目　录

1. 作业要求

已知网络结构如图 1 所示，网络输入/输出如表 1 所示。$f(x)$ 为 x 的符号函数：

$$f(net) = f(w_1 \times x_1 + w_2 \times x_2 + w_3 \times 1)$$

其中，$bias$ 取常数 1，设初始值随机取成$(0.75, 0.5, -0.6)$。对于这 10 组训练参数，若训练数据经过 $f(net)$ 网络计算后的输出与理想输出 Y 不相符，则使用如下有师学习算法对网络权值进行更新：

$$w^t = w^{t-1} + c(d^{t-1} - sign(w^{t-1} * x^{t-1}))x^{t-1}$$

其中 c 为学习率，t 为迭代次数，d^{t-1} 是第 $t-1$ 代的理想输出值。

本次实验，我们的目标是在固定学习率 c 的基础上，通过使用不同的学习率调度策略，实现学习率动态自适应调整。

表 1. 输入/输出训练参数表

训练序号	x_1	x_2	Y
1	1.0	1.0	1
2	9.4	6.4	-1
3	2.5	2.1	1
4	8.0	7.7	-1
5	0.5	2.2	1
6	7.9	8.4	-1
7	7.0	7.0	-1
8	2.8	0.8	1
9	1.2	3.0	1
10	7.8	6.1	-1

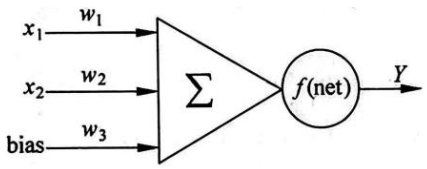

图 1. 神经网络结构示例图

2. 实验原理

2.1 *StepLR*策略

(1) 初始化参数

- $w = [0.75;\ 0.5;\ -0.6]$：权重向量的初始值。这里包括了两个特征的权重和一个偏置项。
- $c = 0.1$：初始学习率，用于在每次更新权值时决定更新步长。
- $gamma = 0.9$：学习率衰减系数，每进行一次固定步数的迭代后会将学习率

乘以 $gamma$ 。

- $step\_size = 1$ ：学习率调整的步长，即每进行一次迭代，就会根据这个步长更新学习率。
- $max\_iterations = 500$ ：最大迭代次数，如果达到最大迭代次数而没有收敛，程序将停止。
- $tolerance = 10^{-9}$ ：收敛容忍度，如果总误差小于这个阈值，则认为训练已经收敛。

(2) 训练数据

- $data$ 包含了 10 个数据点，每个数据点有 2 个特征和一个标签（1 或 -1）。
- $X = [data(:, 1{:}2), ones(size(data, 1), 1)]$ ：特征矩阵，其中每个数据点都加入了一个偏置项 1。
- $Y = data(:, 3)$ ：目标标签向量。

(3) 训练过程

- 每经过 $step\_size$ 次迭代，学习率 c 将乘以 $gamma$ 进行衰减，从而减小权值更新的幅度。
- 对于每一个训练样本，计算其净输入：

$$net = w' \cdot X(i,:)$$

然后使用符号函数 $sign(net)$ 得到预测值。

- 如果输出与实际标签一致，则认为该样本分类正确。如果输出和实际标签不一致，就会更新权值：

$$w = w + c \times error \times X(i,:)'$$

其中 $error = Y(i) - output$ ，是网络输出与实际标签之间的差异，$X(i,:)'$ 是当前样本的特征向量，c 是学习率，控制每次权值更新的幅度。

- 如果所有训练样本的误差小于给定容忍度 $tolerance$ ，则认为训练已经收敛，终止迭代。

(4) 流程图

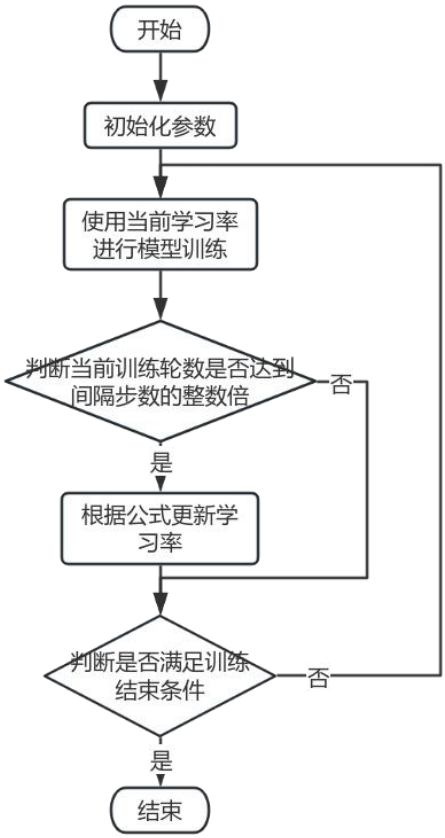

图 3. $StepLR$ 策略训练流程图

2.2 $ExponentialLR$策略

(1) 初始化参数

- $w = [0.75;\ 0.5;\ -0.6]$：权重向量的初始值。这里包括了两个特征的权重和一个偏置项。
- $c_0 = 0.1$：初始学习率。
- $gamma = 0.9$：学习率的指数衰减因子。
- $max\_epochs = 500$：最大训练轮数，用来控制训练的迭代次数。
- $tolerance = 10^{-9}$：误差容忍度，当总误差小于此值时训练停止。

