使用神经网络识别手写数字

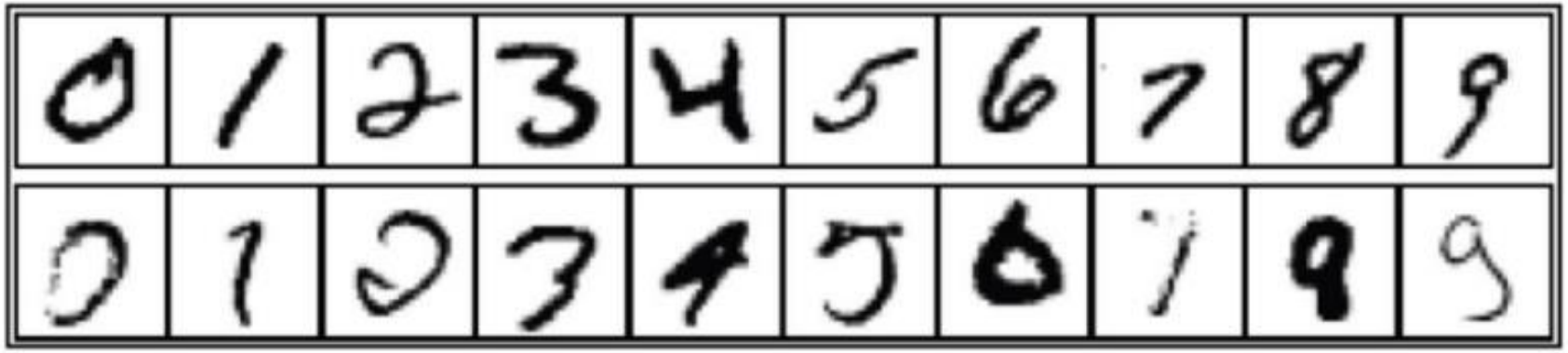
授课教师：庞善民

助教：张浩、刘卓

2023 年 4 月 2 日

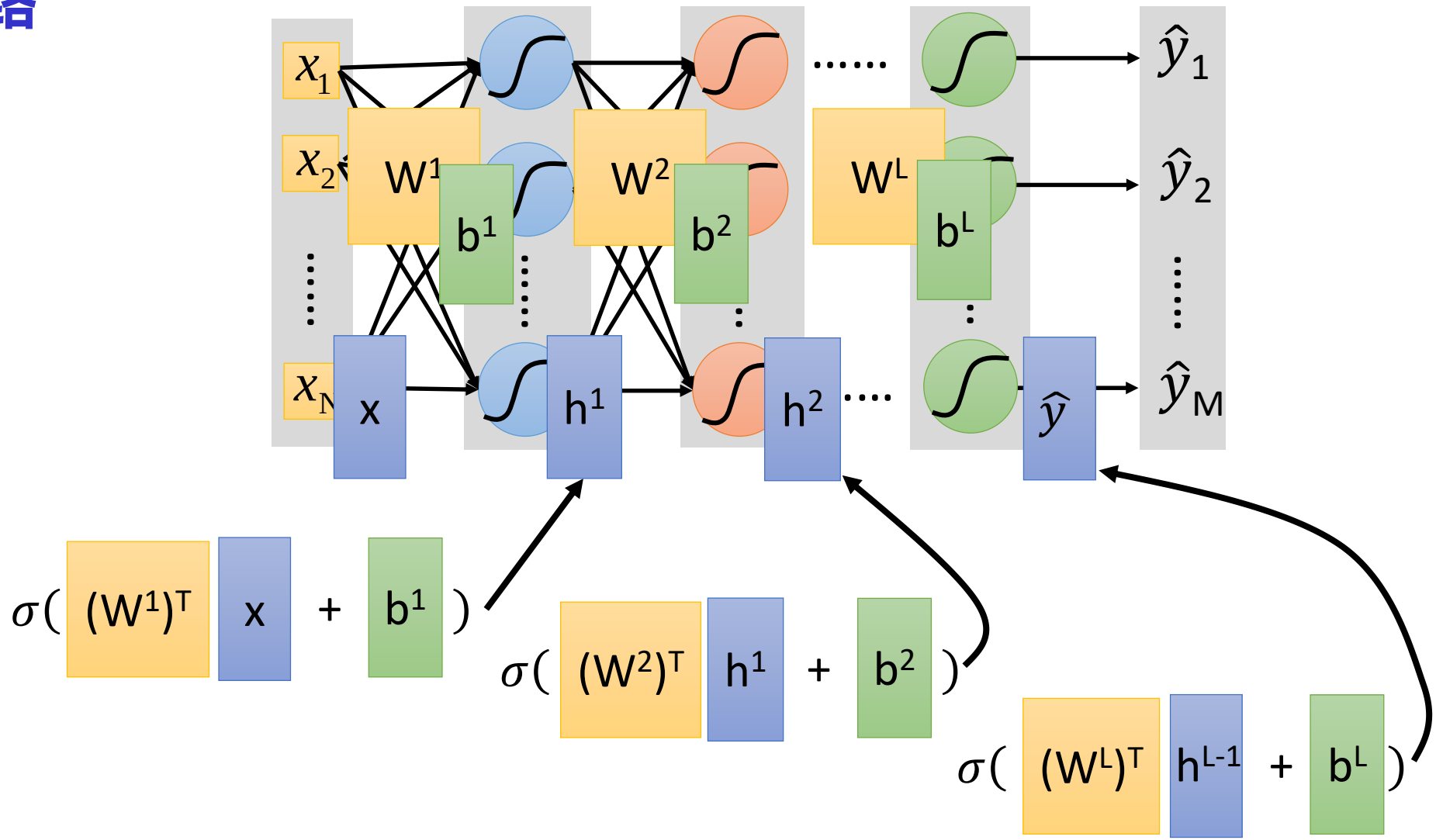
问题描述

试根据神经网络的授课内容，结合**多分类问题**的四个反向传播方程，实现一个手写数字识别程序，使其可以对MNIST数据集图片中的手写数字进行识别。其中，神经网络包含两个隐层，第一个隐层的神经元个数为192，第二个隐层的神经元个数为30。

