1．现假设样本来自三个类，某次训练中的一个batch包含3个训练样本$\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{x_3}$，分别来自第1，2，3类：

(a) 试推导采用单热向量编码时该batch交叉熵损失函数表达式。（提示：设该 batch对应网络输出为 $\boldsymbol{y_1},\boldsymbol{y_2},\boldsymbol{y_3}$）

(b) 如果网络输出为$\boldsymbol{y_1} = (0.65, 0.43, 0.11), \boldsymbol{y_2} = (0.05, 0.51, 0.18), \boldsymbol{y_3} = (0.33, 0.21, 0.72)$， 计算交叉熵损失函数值。

(a)解：设该 batch对应网络输出为 $\boldsymbol{Y_{out}} = (\boldsymbol{y_1} \quad \boldsymbol{y_2} \quad \boldsymbol{y_3})$，其中$\boldsymbol{y_i}$为三元素的列向量。对应的真值经过独热编码后可表示为

$$\boldsymbol{Y} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (\boldsymbol{y_1^*} \quad \boldsymbol{y_2^*} \quad \boldsymbol{y_3^*})。$$

对网络输出$\boldsymbol{Y_{out}}$，用softmax函数进行处理，将网络输出转化为:

$$P(j|y_i) = \frac{e^{y_{ij}}}{\sum_{k=1}^{3} e^{y_{ik}}}$$

∴设 处理后的网络输出$\boldsymbol{\hat{Y}} = (\widehat{\boldsymbol{y_1}} \quad \widehat{\boldsymbol{y_2}} \quad \widehat{\boldsymbol{y_3}})$，其元素$\hat{y}_{ij} = \frac{e^{y_{ij}}}{\Sigma_{k=1}^{3} e^{y_{ik}}}$

$$\because \ J(\boldsymbol{Z}, \boldsymbol{\hat{Z}}) = \sum_{i=1}^{n} H(\boldsymbol{z}_i, \boldsymbol{\hat{z}}_i),\ H(\boldsymbol{q}, \boldsymbol{p}) = -\sum_{j=1}^{c} q_j ln\, p_j$$

$$\therefore H(\boldsymbol{y_1^*}, \widehat{\boldsymbol{y_1}}) = -\sum_{j=1}^{3} y_{1j}^* ln\widehat{y_{1j}} = -ln\widehat{y_{11}}$$

$$\therefore H(\boldsymbol{y_2^*}, \widehat{\boldsymbol{y_2}}) = -\sum_{j=1}^{3} y_{2j}^* ln\widehat{y_{2j}} = -\ ln\widehat{y_{22}}$$

$$\therefore H(\boldsymbol{y_3^*}, \widehat{\boldsymbol{y_3}}) = -\sum_{j=1}^{3} y_{3j}^* ln\widehat{y_{3j}} = -ln\widehat{y_{33}}$$

$$\therefore J(\boldsymbol{Y}, \boldsymbol{\hat{Y}}) = -\ln\left(\frac{e^{y_{11}}}{\Sigma_{k=1}^{3} e^{y_{1k}}}\right) - \ln\left(\frac{e^{y_{22}}}{\Sigma_{k=1}^{3} e^{y_{2k}}}\right) - \ln\left(\frac{e^{y_{33}}}{\Sigma_{k=1}^{3} e^{y_{3k}}}\right)$$

(b)解：将$\boldsymbol{y_1} = (0.65, 0.43, 0.11), \boldsymbol{y_2} = (0.05, 0.51, 0.18), \boldsymbol{y_3} = (0.33, 0.21, 0.72)$代入上式中可得：