(2) 训练数据

- $data$ 包含了 10 个数据点，每个数据点有 2 个特征和一个标签（1 或 -1）。
- $X = [data(:, 1{:}2), ones(size(data, 1), 1)]$：特征矩阵，其中每个数据点都

加入了一个偏置项 1。

- $Y = data(:,3)$：目标标签向量。

(3) 训练过程

- 在每一轮训练中，学习率 c 根据指数衰减公式进行更新：

$$c = c_0 \times \gamma^{epoch-1}$$

并记录每轮的学习率变化。

- 对每个数据点，计算其净输入：

$$net = w' \times X(i,:)'$$

然后通过符号函数 $sign(net)$ 得到预测值。

- 计算模型预测输出与真实标签之间的误差：

$$error = Y(i) - output$$

如果误差不为零，说明预测错了，需要更新权重。

总误差是所有数据点误差的绝对值和：

$$total\_error = total\_error + abs(error)$$

- 如果误差 $error$ 不为零，则根据梯度下降公式更新权重：

$$w = w + c \times error \times X(i,:)'$$

其中 c 是当前的学习率。

- 每次迭代后，使用当前的权重计算所有数据点的预测，并与真实标签对比，计算分类准确率：

$$accuracy = sum(predictions == Y) \,/\, length(Y)$$

- 如果总误差 $total\_error$ 小于设定的容忍度 $tolerance$，则认为模型已经收敛，训练提前结束。
- 输出最终的权重值。

(4) 流程图

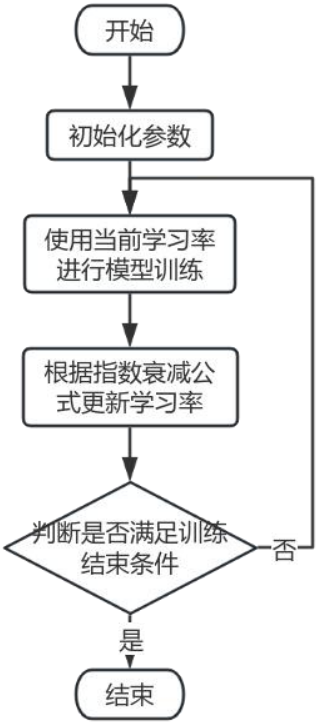

图 4. $ExponentialLR$策略训练流程图

2.3 $ReduceLROnPlateau$策略

(1) 初始化参数

- $w = [0.75;\ 0.5;\ -0.6]$：初始化感知机的权重，包括偏置项。
- $c_0 = 0.1$：初始学习率，用于权重更新。
- $max\_epochs = 500$：最大训练轮数，表示最多进行 500 次迭代。
- $tolerance = 10^{-9}$：训练误差的容忍度，用于判断训练是否收敛。

(2) 训练数据

将数据集分为训练集和验证集：

- $train\_ratio = 0.8$：训练集占比 80%。
- $num\_samples$：数据集的样本数量（10 个数据点）。
- num_train：训练集的样本数量，按 $train\_ratio$ 计算。
- $train\_indices$ 和 $val\_indices$：通过 $randperm$ 函数随机选择训练集和验证集的样本索引。

提取训练集和验证集的输入和输出，加入偏置项（即每个输入样本都增加一个值为 1 的维度）。

(3) 训练过程

- 每次训练开始时，检查验证集的准确率。如果验证准确率连续 3 轮没有提高，则学习率减少为当前学习率的一半，并确保最小学习率不低于min_lr。
- 对于每一个训练样本，计算净输入：

$$net = w' \times X\_train(i,:)'$$

然后通过符号函数 $sign(net)$ 得到模型输出（预测值）。

- 计算误差：

$$error = Y\_train(i) - output$$

- 如果误差不为零（即分类错误），根据感知机规则更新权重：

$$w = w + c \times error \times X\_train(i,:)'$$

其中，c 是当前学习率。

- 在每一轮训练后，使用验证集进行前向传播，并计算验证准确率：

$$predictions\_val = sign(X\_val \times w)$$

计算验证集准确率：

$$accuracy\_val = sum(predictions\_val == Y\_val) \,/\, length(Y\_val)$$

- 如果当前验证集准确率高于历史最佳准确率，则更新 $best\_accuracy$，并重置 $counter$（用于记录连续未提升的轮数）。如果准确率没有提升，$counter + 1$。
- 如果训练误差小于预定的容忍度（$total\_train\_error < tolerance$），则认为训练收敛，提前终止训练。

