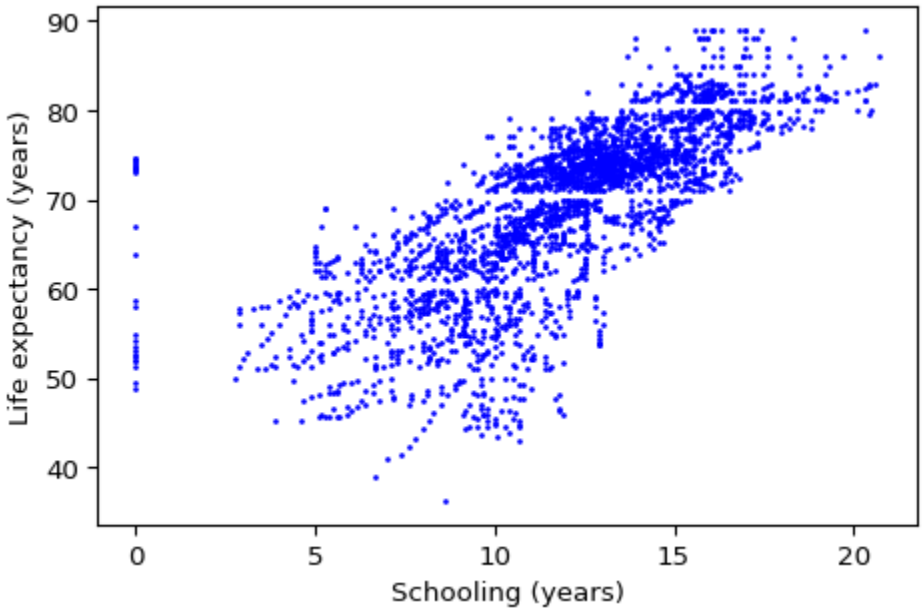

Figure 1: Schooling vs life expectancy.

Regression

Polynomial fits of data.

```
# Linear fit
p1 = np.polynomial.Polynomial.fit(data_clean[:, 1], data_clean[:, 0], deg=1)
print(f"Linear fit = {p1}")

# 2nd degree fit
p2 = np.polynomial.Polynomial.fit(data_clean[:, 1], data_clean[:, 0], deg=2)
print(f"2nd degree fit = {p2}")
```

```
Linear fit = 65.87962877 + 21.77073964 (-1.0 + 0.09661836x)
2nd degree fit = 65.05055294 + 20.782527 (-1.0 + 0.09661836x) +
7.59454959 (-1.0 + 0.09661836x)**2
```

Show fits with data:

```
# Smooth x-values for plotting fits
x_smooth = np.linspace(np.min(data_clean[:, 1]), np.max(data_clean[:, 1]),
200)
# y values for lines
y_p1 = p1(x_smooth)
y_p2 = p2(x_smooth)
```

```
plt.figure(2)
plt.scatter(data_clean[:, 1], data_clean[:, 0], c="blue", s=1, label="Data")
plt.plot(x_smooth, y_p1, c="magenta", label="Linear fit")
plt.plot(x_smooth, y_p2, c="green", label="2nd degree fit")
plt.xlabel("Schooling (years)")
plt.ylabel("Life expectancy (years)")
plt.legend()
plt.show()
```

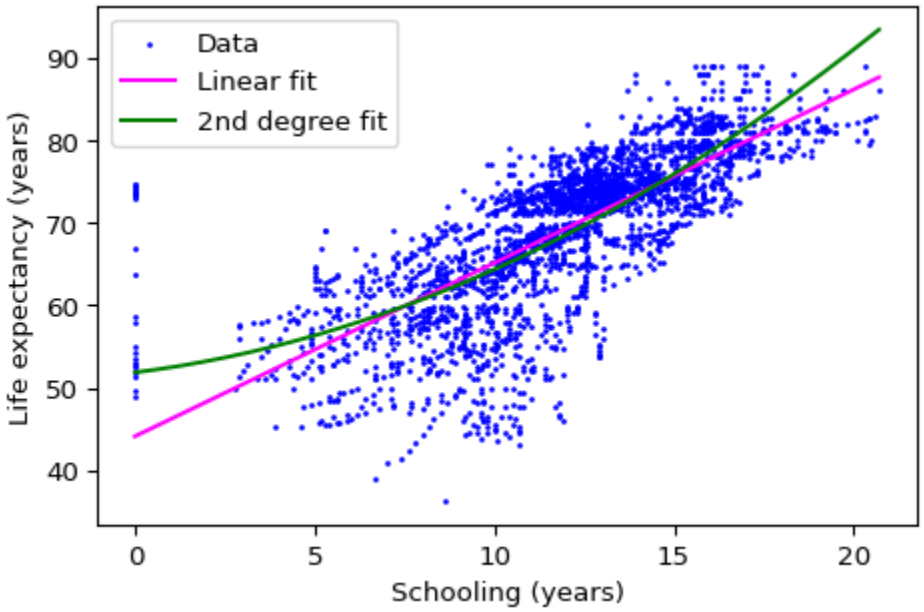

Figure 2: Regression using least squares method on life expectancy depending on schooling.

