模式识别与机器学习 作业二 报告

简述

本次实验中尝试用RNN完成整数求和问题。 具体来说，输入是两个数字序列

$$A = \{a_0, a_1, \dots a_n\}$$
$$B = \{b_0, b_1, \dots b_n\}$$

其中$a_i, b_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$，且$a_n = b_n = 0$
也可以把它们写作整数形式：

$$\mathrm{Int}(A) = \sum_{i=0}^{n} a_i 10^i$$

输出同样是一个长度为n的序列$C = \{c_0, c_1, \dots c_n\}$满足

$$\mathrm{Int}(C) = \mathrm{Int}(A) + \mathrm{Int}(B)$$

且$c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$。不难证明这样的序列是唯一的。

模型

我首先尝试用一个简单的RNN模型：

1. 将a_i和b_i做embedding得到他们的分布式表示x_i, y_i
2. 令$r'_i = [x_i; y_i]$，这里$[;]$表示向量拼接。
3. 将序列r'_i扔进一个单层RNN模型求出一个序列表示r_i
4. 将输出过一个线性层得到最后结果的分布：$p_i = Wr_i + b$，其中W将r_i映射到一个10维空间。
5. 而预测结果为$P\{c_i = d\} = \mathrm{softmax}(p_i)_d$

使用example.py默认给出的交叉熵损失和Adam优化器进行训练。
取最后一个step的模型来进行测试。

结果

因为loss下降的非常快，我只训练了1000个step。得到 test accurancy = 100%

讨论：更难的数据

可以看到测试集上结果非常高。

因为example.py提供的数据生成器只能生成10位以内的数据，我怀疑这个高结果是由于数字太短了。因此我重新写了数据生成器，使之可以生成任意长度的数据。

为了保证训练集和测试集不重合，采用和example中类似的生成方法，若数据最长长度为n，则训练集在$[\frac{10^{n-1}}{2}, 10^n)$中采样，而测试集在$[0, \frac{10^{n-1}}{2})$中采样。

结果为：

数字长度	测试集准确率
10	100%
50	100%
100	99.2%

可以看到即使数字很长，他还是做得很好。

讨论：1元左感知模型

实际上，RNN表现很好的原因可能是因为这个任务具有很好的局部性：如果没有进位，则$c_i = a_i + b_i$是和其他$c_{1\dots i-1}$都无关的。
因此RNN只在进位，实际上是连续进位的情况下有效：

设$c'_k = a_k + b_k$，则c_k实际上是c'_k的进位版本，因此：

$$\exists m, c'_{m-1} < 10, c'_m = c'_{m+1} = \ldots = c'_{k-1} \geq 10$$

则显然c_k只和$c'_{m\dots k}$有关，即：

$$p(c_k|a_{1\dots n}, b_{1\dots n}) = p(c_k|a_{m\dots k}, b_{m\dots k})$$

设$l_k = k - m$为位置k的依赖距离，因此a和b都是随机生成的，l_k不会太大。
事实上，我猜想**对于大多数的k，有$l_k \in 0, 1$。**

因此这启发我设计一个这样的模型：

$$p_i = \mathrm{MLP}([x_i; y_i; x_{i-1}; y_{i-1}])$$
$$P\{c_i = d\} = \mathrm{softmax}(p_i)_d$$

这里$x_i = \mathrm{embedding}(a_i), y_i = \mathrm{embedding}(b_i)$。
而MLP表示一个多层前馈网络，这里我用的是一个用relu激活的二层网络：

$$\mathrm{MLP}(x) = W_2\sigma(W_1 x + b_1) + b_2$$

其中$\sigma(x)_i = \max\{x_i, 0\}$表示relu激活函数。

我称这个模型为1元左感知模型，因为他仅能感知到每个元素左侧的一个元素。 在各个长度的数据集上训练和测试的结果为：

数字长度	测试集准确率	测试集数字准确率
10	65.70%	96.04%
50	8.65%	95.20%
100	0.70%	95.10%

其中准确率是整个数预测正确的频率，即C的准确率
而数字准确率是预测对的数字（即c_i）个数占总数字个数的比例。

可以看出虽然这个模型的准确率不高，但是实际上对大部分数字都预测对了。只不过数越长，数字越多，因此出错的概率也越大。

另外，这个模型不像RNN有递归结构，因此可以充分利用GPU的并行性，因此训练和测试更快。
这个模型的训练用时只有RNN模型的一半左右（在GPU上）。