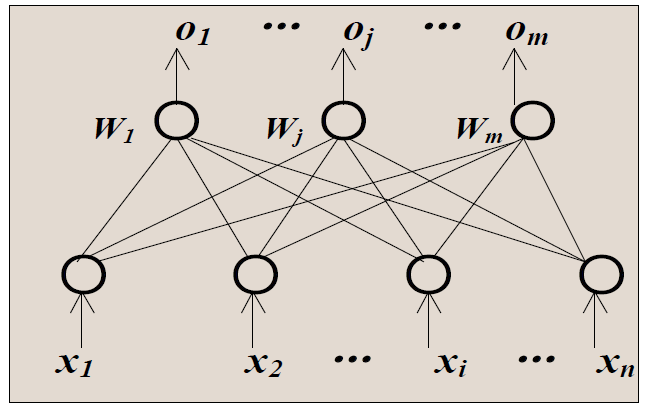
# 计算智能技术实验报告

计算智能技术的课程已经开展了九周，通过自己写程序对bp神经网络部分进行了实验，接下来详细介绍实验结果，关于BP神经网络概念等内容便不再重复介绍。

# 基本概念

## BP神经网络

BP(back propagation)神经网络是1986年由Rumelhart和McClelland为首的科学家提出的概念，是一种按照误差逆向传播算法训练的多层前馈神经网络，是应用最广泛的神经网络。



多层感知器模型

## 反向传播算法（Back Propagation）

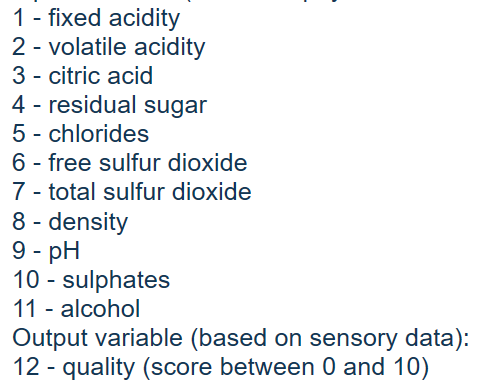
反向传播算法在对神经网络的权重进行更新时使用的非常广泛，当下的深度学习中基本上都采用该算法进行参数更新，只是在策略上有所不同。其背后的基本原理就是在网络的最后一层输出值和期望值之间进行差值的计算，然后通过链式法则，对每一层的权重求导，在当前权重的基础上，减去导数与学习率之间的乘积，如此反向计算一直到输入层，完成一次反向传播，对所有的权重值完成更新。

# UCI数据集

UCI数据集是由加州大学欧文分校提供的用于机器学习算法练习的数据集，目前一共有497个数据集，可以用来进行分类、回归、聚类等任务的练习，本实验中将会选择两个常用的数据集葡萄酒质量预测和车辆评估两个数据集进行BP神经网络的练习。

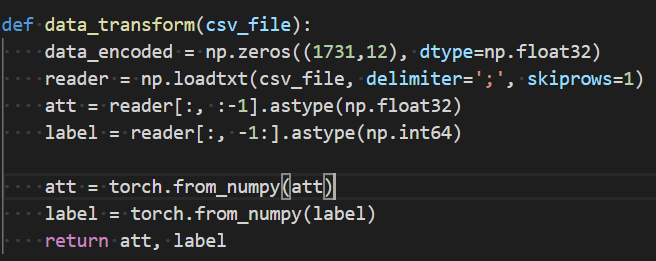
## Wine Quality Dataset

葡萄酒质量数据集包含红葡萄酒和白葡萄酒两种类别在内的一共4898组数据，本实验中选择使用白葡萄酒质量的数据集，包含11个属性11个类别在内的1599组数据，其中质量类别被划分为0-10一共11个级别，实验一的任务便是设计BP神经网络根据输入葡萄酒各个属性的值预测其被划分的类别。



葡萄酒质量属性类别及质量级别

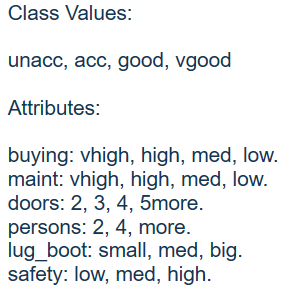
葡萄酒的十一个属性均是以连续浮点值给出，因此不需要对输入进行额外的编码处理，只需要将其从.csv文件中读取出来并转换为需要的张量即可，读取程序如图：



