

识别数字

背景介绍

当我们学习编程的时候，编写的第一个程序一般是实现打印"Hello World"。而机器学习（或深度学习）的入门教程，一般都是 MNIST 数据库上的手写识别问题。原因是手写识别属于典型的图像分类问题，比较简单，同时MNIST数据集也很完备。MNIST数据集作为一个简单的计算机视觉数据集，包含一系列如图1所示的手写数字图片和对应的标签。图片是28x28的像素矩阵，标签则对应着0~9的10个数字。每张图片都经过了大小归一化和居中处理。



图1. MNIST图片示例

MNIST数据集是从 NIST 的Special Database 3 (SD-3) 和Special Database 1 (SD-1) 构建而来。由于SD-3是由美国人口普查局的员工进行标注，SD-1是由美国高中生进行标注，因此SD-3比SD-1更干净也更容易识别。Yann LeCun等人从SD-1和SD-3中各取一半作为MNIST的训练集（60000条数据）和测试集（10000条数据），其中训练集来自250位不同的标注员，此外还保证了训练集和测试集的标注员是不完全相同的。

Yann LeCun早先在手写字符识别上做了很多研究，并在研究过程中提出了卷积神经网络（Convolutional Neural Network），大幅度地提高了手写字符的识别能力，也因此成为了深度学习领域的奠基人之一。如今的深度学习领域，卷积神经网络占据了至关重要的地位，从最早Yann LeCun提出的简单LeNet，到如今ImageNet大赛上的优胜模型VGGNet、GoogLeNet、ResNet等（请参见[图像分类](#)教程），人们在图像分类领域，利用卷积神经网络得到了一系列惊人的结果。

有很多算法在MNIST上进行实验。1998年，LeCun分别用单层线性分类器、多层感知器（Multilayer Perceptron, MLP）和多层卷积神经网络LeNet进行实验，使得测试集上的误差不断下降（从12%下降到0.7%）[1]。此后，科学家们又基于K近邻（K-Nearest Neighbors）算法[2]、支持向量机（SVM）[3]、神经网络[4-7]和Boosting方法[8]等做了大量实验，并采用多种预处理方法（如去除歪曲、去噪、模糊等）来提高识别的准确率。

本教程中，我们从简单的模型Softmax回归开始，带大家入门手写字符识别，并逐步进行模型

优化。

模型概览

基于MNIST数据训练一个分类器，在介绍本教程使用的三个基本图像分类网络前，我们先给出一些定义：

- \mathbf{X} 是输入：MNIST图片是 28×28 的二维图像，为了进行计算，我们将其转化为784维向量，即 $\mathbf{X} = (x_0, x_1, \dots, x_{783})$ 。
- \mathbf{Y} 是输出：分类器的输出是10类数字（0-9），即 $\mathbf{Y} = (y_0, y_1, \dots, y_9)$ ，每一维 y_i 代表图片分类为第 i 类数字的概率。
- \mathbf{L} 是图片的真实标签： $\mathbf{L} = (l_0, l_1, \dots, l_9)$ 也是10维，但只有一维为1，其他都为0。

Softmax回归(Softmax Regression)

最简单的Softmax回归模型是先将输入层经过一个全连接层得到的特征，然后直接通过softmax函数进行多分类[9]。

输入层的数据 \mathbf{X} 传到输出层，在激活操作之前，会乘以相应的权重 \mathbf{W} ，并加上偏置变量 \mathbf{b} ，具体如下：

$$y_i = \text{softmax}\left(\sum_j W_{i,j}x_j + b_i\right)$$

其中 $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

对于有 N 个类别的多分类问题，指定 N 个输出节点， N 维输入特征经过softmax将归一化为 N 个[0,1]范围内的实数值，分别表示该样本属于这 N 个类别的概率。此处的 y_i 即对应该图片为数字 i 的预测概率。

在分类问题中，我们一般采用交叉熵代价损失函数（cross entropy），公式如下：

$$\text{crossentropy}(\text{label}, \mathbf{y}) = - \sum_i \text{label}_i \log(y_i)$$

