

## 目 录

1. 作业要求 .....	3
2. 实验原理 .....	3
2.1 <b>StepLR</b> 策略 .....	3
2.2 <b>ExponentialLR</b> 策略 .....	5
2.3 <b>ReduceLROnPlateau</b> 策略 .....	7
3. 代码实现 .....	9
3.1 固定学习率代码 .....	9
3.2 <b>StepLR</b> 代码 .....	11
3.3 <b>ExponentialLR</b> 代码 .....	14
3.4 <b>ReduceLROnPlateau</b> 代码 .....	16
4. 实验结果 .....	19
4.1 固定学习率结果 .....	19
4.2 <b>StepLR</b> 结果 .....	20
4.3 <b>ExponentialLR</b> 结果 .....	21
4.4 <b>ReduceLROnPlateau</b> 结果 .....	22
4.5 学习率更新策略对比 .....	24
4.6 结论 .....	24

## 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：

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

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

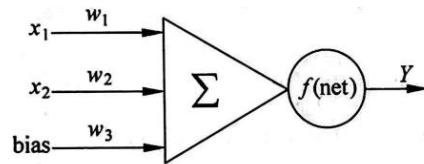


图 1. 神经网络结构示例图

## 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

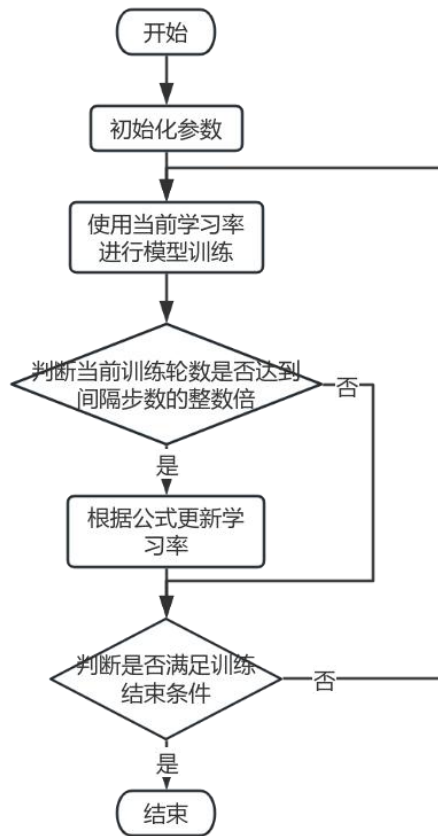


图 3. *StepLR* 策略训练流程图

## 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。
- $\text{max\_epochs} = 500$ ：最大训练轮数，用来控制训练的迭代次数。
- $\text{tolerance} = 10^{-9}$ ：误差容忍度，当总误差小于此值时训练停止。

### (2) 训练数据

- $\text{data}$  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）。
- $X = [\text{data}(:, 1:2), \text{ones}(\text{size}(\text{data}, 1), 1)]$ ：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

