Problem - 1

Training data $\rightarrow (x_1, y_1), \ldots (x_n, y_n) \quad x_i \in \mathbb{R}^d$

$$f_\beta(x) = \langle \beta, x \rangle$$

$$L_{wss}(\beta) = \sum_{i=1}^{n} w_i \left(\langle \beta, x_i \rangle - y_i \right)^2$$

(a) Gradient of $L_{wss}(\beta)$

$$\frac{\partial}{\partial \beta} L_{wss}(\beta) = \frac{\partial}{\partial \beta} \sum_{i=1}^{n} \left(w_i (\langle \beta, x_i \rangle - y_i)^2 \right)$$

$$= \sum_{i=1}^{n} \frac{\partial}{\partial \beta} \left[w_i (\langle \beta, x_i \rangle - y_i)^2 \right]$$

$$= \sum_{i=1}^{n} 2 w_i (\langle \beta, x_i \rangle - y_i) x_i$$

$$\therefore \boxed{\nabla_\beta L_{wss}(\beta) = 2 \sum_{i=1}^{n} w_i \left(\langle \beta, x_i \rangle - y_i \right) x_i}$$

(b) closed form minimizer β^*

$$\nabla_\beta L_{wss}(\beta) = 0$$

$$2\sum_{i=1}^{n} w_i \left(\langle \beta^*, x_i \rangle - y_i \right) x_{ij} = 0 \quad \forall\, j = 1, 2, \cdots d$$

$$\sum_{i=1}^{n} w_i \left(\langle \beta^*, x_i \rangle - y_i \right) x_{ij} = 0$$

$$\Rightarrow \sum_{i=1}^{n} w_i \left(\langle \beta^*, x_i \rangle \right) x_{ij} = \sum_{i=1}^{n} w_i x_{ij} y_i \quad (j = 1, \cdots d)$$

$$\Rightarrow \sum_{i=1}^{n} w_i x_{ij} x_i^T \beta^* = \sum_{i=1}^{n} w_i x_{ij} y_i \quad (j = 1, \cdots d)$$

$$\Rightarrow (X^T W X) \beta^* = X^T W y.$$

$$\Rightarrow \boxed{\beta^* = (X^T W X)^{-1} X^T W y}$$

Problem - 2

(a) Minimize $L(m) = \sum_{i=1}^{n} (y_i - m)^2$

$$L(m) = \sum_{i=1}^{n} \left((y_i - \bar{y}) + (\bar{y} - m) \right)^2$$

$$= \sum_{i=1}^{n} (y_i - \bar{y})^2 + 2(\bar{y} - m) \sum_{i=1}^{n} (y_i - \bar{y}) + n(\bar{y} - m)^2$$

$$\sum_{i=1}^{n} (y_i - \bar{y}) = 0$$

$$\Rightarrow L(m) = \underbrace{\sum_{i=1}^{n} (y_i - \bar{y})^2}_{\text{Constant w.r.t } m} + \underbrace{n(\bar{y} - m)^2}_{\text{Need to minimize with } m}$$

$\therefore$ $L(m)$ is minimized at $m = \bar{y}$

$\therefore$ $\boxed{m = \bar{y}}$ is the unique minimizer

(b) Minimize $L(m) = \max_i |y_i - m|$

Let $y_{(1)} = \min_i(y_i)$ & $y_{(n)} = \max_i y_i$

claim: A minimizer is $m^* = \frac{y_{(1)} + y_{(n)}}{2}$ → midrange

If $m < y_{(1)}$ → the worst error is $|y_{(n)} - m| = y_{(n)} - m$ which strictly decreases as you move m right toward $y_{(1)}$

Similarly for $m > y_{(n)}$ moving left. So any minimizer lies in $(y_{(1)}, y_{(n)})$

For any $m \in [y_{(1)}, y_{(n)}]$ the worst error comes from the extremes.

$$\therefore L(m) = \max\{m - y_{(1)}, y_{(n)} - m\}.$$

The maximum is smallest when the two terms are equal.

$$m - y_{(1)} = y_{(n)} - m \Rightarrow m = \frac{(y_{(1)} + y_{(n)})}{2}$$

(c) $L(m) = \sum_{i=1}^{n} |y_i - m|$ → median

Let the data be sorted → $y_{(1)} \le y_{(2)} \le \cdots \le y_{(n)}$

Let consider pairs around the center.

