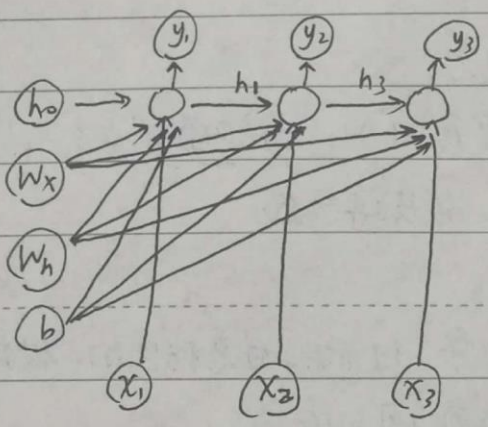
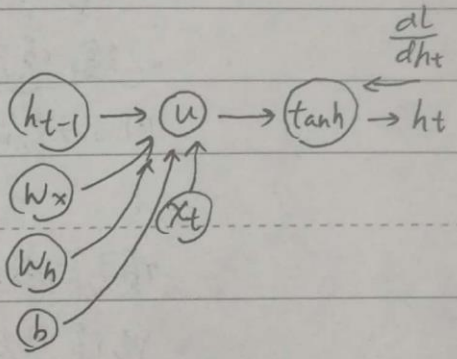
Vanilla RNN

这里就用到了 Two Layer NN 中向量化的 dot 与 + 算子的反向传导.
(详见 assignment1 的 notes), 有了前面的基础, 就很简单了. (下面直接向量化)



抽取其一 →



其中 $u = h_{t-1}W_h + x_tW_x + b$.

~~$dW_h = \frac{dL}{dh_t} h_{t-1}^T$~~

$$\frac{dL}{du} = \frac{dL}{dh_t} \odot d(\tanh(u)) = \frac{dL}{dh_t} \odot (1 - \tanh^2(u)) \quad (\odot \text{是 element-wise 乘法})$$

$$\frac{dL}{dW_h} = h_{t-1}^T \frac{dL}{du} \quad , \quad \frac{dL}{dh_{t-1}} = \frac{dL}{du} W_h^T$$

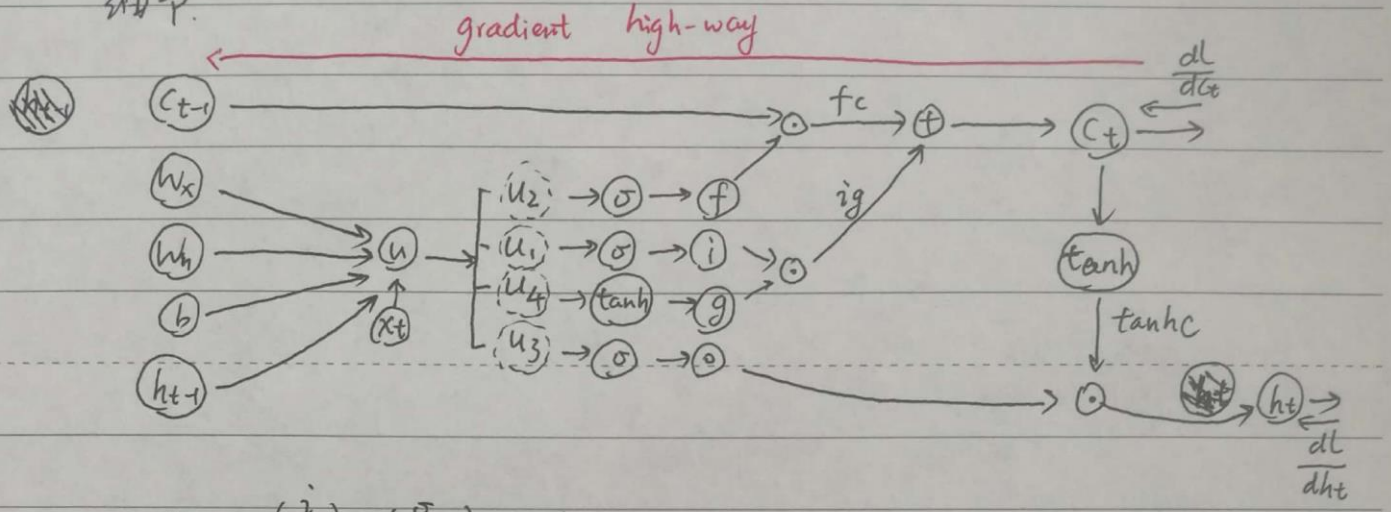
→ 套用 BPNN 向量化反向传导

$$\frac{dL}{dW_x} = x_t^T \frac{dL}{du} \quad , \quad \frac{dL}{dx_t} = \frac{dL}{du} W_x^T$$

$$\frac{dL}{db} = e_N^T \frac{dL}{du}$$

② LSTM

总体结构同 Vanilla RNN，但内部细节上多了门运算. 下面仅讨论内部细节.



其中 $\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$，$c_t = f \odot c_{t-1} + i \odot g$ (acts like accumulator)

$h_t = o \odot \tanh(c_t)$.

$\frac{dL}{do} = \frac{dL}{dh_t} \odot \tanh c$ ，$\frac{dL}{d\tanh c} = \frac{dL}{dh_t} \odot o$ ，$\frac{dL}{dc_t} \mathrel{+}= \frac{dL}{d\tanh c} \odot (1 - \tanh^2(c_t))$

$\frac{dL}{dfc} = \frac{dL}{dc_t}$ → $\frac{dL}{dc_{t-1}} = \frac{dL}{dfc} \odot f$

↳ $\frac{dL}{df} = \frac{dL}{dfc} \odot c_{t-1}$

$\frac{dL}{dig} = \frac{dL}{dc_t}$ → $\frac{dL}{di} = \frac{dL}{dig} \odot g$

↳ $\frac{dL}{dg} = \frac{dL}{dig} \odot i$.

→ $\frac{dL}{du}$ → 用 Vanilla RNN 的 BP 即可.

可见 c_t 像累加器，数量规模不会像矩阵连乘那样扩大/缩小得很快

缓解了 ~~梯~~ $\frac{dL}{du}$ 消失问题，也缓解了正向传播 h_t 爆炸问题 (这些问题大多因 max_eigvalue(W) > 1 或 < 1 且 W 连乘引起爆炸.