图2为softmax回归的网络图，图中权重用黑线表示、偏置用红线表示、+1代表偏置参数的系

数为1。

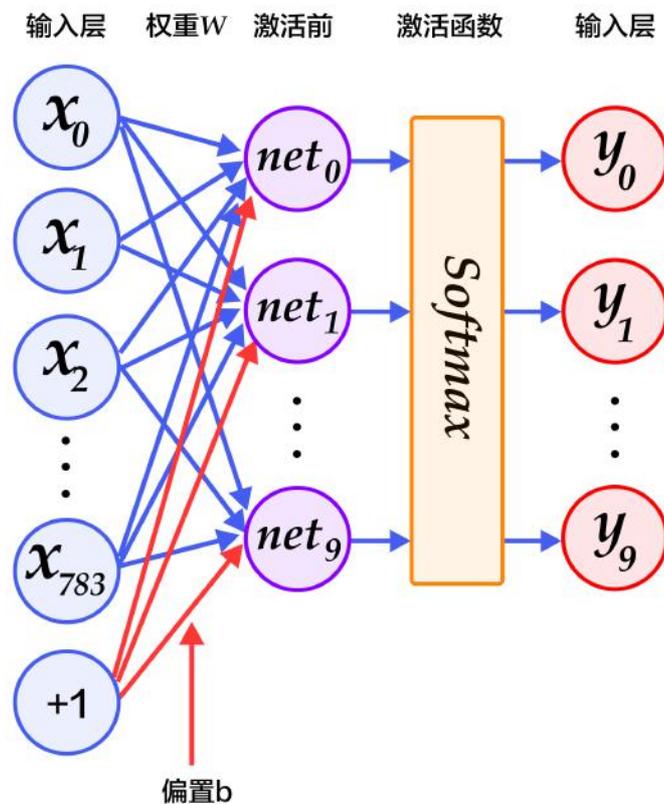


图2. softmax回归网络结构图

多层感知器(Multilayer Perceptron, MLP)

Softmax回归模型采用了最简单的两层神经网络，即只有输入层和输出层，因此其拟合能力有限。为了达到更好的识别效果，我们考虑在输入层和输出层中间加上若干个隐藏层[10]。

1. 经过第一个隐藏层，可以得到 $H_1 = \phi(W_1 X + b_1)$ ，其中 ϕ 代表激活函数，常见的有sigmoid、tanh或ReLU等函数。
2. 经过第二个隐藏层，可以得到 $H_2 = \phi(W_2 H_1 + b_2)$ 。
3. 最后，再经过输出层，得到的 $Y = softmax(W_3 H_2 + b_3)$ ，即为最后的分类结果向量。

图3为多层感知器的网络结构图，图中权重用黑线表示、偏置用红线表示、+1代表偏置参数的系数为1。

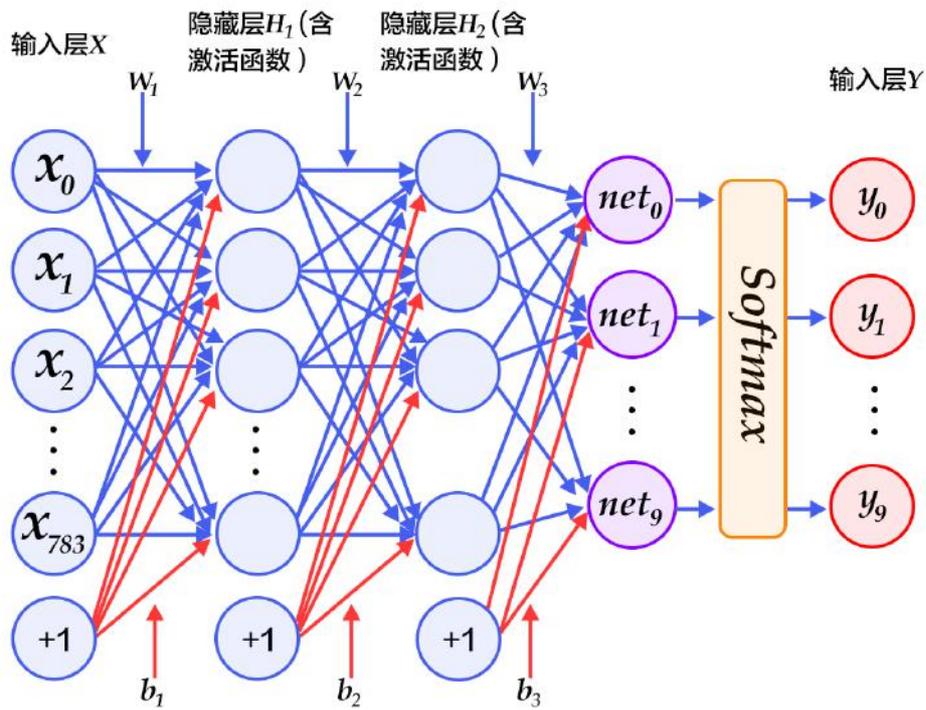


图3. 多层感知器网络结构图

卷积神经网络(Convolutional Neural Network, CNN)

