模式识别机器学习 报告二

综述

本次实验的目标是训练 RNN 网络作为加法器。本质是一个深度序列模型，利用神经网络模型判断输出的出现概率。

利用源代码中的普通 RNN 模型已经可以达到1000位准确无误的计算准确度，结果非常优秀。但是这是基于用相同位数的训练集进行训练得到的结果。如果想让机器想人一样真正掌握加法，其实应该只需要**用低位数进行训练**，原则上掌握加法的网络应该有可以计算任意位数的能力。

本实验将在普通RNN，lstm，和gru网络探究机器的计算学习能力以及**低位训练对高位计算的泛化能力**。

任务一

深度序列模型具有的三个模块分别是：**嵌入层**，**特征层**，**输出层**。

在**嵌入层**中将输入的包含0-9数字的向量映射到更高维空间，可能把一些其他特征给放大了，或者把笼统的特征给分开了，在本实验将10维向量映射到了32维的空间，再将两个向量串联起来。

```
num1 = self.embed_layer(num1)
num2 = self.embed_layer(num2)
input = torch.cat((num1, num2), 2)
```

在特征层使用普通的循环神经网络，输入维度64，输出维度64，堆叠两层。此处注意在 RNN 的初始化時一定要填上batch_first=true。

```
r_out, (h_n) = self.rnn(input, None)
```

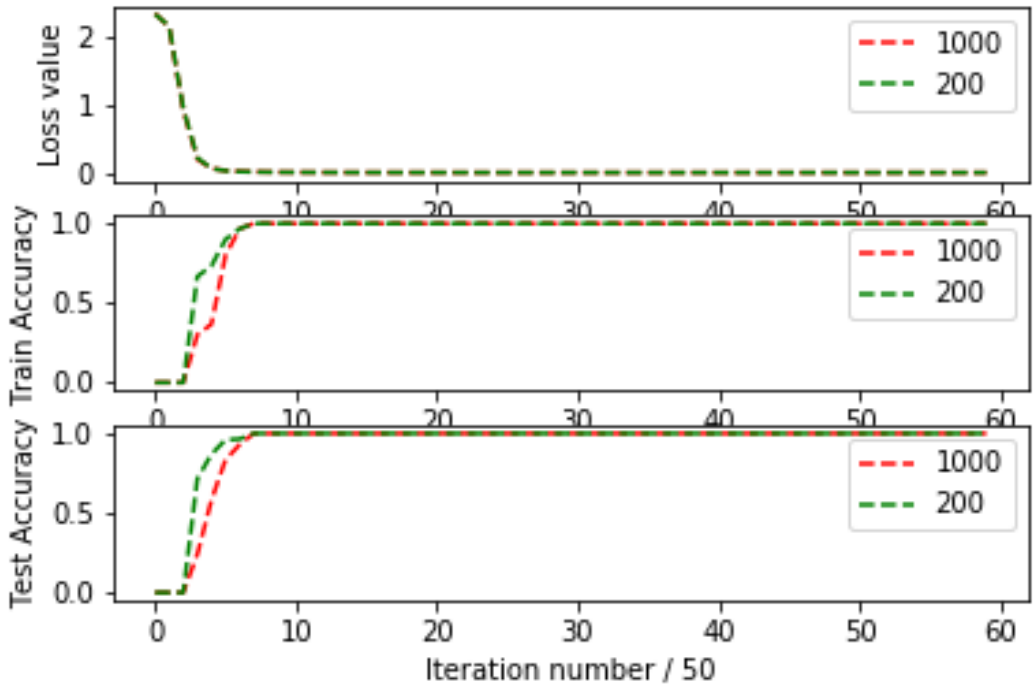
在输出层用全连接层后输出条件概率。

```
logits = self.dense(r_out)
```

在此网络基础上使用Adam优化器优化。

```
optimizer = torch.optim.Adam(params=model.parameters(), lr=0.001, betas=(0.9,
0.999), eps=1e-08, weight_decay=1e-04, amsgrad=False)
```