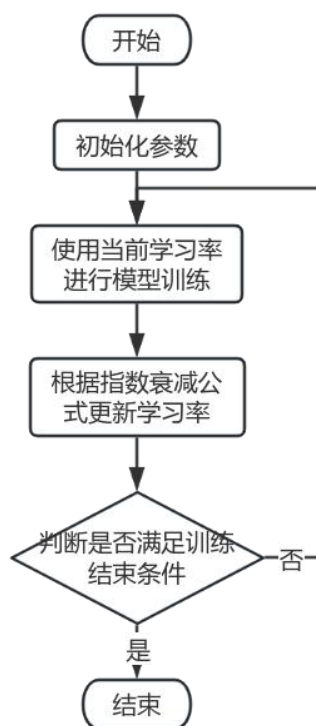


图 4. *ExponentialLR*策略训练流程图

### 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

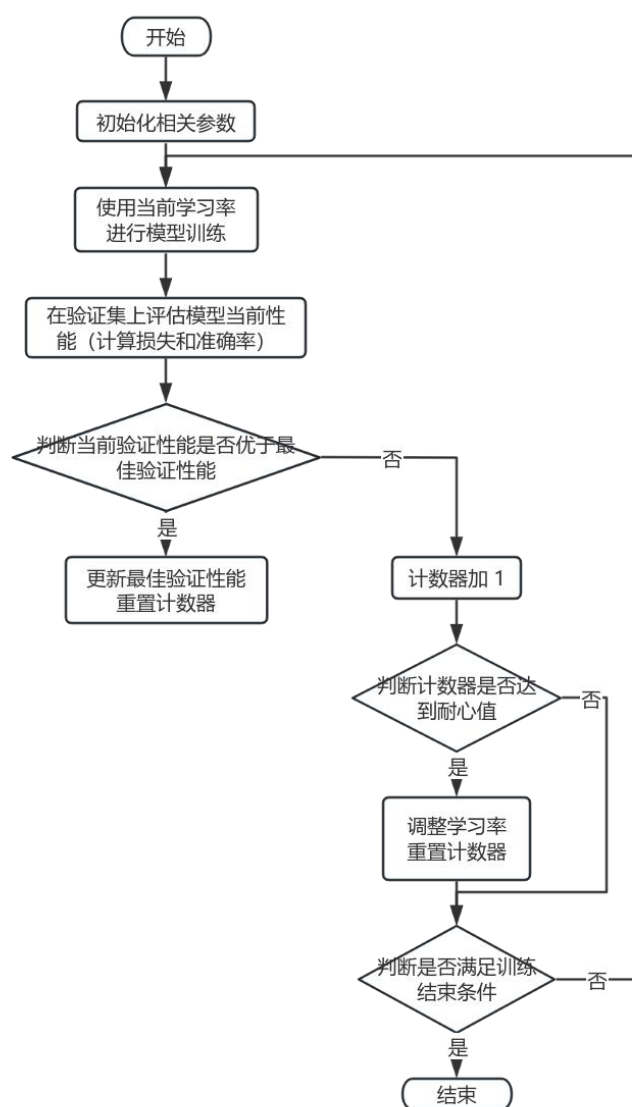


图 5. *ReduceLROnPlateau* 策略训练流程图

## 3. 代码实现

### 3.1 固定学习率代码

1.    % 初始化参数
2.    w = [0.75; 0.5; -0.6]; % 权值初始化
3.    c = 0.2; % 固定学习率
4.    max\_epochs = 50; % 最大训练轮数
5.    tolerance = 1e-9; % 收敛容忍度
- 6.
7.    % 数据



```

8.     data = [1.0, 1.0, 1;
9.             9.4, 6.4, -1;
10.          2.5, 2.1, 1;
11.          8.0, 7.7, -1;
12.          0.5, 2.2, 1;
13.          7.9, 8.4, -1;
14.          7.0, 7.0, -1;
15.          2.8, 0.8, 1;
16.          1.2, 3.0, 1;
17.          7.8, 6.1, -1]; % 训练数据
18.
19. % 提取输入和输出
20. X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
21. Y = data(:, 3);
22.
23. % 记录准确率历史
24. accuracy_history = [];
25.
26. % 训练循环
27. for epoch = 1:max_epochs
28.     % 初始化误差
29.     total_error = 0;
30.
31.     for i = 1:size(X, 1)
32.         % 前向传播
33.         net = w' * X(i, :); % 计算净输入
34.         output = sign(net); % 计算输出, 使用符号函数
35.
36.         % 计算误差
37.         error = Y(i) - output;
38.         total_error = total_error + abs(error);
39.
40.         % 更新权值
41.         if error ~= 0
42.             w = w + c * error * X(i, :); % 使用固定学习率更新权值
43.         end
44.     end
45.
46. % 计算分类准确率
47. predictions = sign(X * w);
48. accuracy = sum(predictions == Y) / length(Y);
49. accuracy_history = [accuracy_history, accuracy];
50.
51. % 检查是否收敛

```

```

52.         if total_error < tolerance
53.             fprintf('Training converged after %d epochs.\n', epoch);
54.             break;
55.         end
56.     end
57.
58.     % 最终权值
59.     fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));
60.
61.     % 绘制准确率变化
62.     figure;
63.     plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
        2);
64.     xlabel('Epoch');
65.     ylabel('Accuracy');
66.     title('Accuracy vs. Epoch');
67.     grid on;

