No.

Date . . .

chap-6 循环神经网络

循环神经网络 RNN

递归神经网络

图网络

No.

Date.

6.1 给网络增加记忆能力.

短期记忆能力

6.1.1 延时神经网络. TDNN

时间维度上共享权重.

延时单元

$$\vec{h}_t^{(l+1)} = f(\vec{h}_t^{(l)}, \cdots, \vec{h}_{t-p+1}^{(l)})$$

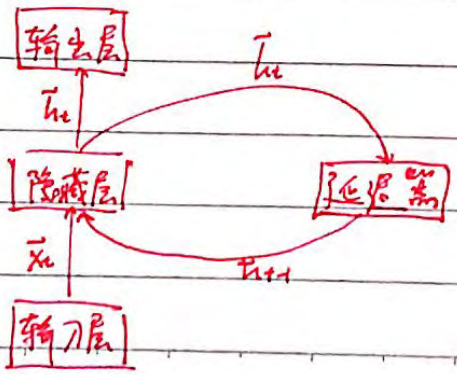
6.1.2 有外部输入的非线性自回归模型.

自回归模型. AR

6.1.3 循环神经网络

$\vec{x}_{1:T} = (\vec{x}_1, \vec{x}_2, \cdots, x_T)$ 输入序列

$\vec{h}_t = f(\vec{h}_{t-1}, \vec{x}_t)$ 状态值



No.

Date. . .

6.2 简单循环网络

SRN

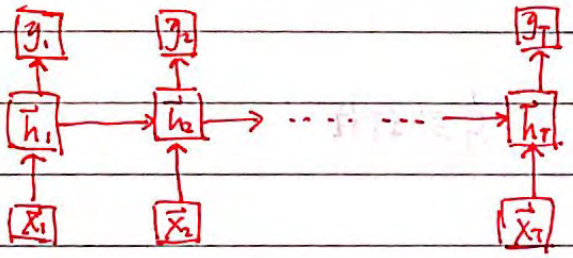
$$\vec{z}_t = U\vec{h}_{t-1} + W\vec{x}_t + \vec{b}$$

$$\vec{h}_t = f(\vec{z}_t)$$

U 状态到状态权重矩阵

W 状态-输入权重矩阵

$\vec{z}_t$ 隐藏层的净输入



6.2.1 循环神经网络的计算能力

$$\vec{h}_t = f(U\vec{h}_{t-1} + W\vec{x}_t + \vec{b}) \quad (6.8)$$

$$\vec{y}_t = V\vec{h}_t \quad (6.9)$$

⇒ 通用近似定理

⇒ 图灵完备

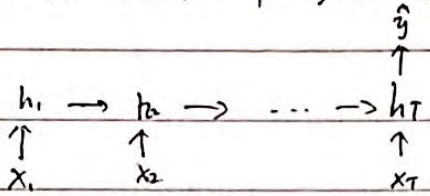
No.

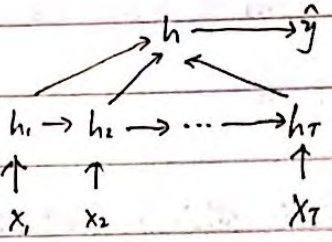
Date.

6.3 应用到机器学习

6.3.1 序列到类别模式 多表

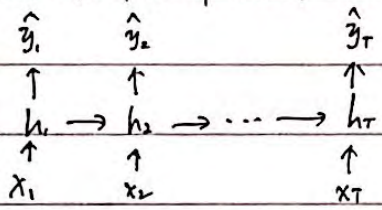
将 $\vec{h}_T$ 看做整个序列的表征表示.





$$\hat{y} = g(\vec{h}_T)$$

6.3.2 同步的序列到序列模式



序列标注

6.3 异步的序列到序列模式

编码器-解码器模型.

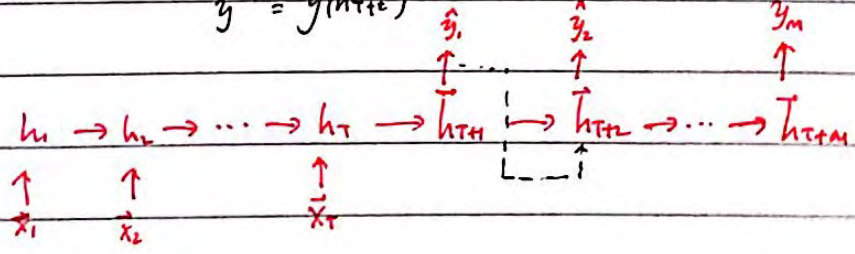
$$\vec{x}_{1:T} = (\vec{x}_1, \cdots, \vec{x}_T)$$

$$y_{1:M} = (y_1, \cdots, y_M)$$

$$\vec{h}_t = f_1(\vec{h}_{t-1}, \vec{x}_t)$$

$$\vec{h}_{T+t} = f_2(\vec{h}_{T+t-1}, \hat{y}_{t-1})$$

$$\hat{y} = g(\vec{h}_{T+t})$$



No.

