南開大學
Nankai University

计算机学院

机器学习导论实验报告

实验三
LeNet5

姓名：赵健坤

学号：2010535

专业：计算机科学与技术

2022 年 12 月 19 日

目录

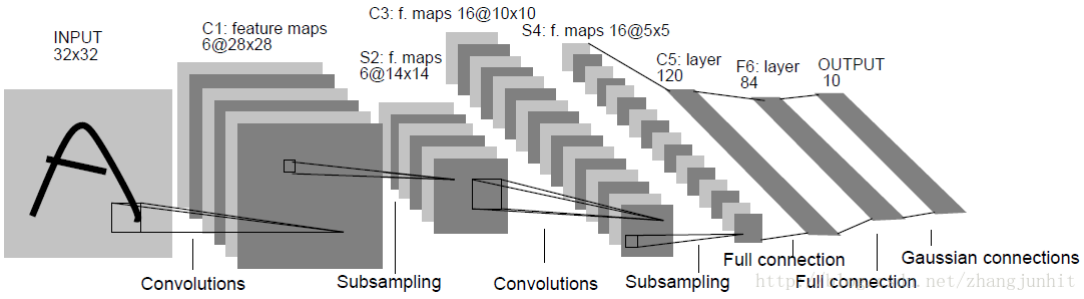

图 1.1: LeNet-5 网络结构

1 网络结构

LeNet 由特征提取模块和分类模块两部分组成，特征提取模块主要由卷积层和降采样层组成，分类模块主要由全连接层组成。与当时流行的模式识别算法不同，LeNet 为特征提取模块和分类模块均可训练的端到端模型。

参照目前深度学习领域的新进展，作者对原始 LeNet-5 做了以下改进，以提升模型性能：

1. 使用 ReLU 而非 sigmoid 作为激活函数。一方面，sigmoid 包含指数运算，会降低训练速度；另一方面，sigmoid 函数在输入绝对值较大时存在严重的梯度消失问题。

2. 使用最大池化代替降采样层。原始版本的 LeNet 的降采样层计算方法为：先对采样窗口内的数据求平均，然后乘以一个可训练的系数，并加上一个可训练的偏差。在经过此种线性层后，还要再通过一层 sigmoid 激活层。LeCun 认为，当系数较大时，这种算法可以滤去噪声。但事实上，使用 ReLU 作为激活函数时，卷积层就可达到相同效果。考虑到最大池化不包含可训练参数且能强化局部特征，作者采用最大池化代替了原文中类似卷积的降采样层。

3. 在原文中，C3 卷积层的每个卷积核只与上一层中指定的通道进行连接。原作者对此给出了两点解释：第一，能减少参数数量；第二，能破坏不同核的对称性，防止不同核关注相同的特征。这种策略事实上是一种静态的 dropout 策略。然而实验结果表明，LeNet-5 在 MNIST 数据集上并无明显的过拟合现象。因此，这种策略是不必要的，作者将 C3 与上一层的 6 个通道全部连接。

4. 作者用仿射变换层 +softmax 替代了原文中的欧氏径向基函数（RBF）输出层。RBF 事实上是将计算结果与标准图案进行像素级对比。这种算法的可解释性较强，但拟合能力事实上不如普通的仿射变换 +softmax。因此，作者使用仿射变换层 +softmax 代替了 RBF 作为输出层。

5. 作者使用 Adam 优化算法，而非原文中的平凡 SGD 算法更新权重。由于 Adam 算法结合了基于梯度二阶矩估计的 AdaGrad 算法和基于动量的 RMSProp 算法，因此能使网络更快收敛。

改进后的 LeNet-5 包含七层神经网络（如图1.1），除降采样层和输出层外，每层中都包含 ReLU 激活层：

1. C1 卷积层。
   卷积核大小为 5×5，步长为 1，无 padding，输入通道数为 1，卷积核数量（输出通道数）为 6，使用偏置项 b。
   输出图像大小变为 $28 \times 28 \times 6$。

2. S2 最大池化层。
   滤波器大小为 2×2，步长为 2。
   输出图像大小变为 $14 \times 14 \times 6$。

3. C3 卷积层。
   输入通道数为 6，卷积核数量（输出通道数）为 16，其他设置与 C1 相同。
   输出图像大小变为 $10 \times 10 \times 16$。

4. S4 降采样层。
   设置与 S2 相同。
   输出图像大小变为 $5 \times 5 \times 16$。

5. C5 卷积层（全连接层）。
   输入通道数为 16，卷积核数量（输出通道数）为 120，其他设置与 C1 相同。由于图像大小仅为 5×5，与卷积核大小相同，因此该层与将输入拉平后进行全连接完全等价。这也是作者在程序中实际的实现方式。
   输出特征向量的维度为 120。