```

### 3.2 StepLR代码

```

1.     % 初始化参数
2.     w = [0.75; 0.5; -0.6]; % 权值初始化
3.     c = 0.1; % 初始学习率
4.     gamma = 0.9; % 学习率缩小倍数
5.     step_size = 1; % 学习率调整步长
6.     max_iterations = 500; % 最大迭代次数
7.     tolerance = 1e-9; % 收敛容忍度
8.     data = [1.0, 1.0, 1;
9.            9.4, 6.4, -1;
10.           2.5, 2.1, 1;
11.           8.0, 7.7, -1;
12.           0.5, 2.2, 1;
13.           7.9, 8.4, -1;
14.           7.0, 7.0, -1;
15.           2.8, 0.8, 1;
16.           1.2, 3.0, 1;
17.           7.8, 6.1, -1]; % 训练数据
18.
19.     % 提取输入和输出
20.     X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
21.     Y = data(:, 3);
22.
23.     % 初始化记录变量

```

```

24. learning_rate_history = []; % 记录学习率变化
25. accuracy_history = []; % 记录正确率变化
26.
27. % 迭代过程
28. for iter = 1:max_iterations
29.     % 动态调整学习率
30.     if mod(iter, step_size) == 0
31.         c = c * gamma;
32.     end
33.     learning_rate_history = [learning_rate_history, c];
34.
35.     % 初始化当前误差
36.     total_error = 0;
37.     correct_classifications = 0; % 正确分类样本数
38.
39.     for i = 1:size(X, 1)
40.         % 前向传播
41.         net = w' * X(i, :)'; % 计算净输入
42.         output = sign(net); % 计算输出, 使用符号函数
43.
44.         % 检查分类是否正确
45.         if output == Y(i)
46.             correct_classifications = correct_classifications + 1;
47.         end
48.
49.         % 计算当前误差
50.         error = Y(i) - output;
51.         total_error = total_error + abs(error);
52.
53.         % 权值更新
54.         if error ~= 0
55.             w = w + c * error * X(i, :)';
56.         end
57.     end
58.
59.     % 记录正确率
60.     accuracy = correct_classifications / size(X, 1);
61.     accuracy_history = [accuracy_history, accuracy];
62.
63.     % 检查误差是否在容忍度范围内
64.     if total_error < tolerance
65.         fprintf('Training converged after %d iterations.\n', iter);
66.         break;
67.     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:  $w_1x_1 + w_2x_2 + b = 0 \Rightarrow x_2 = -(w_1/w_2)x_1 - (b/w_2)$ 
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

```
1.  % 初始化参数
2.  w = [0.75; 0.5; -0.6]; % 权值初始化
3.  c0 = 0.1; % 初始学习率
4.  gamma = 0.9; % 指数衰减因子
5.  max_epochs = 500; % 最大训练轮数
6.  tolerance = 1e-9; % 收敛容忍度
7.
8.  % 数据
9.  data = [1.0, 1.0, 1;
10.         9.4, 6.4, -1;
11.         2.5, 2.1, 1;
12.         8.0, 7.7, -1;
13.         0.5, 2.2, 1;
14.         7.9, 8.4, -1;
15.         7.0, 7.0, -1;
16.         2.8, 0.8, 1;
17.         1.2, 3.0, 1;
18.         7.8, 6.1, -1]; % 训练数据
19.
20. % 提取输入和输出
21. X = [data(:, 1:2), ones(size(data, 1), 1)]; % 加入偏置项
22. Y = data(:, 3);
23.
24. % 记录学习率和误差历史
25. learning_rate_history = [];
26. accuracy_history = [];
27.
28. % 训练循环
29. for epoch = 1:max_epochs
30.     % 更新学习率 (指数衰减)
31.     c = c0 * gamma^(epoch - 1);
32.     learning_rate_history = [learning_rate_history, c]; % 记录学习率
33.
34.     % 初始化误差
35.     total_error = 0;
36.
37.     for i = 1:size(X, 1)
38.         % 前向传播
39.         net = w' * X(i, :)'; % 计算净输入
40.         output = sign(net); % 计算输出, 使用符号函数
41.
```