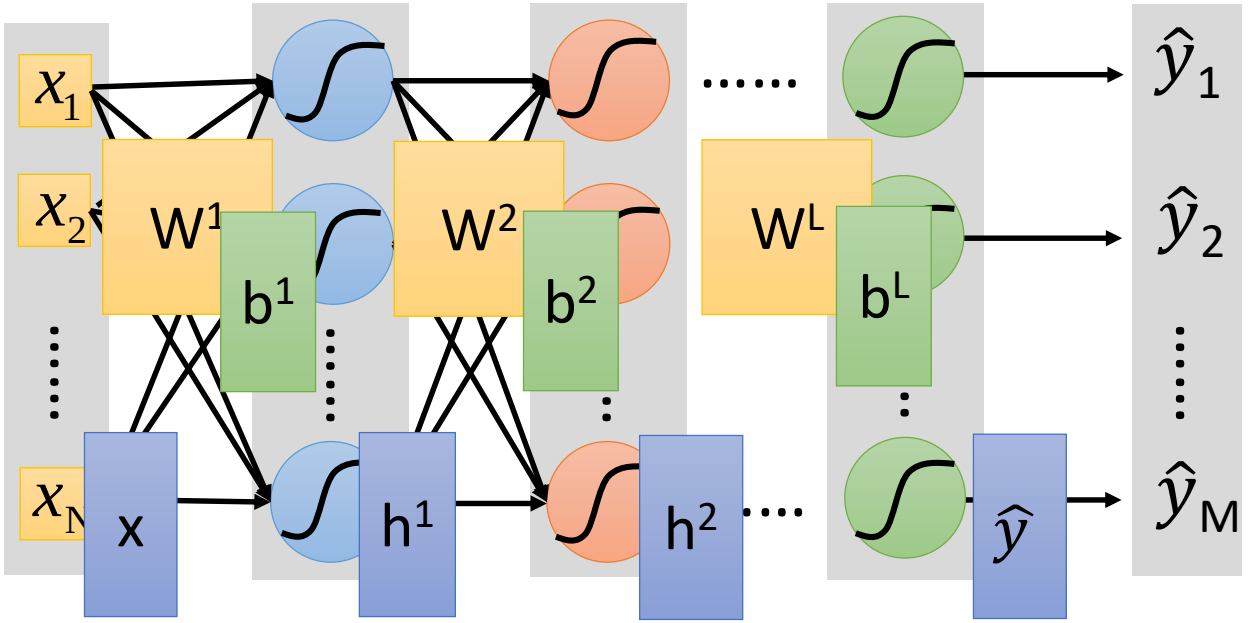


注：损失函数采用交叉熵损失函数，激活函数使用`Sigmoid`激活函数。

神经网络


x_1
x_2
x_N
x
W¹
b¹
h¹
W²
b²
h²
Wᴸ
bᴸ
ŷ
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_M$
σ((W¹)ᵀ x + b¹)
σ((W²)ᵀ h¹ + b²)
σ((Wᴸ)ᵀ hᴸ⁻¹ + bᴸ)