测试结果，**1000以内数字测试结果准确率100%**



。

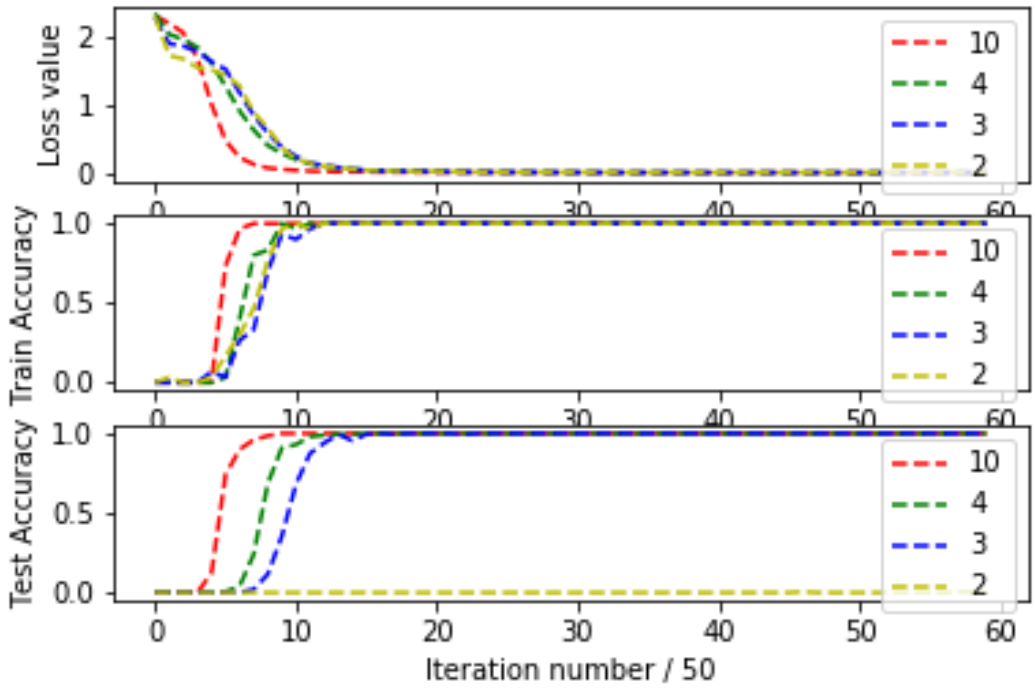
训练集最高位数 K	测试集最高位数	测试集准确度
20	20	100%
200	200	100%
1000	1000	100%

任务二

实验一

类比人类的学习加法的过程，我们并不是只会一定位数的运算，而是在掌握了进位的能力后，可以算任意位数的加法，因此我对保证测试集仍为最高位1000位的基础上，减少训练集的位数，并观察普通RNN网络的准确性变化。通过表格中的结果可以看到，即使训练集的最高位数仅有各位，**两层RNN网络**仍在非常高位的测试集上验证了学会了加法的能力。