卷积层

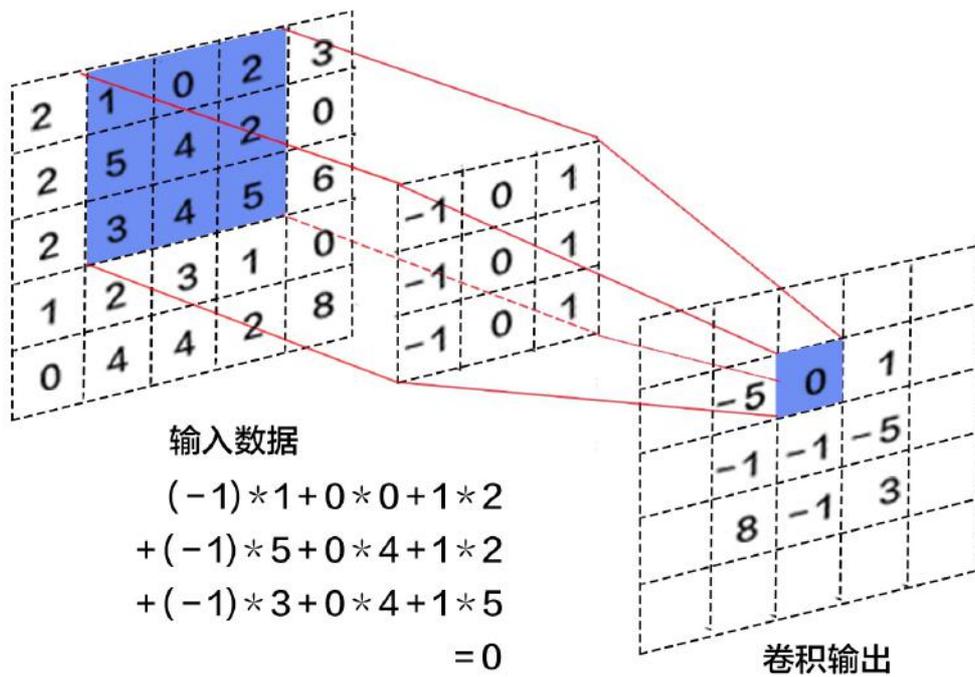


图4. 卷积层图片

卷积层是卷积神经网络的核心基石。该层的参数由一组可学习的过滤器（也叫作卷积核）组成。在前向过程中，每个卷积核在输入层进行横向和纵向的扫描，与输入层对应扫描位置进行卷积，得到的结果加上偏置并用相应的激活函数进行激活，结果能够得到一个二维的激活图(activation map)。每个特定的卷积核都能得到特定的激活图(activation map)，如有的卷积核可能对识别边角，有的可能识别圆圈，那这些卷积核可能对于对应的特征响应要强。

图4是卷积层的一个动态图。由于3D量难以表示，所有的3D量（输入的3D量（蓝色），权重3D量（红色），输出3D量（绿色））通过将深度在行上堆叠来表示。如图4，输入层是 $W_1 = 5, H_1 = 5, D_1 = 3$ ，我们常见的彩色图片其实就是类似这样的输入层，彩色图片的宽和高对应这里的 W_1 和 H_1 ，而彩色图片有RGB三个颜色通道，对应这里的 D_1 ；卷积层的参数为 $K = 2, F = 3, S = 2, P = 1$ ，这里的 K 是卷积核的数量，如图4中有 $FilterW_0$ 和 $FilterW_1$ 两个卷积核， F 对应卷积核的大小，图中 W_0 和 W_1 在每一层深度上都是 3×3 的矩阵， S 对应卷积核扫描的步长，从动态图中可以看到，方框每次左移或下移2个单位， P 对应Padding扩展，是对输入层的扩展，图中输入层，原始数据为蓝色部分，可以看到灰色部分是进行了大小为1的扩展，用0来进行扩展；图4的动态可视化对输出层结果（绿色）进行迭代，显示每个输出元素是通过将突出显示的输入（蓝色）与滤波器（红色）进行元素相乘，将其相加，然后通过偏置抵消结果来计算的。

