实验文档

目录

代码基本结构

整个 BP 网络由 network 包下的 DataNode、NeuronNode、NeuronSystem 三个类组成，NeuronNode。DataNode 是存放训练数据的最小单元，有两个数组，一个是输入数组，表征输入的向量值；另一个是期望输出数组，代表系统应该达到的预测输出向量值。

NeuronNode 是神经网络中单个的神经元，分为输入层 INPUT、隐藏层 HIDDEN、输出层 OUTPUT 三种类型，目前有两种 Sigmoid 函数可供选择：log 和 tan。单个神经元有四个属性，分别是正向和反向的输入、输出值，涉及到正向和反向两种场景，在正向传播的时候，正向输出的值应该是正向输入值经过 Sigmoid 函数映射，反向输出的值应该是反向输入的值乘以 Sigmoid 函数在该节点的偏导数。并且每次更新正向输入和反向输入的值的时候，会导致正向和反向输出的值改变。

NeuronSystem 是维护整个 BP 网络的系统，主要属性有网络结构（输入层节点数、隐藏层结构、输出层节点数）、神经元 NeuronNode 对象数组（输入层、隐藏层、输出层）、每相邻两层之间的权重数组信息（用三维数组表示）、每一个神经元对应的偏置（bias 用二维数组表示）、以及学习率（调整的步长 rating）。

在对系统进行初始化时，需要提供网络结构、学习率信息。然后可以利用已有数据对网络进行训练，主要分为三个步骤。首先，对于每一个输入向量，神经网络从前往后依次传递更新值。然后，根据预期的输出，从后往前更新每一层的总误差对当前神经元输出值的偏导。最后，利用反向传播得到的偏导，从前往后更新每一层的权重和偏置值。

在对已有数据的利用上，每一次训练为将所有的输入和预期对系统进行调整，然后将原有数据顺序打乱，再次进行训练，往往需要进行上万次训练。最后，在预测输出时，可以正向设置输入层的值，然后向后传递，最后在输出层能看到预测值。

对于图片识别，在 bmp 包下有一个 BMPResolver 类，传入图片的宽度、高度和图片路径后，对象对黑白 BMP 位图进行处理，并且把 BMP 读取顺序从上到下调整为从左到右，然后将二维的特征信息数组转换为一维的特征数组，提供输出接口。

调用的时候，模拟 sin 函数在 Main 类中直接调用 testSin(dataSize, trainingTimes, SIGMOID_FUNCTION_TYPE)，dataSize 为训练集大小，trainingTimes 为训练次数，最后的参数为 sigmoid 函数类型。模拟图片识别时，直接调用 testClassification(trainingTimes, trainingPercent, SIGMOID_FUNCTION_TYPE)，trainingPercent 为所有图片中训练集所占比重（0-1），相应的剩余为测试集，其他参数同上。网络结构和 learning rate 已经调整到

较优状态预设在函数中。

不同网络结构、网络参数的实验比较

Part1 模拟 sin 函数

tan-sigmoid 函数：

网络结构和学习率：

经过漫长的尝试不同隐藏层结构和不同的 learning rate（具体实现为循环套循环），初步发现网络结构在两层、三层时，每层神经元在 5 个以上、10 个以内时，learning rate 在 0.1-0.2 左右时，能够取得较小的预测误差。然后对 learning rate 进行循环测试，然后进行更加细微的迭代测试。

最后结果发现在隐藏层结构为：[7, 7]，rating = 0.134 时 五万次训练后平均预测误差 error = 0.003，耗时约为十秒。而且在一定范围内，随着训练次数的增多，效果越好。

训练次数：

在二十万次迭代以内当然是越多越好啊，随着次数的增多，预测误差会越来越小，但是耗时也会显著上升。如五万次训练耗时十秒，误差 0.003；十万次训练后耗时二十五秒，误差为 0.0024；十五万次训练耗时四十五秒，误差 0.02；二十万次训练后误差能达到 0.0014，但耗时约为一分钟。

但是再更多的时候，迭代次数上升，耗时增加，却并不能获得预测误差的减小。如经过四十万次训练后误差为 0.0014，五十万次迭代误差为 0.0017，效果反而更差。

log-sigmoid 函数：

注意事项：由于 log-sigmoid 函数值域为 0-1，所以在模拟 sin 函数时，我们需要将原来的值域-1 到 1 投影到 0-1 上。

网络结构和学习率：

以两万次训练为例，经过类似 tan-sigmoid 函数的循环测试，发现网络结构在一层时，神经元个数在 8 左右，learning rate 在 3.55 时，能够取得接近 0.004 的平均误差。然后缩小查找范围，再次测试相对较优的网络结构和学习率。

最后发现，仍然在神经元结构为[9]，learning rate 在 3.5 左右能取得 0.003 的平均预测误差。然后经过调整 learning rate，发现 log-sigmoid 函数对 learning rate 的变化较为迟钝，比起 tan-sigmoid 函数往往需要较大的 rating 变动，才能看到明显的预测误差变动。并且发现 learning rate 在 5 以内基本都能满足误差小于 0.001 的要求，并且具体数值对误差的影响不大。最终得出的较优参数设置为：网络结构[9]，learning rate 为 3.5，预计误差为 0.003，耗时为两秒。

