Question 1-a

Line 1 (A → C):

$$a = 5, \quad b = 8, \quad c = 3, \quad d = 5$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \frac{1}{(5)(5) - (8)(3)} \begin{bmatrix} 5 - 8 \\ 5 - 3 \end{bmatrix}$$

$$= \frac{1}{25 - 24} \begin{bmatrix} -3 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

$$w_1 = -3, \quad w_2 = 2$$

$$-1 - 3x_1 + 2x_2 = 0$$

Line 2 (A → B):

$$a = 5, \quad b = 8, \quad c = 6, \quad d = 4$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \frac{1}{(5)(4) - (8)(6)} \begin{bmatrix} 4 - 8 \\ 5 - 6 \end{bmatrix}$$

$$= \frac{1}{20 - 48} \begin{bmatrix} -4 \\ -1 \end{bmatrix} = \frac{1}{-28} \begin{bmatrix} -4 \\ -1 \end{bmatrix}$$

$$w_1 = \frac{4}{28} = \frac{2}{14} = \frac{1}{7}, \quad w_2 = \frac{1}{28}$$

$$-1 + \frac{1}{7}x_1 + \frac{1}{28}x_2 = 0$$

Line 3 (B → C):

$$a = 6, \quad b = 4, \quad c = 3, \quad d = 5$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \frac{1}{(6)(5) - (4)(3)} \begin{bmatrix} 5 - 4 \\ 6 - 3 \end{bmatrix}$$

$$= \frac{1}{30 - 12} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \frac{1}{18} \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$w_1 = \frac{1}{18}, \quad w_2 = \frac{1}{6}$$

$$-1 + \frac{1}{18}x_1 + \frac{1}{6}x_2 = 0$$

Question 2-b

Line 1 (A → C):

Equation:

$$g_1(x_1, x_2) = -1 - 3x_1 + 2x_2$$

$$g_1(6, 4) = -1 - 3(6) + 2(4) = -1 - 18 + 8 = -11$$

Since $g_1(6, 4) < 0$, the inequality defining the half-plane is:

$$-1 - 3x_1 + 2x_2 > 0$$

Line 2 (A → B):

Equation:

$$g_2(x_1, x_2) = -1 + \frac{1}{7}x_1 + \frac{1}{28}x_2$$

$$g_2(3, 5) = -1 + \frac{1}{7}(3) + \frac{1}{28}(5)$$

$$= -1 + \frac{3}{7} + \frac{5}{28}$$

$$= -1 + 0.4286 + 0.1786 = -0.3929$$

Since $g_2(3, 5) < 0$, the inequality defining the half-plane is:

$$-1 + \frac{1}{7}x_1 + \frac{1}{28}x_2 > 0$$

Line 3 (B → C):

Equation:

$$g_3(x_1, x_2) = -1 + \frac{1}{18}x_1 + \frac{1}{6}x_2$$

$$g_3(5, 8) = -1 + \frac{1}{18}(5) + \frac{1}{6}(8)$$

$$= -1 + 0.2778 + 1.3333 = 0.6111$$

Since $g_3(5, 8) > 0$, the inequality defining the half-plane is:

$$-1 + \frac{1}{18}x_1 + \frac{1}{6}x_2 < 0$$

Final Inequalities for the Triangle Interior:

$$-1 - 3x_1 + 2x_2 > 0$$

$$-1 + \frac{1}{7}x_1 + \frac{1}{28}x_2 > 0$$

$$-1 + \frac{1}{18}x_1 + \frac{1}{6}x_2 < 0$$

Question 1-c

Problem 2: Multi-Layer Perceptron (MLP)

In [25]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
np.set_printoptions(precision=3,suppress=True)
import torch

with open('./winequality-white.csv', 'r') as f:
  X = np.genfromtxt(f,delimiter=',',skip_header=1)
  X, y = X[:,:-1], X[:,-1]
  print('Features:', X.shape)
  print('Labels:', np.unique(y))


y_zero_indexed = y - 3
y_tensor = torch.tensor(y_zero_indexed, dtype=torch.int64)
y_one_hot = torch.nn.functional.one_hot(y_tensor)
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

X_train, X_temp, y_train, y_temp = train_test_split(X_normalized, y_one_hot,
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.