(4) 流程图

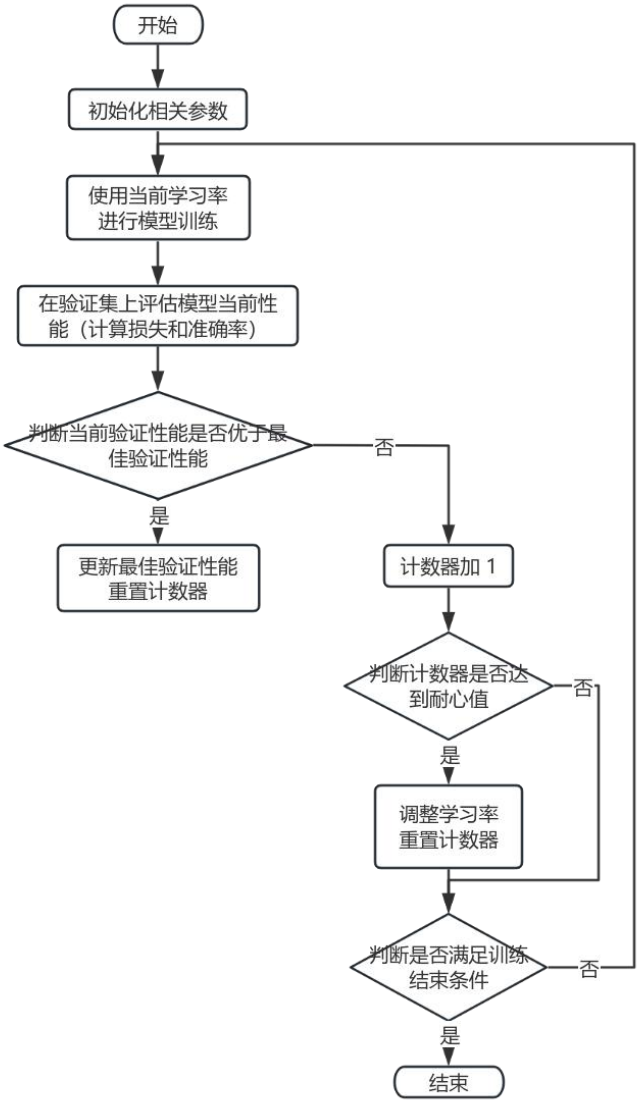

图 5. *ReduceLROnPlateau* 策略训练流程图

3. 代码实现

3.1 固定学习率代码

```
% 初始化参数
w = [0.75; 0.5; -0.6]; % 权值初始化
c = 0.2; % 固定学习率
max_epochs = 50; % 最大训练轮数
tolerance = 1e-9; % 收敛容忍度

% 数据
```

```
data = [1.0, 1.0, 1;
        9.4, 6.4, -1;
        2.5, 2.1, 1;
        8.0, 7.7, -1;
        0.5, 2.2, 1;
        7.9, 8.4, -1;
        7.0, 7.0, -1;
        2.8, 0.8, 1;
        1.2, 3.0, 1;
        7.8, 6.1, -1]; % 训练数据

% 提取输入和输出
X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
Y = data(:, 3);

% 记录准确率历史
accuracy_history = [];

% 训练循环
for epoch = 1:max_epochs
    % 初始化误差
    total_error = 0;

    for i = 1:size(X, 1)
        % 前向传播
        net = w' * X(i, :)'; % 计算净输入
        output = sign(net); % 计算输出，使用符号函数

        % 计算误差
        error = Y(i) - output;
        total_error = total_error + abs(error);

        % 更新权值
        if error ~= 0
            w = w + c * error * X(i, :)'; % 使用固定学习率更新权值
        end
    end

    % 计算分类准确率
    predictions = sign(X * w);
    accuracy = sum(predictions == Y) / length(Y);
    accuracy_history = [accuracy_history, accuracy];

    % 检查是否收敛
```

```
    if total_error < tolerance
        fprintf('Training converged after %d epochs.\n', epoch);
        break;
    end
end

% 最终权值
fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));

% 绘制准确率变化
figure;
plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
2);
xlabel('Epoch');
ylabel('Accuracy');
title('Accuracy vs. Epoch');
grid on;
```

