# 神经网络分类器

姓名：阮思琳 学号：2022302131290

本次作业的目的是使用两层隐含层的神经网络对ROI数据进行分类，并将训练好的模型应用于影像数据分类。具体目标包括数据预处理、模型训练、模型验证与测试以及对影像数据的分类处理。

##### 原理

1. **神经网络**

神经网络由输入层、一个或多个隐藏层和输出层组成。在本实习中，采用了两层隐藏层的前馈神经网络模型：

输入层：接收输入数据，每个神经元对应一个输入特征。

隐藏层：包括第一层隐藏层（10个神经元）和第二层隐藏层（15个神经元），每个神经元使用激活函数进行非线性变换。

输出层：生成分类结果，每个神经元对应一个类别。

1. **前向传播**

在前向传播过程中，输入数据通过各层的神经元，经过加权求和和激活函数的非线性变换，最终得到输出结果。公式如下：

**第一层隐藏层**：

**第二层隐藏层**：

**输出层**：

其中，分别是输入层到第一层隐藏层、第一层隐藏层到第二层隐藏层、第二层隐藏层到输出层的权重矩阵，是激活函数（本实习中采用sigmoid函数），softmax函数用于多分类任务的输出层激活。

1. **反向传播**

反向传播算法用于计算损失函数对每层权重的梯度，并通过梯度下降法更新权重，以最小化损失函数。主要步骤如下：

**输出层**误差计算：

其中，是实际标签的one-hot向量。

**输出层**梯度计算：

**第二层隐藏层**误差计算：

**第一层隐藏层**误差计算：

其中，是激活函数的导数。

**权重更新**：

其中，是学习率。

1. **损失函数**

本实习中使用交叉熵损失函数来评估模型的分类误差，公式如下：

其中，是类别数，是预测概率，是实际标签的one-hot编码。

通过上述过程，逐步调整模型的权重参数，使得损失函数逐渐减小，最终得到一个较为准确的分类模型。

##### 编程过程

1. 数据准备

将将ROI数据整合成一个矩阵`roidata`，每行代表一个样本，前6列是波段值，第7列是类别标签。

1. roidata=[];
2. for i=1:roi.NumOfROIs
3. roidata=[roidata;roi.data{i},i\*ones(roi.NumOfPerROIs(i),1)];
4. end

划分训练集、验证集、测试集：

1. ratioTraining = 0.7;
2. redioValidation = 0.15;
3. radioTesting = 0.15;
4. numTraining = int32(N \* ratioTraining);
5. numValidation = int32(N \* redioValidation);
6. numTesting = N - numTraining - numValidation;
7. dataset\_train\_x = roidata\_x(1:numTraining, :);
8. dataset\_train\_y = roidata\_y(1:numTraining, :);
9. dataset\_validation\_x = roidata\_x(numTraining+1:numTraining+numValidation, :);
10. dataset\_validation\_y = dataset\_train\_y(numTraining+1:numTraining+numValidation, :);
11. dataset\_test\_x = roidata\_x(numTraining+numValidation+1:end, :);
12. dataset\_test\_y = roidata\_y(numTraining+numValidation+1:end, :);
13. 神经网络训练

