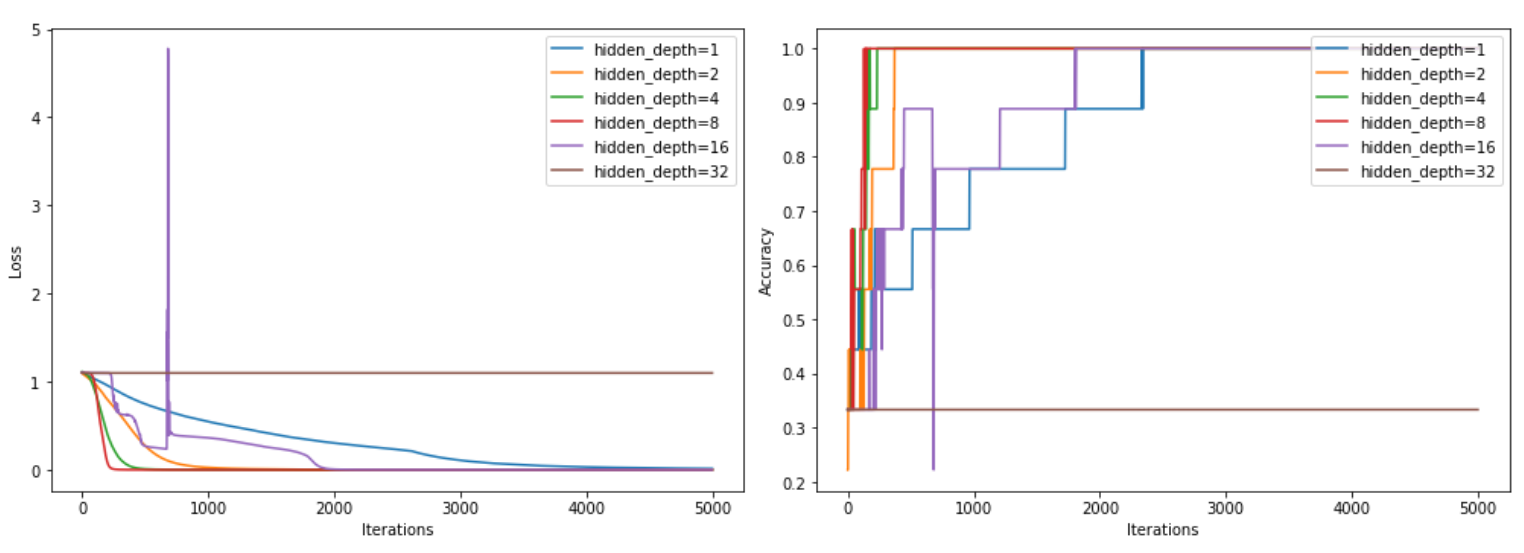
Lecture 9编程作业

李星毅 U201712072 自实1701

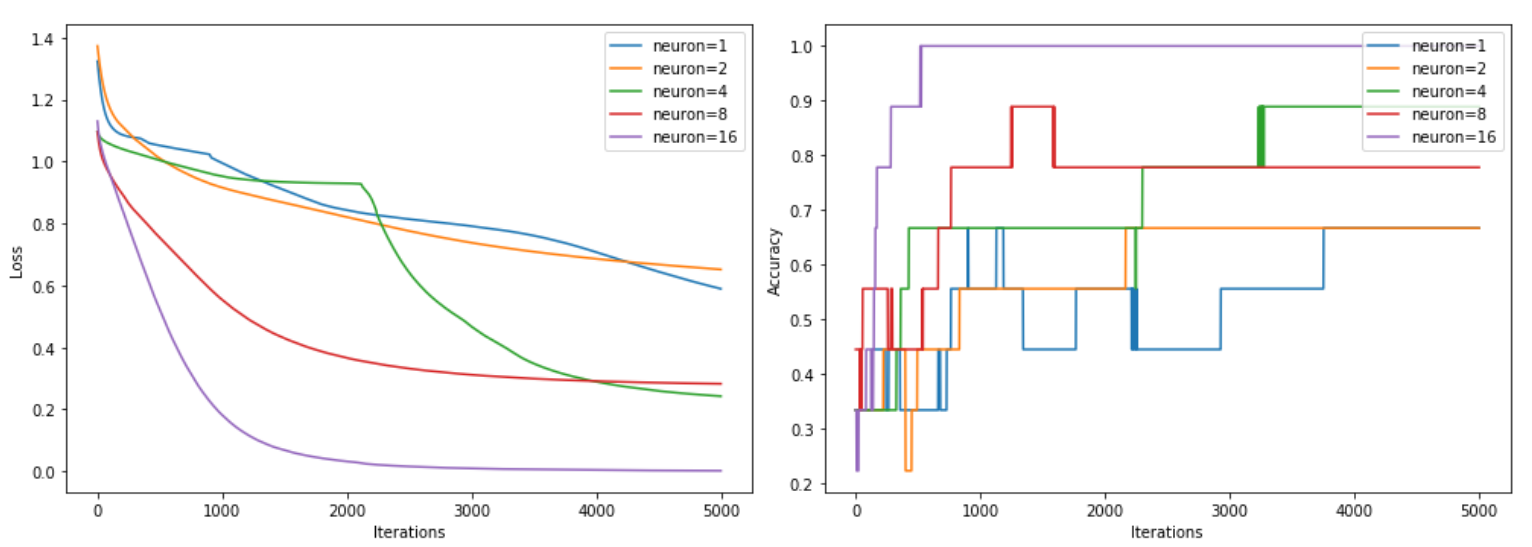
注：本次实验的系统是Windows 10，采用PyTorch框架，利用GTX1060加速。在实验中，我使用的损失函数是交叉熵，由于是多类分类，输出层激活函数选用Softmax。下面我利用以下三个题目的数据进行实验，从隐含层数（实验一）、隐藏层神经元个数（实验二）、学习率（实验三）、激活函数（实验四）等方面讨论他们对神经网络性能的影响。需要说明的是，我采用控制变量法来研究各个因素对神经网络性能的影响，其中，实验一中控制隐藏层神经元个数为16，学习率为0.001，优化器为Adam，激活函数是ReLU，而隐藏层数的变化为1-2-4-8-16-32；实验二控制隐藏层数为1层，学习率为0.001，优化器为Adam，激活函数是ReLU，而隐藏层神经元个数的变化为2-4-8-16-32；实验三控制隐藏层神经元个数为16，控制隐藏层数为8层，优化器为Adam，激活函数是ReLU，而学习率的变化为0.0001-0.001-0.01-0.1；实验四控制隐藏层神经元个数为16，控制隐藏层数为8层，学习率为0.001，优化器为Adam，而激活函数的变化为ReLU-tanh-Sigmoid。