3.2 *StepLR*代码

```
% 初始化参数
w = [0.75; 0.5; -0.6]; % 权值初始化
c = 0.1; % 初始学习率
gamma = 0.9; % 学习率缩小倍数
step_size = 1; % 学习率调整步长
max_iterations = 500; % 最大迭代次数
tolerance = 1e-9; % 收敛容忍度
data = [1.0, 1.0, 1;
        9.4, 6.4, -1;
        2.5, 2.1, 1;
        8.0, 7.7, -1;
        0.5, 2.2, 1;
        7.9, 8.4, -1;
        7.0, 7.0, -1;
        2.8, 0.8, 1;
        1.2, 3.0, 1;
        7.8, 6.1, -1]; % 训练数据

% 提取输入和输出
X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
Y = data(:, 3);

% 初始化记录变量
```

```
learning_rate_history = []; % 记录学习率变化
accuracy_history = []; % 记录正确率变化

% 迭代过程
for iter = 1:max_iterations
    % 动态调整学习率
    if mod(iter, step_size) == 0
        c = c * gamma;
    end
    learning_rate_history = [learning_rate_history, c];

    % 初始化当前误差
    total_error = 0;
    correct_classifications = 0; % 正确分类样本数

    for i = 1:size(X, 1)
        % 前向传播
        net = w' * X(i, :)'; % 计算净输入
        output = sign(net); % 计算输出，使用符号函数

        % 检查分类是否正确
        if output == Y(i)
            correct_classifications = correct_classifications + 1;
        end

        % 计算当前误差
        error = Y(i) - output;
        total_error = total_error + abs(error);

        % 权值更新
        if error ~= 0
            w = w + c * error * X(i, :)';
        end
    end

    % 记录正确率
    accuracy = correct_classifications / size(X, 1);
    accuracy_history = [accuracy_history, accuracy];

    % 检查误差是否在容忍度范围内
    if total_error < tolerance
        fprintf('Training converged after %d iterations.\n', iter);
        break;
    end
```

```
68.   end
69.
70.   % 输出最终权值
71.   fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));
72.
73.   % 绘制学习率变化情况
74.   figure;
75.   subplot(2, 1, 1);
76.   plot(1:length(learning_rate_history), learning_rate_history, 'b-',
      'LineWidth', 2);
77.   xlabel('Iteration');
78.   ylabel('Learning Rate');
79.   title('Learning Rate vs. Iteration');
80.   grid on;
81.
82.   % 绘制正确率变化情况
83.   subplot(2, 1, 2);
84.   plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
      2);
85.   xlabel('Iteration');
86.   ylabel('Accuracy');
87.   title('Accuracy vs. Iteration');
88.   grid on;
89.
90.
91.   % 绘制分类图
92.   figure;
93.   hold on;
94.
95.   % 绘制数据点
96.   gscatter(data(:, 1), data(:, 2), Y, 'rb', 'o', 8);
97.
98.   % 绘制决策边界
99.   % 计算边界：w1*x1 + w2*x2 + b = 0 => x2 = -(w1/w2)*x1 - (b/w2)
100.  x1 = linspace(min(data(:, 1)), max(data(:, 1)), 100);
101.  x2 = - (w(1)/w(2)) * x1 - (w(3)/w(2));
102.
103.  plot(x1, x2, 'k--', 'LineWidth', 2); % 绘制决策边界
104.  xlabel('Feature 1');
105.  ylabel('Feature 2');
106.  title('Final Classification with Decision Boundary');
107.  legend('Class 1', 'Class -1', 'Decision Boundary');
108.  grid on;
109.  hold off;
```

3.3 *ExponentialLR* 代码

```
% 初始化参数
w = [0.75; 0.5; -0.6]; % 权值初始化
c0 = 0.1; % 初始学习率
gamma = 0.9; % 指数衰减因子
max_epochs = 500; % 最大训练轮数
tolerance = 1e-9; % 收敛容忍度

% 数据
data = [1.0, 1.0, 1;
        9.4, 6.4, -1;
        2.5, 2.1, 1;
        8.0, 7.7, -1;
        0.5, 2.2, 1;
        7.9, 8.4, -1;
        7.0, 7.0, -1;
        2.8, 0.8, 1;
        1.2, 3.0, 1;
        7.8, 6.1, -1]; % 训练数据

% 提取输入和输出
X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
Y = data(:, 3);

% 记录学习率和误差历史
learning_rate_history = [];
accuracy_history = [];

% 训练循环
for epoch = 1:max_epochs
    % 更新学习率 (指数衰减)
    c = c0 * gamma^(epoch - 1);
    learning_rate_history = [learning_rate_history, c]; % 记录学习率

    % 初始化误差
    total_error = 0;

    for i = 1:size(X, 1)
        % 前向传播
        net = w' * X(i, :)'; % 计算净输入
        output = sign(net); % 计算输出，使用符号函数

```

```
        % 计算误差
        error = Y(i) - output;
        total_error = total_error + abs(error);

        % 更新权值
        if error ~= 0
            w = w + c * error * X(i, :)'; % 学习率调整后的权值更新
        end
    end

    % 计算分类准确率
    predictions = sign(X * w);
    accuracy = sum(predictions == Y) / length(Y);
    accuracy_history = [accuracy_history, accuracy];

    % 检查是否收敛
    if total_error < tolerance
        fprintf('Training converged after %d epochs.\n', epoch);
        break;
    end
end

% 最终权值
fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));

% 绘制学习率和准确率变化
figure;

% 学习率变化图
subplot(2, 1, 1);
plot(1:length(learning_rate_history), learning_rate_history, 'b-',
'LineWidth', 2);
xlabel('Epoch');
ylabel('Learning Rate');
title('Learning Rate vs. Epoch');
grid on;

% 准确率变化图
subplot(2, 1, 2);
plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
2);
xlabel('Epoch');
ylabel('Accuracy');
title('Accuracy vs. Epoch');
```

```
grid on;
% 绘制分类图
figure;
hold on;

% 绘制数据点
gscatter(data(:, 1), data(:, 2), Y, 'rb', 'o', 8);

% 绘制决策边界
% 计算边界：w1*x1 + w2*x2 + b = 0 => x2 = -(w1/w2)*x1 - (b/w2)
x1 = linspace(min(data(:, 1)), max(data(:, 1)), 100);
x2 = - (w(1)/w(2)) * x1 - (w(3)/w(2));

plot(x1, x2, 'k--', 'LineWidth', 2); % 绘制决策边界
xlabel('Feature 1');
ylabel('Feature 2');
title('Final Classification with Decision Boundary');
legend('Class 1', 'Class -1', 'Decision Boundary');
grid on;
hold off;
```

