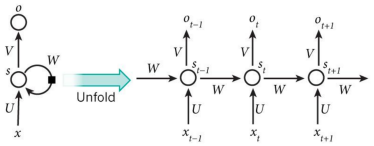
3.序列建模（RNN系列）

1.RNN

RNN通过不同时刻的参数共享，并借助循环核提取序列特征信息，实现对序列数据的建模。RNN可以用来处理不定长度的序列数据，但是他的本身参数在每个序列中是固定的。下图是RNN的具体结构，两个结构都表示RNN但是左图表示RNN的内部结构，右图表示RNN的工作状态：



s_t表示记忆体内当前时刻存储的状态信息。

x_t为当前时刻的输入信息。

y_t表示当前时刻输出的特征信息。

RNN的数学表达式可以表达如下：

$$s_t = tanh(W_s[x_t; s_{t-1}] + b_s)$$

$$y_t = softmax(h_t w_{sy} + by)$$

长时依赖问题\梯度消失问题：

上述公式中的W_s可以切分成$[W^T; U^T]^T$,s_t替换成h_t,RNN的定义公式可以写为：

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

$$进一步，可表示为：h_t = f_t(x_t, h_{t-1}; W, U)$$

其中h_t是每一步的输出，它由当前输入x_t和前一刻的输出h_{t-1}共同决定，而W, U是可训练参数。为了简化分析，我们假设x_t, h_t, W, U都为一维向量，却又不失一般性，并且对于$t+1$时刻产生的误差e_{t+1},在t个序列中，误差e_{t+1}对于W的梯度由$t+1$条路径构成，每条路径的梯度计算为：

$$\frac{\partial e_{t+1}}{\partial W} = \frac{\partial e_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial W}$$

$$\frac{\partial e_{t+1}}{\partial W}=\frac{\partial e_{t+1}}{\partial h_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}\frac{\partial h_t}{\partial W}$$

....

$$\frac{e_{t+1}}{W}=\frac{\partial e_{t+1}}{\partial h_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$

误差对W的梯度即为每条路径之和：

$$\frac{\partial e_{t+1}}{\partial W}=\sum_{1<k\leq t}\frac{\partial e_{t+1}}{\partial h_t}\prod_{k<i\leq t}\frac{h_i}{h_{i-1}}\frac{\partial h_k}{\partial W}$$

上述式子中，$|\prod_{k<i\leq t}\frac{\partial h_i}{\partial h_i-1}|\leq\eta^{t-k}$,从而说明当距离足够远时，$\eta$大于1会引起梯度爆炸，小于1会引起梯度消失（此为RNN的长时依赖问题）。

长时依赖问题\梯度消失解决方案：

除了一些优化技巧外，我们解决长时依赖问题的方式就是改变模型，采用一种类似resnet的改进策略，缓解梯度消失：

$$h_t=h_{t-1}+f(x_t,h_{t-1};\theta).....(1)$$

但是上述公式中$\frac{\partial h_i}{\partial h_{i-1}}$大于1.可能会引起1.**梯度爆炸；2.记忆容量问题**，随机h_t信息的不断输入，假设f为Logistc函数，会随着h_t的不断增大而接近饱和，从而使丧失的信息越来越多。

LSTM和GRU便是在公式1的基础上做改进。

2.LSTM

LSTM缓解了RNN存在的长序依赖问题，LSTM的运行过程如下：

1.将上个时刻的输出信息（隐藏层信息）和当前时刻的输入信息合并，进行全连接处理（到底是几维），获得取值为0到1之间的门结构（遗忘门、输入门、输出门）。

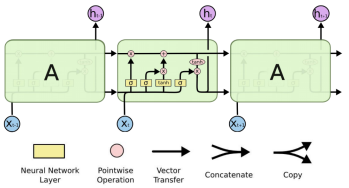
2.将上个时刻的待保留信息作为遗忘门的输入，并通过遗忘门处理，确定信息的丢弃或保留。（这也是LSTM缓解梯度消失的重要结构）

3.将上个时刻的输出信息（隐藏层信息）和当前时刻的输入信息的合并信息通过非线性处理，并通过输入门。（这是LSTM增强模型非线性表达能力的重要结构）

4.将输入门的输出信息和遗忘门的输出信息合并作为当前时刻对于下个时刻的待保留信息。

5.再将待保留信息依次通过非线性处理和输出门，作为当前时刻的输出信息（隐藏层丝信息）

工作流程图如下图所示：



LSTM的数学表达式如下：

$$遗忘门：f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

$$输入门：i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$输出门：o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

上个时刻待保留信息C_{t-1}通过遗忘门，上个时刻输出信息经过非线性处理后$\tilde{C}_t$通过输入门，并合并：$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

$$C_t通过非线性处理，并通过输出门：h_t = o_t * tanh(C_t)$$

3.GRU

和LSTM不太一样，GRU不引入额外的记忆单元，GRU网络也是在公式**1** 的基础上引入一个更新门来控制当前状态需要从历史状态中保留多少信息，以及需要从候选状态中接受多少新信息，GRU的更新公式为：

$$h_t = z_t h_{t-1} + (1 - z_t) f(x_t, h_{t-1}; \theta)$$

在GRU中，h_{t-1}表示上一时刻的输出信息，h_t表示当前时刻的输出信息，x_t表示当前时刻的输入信息，z_t表示更新门：

$$z_t = \sigma(W_z[x_t; h_{t-1}] + b_z)$$

在GRU中$f(x_t, h_{t-1}; \theta)$定义为：

$$\tilde{h}_t = tanh(W_h[x_i; r_t h_{t-1}] + b_h)$$

r_t表示重置门：

$$r_t = \sigma(W_r[x_i; h_{t-1}] + b_r)$$

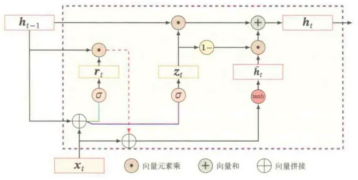
下图给出了GRU循环单元的结构，其计算过程为：

1.首先利用上一个时刻的外部状态信息和当前时刻的输入信息，计算出更新门信息z_t和重置门信息r_t。

2.其次，通过更新门z_t计算出上一个时刻需要保留的信息$z_t h_{t-1}$

3.然后，通过重置门计算出候选状态$\tilde{h}_t$需要的上个时刻信息$r_t h_{t-1}$,并将其与输入信息x_t合并，计算出候选状态$\tilde{h}_t$。

4.将候选状态$\hat{h}_t$和上个时刻保留的信息h_t通过更新门合并，计算出当前时刻的输出信息$z_t h_{t-1} + (1 - z_t)\tilde{h}(t)$。



GRU网络的循环单元结构