6. F6 全连接层。输入维度为 120，输出维度为 84。

7. 输出层。输入维度为 84，输出维度为 10（类别数）。

损失函数采用交叉熵损失函数。

2 代码细节

2.1 整体架构

本次实验代码分布在以下五个文件中：

1. main.ipynb
   以 Jupiter Notebook 方式调用各子文件中的接口，顺序完成数据加载和预处理、模型初始化和训练、模型评估、训练过程及结果可视化四项工作。

2. data.py
   顺序读取 MNIST 数据集的四个文件，将原数据集中的 28×28 原始图像转换为模型输入所需的 $32 \times 32 \times 1$，并转换为与模型匹配的 float32 类型。从训练集中划分出大小为 1000 的验证集，完成数据归一化，并返回包含训练集、验证集和测试集数据及标签的字典。函数调用链为 get_mnist_data()->load_data()->load_mnist()。

3. train.py
   该文件包含模型完成模型训练的类 Solver 及完成权重更新的优化器函数 sgd 和 adam。
   优化器具有统一接口，其接收每层的权重和梯度，以及包含学习率在内的优化器超参数 config。值得注意的是，adam 需要在每次迭代时更新 m、v、t 等参数，因此优化器除返回新的权重外，还要返回新的超参数。
   Solver 类初始化时接收模型、数据集及一系列超参数，并有 train() 和 check_accuracy() 两个方法分别负责模型训练和评估。模型训练及评估均采用 minibatch 方法，调用 model.loss() 接口完

成前向传播和反向传播。
在 train() 函数中，当训练超过 10 epochs 后时，学习率按 0.7 比例下降。每个 epoch 结束后，都将调用 check_accuracy() 评估训练集和验证集准确率，并将其保存在数组中以备分析。每 10 个 epoch 后，将保存一次模型参数。

4. model.py
定义模型类 LeNet5，该类与 PyTorch 中的 nn.module 功能相似。其构造函数完成参数初始化：其中权重采用 $N(0, 0.001^2)$ 随机初始化，偏置项全部初始化为 0。loss 函数调用 layers.py 中定义的各层的 forward 和 backward 接口完成前向和反向传播，并返回损失函数值和梯度。

5. layers.py
该函数内定义卷积层、ReLU 激活层、最大池化层、全连接层的 forward 与 backward 函数，并定义 softmax 输出层。其中最重要的为卷积层的实现，作者将在2.2中详细讨论。

2.2 卷积层实现

考虑到模型不大，作者并未采用 img2col 加速卷积过程，而是使用了传统的嵌套循环，并**使用 numpy 提供的广播功能和切片功能进行循环展开**。前向、反向传播均使用两层嵌套循环遍历输出图中的每个像素。以下假定 N 为输入图像数、C 为输入通道数、F 为卷积核数量、HH、WW 为卷积核高度和宽度，分别讨论其前向、反向传播过程。

forward 函数中将输入图中的对应位置切片，并调整维度为 [N, 1, C, HH, WW]；对应的，调整卷积核维度为 [1, F, C, HH, WW]，两者对位相乘维度为 [N, F, C, HH, WW]。求和，并加入偏置项，得维度 [N, F]，即每张输入图片对应位置得全部输出通道数值。

backward 函数需要计算 x、w 和 b 的梯度。计算 dx 时，将 dout 切片并调整维度为 [N, 1, 1, 1, F]，将 w 的输出维度调至末尾并调整维度为 [1, C, HH, WW, F]，两者对位相乘得维度 [N, C, HH, WW, F]，求和得维度 [N, C, HH, WW]，即指定像素在感受野各像素上的梯度值。计算 dw 时，需要将 tout 转置，并调整维度为 [F, 1, 1, 1, N]；x 的第一维 N 需要被调整到最后，形成维度 [C, HH, WW, N]。两者对位相乘得维度 [F, C, HH, WW, N]，求和得维度 [F, C, HH, WW]，即指定像素在各卷积核上的梯度值。最后，将该像素对应 dout 直接累加至各偏置项梯度。这里有一点需要注意：x 的梯度本身映射至 padding 后的输入 din，因此在输出 dx 时需要将 padding 减裁掉。