初始化权重：

1. W1 = 2 \* rand(cell1, bands) - 1;
2. W2 = 2 \* rand(cell2, cell1) - 1;
3. W3 = 2 \* rand(class, cell2) - 1;

训练过程：

在每个训练周期（epoch）中，使用梯度下降法进行前向传播和反向传播，更新权重。

1. for epoch = 1:numEpoch
2. for i = 1:numTraining
3. x = dataset\_train\_x(i, :)';
4. d = dataset\_train\_y(i, :)';
5. % 前向传播
6. v1 = W1 \* x;
7. y1 = sigmoid(v1);
8. v2 = W2 \* y1;
9. y2 = sigmoid(v2);
10. v = W3 \* y2;
11. y = softmax(v);
12. % 反向传播
13. e = y - d;
14. delta = e;
15. e2 = W3' \* delta;
16. delta2 = sigmoid\_derivative(v2) .\* e2;
17. e1 = W2' \* delta2;
18. delta1 = sigmoid\_derivative(v1) .\* e1;
19. % 更新权重
20. W1 = W1 - learning\_rate \* delta1 \* x';
21. W2 = W2 - learning\_rate \* delta2 \* y1';
22. W3 = W3 - learning\_rate \* delta \* y2';
23. end
24. % 验证集验证
25. sum\_e = 0;
26. for i = 1:numValidation
27. x = dataset\_validation\_x(i, :)';
28. d = dataset\_validation\_y(i, :)';
29. v1 = W1 \* x;
30. y1 = sigmoid(v1);
31. v2 = W2 \* y1;
32. y2 = sigmoid(v2);
33. v = W3 \* y2;
34. y = softmax(v);
35. sum\_e = sum\_e + error(y, d);
36. end
37. fprintf("epoch=%d\n", epoch)
38. fprintf("error=%d\n", sum\_e)
39. randIndex = randperm(numTraining);
40. dataset\_train\_x = dataset\_train\_x(randIndex, :);
41. dataset\_train\_y = dataset\_train\_y(randIndex, :);
42. e\_line(epoch) = sum\_e;
43. end
44. 模型评估

绘制验证集交叉熵总和曲线：

1. hold on
2. plot(e\_line)
3. ylabel("验证集上的交叉熵总和")
4. xlabel("训练次数")
5. hold off

测试集评估：

计算混淆矩阵和整体精度。

1. confusionMatrix = zeros(class, class);
2. for i = 1:numTesting
3. x = dataset\_test\_x(i, :)';
4. d = dataset\_test\_y(i, :)';
5. v1 = W1 \* x;
6. y1 = sigmoid(v1);
7. v2 = W2 \* y1;
8. y2 = sigmoid(v2);
9. v = W3 \* y2;
10. y = softmax(v);
11. [~, rst] = max(y);
12. [~, real] = max(d);
13. confusionMatrix(rst, real) = confusionMatrix(rst, real) + 1;
14. end
15. precision = sum(diag(confusionMatrix)) / sum(confusionMatrix, "all");
16. fprintf("混淆矩阵：\n")
17. disp(confusionMatrix)
18. fprintf("整体精度：%f\n", precision)
19. 影像分类

读取和预处理影像：

1. tif = imread("tmpicture.tif");
2. [M, N, bands] = size(tif);
3. tif = double(reshape(tif, M\*N, bands));
4. tif = (tif - means) ./ stds;
5. class = zeros(M\*N, 1);

影像分类：

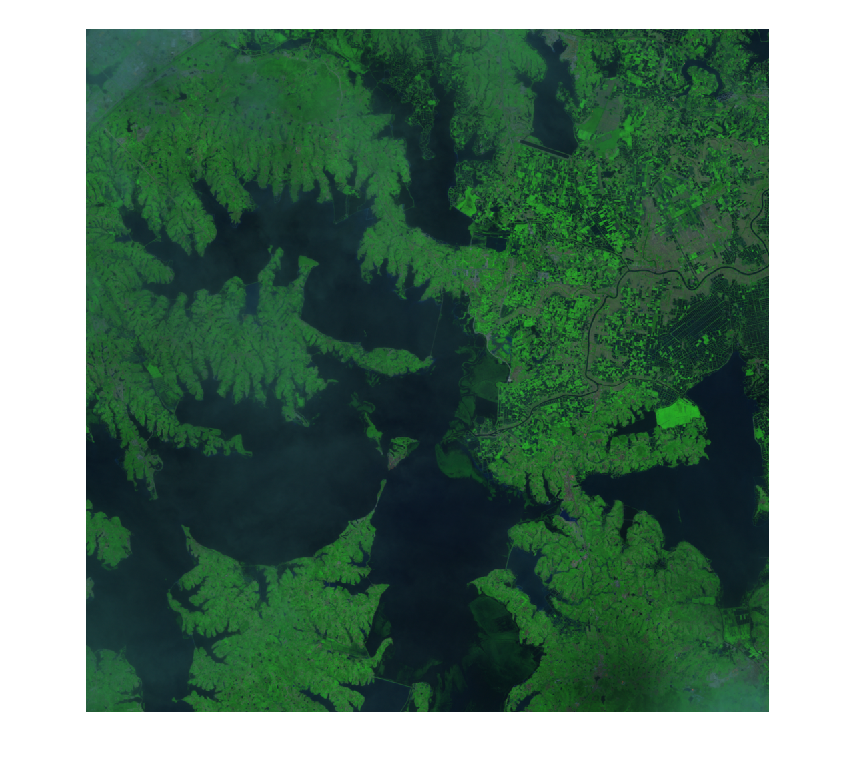
1. for i = 1:M\*N
2. x = tif(i, :)';
3. v1 = W1 \* x;
4. y1 = sigmoid(v1);
5. v2 = W2 \* y1;
6. y2 = sigmoid(v2);
7. v = W3 \* y2;
8. y = softmax(v);
9. [~, class(i)] = max(y);
10. end
11. class = reshape(class, M, N);

