Homework_3

Part_1

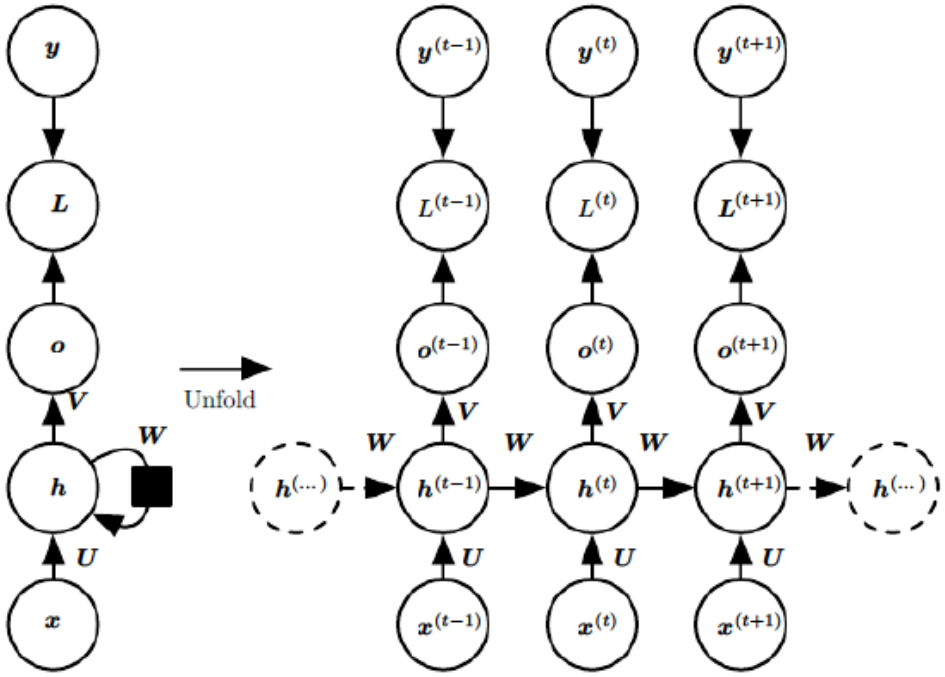


图 3-1

如上图 3-1，是 RNN 的标准结构，图中每个箭头代表做一次变换，也就是说箭头连接带有权值。左侧是折叠起来的样子，右侧是展开的样子，左侧中 h 旁边的箭头代表此结构中的“循环“体现在隐层。

在展开结构中我们可以观察到，在标准的 RNN 结构中，隐层的神经元之间也是带有权值的。也就是说，随着序列的不断推进，前面的隐层将会影响后面的隐层。图中 O 代表输出， L 代表损失函

数，x 是输入，h 是隐层单元，y 为训练集的标签。这些元素右上角带的 t 代表 t 时刻的状态，其中需要注意的是，因策单元 h 在 t 时刻的表现不仅由此刻的输入决定，还受 t 时刻之前时刻的影响。V、W、U 是权值，同一类型的权连接权值相同。我们可以看到，“损失“也是随着序列的推进而不断积累的。

这里先给出 RNN 的前向传播：

对于 t 时刻：

$$h^{(t)}=\varphi(Ux^{(t)}+Wh^{(t-1)}+b)\ h^{(t)}=\varphi(Ux^{(t)}+Wh^{(t-1)}+b)\ h^{(t)}=\varphi(Ux^{(t)}+Wh^{(t-1)}+b)$$

