PRML Assignment 2 报告

1. 模型设置

1.1 简单模型 简单模型的主体为单层 RNN 层。首先将输入的两个数字 x 和 y 按数位拆开，令 x_i 和 y_i 分别表示这两个数字的第 i 位，高位通过补 0 来保持两个数字数位相同。在网络中，先将 x_i 和 y_i 通过 embedding 层映射到 32 维空间中，然后将 x_i 和 y_i 按位拼接为 64 位向量，之后就可以作为一个序列输入至 RNN 层中。最后将 RNN 输出的每个数位经过一个 32 维到 10 的全连接层，取其中最大值的下标作为该位的预测结果。

训练时将输出经过 softmax 函数后计算交叉熵损失，使用 Adam 算法进行参数学习。学习过程一共 500 步，每步为一个大小 64 的 batch 进行学习。其中 Adam 算法的学习率设定为 0.03。

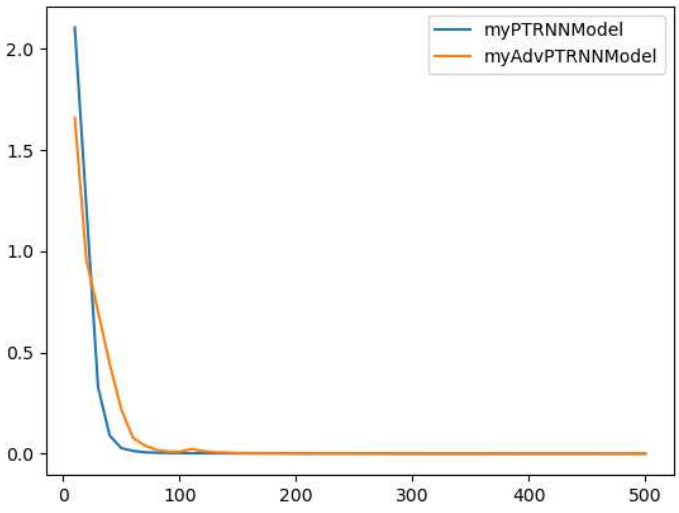
1.2 改进模型 该模型尝试模仿加法运算的结构。首先在加法过程中涉及到的数字最大为 20（$9+9+2$，其中 2 是最大进位），因此一开始的嵌入只映射到 21 维空间。该模型对数字的处理和简单模型相同，只是嵌入后不进行向量的拼接，而是使用一个输入维度和隐藏层维度均为 21 的 GRU 单元作为“加法器”，将 x_i 作为加法器的隐藏层输入，y_i 作为外部输入。其输出结果是一个 21 维的向量 g_i 表示 x_i 和 y_i 相加后的结果。然后将 g_i 和上一位的进位输出 h_{i-1} 又通过加法器连接，得到该位相加的最终结果 $\hat{h}_i$。$\hat{h}_i$ 一方面经过一个 21 维到 10 维的全连接层作为该位的输出，另一方面经过一个 21 维到 21 维的全连接层作为进位输出。上述结构作为该模型的网络的基本单元，串接起来构造一个加法网络。

训练过程的设置和简单模型基本一致，除了 Adam 算法的学习率设置为 0.05。

测试时按和生成训练集相同的方式生成 10^4 个数据进行计算，并且计算正确率。

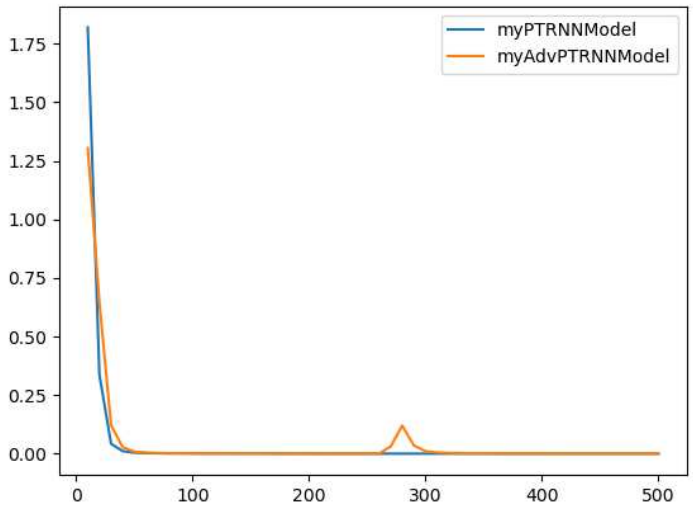
2. 参数学习

2.1 10 位数以内的数据 首先在 $0 \leqslant x, y < 10^{10}$ 的范围的数据上进行学习。两个模型的学习过程中 loss 的变化过程如下所示：



二者在前 100 步内 loss 快速下降，最后均逐渐往 0 靠近。前 100 步内，改进模型一开始 loss 下降快，之后速度放慢，简单模型 loss 逐渐低于改进版本。最终二者在测试集上的正确率均为 100%。

2.2 100 位数以内的数据 先将 x 和 y 的范围上限改为 10^{100}。两个模型学习过程中 loss 的变化如下：



基本上和先前的数据表现一致，但是改进模型在训练过程中出现了 loss 变化的波动。实际在多次测试的过程中改进模型并没有简单模型表现稳定。最终二者的正确率依然为 100%，数据位长的变化并没有特别影响两个模型的表现。

2.3 分析 改进模型虽然结构上应该更贴近本问题，但是由于其结构复杂性，表现并不如单纯的 RNN 稳定。在维度上改进模型更为精简，并且能力和简单模型相当。

实际上因为加法问题不同的数位之间的影响，随着数位之间的距离的增加，是指数级下降的。并且对于每一个数位，其结果只由 x_i、y_i 和上一位的进位决定，实际状态数量非常少，因此并不会因为长程依赖问题导致数位增加时学习能力下降。总体而言，基于 RNN 的网络是比较适合做加法问题的。

3. 程序运行

运行 10 位数以内的数据：

```
python source.py
```

运行 100 位数以内的数据：

```
python source.py --len=100
```

如果要显示两个模型学习过程中 loss 的变化，可以在命令行中加上 --show-loss-history 选项。