实验文档

目录

[**代码基本结构** 1](#_Toc495331240)

[**不同网络结构、网络参数的实验比较** 2](#_Toc495331241)

[**Part1 模拟sin函数** 2](#_Toc495331242)

[**tan-sigmoid函数：** 2](#_Toc495331243)

[**log-sigmoid函数：** 2](#_Toc495331244)

[**两种函数比较：** 2](#_Toc495331245)

[**Part2 对14个手写汉字进行分类** 3](#_Toc495331246)

[**对反向传播算法的个人理解** 3](#_Toc495331247)

**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：**

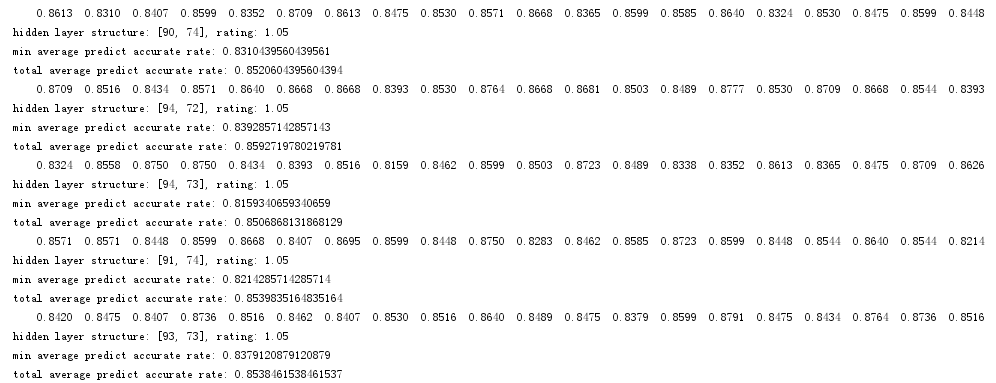
从训练次数和训练用时上看，对于sin函数的预测，log-sigmoid函数表现的比tan-sigmoid函数更为出色。并且log-sigmoid函数能够容忍一个更大范围变动的learning rate，并且保持预测误差0.002（三万次，耗时三秒）左右。另外，log-sigmoid函数比起tan-sigmoid函数适合更简单的隐藏层神经元结构，即log-sigmoid函数在耗时、learning rate容错、隐藏层结构等方面都具有优势，能取得更小的预测误差。

**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）。



**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