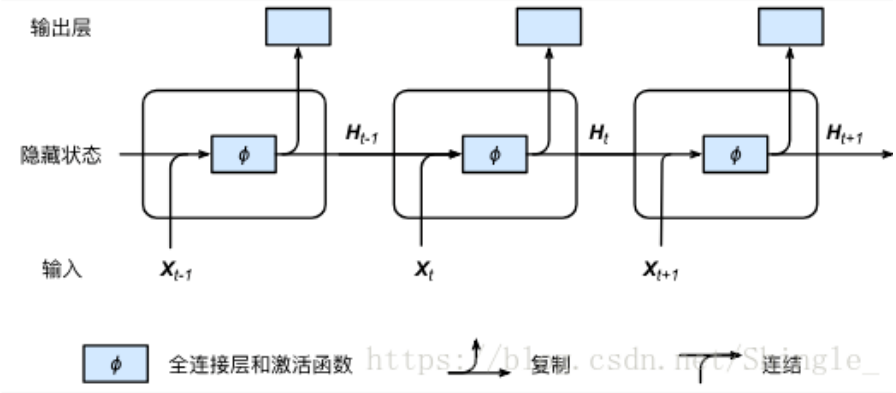
1. RNN



$$H_t = tanh(W_h H_{t-1} + W_x X_t + b_h)$$

长期依赖问题

理论上，通过调整参数，RNN是可以学习到时间久远的信息的。但是，实践中的结论是，RNN很难学习到这种信息的。每个时间步都会乘以系数W，RNN 会丧失学习时间间隔较大的信息的能力，导致长期记忆失效。

梯度爆炸/梯度消失问题

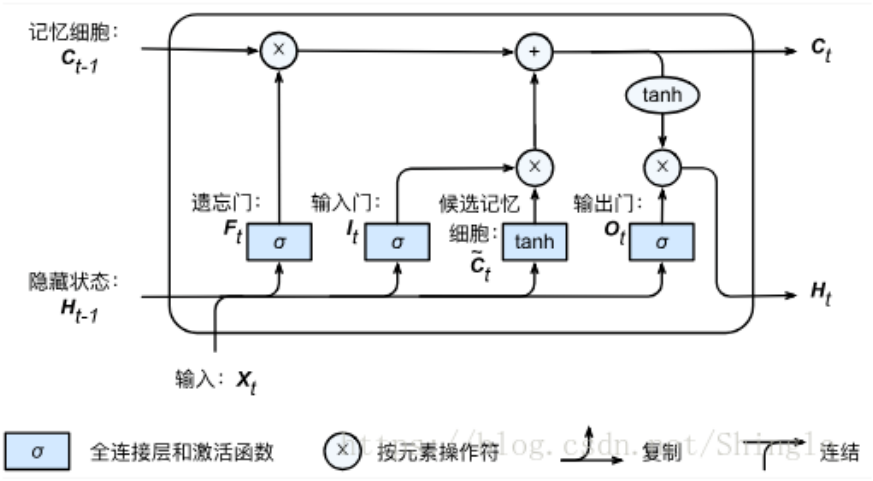
RNN常用的两种激活函数，sigmoid和tanh。如果选择sigmoid函数作为激活函数，即$f(z) = \frac{1}{1+e^{-z}}$，其导数为$f^{'}(z) = f(z)(1-f(z))$，导数的取值范围为0~0.25。如果选择tanh作为激活函数，即$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$，其导数为$f^{'}(z) = 1 - f(z)^2$，导数的取值范围为0~1。并且RNN网络的同一层中，所有时间步的W都是共享的。 因此，当选用sigmoid或者tanh作为激活函数时，大多数时候$f^{'}(z)$都是大于0小于1的，如果W的值也是大于0小于1，在计算式(3)涉及到多次$f^{'}(z)$和W的多次连乘，结果会趋于0，从而造成了**梯度消失**的问题。如果W的值特别大，则多次连乘后就会出现**梯度爆炸**的问题。

- 方法一：如果使用relu作为激活函数，即$f(z) = max(0, z)$，relu的导数在x>0时恒为1，一定程度上可以**缓解梯度消失**问题，但是如果W的值特别大，也会出现**梯度爆炸**的问题。而且当$z < 0$时，导数恒为0，会造成部分神经元无法激活（可通过设置小学习率部分解决）
- 方法二：梯度裁剪（当梯度小于某个阈值时，梯度就为某个阈值）
- 方法三：LSTM / GRU， 一定程度上模仿了长时记忆，在从t 到 t-1 的更新时都引入了加法，能够一定程度平衡梯度消失等问题

RNN等会随着序列增常而参数增多吗?

不会，RNN中的每个时间步相当于共享参数

2. LSTM



遗忘门：$F_t = sigmoid(W_{fh}H_{t-1} + W_{fx}X_t + B_f)$

输入门：$I_t = sigmoid(W_{ih}H_{t-1} + W_{ix}X_t + B_i)$

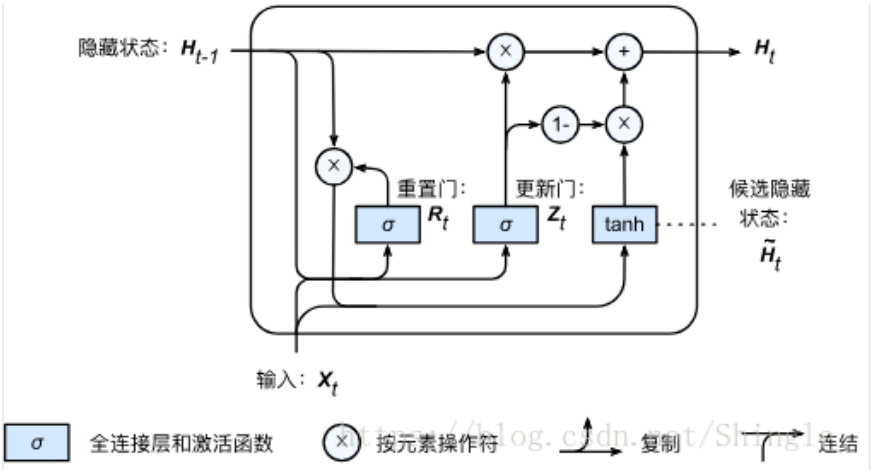
输出门：$O_t = sigmoid(W_{oh}H_{t-1} + W_{ox}X_t + B_o)$

候选记忆细胞：$\hat{C}_t = tanh(W_{ch}H_{t-1} + W_{cx}X_t + B_c)$

记忆细胞：$C_t = F_t \odot C_{t-1} + I_t \odot \hat{C}_t$

隐藏状态：$H_t = O_t \odot tanh(C_t)$

3. GRU



重置门：$R_t = sigmoid(W_{rh}H_{t-1} + W_{rx}X_t + B_r)$

更新门：$Z_t = sigmoid(W_{zh}H_{t-1} + W_{zx}X_t + B_z)$

候选隐藏状态：$\hat{H}_t = tanh(W_{hh}(R_t \odot H_{t-1}) + W_{hx}X_t + B_h)$

隐藏状态：$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \hat{H}_t$

优缺点：

- 将输入门和遗忘门合并为更新门，同时做了其他改动，使得网络更简单
- GRU 参数更少因此更容易收敛，但是数据集很大的情况下，LSTM表达性能更好