Gated Recurrent Neural Networks

本文主要介绍 RNN 中常用的 LSTM 和 GRU 单元，并进行比较。

Introduction

RNN 在许多机器学习任务中有比较好的结果，特别是输入和输出是变长的情况，这些好的效果往往是通过加入一些高级的 recurrent hidden units 来实现的，而不是传统的 $tanh$ 单元，如 LSTM，GRU。本文在序列建模任务中评估了这两种单元和更传统的 $tanh$ 单元。

根据实验结果，在一些数据集上，使用固定数目的参数，GRU 要比 LSTM 表现好。

Backgroud

RNN 可以处理变长序列输入，通过一个递归隐藏状态，其值依赖于上一个时刻的值。即，给定序列 $x = (x_1, x_2, \dots, x_T)$，隐藏状态更新为：

$$h(t) = \begin{cases} 0, & \text{t=0} \\ \phi(h_{t-1}, x_t), & \text{otherwise} \end{cases}$$

其中，ϕ 是非线性函数，比如 sigmoid，上式也可写为：

$$h_t = g(Wx_t + Uh_{t-1})$$

有生成任务的 RNN（比如文本生成、机器翻译等），给定当前状态 h_t，对序列的下一个元素生成一个概率分布，并可以通过使用 EOS 来获取变长序列的分布。序列概率可分解为：

$$p(x_1, \cdots, x_T) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_T|x_1, \cdots, x_{T-1})$$

每个条件概率分布可建模为：

$$p(x_t|x_1, \cdots, x_{t-1}) = g(h_t),$$

h_t 就是上面的隐藏状态。

但该模型的问题是，获取长时依赖是很难的，因为存在梯度消失（大多数情况）和梯度爆炸（出现较少）问题，这使得基于梯度的优化方法难以奏效。有两种主流的方法来解决该问题：一是设计比 SGD 更好的学习算法，比如使用 *clipped gradient*，二是设计更高级的激活函数或循环单元，这就引入了 LSTM 和 GRU。

Gated Recurrent Neural Networks

Long Short-Term Memory Unit

长短时记忆（lstm）单元最早提出于 1997 年，随后，有很多种方法对其做了微小改动。每个 LSTM 单元在

时刻 t 有记忆单元 c_t，则 LSTM 单元的输出为 h_t：

$$h_t = o_t \tanh(c_t)$$

其中，o_t 是输出门，用来控制长期记忆对当前输出的影响，输出门的计算为：

$$o_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t)$$

σ 为 sigmoid 函数，V_o 为对角矩阵。 记忆单元 c_t 通过部分忘记已有记忆，然后加入新的记忆内容 $\tilde{c_t}$ 来更新：

$$c_t = f_t c_{t-1} + i_t \tilde{c_t}$$

其中，新的记忆内容为：

$$\tilde{c_t} = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})$$

当前记忆被忘记的程度由遗忘门 f_t 控制，新的记忆内容被加入到记忆单元的程度由输入门 i_t 控制，计算方式为：

$$f_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1}),$$
$$i_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}).$$

V_f 和 V_i 都是对角矩阵。

LSTM 通过这些门来决定是否保留当前记忆单元，直觉上来说，如果 LSTM 单元在早阶段能捕获到输入序列的重要特征，它就可以容易地将这些信息传递下去。

Gated Recurrent Unit

与 LSTM 相似的是 GRU 也有门单元，来控制 unit 之间的信息流，不同的是，GRU 没有单独的记忆单元。

时刻 t GRU 的激活 h_t 是上一个激活 h_{t-1} 和 候选激活的 $\tilde{h_t}$ 的线性插值：

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h_t}$$

其中，更新门 z_t 控制单元更新程度，计算如下：

$$z_t = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})$$

将已有状态和新状态做线性求和和 LSTM 是类似的，但 GRU 没有机制来控制单元 expose 的程度，在每个时刻都 expose 整个状态。

候选激活$\tilde{h_t}$ 计算方式为：

$$\tilde{h_t} = \tanh(W \mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

其中，$\mathbf{r}_t$ 是一组重置门，当关闭的时候（r_t 接近 0），重置门相当于让单元读取输入序列的第一个字符，忘记之前计算的状态。

重置门 r_t 计算如下：

$$r_t = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1})$$

更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门控制前一状态有多少被写入到当前的候选集 $\tilde{h_t}$ 上，重置门越小，前一状态的信息被写入的越少。

Discussion

LSTM 和 GRU 的相同点主要为：

- 传统的 recurrent unit 仅仅使用根据当前输入和上个隐藏状态的值计算出的新值来替代单元的内容，而 LSTM 和 GRU 会保留当前内容，在上面加入新内容。

不同点在于：

1. 对 memory 的控制：

   LSTM: 用输出门控制，传递给下一个 unit

   GRU: 直接传递给下一个 unit，不作任何控制

2. input gate 和 reset gate 作用位置不同

   LSTM: 计算 $\tilde{c_t}$ 时，不对上一时刻的信息做任何控制，而是用 forget gate 独立实现这一点； GRU: 计算 $\tilde{h_t}$ 时，利用 reset gate 对上一时刻的信息进行控制。

标准 LSTM 和 GRU 的差别并不大，但是都比 tanh 要明显好很多，所以在选择标准 LSTM 或者 GRU 的时候还要看具体的任务是什么。

使用 LSTM 的原因之一是解决 RNN Deep Network 的梯度错误累积太多，以至出现梯度消失和梯度爆炸，使训练难以继续。GRU 的构造更简单：比 LSTM 少一个 gate，这样就少几个矩阵乘法。在训练数据很大的情况下 GRU 能节省很多时间。