葡萄酒数据读取程序

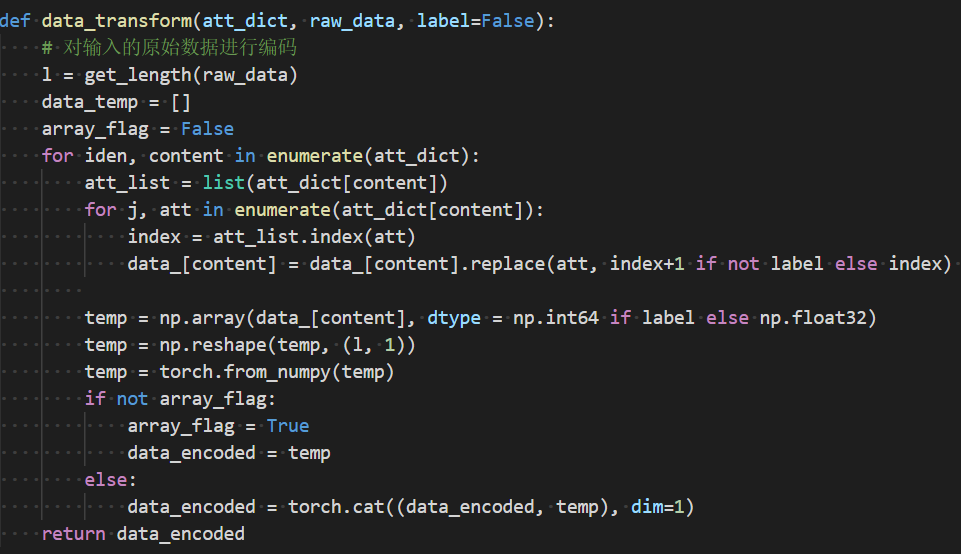
## Car Evaluation Dataset

车辆评估数据集包含价格、保险费、车门数量、乘客数量、行李空间、安全性在内的6个属性4个类别一共1728组数据。实验二的任务便是根据输入车辆的属性对车辆的类别进行评估。数据集的属性及类别信息如下图。



车辆属性及类别信息

从上图可以看到，属性值和类别值均是以抽象概念给出，无法直接用神经网络进行计算，因此需要首先对输入和输出值进行编码，将其转换为神经网络可以计算的形式。数据转换的程序如下图所示：

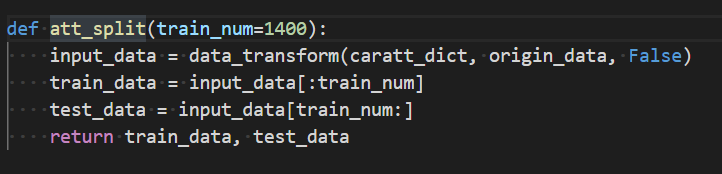


车辆数据转换程序

为了简化编码的过程，实验中并没有采用one-hot的编码格式，而是直接使用1、2、3等数字对不同的属性进行编码，根据编程的需要将属性值转换为浮点值，类别值转化为整型。

## 数据集划分

为了验证BP神经网络的效果，需要将数据集划分为训练集和测试集，如果实验数据充分，还可以再添加一个验证集，但是由于本实验中使用的数据集数量均不足2000，因此只划分了训练集和测试集，其中用于训练的数据包含1400组，剩余的数据用于测试。数据集的划分程序如下图：



数据集划分程序

# 实验细节

## 编程框架

为了能够高效地进行BP神经网络的编写，本实验选择了Python作为编程语言，并使用PyTorch作为神经网络编程的工具。使用PyTorch的好处在于，只需要将前向传播的网络搭建起来即可，反向传播过程中的梯度计算和权值更新可以自动完成，不需要人为再进行计算。

## 实验任务

本实验的任务是在UCI数据集上研究改进BP神经网络的方法，如何提高神经网络的精度。根据上述数据集的属性结构，本实验将神经网络的任务设计为多分类，根据输入的属性对其标签进行分类。

实验的主要内容包括：

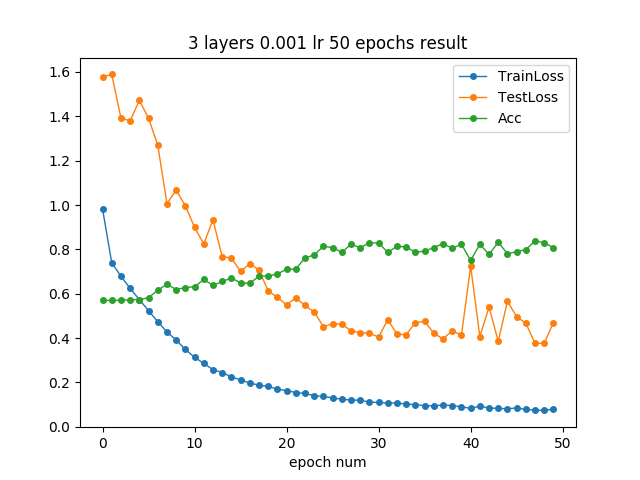
1. 分析BP网络层数对模型精度的影响。
2. 隐层神经元个数对模型精度的影响。
3. 分析学习率对模型精度的影响。
4. 分析训练次数对模型精度的影响。
5. 分析使用Dropout策略对模型精度的影响。

接下来依次给出在两组数据集上不同变量下的实验可视化结果。

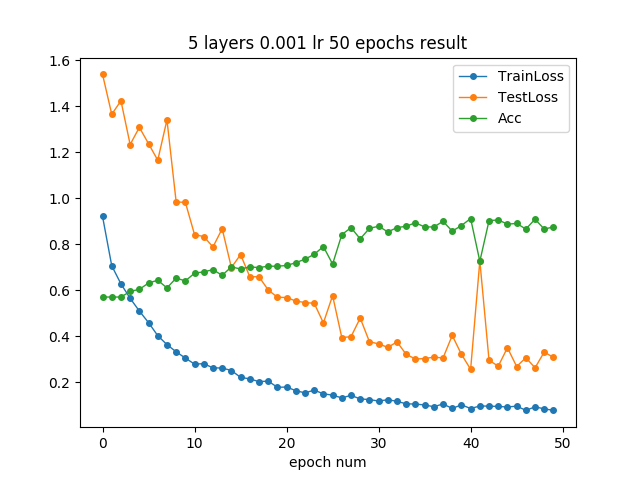
## 车辆数据集实验结果

## 网络层数的影响

设置学习率为0.001，在全部数据集上训练的次数为50次，隐层的神经元均为16，设置网络的层数分别为3层和5层，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



