PRML Assignment 2 report

Experiment 1

模型

第一个模型是最简单的单层rnn模型，将每一位数据用embedding映射到32个神经元里，再拼接起来，得到大小为64的单元。

同样得到大小为64的输出，再线性变换到长度为10的向量中，分别为该位为各个数字的概率。

结果

当训练次数为50次时，准确率在0.7左右；随后准确率快速上升，训练次数到60时基本为1.

分析

效果很好的原因初步认为是数据过小，因为默认的数据最大只有九位数。

Experiment 2

模型

模型完全同上，尝试扩大数据规模，将数据扩大至100位（见data.py改动）。

结果

效果也不错，训练100次时准确率0.5；150次时为1；

分析

仔细想想rnn的结构，数据位数的区别仅仅的rnn循环次数（也可以说宽度）的区别。对于一个结构良好的rnn加法器来说就是重复性工作。

Experiment 3

模型

尝试将embedding的size缩小至10。仍然使用大规模数据。

结果

效果很差，当训练次数300次时准确率仅仅0.32.

分析

本来觉得数字只有10种可能就没必要编码到很多神经元里面，但好像想法错误了，影响还是很大的。

Experiment 4

模型

尝试将embedding的size增大至64。仍然使用大规模数据。

结果

效果立竿见影，当训练次数75次时就有1的准确率；50次时也有0.66.

分析

增大rnn规模能有效增加效果。

Experiment 5

模型

将rnn变为双层，测试效率（embedding恢复为32）。

结果

训练速度慢了很多，同时效果也好了不少。训练次数为50次时准确率为0.13；训练次数为75次时准确率为0.66；训练次数为100次时准确率为1；

分析

总的效率和单层的差不多，甚至还差一些。

总结

最简单的单层rnn效果就很好了，只要rnn大小不要太小。原因其实也很直观，把rnn展开以后就是前一项影响后一项，又由于是加法，最多进位1，因此单层就很够用了。也完全不需要LSTM这种网络，因为每一位是否进位都只影响其后面一位，没有长距离依赖。