全连接神经网络

0 概述

实现了可以分类CIFAR-10数据集的以下几个网络：

1. 全连接神经网络无优化
2. 进行了L2正则化的全连接神经网络
3. 使用了动量法的全连接神经网络
4. 结合了动量法和L2正则化的全连接神经网络

1 无优化全连接神经网络

1.1 函数设计

本项目中实现了一个简单的全连接神经网络，主要包含以下功能函数：

1. **数据加载**：`load_cifar10(path)` 函数用于加载 CIFAR-10 数据集，使用 Python 的 `pickle` 模块处理数据。
2. **前向传播**：
   - `fully_connected(X, W, b)`：计算全连接层的输出。
   - `relu(X)`：实现 ReLU 激活函数。
   - `softmax(X)`：用于计算输出层的概率分布。
3. **损失计算**：
   - `cross_entropy_loss(y_pred, y_true)`：计算交叉熵损失，用于评估模型预测的准确性。
4. **准确率计算**：
   - `compute_accuracy(y_pred, y_true)`：根据预测结果和真实标签计算模型的准确率。
5. **反向传播**：

- `backward(X, y_true, W1, b1, W2, b2, hidden_output, y_pred, learning_rate)`：计算梯度并更新权重和偏置。

6. **参数保存与加载**：
   - `save_parameters(parameter2save, filename='parameters')` 和 `load_parameters(filename='parameters')` 用于保存和加载模型参数，以便于后续测试。

1.2 主程序设计

主程序部分采用双层循环结构，外层循环控制训练的 epoch 数量，内层循环按批次处理训练数据。在每个 epoch 中，模型进行以下步骤：

1. **数据批次处理**：将训练数据分成小批次，以提升训练效率。
2. **前向传播**：计算每个批次的隐藏层输出和最终输出，获得模型的预测结果。
3. **损失计算**：根据预测结果和真实标签计算损失。
4. **反向传播**：更新模型的权重和偏置，优化网络性能。

在每个 epoch 结束时，还会计算并输出测试集的准确率，并将结果记录到 `accuracy_array` 中，最后绘制准确率与 epoch 数的关系图。

通过这种结构设计，程序能够有效地训练神经网络，并在训练过程中监控性能变化。

2 使用动量法的全连接神经网络

使用动量法优化的全连接神经网络的实现。可以有效加速收敛并减小振荡。

注：与无优化网络相同的部分不再赘述

2.1 动量法部分

增加了动量的概念来对参数进行调整，以下是实现动量法的主要内容：

1. **参数初始化**：
   - 新增了用于存储动量的变量 `V_W1`, `V_b1`, `V_W2`, `V_b2`，初始化为零。

```
V_W1 = 0
V_b1 = 0
V_W2 = 0
V_b2 = 0
```

2. **反向传播函数**：
   - `backward()` 函数中，更新了权重和偏置的方式，以包括动量更新：

```
V_W1 = gamma * V_W1 + learning_rate * dL_dW1
V_b1 = gamma * V_b1 + learning_rate * dL_db1
V_W2 = gamma * V_W2 + learning_rate * dL_dW2
V_b2 = gamma * V_b2 + learning_rate * dL_db2
```

   - 然后通过减去这些动量更新值来更新权重和偏置：

```
W1 -= V_W1
b1 -= V_b1
W2 -= V_W2
b2 -= V_b2
```

3. **动量系数**：
    - 在参数初始化时定义了动量系数 `gamma`，设置为 `0.9`，以控制动量的影响。

3 使用L2正则化的全连接神经网络

使用L2正则化优化的全连接神经网络的实现。L2正则化旨在通过在损失函数中增加权重的惩罚项，以防止过拟合并提高模型的泛化能力。

注：与无优化网络相同的部分不再赘述

3.1 L2正则化部分

L2正则化通过在损失函数中加入权重的平方和来实现。以下是实现L2正则化的主要内容：

1. **正则化参数**：
    - 在初始化阶段，新增了正则化参数 `lambda_L2`，设置为 `0.01`，用于控制L2正则化的强度。

```
lambda_L2 = 0.01
```