池化层

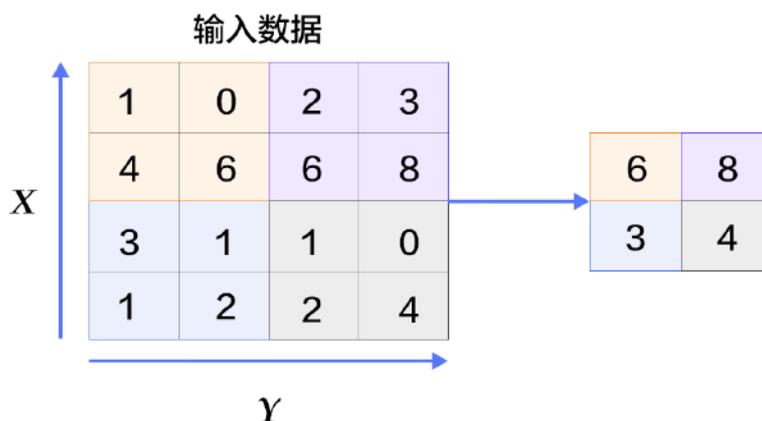


图5. 池化层图片

池化是非线性下采样的一种形式，主要作用是通过减少网络的参数来减小计算量，并且能够在一定程度上控制过拟合。通常在卷积层的后面会加上一个池化层。池化包括最大池化、平均池化等。其中最大池化是用不重叠的矩形框将输入层分成不同的区域，对于每个矩形框的数取最大值作为输出层，如图5所示。

LeNet-5网络

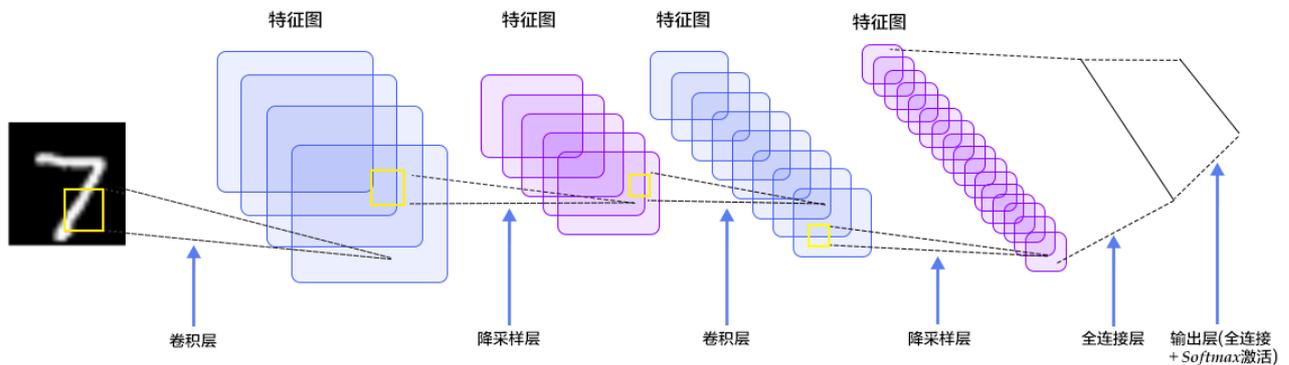


图6. LeNet-5卷积神经网络结构

LeNet-5是一个最简单的卷积神经网络。图6显示了其结构：输入的二维图像，先经过两次卷积层到池化层，再经过全连接层，最后使用softmax分类作为输出层。卷积的如下三个特性，决定了LeNet-5能比同样使用全连接层的多层感知器更好地识别图像：