for $i = 1, 2, \cdots \lfloor n/2 \rfloor$, pair $(y_{(i)}, y_{(n-i+1)})$

Let $g_i(m) = |y_{(i)} - m| + |y_{(n-i+1)} - m|$

If $m \le y_{(i)}$: $g_i(m) = (y_{(i)} - m) + (y_{(n-i+1)} - m) = y_{(i)} + y_{(n-i+1)} - 2m$

$g_i(m)$ decreases as m increases

If $m \ge y_{(n-i+1)}$: $g_i(m) = (m - y_{(i)}) + (m - y_{(n-i+1)}) = 2m - (y_{(i)} + y_{(n-i+1)})$

$g_i(m)$ increases a m increases

If $y_{(i)} \le m \le y_{(n-i+1)}$: $g_i(m) = (m - y_{(i)}) + (y_{(n-i+1)} - m) = y_{(n-i+1)} - y_{(i)}$

$g_i(m)$ → constant (& it's minimum)

$L(m) = \sum_i g_i(m)$ is minimized when m lies in the intersection of all these minimizing intervals.

$$\bigcap_{i=1}^{\lfloor n/2 \rfloor} \left[y_{(i)}, y_{(n-i+1)} \right]$$

If n is odd, the intersection collapses to the single point $y_{(n+1)/2}$ → the middle value (median)

⇒ Unique minimizer: $m^* = y_{(n+1)/2}$

If n is even, the intersection is the interval $\left(y_{(n/2)}, y_{(n/2+1)} \right)$

⇒ Every m^* in this interval minimizes $L(m)$

∴ Any median minimizes $\sum_i |y_i - m|$

Hence proven

Problem-3

(a) $$f(x) = \begin{cases} a_1 + s_1 x_i : x_i < \lambda \\ a_2 + s_2 x_i : x_i \geq \lambda \end{cases} \qquad a_1 + s_1\lambda = a_2 + s_2\lambda$$

$$a_2 + s_2\lambda = a_1 + s_1\lambda \Rightarrow a_2 = a_1 + (s_1 - s_2)\lambda$$

for $x_i \geq \lambda$

$$f(x) = a_2 + s_2 x_i = a_1 + (s_1 - s_2)\lambda + s_2 x_i$$

$$= \underbrace{a_1 + s_1\lambda - s_2\lambda}_{\text{new intercept}} + s_2 x_i$$

Thus the model is equivalent to

$$f(x) = \begin{cases} a_1 + s_1 x_i & : x_i < \lambda \\ a_1 + s_1\lambda - s_2\lambda + s_2 x_i & : x_i \geq \lambda \end{cases}$$

(b) $h_\lambda(x) = (x-\lambda)_+ = \max\{0, x-\lambda\}$

$$f(x) = \beta_0 + \beta_1 x + \beta_2 h_\lambda(x).$$

If $x < \lambda \rightarrow h_\lambda(x) = 0$

$$f(x) = \beta_0 + \beta_1 x \Rightarrow a_1 = \beta_0 \quad s_1 = \beta_1$$

If $x \geq \lambda \rightarrow h_\lambda(x) = x - \lambda$

$$f(x) = \beta_0 + \beta_1 x + \beta_2 (x-\lambda) = (\beta_0 - \beta_2\lambda) + (\beta_1 + \beta_2) x$$

$$f(x) = (\beta_0 - \beta_2\lambda) + (\beta_1 + \beta_2) x$$

$$\Rightarrow s_2 = \beta_1 + \beta_2 \quad \& \quad a_1 = \beta_0$$

Let $X \in \mathbb{R}^{n \times 3} \Rightarrow x_i = [1, x_i, h_\lambda(x_i)]$

& target vector $y = (y_1, \dots, y_n)^T$

$$\hat{\beta} = \arg\min_{\beta} \| y - X\beta \|_2^2$$

$\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2) \Rightarrow$ $\boxed{\hat{a}_1 = \hat{\beta}_0}$ $\boxed{\hat{s}_1 = \hat{\beta}_1}$ $\boxed{\hat{s}_2 = \hat{\beta}_1 + \hat{\beta}_2}$