1，现有如下9个样本，假设样本（3，0.4）、（1，1）和（3，3）属于第一类，样本（2，0.5）、（3，1）和（1，3）属于第二类，样本（1，2）、（2，2）和（3，2）属于第三类，请自行设计神经网络实现上述样本的分类，希望能通过设计不同的隐含层数、每层的节点数、不同的学习率、不同的激活函数等对实验结果进行讨论。

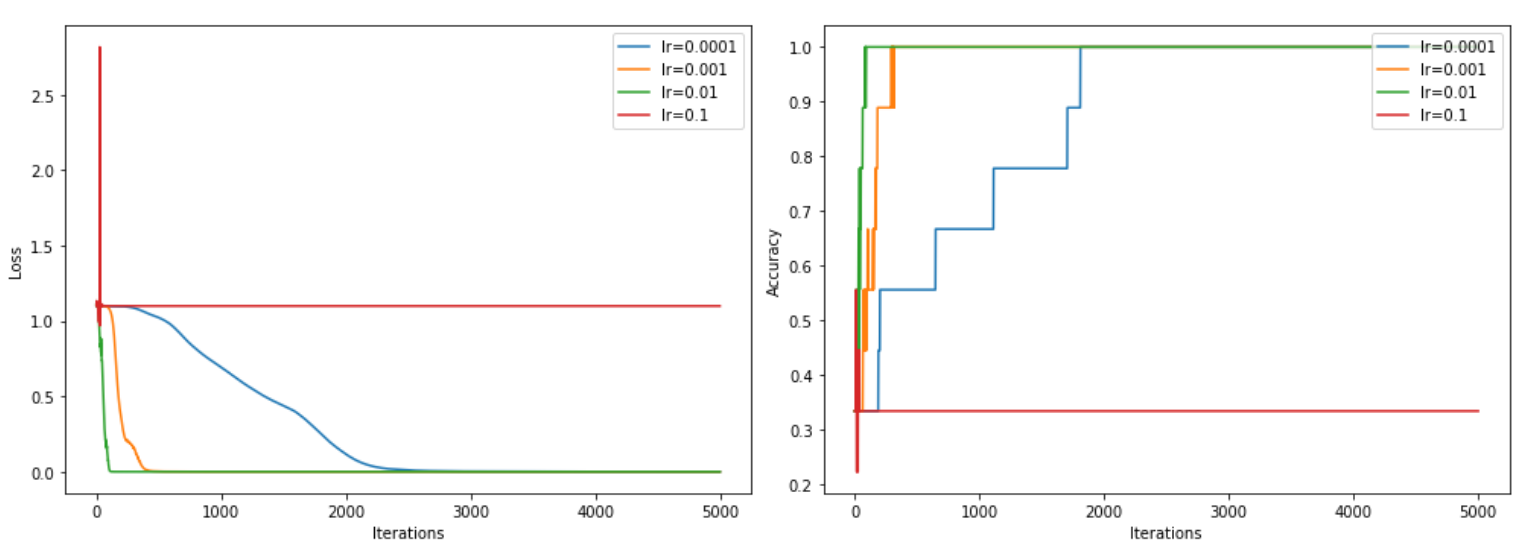
1. 隐含层数对网络性能的影响



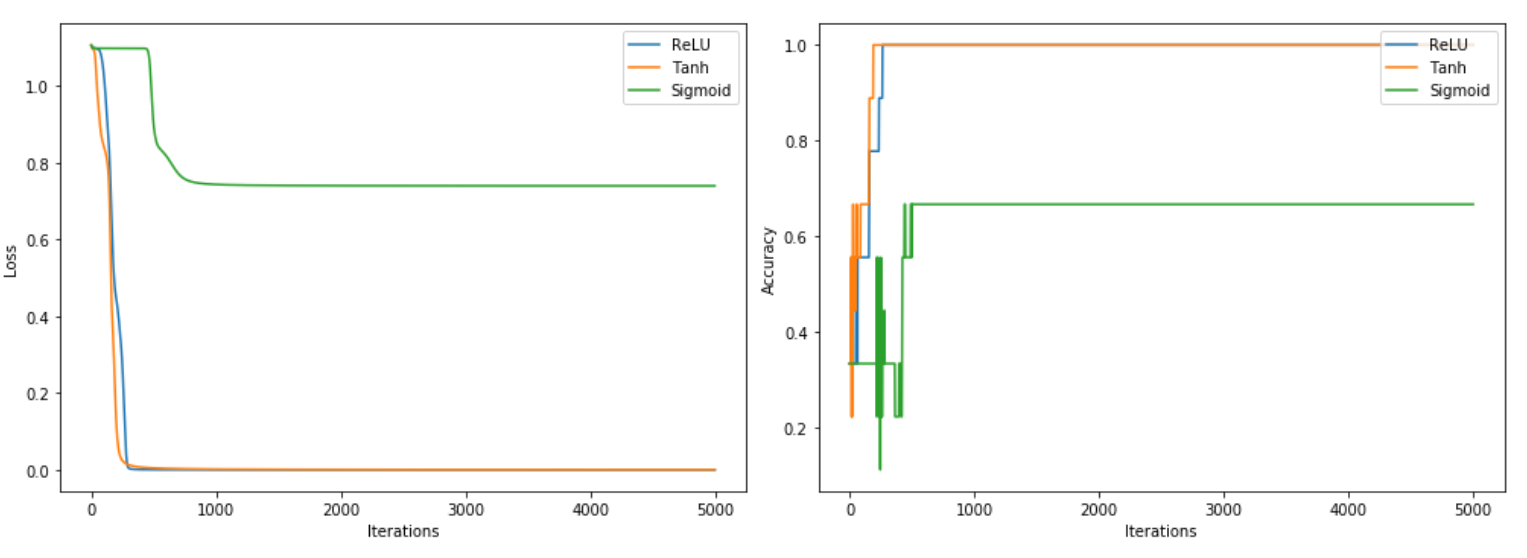
