使用支持向量机和卷积神经网络进行手写数字识别的比较研究

陈昊天

（浙江理工大学计算机科学与技术学院，杭州 310018）

摘　要：本文通过比较支持向量机（SVM）和卷积神经网络（CNN）在手写数字识别任务中的性能，探讨了它们的算法原理、实现流程和机器学习框架选型。在 SVM 算法中，通过在特征空间中找到一个最优的超平面来进行分类，具有较好的泛化能力和鲁棒性，但计算复杂度较高。在 CNN 算法中，通过卷积层、池化层和全连接层构建网络结构，能够自动学习图像特征表示，但训练和推理过程需要较大的计算资源。通过实验比较，发现 CNN 算法在手写数字识别任务中取得了更高的准确率。本文还介绍了 Scikit-learn 和 PyTorch 两个常用的机器学习框架，它们提供了丰富的算法和工具，方便用户进行数据处理和模型选择。

关键词：支持向量机；卷积神经网络；手写数字识别

0 引言

手写数字识别是计算机视觉领域的一个重要任务，对于自动化识别和处理手写文本具有重要意义。在过去的几十年中，研究者们提出了许多不同的方法来解决这个问题，其中支持向量机（SVM）和卷积神经网络（CNN）是两种常用的方法。SVM 通过在特征空间中找到一个最优的超平面来进行分类，而 CNN 通过学习图像的局部特征来实现对不同数字的准确分类。本文旨在比较这两种方法在手写数字识别任务中的性能，并讨论它们的算法原理和实现流程。

1 算法原理介绍

1.1 支持向量机

支持向量机（Support Vector Machine，SVM）是一种常用的机器学习算法，被广泛应用于分类和回归任务。在手写数字识别任务中，SVM 可以用于将输入的图像数据分为不同的数字类别。

SVM 的基本原理是通过在特征空间中找到一个最优的超平面来进行分类。在二分类问题中，SVM 试图找到一个能够最大化样本间间隔的超平面，使得不同类别的数据点能够被尽可能大的间隔分开。为了实现这个目标，SVM 引入了支持向量，这些向量是离超平面最近的训练样本点。通过对支持向量进行优化，SVM 可以构建一个决策边界，将不同类别的数据点正确地分类。

在手写数字识别任务中，我们首先将输入的图像数据转化为特征向量，通常是通过对图像进行预处理和特征提取得到的。然后，我们使用 SVM 算法对这些特征向量进行训练，建立一个分类模型。在训练过程中，SVM 通过优化一个目标函数来寻找最优的超平面参数，使得训练样本点被正确分类并且间隔最大化。一旦模型训练完成，我们可以使用该模型对新的手写数字图像进行分类预测。

在代码示例中，我们使用了 sklearn 库中的 SVM 实现。首先，我们从 MNIST 数据集中获取手写数字图像数据，并进行预处理，将像素值缩放到 0-1 的范围。然后，我们将数据集划分为训练集和测试集，并创建一个 SVM 分类器对象。接下来，我们使用训练集数据对分类器进行训练，得到最优的超平面参数。最后，我们使用测试集数据对模型进行预测，并计算准确率来评估模型的性能。

通过使用支持向量机算法，我们可以实现对手写数字的识别任务。SVM 的优点之一是具有较好的泛化能力，可以处理高维特征空间和小样本数据。SVM 还具有对异常值和噪声数据的较好鲁棒性。然而，SVM 的计算复杂度较高，不利于大规模数据集和高维特征空间的处理。在实际应用中，需要综合考虑算法的性能和计算资源的限制来

选择合适的机器学习方法。

1.2 卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）是一种在计算机视觉领域广泛应用的深度学习算法，被用于图像分类、目标检测和图像生成等任务。在手写数字识别中，CNN 可以通过学习图像的局部特征来实现对不同数字的准确分类。

CNN 的基本原理是通过使用卷积层、池化层和全连接层来构建网络结构。卷积层通过卷积运算提取图像的局部特征，捕捉到不同位置的模式信息。池化层用于降低特征图的尺寸，减少计算量，并保留重要的特征。全连接层将提取的特征映射到不同类别的概率分布，实现分类任务。