但此泛化能力止步于用**两位数**训练，从测试集结果可以看到一直是零。

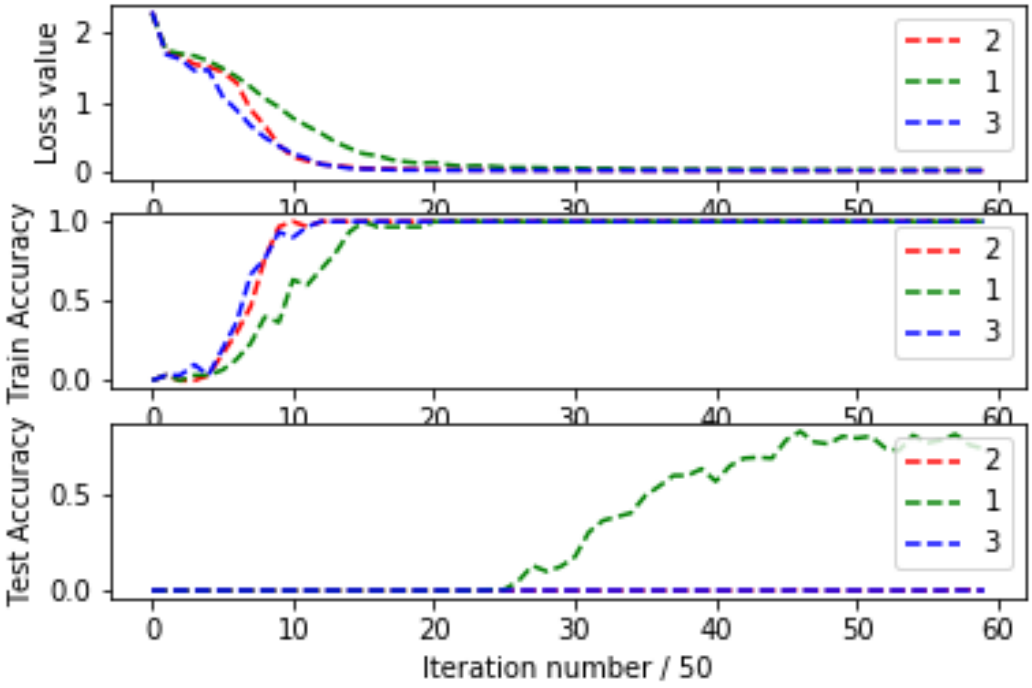


训练集最高位数	测试集最高位数	测试集准确度
100	1000	100%
10	1000	100%
4	1000	100%
3	1000	100%
2	1000	100%

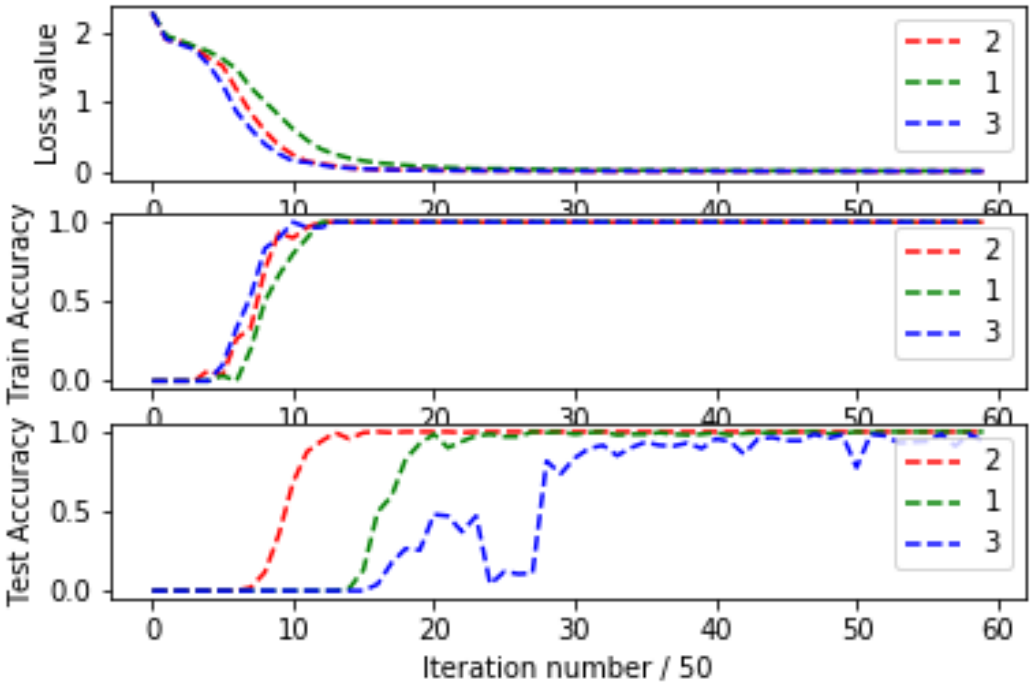
实验二

为了进一步研究**rnn的堆叠情况**对加法的泛化能力的影响。我将调整rnn的堆叠层数，并观查堆叠层数对训练集为2位数時测试集准确度的变化和训练集为三位時测试集准确度的变化。

首先看下图**训练集为两位数**時，1000位测试集的准确度只有**1层rnn**在慢慢趋近1，而堆叠2层和3层rnn完全没有表示能力。



其次，我们再进行训练集为三位数，测试集为1000位数的实验，结果如下图。可以看到训练集为3位数時，rnn任何堆叠情况测试集准确度都可以收敛到1，但是2层rnn收敛最快，1层其次，但是3层收敛最慢。