```
42.         % 计算误差
43.         error = Y(i) - output;
44.         total_error = total_error + abs(error);
45.
46.         % 更新权值
47.         if error ~= 0
48.             w = w + c * error * X(i, :); % 学习率调整后的权值更新
49.         end
50.     end
51.
52.     % 计算分类准确率
53.     predictions = sign(X * w);
54.     accuracy = sum(predictions == Y) / length(Y);
55.     accuracy_history = [accuracy_history, accuracy];
56.
57.     % 检查是否收敛
58.     if total_error < tolerance
59.         fprintf('Training converged after %d epochs.\n', epoch);
60.         break;
61.     end
62. end
63.
64. % 最终权值
65. fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));
66.
67. % 绘制学习率和准确率变化
68. figure;
69.
70. % 学习率变化图
71. subplot(2, 1, 1);
72. plot(1:length(learning_rate_history), learning_rate_history, 'b-',
       'LineWidth', 2);
73. xlabel('Epoch');
74. ylabel('Learning Rate');
75. title('Learning Rate vs. Epoch');
76. grid on;
77.
78. % 准确率变化图
79. subplot(2, 1, 2);
80. plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
       2);
81. xlabel('Epoch');
82. ylabel('Accuracy');
83. title('Accuracy vs. Epoch');
```

```

84. grid on;
85. % 绘制分类图
86. figure;
87. hold on;
88.
89. % 绘制数据点
90. gscatter(data(:, 1), data(:, 2), Y, 'rb', 'o', 8);
91.
92. % 绘制决策边界
93. % 计算边界:  $w_1x_1 + w_2x_2 + b = 0 \Rightarrow x_2 = -(w_1/w_2)x_1 - (b/w_2)$ 
94. x1 = linspace(min(data(:, 1)), max(data(:, 1)), 100);
95. x2 = - (w(1)/w(2)) * x1 - (w(3)/w(2));
96.
97. plot(x1, x2, 'k--', 'LineWidth', 2); % 绘制决策边界
98. xlabel('Feature 1');
99. ylabel('Feature 2');
100. title('Final Classification with Decision Boundary');
101. legend('Class 1', 'Class -1', 'Decision Boundary');
102. grid on;
103. hold off;

```

### 3.4 *ReduceLROnPlateau*代码

```

1. % 初始化参数
2. w = [0.75; 0.5; -0.6]; % 权值初始化
3. c0 = 0.1; % 初始学习率
4. max_epochs = 500; % 最大训练轮数
5. tolerance = 1e-9; % 收敛容忍度
6.
7. % 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）
8. data = [1.0, 1.0, 1;
9.         9.4, 6.4, -1;
10.        2.5, 2.1, 1;
11.        8.0, 7.7, -1;
12.        0.5, 2.2, 1;
13.        7.9, 8.4, -1;
14.        7.0, 7.0, -1;
15.        2.8, 0.8, 1;
16.        1.2, 3.0, 1;
17.        7.8, 6.1, -1]; % 训练数据
18.
19. train_ratio = 0.8; % 训练集占比
20. num_samples = size(data, 1);

```

```

21. num_train = floor(num_samples * train_ratio);
22. train_indices = randperm(num_samples, num_train);
23. val_indices = setdiff(1:num_samples, train_indices);
24.
25. data_train = data(train_indices, :);
26. data_val = data(val_indices, :);
27.
28. % 提取训练集输入和输出，并加入偏置项
29. X_train = [data_train(:, 1:2), ones(size(data_train, 1), 1)];
30. Y_train = data_train(:, 3);
31.
32. % 提取验证集输入和输出，并加入偏置项
33. X_val = [data_val(:, 1:2), ones(size(data_val, 1), 1)];
34. Y_val = data_val(:, 3);
35.
36. % 设置 ReduceLROnPlateau 相关参数
37. patience = 3; % 当验证准确率不再提升，等待多少轮后降低学习率
38. factor = 0.5; % 学习率降低因子
39. min_lr = 1e-6; % 最小学习率，防止学习率降得过低
40. best_accuracy = 0; % 初始最佳验证准确率设为 0
41. counter = 0; % 计数器，用于记录验证准确率未提升的轮数
42.
43. % 记录学习率和误差历史
44. learning_rate_history = [];
45. accuracy_history = [];
46.
47. % 训练循环
48. for epoch = 1:max_epochs
49.     % 更新学习率（基于 ReduceLROnPlateau 策略）
50.     if counter >= patience
51.         c0 = max(c0 * factor, min_lr); % 降低学习率，确保不低于最小学习率
52.         counter = 0;
53.     end
54.     c = c0; % 当前学习率
55.     learning_rate_history = [learning_rate_history, c]; % 记录学习率
56.
57.     % 初始化训练误差和验证误差
58.     total_train_error = 0;
59.     total_val_error = 0;
60.
61.     % 训练集前向传播与权值更新
62.     for i = 1:size(X_train, 1)
63.         % 前向传播
64.         net = w' * X_train(i, :); % 计算净输入

```