3.4 *ReduceLROnPlateau*代码

```
% 初始化参数
w = [0.75; 0.5; -0.6]; % 权值初始化
c0 = 0.1; % 初始学习率
max_epochs = 500; % 最大训练轮数
tolerance = 1e-9; % 收敛容忍度

% 划分训练集和验证集（这里简单按比例划分，可根据实际调整）
data = [1.0, 1.0, 1;
        9.4, 6.4, -1;
        2.5, 2.1, 1;
        8.0, 7.7, -1;
        0.5, 2.2, 1;
        7.9, 8.4, -1;
        7.0, 7.0, -1;
        2.8, 0.8, 1;
        1.2, 3.0, 1;
        7.8, 6.1, -1]; % 训练数据

train_ratio = 0.8; % 训练集占比
num_samples = size(data, 1);
```

```
num_train = floor(num_samples * train_ratio);
train_indices = randperm(num_samples, num_train);
val_indices = setdiff(1:num_samples, train_indices);

data_train = data(train_indices, :);
data_val = data(val_indices, :);

% 提取训练集输入和输出，并加入偏置项
X_train = [data_train(:, 1:2), ones(size(data_train, 1), 1)];
Y_train = data_train(:, 3);

% 提取验证集输入和输出，并加入偏置项
X_val = [data_val(:, 1:2), ones(size(data_val, 1), 1)];
Y_val = data_val(:, 3);

% 设置 ReduceLROnPlateau 相关参数
patience = 3; % 当验证准确率不再提升，等待多少轮后降低学习率
factor = 0.5; % 学习率降低因子
min_lr = 1e-6; % 最小学习率，防止学习率降得过低
best_accuracy = 0; % 初始最佳验证准确率设为 0
counter = 0; % 计数器，用于记录验证准确率未提升的轮数

% 记录学习率和误差历史
learning_rate_history = [];
accuracy_history = [];

% 训练循环
for epoch = 1:max_epochs
    % 更新学习率（基于 ReduceLROnPlateau 策略）
    if counter >= patience
        c0 = max(c0 * factor, min_lr); % 降低学习率，确保不低于最小学习率
        counter = 0;
    end
    c = c0; % 当前学习率
    learning_rate_history = [learning_rate_history, c]; % 记录学习率

    % 初始化训练误差和验证误差
    total_train_error = 0;
    total_val_error = 0;

    % 训练集前向传播与权值更新
    for i = 1:size(X_train, 1)
        % 前向传播
        net = w' * X_train(i, :)'; % 计算净输入
```

```
        output = sign(net); % 计算输出，使用符号函数

        % 计算误差
        error = Y_train(i) - output;
        total_train_error = total_train_error + abs(error);

        % 更新权值
        if error ~= 0
            w = w + c * error * X_train(i, :)'; % 学习率调整后的权值更新
        end
    end

    % 验证集前向传播，计算验证准确率
    predictions_val = sign(X_val * w);
    accuracy_val = sum(predictions_val == Y_val) / length(Y_val);
    accuracy_history = [accuracy_history, accuracy_val];

    % 根据验证准确率更新最佳准确率与计数器
    if accuracy_val > best_accuracy
        best_accuracy = accuracy_val;
        counter = 0;
    else
        counter = counter + 1;
    end

    % 检查是否收敛（这里简单根据训练误差判断，可优化）
    if total_train_error < tolerance
        fprintf('Training converged after %d epochs.\n', epoch);
        break;
    end
end

% 最终权值
fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));

% 绘制学习率和准确率变化
figure;

% 学习率变化图
subplot(2, 1, 1);
plot(1:length(learning_rate_history), learning_rate_history, 'b-',
'LineWidth', 2);
xlabel('Epoch');
ylabel('Learning Rate');
```

```
title('Learning Rate vs. Epoch');
grid on;

% 准确率变化图
subplot(2, 1, 2);
plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth', 2);
xlabel('Epoch');
ylabel('Accuracy');
title('Accuracy vs. Epoch');
grid on;

% 绘制分类图
figure;
hold on;

% 绘制数据点
gscatter(data(:, 1), data(:, 2), data(:, 3), 'rb', 'o', 8);

% 绘制决策边界
% 计算边界：w1*x1 + w2*x2 + b = 0 => x2 = -(w1/w2)*x1 - (b/w2)
x1 = linspace(min(data(:, 1)), max(data(:, 1)), 100);
x2 = - (w(1)/w(2)) * x1 - (w(3)/w(2));

plot(x1, x2, 'k--', 'LineWidth', 2); % 绘制决策边界
xlabel('Feature 1');
ylabel('Feature 2');
title('Final Classification with Decision Boundary');
legend('Class 1', 'Class -1', 'Decision Boundary');
grid on;
hold off;
```