构建分类结果图像：

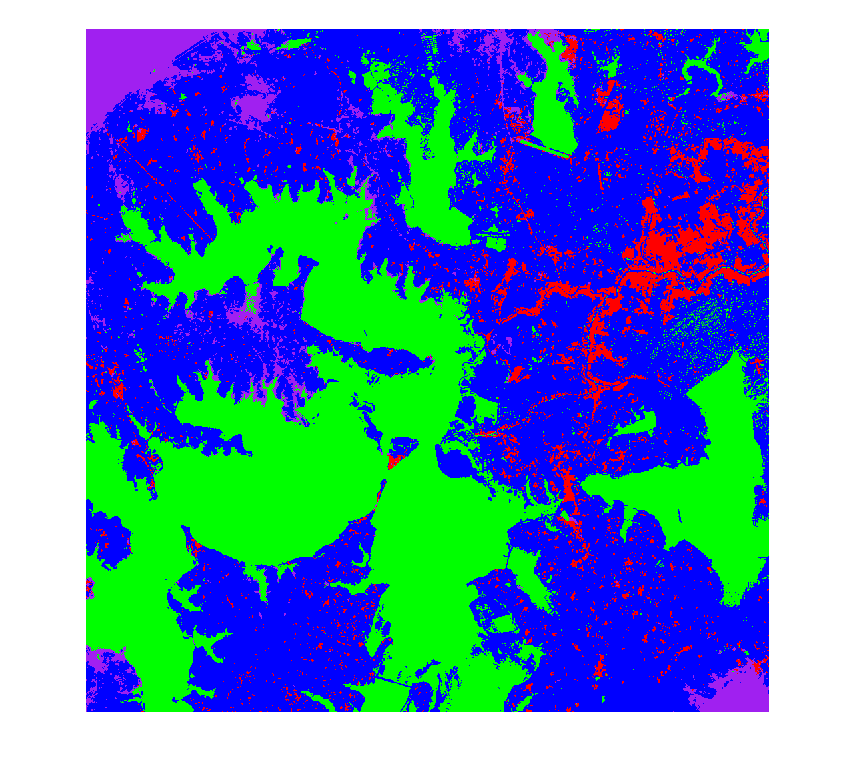
1. colors\_R = [0, 255, 0, 160, 0];
2. colors\_G = [255, 0, 0, 32, 0];
3. colors\_B = [0, 0, 255, 240, 0];
4. new\_tif = cat(3, colors\_R(class), colors\_G(class), colors\_B(class));
5. new\_tif = uint8(new\_tif);
6. figure, imshow(new\_tif)