- 神经元的三维特性：卷积层的神经元在宽度、高度和深度上进行了组织排列。每一层的神经元仅仅与前一层的一块小区域连接，这块小区域被称为感受野(receptive field)。
- 局部连接：CNN通过在相邻层的神经元之间实施局部连接模式来利用空间局部相关性。这样的结构保证了学习后的过滤器能够对于局部的输入特征有最强的响应。堆叠许多这样的层导致非线性“过滤器”变得越来越“全局”。这允许网络首先创建输入的小部分的良好表示，然后从它们组合较大区域的表示。
- 共享权重：在CNN中，每个滤波器在整个视野中重复扫描。这些复制单元共享相同的参数化（权重向量和偏差）并形成特征图。这意味着给定卷积层中的所有神经元检测完全相同的特征。以这种方式的复制单元允许不管它们在视野中的位置都能检测到特征，从而构成平移不变性的性质。

更详细的关于卷积神经网络的具体知识可以参考[斯坦福大学公开课](#)和[图像分类教程](#)。

常见激活函数介绍

- sigmoid激活函数： $f(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}}$
- tanh激活函数： $f(x) = \text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

实际上，tanh函数只是规模变化的sigmoid函数，将sigmoid函数值放大2倍之后再向下平移1个单位： $\text{tanh}(x) = 2\text{sigmoid}(2x) - 1$ 。

- ReLU激活函数： $f(x) = \max(0, x)$

更详细的介绍请参考[维基百科激活函数](#)。

数据准备

数据介绍与下载

执行以下命令，下载MNIST数据库并解压缩，然后将训练集和测试集的地址分别写入train.list和test.list两个文件，供PaddlePaddle读取。

```
1. ./data/get_mnist_data.sh
```

```
1. ./load_data.py
```

```
1. # Define a py data provider
2. @provider(
3.     input_types={'pixel': dense_vector(28 * 28),
4.                 'label': integer_value(10)})
5. def process(settings, filename): # settings is not used currently.
6.     # 打开图片文件
7.     with open( filename + "-images-idx3-ubyte", "rb") as f:
8.         # 读取开头的四个参数，magic代表数据的格式，n代表数据的总量，rows和cols分
           别代表行数和列数
9.         magic, n, rows, cols = struct.unpack(">IIII", f.read(16))
10.        # 以无符号字节为单位一个一个的读取数据
11.        images = np.fromfile(
12.            f, 'ubyte',
13.            count=n * rows * cols).reshape(n, rows, cols).astype('float
           32')
14.        # 将0~255的数据归一化到[-1,1]的区间
15.        images = images / 255.0 * 2.0 - 1.0
16.
17.
18.        # 打开标签文件
19.        with open( filename + "-labels-idx1-ubyte", "rb") as l:
20.            # 读取开头的两个参数
21.            magic, n = struct.unpack(">II", l.read(8))
22.            # 以无符号字节为单位一个一个的读取数据
```

```

23.         labels = np.fromfile(l, 'ubyte', count=n).astype("int")
24.
25.     for i in xrange(n):
26.         yield {"pixel": images[i, :], 'label': labels[i]}

```

```

1.     if not is_predict:
2.         data_dir = './data/'
3.         define_py_data_sources2(
4.             train_list=data_dir + 'train.list',
5.             test_list=data_dir + 'test.list',
6.             module='mnist_provider',
7.             obj='process')

```

```

1.     settings(
2.         batch_size=128,
3.         learning_rate=0.1 / 128.0,
4.         learning_method=MomentumOptimizer(0.9),
5.         regularization=L2Regularization(0.0005 * 128))

```

```

1.     data_size = 1 * 28 * 28
2.     label_size = 10
3.     img = data_layer(name='pixel', size=data_size)
4.
5.     predict = softmax_regression(img) # Softmax回归
6.     #predict = multilayer_perceptron(img) #多层感知器
7.     #predict = convolutional_neural_network(img) #LeNet5卷积神经网络
8.
9.     if not is_predict:
10.         lbl = data_layer(name="label", size=label_size)
11.         inputs(img, lbl)
12.         outputs(classification_cost(input=predict, label=lbl))
13.     else:
14.         outputs(predict)

```

```

1.     def softmax_regression(img):
2.         predict = fc_layer(input=img, size=10, act=SoftmaxActivation())
3.         return predict

```

```

1.     def multilayer_perceptron(img):
2.         # 第一个全连接层, 激活函数为ReLU
3.         hidden1 = fc_layer(input=img, size=128, act=ReluActivation())