4. 实验结果

4.1 固定学习率结果

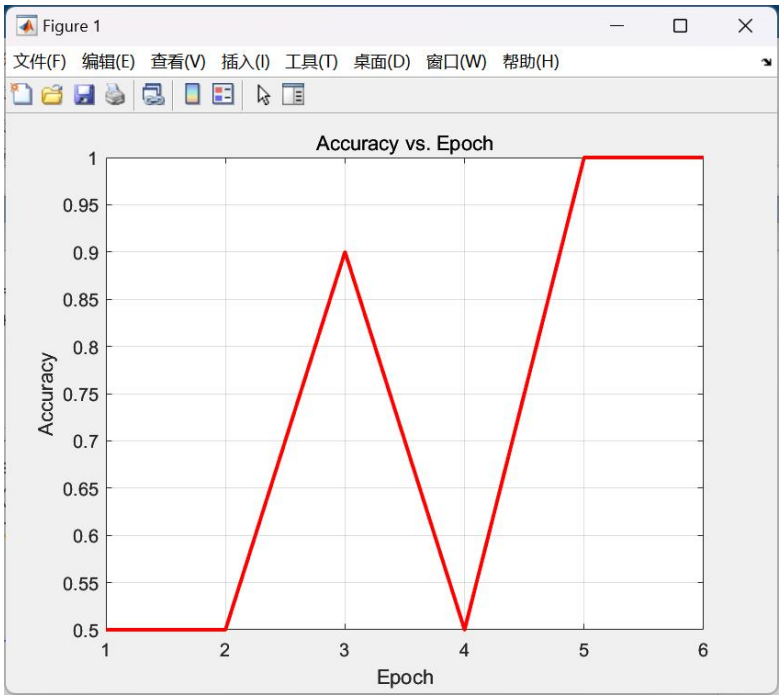

图 6. 固定学习率的准确性变化曲线

```
Training converged after 6 epochs.
Final weights: -0.6900, -0.2600, 3.8000
```

图 7. 固定学习率的最终权值

4.2 *StepLR*结果

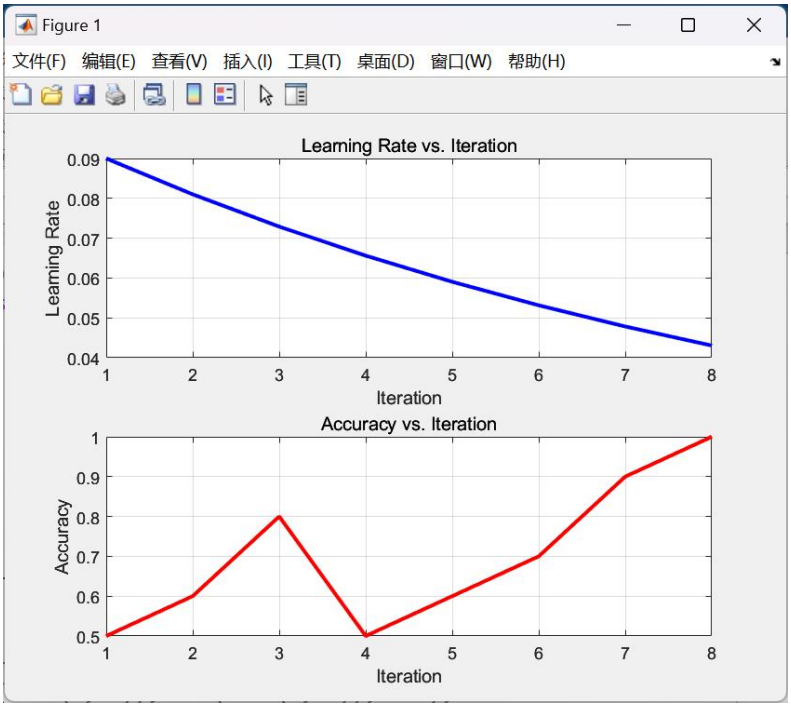

图 8. *StepLR* 的学习率与准确率的变化曲线

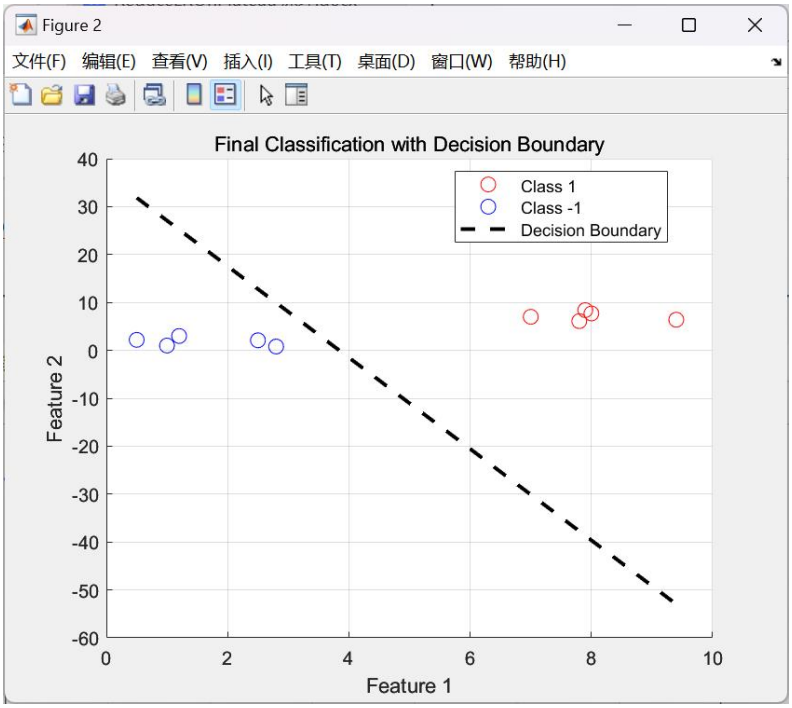

图 9. *StepLR* 产生的决策边界

```
Training converged after 8 iterations.
Final weights: -0.3373, -0.0354, 1.2960
```