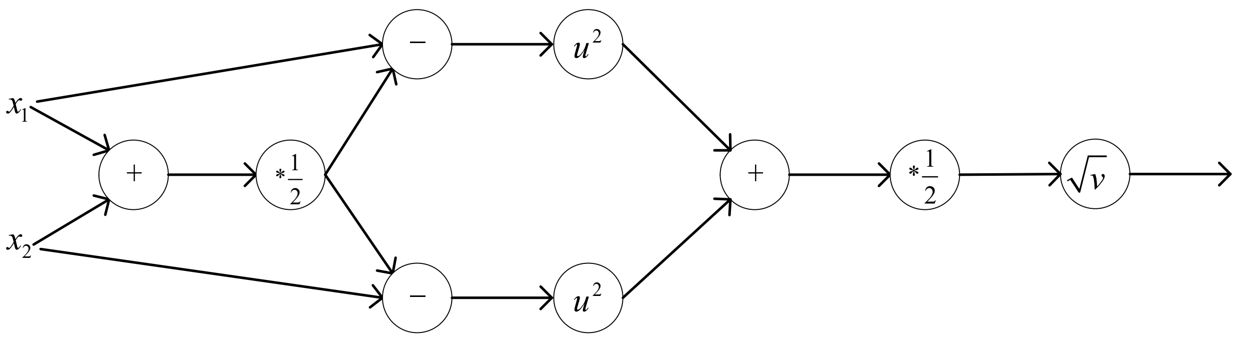
$$J(\boldsymbol{Y}, \boldsymbol{\hat{Y}}) = 2.547$$

2. 假设输入有 2 个样本x_1, x_2:

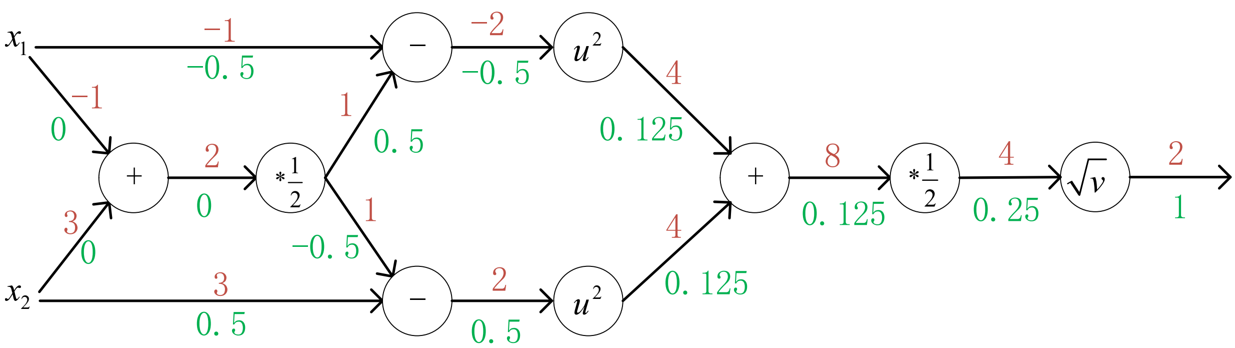
(1)请画计算x_1, x_2标准差的计算图;

(2) 标出当$x_1 = -1$, $x_2 = 3$时输出对图中每个节点输入变量的梯度值, 并求出x_1, x_2总的梯度值

（1）解：标准差表达式：$\sigma = \sqrt{\frac{(x_1-\bar{x})^2+(x_2-\bar{x})^2}{2}}$, $\bar{x} = \frac{1}{2}(x_1 + x_2)$