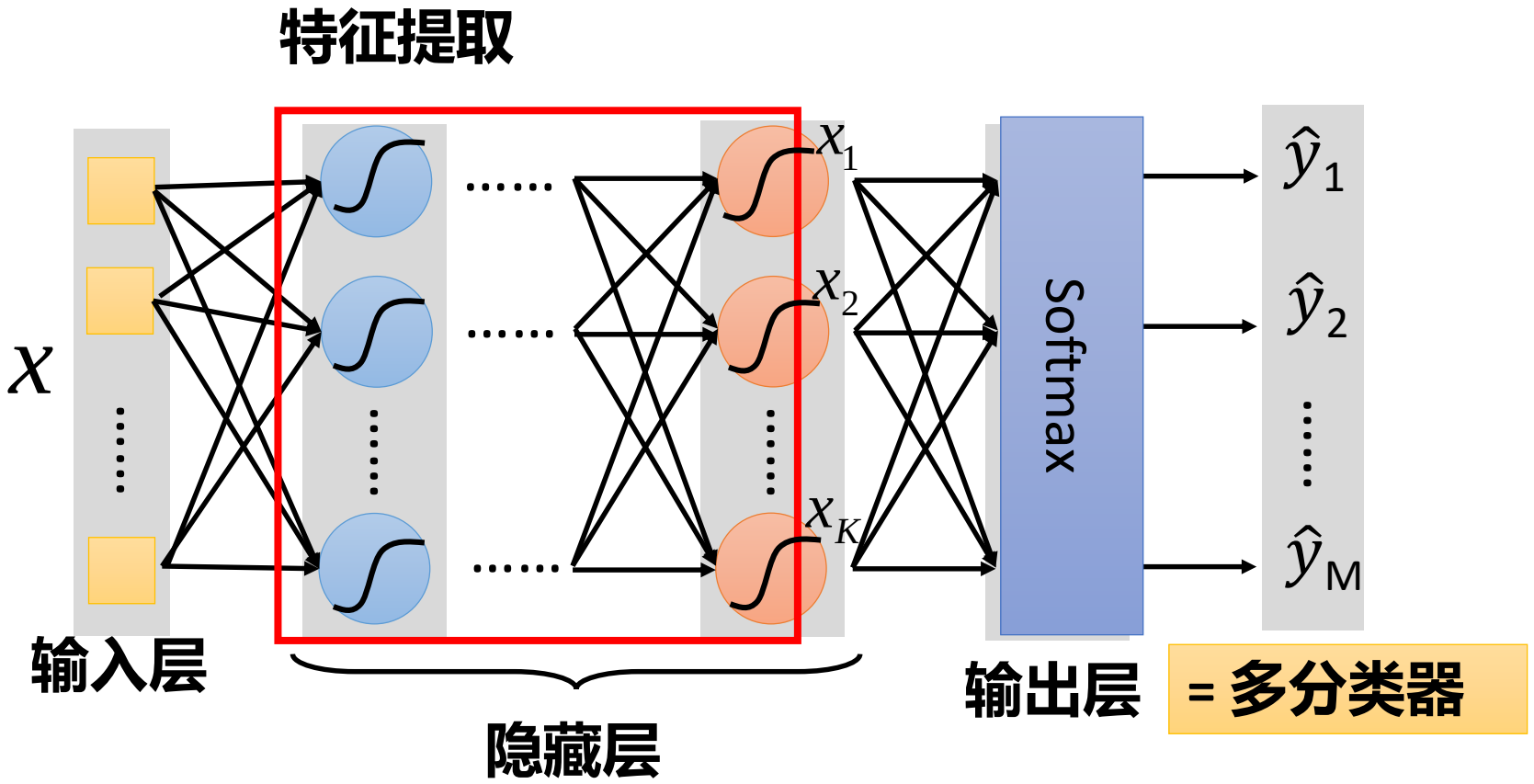
神经网络



$$\hat{y} = f(x)$$ **使用第三方库Numpy来加速矩阵运算**

$$= \sigma((W^L)^T \cdots \sigma((W^2)^T \sigma((W^1)^T x + b^1) + b^2) \cdots + b^L)$$

神经网络



MNIST 数据集简介

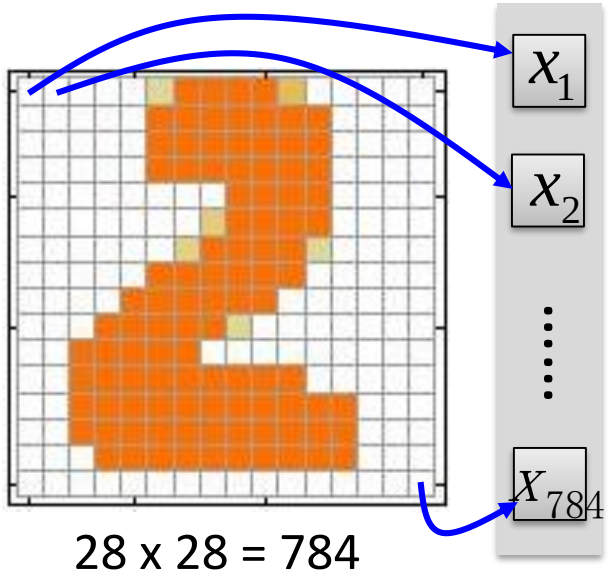
MNIST 的名字来源于 NIST(美国国家标准与技术研究所) 收集的两个数据集改进后的子集。

数据分为两个部分，第一部分包含 60000 幅用于**训练数据**的图像，第二部分是 10000 幅用于**测试数据**的图像。这些图像是 28×28 大小的灰度图像，值为 0.0 表示白色，值为 1.0 表示黑色，中间数值表示逐渐暗淡的灰色。