```

65.         output = sign(net); % 计算输出, 使用符号函数
66.
67.         % 计算误差
68.         error = Y_train(i) - output;
69.         total_train_error = total_train_error + abs(error);
70.
71.         % 更新权值
72.         if error ~= 0
73.             w = w + c * error * X_train(i, :); % 学习率调整后的权值更新
74.         end
75.     end
76.
77.     % 验证集前向传播, 计算验证准确率
78.     predictions_val = sign(X_val * w);
79.     accuracy_val = sum(predictions_val == Y_val) / length(Y_val);
80.     accuracy_history = [accuracy_history, accuracy_val];
81.
82.     % 根据验证准确率更新最佳准确率与计数器
83.     if accuracy_val > best_accuracy
84.         best_accuracy = accuracy_val;
85.         counter = 0;
86.     else
87.         counter = counter + 1;
88.     end
89.
90.     % 检查是否收敛 (这里简单根据训练误差判断, 可优化)
91.     if total_train_error < tolerance
92.         fprintf('Training converged after %d epochs.\n', epoch);
93.         break;
94.     end
95. end
96.
97. % 最终权值
98. fprintf('Final weights: %.4f, %.4f, %.4f\n', w(1), w(2), w(3));
99.
100. % 绘制学习率和准确率变化
101. figure;
102.
103. % 学习率变化图
104. subplot(2, 1, 1);
105. plot(1:length(learning_rate_history), learning_rate_history, 'b-',
      'LineWidth', 2);
106. xlabel('Epoch');
107. ylabel('Learning Rate');

```