Date. . .

6.4 参数学习

$L_t = L(y_t, g(\vec{h}_t))$ t时刻的损失函数

$g(\vec{h}_t)$ t时刻的输出

$L = \sum_t L_t$ 整个序列上的损失函数

$\frac{\partial L}{\partial U} = \sum_t \frac{\partial L_t}{\partial U}$ ⟵ $\vec{h}_t = f(U\vec{h}_{t-1} + W\vec{x}_t + b)$

随时间反向传播 BPTT

实时循环学习法 RTRL

6.4.1 随时间反向传播算法

BPTT将RNN看成一个展开的多层前馈网络，第一层对应每个时刻.

在展开的网络中所有层的参数是共享的，因此参数的真实梯度是将所有展开层的参数梯度之和.

⇒ 计算偏导数 $\frac{\partial L_t}{\partial U}$

$\vec{z}_k = U\vec{h}_{k-1} + W\vec{x}_k + \vec{b}$ 时刻k时的净输入.

$$\frac{\partial L_t}{\partial U_{ij}} = \sum_k tr\left(\left(\frac{\partial L_t}{\partial \vec{z}_k}\right)^T \frac{\partial^+ \vec{z}_k}{\partial U_{ij}}\right)$$

$$= \sum_k \left(\frac{\partial^+ \vec{z}_k}{\partial U_{ij}}\right)^T \frac{\partial L_t}{\partial \vec{z}_k}$$ ← why (6.30)

No.

Date.

$\frac{\partial^+ \vec{z}_k}{\partial U_{ij}}$ 表示直接导数

$$\frac{\partial^+ \vec{z}_k}{\partial U_{ij}} = \begin{bmatrix} 0 \\ [\vec{h}_{k-1}]_j \\ 0 \end{bmatrix} \triangleq \mathbb{I}_i([\vec{h}_{k-1}]_j) \tag{6.31}$$

$\delta_{t,k} = \frac{\partial L_t}{\partial \vec{z}_k}$ 定义为 t 时刻的损失对 k 时刻隐藏神经层的净输入 $\vec{z}_k$ 的导数

$$\delta_{t,k} = \frac{\partial L_t}{\partial \vec{z}_k}$$

$$= \frac{\partial \vec{h}_k}{\partial \vec{z}_k} \cdot \frac{\partial \vec{z}_{k+1}}{\partial \vec{h}_k} \cdot \frac{\partial L_t}{\partial \vec{z}_{k+1}}$$

$$= \mathrm{diag}(f'(\vec{z}_k)) \cdot U^T \cdot \delta_{t,k+1} \tag{6.36}$$

$$\vec{z}_t = U\vec{h}_{t-1} + W\vec{x}_t + \vec{b}$$

$$\vec{h}_t = f(\vec{z}_t) \tag{6.32}$$

将式 (6.36) $\delta_{t,k}$ 和 (6.31) $\frac{\partial^+ \vec{z}_k}{\partial U_{ij}}$ 代入 (6.30) 得到 $\frac{\partial L_t}{\partial U_{ij}}$

$$\frac{\partial L_t}{\partial U_{ij}} = \sum_{k=1}^{t} [\delta_{t,k}]_i [\vec{h}_{k-1}]_j$$

$$= \sum_{k=1}^{t} \delta_{t,k} \vec{h}_{k-1}^T$$

No.

Date. . .

⇒ 参数梯度

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \sum_{h=1}^{t} \delta_{t,h} \vec{h}_{h-1}^{T}$$

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \sum_{h=1}^{t} \delta_{t,h} \vec{x}_{t}^{T}$$

$$\frac{\partial L}{\partial b} = \sum_{t=1}^{T} \sum_{h=1}^{t} \delta_{t,h}$$

⇒ 计算复杂度

一个完整的前向计算和反向计算.

6.4.2 实时循环学习算法

$$\frac{\partial L_t}{\partial U_{ij}} = \left(\frac{\partial \vec{h}_t}{\partial U_{ij}}\right)^{T} \cdot \frac{\partial L_t}{\partial \vec{h}_t}$$

⇒ 两种算法比较

BPTT: 一般网络输出维度 < 输入维度, BPTT算法更适合. 时间复杂高.