注：本实验提供的MNIST数据集将第一部分的60000幅图像进一步分为50000幅训练数据和10000幅验证数据。

实现 MNIST 手写数字识别

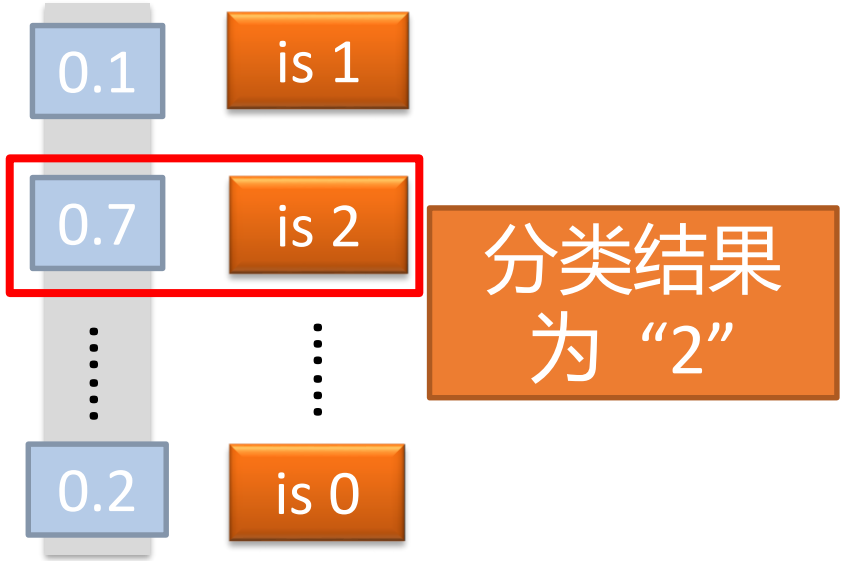
- Input



28 x 28 = 784

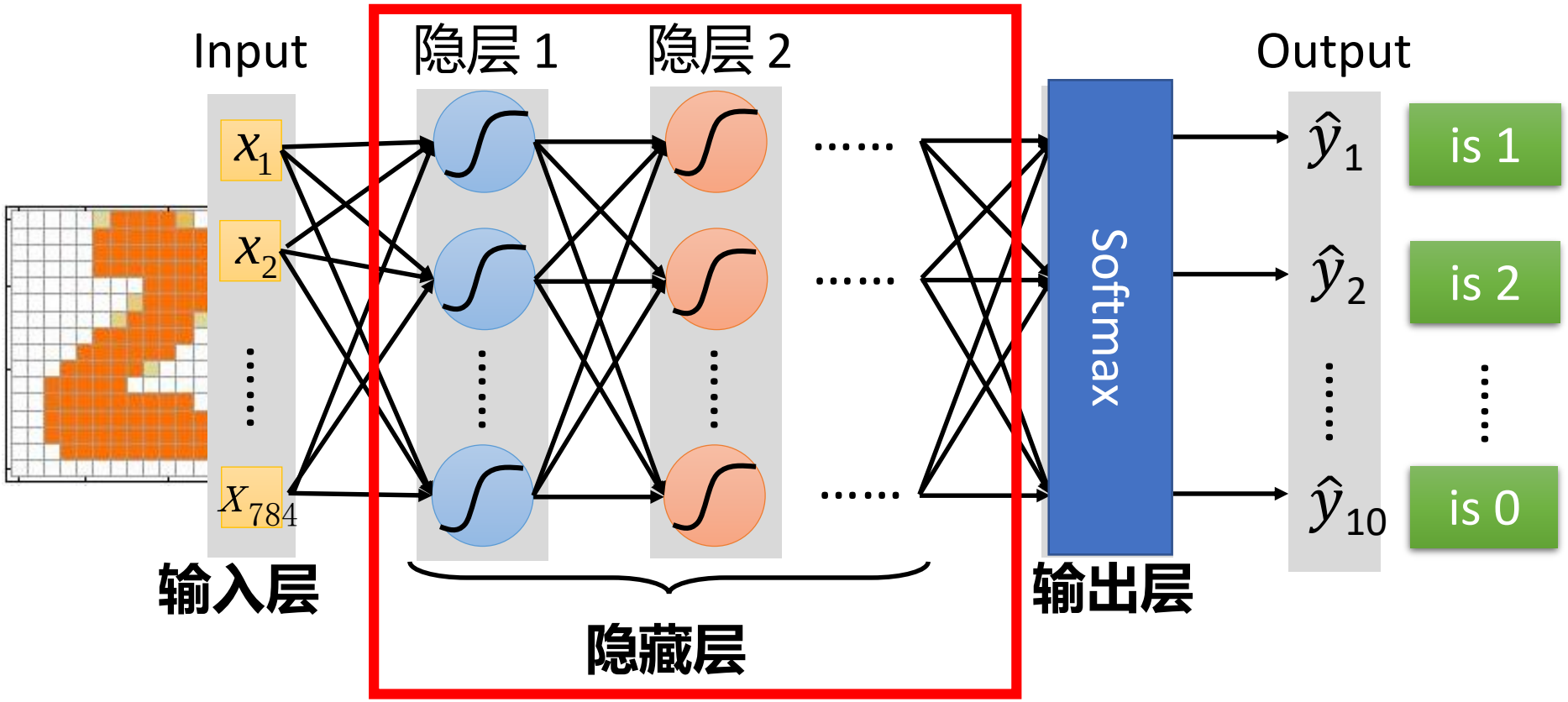
黑色 → 1.0
白色 → 0.0

- Output



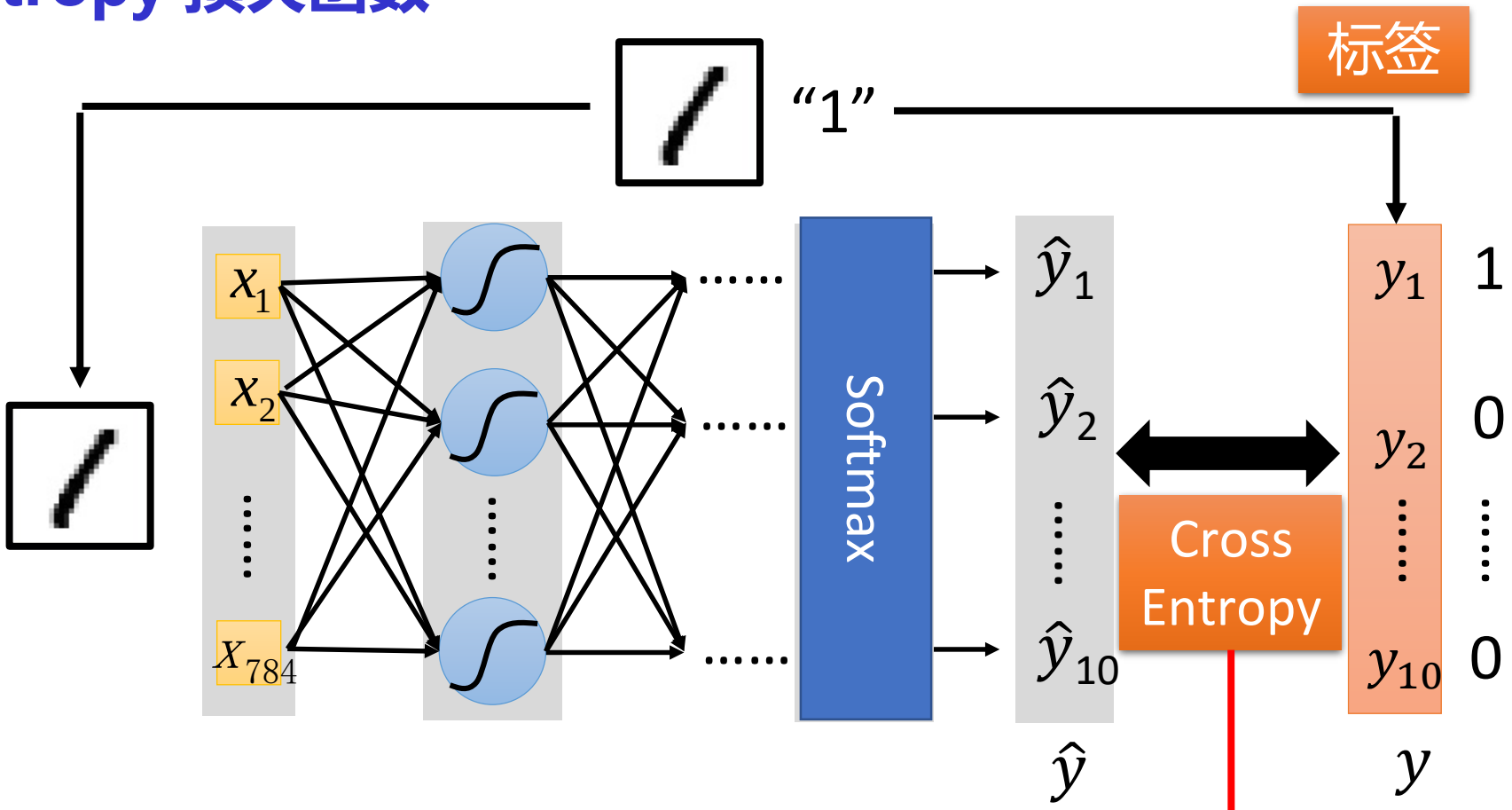
每一维输出大小可以理解为网络预测为对应标签的概率大小。

实现 MNIST 手写数字识别



注：本实验要求神经网络包含两个隐层，第一个隐层的神经元个数为192，第二个隐层的神经元个数为30。

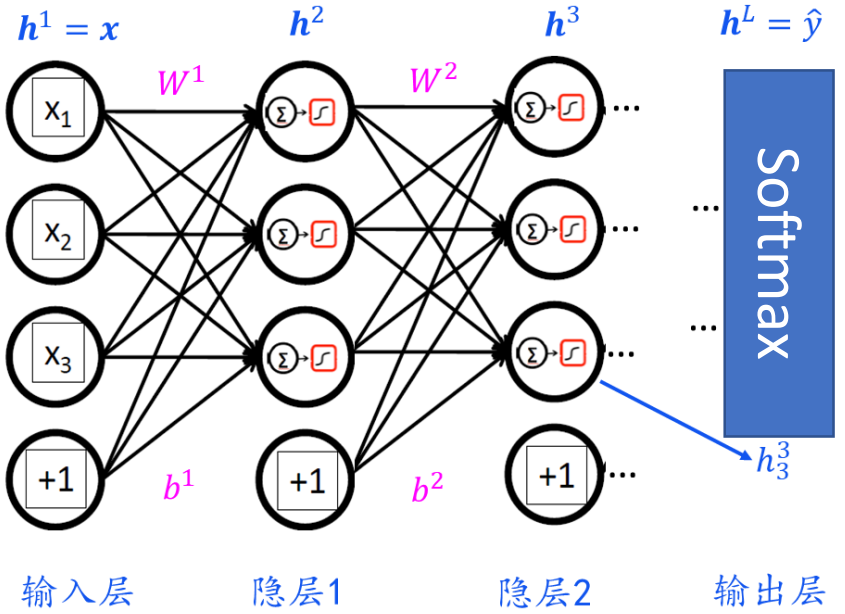
Cross-Entropy 损失函数



$$C(y, \hat{y}) = -\sum_{i=1}^{10} y_i ln\, \hat{y}_i$$

NN in practice

- (BP1) $\delta^L = (\boldsymbol{h}^L - y)$
- (BP2) $\delta^l = \sigma'(\boldsymbol{z}^l) \odot (\boldsymbol{W}^l \delta^{l+1})$
- (BP3) $\frac{\partial E}{\partial b^{l-1}} = \delta^l$
- (BP4) $\frac{\partial E}{\partial W^{l-1}} = \boldsymbol{h}^{l-1}(\delta^l)^T$