```

108. title('Learning Rate vs. Epoch');
109. grid on;
110.
111. % 准确率变化图
112. subplot(2, 1, 2);
113. plot(1:length(accuracy_history), accuracy_history, 'r-', 'LineWidth',
      2);
114. xlabel('Epoch');
115. ylabel('Accuracy');
116. title('Accuracy vs. Epoch');
117. grid on;
118.
119. % 绘制分类图
120. figure;
121. hold on;
122.
123. % 绘制数据点
124. gscatter(data(:, 1), data(:, 2), data(:, 3), 'rb', 'o', 8);
125.
126. % 绘制决策边界
127. % 计算边界:  $w_1x_1 + w_2x_2 + b = 0 \Rightarrow x_2 = -(w_1/w_2)x_1 - (b/w_2)$ 
128. x1 = linspace(min(data(:, 1)), max(data(:, 1)), 100);
129. x2 = - (w(1)/w(2)) * x1 - (w(3)/w(2));
130.
131. plot(x1, x2, 'k--', 'LineWidth', 2); % 绘制决策边界
132. xlabel('Feature 1');
133. ylabel('Feature 2');
134. title('Final Classification with Decision Boundary');
135. legend('Class 1', 'Class -1', 'Decision Boundary');
136. grid on;
137. hold off;

```

## 4. 实验结果

### 4.1 固定学习率结果

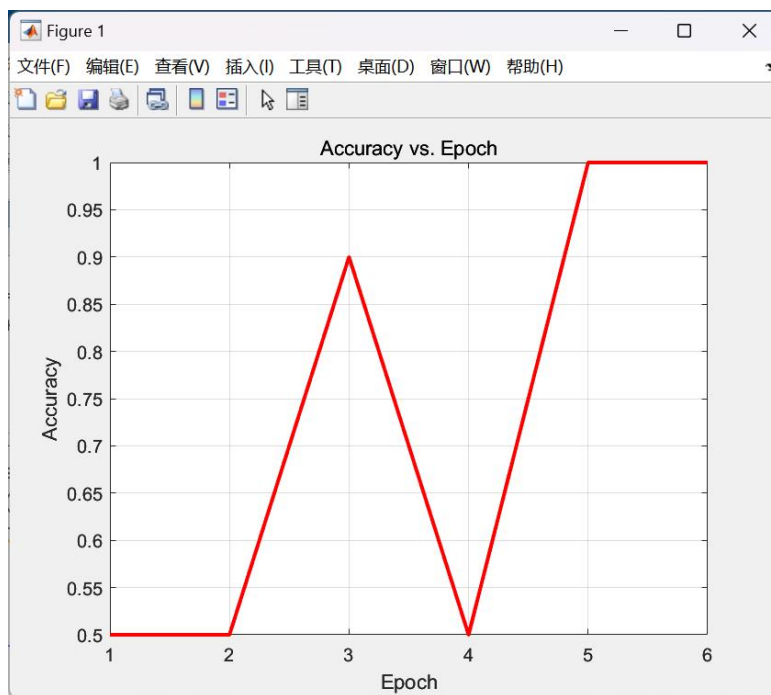


图 6. 固定学习率的准确性变化曲线

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

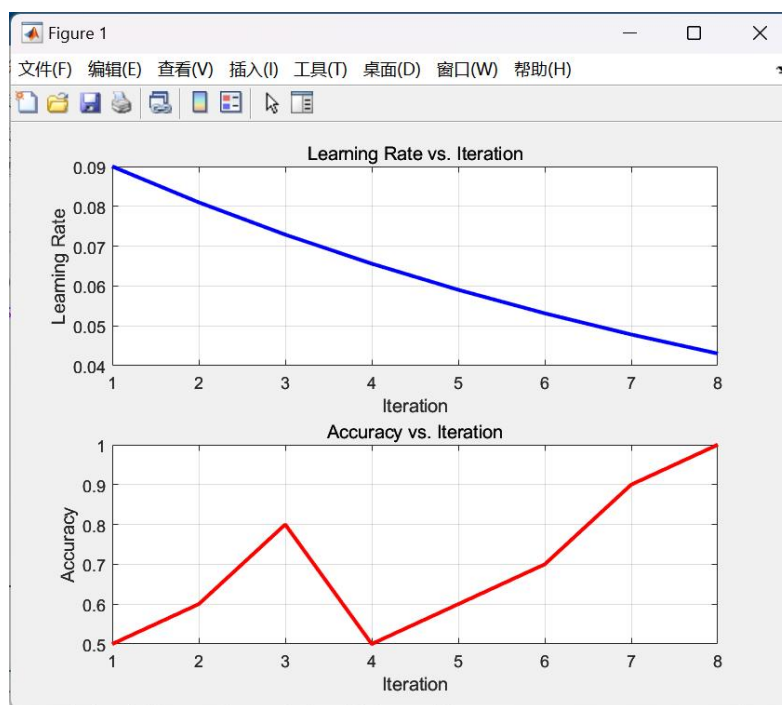


图 8. *StepLR* 的学习率与准确率的变化曲线

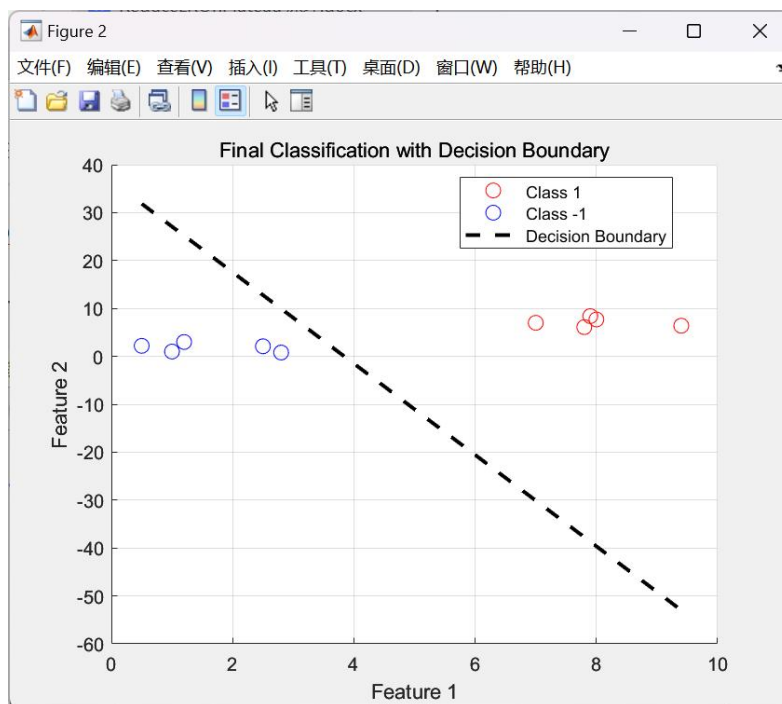


图 9. *StepLR* 产生的决策边界