在手写数字识别任务中，我们首先将输入的图像数据进行预处理，如归一化和转换为张量格式。然后，我们构建一个包含卷积层、池化层和全连接层的 CNN 模型。卷积层通过应用滤波器来提取图像的局部特征，并使用非线性激活函数进行特征映射。池化层通过对特征图进行下采样来减小特征图的尺寸。全连接层将特征映射到不同类别的概率分布，并使用 softmax 函数进行分类。

在代码示例中，我们使用了 PyTorch 库来构建和训练 CNN 模型。首先，我们定义了一个包含卷积层、池化层和全连接层的网络结构。然后，我们使用训练集数据对模型进行训练，并使用测试集数据评估模型的性能。在训练过程中，我们定义了损失函数（交叉熵损失）和优化器（Adam 优化器），并使用反向传播算法更新模型参数。

通过使用卷积神经网络算法，我们可以实现对手写数字的识别任务。CNN 的优点之一是能够自动学习图像的特征表示，无需手工设计特征。CNN 还具有对平移、缩放和旋转等图像变换具有一定鲁棒性的特点。然而，CNN 的训练和推理过程可能需要较大的计算资源和数据集规模。在实际应用中，需要综合考虑算法的性能和计算资源的限制来选择合适的机器学习方法。

2 机器学习框架选型介绍

2.1 Scikit-learn 机器学习框架

Scikit-learn 是一个 Python 的开源机器学习库，提供了丰富的机器学习算法和工具，用于数据挖掘和数据分析。Sklearn 的设计理念是简单而高效，它建立在 NumPy、SciPy 和 matplotlib 等科学计算库的基础上，为用户提供了一种便捷的方式来实现各种机器学习任务。

Sklearn 提供了包括分类、回归、聚类、降维等多种机器学习算法，涵盖了从传统的统计学习方法到最新的深度学习技术。Sklearn 还提供了一系列的数据预处理、特征工程和模型评估的工具，使得用户能够更加方便地进行数据处理和模型选择。此外，Sklearn 还支持模型的保存和加载，便于在实际应用中进行部署和使用。

Sklearn 的特点之一是其简单易用的 API 接口，使得用户能够快速上手并进行实验和调试。Sklearn 提供了一致的函数命名和参数设置，使得不同的算法可以方便地进行比较和替换。此外，Sklearn 还提供了丰富的文档和示例代码，方便用户学习和参考。

2.2 PyTorch 机器学习框架

PyTorch 是一个开源的深度学习框架，基于 Python 语言。PyTorch 的设计目标是提供一个灵活且高效的深度学习平台，使得研究人员和工程师能够更好地构建和训练深度神经网络模型。

PyTorch 的核心是张量 (Tensor) 操作，它提供了类似于 NumPy 的多维数组操作接口，但在 GPU 上执行时具有高效的计算能力。PyTorch 的一个显著特点是其动态计算图机制，即在计算过程中可以动态地构建和修改计算图，使得用户能够更加灵活地定义复杂的模型结构和训练过程。

PyTorch 提供了丰富的深度学习模块和函数，包括各种常用的神经网络层、损失函数和优化器等。用户可以通过继承 PyTorch 的模型类来自定义自己的网络模型，并使用自动求导机制实现反向传播算法。此外，PyTorch 还支持异步计算和分布式训练，使得用户能够更好地利用多个计算设备

和集群资源。

PyTorch 的易用性和灵活性使得它在学术界和工业界都得到了广泛的应用。它不仅支持深度学习研究的创新工作，还被许多大型公司用于实际的深度学习应用开发。此外，PyTorch 还有一个活跃的社区，提供了丰富的文档、教程和开源项目，方便用户学习和交流。

3 算法实现流程

3.1 支持向量机算法实现

1. 下载和加载 MNIST 数据集，该数据集包含手写数字的图像和对应的标签。通过 fetch_openml 函数获取 MNIST 数据集的图像数据 X 和标签数据 y。将图像像素值缩放到 0-1 的范围。

$$X = \frac{X}{255.0}$$

2. 划分训练集和测试集。使用 train_test_split 函数将数据集划分为训练集和测试集，其中测试集占总数据集的 20%。划分后得到训练集图像数据 X_train、训练集标签数据 y_train、测试集图像数据 X_test 和测试集标签数据 y_test。