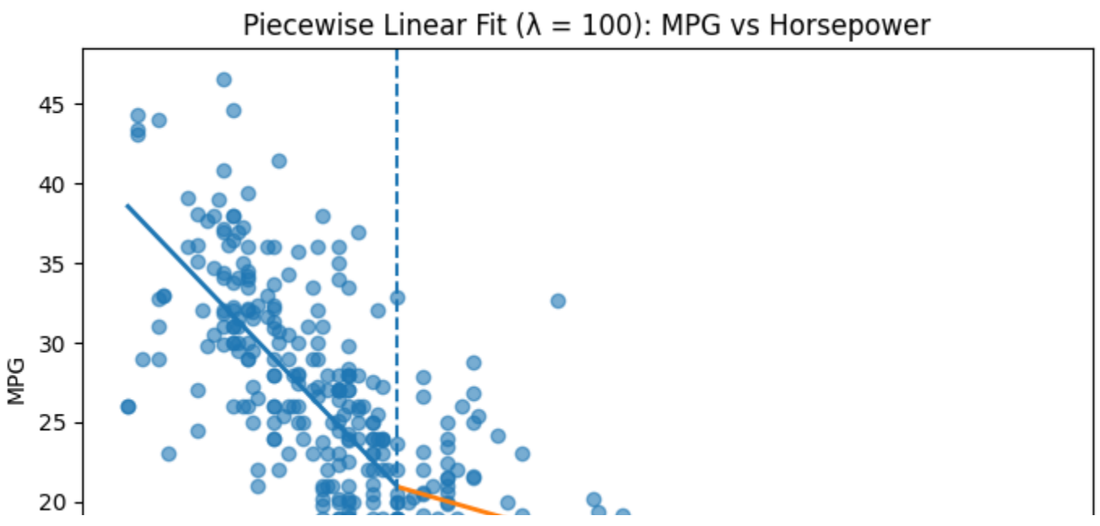
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path

csv_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'
df = pd.read_csv(
    csv_path, delim_whitespace=True,
    names=['mpg', 'cylinders', 'displacement',
           'horsepower', 'weight', 'acceleration',
           'model year', 'origin', 'car name'])

print(df.shape)
df.head()
```

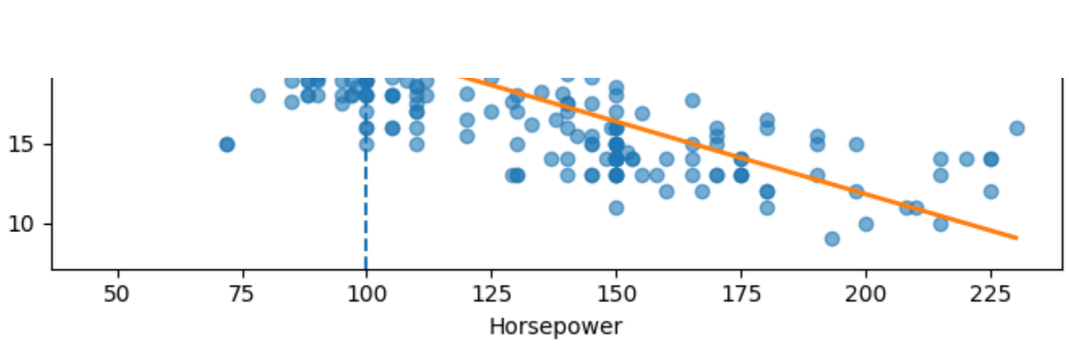
```
/tmp/ipython-input-3835944077.py:7: FutureWarning: The 'delim_whitespace' keyword in pd.read_csv is deprecated and will be removed in a future version. Use ``sep='\s
  df = pd.read_csv(csv_path, delim_whitespace=True, names=['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car n
(398, 9)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

```
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')

df = df[['mpg', 'horsepower']].dropna().copy()

df = df[df['horsepower'].apply(lambda v: np.isfinite(v))]
df = df[df['mpg'].apply(lambda v: np.isfinite(v))]

y = df['mpg'].to_numpy().astype(float)
x = df['horsepower'].to_numpy().astype(float)

lam = 100.0
h = np.maximum(0.0, x - lam)
X = np.column_stack([np.ones_like(x), x, h])

beta_hat, residuals, rank, svals = np.linalg.lstsq(X, y, rcond=None)
b0, b1, b2 = beta_hat

a1 = b0
s1 = b1
s2 = b1 + b2

def f_hat(x_val):
    return b0 + b1 * x_val + b2 * np.maximum(0.0, x_val - lam)
```

```
x_left = np.linspace(x.min(), min(lam, x.max()), 200)
x_right = np.linspace(max(lam, x.min()), x.max(), 200)

y_left = a1 + s1 * x_left
y_right = (a1 + s1*lam - s2*lam) + s2 * x_right

print(7*'\n')
plt.figure(figsize=(7, 5))
plt.scatter(x, y, alpha=0.6)
plt.plot(x_left, y_left, linewidth=2)
plt.plot(x_right, y_right, linewidth=2)
plt.axvline(lam, linestyle='--')

plt.xlabel('Horsepower')
plt.ylabel('MPG')
plt.title('Piecewise Linear Fit (λ = 100): MPG vs Horsepower')
plt.tight_layout()
plt.show()

print("Estimated parameters:")
print(f"a1 = {a1:.4f}")
print(f"s1 = {s1:.4f}")
print(f"s2 = {s2:.4f}")

print("\nPiecewise model:")
print(f"f(x) = a1 + s1*x,                for x < {lam}")
print(f"f(x) = a1 + s1*{lam} - s2*{lam} + s2*x,  for x >= {lam}")
```






```
Estimated parameters:
a1 = 53.5772
s1 = -0.3264
s2 = -0.0914

Piecewise model:
f(x) = a1 + s1*x,                  for x < 100.0
f(x) = a1 + s1*100.0 - s2*100.0 + s2*x,  for x >= 100.0
```

