PB22111645 朱恩松

1.代码解释

在源程序已有相关注释，此处不再给出。

2.鸢尾花数据集超参数

1.mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=1000, random_state=42)

```
测试集合的 y 值:  [1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0,
2, 0, 2, 2, 2, 2, 2, 0, 0]
神经网络预测的的 y 值:  [np.int64(1), np.int64(0), np.int64(2), np.int64(1),
np.int64(1), np.int64(0), np.int64(1), np.int64(2), np.int64(1), np.int64(1),
np.int64(2), np.int64(0), np.int64(0), np.int64(0), np.int64(0), np.int64(1),
np.int64(2), np.int64(1), np.int64(1), np.int64(2), np.int64(0), np.int64(2),
np.int64(0), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2),
np.int64(0), np.int64(0)]
预测的准确率为:  1.0
层数为:  3
迭代次数为:  600
损失为:  0.0663501397624132
激活函数为:  softmax
```

2.mlp = MLPClassifier(hidden_layer_sizes=(50, 50), max_iter=400, random_state=42)

```
测试集合的 y 值:  [1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0,
2, 0, 2, 2, 2, 2, 2, 0, 0]
神经网络预测的的 y 值:  [np.int64(1), np.int64(0), np.int64(2), np.int64(1),
np.int64(1), np.int64(0), np.int64(1), np.int64(2), np.int64(1), np.int64(1),
np.int64(2), np.int64(0), np.int64(0), np.int64(0), np.int64(0), np.int64(1),
np.int64(2), np.int64(1), np.int64(1), np.int64(2), np.int64(0), np.int64(2),
np.int64(0), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2),
np.int64(0), np.int64(0)]
预测的准确率为:  1.0
层数为:  4
迭代次数为:  318
损失为:  0.04417235559193613
激活函数为:  softmax
```

3.mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)

```
测试集合的 y 值:  [1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0,
2, 0, 2, 2, 2, 2, 2, 0, 0]
神经网络预测的的 y 值:  [np.int64(1), np.int64(0), np.int64(2), np.int64(1),
```

```
np.int64(1), np.int64(0), np.int64(1), np.int64(2), np.int64(1), np.int64(1),
np.int64(2), np.int64(0), np.int64(0), np.int64(0), np.int64(0), np.int64(1),
np.int64(2), np.int64(1), np.int64(1), np.int64(2), np.int64(0), np.int64(2),
np.int64(0), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2),
np.int64(0), np.int64(0)]
预测的准确率为： 1.0
层数为： 3
迭代次数为： 492
损失为： 0.06268353735470718
激活函数为： softmax
```

4.分析

由1，2对比可知，增大层数可以减低迭代次数与损失。 由1，3对比可知，增大神经元个数可以减低迭代次数与损失。 由2，3对比可知，当神经元个数相同时，将神经元适当分配到不同层时，可以降低迭代次数与损失。

3.用时

约一天。