1. 隐藏层神经元个数对网络性能的影响



1. 学习率对网络性能的影响

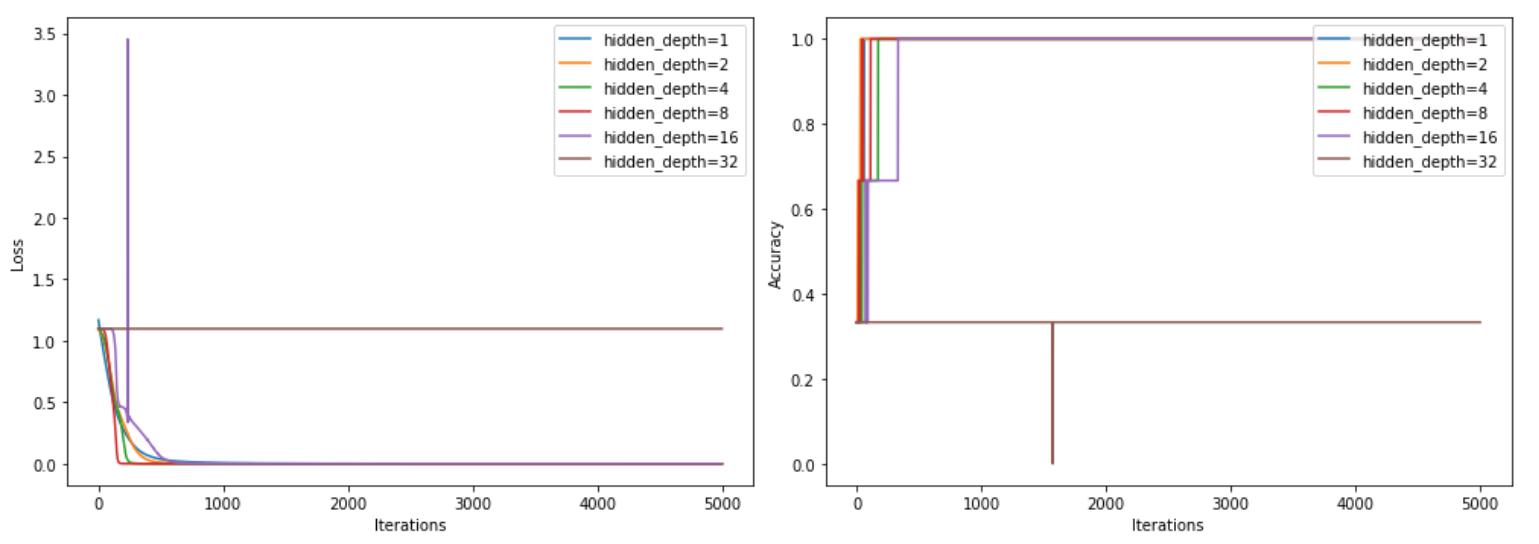


1. 激活函数对网络性能的影响

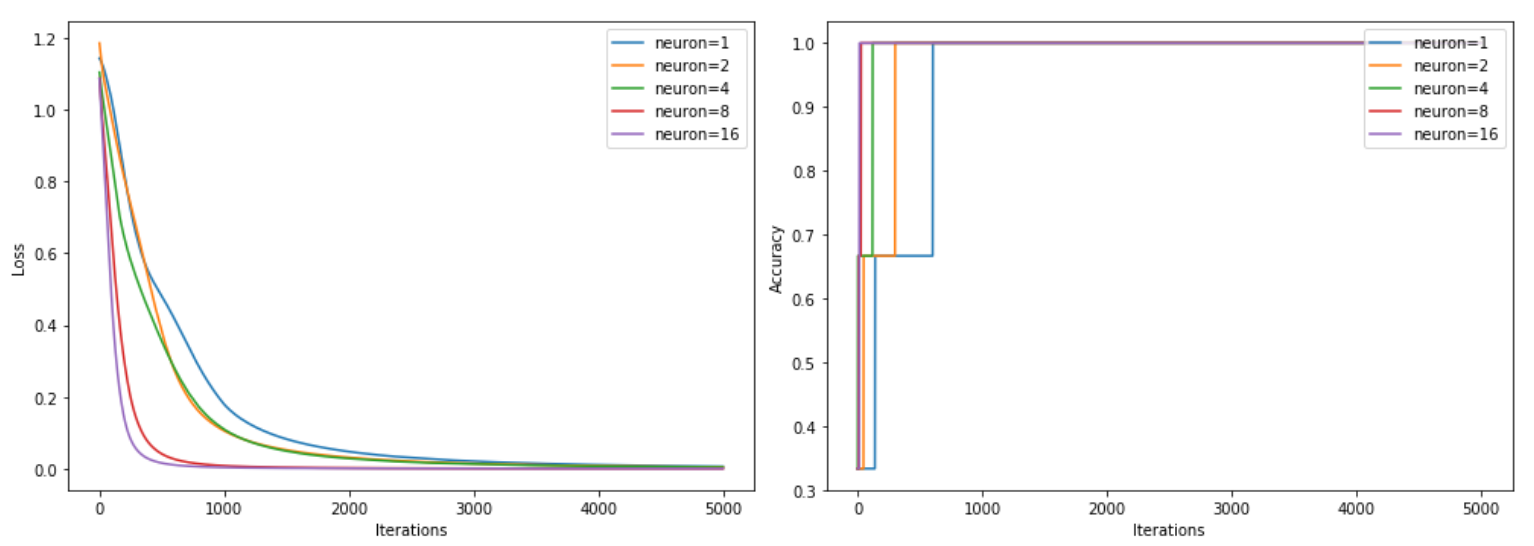


2，用3\*3的矩阵共9维特征来表示字母，假设向量表示字母“I”，向量表示字母“T”，向量表示字母“U”，请自行设计神经网络实现对这三个字母的识别，希望能通过设计不同的隐含层数、每层的节点数、不同的学习率、不同的激活函数等对实验结果进行讨论。