1. 输入训练集合$\{\boldsymbol{x}_i, \boldsymbol{i} = \boldsymbol{1}, \dots, \boldsymbol{n}\}$
2. **For i =1:n/m**
3. 对于批量数据中的每一个$\boldsymbol{x}$
   a. **前向传播**：对每个$l = 2, \dots, L$，计算$z^{x,l} = (\boldsymbol{W}^{l-1})^T \boldsymbol{h}^{x,l-1} + \boldsymbol{b}^{x,l-1}$和$\boldsymbol{h}^{x,l} = \sigma(\boldsymbol{z}^{x,L})$
   b. **输出误差$\boldsymbol{\delta}^{x,L}$**：计算向量 $\delta^{x,L} = (\boldsymbol{h}^{x,L} - y^x)$
   c. **反向传播误差$\boldsymbol{\delta}^{x,l}$ $(\boldsymbol{l} = \boldsymbol{L} - \boldsymbol{1}, \dots, \boldsymbol{2})$**：计算向量$\boldsymbol{\delta}^{x,l} = \sigma'(\boldsymbol{z}^{x,l}) \odot (\boldsymbol{W}^l \boldsymbol{\delta}^{x,l+1})$
4. 梯度下降：对每个$l = 2, \dots, L$，$\boldsymbol{W}^{l-1} \to \boldsymbol{W}^{l-1} - \frac{\eta}{m} \sum_x \boldsymbol{h}^{x,l-1} (\boldsymbol{\delta}^{x,l})^T$，$\boldsymbol{b}^{l-1} \to \boldsymbol{b}^{l-1} - \frac{\eta}{m} \sum_x \boldsymbol{\delta}^{x,l}$。
5. End i （1个epoch，迭代期）
6. 进行多个epoch循环，直至收敛。

实现 MNIST 手写数字识别

编程语言：

- Python3

实验平台：

- Visual Studio Code

简称 VS Code，是一款由微软开发且跨平台的免费源代码编辑器

标准库：

- import pickle
- import gzip
- import random

第三方库：

- import numpy as np
- import matplotlib.pyplot as plt

import numpy as np # 00

- np.reshape(*a, newshape, order='C'*)

  保持 a 数值不变的情况下赋予新的形状

```
>>> a = np.arange(6).reshape((3, 2))
>>> a
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> np.reshape(a, (2, 3))
array([[0, 1, 2],
       [3, 4, 5]])
```

- np.linalg.norm(*x, ord=None, axis=None, keepdims=False*)

  计算矩阵或向量 x 的范数

```
>>> a = np.arange(9) - 4
>>> a
array([-4, -3, -2, ...,  2,  3,  4])
>>> np.linalg.norm(a)
7.745966692414834
```

NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

import numpy as np # 01

- np.zeros(shape, dtype=float, order='C', *, like=None)

  返回 shape 形状的 array，其中的值为 0

```
>>> np.zeros(5)
array([ 0.,  0.,  0.,  0.,  0.])
>>> np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

- np.random.randn(*d0, d1, ..., dn*)

  返回随机生成的指定形状的标准正态分布，其均值为0，方差为1

```
>>> np.random.randn()
2.192387535537315
>>> 3 + 2.5 * np.random.randn(2, 4)
array([[-4.49401501,  4.00950034,
        -1.81814867,  7.29718677],

       [ 0.39924804,  4.68456316,
        4.99394529,  4.84057254]])
```

NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

import numpy as np # 02

- np.dot(a, b, out=None)

  两个数组的点积

```
>>> np.dot(3, 4)
12
>>> a = [[1, 0], [0, 1]]
>>> b = [[4, 1], [2, 2]]
>>> np.dot(a, b)
array([[4, 1],
       [2, 2]])
```

- np.argmax(a, axis=None, out=None, *, keepdims=<no value>)

  返回沿某一轴的最大值的索引

```
>>> b = np.arange(6)
>>> b
array([0, 5, 2, 3, 4, 5])
>>> np.argmax(b)
1
```

NumPy

NumPy是Python语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

import matplotlib.pyplot as plt

