图标

描述已自动生成

**计算机视觉**

**上机实验一报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **专业** | **：** | **计算机电脑** |  |
| **班级** | **：** | **CS2002** |  |
| **学号** | **：** | **I201920029** |  |
| **姓名** | **：** | **冯就康** |  |
| **电话** | **：** | **15623031879** |  |
| **邮箱** | **：** | **sokhorng526@gmail.com** |  |

Contents

[I. 简介 3](#_Toc153802983)

[1. 前馈神经网络的简介 3](#_Toc153802984)

[2. 前馈神经网络的重要性及其应用 3](#_Toc153802985)

[II. 前馈神经网络的架构 4](#_Toc153802986)

[III. 模型训练过程 5](#_Toc153802987)

[1. 数据准备： 5](#_Toc153802988)

[2. 损失函数和优化器： 5](#_Toc153802989)

[3. 训练循环： 5](#_Toc153802990)

[4. 权重调整： 5](#_Toc153802991)

[5. 迭代过程： 5](#_Toc153802992)

[IV. 实验结果以及分析 6](#_Toc153802993)

[1. 实验结果 6](#_Toc153802994)

[2. 实验分析 6](#_Toc153802995)

[V. 实验总结 8](#_Toc153802996)

[1. 实验总结： 8](#_Toc153802997)

# 简介

## 前馈神经网络的简介

前馈神经网络（FNN）是最简单的人工神经网络之一。前馈神经网络的简单如下：

* **信息流方向**：在 FNN 中，信息仅沿一个方向移动（向前），从输入节点经过隐藏节点（如果有），到达输出节点。网络中不存在循环或环路。
* **层**：FNN 通常由多个层组成：一个输入层、一个或多个隐藏层和一个输出层。输入层接收输入数据，隐藏层对输入执行转换，输出层产生最终输出。
* **权重**：网络中神经元之间的连接与权重相关，权重描述了连接的强度。这些权重在训练过程中进行调整，以尽量减少网络输出与实际输出之间的差异。
* **函数逼近**：前馈神经网络的目的是逼近函数。它们通过函数 f(x; θ) 将输入 x 映射到输出 y，其中 θ 表示网络的权重。
* **训练**：现代前馈网络使用反向传播方法进行训练，该方法调整网络的权重以最小化网络输出与实际输出之间的误差。
* **应用**：FNN 是许多其他类型神经网络的基础，并用于各种应用，包括模式识别、分类、非线性回归和函数逼近。

## 前馈神经网络的重要性及其应用

前馈神经网络 (FNN) 是人工智能和机器学习领域的基础。以下是它们的重要性及其一些应用：

**重要性**：

* **高级网络的基础**：FNN 是更高级神经网络架构的基础。
* **函数逼近**：FNN 用于逼近函数，将输入 x 映射到输出 y。
* **模式识别**：FNN 有助于从各种类型的输入中识别和创建模式。

**应用**：

* **对象检测**：FNN 用于照片中的对象检测，如 Google Photos 等应用程序中所示。
* **模式识别和分类**：FNN 帮助软件开发人员进行模式识别和分类。
* **非线性回归和函数逼近**：FNN 用于非线性回归和函数逼近。
* **搜索引擎和机器翻译**：依赖 FNN 的深度学习技术用于搜索引擎和机器翻译。

FNN 的多功能性和鲁棒性使其成为各个领域的强大工具。

# 前馈神经网络的架构

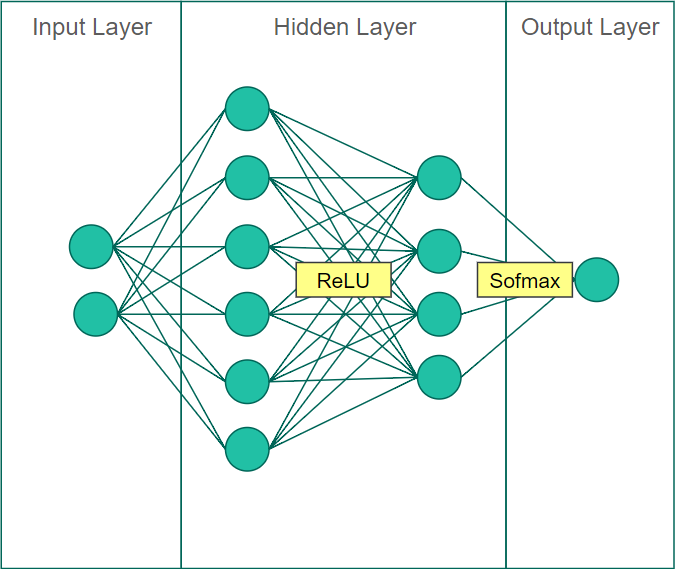
前馈神经网络的架构如图1：

图 1. 前馈神经网络架构

让我们分解一下 FeedForwardNet 模型：

**架构**：

使用 PyTorch 的 nn.Module 构建的前馈神经网络 (FNN)。由两个完全连接的（线性）层、一个 ReLU 激活函数和前向传递中的 softmax 函数组成。

