# lwr-fl

April 11, 2025

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Data
x = np.array([5, 1, 2, 1])
y = np.array([25, 5, 7, 8])
x0 = 2  # The point where we want to make a prediction
tau = 0.4  # Bandwidth
```

[3]:
```
# Design matrix (adding intercept column)
X = np.column_stack((np.ones(len(x)), x))
```

[4]:
```
# Calculate weights for each data point
weights = np.exp(-((x - x0) ** 2) / (2 * tau ** 2))
```

[5]:
```
# Weight matrix W (diagonal matrix of weights)
W = np.diag(weights)
```

[6]:
```
# Compute the weighted least squares estimate of beta using pseudoinverse
X_T_W = np.dot(X.T, W)
X_T_W_X_inv = np.linalg.pinv(np.dot(X_T_W, X))  # Using pinv for stability
```

[7]:
```
beta_hat = np.dot(np.dot(X_T_W_X_inv, X_T_W), y)
print("Estimated coefficients (beta):", beta_hat)
```

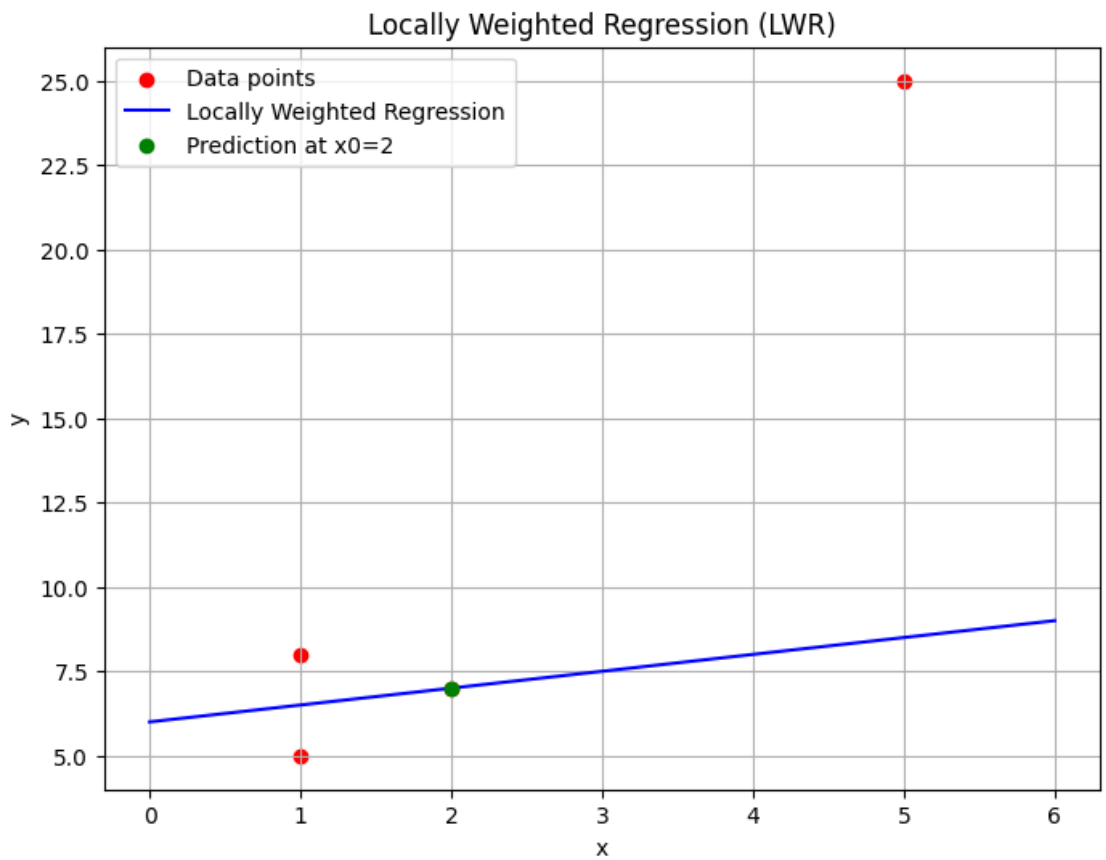
```
Estimated coefficients (beta): [6.  0.5]
```

[8]:
```
# Predicted value at x0
y0 = np.dot(np.array([1, x0]), beta_hat)
print(f"Predicted value at x0 = {x0}: {y0}")
```

```
Predicted value at x0 = 2: 7.000000000040318
```

[10]:
```
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='red', label='Data points')
x_range = np.linspace(min(x) - 1, max(x) + 1, 100)
y_range = np.array([np.dot(np.array([1, xi]), beta_hat) for xi in x_range])
plt.plot(x_range, y_range, color='blue', label='Locally Weighted Regression')
```

```
plt.scatter(x0, y0, color='green', label=f'Prediction at x0={x0}', zorder=3)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Locally Weighted Regression (LWR)')
plt.legend()
plt.grid(True)
plt.show()
```




[ ]:

```
```