$$h^{(t)}=\varphi(Ux^{(t)}+Wh^{(t-1)}+b)$$

其中$\varphi()$为激活函数，一般来说会选择 tanh 函数，b 为偏置。

t 时刻的输出：

$$O^{(t)}=Vh^{(t)}+c$$

最终模型的输出为：

$$\hat{y}^{(t)}=\sigma(O^{(t)})$$

其中 σ 为激活函数，通常 RNN 用于分类，故这里一般用 softmax 函数。

1） 根据上述前向传播和结构图，推导 RNN 的中 U、V、W 的更新过程，（W、U 的更新仅需给出第三时刻，即 t=3 的更新）。

2） 我们常用的激活函数有 sigmoid()、tanh()和 ReLU()三种函数，请证明三种函数应用在 RNN 中时，会出现哪种梯度问题，并给出简要证明。

选做题：

请根据下面 LSTM 的结构图和前向传播过程推导出 LSTM 的反向传播：

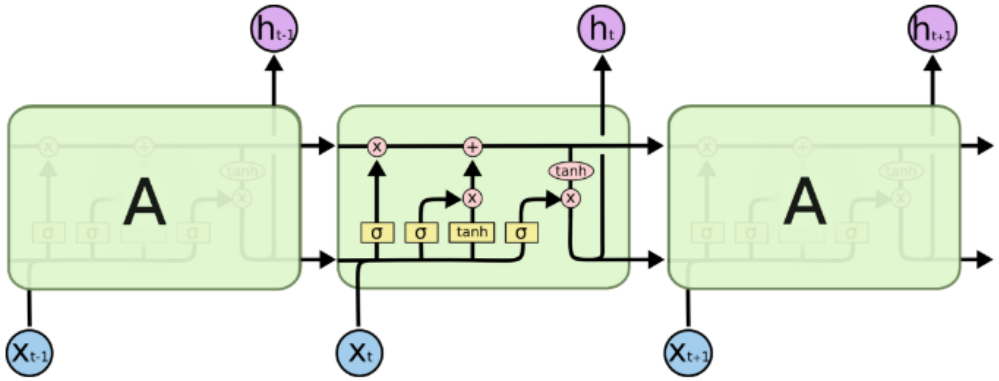


图 3-2

从上图中可以看出，在每个序列索引位置 t 时刻向前传播的除了和 RNN 一样的隐藏状态 $h^{(t)}$，还多了另一个隐藏状态，如图中上面的长横线。这个隐藏状态我们一般称为细胞状态(Cell State)，记为 $C^{(t)}$。如下图所示：

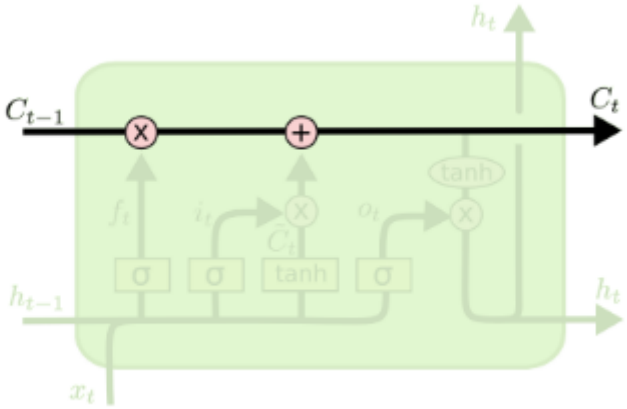


图 3-3

除了细胞状态，LSTM 图中还有了很多奇怪的结构，这些结构一般称之为门控结构(Gate)。LSTM 在在每个序列索引位置 t 的门一般

包括遗忘门，输入门和输出门三种。

1）遗忘门子结构如下图所示：

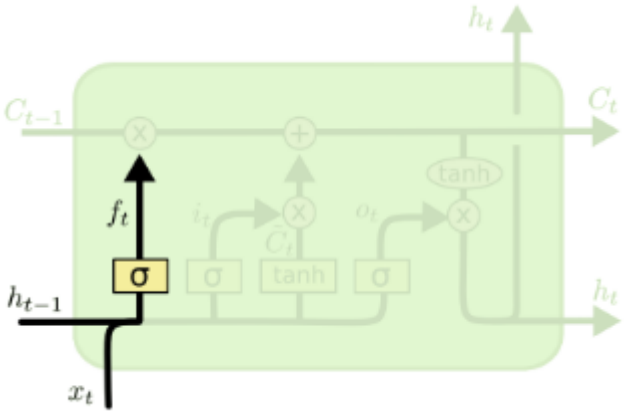


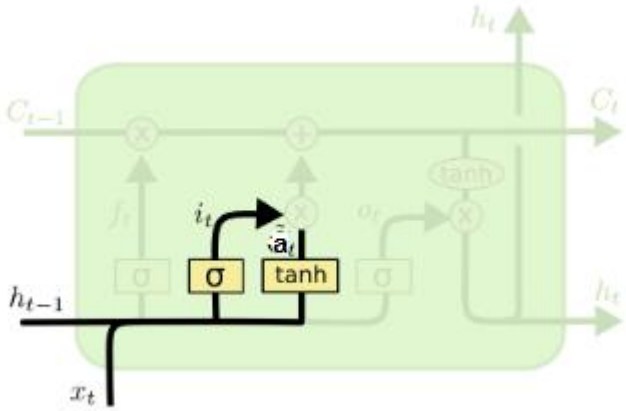
图 3-4

图中输入的有上一序列的隐藏状态 $h^{(t-1)}$和本序列数据 $x^{(t)}$，通过一个激活函数，一般是 sigmoid，得到遗忘门的输出 $f^{(t)}$。由于 sigmoid 的输出 $f^{(t)}$在[0,1]之间，因此这里的输出 $f^{(t)}$代表了遗忘上一层隐藏细胞状态的概率。用数学表达式为：