```
def plot_result(epochs, test_cost, test_accuracy, training_cost, training_accuracy, file_name):
    """绘制训练集和测试集的损失及准确率，并将所得结果保存"""
    epoch = np.arange(epochs)
    plt.subplot(1, 2, 1)
    plt.plot(epoch, test_cost, 'r', label='test_cost')
    plt.plot(epoch, training_cost, 'k', label='training_cost')
    plt.title("Cost Range")
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(epoch, test_accuracy, 'r', label='test_accuracy')
    plt.plot(epoch, training_accuracy, 'k', label='training_accuracy')
    plt.title("Accuracy Range")
    plt.legend()
    plt.savefig('../output/' + file_name)
```

matplotlib

Pyplot 是 Matplotlib 的子库，提供了和 MATLAB 类似的绘图 API。包含一系列绘图函数的相关函数，每个函数会对当前的图像进行一些修改，例如：给图像加上标记，生新的图像，在图像中产生新的绘图区域等等。

代码实现

代码的核心片段是一个 Network 类，用来表示一个神经网络，下面是初始化 Network 对象的代码

```
class Network(object):

    def __init__(self, sizes, cost=CrossEntropyCost):
        """
        参数 sizes 列表中包括神经网路中各层神经元的个数，例如 [784, 192, 30, 10] 表示一个包含两个
        隐藏层的神经网络，输入层包含 784 个神经元，第一隐层包含 192 个神经元，第二隐层包含 30 个神
        经元，输出层包含 10 个神经元。此外，对网络中的 biases 和 weights 进行了初始化。
        """
        self.num_layers = len(sizes)
        self.sizes = sizes
        self.default_weight_initializer()
        self.cost = cost

    def default_weight_initializer(self):
        """初始化 weights 和 biases 均值为 0，标准差为 1 的高斯分布，输入层的神经元不设置 biases """
        self.biases = [np.random.randn(y, 1) for y in self.sizes[1:]]
        self.weights = [np.random.randn(x, y) for x, y in zip(self.sizes[:-1], self.sizes[1:])]
```

代码实现

定义 one-hot 化函数，sigmoid 激活函数，softmax 函数

```
def vectorized_result(j):
    """将对应的数字 (0...9) 转化为对应的 one-hot 向量，该向量的 j 下标对应的取值为 1.0，其他为 0 """
    e = np.zeros((10, 1))
    e[j] = 1.0
    return e

def sigmoid(z):
    """sigmoid 激活函数"""
    return 1.0/(1.0+np.exp(-z))

def sigmoid_prime(z):
    """sigmoid 激活函数的导数"""
    return sigmoid(z)*(1-sigmoid(z))

def softmax(z):
    """softmax 函数"""
    e_x = np.exp(z - np.max(z))
    return e_x/e_x.sum()
```

代码实现

定义 Cross-Entropy 损失函数

```python
class CrossEntropyCost(object):

    @staticmethod
    def fn(a, y):
        """
        返回 a 和标签 y 之间的损失，其中 np.nan_to_num 是用来确保数值稳定性，比如 a 和 y 同时
        为 1.0 时，(1-y)*np.log(1-a) 应返回 nan，但使用 np.nan_to_num 可以使其转化为正确的数
        值 0.0
        """
        return np.sum(np.nan_to_num(-y*np.log(a)-(1-y)*np.log(1-a)))

    @staticmethod
    def delta(z, a, y):
        """返回输出层的误差方程 δ^L """
        return (a-y)
```

代码实现

实现 MNIST 数据加载，定义 load_data() 函数及 load_data_wrapper() 读取给定文件的数据并返回

```
def load_data():
    """
    以元组的形式加载 MNIST 数据集，包括训练集、测试集、验证集。其中，训练集是一个二维元组，第一
    维具有50000 组条目，每个条目有 784 个数值，代表单个 MNIST 图片的 28 * 28 = 784 像素值；第
    二维同样具有50000 组条目，每个条目对应该手写数字的标签，取值范围为 (0...9)。验证集和测试集
    的数据组成方式与训练集相似，只是仅包含 10000 张照片。
    """
    f = gzip.open('../data/mnist.pkl.gz', 'rb')
    training_data, validation_data, test_data = pickle.load(f, encoding='bytes')
    f.close()
    return (training_data, validation_data, test_data)


def load_data_wrapper():
    """
    在 load_data 函数的基础上，返回一个 (training_data, validation_data, test_data) 元组。其
    中，training_data 是包括 50000 个二维元组 (x, y) 的列表，x 是一个 784 维的 numpy.ndarray,
```

代码实现

实现 MNIST 数据加载，定义 load_data() 函数及 load_data_wrapper() 读取给定文件的数据并返回

```
    表示输入图像（28*28 = 784）的像素信息，y 是对应的 one-hot 标签向量。validation_data 和 test
    _data 是包括 10000 个二维元组（x，y）的列表，x 是一个 784 维的 numpy.ndarray，表示输入图像
    (28*28) 的像素信息，y 是对应的标签值。
    """
    tr_d, va_d, te_d = load_data()

    training_inputs = [np.reshape(x, (784, 1)) for x in tr_d[0]]
    training_results = [vectorized_result(y) for y in tr_d[1]]
    training_data = zip(training_inputs, training_results)

    validation_inputs = [np.reshape(x, (784, 1)) for x in va_d[0]]
    validation_data = zip(validation_inputs, va_d[1])

    test_inputs = [np.reshape(x, (784, 1)) for x in te_d[0]]
    test_data = zip(test_inputs, te_d[1])