```
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

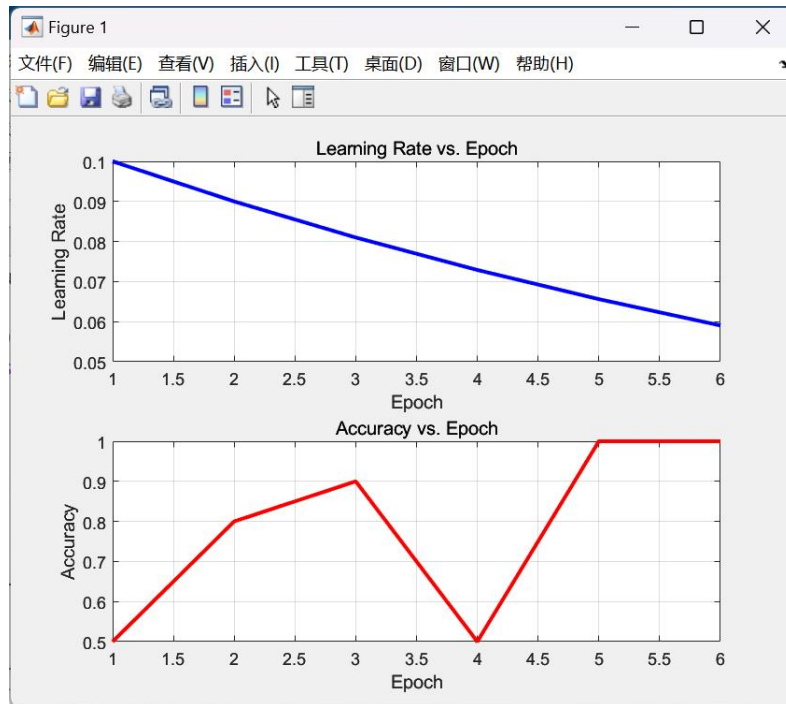


图 11. *ExponentialLR* 的学习率与准确率的变化曲线

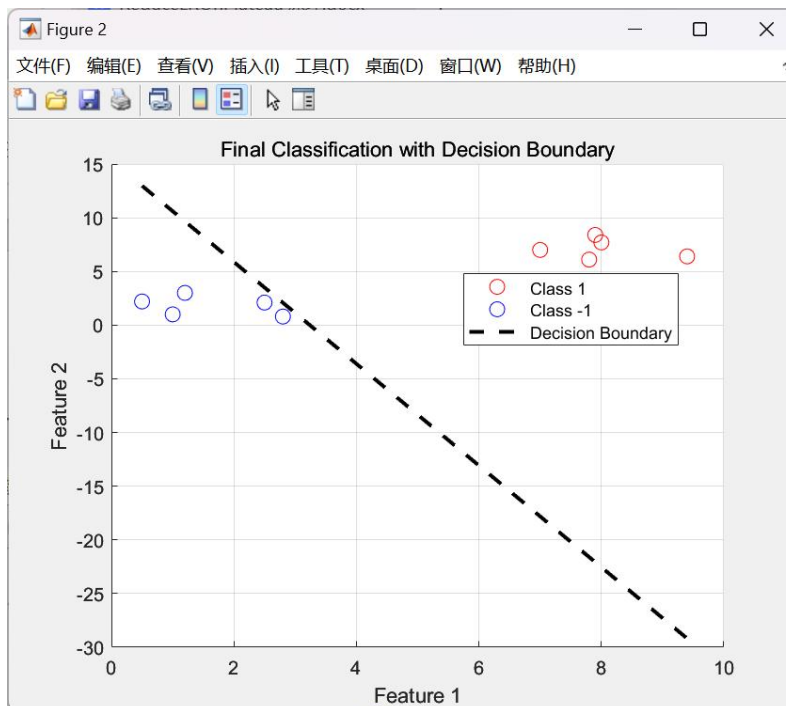


图 12. *ExponentialLR* 产生的决策边界

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

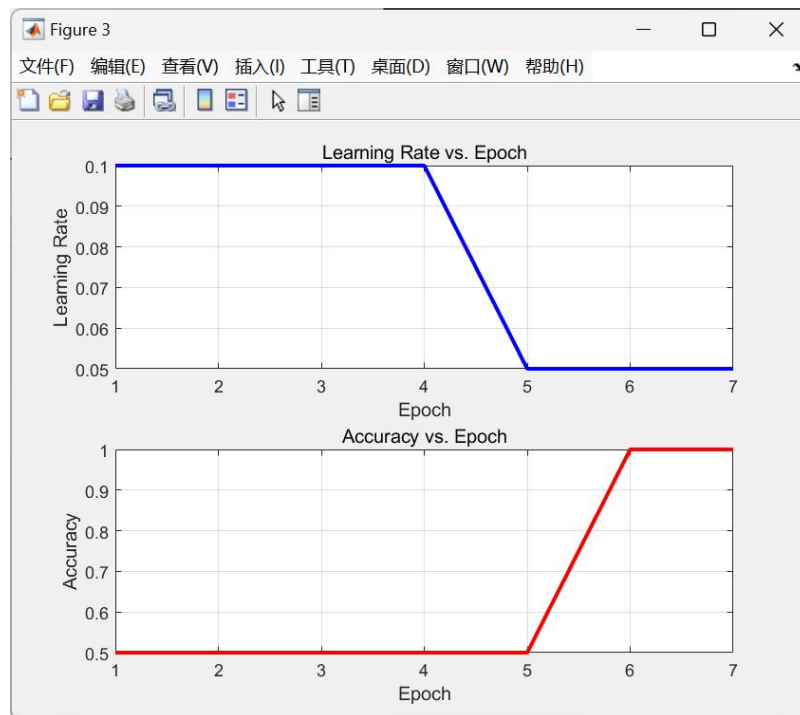