X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
X_val_tensor = torch.tensor(X_val, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
y_val_tensor = torch.tensor(y_val, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)
```

```
Features: (4898, 11)
Labels: [3. 4. 5. 6. 7. 8. 9.]
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_61044/3019548504.
py:27: UserWarning: To copy construct from a tensor, it is recommended to us
e sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_gr
ad_(True), rather than torch.tensor(sourceTensor).
  y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_61044/3019548504.
py:28: UserWarning: To copy construct from a tensor, it is recommended to us
e sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_gr
ad_(True), rather than torch.tensor(sourceTensor).
  y_val_tensor = torch.tensor(y_val, dtype=torch.float32)
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_61044/3019548504.
py:29: UserWarning: To copy construct from a tensor, it is recommended to us
e sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_gr
ad_(True), rather than torch.tensor(sourceTensor).
  y_test_tensor = torch.tensor(y_test, dtype=torch.float32)
```

In [26]:
```
import torch.nn as nn
import torch.optim as optim
# 5. Define the MLP model
class MLP(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(MLP, self).__init__()
        self.layer1 = nn.Linear(input_dim, hidden_dim)
        self.layer2 = nn.Linear(hidden_dim, hidden_dim)
        self.output_layer = nn.Linear(hidden_dim, output_dim)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.relu(self.layer2(x))
        x = self.output_layer(x)
        return x

# Initialize the model, loss function, and optimizer
input_dim = X_train.shape[1]  # 11 features
hidden_dim = 64
output_dim = 7  # 7 classes

model = MLP(input_dim, hidden_dim, output_dim)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 6. Train the model
epochs = 100
batch_size = 32

for epoch in range(epochs):
    model.train()
    permutation = torch.randperm(X_train_tensor.size(0))

    for i in range(0, X_train_tensor.size(0), batch_size):
        indices = permutation[i:i+batch_size]
        batch_X, batch_y = X_train_tensor[indices], y_train_tensor[indices]

        # Forward pass
        optimizer.zero_grad()
        outputs = model(batch_X)

        # Compute loss
        loss = criterion(outputs, batch_y.argmax(dim=1))

        # Backward pass and optimization
        loss.backward()
        optimizer.step()

    if epoch % 10 == 0:
        print(f'Epoch {epoch}/{epochs}, Loss: {loss.item()}')

# 7. Evaluate the model on the validation set
model.eval()
with torch.no_grad():
    val_outputs = model(X_val_tensor)
    _, val_pred = torch.max(val_outputs, 1)
    _, val_true = torch.max(y_val_tensor, 1)
    val_accuracy = accuracy_score(val_true, val_pred)
    print(f'Validation Accuracy: {val_accuracy * 100:.2f}%')

# 8. Evaluate the model on the test set
with torch.no_grad():
    test_outputs = model(X_test_tensor)
    _, test_pred = torch.max(test_outputs, 1)
    _, test_true = torch.max(y_test_tensor, 1)
    test_accuracy = accuracy_score(test_true, test_pred)
    print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
```

```
Epoch 0/100, Loss: 1.0003063678741455
Epoch 10/100, Loss: 1.3477425575256348
Epoch 20/100, Loss: 0.9554032683372498
Epoch 30/100, Loss: 0.6714476346969604
Epoch 40/100, Loss: 1.0337555408477783
Epoch 50/100, Loss: 0.9108670353889465
Epoch 60/100, Loss: 0.8693878650665283
Epoch 70/100, Loss: 0.6687616109848022
Epoch 80/100, Loss: 0.6591122150421143
Epoch 90/100, Loss: 0.8749755024909973
Validation Accuracy: 57.24%
Test Accuracy: 58.67%
```