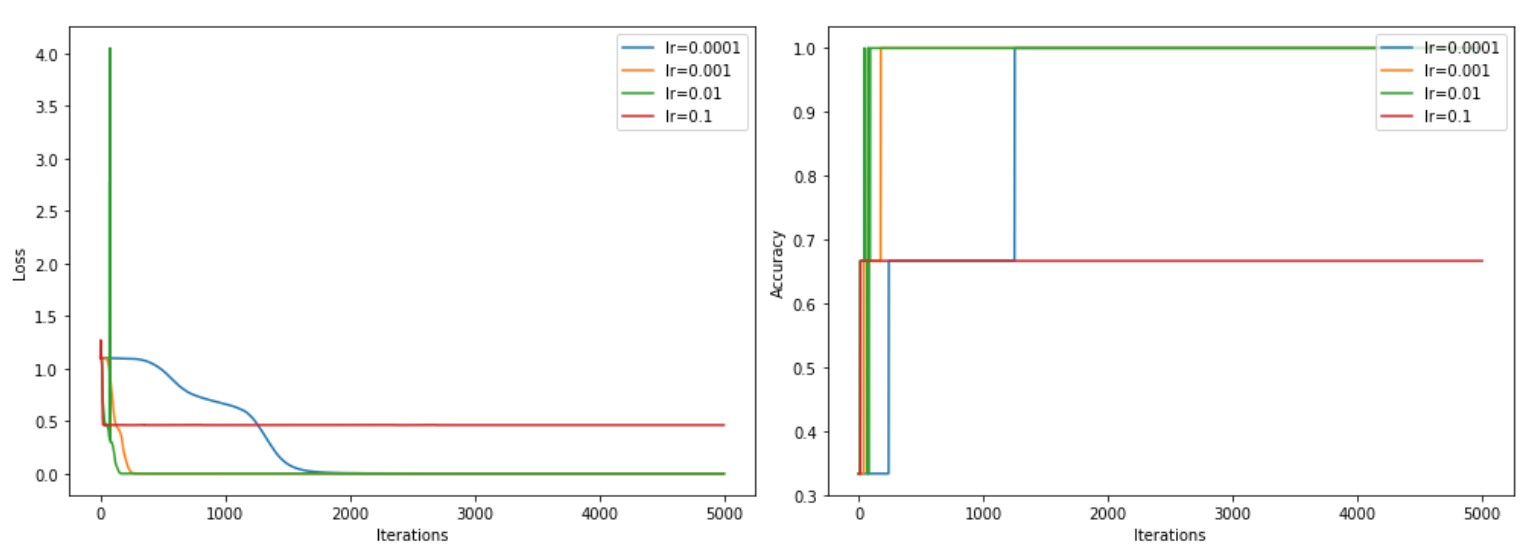
1. 隐含层数对网络性能的影响



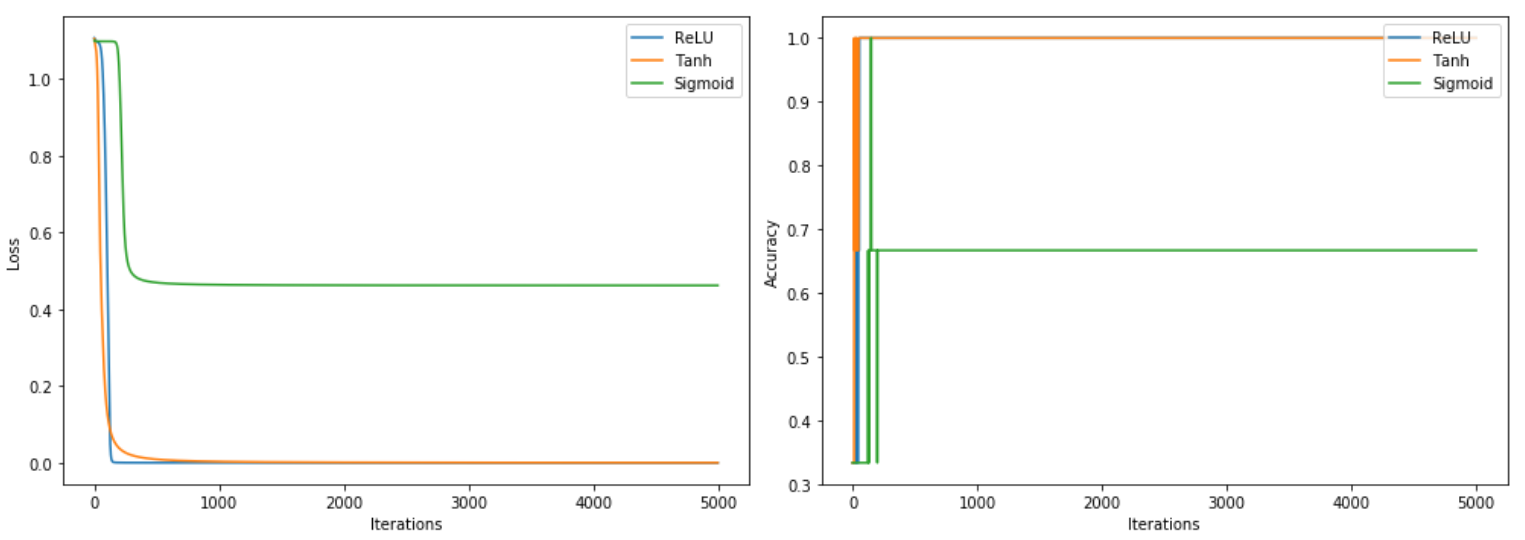
1. 隐藏层神经元个数对网络性能的影响



1. 学习率对网络性能的影响

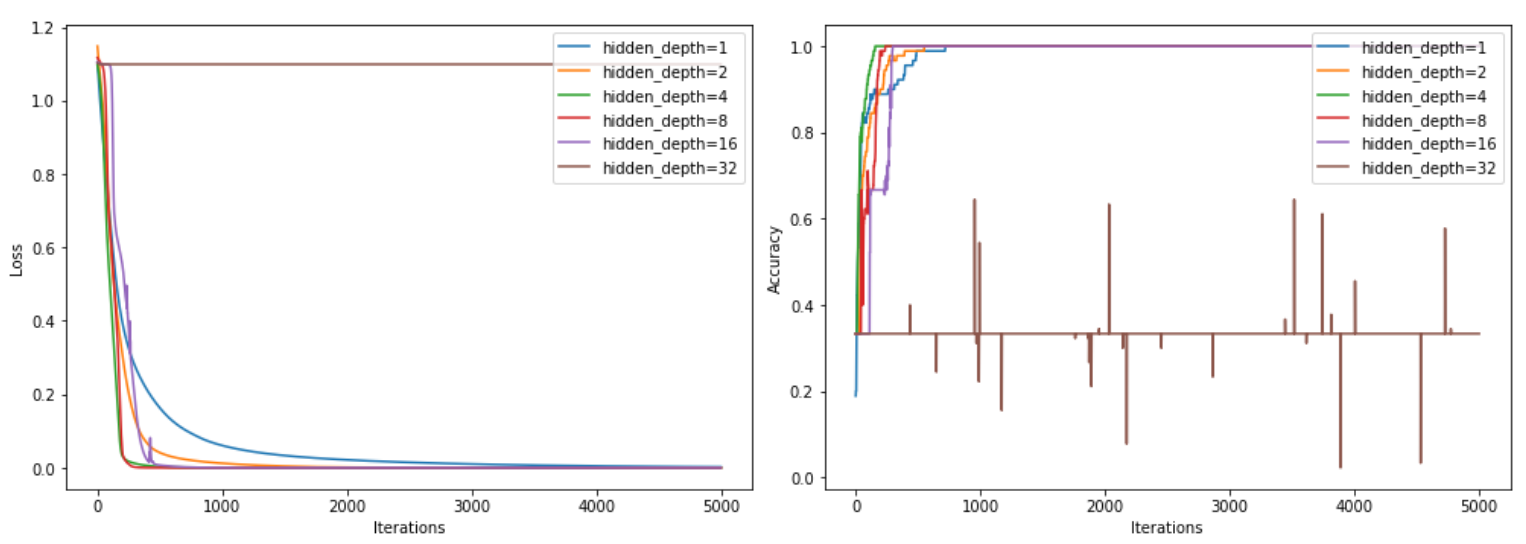


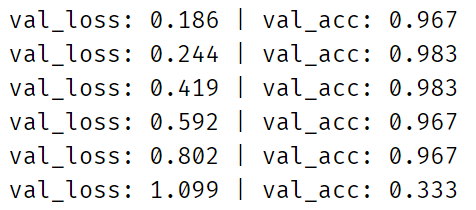