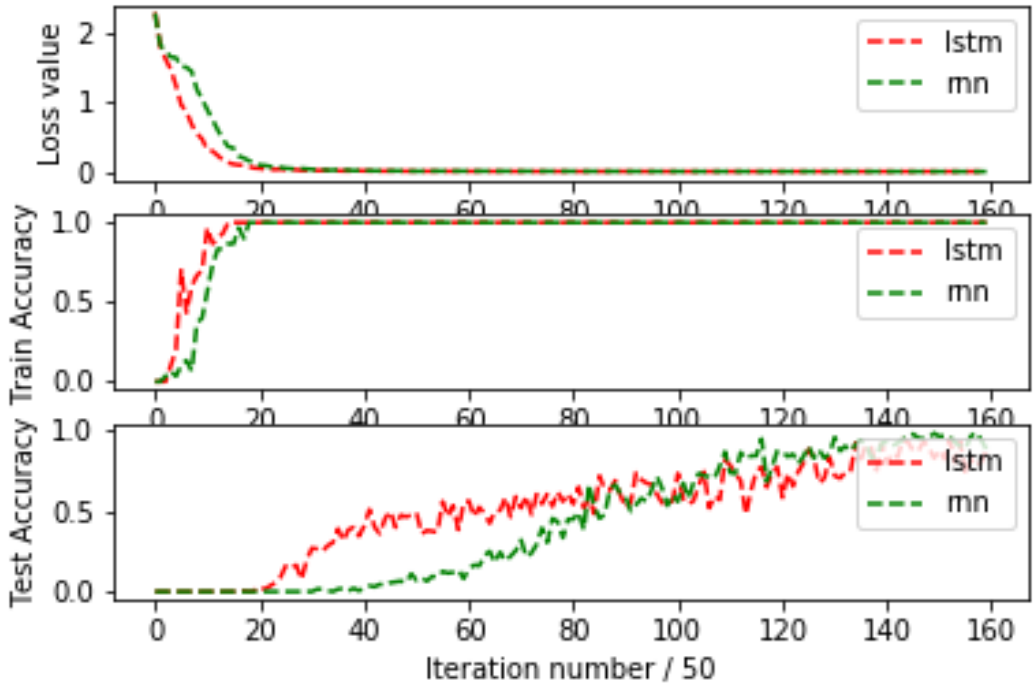


通过实验二，我们得到：一层rnn有能力具有在两位数训练集上泛化到1000位测试集的能力，尽管它的收敛速度会比较慢。基于此实验结果，接下来的实验尽可能在能少的堆叠层上进行。

实验三

为了探究更多类型的特征层的分类效果，在这一实验对网络结构 **lstm 和GRU**网络进行了探究。同样进行用两位数的训练集，1000位测试的实验。由于GRU网络的表现过差，我们只讨论lstm的结果。

当我加大迭代次数，此时单层 lstm 和单层 rnn 都展现出非常好的泛化能力，尽管以2位数作为训练集但是在1000位数的测试集最终可以收敛到 90%以上。但两者没有表现出太大差别，大约在相同的迭代次数开始收敛，但rnn收敛次数能稍快一些。



网络	训练集最高位数	测试集最高位数	测试集准确度
lstm	4	1000	100%
lstm（1层）	3	1000	100%
lstm（1层）	2	1000	92%
gru	4	1000	92.5%
gru	3	1000	0%

讨论

让机器学习运算并不是让机器只能在训练集的位数上进行运算，而相反希望机器拥有学会进位的能力从而在任意位数上学会运算。因此本报告基于本身已经非常好的rnn模型着重讨论了如何能用更低的位数去掌握高位运算能力。从结果中可以看到 **一层rnn** 和 **一层lstm**可以使用两位数字的训练集掌握在1000位数字上运算加法的能力，这种rnn的泛化能力超过我的想象。

我在考虑这个assignment時，review过几篇让机器学会加法的文章，有的使用NTM增强记忆来实现对更高位的运算，或者使用名为N-GPU的模型来实现低位训练到高位测试的泛化。其中来自googlebrain的N-GPU这篇着重用它的模型和堆叠式rnn和带注意力的lstm进行了对比。但是鉴于我的report的实验结果，堆叠层数会对rnn的低位泛化能力产生严重的影响，所以我很怀疑这篇文章的rnn和lstm之所以结果比较差是因为他使用了堆叠。而如果本身不堆叠的rnn和lstm的泛化能力已经非常惊人了，唯一的缺点是收敛比较慢，但是因为训练集本身位数非常小，花的时间也不多。

代码调用

python source.py 100 100 2 "rnn" 1000

第一个参数为训练集最高位数。

第二个参数为测试集最高位数。

第三个参数为网络堆叠层数。

第四个参数为网络类型，可选“lstm","gru",”rnn“。

第五个参数为迭代次数，推荐1000或者3000。