3 实验环境

作者使用的 python 版本为 3.9.12。代码主体全部使用 numpy 完成。保存和加载模型时，使用了 pickle 库。实验结果分析时，使用了 pandas、seaborn 和和 matplotlib 绘图，并使用 sklearn 中的 confusion_matrix 生成混淆矩阵。

4 实验结果及分析

4.1 超参数设定

为确定最优超参数，作者进行了如下对比：

超参数	值	超参数	值
初始学习率	10^{-3}	batch size	200
学习率衰减率	0.8	验证集大小	1000
epoch	40	采样方式	顺序采样
优化器	Adam		

表 1: 实验超参数设定

1. 有学习率衰减与无学习率衰减
   若不采用学习率衰减，模型将在第 8 个 epoch 时收敛，此时训练集准确率为 98.55%，验证集准确率为 98.9%。学习率衰减使模型在训练集上的表现提升 1.44%，在验证集上的表现提升 0.3%。

2. Adam 与朴素 SGD 对比
   若采用朴素 SGD，模型经过首个 epoch 后的训练集准确率为 63.97%，测试集准确率为 68.9%；而采用 Adam，模型经过首个 epoch 后的训练集准确率为 97.54%，测试集准确率为 96.5%。由此可见，Adam 能显著加快训练速度。

3. batch size 对比
   由于 LeNet-5 中无 batchnorm 层，且数据量较小，可以采用较大的 batch size。作者分别分别尝试了 50、100、200、500 四个 batch size。当 batch size 为 50 时，模型常常出现梯度不稳定而 loss 不下降的情况；当 batch size 为 500 时，模型训练速度又过慢，难以在短时间内收敛到最优值；而 batch size 在 100 和 200 时，性能比较接近。相比之下，batch size=100 时收敛速度较快，而 batch size=200 时对训练集拟合效果较好。因此，作者最后选择 200 作为 batch size。

4. 训练集采样方式对比
   作者在实验时发现，训练集采样方式会对实验结果产生很大影响：在相同 batch size 下，bootstrap 采样（随机采样）会比顺序采样（每个 epoch 遍历所有训练样本）收敛更快，且最终测试集准确率相近。作者认为，这是由于 bootstrap 采样时采集到的重复值会引导模型更快收敛。但为了充分利用训练集的全部样本，提升最终准确率，作者仍采用顺序采样作为最后的采样方法。

综上，实验采用的超参数如表1 训练完毕后，在测试集上的准确率为 **99.22%**，与 LeCun 报道的 99.27% 准确率持平。

4.2 训练过程分析

训练过程中的损失函数值、训练集准确率及验证集准确率如图4.2、4.3：

训练过程中，损失函数下降平稳，训练集准确率和验证机准确率变化符合预期，未发生明显过拟合。综合以上指标，作者判定模型已经于第 31 个 epoch 收敛。在此之后，损失函数围绕 0.016 波动，而训练集、验证集准确率均不再变化。

4.3 测试集评估结果分析

作者根据测试机预测标签与真实标签绘制了如图4.4中的混淆矩阵：

由图可见，模型较易混淆的数字为 4 和 9（11 例）、3 和 5（6 例）、7 和 1（5 例）。这些都是实际情况中较难辨别的数字组合。由此可见，模型没有显著的性能短板，具有较好的拟合和泛化能力。