图 10. *StepLR* 的最终权值

4.3 *ExponentialLR* 结果

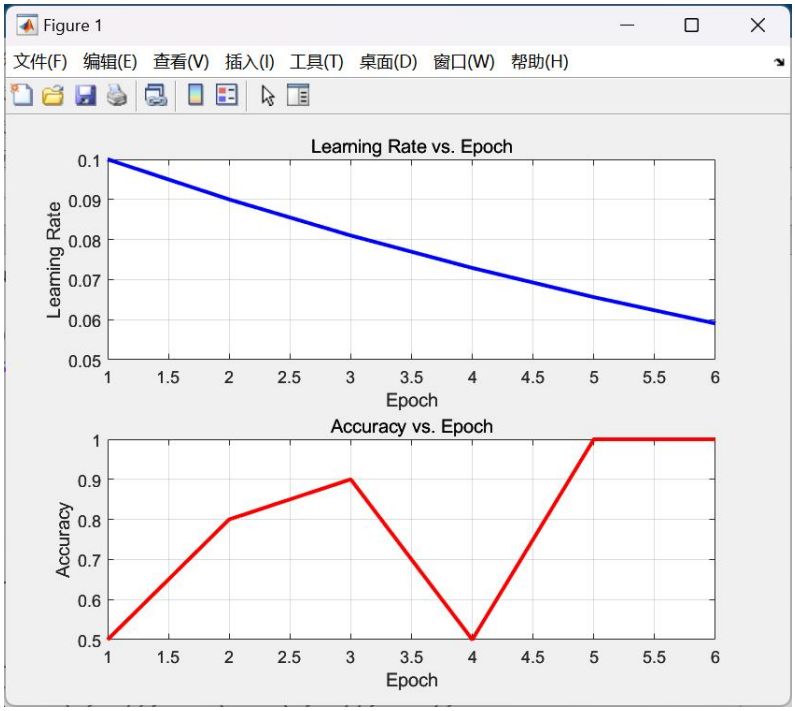

图 11. *ExponentialLR* 的学习率与准确率的变化曲线

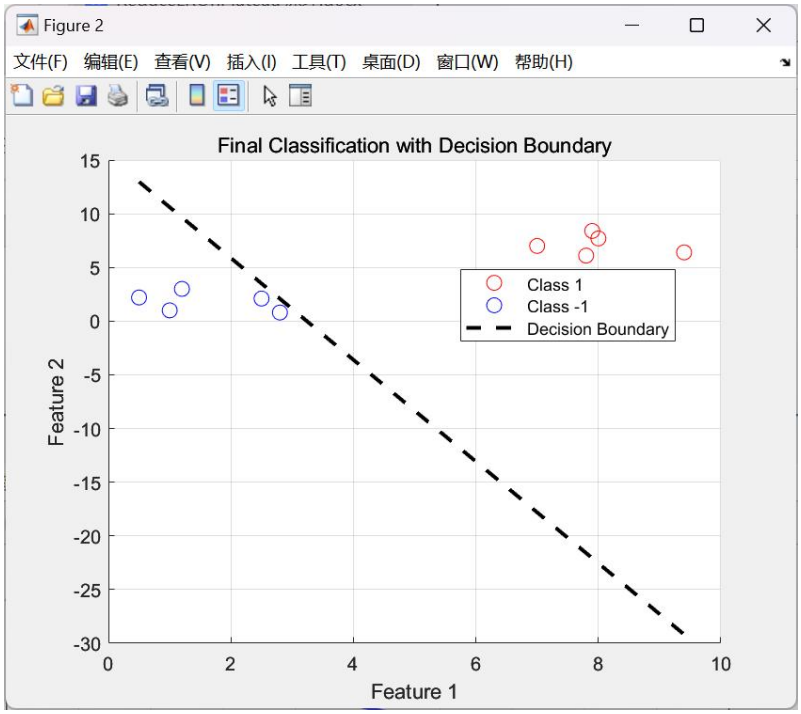

图 12. *ExponentialLR* 产生的决策边界

```
Training converged after 6 epochs.
Final weights: -0.3703, -0.0782, 1.1997
```