网络层数等于3实验结果

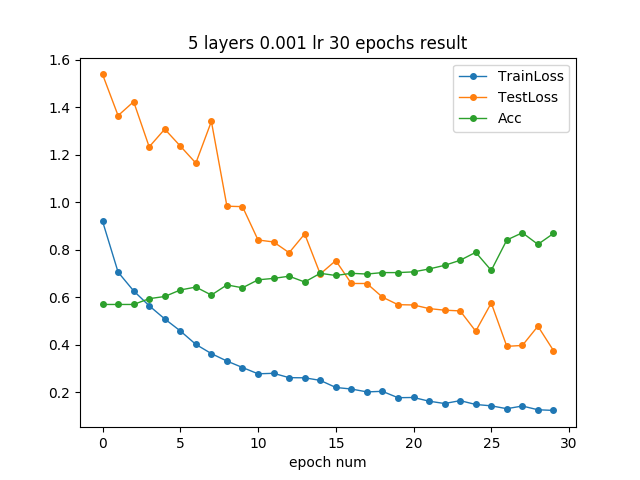


网络层数等于5实验结果

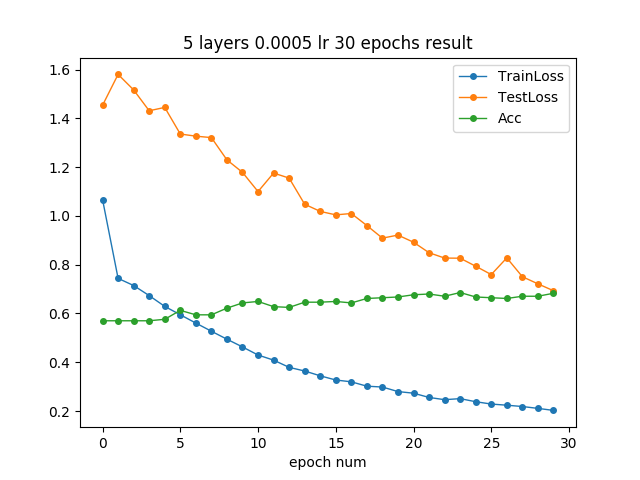
其中网络层数等于3层的模型，在训练集上的损失函数最小等于0.0739，在测试集上的损失函数最小为0.3757，在测试集上的准确率最高为0.739；网络层数等于5层的模型在训练集上的损失函数最小等于0.077，测试集上的损失函数最小为0.2557，测试集上的准确率最高为0.9116。通过增加网络层数的数量，有效提高了分类的准确率。

## 学习率的影响

在全部数据集上训练的次数为30次，隐层的神经元均为16，设置网络的层数为5层，设置学习率分别为0.001和0.0005，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



学习率等于0.001实验结果

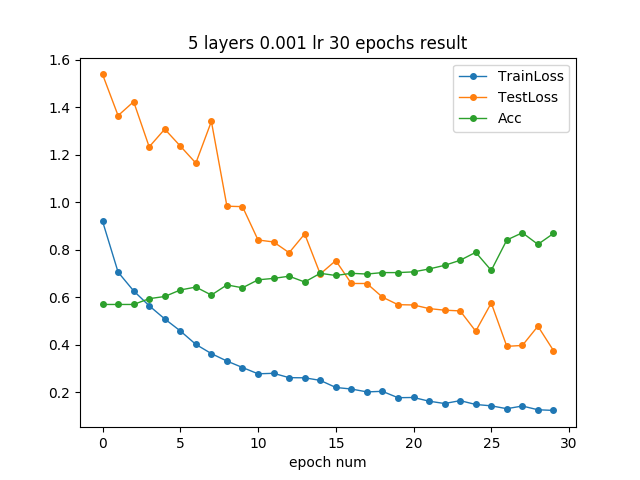


学习率等于0.0005实验结果

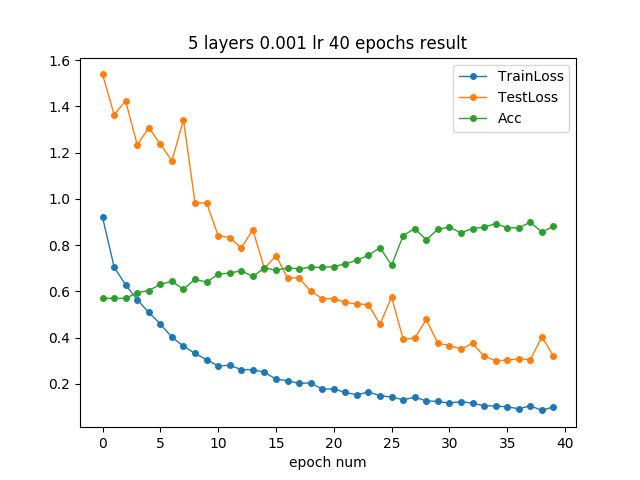
其中学习率等于0.001的模型，在训练集上的损失函数最小等于0.124，在测试集上的损失函数最小为0.376，在测试集上的准确率最高为0.872；学习率等于0.0005的模型在训练集上的损失函数最小等于0.2025，测试集上的损失函数最小为0.693，测试集上的准确率最高为0.68。通过增大网络模型的学习率，能够有效提高模型的准确率，但是从实验的图中可以看到，更大的学习率再训练的过程中对应着模型效果的浮动也会加剧，不如小学习率对应的训练过程平滑稳定。