2. **反向传播函数**：
    - 在 `backward()` 函数中，修改了计算梯度的部分，以考虑L2正则化的影响：

```
dL_dW2 = hidden_output.T.dot(dL_dy_pred) - lambda_L2 * W2
dL_dW1 = X.T.dot(dL_dhidden) - lambda_L2 * W1
```

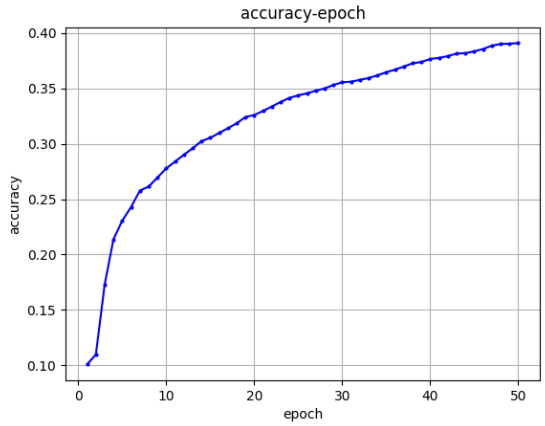
这里，通过减去权重乘以正则化参数的项来实现L2正则化。

4 融合了L2和momentum的神经网络

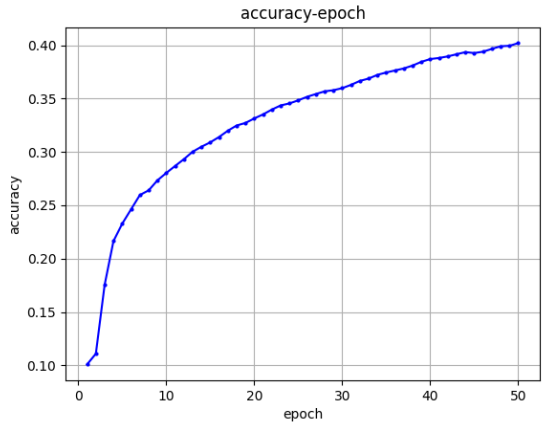
注：不再赘述

5 模型对比实验

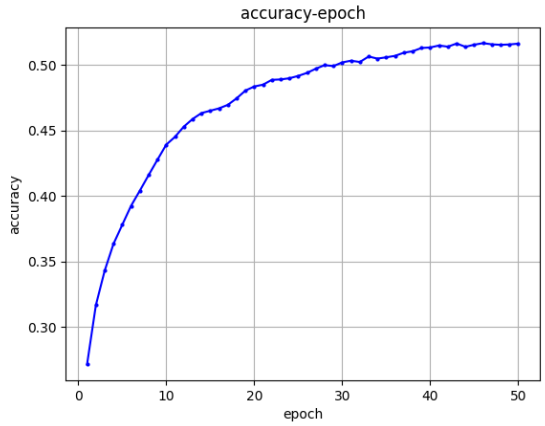
1. **无优化的全连接网络**：未使用任何正则化或优化方法的全连接网络在50个epoch后准确率约为0.38，呈现出较为缓慢的上升趋势。这种情况表明，网络可能面临过拟合或欠拟合问题，学习效果较为有限。



2. **使用L2正则化的全连接网络**：使用L2正则化的网络相比无优化网络略有提升，50个epoch后准确率接近0.40。L2正则化抑制了过拟合，使得模型泛化能力有所增强，但提升幅度不显著。



3. **使用动量法的全连接网络**：动量法加速了网络的学习，准确率迅速上升，50个epoch后达到了0.52。这表明动量法在提高训练效率和准确性方面效果显著。



4. **L2正则化与动量法融合的全连接网络**：曲线在50个epoch时并没有单纯的动量法准确率高，但值得注意的是图4的曲线显然更加平和，准确率下降的情况发生的次数更少，说明L2抑制了过拟合情况的发生，可以预想再更多的epoch训练之后，应该可以达到更好的训练结果，同时应当对这个模型进行超参数调节的交叉验证。