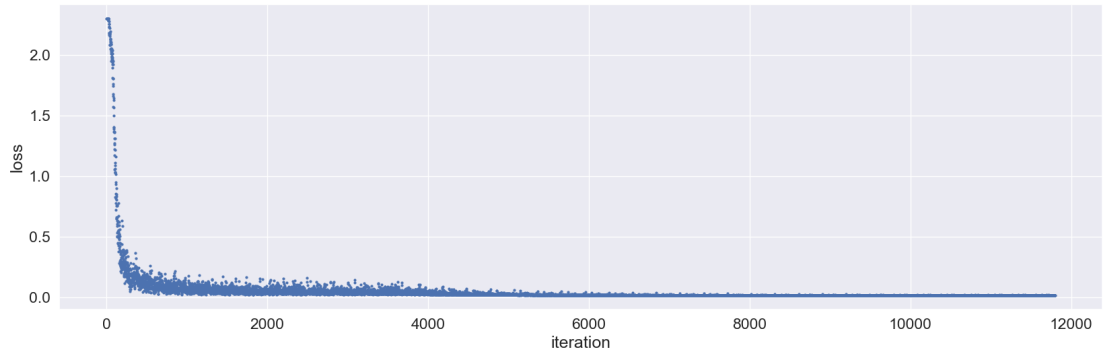

图 4.2: 每次迭代后损失函数值

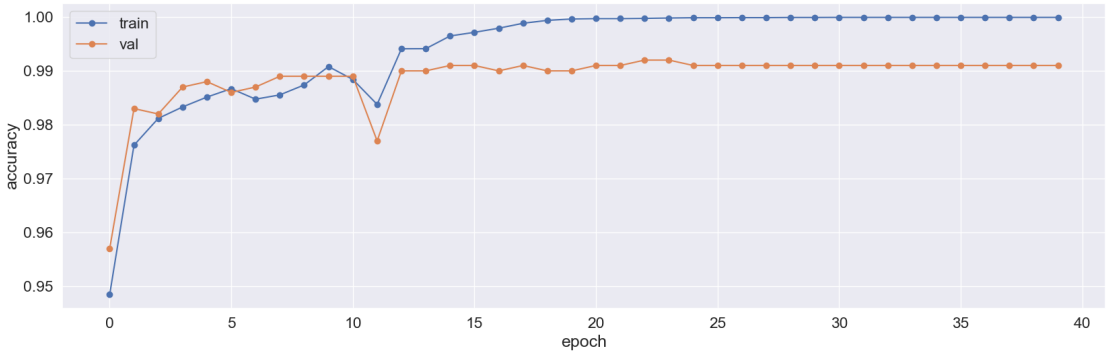

图 4.3: 训练过程中训练集和测试集准确率变化

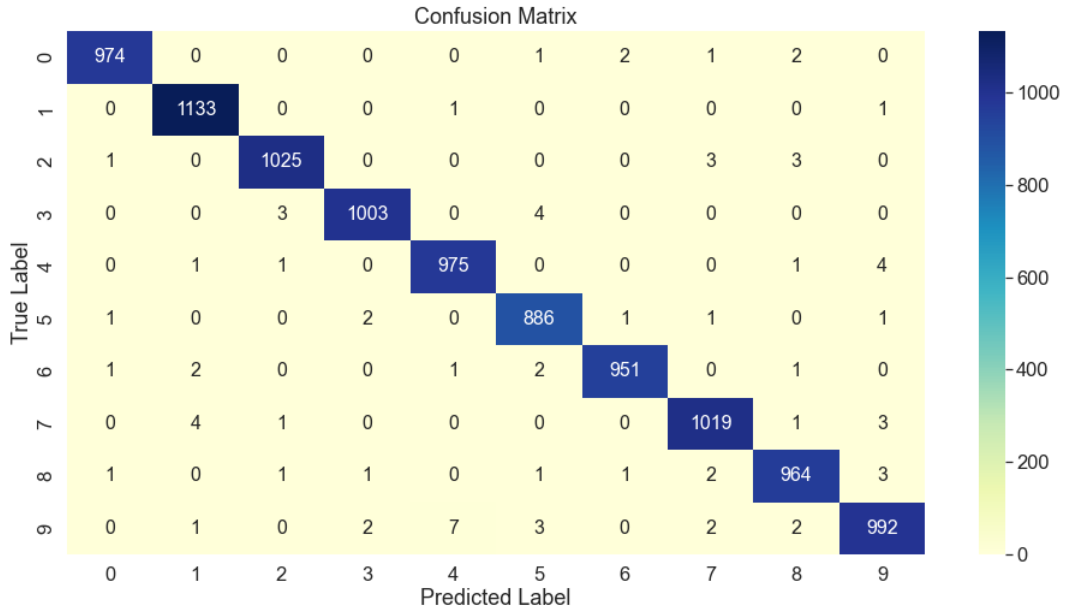

图 4.4: 训练过程中训练集和测试集准确率变化