（2）梯度值如下绿色数字，前向计算值如下红色。



x_1的梯度值： −0.5； x_2的梯度值：+0.5

实践题：

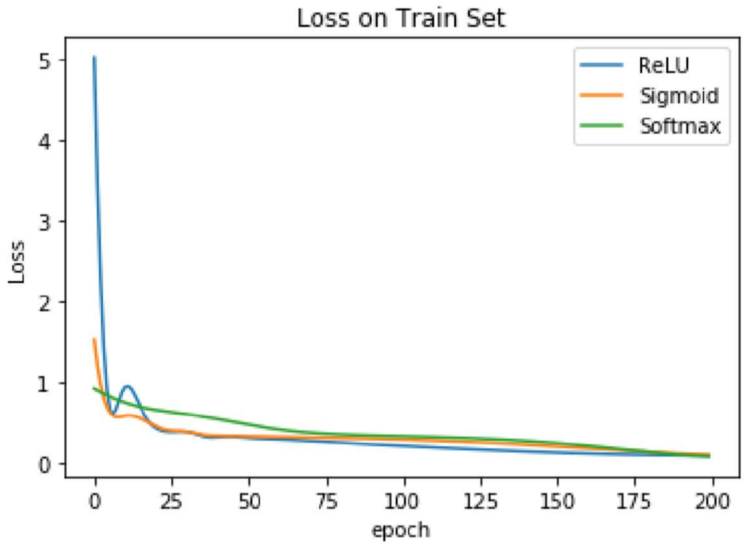
(1)实现一个三层神经网络，并使用 iris 数据集前 80%训练、后 20%测试，要求测试 错误率小于 5%，分析至少三种非线性激活函数的影响。

Iris.ipynb 有详细的文件介绍。

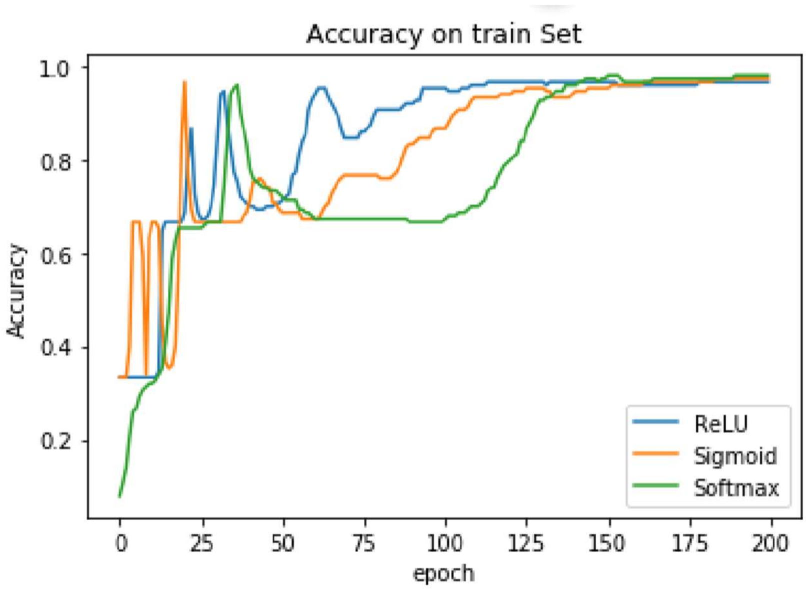
网络结构采用了最简单的：**输入层----全连接隐藏层(40 个神经元)——输出层**，采用均方误差作为损失函数。隐藏层的激活函数对比了 ReLU，Sigmoid，Softmax 三种