## 训练次数的影响

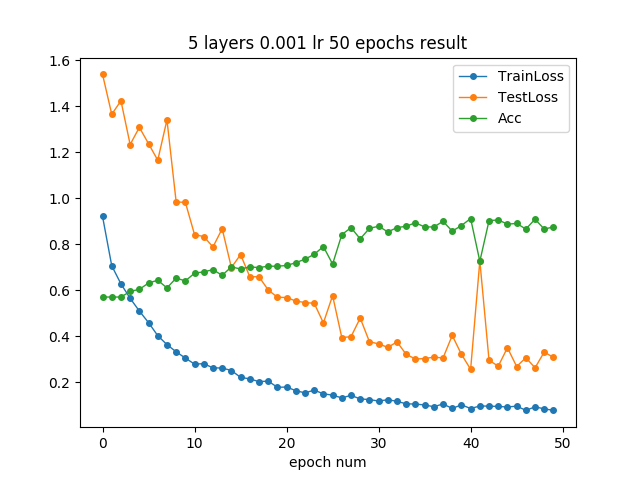
设置隐层的神经元个数均为16，网络的层数为5层，设置学习率分别为0.001，训练的次数分别为30次、40次、50次，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



训练次数30次实验结果



训练次数40次实验结果



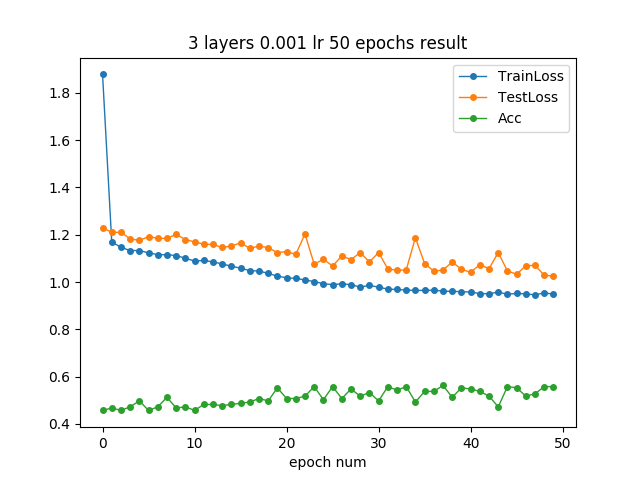
训练次数50次实验结果

其中训练次数等于30次的模型，在训练集上的损失函数最小等于0.124，在测试集上的损失函数最小为0.376，在测试集上的准确率最高为0.872；训练次数等于40次的模型在训练集上的损失函数最小等于0.085，测试集上的损失函数最小为0.30，测试集上的准确率最高为0.899；训练次数等于50次的模型在训练集上的损失函数最小等于0.077，测试集上的损失函数最小为0.2557，测试集上的准确率最高为0.9116。通过增加训练的次数，能够有效提高模型的准确率，但是从实验的结果来看，随着训练次数的增加，模型准确率的提高幅度逐渐减小，最终实现收敛。

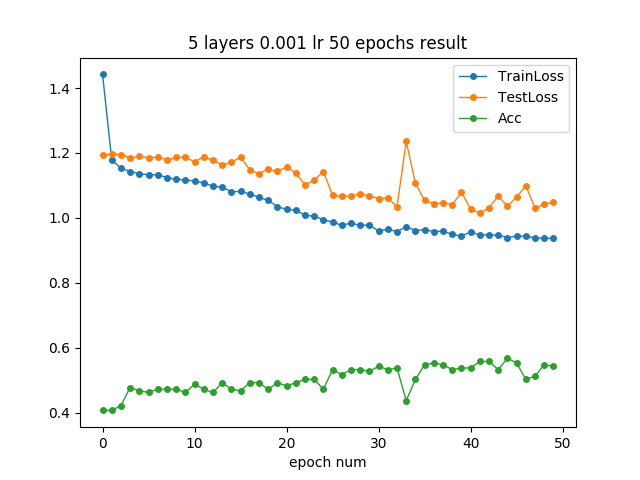
## 葡萄酒质量数据集实验结果

## 网络层数的影响

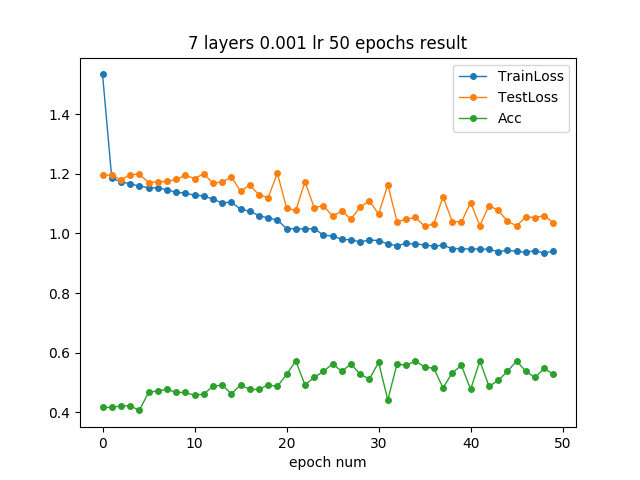
设置学习率为0.001，在全部数据集上训练的次数为50次，隐层的神经元均为16，设置网络的层数分别为3层、5层和7层，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