1. 激活函数对网络性能的影响



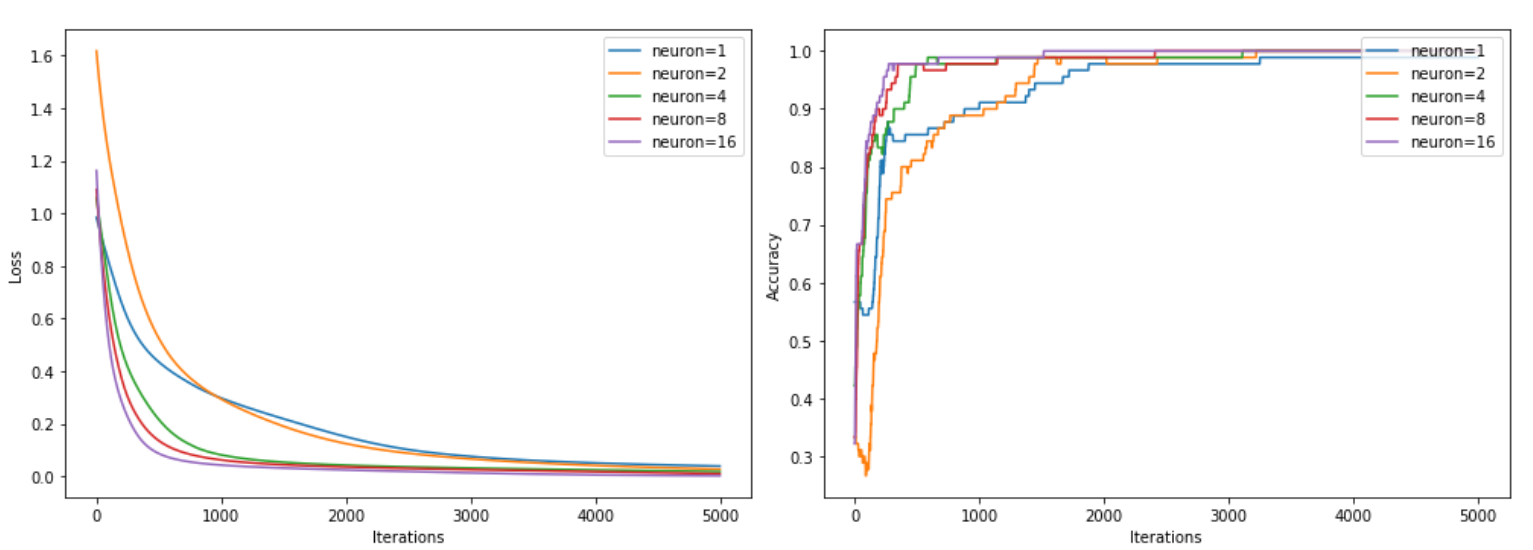
3，IRIS数据集有三类目标，每个类别有50个样本，每个样本有四维特征。，自行设计神经网络实现对这三个目标的识别，实验时每个类别随机选30个样本进行训练，另外20个样本用于测试。希望能通过设计不同的隐含层数、每层的节点数、不同的学习率、不同的激活函数等对实验结果进行讨论。