```

```

4.     # 第二个全连接层, 激活函数为ReLU
5.     hidden2 = fc_layer(input=hidden1, size=64, act=ReluActivation())
6.     # 以softmax为激活函数的全连接输出层, 输出层的大小必须为数字的个数10
7.     predict = fc_layer(input=hidden2, size=10,
act=SoftmaxActivation())
8.     return predict

```

```

1. def convolutional_neural_network(img):
2.     # 第一个卷积-池化层
3.     conv_pool_1 = simple_img_conv_pool(
4.         input=img,
5.         filter_size=5,
6.         num_filters=20,
7.         num_channel=1,
8.         pool_size=2,
9.         pool_stride=2,
10.        act=TanhActivation())
11.    # 第二个卷积-池化层
12.    conv_pool_2 = simple_img_conv_pool(
13.        input=conv_pool_1,
14.        filter_size=5,
15.        num_filters=50,
16.        num_channel=20,
17.        pool_size=2,
18.        pool_stride=2,
19.        act=TanhActivation())
20.    # 全连接层
21.    fc1 = fc_layer(input=conv_pool_2, size=128, act=TanhActivation())
22.    # 以softmax为激活函数的全连接输出层, 输出层的大小必须为数字的个数10
23.    predict = fc_layer(input=fc1, size=10, act=SoftmaxActivation())
24.    return predict

```

```

1. config=mnist_model.py # 在mnist_model.py中可以选择网络
2. output=./softmax_mnist_model
3. log=softmax_train.log
4.
5. paddle train \
6. --config=$config \ # 网络配置脚本
7. --dot_period=10 \ # 每训练 `dot_period` 个批次后打印
  一个 `.`
8. --log_period=100 \ # 每隔多少batch打印一次日志
9. --test_all_data_in_one_period=1 \ # 每次测试是否用所有的数据
10. --use_gpu=0 \ # 是否使用GPU

```

```
11. --trainer_count=1 \           # 使用CPU或GPU的个数
12. --num_passes=100 \          # 训练进行的轮数（每次训练使用完所有数据为1轮）
13. --save_dir=$output \       # 模型存储的位置
14. 2>&1 | tee $log
15.
16. python -m paddle.utils.plotcurve -i $log > plot.png
```

```
1. I0117 12:52:29.628617 4538 TrainerInternal.cpp:165] Batch=100 samples
   =12800 AvgCost=2.63996 CurrentCost=2.63996 Eval:
   classification_error_evaluator=0.241172 CurrentEval:
   classification_error_evaluator=0.241172
2. ....
3. I0117 12:52:29.768741 4538 TrainerInternal.cpp:165] Batch=200 samples
   =25600 AvgCost=1.74027 CurrentCost=0.840582 Eval:
   classification_error_evaluator=0.185234 CurrentEval:
   classification_error_evaluator=0.129297
4. ....
5. I0117 12:52:29.916970 4538 TrainerInternal.cpp:165] Batch=300 samples
   =38400 AvgCost=1.42119 CurrentCost=0.783026 Eval:
   classification_error_evaluator=0.167786 CurrentEval:
   classification_error_evaluator=0.132891
6. ....
7. I0117 12:52:30.061213 4538 TrainerInternal.cpp:165] Batch=400 samples
   =51200 AvgCost=1.23965 CurrentCost=0.695054 Eval:
   classification_error_evaluator=0.160039 CurrentEval:
   classification_error_evaluator=0.136797
8. ....I0117 12:52:30.223270 4538 TrainerInternal.cpp:181] Pass=0 Batc
   h=469 samples=60000 AvgCost=1.1628 Eval:
   classification_error_evaluator=0.156233
9. I0117 12:52:30.366894 4538 Tester.cpp:109] Test samples=10000 cost=0.
   50777 Eval: classification_error_evaluator=0.0978
```

```
1. python plot_cost.py softmax_train.log
```

```
1. python evaluate.py softmax_train.log
```

