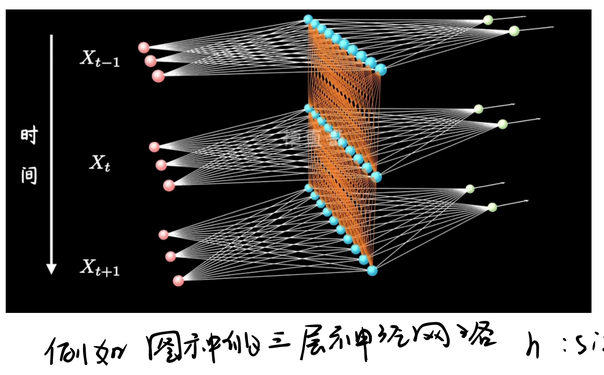
循环神经网络



在序列本身间是有关联的

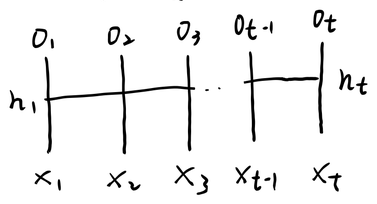
例如 图示中的三层神经网络 h : size (1×s)　b: size (1×1)

令输入为 X : size (1×n)　W_{xh} : size (n×s)　W_{hh} : size (s×s)

$h_t = W_{xh} X_t + W_{hh} h_{t-1} + b_h$

$o = W_{ho} \sigma(h_t) + b_o$

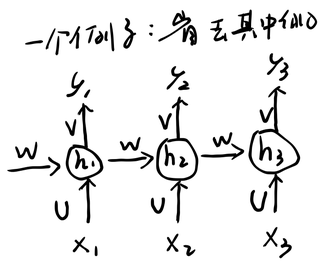
反向传播：



定义损失函数：

$J = J_1(o_1) + J_2(o_2) + \cdots + J_t(o_t)$

一个例子：省去其中的偏置元



其中 $h_t = \sigma(Ux + wh_{t-1})$

$y_t = Vh_t$

参数是共享的

$h_1 = \sigma(Ux_1 + wh_0)$　$h_2 = \sigma(Ux_2 + wh_1)$　$h_3 = \sigma(Ux_3 + wh_2)$

$y_1 = Vh_1$　$y_2 = Vh_2$　$y_3 = Vh_3$

$L = \sum L_i$

在 $t=3$ 时刻对 V, W, U 求偏导

$$\frac{\partial L_3}{\partial V} = \frac{\partial L_3}{\partial y_3} \frac{\partial y_3}{\partial V}$$

$$\frac{\partial L_3}{\partial U} = \frac{\partial L_3}{\partial y_3} \frac{\partial y_3}{\partial h_3} \left(\frac{\partial h_3}{\partial U} + \frac{\partial h_3}{\partial h_2} \left(\frac{\partial h_2}{\partial U} + \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial U} \right) \right) = \sum_{k=1}^{t} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U} = \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U}$$

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial y_3} \frac{\partial y_3}{\partial h_3} \left(\frac{\partial h_3}{\partial W} + \frac{\partial h_3}{\partial h_2} \left(\frac{\partial h_2}{\partial W} + \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} \right) \right) = \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

(Rule) $\sigma' = diag(0, 1, \cdots)$

$$\Rightarrow \frac{\partial L}{\partial V} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial V} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial V}$$

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U}$$

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

其中 $\frac{\partial h_j}{\partial h_{j-1}} = \sigma' W$　故 $\frac{\partial h_t}{\partial h_k} = (\sigma' W)^{t-k}$

↳ 相当于对 W 某些行置为0（乘对角矩阵）

代入

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} (\sigma' W)^{t-k} \frac{\partial h_k}{\partial W}$$

若 $\sigma' W < 1$　当 $t-k$ 过大时　$\sigma' W \to 0$ 导致梯度消失

若 $\sigma' W > 0$　当 $t-k$ 过大时　$\sigma' W \to \infty$ 导致梯度爆炸

$y = \sigma(x)$ 其中 $x, y \in \mathbb{R}^n$ 则

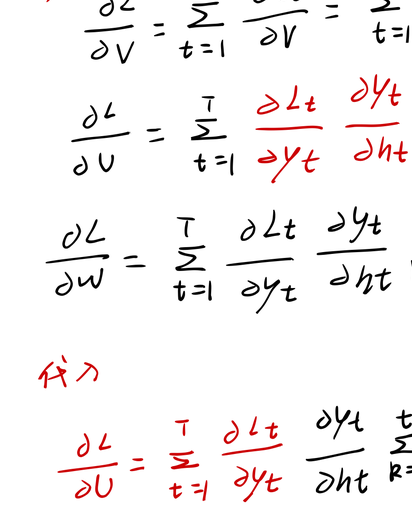
$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sigma\left(\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}\right)$$

$$\frac{\partial y}{\partial x} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_n} \end{pmatrix} = diag(0, 1, 0 \cdots)$$

↙ 一些0和1

$$\begin{cases} \frac{\partial g(u(x))}{\partial x} = \frac{\partial g}{\partial u} \frac{\partial u}{\partial x} & \text{其中 } g, u, x \text{ 都是向量.} \\ (Ax)' = A & A \text{ 是矩阵, } x \text{ 是向量.} \end{cases}$$

人人重实现代码解释



(batch-size × T) 实际的网络输入

定义的 RNN 网络只有三层即 输入层，隐藏层，输出层

上图所示是 RNN 的一个的时间步的结构

上述的 x 在我的设计里对应的是一个字母的 one-hot 编码

因此 RNN input-size 为 28，output-size 也是 28 从而输出也对应一个字母

开始训练时 采用批量训练

```
for x,y in train-iter:
    y_hat = net(x)
```

因此初始化 h_0 为一个大小为 (batch-size, hidden-size) 的全零矩阵

这样 $(h_1, h_2, \cdots, h_T)$ 就得到一个形状为 (时间步, batch-size, hidden-size) 的张量

最后输出的 $(y_1, y_2, \cdots, y_T)$ 的形状为 (时间步, batch-size, 28)　(28是词表大小)

而这里的 y 的形状是 (batch-size, 时间步)

因此计算 loss 时需要转置

即：

```
y = y.T.reshape(-1)
y_hat = y_hat.reshape(-1, 28)
l = loss(y_hat, y)
```