    return (training_data, validation_data, test_data)
```

部分实验结果

可供参考的超参数取值：epochs = 20，mini_batch_size = 10，eta = 0.5

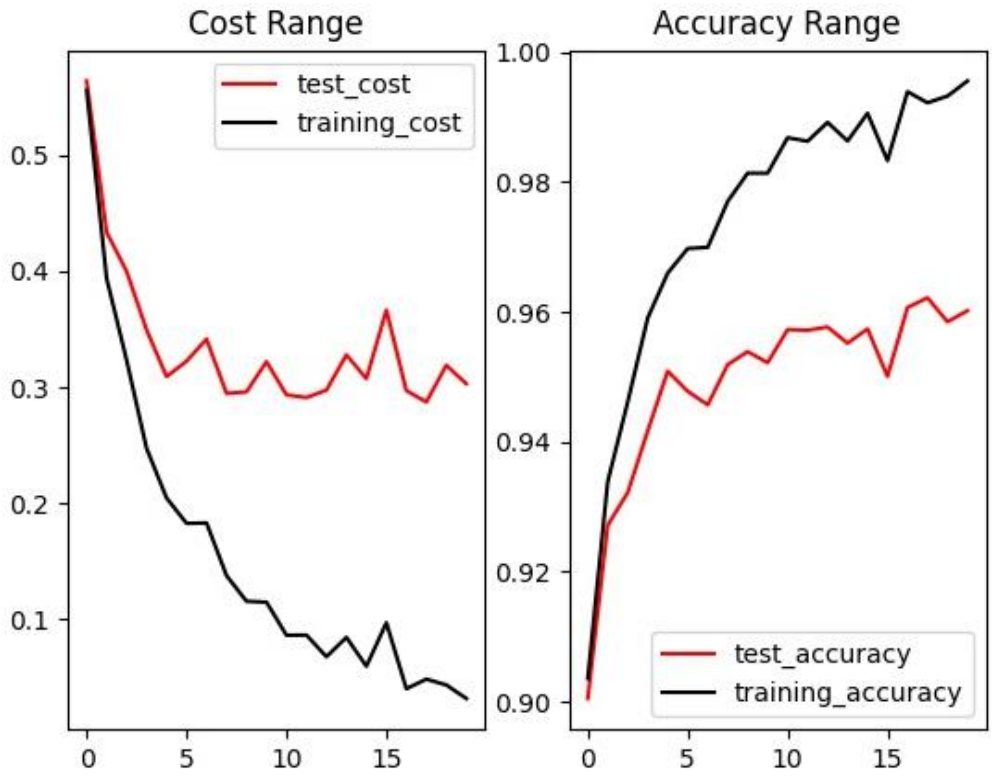


图1 交叉熵损失和准确率随迭代周期的变化曲线

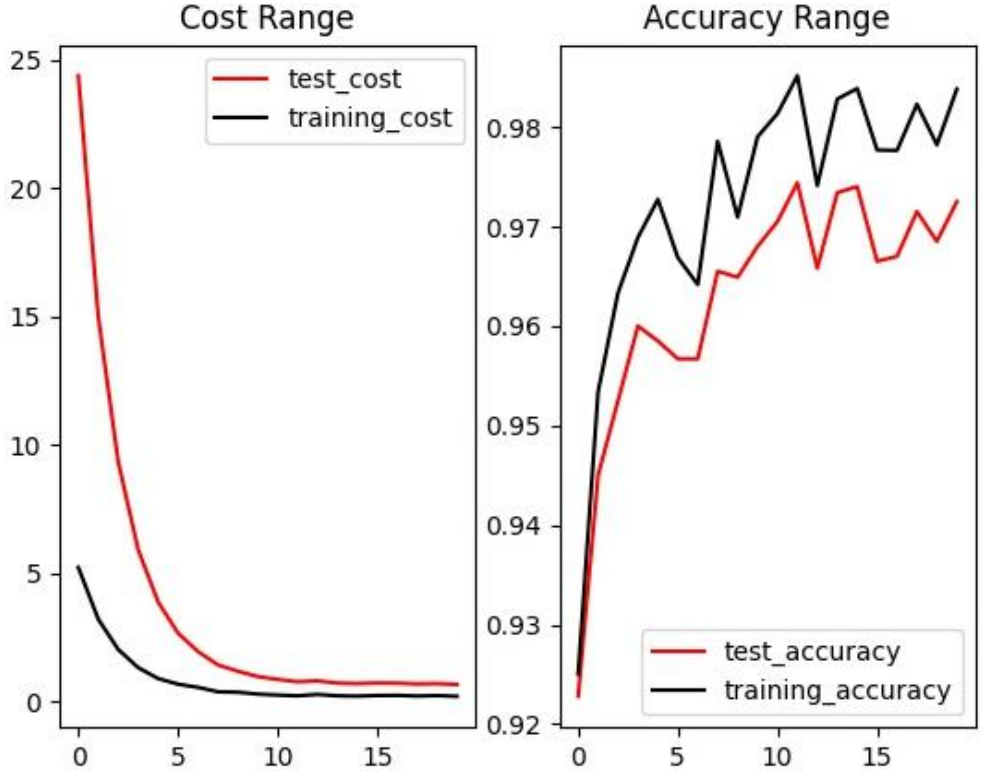


图2 在图1的基础上加入 *L2* 正则化后的结果

Thank You　Q & A

张浩：1050852440@qq.com

刘卓：lzpmbw@163.com