图 13. *ExponentialLR* 最终权值

4.4 *ReduceLROnPlateau* 结果

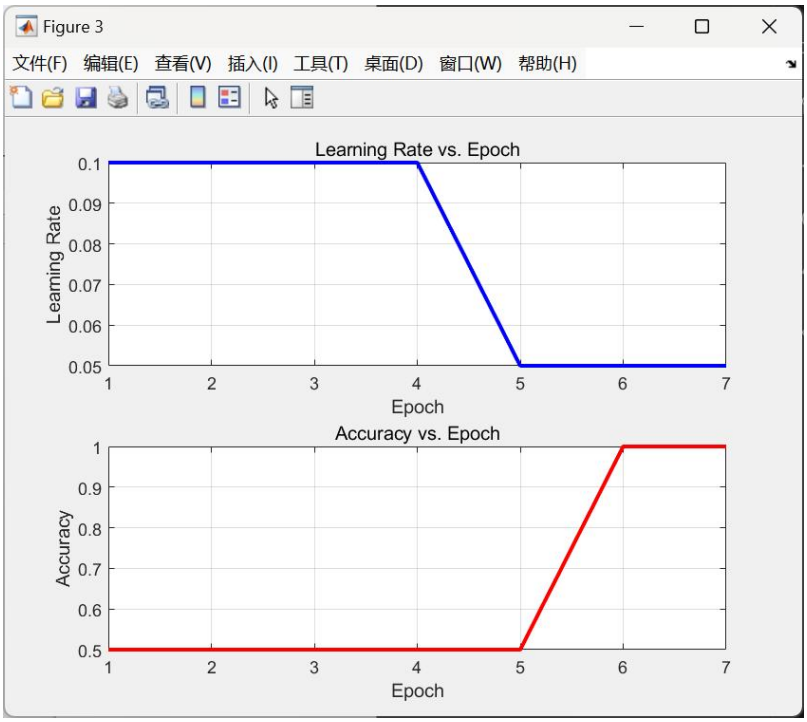

图 14. *ReduceLROnPlateau* 的学习率与准确率的变化曲线

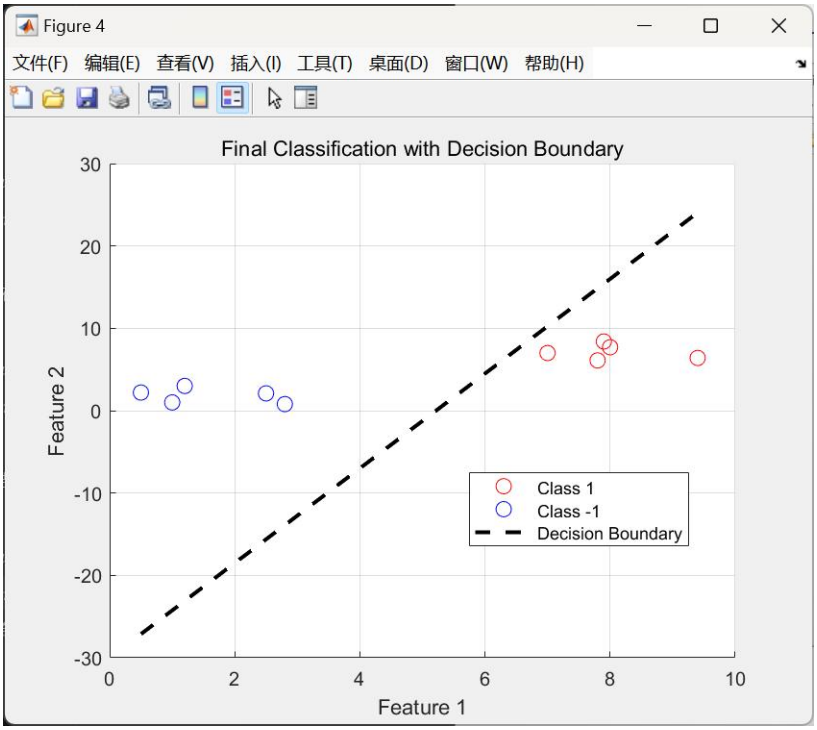

图 15. *ReduceLROnPlateau* 产生的决策边界

```
Training converged after 7 epochs.
Final weights: -0.2300, 0.0400, 1.2000
```

图 16. $ReduceLROnPlateau$ 的最终权值

4.5 学习率更新策略对比

	固定学习率	$StepLR$	$ExponentialLR$	$ReduceLROnPlateau$
复杂性	低	中	中	高
稳定性	高 （在合适的学习率下）	中	中	高 （智能调整）
自适应性	低	中	高	最高
超参数	1 个 （学习率）	2 个 （步长、衰减系数）	1 个 （衰减因子）	3 个 （耐心度、降低因子、最小学习率）
计算成本	低	中	中	高
适用场景	简单任务	中等复杂任务	复杂任务	各种复杂任务

4.6 结论

在选择学习率调整策略时，需要根据具体任务、数据分布、模型结构以及计算资源等因素进行综合考虑。

固定学习率策略简单稳定，但难以选择最佳学习率；$StepLR$ 和 $ExponentialLR$ 策略具有一定的灵活性，但需要仔细调整超参数；$ReduceLROnPlateau$ 策略智能调整学习率，适用性广，但计算成本较高且需要设置多个超参数。

因此，在实际应用中，可以根据具体需求选择合适的策略进行训练。