神经网络与模糊系统报告：对 MNIST 数据集进行手写数字识别

姓名：高峰 学号：2020110306

一、 数据集简介

图 1 MNIST 数据集部分数据

MNIST 数据集来自美国国家标准与技术研究所。训练集由来自 250 个不同人手写的数字构成是机器学习领域中非常经典的一个数据集，由 60000 个训练样本和 10000 个测试样本组成，每个样本都是一张 28 * 28 像素的灰度手写数字图片。目前常见的数据集有中科院手写汉字数据集 CASIA-OLHWDB 和 CASIA-HWDB、USPS 美国邮政服务手写数字识别库、HCL2000 脱机手写汉字库（BUPT）,常见方法有 kNN、SVM 等传统机器学习方法和 DNN、CNN 等深度学习方法。

二、 基于 kNN（k 近邻）算法

1. 原理

K 近邻（k-Nearest Neighbor）是一种常用的监督学习方法，给定测试样本，基于某种距离度量找出训练集中与其最靠近的 k 个训练样本，然后基于这 k 个“邻居”的信息进行预测。图 2 为 k 近邻分类器示意图，k 取不同值，分类结果会不同，且不同距离计算方式，结果也会不同。给定测试样本$\boldsymbol{x}$，若其最近邻样本为$\boldsymbol{z}$，则分类器出错概率为

$$P(err) = 1 - \sum_{c \in y} P(c|\boldsymbol{x}) p(c|\boldsymbol{z})$$

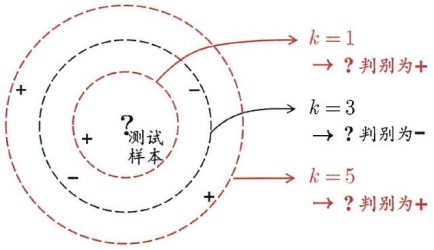


图 2 虚线为等距线；测试样本在 k 为不同值的预测

假设样本 i.i.d，对任意测试样本，总能在任意近的范围内找到上述中的训练样本$\boldsymbol{z}$，令$c^* = \arg \max_{c \in \mathcal{Y}} P(c \mid \boldsymbol{x})$表示贝叶斯最优分类器结果，有（即泛化错误率不超过贝叶斯最优分类器的两倍）

$$P(err) \simeq 1 - \sum_{\alpha \in \mathcal{Y}} P^2(c \mid \boldsymbol{x}) \leqslant 1 - P^2(c^* \mid \boldsymbol{x}) = (1 + P(c^* \mid \boldsymbol{x}))(1 - P(c^* \mid \boldsymbol{x})) \leqslant 2 \times (1 - P(c^* \mid \boldsymbol{x}))$$

2. 实验方法与设置

对训练数据和测试数据进行处理，将数据转化为 numpy 矩阵，并将 28*28 的矩阵转化为一维行向量，然后对数据进行归一化处理。分别实现了 kNN 中的欧拉距离公式和曼哈顿距离公式。利用距离公式得到测试数据到每个训练数据的距离，然后将这些数据根据距离进行升序排序，选择前 k 个数据。利用这 k 个数据所对应的类别，得到测试数据所属的类别，下图 3 为欧氏距离分类部分代码，下列公式分别为两个 n 维向量的曼哈顿距离和欧式距离。

```
# 使用欧拉公式作为距离测量
if dis == 'E':
    for i in range(num_test):
        distances = np.sqrt(np.sum(((self.Xtr - np.tile(X_test[i],
                                                        (self.Xtr.shape[0], 1)))) ** 2, axis=1))
        nearest_k = np.argsort(distances)
        topK = nearest_k[:k]
        class_count = {}
        for i in topK:
            class_count[self.ytr[i]] = class_count.get(self.ytr[i], 0) + 1
        sorted_class_count = sorted(class_count.items(), key=lambda elem: elem[1], reverse=True)
        label_list.append(sorted_class_count[0][0])
    return np.array(label_list)
```

图 3 kNN 核心部分代码

```
train_data: (60000, 784)
train_label: 60000
test_data: (5000, 784)
test_labels: 5000
Got 5000 / 5000 correct when k= 1 => accuracy: 1.000000
Got 4956 / 5000 correct when k= 3 => accuracy: 0.991200
Got 4928 / 5000 correct when k= 5 => accuracy: 0.985600
```

图 4 取 5000 测试样本进行不同 k 值预测

$$d_{12,M} = \sum_{k=1}^{N} |x_{1k} - x_{2k}| \qquad d_{12,E} = \sqrt{\sum_{k=1}^{N} (x_{1k} - x_{2k})^2}$$

3. 实验结果和分析

当度量距离采用欧式距离，且测试样本数量取 5000 时，分别使用 k=1,3,5 进行分类，准确率结果如图 4 所示。当度量距离采用曼哈顿距离，且测试样本数量取 1000 时，准确率结果远不如欧氏距离效果。当 k=1,3,5 时准确率分别为 0.147，0.143，0.154。结果表明，曼哈顿距离并不适合该数据集，但是欧氏距离对该数据集的拟合程度非常好，准确率可以分别达到 1.00，0.9912，0.9856。

编写代码已上传 github 地址：https://github.com/reacher1130/MNIST_DEMO_PYTORCH

三、 基于 CNN（卷积神经网络）

使用 PyTorch 框架，编写了 AlexNet 网络，由于数据集尺寸过小，对 **AlexNet 重新修改网络模型大小**，即 AlexNet_Small，以及对 **ResNet-18 网络重新修改尺寸**，下载好 MNIST 数据集，使用 Dataloader 传入上述三个网络。一般而言，任何一个 CNN 网络都分为特征提取部分和分类部分，前者包含若干 CNN 层及池化层，后者包含若干线性层和其他层，统一使用动量为 0.9，学习率为 0.01 的 SGD 优化算法以及交叉熵损失函数进行分类。