网络层数等于3的实验结果



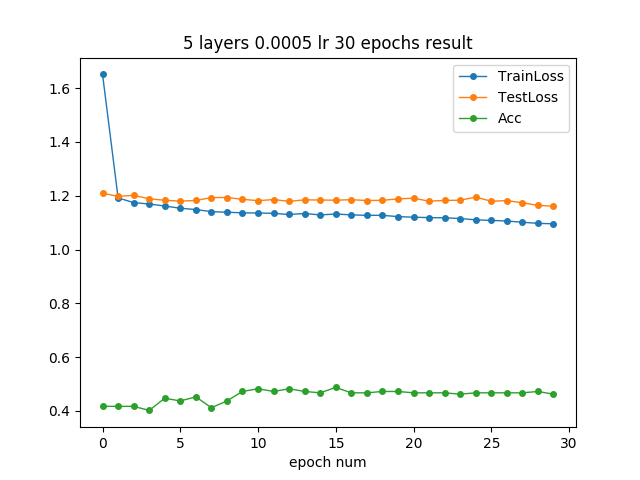
网络层数等于5的实验结果



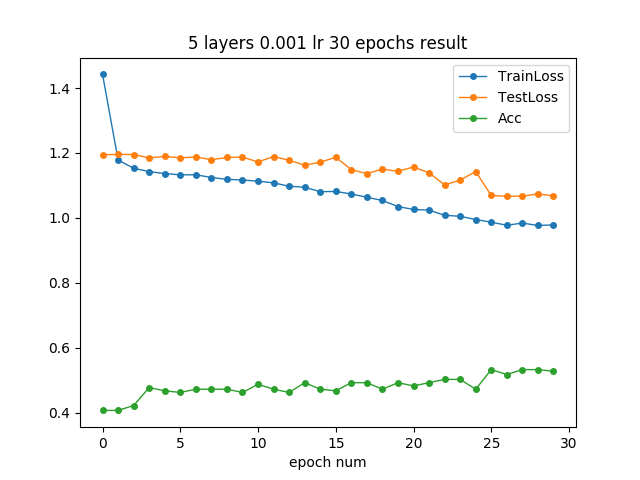
其中网络层数等于3层的模型，在训练集上的损失函数最小等于0.9457，在测试集上的损失函数最小为1.026，在测试集上的准确率最高为0.563；网络层数等于5层的模型在训练集上的损失函数最小等于0.9366，测试集上的损失函数最小为1.015，测试集上的准确率最高为0.568；，网络层数等于7层的模型，在训练集上的损失函数最小等于0.933，在测试集上的损失函数最小为1.024，在测试集上的准确率最高为0.573通过增加网络层数的数量，有效提高了分类的准确率，降低了在测试集上的损失函数值。

## 学习率的影响

在全部数据集上训练的次数为30次，隐层的神经元均为16，设置网络的层数为5层，设置学习率分别为0.001和0.0005，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



学习率等于0.0005实验结果

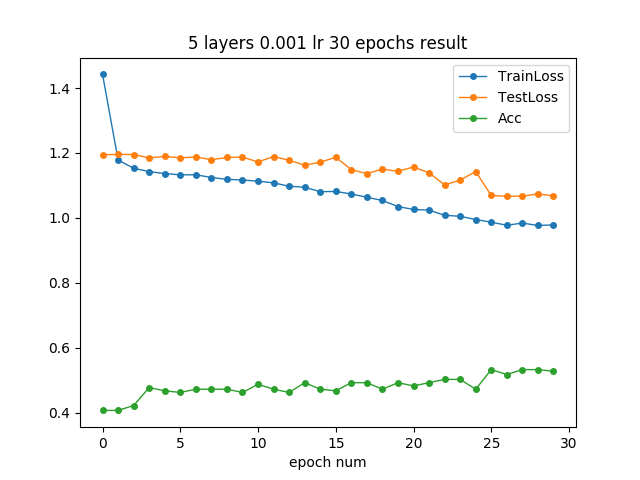


学习率等于0.001实验结果

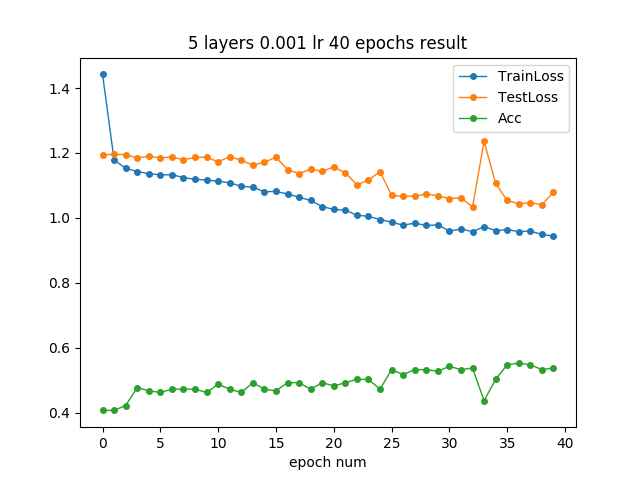
其中学习率等于0.0005的模型，在训练集上的损失函数最小等于1.095，在测试集上的损失函数最小为1.161，在测试集上的准确率最高为0.487；学习率等于0.001的模型在训练集上的损失函数最小等于0.976，测试集上的损失函数最小为1.066，测试集上的准确率最高为0.5327。通过增大学习率，同样能够有效提高模型分类的准确率，较小的学习率对应的训练过程更加平滑。