Condition number

Condition number for 1st and 2nd degree polynomials.

```
# Linear fit condition number
A1 = np.ones((len(data_clean[:, 1]), 2))
A1[:, 1] = data_clean[:, 1]
A1TA1 = A1.T @ A1
cond1 = np.linalg.cond(A1TA1)
```

```
print(f"Condition number for linear fit: {cond1}")
print(f"Loss of up to {round(math.log10(cond1))} digits of significance.")
print("-----------------------------")
# 2nd degree fit condition number
A2 = np.ones((len(data_clean[:, 1]), 3))
A2[:, 1] = data_clean[:, 1]
A2[:, 2] = data_clean[:, 1] ** 2
A2TA2 = A2.T @ A2
cond2 = np.linalg.cond(A2TA2)
print(f"Condition number for 2nd degree fit: {cond2}")
print(f"Loss of up to {round(math.log10(cond2))} digits of significance.")
```

```
Condition number for linear fit: 2177.0870489861745
Loss of up to 3 digits of significance.
-----------------------------
Condition number for 2nd degree fit: 1607367.7520432896
Loss of up to 6 digits of significance.
```

Linear fit has a smaller condition number, indicating it's a better fit with less error than the 2nd degree fit.

Residuals

Checking distribution of residuals for both fits using $r = y - A\hat{x}$.

```
resid1 = data_clean[:, 0] - p1(data_clean[:, 1])
resid2 = data_clean[:, 0] - p2(data_clean[:, 1])
```

```
plt.figure(3)
plt.subplot(2, 1, 1)
plt.hist(resid1, bins=200, color="magenta")
plt.title("Residuals for linear fit")
plt.subplot(2, 1, 2)
plt.hist(resid2, bins=200, color="green")
plt.title("Residuals for 2nd degree fit")
plt.tight_layout()
plt.show()
```

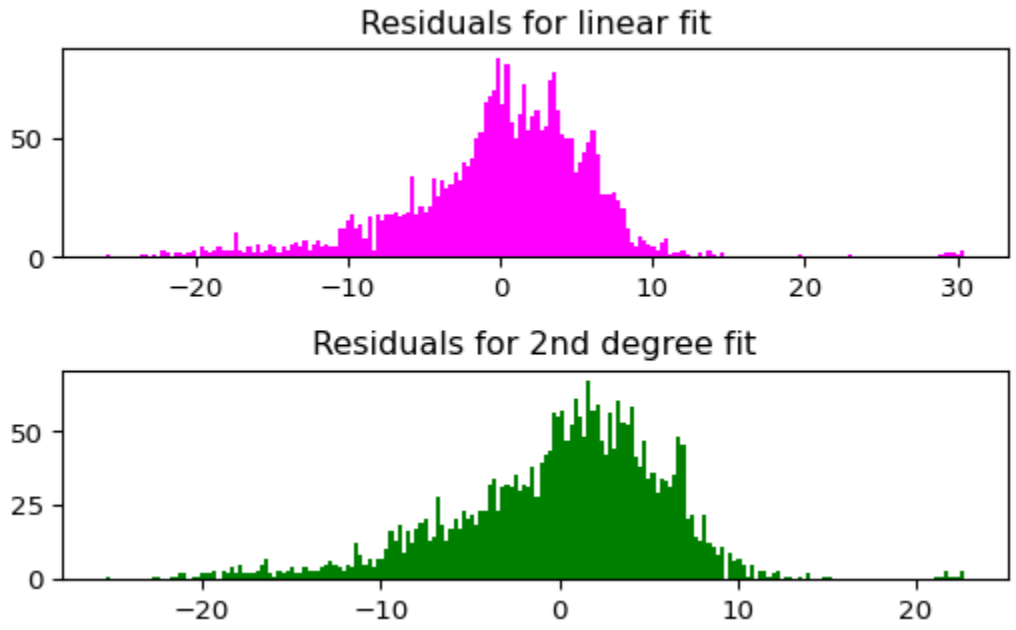

Figure 3: Residuals for linear and 2nd degree polynomial fits.

The residuals look pretty normally distributed for both fits.