图 14. *ReduceLROnPlateau* 的学习率与准确率的变化曲线

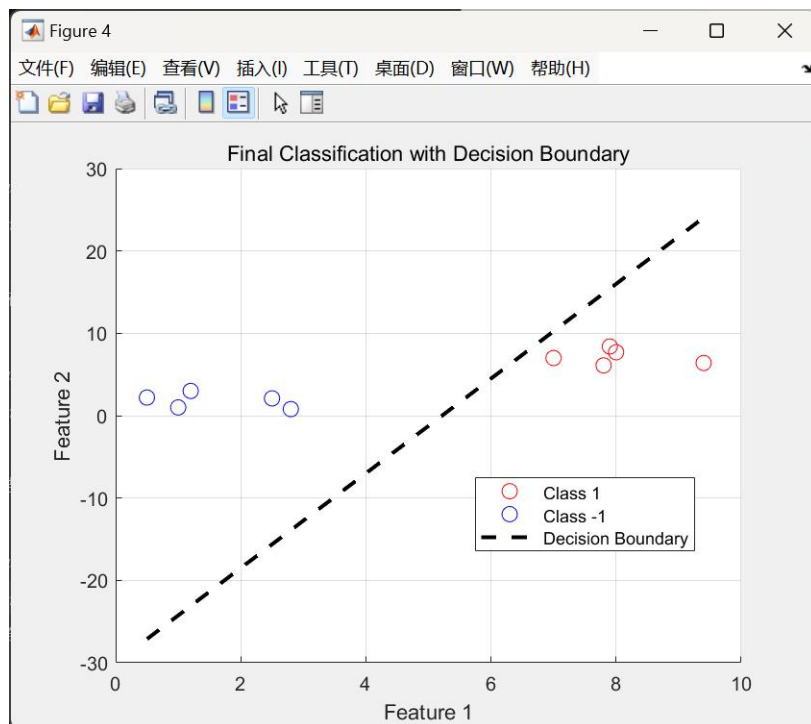


图 15. *ReduceLROnPlateau* 产生的决策边界

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	<i>StepLR</i>	<i>ExponentialLR</i>	<i>ReduceLROnPlateau</i>
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。