**层数**：

**输入层**：网络的输入层由 nn.Linear(2, 16) 定义。这意味着网络采用大小为 2 的输入并将其转换为大小为 16 的输出。

**隐藏层**：网络的隐藏部分由 ReLU 激活函数 (nn.ReLU()) 和全连接层 (nn.Linear(16, 4)) 组成。 ReLU 函数将非线性引入模型，使其能够学习更复杂的模式。全连接层将前一层的 16 维输出转换为 4 维输出。

**输出层**：网络的输出层由nn.Linear(4, 1)定义。该层将前一层的 4 维输出转换为 1 维输出。然后将 softmax 函数应用于前向传递中的该输出，确保输出值的概率之和为 1。

**权重和偏差**：

在网络中，权重和偏差是线性层内的参数。这些参数是随机初始化的，并通过训练过程学习。训练过程的目标是调整这些权重和偏差，以最小化网络预测与实际值之间的差异。权重决定网络中神经元之间的连接强度，而偏差则通过允许激活函数向左或向右移动来实现模型的灵活性。

# 模型训练过程

前馈神经网络的训练过程：

## 数据准备：

首先将数据分为训练集和测试集。然后，将数据转换为 PyTorch 张量，并为训练和测试数据创建数据集和数据加载器。

## 损失函数和优化器：

* 将损失函数定义为CrossEntropyLoss，常用于多类分类问题。
* 选择 Adam 作为优化器。

## 训练循环：

训练模型指定的轮数。在每个时期，迭代所有批次的训练数据。

## 权重调整：

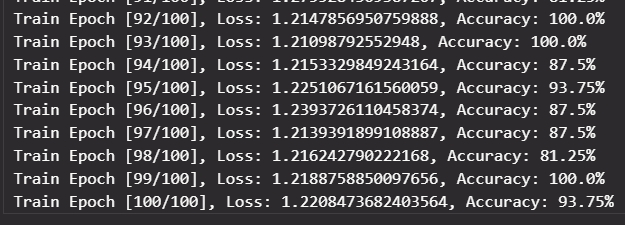
* **前向传递**：在每次迭代中，将输入传递给模型以获得输出。然后，计算输出和目标之间的损失。
* **向后传递和优化**：通过调用 loss.backward() 执行反向传播，计算损失相对于模型参数（即权重和偏差）的梯度。然后，调用optimizer.step()，它根据计算的梯度更新模型参数。

## 迭代过程：

* 数据集多次通过网络，每个时期一次。每个通道由前向通道（计算预测输出和损失）和后向通道（计算梯度并更新模型参数）组成。
* 数据集通过网络的次数（即纪元数）是一个可以调整的超参数。更多的 epoch 可能会导致训练集上更好的性能，但太多的 epoch 可能会导致模型过度拟合。

# 实验结果以及分析

## 实验结果

100个Epochs训练过程的损失函数和准确率：

测试集的损失函数和准确率：

## 实验分析

我们讨论一下模型的最终训练和测试精度：

**训练准确度：**

最终训练准确率在第100个epoch时为93.75%。这意味着模型在训练集的最后一个时期正确地分类的准确率为93.75%。

然而，值得注意的是，训练的准确性会随着时代的变化而波动。例如，在第90、92和93个时期达到100%，但在第98个时期下降到81.25%。这种波动可能是由于mini-batch梯度下降引入的随机性和模型的复杂性造成的。

**测试准确度：**

测试准确率为91.0%。这意味着模型在测试集中正确分类了 91.0%。

测试精度略低于最终训练精度。这是正常且符合的预期，因为模型在训练期间从未见过测试数据。我们的目标是使测试精度尽可能接近训练精度，以确保模型能够很好地泛化到未见过的数据。

**过拟合和欠拟合：**

如果训练精度明显高于测试精度，则可能是过度拟合的迹象，这意味着模型在训练数据上表现良好，但在未见过的数据上表现不佳。

如果两个准确度都很低，则可能是拟合不足的迹象，这意味着模型太简单，无法学习数据中的潜在模式。

在我们的情况下，该模型似乎表示得很好，因为测试精度接近训练精度。然而，我们还需要监控训练准确性的波动。如果它继续大幅波动，则可能表明学习率太高，或者模型太复杂。

# 实验总结

## 实验总结：

前馈神经网络模型最终训练准确率达到93.75%，测试准确率达到91.0%。这表明该模型能够有效地从训练数据中学习，并很好地推广到未见过的数据。

训练准确率在各个时期之间波动，有时达到 100%，但有时会下降到 81.25%。这表明学习过程可能会受到学习率或模型复杂性等因素的影响。

训练集和测试集的准确率证明了前馈神经网络模型在给定任务中的有效性，显示了此类模型在解决类似问题方面的潜力。

训练精度的波动表明模型的超参数或架构可能还有进一步优化的空间，因为它可以提高模型性能和稳定性。