3. 创建 SVM 分类器。使用 svm.SVC 函数创建一个支持向量机（SVM）分类器对象 clf。

其中，决策函数可表示为：

$$f(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b)$$

目标函数可表示为：

$$\min_{\alpha,b} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i$$

有以下约束条件：

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \quad i = 1, \ldots, n$$

其中，X 是输入特征矩阵，$\mathrm{x_i}$和x_j是训练样本，y_i和y_j是训练样本的标签，$\mathrm{K(x_i, x_j)}$是核函数，α_i是对应于样本$\mathrm{x_i}$的拉格朗日乘子，b 是偏置项，n 是训练样本数量，C 是正则化参数。

4. 训练模型。使用训练集的图像数据 X_train 和标签数据 y_train，调用 clf 的 fit 方法对模型进行训练，从而学习出一个分类模型。

5. 在测试集上进行预测。使用训练好的模型 clf，调用 predict 方法对测试集的图像数据 X_test 进行预测，得到预测结果 y_pred。

6. 计算准确率。使用 sklearn.metrics 模块中的 accuracy_score 函数，将真实标签数据 y_test 和预测结果 y_pred 作为参数，计算分类器在测试集上的准确率 accuracy。

7. 输出准确率。将准确率 accuracy 乘以 100，并使用 print 函数打印出"Accuracy: xx.xx%"的格式化字符串，其中 xx.xx 为准确率的百分比形式。

3.2 卷积神经网络算法实现

1. 超参数设置。定义批大小（BATCH_SIZE）、设备（DEVICE，可以是 CPU 或 GPU）、迭代轮数（EPOCHS）等超参数。

2. 图像处理。定义图像处理的管道，包括将图像转换为张量（transforms.ToTensor()）和对图像进行归一化处理（transforms.Normalize()）。

3. 数据集准备。下载并加载 MNIST 数据集，包括训练集和测试集。通过 datasets.MNIST 类指定数据集路径、是否下载、图像变换等参数。然后使用 DataLoader 类将数据集封装成可迭代的数据加载器，用于批量加载数据。

3. 网络模型构建。定义一个名为 Digit 的网络模型，该模型包含了卷积层、全连接层和激活函数等组件。通过继承 nn.Module 类并实现 forward 方法来定义网络模型的结构。

Convolution 卷积运算可表示为：

$$\text{Convolution}(\mathbf{X}, \mathbf{W})i, j = \mathbf{X} * \mathbf{W}i, j = \sum_{m=1}^{M} \sum_{n=1}^{N} \mathbf{X}i + m - 1, j + n - 1 \cdot \mathbf{W}m, n$$

其中，X 表示输入数据，W 表示权重参数。

交叉熵损失函数可表示为：

$$\text{CrossEntropyLoss}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\mathbf{Y}i,j \cdot \log(\mathbf{X}i,j)$$

其中，$Y_{i,j}$表示真实标签的第 i 个样本的第 j 个元素，$X_{i,j}$表示模型输出的第 i 个样本的第 j 个元素。

5. 优化器设置。创建一个优化器对象，使用 Adam 优化算法来更新网络模型的参数。

6. 训练方法。定义一个名为 train_model 的训练方法，用于训练网络模型。在每个训练批次中，将数据和目标标签加载到设备上，并将优化器的梯度置零。然后通过前向传播获得模型输出，并计算损失值。接着进行反向传播计算梯度，并通过优化器更新模型参数。在每隔一定批次时，打印当前轮次和损失值。

7. 测试方法。定义一个名为 test_model 的测试方法，用于评估网络模型在测试集上的性能。在测试阶段，将模型设置为评估模式，并在不进行梯度计算的情况下进行前向传播。计算测试集上的损失值和准确率，并打印输出。

8. 训练和测试过程。通过一个循环迭代的方式，依次调用 train_model 和 test_model 方法进行模型的训练和评估。每个周期内训练模型并在测试集上进行评估。