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

其中W_f, U_f, b_f 为线性关系的系数和偏倚，和 RNN 中的类似。σ 为 sigmoid 激活函数。

2）输入门如下图所示：



从图中可以看到输入门由两部分组成，第一部分使用了 sigmoid 激活函数，输出为 $i^{(t)}$,第二部分使用了 tanh 激活函数，输出为 $a^{(t)}$, 两者的结果后面会相乘再去更新细胞状态。用数学表达式即为：

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$a^{(t)} = tanh(W_a h^{(t-1)} + U_a x^{(t)} + b_a)$$

其中$W_i, U_i, b_i, W_a, U_a, b_a$，为线性关系的系数和偏倚，和 RNN 中的类似。$\sigma$为 sigmoid 激活函数。

3）前面的遗忘门和输入门的结果都会作用于细胞状态 $C^{(t)}$，细胞的更新情况如下图：

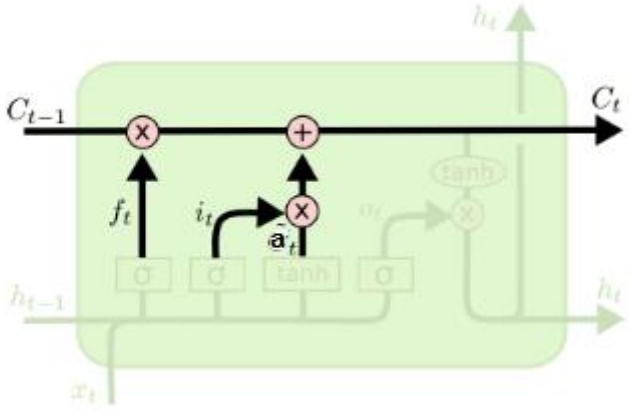


图 3-5

细胞状态 $C^{(t)}$由两部分组成，第一部分是 $C^{(t-1)}$和遗忘门输出 $f^{(t)}$的乘积，第二部分是输入门的 $i^{(t)}$和 $a^{(t)}$的乘积，数学表达式如下：

$$C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)}$$

⊙ 为 Hadamard 积。

4）之后输出门结构如下：

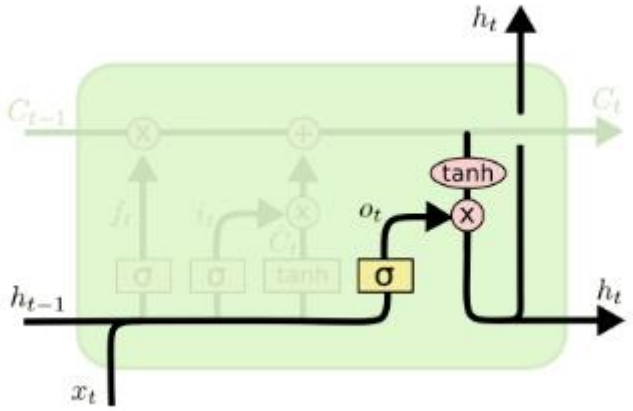


图 3-6

隐藏状态 $h^{(t)}$的更新由两部分组成，第一部分是 $o^{(t)}$, 它由上一序列的隐藏状态 $h^{(t-1)}$和本序列数据 $x^{(t)}$，以及激活函数 sigmoid 得到，第二部分由隐藏状态 $C^{(t)}$和 tanh 激活函数组成, 即：

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$h(t) = o^{(t)} \odot tanh(C^{(t)})$$

最后更新当前序列索引预测输出：

$$\hat{y}^{(t)} = \sigma(Vh^{(t)} + c)$$

请给出 LSTM 的反向传播过程：

Part_2

给大家提供了三个 Python 的文件，分别为 data_processor.py，model.py，train.py，其中 data_processor.py 是数据处理文件，需要用到的数据在 data 文件夹中；model.py 文件中定义了模型，train.py 文件负责模型的训练和测试。

我删除了三个文件中的部分必须代码，data_processor 中需要补写 tokenizer()和 get_stop_words()，之后的 load_data 部分可以选择使用不同的词向量，model 中需要定义模型，train 中需要定义优化函数。

希望大家能够把这部分代码填上，模型的选择没有限制，希望大家能够给出较好的训练结果。

（注：在一开始运行 train.py 的时候，同学们会遇到“Python int too large to convert to C long”的报错。原因是 Python 的 int 没有上限的，但是 C 的 int 有上限，如果没有对大整数做调整，在传入参数时就会出错。报错的地方在 data_processor.py 文件中的“train, val = data.TabularDataset.splits()”位置，我选择的修改方法是到 utils.py 文件（utils 的路径："D:\Anaconda1\envs\pytorch\lib\site-packages\torchtext\utils.py"）里将第 130 行的“csv.field_size_limit(sys.maxsize)”注释掉，如果同学们有什么其他好的修改方式，欢迎交流）。