结果如下：

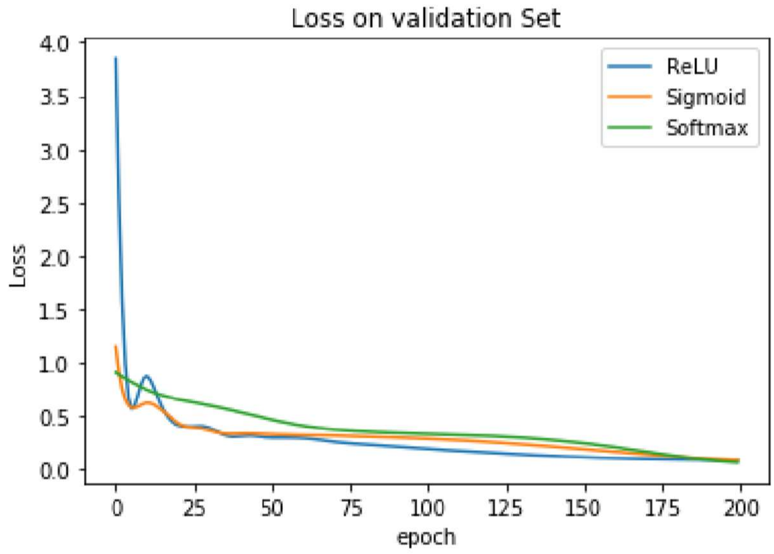
训练过程中在训练集的损失降低情况：



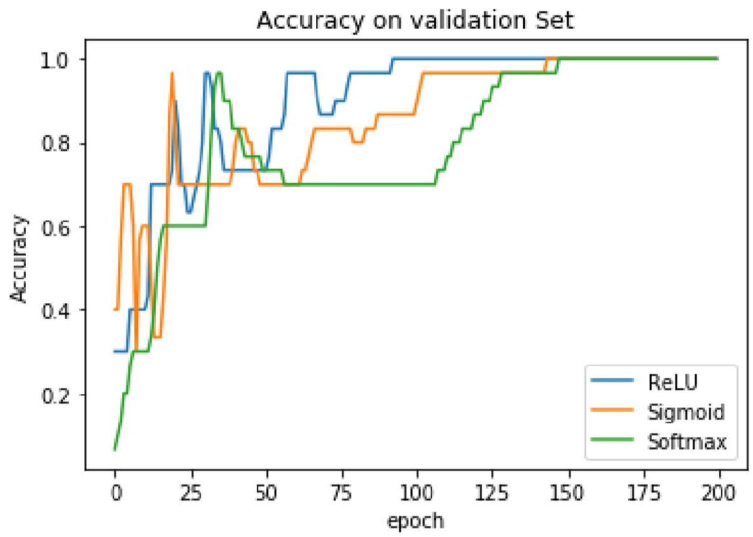
训练过程中在训练集的分类准确性比较：



训练过程中在测试集的损失降低情况：



训练过程中在测试集的分类准确性比较：




```
ReLU:test_accuracy: 1.0
Softmax:test_accuracy: 1.0
Sigmoid:test_accuracy: 1.0
```

