PRML Assignment2 Report

问题

本次的作业要求使用RNN来完成两个整数求和。

具体来说，输入两个序列A, B

$$A = \{a_0, a_1, \ldots, a_k\}$$

$\forall 0 \leq i \leq k, 0 \leq a_i < 10$

$$B = \{b_0, b_1, \ldots, b_k\}$$

$\forall 0 \leq i \leq k, 0 \leq b_i < 10$

求一个序列

$$C = \{c_0, c_1, \ldots, c_k\}$$

满足$\forall \leq i \leq k, 0 \leq c_i < 10$，且有

$$\sum_{i=0}^{k} c_i \cdot 10^i = \sum_{i=0}^{k} (a_i + b_i) \cdot 10^i$$

模型

模型1

首先完成一个最简单的RNN模型：

1. 先对输入的a_i, b_i做一个embedding到(32,)的向量x_a, x_b
2. 把x_a和x_b拼接成一个(64,)的向量x_{cat}
3. 让x_{cat}通过一个单层rnn，得到(64,)的向量r
4. 让r通过一个线性层得到一个(10,)的向量logits，表示softmax(logits)表示数字这一位是0-9的概率

在完成代码的时候要注意定义RNN的时候要设置属性*batch_first*为真，才能满足data.py造的数据。

实际在用10位数的加法训练的时候这个模型的效果非常好，当设置$steps = 500$时，就可以稳定达到100%的正确率。

猜测是因为位数太少的原因，写了一个可以生成任意位数数据的方法，

当做100位数加法的时候，当设置$steps = 500$时，也可以稳定达到100%的正确率。

当做500位数加法的时候，当设置$steps = 500$时，多次尝试后，最高可以有98.7%的正确率，但最低只能达到80.45%的正确率。当设置$steps = 1000$时，多次尝试后，最高可以有99.8%的正确率，最低也能达到98.2%的正确率。当设置$steps = 2000$时，可以稳定达到100%的正确率。

模型2

考虑怎么才能在更少的steps中做500位加法时，稳定达到100%的正确率。

考虑把每层网络变大：

修改后的RNN模型：

1. 先对输入的a_i, b_i做一个embedding到(64,)的向量x_a, x_b
2. 把x_a和x_b拼接成一个(128,)的向量x_{cat}
3. 让x_{cat}通过一个单层rnn，得到(128,)的向量r
4. 让r通过一个线性层得到一个(10,)的向量logits，表示softmax(logits)表示数字这一位是0-9的概率

同样测试500位数的加法，当设置$steps = 500$时，可以稳定达到100%的正确率。

模型3

思考有没有其他的方法也能减少需要的steps，于是想到可以使用双层的rnn模型。

修改后的RNN模型：

1. 先对输入的a_i, b_i做一个embedding到(32,)的向量x_a, x_b
2. 把x_a和x_b拼接成一个(64,)的向量x_{cat}
3. 让x_{cat}通过一个双层rnn，得到(64,)的向量r
4. 让r通过一个线性层得到一个(10,)的向量logits，表示softmax(logits)表示数字这一位是0-9的概率

同样测试500位数的加法，当设置$steps = 500$时，可以稳定达到100%的正确率。

考虑运行的效率，模型2和模型3在我的电脑上分别需要运行67s和69s，时间相差并不大。

分析

为什么最简单的RNN就可以在这个多位数加法中起到这么好的效果？

猜测是因为加法只需要依赖上一位的结果，不需要前面的结果，几乎不存在长程依赖问题，所以最简单的RNN就可以很适合地解决这个问题。