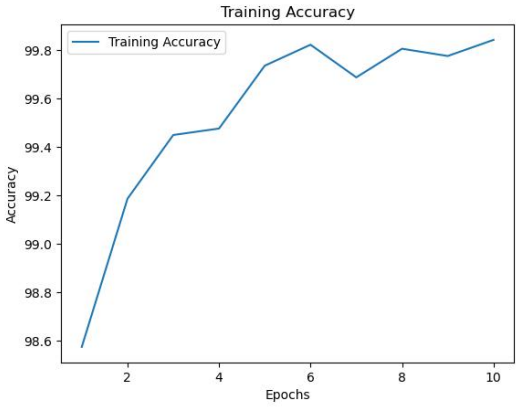
4 算法结果对比

SVM 是一种监督学习算法，用于二分类和多分类问题。在训练阶段，SVM 通过找到一个最优的超平面来分割不同类别的数据点。这个超平面被选择为最大化样本点到超平面的间隔（即最大化间隔），以提高模型的鲁棒性和泛化能力。

SVM 的训练过程是一个一次性的过程，也就是说，一旦找到了最优的超平面，模型的训练就完成了。不需要进行多轮训练来进一步优化模型。

CNN 是一种深度学习算法，主要用于图像识别和计算机视觉任务。在训练阶段，CNN 通过多个卷积层、池化层和全连接层组成的网络结构进行前向传播和反向传播，通过不断调整网络中的权重和参数来最小化损失函数。

CNN 的训练过程是一个迭代的过程，通常需要进行多轮训练（即多个 epochs）来不断优化模型的性能。每轮训练都会在训练数据上进行前向传播、计算损失、反向传播和参数更新，以逐步提升模型的准确性。



CNN 算法准确率-训练轮数统计数据

由于 SVM 和 CNN 的算法原理和训练方式不同，SVM 算法并不需要像 CNN 算法一样进行多轮训练。SVM 通过找到一个最优的超平面（决策边界）来划分数据点，一旦找到这个最优超平面，就可以直接将其应用于新的数据进行分类。相反，CNN 使用深度神经网络进行学习和特征提取，需要通过多轮训练来不断调整网络参数以提高性能。

实验所使用的测试机 CPU 为 16GB RAM 的 Apple M1 Pro，支持向量机 SVM 算法进行了 1 轮训练，占用 100%CPU，耗时 182.25 秒。其准确度为 97.64%。卷积神经网络 CNN 算法进行了 10 轮训练，占用 190%CPU，总耗时 182.44 秒，准确度逐渐逼近 99%。需要注意的是，CNN 算法可以通过自定义超参数与迭代过程，为算法性能调优带来更明显的优势。

5 结语

通过对支持向量机（SVM）和卷积神经网络（CNN）在手写数字识别任务中的比较研究，我们发现 CNN 表现出更高的准确率。SVM 具有较好的泛化能力和鲁棒性，在处理高维特征空间和小样本数据时表现优秀。然而，SVM 的计算复杂度较高，不适用于大规模数据集和高维特征空间的处理。相比之下，CNN 能够自动学习图像的特征表示，无需手工设计特征，具有一定的平移、缩放和旋转等图像变换鲁棒性。然而，CNN 的训练和推理过程可能需要较大的计算资源和数据集规模。在选择机器学习方法时，需要综合考虑算法的性能和计算资源的限制。此外，本文还介绍了 Scikit-learn 和 PyTorch 两个常用的机器学习框架，它们提供了丰富的算法和工具，方便用户进行数据处理和模型选择。

参考文献：

[1] 丁世飞，齐丙娟，谭红艳. 支持向量机理论与算法研究综述[J]. 电子科技大学学报，2011, 40(1): 2-10.

[2] 蒙庚祥，方景龙. 基于支持向量机的手写体数字识别系统设计[J]. 计算机工程与设计，2005, 26(6): 1592-1594.

[3] 周飞燕，金林鹏，董军. 卷积神经网络研究综述[J]. 计算机学报，2017, 40(6): 1229-1251.

[4] 黄一天，陈芝彤. Pytorch 框架下基于卷积神经网络实现手写数字识别[J]. 电子技术与软件工程，2018 (19): 147-147.

[5] 吕红. 基于卷积神经网络的手写数字识别系统的设计[J]. 智能计算机与应用，2019, 9(2): 54-56