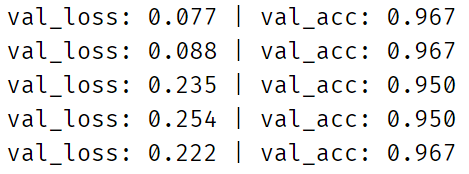
1. 隐含层数对网络性能的影响



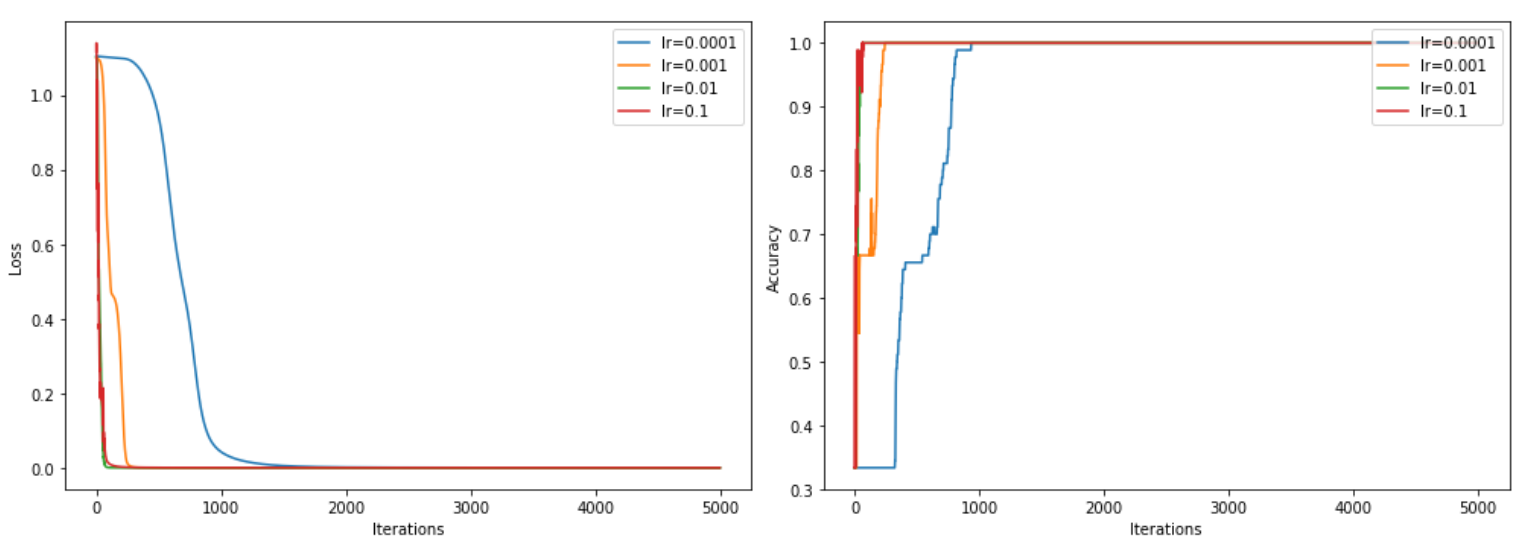


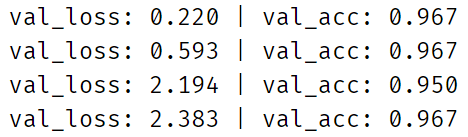
1. 隐藏层神经元个数对网络性能的影响



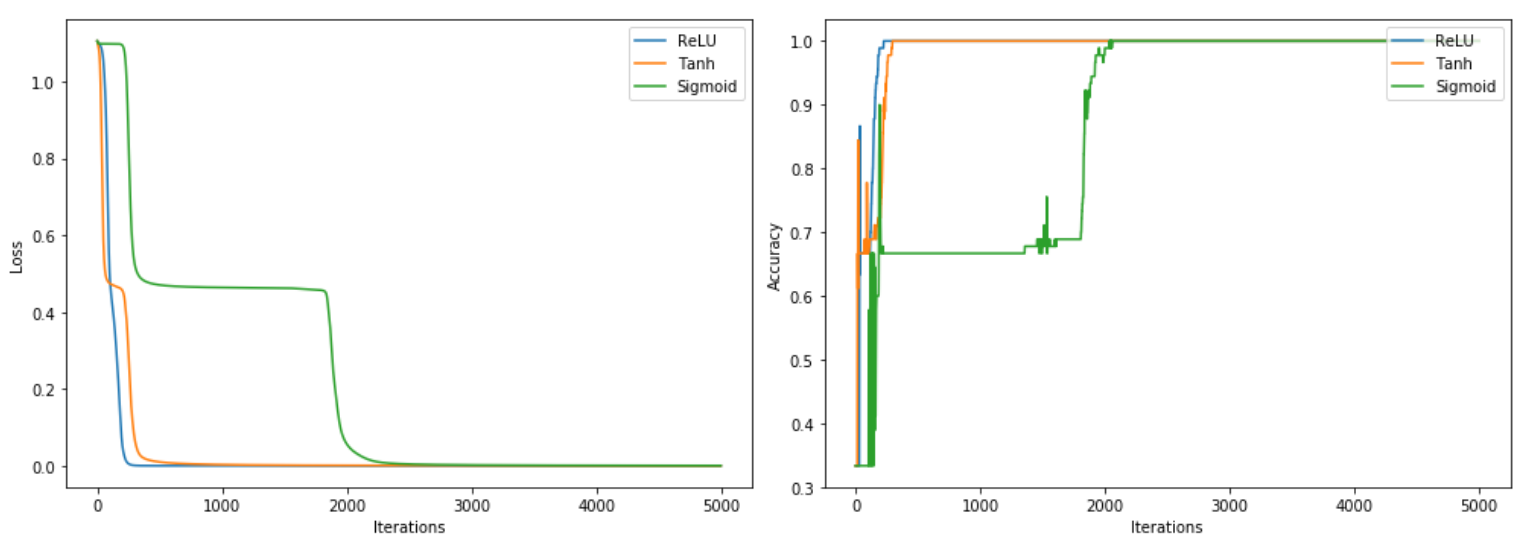


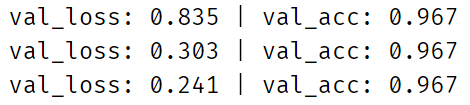
1. 学习率对网络性能的影响



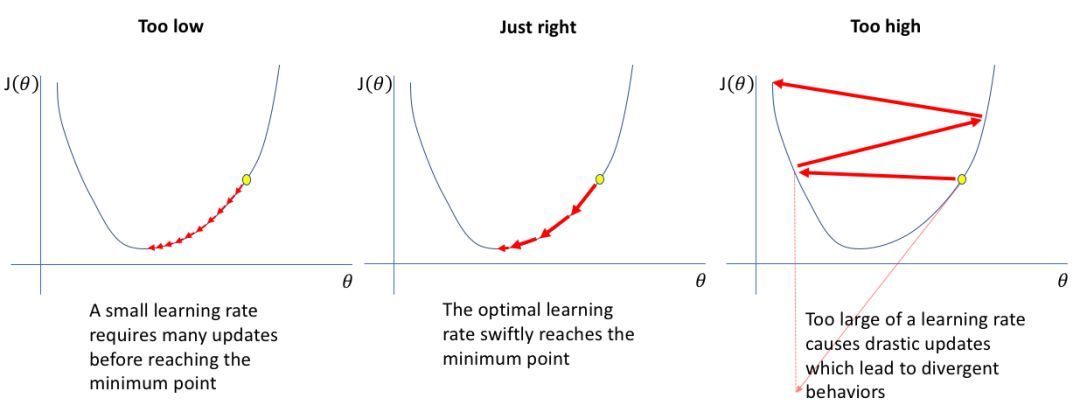


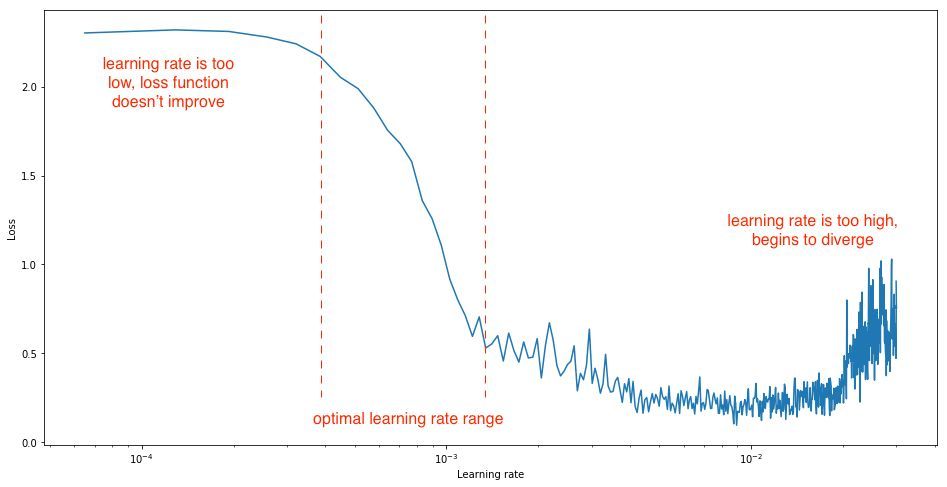
1. 激活函数对网络性能的影响





**实验总结与分析**

1. **隐含层数对网络性能的影响**
2. 从理论上讲，神经网络的隐藏层数越多，性能越好，拟合能力越强（这就解释了为什么深度学习超越传统机器学习算法甚至人类的原因），深的网络一般会比浅的网络效果好，如果要进一步地提升模型的准确率，最直接的方法就是把网络设计得越深越好，这样模型的准确率也就会越来越准确。但是，实际上更深的层数可能会带来过拟合的问题，同时也会增加训练难度，使模型难以收敛。
3. 我们可以从实验中看到，随着网络层数的不断增加，模型精度不断得到提升，而当网络层数增加到一定的数目以后，训练精度和测试精度下降，这说明当网络变得很深以后，深度网络就变得更加难以训练了。这是所谓的深度学习的难点之一：神经网络越深，优化越困难，过去的几十年间深度神经网络的优化的困难性也是阻碍它们成为主流的一个重要因素。为了能让神经网络变得更深，有一系列的研究人员提出自己的方法，其中最为经典的是何凯明团队提出的深度残差网络ResNet。
4. 为什么网络越深越难优化？原因应该是神经网络在反向传播过程中要不断地传播梯度，而当网络层数加深时，梯度在传播过程中会逐渐消失，导致无法对前面网络层的权重进行有效的调整。