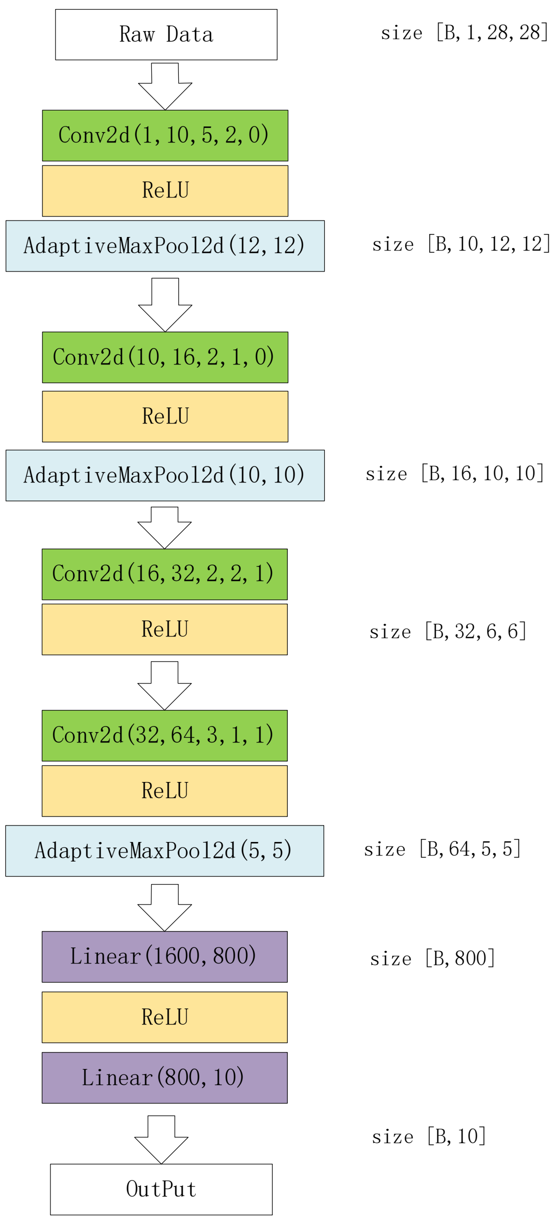
最终测试结果：

分析比较：从训练过程中的准确性变化曲线可以看出，三次实验中途均有较高准确性，但分类正确率马上降低，最后达到较高准确性并稳定，这是因为中途达到了局部最优解的原因，Adam 优化使网络跳出局部最优，找到更优解。三种不同的激活函数都能达到较好的实验结果，但可以看出，使用 ReLU 的收敛速度更快，而 Softmax 最慢。

(2)设计并实现一个深度学习网络结构，能够在 MNIST 数据集上（前 6 万个训练， 后 1 万个测试)获得至少 99%的测试精度

MNIST.ipynb 有详细的文件介绍。

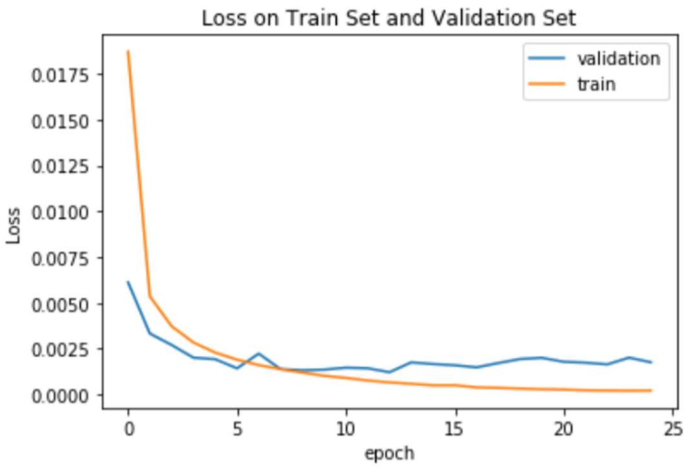
网络结构分析与介绍：



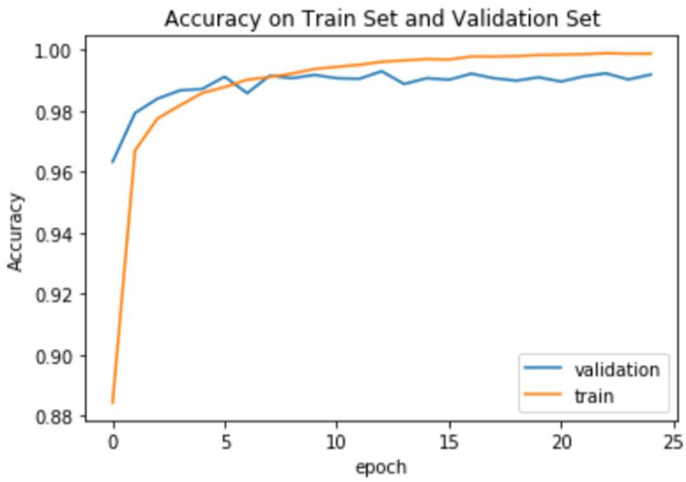
其中 Conv2d 代表二维卷积，ReLU 代表 ReLU 激活函数，Sigmoid 同理。Linear 表示全连接层，AdaptiveMaxPool2d 表示二维的自适应最大池化，每层处理后的数据尺寸在右侧标出。