RTRL: 在限序列, 无限序列.

No.

Date.

6.5 长期依赖问题

在BPTT算法中，将公式(6.34)展开，得到

$$\delta_{t,k} = \prod_{\tau=k}^{t-1} \left(\mathrm{diag}(f'(z_\tau))\, U^T\right) \cdot \delta_{t,t}$$

令 $\gamma \cong \| \mathrm{diag}(f'(z_\tau))\, U^T \|$.

$$\delta_{t,k} = \prod_{\tau=k}^{t-1} \gamma^{t-k} \delta_{t,t}$$

$\gamma > 1$，梯度爆炸

$\gamma < 1$，梯度消失。无法影响间隔远处的信息。

长期依赖问题

6.5.1 改进方案

梯度爆炸

权重衰减，L_1或L_2范数的正则化，使$\gamma < 1$

梯度截断

梯度消失

改变模型

让$U = I$，$f'(z_i) = 1$，即

$\vec{h}_t = \vec{h}_{t-1} + g(\vec{x}_t; \theta)$　丢失了非线性性质

增加记忆容量

$$\vec{h}_t = \vec{h}_{t-1} + g(\vec{x}_t, \vec{h}_{t-1}; \theta)$$

No.

Date.

6.6 基于门控的循环神经网络.

1. 增加一些额外的存储单元，外部记忆

2. 进行选择性遗忘，选择性更新

6.6.1 长短期记忆网络. (LSTM)

新的内部状态 c_t (记忆单元)

c_t 专门进行线性的循环信息传递，同

同时非线性输出信息给隐藏层的外部状态 $\vec{h}_t$

$$\vec{c}_t = \vec{f}_t \odot \vec{c}_{t-1} + \vec{i}_t \odot \tilde{c}_t \quad \text{线性}$$

$$\vec{h}_t = \vec{o}_t \odot \tanh(c_t) \quad \text{非线性}$$

$\tilde{c}_t$ 候选状态.

$$\tilde{c}_t = \tanh(W_c \vec{x}_t + U_c \vec{h}_{t-1} + \vec{b}_c)$$

门机制

取值在 (0,1) 之间.

遗忘门 $\vec{f}_t$ c_{t-1} 保留 ~~保留~~ 多少信息

输入门 $\vec{i}_t$ $\tilde{c}_t$ 有少信息要保存.

输出门 $\vec{o}_t$ $\vec{c}_t$ 有多少信息作为 $\vec{h}_t$

$\vec{f}_t = 0$, $\vec{i}_t = 1$ 时 历史信息被清空

$\vec{f}_t = 1$, $\vec{i}_t = 0$ 时. 复制历史信息

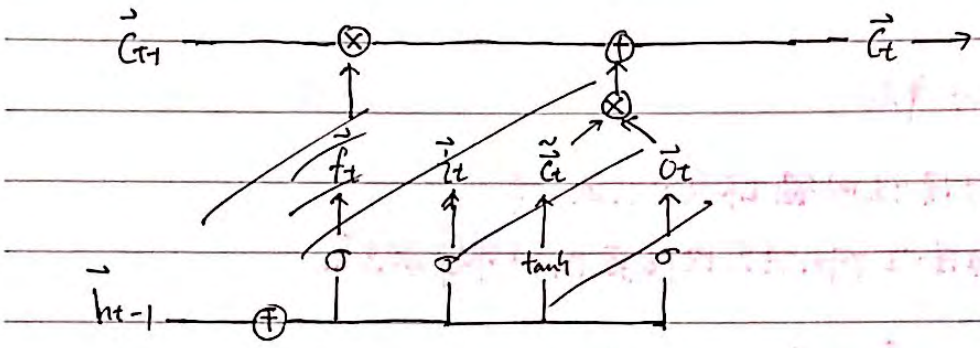
No.