5. 对于一般简单的数据集，一两层隐藏层通常就足够了。但对于涉及时间序列或计算机视觉的复杂数据集，则需要额外增加层数。单层神经网络只能用于表示线性分离函数，也就是非常简单的问题，比如分类问题中的两个类可以用一条直线整齐地分开。多个隐藏层可以用于拟合非线性函数。
6. **隐藏层神经元个数对网络性能的影响**
7. 从理论上讲，在隐藏层中使用太少的神经元将导致欠拟合，网络不能具有必要的学习能力和信息处理能力。相反，使用过多的神经元同样会导致一些问题。比如，大大增加网络结构的复杂性，可能会导致过拟合。当神经网络具有过多的节点（过多的信息处理能力）时，训练集中包含的有限信息量不足以训练隐藏层中的所有神经元，因此就会导致过拟合，而且会使网络的学习速度变得很慢。
8. 这也从实验中得到验证，可以看到，随着神经元个数的增加，网络性能有了相应提升，但是所带来的运算时间开销和内存开销是呈指数级上升的。因此神经元数量要看实际情况而定，不是盲目地认为隐藏层神经元个数越多越好，而应当认为“最合适的才是最好的”。
9. 需要注意的是，与在每一层中添加更多的神经元相比，添加隐藏层数将获得更大的性能提升。因此，不要在一个隐藏层中加入过多的神经元。总而言之，隐藏层神经元是最佳数量需要自己通过不断试验获得，建议从一个较小数值开始，如果欠拟合然后慢慢添加更多的层和神经元，如果过拟合就减小层数和神经元。此外，在实际过程中还可以考虑引入Batch Normalization，Dropout，正则化等缓解过拟合的方法。
10. **学习率对网络性能的影响**
11. 如果把学习率设置太低，训练会进展地很慢。然而，如果学习率被设置的太高，可能将会导致不停震荡甚至发散的后果。 
12. 从实验中可以看出，不同的情况下，学习率应选择不同的值（在不同情况下，较好的学习率是不同的）。这个没有一个固定的准则来告诉我们那个学习率最好，应当通过不停地实验，来根据不同的数据和模型来选择一个最恰当的学习率。