结果如下：

训练过程中，在训练集与测试集的损失降低情况如图：



训练过程中，在训练集与测试集的分类准确性如图：



test_accuracy 0.9929

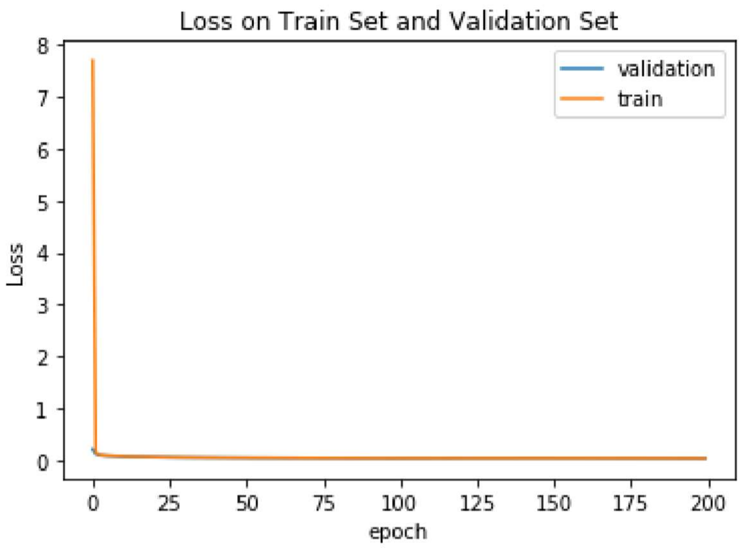
最终模型的分类准确性（在测试集上）

从图中可以看出存在一定的过拟合现象，因为测试集的损失在后期开始波动，而训练集的损失还在持续降低。之前我有另一个网络结构，很好地解决了过拟合情况，但是其精确度最高只能达到 0.989，离要求的 0.99 仅有一点点距离。最终还是选择了略有过拟合现象但是精度更高的版本。

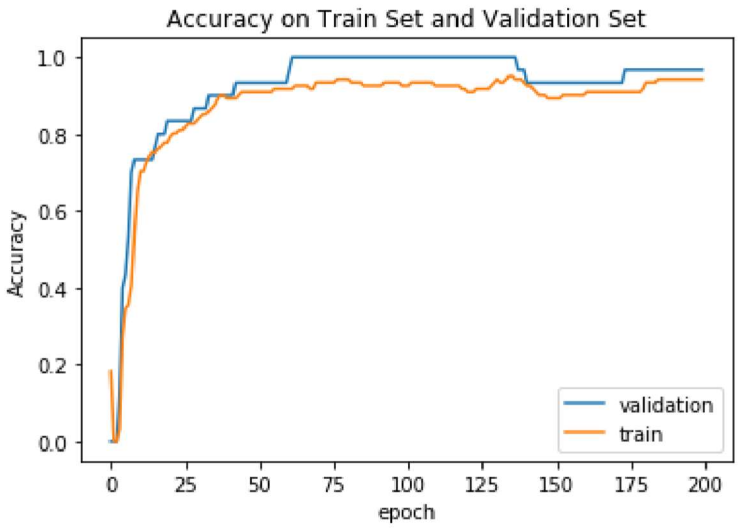
仅使用 numpy 实现三层神经网络 BP 训练算法(输入 d 维向量，中间 h 个隐含神经元，输出 c>1 类单热向量编码，隐含层使用 sigmoid 激活函数，输入输出层使用 线性激活函数)，损失函数用均方误差或者交叉熵。在 iris 数据集上对 1) 中实现的算法测试，并与实践题一的结果进行比较

network with numpy.ipynb 有详细的文件介绍，采用了均方损失误差作为损失函数。

训练过程中，在训练集与测试集的损失降低情况如图：



训练过程中，在训练集与测试集的分类准确性情况如图：



最终模型的分类准确性：

```
test_accuracy 0.9666666666666667
```

程序实现了在创建模型时，可以自己设置输入维度、输出维度、隐藏层神经元数目。

我在训练时采用了与第一题相同的网络结构：输入层----全连接隐藏层(40个神经元)——输出层，采用均方误差作为损失函数。隐藏层的激活函数使用了 Sigmoid

与实践题1进行比较可以发现，用numpy实现的神经网络最终在测试集上的表现略逊于pytorch实现的。据我分析，这是因为pytorch版本中我使用了Adam优化，而numpy版本使用的是梯度下降算法进行优化。可能是导致最终模型性能差异的原因。但是numpy版本的训练过程中曲线更加平滑。