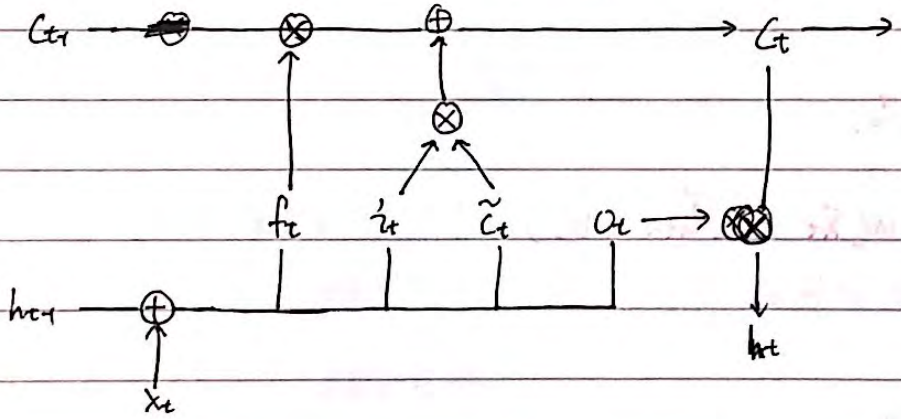
Date.

① $\vec{h}_{t-1}$ 和 $\vec{x}_t$ $\Rightarrow \vec{f}_t, \vec{i}_t, \tilde{\vec{C}}_t, \vec{O}_t$ (4个 State)

② $\vec{f}_t$ 和 $\vec{i}_t$, $\tilde{\vec{C}}_t$ $\Rightarrow \vec{C}_t$

③ $\vec{O}_t$ $\Rightarrow \vec{h}_t$





$$\begin{bmatrix} \tilde{\vec{C}}_t \\ \vec{O}_t \\ \vec{i}_t \\ \vec{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} \vec{x}_t \\ \vec{h}_{t-1} \end{bmatrix} + b \right)$$

$$\vec{C}_t = \vec{f}_t \odot \vec{C}_{t-1} + \vec{i}_t \odot \tilde{\vec{C}}_t$$

$$\vec{h}_t = \vec{O}_t \odot \tanh(\vec{C}_t)$$

No.

Date.

=) 记忆

短期记忆 h_t

长的短期记忆 C_t

6.6.2 LSTM网络的各种变化

无遗忘门的 LSTM 网络

peephole 连接

耦合输入门和遗忘门

6.6.3 门控循环单元网络 (GRU)

输入门
遗忘门 → 更新门

$\tilde{h}_t$ 和 $\tilde{h}_{t-1}$ 之间引入线性依赖关系

h_{t-1} → ⊗

→ ⊕

No.

Date.

~~救世主是由先知和设计师的[illegible]~~

[illegible]

救世主 $\tilde{h}_t = \tanh(W_c \vec{x}_t + U(r_t \odot h_{t-1}) + b_h)$

↑ 救世主 ↑ 现状 ↑ 设计师 ↑ [illegible]

设计师控制上一代救世主有多少信息传至新代

设计师 $r_t = \sigma(W_r \vec{x}_t + U_r h_{t-1} + b_r)$

↑ 现状 ↑ 上一代救世主

救世主的结果 $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$

↑ 先知 ↑ 上一代救世主 ↑ 先知影响 ↑ 救世主

先知影响上代救世主

以先知影响新救世主

先知和设计师前后一致

先知 $z_t = \sigma(W_z \vec{x}_t + U_z h_{t-1} + b_z)$

↑ 现状 ↑ 上一代救世主.

$r_t = \sigma(W_r \vec{x}_t + U_r h_{t-1} + b_r)$

$z_t = \sigma(W_z \vec{x}_t + U_z h_{t-1} + b_z)$

$\tilde{h}_t = \tanh(W_c \vec{x}_t + U(r_t \odot h_{t-1}) + b_h)$

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

重置门 $r_t = 0$ 忘历史信息

$= 1$ 和历史相关. SRN

更新门 $z_t = 0, r=1$ SRN

No.

Date.

6.7 深层循环神经网络.

增加隐藏状态到输出 $\vec{h}_t \to \vec{y}_t$

转为到隐藏状态 $\vec{x}_t \to \vec{h}_t$

6.7.1 堆叠循环神经网络. (SRNN)

$$\vec{h}_t^{(l)} = f(U^{(l)} \vec{h}_{t-1}^{(l)} + W^{(l)} h_t^{(l-1)} + b^{(l)})$$

6.7.2 双向循环神经网络. (Bi-RNN)

$$h_t^{(1)} = f(U^{(1)} \vec{h}_{t-1}^{(1)} + W^{(1)} x_t + b^{(1)})$$

$$h_t^{(2)} = f(U^{(2)} h_{t+1}^{(2)} + W^{(2)} x_t + b^{(2)})$$

$$h_t = h_t^{(1)} \oplus h_t^{(2)}$$

No.

Date. . .

6.3 扩展到图结构.

每个节点都收到其之节点的信息.

6.3.1 递归神经网络 (RecNN)