13. 对于学习率的选择问题，有一个找寻最优学习率的系统化方法：逐步提高每一次小批量（迭代）的学习率，记录每一次增量之后的损失。对于太小的学习率来说，损失函数可能减小，但是却是按照非常慢的速率减小。当进入了最优学习率区域，将会观察到在损失函数上一次非常大的下降。进一步增加学习速率会造成损失函数值“跳来跳去”甚至在最低点附近发散。因此最好的学习速率应当大致位于损失函数上最陡下降的区域。



1. 也有一些研究人员提出，可以在优化器上下功夫，比如使之自适应的选择学习率，典型的例子就是Adam。
2. 或者，可以根据验证集上的表现手动降低学习率，例如PyTorch提供torch.optim.lr\_scheduler.ReduceLROnPlateau函数：当某指标不再变化（下降或升高），调整学习率，这是非常实用的学习率调整策略。例如，当验证集的loss不再下降时，进行学习率调整；或者监测验证集的accuracy，当accuracy不再上升时，则调整学习率。
3. **激活函数对网络性能的影响**
4. 激活函数的作用是为了增加神经网络的非线性。如果没有激活函数，每层都相当于矩阵相乘，就算叠加了若干层之后，无非还是个矩阵相乘罢了的线性函数而已。
5. 从实验中可以看到，tanh的表现略逊于ReLU，且tanh和ReLU都比Sigmoid好。
6. Sigmoid通常被用为二分类问题输出层激活函数。由于梯度消失问题，有时要避免使用Sigmoid和tanh（对于深层网络，Sigmoid和tanh函数反向传播时，很容易就会出现梯度消失的情况，因为在Sigmoid和tanh接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失，从而无法完成深层网络的训练）。
7. ReLU函数是一个通用的激活函数，目前在大多数情况下使用，但是ReLU函数只能在隐藏层中使用。ReLU会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。一般情况下可以从ReLU函数开始，如果ReLU函数没有提供最优结果，再尝试其他激活函数，比如Leaky ReLU。