Weighted Least-Squares

Let $\mathbf{y}$ be a vector of n observations, $y_1, \ldots, y_n$, and suppose we wish to approximate $\mathbf{y}$ by a vector $\hat{\mathbf{y}}$ that belongs to some specified subspace of $\mathbb{R}^n$: $A\mathbf{x} = \hat{\mathbf{y}}, \hat{\mathbf{y}} \in Col(A)$. Denote the entries in $\hat{\mathbf{y}}$ by $\hat{y}_1, \ldots, \hat{y}_n$.

Then the *sum of the squares for error (SSE)* in approximating $\mathbf{y}$ by $\hat{\mathbf{y}}$ is

$$SSE = (y_1 - \hat{y}_1)^2 + \ldots + (y_n - \hat{y}_n)^2 = \|\mathbf{y} - \hat{\mathbf{y}}\|^2$$

Now suppose the measurements that produced the entries in $\mathbf{y}$ are not equally reliable. The entries in $\mathbf{y}$ might be computed from various samples of measurements with unequal sample sizes. Then it becomes appropriate to weight the squared errors in SSE in such a way that more importance is assigned to the more reliable measurements. If the weights are denoted by $w_1^2, \ldots, w_n^2$, then the weighted sum of the squares for error is

$$\text{Weighted } SSE = w_1^2(y_1 - \hat{y}_1)^2 + \ldots + w_n^2(y_n - \hat{y}_n)^2$$

Suppose the errors in measuring the $y_i (i = 1, \ldots, n)$ are independent random variables with means equal to zero and variances of $\sigma_1^2, \ldots, \sigma_n^2$. Then the appropriate weights are $w_i^2 = \frac{1}{\sigma_i^2}$. The larger the variance of the error, the smaller the weight.

It is sometimes convenient to transform a weighted least-squares problem into an equivalent ordinary least-squares problems. Let W be the diagonal matrix with (positive) $w_1, \ldots, w_n$ on its diagonal, so that

$$W\mathbf{y} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} w_1 y_1 \\ w_2 y_2 \\ \vdots \\ w_n y_n \end{bmatrix}$$

with a similar expression for $W\hat{\mathbf{y}}$.
Observe that $w_j^2(y_j - \hat{y_j})^2$ can be written as $(w_j y_j - w_j \hat{y_j})^2$. It follows that Weighted $SSE = \|W\mathbf{y} - W\hat{\mathbf{y}}\|^2$.

Now suppose the approximating vector $\hat{\mathbf{y}}$ is to be constructed from the columns of a matrix A. Then we seek an $\hat{\mathbf{x}}$ that makes $A\hat{\mathbf{x}} = \hat{\mathbf{y}}$ as close to $\mathbf{y}$ as possible. However, the measure of closeness is the weighted error,

$$\|W\mathbf{y} - W\hat{\mathbf{y}}\|^2 = \|W\mathbf{y} - WA\hat{\mathbf{x}}\|^2$$

Thus $\hat{\mathbf{x}}$ is the (ordinary) least-squares solution of the equation

$$WA\mathbf{x} = W\mathbf{y}$$

The normal equation for the least-squares solution is

$$(WA)^T WA\mathbf{x} = (WA)^T W\mathbf{y}$$

Example 1: Find the least-squares line $y = \beta_0 + \beta_1 x$ that best fits the data $(-2, 3), (-1, 5), (0, 5), (1, 4)$, and $(2, 3)$. Suppose the errors in measuring the y-values of the last two data points are greater than for the other points. Weight these data half as much as the rest of the data.

Solution: Let us write X for the matrix A and $\boldsymbol{\beta}$ for the vector $\mathbf{x}$, and obtain

$$X = \begin{bmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 3 \\ 5 \\ 5 \\ 4 \\ 3 \end{bmatrix}$$

The weighting matrix is $W = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$. Left multiplication by W scales the rows of X and $\mathbf{y}$:

$$WX = \begin{bmatrix} 2 & -4 \\ 2 & -2 \\ 2 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad W\mathbf{y} = \begin{bmatrix} 6 \\ 10 \\ 10 \\ 4 \\ 3 \end{bmatrix}$$

For the normal equation, compute $(WX)^T WX = \begin{bmatrix} 14 & -9 \\ -9 & 25 \end{bmatrix}$ and $(WX)^T W\mathbf{y} = \begin{bmatrix} 59 \\ -34 \end{bmatrix}$ and solve

$$\begin{bmatrix} 14 & -9 \\ -9 & 25 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} 59 \\ -34 \end{bmatrix}$$

The solution of the normal equation is (to two significant digits) $\beta_0 = 4.3$ and $\beta_1 = 0.20$. The desired line is

$$y = 4.3 + 0.20x$$

In contrast, the ordinary least-squares line for these data is

$$y = 4.0 - 0.10x$$

Example 2:

1. To measure the takeoff performance of an airplane, the horizontal position of the plane was measured every second, from $t = 0$ to $t = 12$. The positions (in feet) were: 0, 9.8, 29.9, 62.0, 104.7, 159.1, 222.0, 294.5, 380.4, 471.1, 571.7,686.8, and 809.2.

(a) Find the least-squares cubic curve $y = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3$ for these data.

(b) Estimate the velocity of the plane when $t = 4.5$ seconds?

2. Refer to the data in the previous example concerning the takeoff performance of an airplane. Suppose the possible measurement errors become greater as the speed of the airplane increases, and let W be the diagonal weighting matrix whose diagonal entries are 1,1,1,0.9, 0.9, 0.8, 0.7,0.6, 0.5, 0.4, 0.3, 0.2, and 0.1. Find the cubic curve that fits the data with minimum weighted least-squares error, and use it to estimate the velocity of the plane when $t = 4.5$ seconds.

Listing 1: 2D Weighted Least Squares and Polynomial fitting

```
# Python code for Airplane Takeoff Performance
import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt

# Data: Time (t) and Positions (y)
t = np.arange(13)
y = np.array([0, 9.8, 29.9, 62.0, 104.7, 159.1, 222.0, 294.5, 380.4, 471.1, 571.

# Part (a): Ordinary Least Squares for Cubic Fit
X = np.vstack([np.ones_like(t), t, t**2, t**3]).T
beta = inv(X.T @ X) @ X.T @ y

# Part (b): Estimate Velocity at t=4.5
velocity = beta[1] + 2 * beta[2] * 4.5 + 3 * beta[3] * (4.5**2)
print(f"Velocity at t=4.5 seconds (OLS): {velocity:.4f} ft/s")

# Part (c): Weighted Least Squares Fit
weights = np.array([1,1,1,0.9,0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1])
W = np.diag(weights)
beta_w = inv(X.T @ W @ X) @ X.T @ W @ y

# Estimate Velocity with Weighted Fit
target_t = 4.5
velocity_w = beta_w[1] + 2 * beta_w[2] * target_t + 3 * beta_w[3] * (target_t**2
print(f"Velocity at t=4.5 seconds (Weighted): {velocity_w:.4f} ft/s")
```