训练次数：

同上，在一定范围内，训练次数的增多能够降低预测误差。当训练次数为三万次时，耗时三秒，预测误差为 0.002。当次数再增多时，如五万次训练，耗时五秒，预测误差 0.003；十万次训练，耗时十秒左右，预测误差 0.04，即更多次数训练也不能降低预测误差。

两种函数比较：

从训练次数和训练用时上看，对于 sin 函数的预测，log-sigmoid 函数表现的比 tan-sigmoid 函数更为出色。并且 log-sigmoid 函数能够容忍一个更大范围变动的 learning rate，并且保持预测误差 0.002（三万次，耗时三秒）左右。另外，log-sigmoid 函数比起 tan-sigmoid

函数适合更简单的隐藏层神经元结构，即 log-sigmoid 函数在耗时、learning rate 容错、隐藏层结构等方面都具有优势，能取得更小的预测误差。

Part2 对 14 个手写汉字进行分类

网络结构和学习率：

参数的确定同 part1，先粗范围筛选可能的较优参数，初步确定为两层网络结构，并且第一层 70-100、第二层 60-80 之间，rating 为 0.55-1.55 以内时，经过 14 种各 200 个单类样本的 50 次迭代后，对 14 种各 50 个单类样本预测准确率较能达到。然后经过进一步确定，发现第一层在 90-94、第二层在 72-74 时，learning rate 为 1.05 时，预测准确率能达到 0.85 以上。经过反复比较，最终确定了相对较优的参数设置如下：网络结构[90,74]、学习率 1.05，平均预测准确率能达到 0.85。

下图为较好的 5 个网络结构（[90, 74]、[94, 72] 、[94, 73] 、[91, 74] 、[93, 73]），且 learning rate 为 1.05 时，在 200*14 组数据训练 50 此后，20 次预测，每次预测 50*14 数据，每次的平均预测准确率都能达到 0.85 以上，且最低预测准确率都高于 0.82（一般为 0.83 及以上）。

```
    0.8613  0.8310  0.8407  0.8599  0.8352  0.8709  0.8613  0.8475  0.8530  0.8571  0.8668  0.8365  0.8599  0.8585  0.8640  0.8324  0.8530  0.8475  0.8599  0.8448
hidden layer structure: [90, 74], rating: 1.05
min average predict accurate rate: 0.8310439560439561
total average predict accurate rate: 0.8520604395604394
    0.8709  0.8516  0.8434  0.8571  0.8640  0.8668  0.8668  0.8393  0.8530  0.8764  0.8668  0.8681  0.8503  0.8489  0.8777  0.8530  0.8709  0.8668  0.8544  0.8393
hidden layer structure: [94, 72], rating: 1.05
min average predict accurate rate: 0.8392857142857143
total average predict accurate rate: 0.8592719780219781
    0.8324  0.8558  0.8750  0.8750  0.8434  0.8393  0.8516  0.8159  0.8462  0.8599  0.8503  0.8723  0.8489  0.8338  0.8352  0.8613  0.8365  0.8475  0.8709  0.8626
hidden layer structure: [94, 73], rating: 1.05
min average predict accurate rate: 0.8159340659340659
total average predict accurate rate: 0.8506868131868129
    0.8571  0.8571  0.8448  0.8599  0.8668  0.8407  0.8695  0.8599  0.8448  0.8750  0.8283  0.8462  0.8585  0.8723  0.8599  0.8448  0.8544  0.8640  0.8544  0.8214
hidden layer structure: [91, 74], rating: 1.05
min average predict accurate rate: 0.8214285714285714
total average predict accurate rate: 0.8539835164835164
    0.8420  0.8475  0.8407  0.8736  0.8516  0.8462  0.8407  0.8530  0.8516  0.8640  0.8489  0.8475  0.8379  0.8599  0.8791  0.8475  0.8434  0.8764  0.8736  0.8516
hidden layer structure: [93, 73], rating: 1.05
min average predict accurate rate: 0.8379120879120879
total average predict accurate rate: 0.8538461538461537
```

对反向传播算法的个人理解

具体的调整规则就是推公式啦，自己感觉 BP 算法的模型训练就像一个从后向前调整的思想。对于每一个训练数据，将模型的预测输出与期望的输出进行比较，然后将误差从后向前传播，并且利用梯度下降算法，对神经元的权重、网络结构的偏置进行调整。

个人感觉比较麻烦的是参数调整，唉，目前的做法是多层循环，慢慢熬时间炼丹。相对来说两三层的神经网络结构比较适合图片识别部分，而对 sin 的模拟比较适合用一层神经元，并且两种情况下都比较适合用 log-sigmoid 激活函数，log-sigmoid 激活函数的好处已经在 part1 部分描述了。

另外，在对模型的训练过程中，我选用了 80%的数据作为训练集，20%数据用来验证模型的预测准确性，并且每次的训练集和测试集都是随机划分（也根据手写体的不同种类平均随机划分），并且验证集绝对不用于调整模型参数，只用来评估模型的准确率。然后进行数十次

完整的测试+验证，得出了以上的较优参数。

14307110274
文进