Neural Network and Applications
Homework 4

陈轶洲 MF20330010

December 2, 2020

1

首先设计类 DBMOON() 生成双月数据:

```python
class DBMOON:
    def __init__(self, n, r, w, d):
        self.n = n
        self.r = r
        self.w = w
        self.d = d

        self.dataA = None
        self.dataB = None
        self.dataDB = None

    def gen_dbdata(self):
        theta1 = np.random.uniform(0, np.pi, size=self.n)
        theta2 = np.random.uniform(-np.pi, 0, size=self.n)
        w1 = np.random.uniform(-self.w/2, self.w/2, size=self.n)
        w2 = np.random.uniform(-self.w/2, self.w/2, size=self.n)
        one = np.ones_like(theta1)

        self.dataA = np.array([(self.r+w1)*np.cos(theta1), (self.r+w2)*np.sin(theta1), one])
        self.dataB = np.array([self.r+(self.r+w2)*np.cos(theta2), -self.d+(self.r+w2)*np.sin(theta2), -one])

        dataA = self.dataA.copy()
        dataB = self.dataB.copy()

        return dataA, dataB

    def plot(self):
        fig = plt.figure()
        plt.scatter(self.dataA[0, :], self.dataA[1, :], marker='x')
        plt.scatter(self.dataB[0, :], self.dataB[1, :], marker='+')
        plt.savefig("dbdata")
```

Figure 1: 类 DBMOON

利用该类中的 gen_dbdata(1000，10，6，-2) 生成 2000 个样本点，如下图所示:

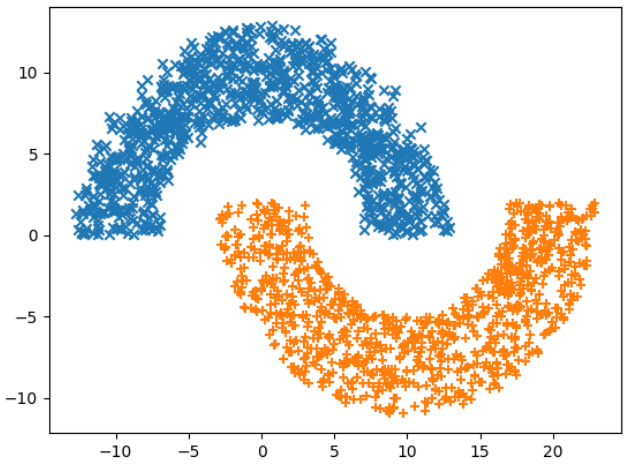

Figure 2: 样本点

设计类 SLP 实现单层感知器的功能：

```python
class SLP:
    def __init__(self, input_size, learning_rate):
        self.size = input_size
        self.lr = learning_rate
        self.w = np.random.randn((self.size)).reshape((self.size, 1))
        self.x = None
        self.out = None
        self.dw = None

    def forward(self, x):
        out = np.sign(self.w.T.dot(x))
        self.x = x
        self.out = out
        return out

    def backward(self, t):
        self.dw = self.lr*((t-self.out)*self.x)
        # print(self.x.shape)
        # print(self.dw.shape)
        dw = self.dw.copy().reshape((self.size, 1))
        return dw

    def train(self, x, t, iters):
        for i in range(iters):
            for j in range(x.shape[1]):
                y = self.forward(x[:, j])
                dw = self.backward(t[j].reshape((1,)))
                # print('w:{}'.format(self.w))
                # print('dw:{}'.format(dw))
                self.w += dw
        result = self.forward(x)
        return result
```

Figure 3: 单层感知器

利用此单层感知器进行数据分类的学习，完成后画出决策边界：

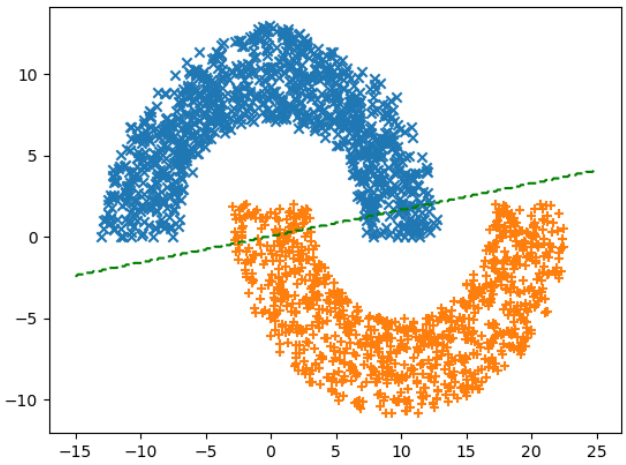

Figure 4: 单层感知器分类结果

接着利用大作业中完成的 BP 神经网络训练分类模型，该模型包含两个隐藏层，下图为 BP 神经网络训练得到的决策边界:

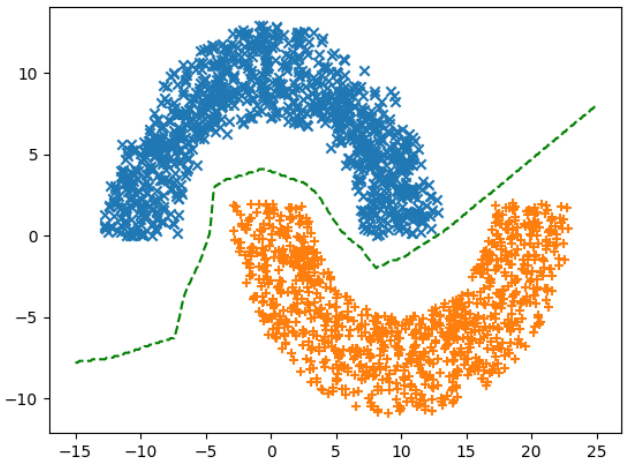

Figure 5: BP 神经网络分类结果

综上所述，使用单层感知器无法进行非线性分类，而 BP 神经网络较好地完成了双月数据的分类。这是因为 BP 神经网络使用了隐藏层的神经元帮助学习。

显而易见的，单个神经元只能学习到线性函数，而每个隐藏层神经元只需要学习决策边界中的某一段线性特征，利用隐藏层的堆叠将这些线性特征进行仿射变换从而学习到整个非线性的决策边界。

2

首先下载 optdigits 数据集中的 optdigits.tra、optdigits.tes，并从中获得训练集与测试集：

```
8   digits_train = pd.read_csv("../dataset/optdigits.tra", header=None)
9   digits_test = pd.read_csv("../dataset/optdigits.tes", header=None)
10  # print(digits_train)
11  x_train = np.array(digits_train[np.arange(64)])
12  y_train = np.array(digits_train[64])
13  x_test = np.array(digits_test[np.arange(64)])
14  y_test = np.array(digits_test[64])
15  # print(y_test.shape)
```

Figure 6: optdigits 数据集

调用 sklearn 库中的 MLPClassifier 类实现单隐藏层 BP 网络：

```
17  clf = MLPClassifier(hidden_layer_sizes=(100,))
18  clf.fit(x_train, y_train)
19  # print(clf.predict(x_test))
20  print(clf.score(x_test,y_test))
```

Figure 7: MLPClassifier

通过实验求出不同神经元个数下模型最终的预测正确率：

Table 1: 神经元个数对结果影响

神经元	10	20	30	40	50	60	70	80	90	100
准确率	0.933	0.956	0.954	0.958	0.956	0.954	0.963	0.960	0.962	0.968

通过上表绘制出神经元个数与预测准确率的关系图，由图可知随着隐藏层神经元个数的增加，预测准确率呈震荡上升趋势：

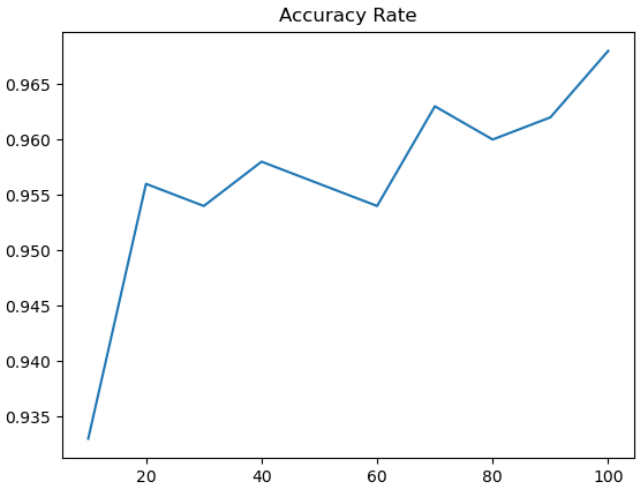

Figure 8: 神经元与准确率关系图

进一步提高预测准确率的方法:

1) 增加迭代轮次;

2) 加深模型的隐藏层;

3) 选择不同的激活函数。

3

利用大作业一中完成的多层感知器训练规模为 1-5-1 的神经网络，选择均方误差作为损失函数，在完成 8000 轮 epoch 训练后，模型误差为 0.01063849

```
TERMINAL   PROBLEMS 22   OUTPUT   DEBUG CONSOLE

No.7996 epoches' loss:[0.01066573]
No.7997 epoches' loss:[0.01083111]
No.7998 epoches' loss:[0.01064516]
No.7999 epoches' loss:[0.01067478]
No.8000 epoches' loss:[0.01063849]
params:{'W1': array([[ 0.48179381, -0.92318943, -0.14051506,  0.34767571, -0.73806371]])
       [-1.35000012],
       [-0.20548011],
```

Figure 9: 训练误差

由下图可知训练结果已收敛:

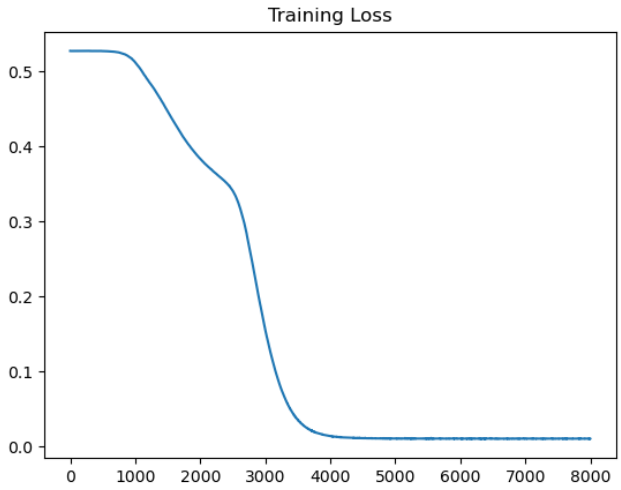

Figure 10: sin(x) 曲线损失函数

利用 100 个样本点训练得到的拟合结果

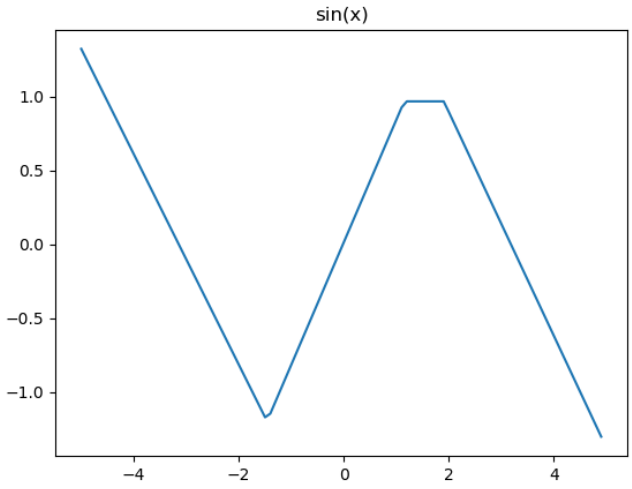

Figure 11: sin(x) 拟合曲线

此模型隐藏层参数如下:

输入层-隐藏层：

$$W_1 = \begin{bmatrix} 0.48179381 \\ -0.92318943 \\ -0.14051506 \\ 0.34767571 \\ -0.73806371 \end{bmatrix} \quad B_1 = \begin{bmatrix} -0.91727367 \\ -1.35000012 \\ -0.20548011 \\ -0.66199873 \\ 0.84841974 \end{bmatrix} \tag{3.1}$$

隐藏层-输出层：

$$W_2 = \begin{bmatrix} -1.03476218 \\ 1.63334207 \\ 0.24860539 \\ -0.74677516 \\ -1.12417803 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.9694142 \end{bmatrix} \tag{3.2}$$

以输入 $x = -5$ 为例，计算过程为：

$$y = Relu(W_1^T x + B_1) = \begin{bmatrix} 0.04631395 \\ 0 \\ 0 \\ 0.03335269 \\ 0 \end{bmatrix} \quad z = W_2^T y + B_2 = \begin{bmatrix} 0.89658331 \end{bmatrix} \tag{3.3}$$