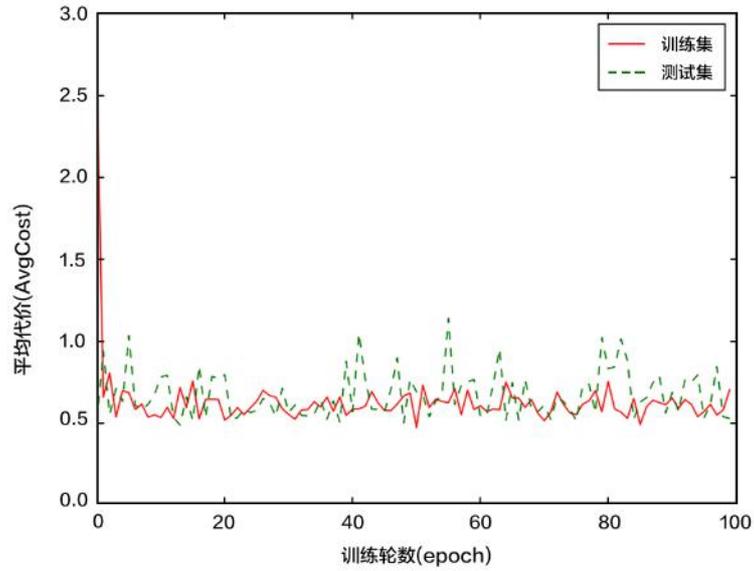


图7. softmax回归的误差曲线图

评估模型结果如下：

1. Best pass is 00013, testing Avgcost is 0.484447
2. The classification accuracy is 90.01%

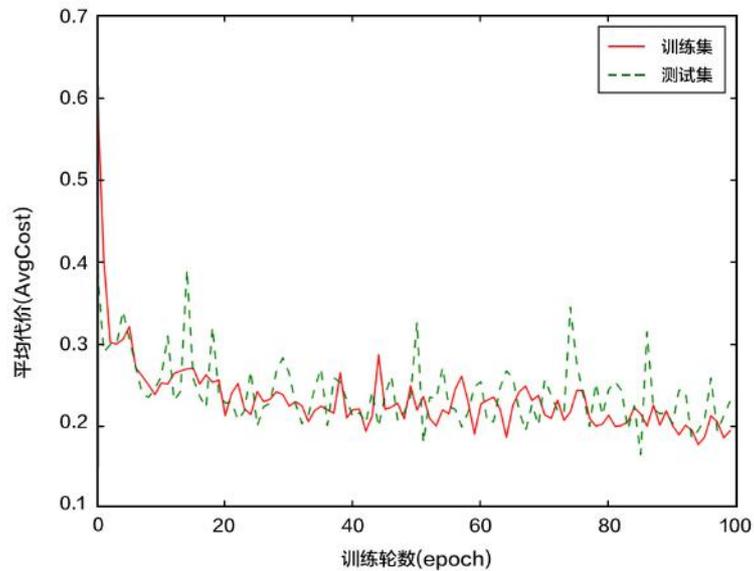


图8. 多层感知器的误差曲线图

评估模型结果如下：

1. Best pass is 00085, testing Avgcost is 0.164746

2. The classification accuracy is 94.95%

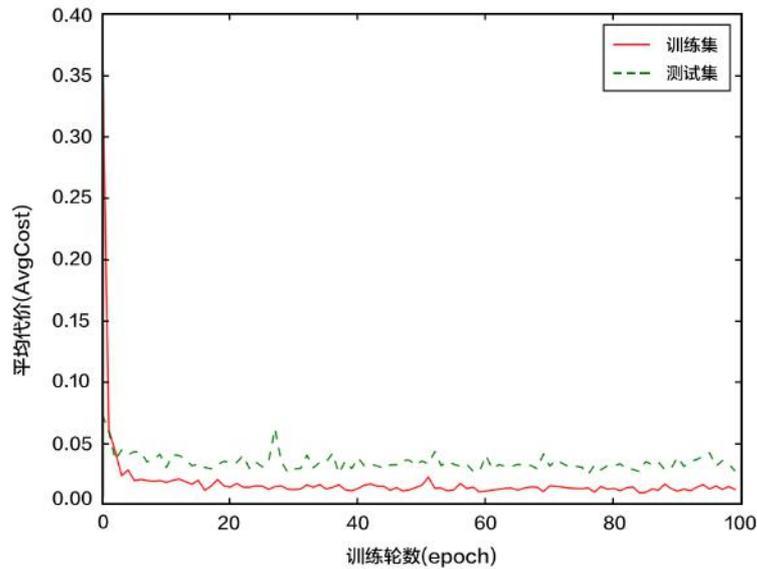


图9. 卷积神经网络的误差曲线图

评估模型结果如下：

1. Best pass is 00076, testing Avgcost is 0.0244684
2. The classification accuracy is 99.20%

从评估结果可以看到，卷积神经网络的最好分类准确率达到惊人的99.20%。说明对于图像问题而言，卷积神经网络能够比一般的全连接网络达到更好的识别效果，而这与卷积层具有局部连接和共享权重的特性是分不开的。同时，从图9中可以看到，卷积神经网络在很早的时候就能达到很好的效果，说明其收敛速度非常快。