Start coding or generate with AI.

Problem – 4

(a) $$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{pmatrix}$$

$$\Rightarrow \quad X = [1 \; Z] \qquad 1 \in \mathbb{R}^n, \; Z \in \mathbb{R}^{n \times d}$$

The linear predictor $\hat{y} = X\beta = \beta_0 1 + Z\beta$

$\beta = (\beta_0, \beta) \in \mathbb{R}^{1+d}$

Let column means be $\mu = \frac{1}{n} Z^T 1 \in \mathbb{R}^d$

Let $Z_c = Z - 1\mu^T$ (center each feature)

$$\beta_0 1 + Z\beta = \beta_0 1 + (Z_c + 1\mu^T)\beta = (\beta_0 + \mu^T\beta) 1 + Z_c \beta$$

For centered features $\beta_0' = \beta_0 + \mu^T\beta \qquad \beta' = \beta_0$

Conversely, (β_0', β') on Z_c, choose $\beta_0 = \beta_0' - \mu^T\beta'$, $\beta = \beta'$ to get same $\hat{y}$ on Z. $\therefore$ The hypothesis classes coincide

$\therefore$ The prediction sets achieve the same minimum training loss before and after centering.

(b) Let $\sigma = (\sigma_1, \dots \sigma_d)$ → column standard deviations

$$S = \text{diag}(\sigma_1, \dots \sigma_d)$$

$$Z_s = ZS^{-1}$$

For any (β_0, β) $\quad \beta_0 1 + Z\beta = \beta_0 1 + Z_s(S\beta)$

So for scaled features

$\beta_0' = \beta_0$ $\quad \beta' = S\beta$. to get same $\hat{y}$.

The map is invertible, so the prediction sets coincide.

The minimum training loss is uncharged & likewise the test loss.

(c) What if use ℓ_1 or ℓ_∞ loss instead of ℓ_2 ?

The above arguments did not rely on the squared loss. Only on the fact that the feature transformers are invertible & that the intercept column is present.

Because the set of achievable predictions $\{\hat{y}\}$ is identical before and after an invertible affine reparametrization, the minimum possible value of any loss that depends only on residuals $r = y - \hat{y}$ is the same.

With ℓ_1 or ℓ_∞, centering/normalizing still cannot improve the optimal loss - they only reparametrize the same prediction functions

∴ The optimal loss is invariant