## 训练次数的影响

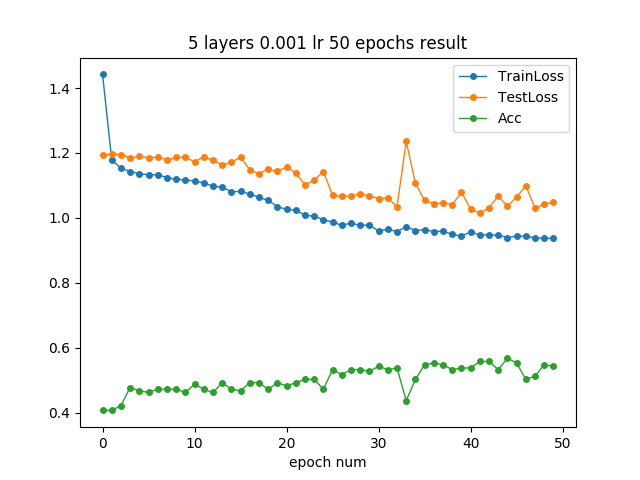
在全部数据集上训练的学习率等于0.001，隐层的神经元均为16，设置网络的层数为5层，分别训练30次、40次、50次，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



训练次数30次的实验结果



训练次数40次的实验结果

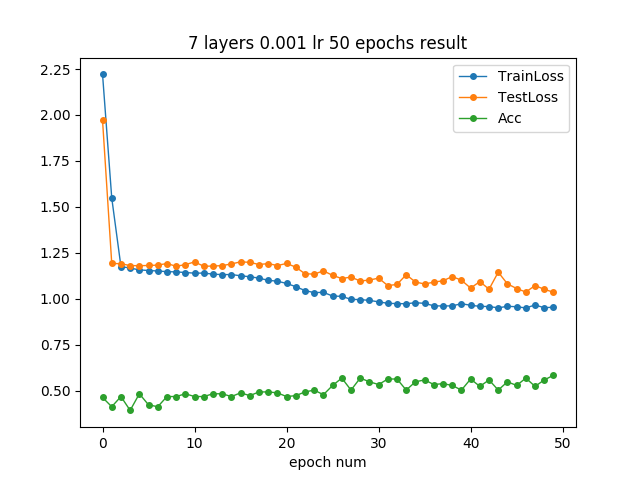


训练次数50次实验结果

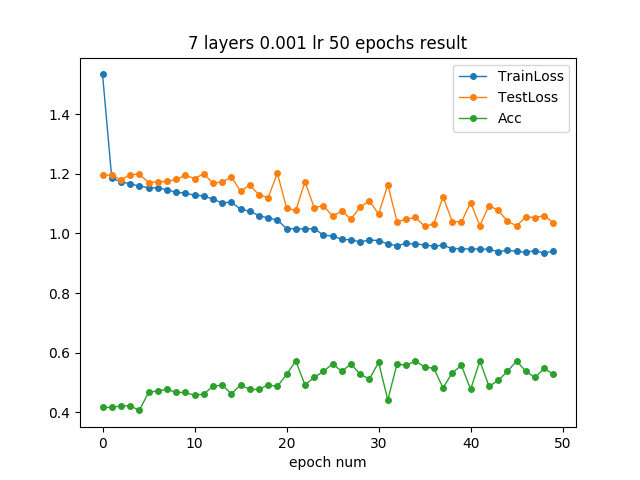
其中训练次数等于30次的模型，在训练集上的损失函数最小等于0.976上的损失函数最小为1.066,在测试集上的准确率最高为0.533；训练次数等于40次的模型在训练集上的损失函数最小等于0.944，测试集上的损失函数最小为1.034，测试集上的准确率最高为0.553；训练次数等于50次的模型在训练集上的损失函数最小等于0.936，测试集上的损失函数最小为1.015，测试集上的准确率最高为0.568。同样的，通过增加训练的次数，能够有效提高模型的准确率，降低在测试集上的损失函数值，随着训练次数的增加，模型准确率的提高幅度逐渐减小，逐渐趋于平缓。

## 隐层神经元的影响

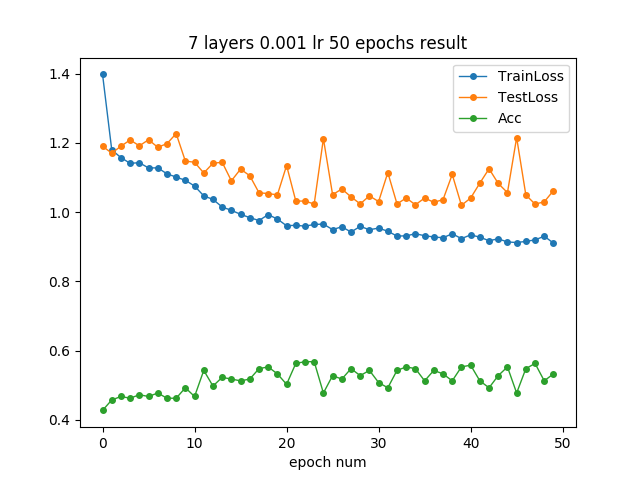
在全部数据集上训练的学习率等于0.001，隐层的神经元均为16，网络的层数为7层，训练50次，分别设置隐层的神经元的个数等于8、16、32，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