应用模型

预测命令与结果

脚本 `predict.py` 可以对训练好的模型进行预测，例如softmax回归中：

1.

```
python predict.py -c softmax_mnist.py -d data/raw_data/ -m softmax_mnist_model/pass-00047
```

- -c 指定模型的结构
- -d 指定需要预测的数据源，这里用测试数据集进行预测
- -m 指定模型的参数，这里用之前训练效果最好的模型进行预测

根据提示，输入需要预测的图片序号，分类器能够给出各个数字的生成概率、预测的结果（取最大生成概率对应的数字）和实际的标签。

```
1. Input image_id [0~9999]: 3
2. Predicted probability of each digit:
3. [[ 1.00000000e+00  1.60381094e-28  1.60381094e-28  1.60381094e-28
4.      1.60381094e-28  1.60381094e-28  1.60381094e-28  1.60381094e-28
5.      1.60381094e-28  1.60381094e-28]]
6. Predict Number: 0
7. Actual Number: 0
```

从结果看出，该分类器接近100%地认为第3张图片上面的数字为0，而实际标签给出的类也确实如此。

总结

本教程的softmax回归、多层感知器和卷积神经网络是最基础的深度学习模型，后续章节中复杂的神经网络都是从它们衍生出来的，因此这几个模型对之后的学习大有裨益。同时，我们也观察到从最简单的softmax回归变换到稍复杂的卷积神经网络的时候，MNIST数据集上的识别准确率有了大幅度的提升，原因是卷积层具有局部连接和共享权重的特性。在之后学习新模型的时候，希望大家也要深入到新模型相比原模型带来效果提升的关键之处。此外，本教程还介绍了PaddlePaddle模型搭建的基本流程，从datapvider的编写、网络层的构建，到最后的训练和预测。对这个流程熟悉以后，大家就可以用自己的数据，定义自己的网络模型，并完成自己的训练和预测任务了。

参考文献

1. LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "[Gradient-based learning applied to document recognition.](#)" Proceedings of the IEEE 86, no. 11 (1998): 2278-2324.
2. Wejéus, Samuel. "[A Neural Network Approach to Arbitrary Symbol Recognition on](#)

- [Modern Smartphones.](#)" (2014).
3. Decoste, Dennis, and Bernhard Schölkopf. "[Training invariant support vector machines.](#)" *Machine learning* 46, no. 1-3 (2002): 161-190.
 4. Simard, Patrice Y., David Steinkraus, and John C. Platt. "[Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis.](#)" In *ICDAR*, vol. 3, pp. 958-962. 2003.
 5. Salakhutdinov, Ruslan, and Geoffrey E. Hinton. "[Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure.](#)" In *AISTATS*, vol. 11. 2007.
 6. Cireşan, Dan Claudiu, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. "[Deep, big, simple neural nets for handwritten digit recognition.](#)" *Neural computation* 22, no. 12 (2010): 3207-3220.
 7. Deng, Li, Michael L. Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoffrey E. Hinton. "[Binary coding of speech spectrograms using a deep auto-encoder.](#)" In *Interspeech*, pp. 1692-1695. 2010.
 8. Kégl, Balázs, and Róbert Busa-Fekete. "[Boosting products of base classifiers.](#)" In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 497-504. ACM, 2009.
 9. Rosenblatt, Frank. "[The perceptron: A probabilistic model for information storage and organization in the brain.](#)" *Psychological review* 65, no. 6 (1958): 386.
 10. Bishop, Christopher M. "[Pattern recognition.](#)" *Machine Learning* 128 (2006): 1-58.



本教程由PaddlePaddle创作，采用[知识共享 署名-非商业性使用-相同方式共享 4.0 国际 许可协议](#)进行许可。