1. 原理

（1） AlexNet

该网络共包含 8 个权重层，其中 5 个卷积层，3 个全连接层，如图 5 所示。1，2 卷积层后连有 LRN 层，每个 LRN 及最后层卷积层后跟有最大池化层，并且每层均为 RELU 激活函数。FC 层后使用了 DropOut 以解决过拟合。

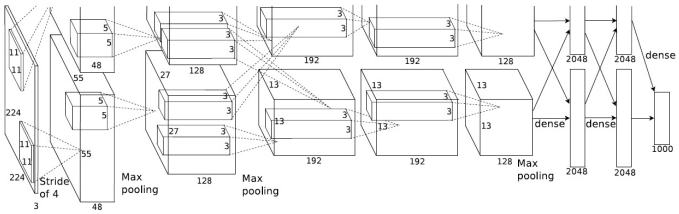


图 5 AlexNet 网络结构

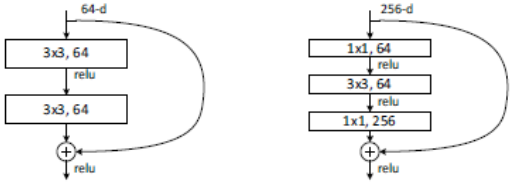


图 6 ResNet 中的残差块单元

（2） ResNet-18

由于网络越来越深，ResNet 应运而生，其将 Conv 前的输入加到输出上，如图 6 所示，称为残差块（BasicBlock）。其解决了 DNN 的梯度弥散和爆炸问题以及 DNN 精度随着模型的加深，会不再上升或大幅度下降的问题。我选择 18 深度的 ResNet，其中 18 指定的是带有权重的 18 层，包括 Conv 层和 FC 层，不包括池化层和 BN 层，同样修改网络大小及参数。

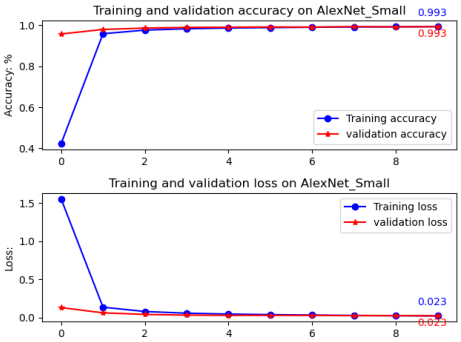
2. 实验方法与设置

对于 AlexNet 分别选择大网络和修改为小网络模型。其中大网络中的特征部分输入为[1,64,5,5]维张量，输出为 9216 维向量传入分类部分逐渐减小特征数量直到 10 维。而小网络中的特征部分输入为[1,32,3,3]维张量，输出为 2048 维向量传入分类部分逐渐减小特征数量直到 10 维。

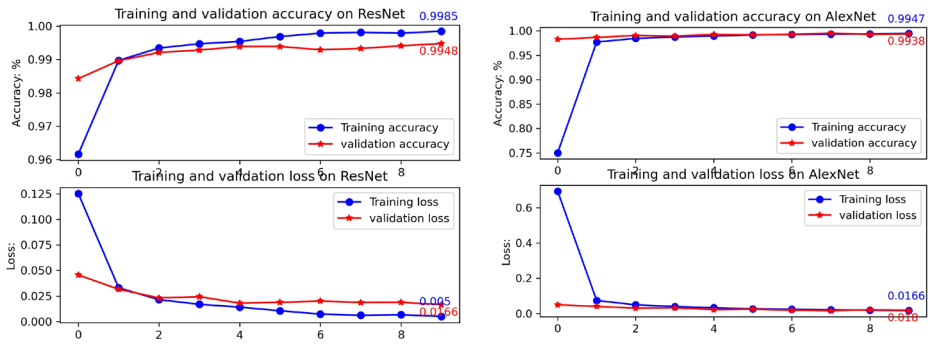
对 ResNet-18 网络，论文结构如图 7 所示。修改后的网络参数部分如图 8 所示，Block 中的大小依次减小 **1/4**。

layer name	output size	18-layer
conv1	112×112	7×7, 64, stride 2
		3×3 max pool, stride 2
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2
average pool, 1000-d fc, softmax		1×1
FLOPs		1.8×10^9

```
ResNet(
  (conv1): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (max_pool): MaxPool2d(kernel_size=3, stride=1, padding=1, dilation=1, ceil_mode=False)
  (block1): Sequential(
    (0): Basicblock(
      (conv1): Sequential(
        (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
      )
      (conv2): Sequential(
        (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (shortcut): Sequential()
    )
```



3. 实验结果与分析



三种网络的训练和测试情况如图所示，共 10 个 epoch，其中将最后 epoch 的结果写在图上，同时，对每一类数字分别进行测试，结果如右格所示。此外，每个网络的参数量大小也给出，可以发现效果最好的是 ResNet 且参数量最小。通过此次实验，我的代码能力和 pytorch 网络搭建能力有所提升。

	AlexNet	AlexNet-S	ResNet-18
0	100%	99%	99%
1	100%	100%	100%
2	99%	97%	99%
3	97%	98%	97%
4	100%	100%	100%
5	99%	98%	99%
6	99%	98%	98%
7	100%	100%	100%
8	99%	98%	99%
9	100%	98%	100%
M	56.83M	3.87M	0.79M

编写代码已上传 github 地址：https://github.com/reacher1130/MNIST_DEMO_PYTORCH