神经元个数等于8实验结果



神经元个数等于16实验结果



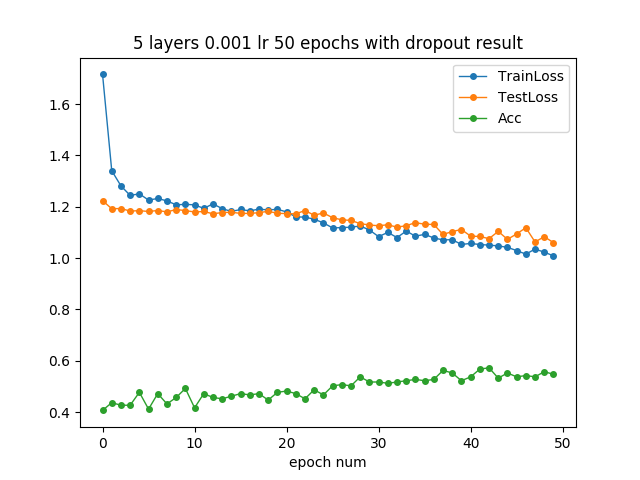
神经元个数等于32实验结果

其中神经元个数等于8的模型，在训练集上的损失函数最小等于0.949上的损失函数最小为1.035,在测试集上的准确率最高为0.583；神经元个数等于16的模型在训练集上的损失函数最小等于0.933，测试集上的损失函数最小为1.024，测试集上的准确率最高为0.573；神经元个数等于32的模型在训练集上的损失函数最小等于0.91，测试集上的损失函数最小为1.02，测试集上的准确率最高为0.568。从上面的可视化结果中，可以看出，随着神经元个数的增加，模型在训练的过程中损失函数的波动逐渐变大。

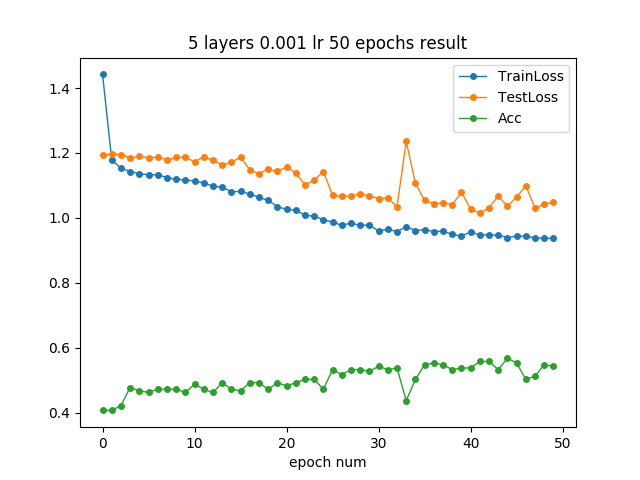
此外，随着神经元个数的增加，在训练集上的损失函数逐渐减小，但是在测试集上的表现却并没有随之提升，反而出现了一定程度的降低。对此的解释是我们使用的数据集数量有限，输入的变量较少，分类模型并不需要过于复杂的网络结构，因此随着隐层神经元个数的增加，模型的非线性程度增加，因此出现了过拟合的情况。

## 添加Dropout的影响

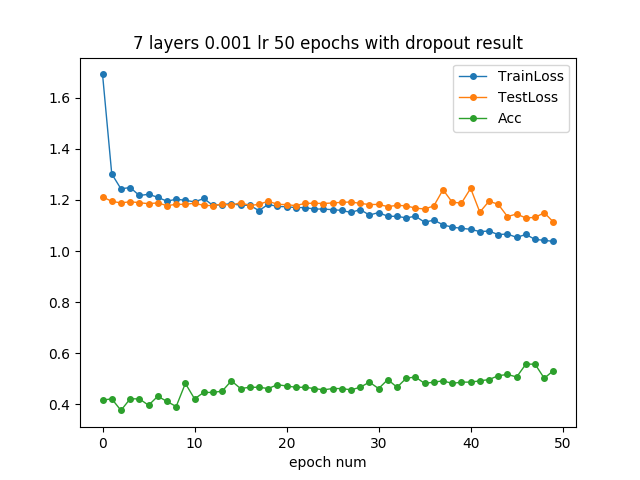
在全部数据集上训练的学习率等于0.001，隐层的神经元均为16，设置网络的层数为5层和7层，训练50次，分别使用或不适用Dropout，使用交叉熵作为分类任务的损失函数，查看在训练集和测试集上的损失函数以及在测试集上分类的准确率，实验结果如下：