##### 结果分析

1. 假彩色图像



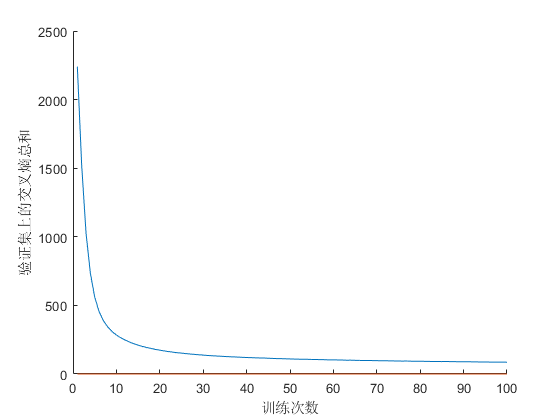
1. 分类处理图像



**分类类别：**

* Water（水域）——绿色
* Land（陆地）——蓝色
* Towns（城镇）——红色
* City（城镇）——紫色

1. 交叉熵总和曲线



**结果分析：**

通过实验，神经网络模型在训练过程中逐渐减少了验证集上的交叉熵，总体误差逐渐减小。测试集上的混淆矩阵显示，模型能够较好地分类不同类别的数据，整体精度达到了预期的效果。在影像分类部分，分类结果图像清晰展示了不同类别在影像中的分布情况。

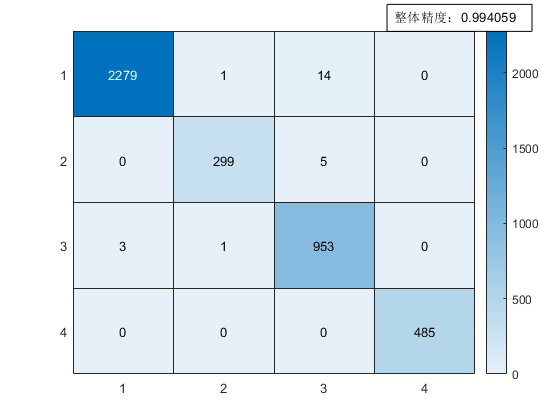
优点：

* 整体分类效果良好，分类性能较优；
* 图像中的水域和陆地区域被有效地识别并保留，轮廓和细节清晰；

缺点：

* 城镇（红色）分类精度较低，存在明显的噪声或伪影；
* 图像右下方的水域存在被误分类为陆地的情况；

1. 混淆矩阵及精度分析



* 精度值：99.41%
* 根据混淆矩阵计算得到的总精度表明，模型正确分类了大约 99.41% 的样本。这个精度值反映了模型的整体性能良好。

##### 遇见问题与改进方法

* 数据归一化问题

在处理ROI数据时，发现不同波段的数据范围差异较大，如果不进行归一化处理，可能会导致神经网络训练过程中的权重更新不稳定，影响模型的收敛性和分类效果。

改进方法：

在将数据输入神经网络之前，使用`zscore`函数对数据进行了归一化处理，确保每个波段的数据分布均匀，均值为0，标准差为1。归一化处理使得训练过程更加稳定，提升了模型的性能。

[roidata(:,1:bands), means, stds] = zscore(roidata(:,1:bands));

* 标签处理问题

原始ROI数据中的类别标签是标量形式，但在神经网络训练过程中需要使用one-hot编码的形式。如果直接使用原始标签，可能会导致网络无法正确学习分类任务。

改进方法：

通过遍历每个样本，将其类别标签转换为one-hot编码形式，这样每个样本的标签向量中只有一个元素为1，其余为0。

* 模型训练收敛问题

在初始训练过程中，发现模型的收敛速度较慢，且容易陷入局部最优解，导致模型无法获得较好的分类性能。

改进方法：

1. 学习率调整：通过多次实验调整学习率，将学习率设置为较小的值（0.001），确保每次权重更新步长较小，避免训练过程中的震荡。

2. 权重初始化：使用均匀分布随机初始化权重，确保初始权重分布在合理范围内，避免神经网络初始状态过于偏向某个方向。

W1 = 2 \* rand(cell1, bands) - 1;

W2 = 2 \* rand(cell2, cell1) - 1;

W3 = 2 \* rand(class, cell2) - 1;