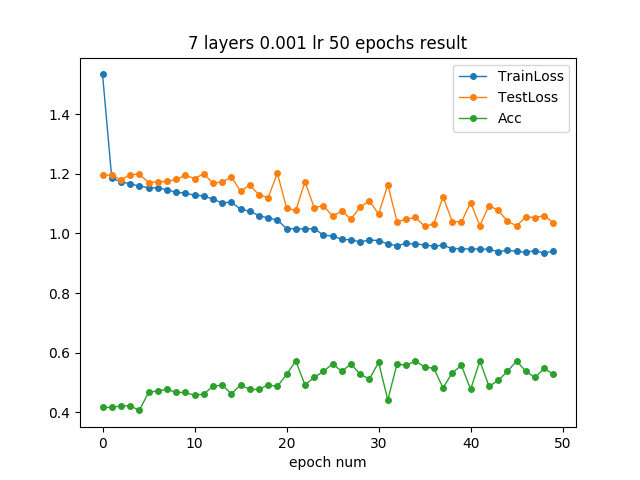
5层网络使用dropout实验结果



5层网络不使用dropout实验结果



7层网络使用dropout实验结果



7层网络不使用dropout实验结果

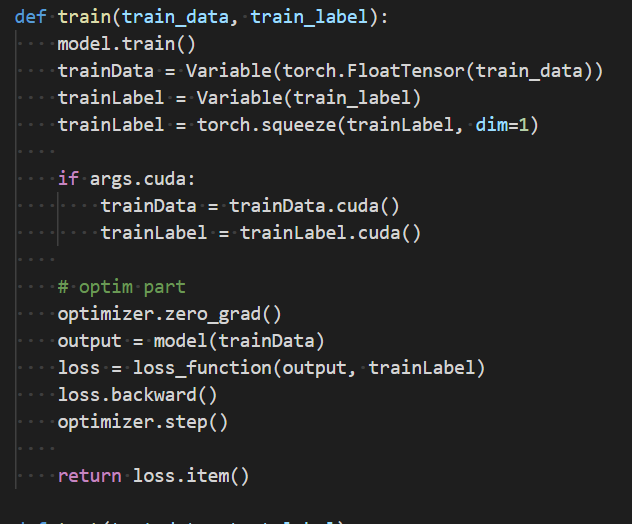
其中网络层数等于5的模型，在不使用Dropout层时，在训练集上的损失函数最小等于0.936，在测试集上的损失函数最小为1.014,在测试集上的准确率最高为0.568；使用Dropout层时，在训练集上的损失函数最小等于1.008，测试集上的损失函数最小为1.06，测试集上的准确率最高为0.573；网络层数等于7的模型，在不使用Dropout时，训练集上的损失函数最小等于0.933，测试集上的损失函数最小为1.024，测试集上的准确率最高为0.573；使用Dropout时，在训练集上的损失函数最小等于1.038，测试集上的损失函数最小为1.114，测试集上的准确率最高为0.558。

通过上图可以看出，在添加了Dropout层之后，模型在测试集上的准确率会有一定的降低，但是训练的过程会相对更加平缓。这样的结果也符合添加Dropout后的效果，通过将神经元随机失活，能够提高模型的泛化能力，避免模型的过拟合。

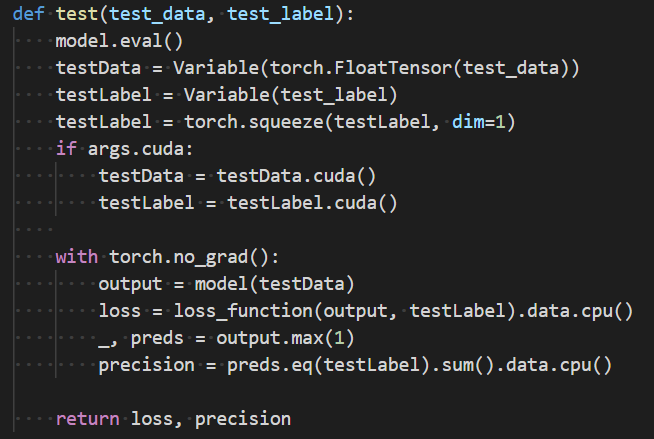
## 实验总结

根据在两个UCI数据集上的实验，可以发现，BP神经网络的效果受多个变量的影响，其中包括网络的层数、学习率、训练的次数、神经元的个数、是否使用Dropout层等，此外，训练集的大小也直接决定了模型在测试集上的效果。为了尽可能提升模型的效果，根据上述实验结果来总结，应该首先选择一个较小的学习率，如果训练集数据不够充分，先使用较少的神经元和网络层数，在控制变量的条件下，增加某一个变量的值，查看模型的结果，再做下一步的调整。网络的层数和神经元的个数越多，网络的非线性程度越大，拟合能力越强，需要根据输入数据的复杂程度适当调整网络的层数以及神经元的个数，避免出现过拟合的情况。当模型在训练的过程中损失函数震荡幅度较大，可以考虑添加Dropout层来稳定训练的过程。

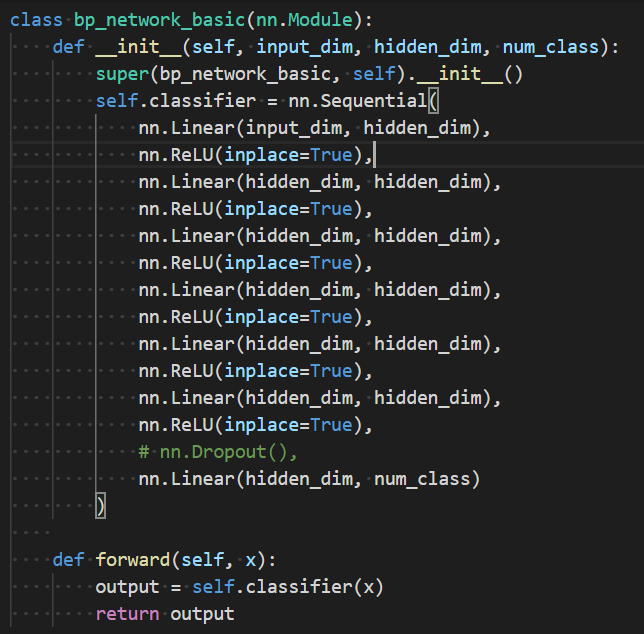
# 部分实验代码截图



训练代码截图



测试代码截图



BP网络代码截图