Recurrent Neural Network (RNN)

vanilla RNN, LSTM and GRU

李凌璇

Center for Intelligence of Science and Technology(CIST)
School of Computer Science

2019-3-6

目录

序列问题

形式化描述

- 输入：$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$
- 输出：$\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$

1960 前

- 序列问题由 Markov 链、HMM 解决

HMM 存在的问题

- Viterb 算法：时间复杂度和空间复杂度均为 $O(|S|^2)$
- Markov 链假设：当前时刻的状态只依赖于上一时刻的状态

早期 RNN

1982 年

- Hopfield 引入了具有识别能力的 RNN

1986 年

- Jodan 首先将 RNN 应用于监督学习上
- $h_t = \sigma_h\left(W_h x_t + U_h y_{t-1}\right) + b_h$
- $y_t = \sigma_y\left(W_y h_t + b_y\right)$

1990 年

- Elman 提出了更简单的 RNN 网络结构
- $h_t = \sigma_h\left(W_h x_t + U_h h_{t-1}\right) + b_h$
- $y_t = \sigma_y\left(W_y h_t + b_y\right)$

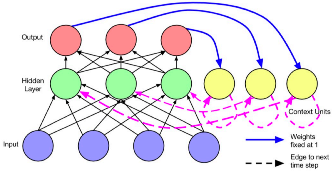


Figure: Jordan RNN 结构图

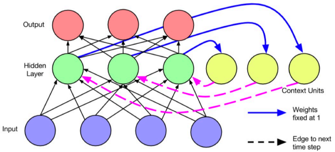


Figure: Elman RNN 结构图

vanilla RNN

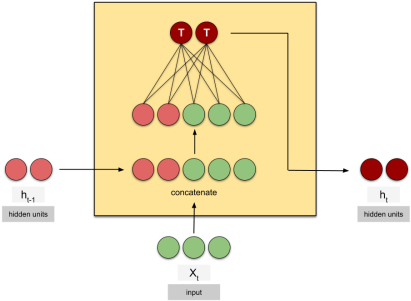


Figure: RNN 单元结构图

- 整体表达式：

$$h_t = f_W\left(h_{t-1}, x_t\right)$$

- 展开为：

$$\begin{aligned} h_t &= \tanh\left(W_h\left[h_{t-1}, x_t\right] + b_h\right. \\ &= \tanh\left(W_{hh}h_{t-1} + W_{xh}x_t + b_h\right) \end{aligned}$$

- T 步循环：

$$\begin{aligned} h_t = &\tanh(W_{hh}(\tanh(\cdots\tanh(W_{hh} \\ &h_{t-T-1} + W_{xh}x_{t-T} + b_h)\cdots)) \end{aligned}$$

- RNN 具有对历史信息的记忆功能

vanilla RNN

梯度反传 (BPTT)

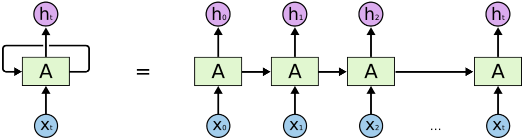


Figure: RNN 展开示意图

- 对每个循环单元而言
  $h_t = \tanh\left(W_{hh}h_{t-1} + W_{xh}x_t + b_h\right)$
  $\hat{y}_t = \mathrm{softmax}\left(Vh_t\right)$
- 误差函数
  $E_t\left(y_t, \hat{y}_t\right) = -y_t \log \hat{y}_t$
  $E(y, \hat{y}) = \sum_t E_t\left(y_t, \hat{y}_t\right) = -\sum_t y_t \log \hat{y}_t$

梯度反传

- 在 t 时刻:

$$\frac{\partial E_t}{\partial net_t} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial net_t} = E'(\hat{y}_t)\,\sigma'(net_t)$$

- 对 V 求导:

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial net_t}\frac{\partial net_t}{\partial V} = \frac{\partial E_t}{\partial net_t} h_t$$

- 对 W_{xh} 求导:

$$\frac{\partial E}{\partial W_{xh}} = \frac{\partial E}{\partial h_t}\frac{\partial h_t}{\partial W_{xh}} = \frac{\partial E}{\partial h_t} x_t$$

- 对 W_{hh} 求导:

$$\frac{\partial E_t}{\partial W_{hh}} = \frac{\partial E_t}{\partial net_t}\frac{\partial net_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}}$$

- $h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$,所以 h_{t-1} 依赖于 h_{t-2},$\cdots$,h_1 依赖于 h_0。
- 所以:

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{k=0}^{t}\frac{\partial E_t}{\partial net_t}\frac{\partial net_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W_{hh}}$$

- 总误差对 W_{hh} 求导:

$$\frac{\partial E}{\partial W_{hh}} = \sum_{t=0}^{T}\frac{\partial E_t}{\partial W_{hh}}$$

梯度消失/爆炸

梯度消失/爆炸问题

- 由导数连乘导致

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial net_t} \frac{\partial net_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_{hh}}$$

而 $\frac{\partial h_t}{\partial h_k}$ 依据导数的链式法则，实质上是 $\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$。所以

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial net_t} \frac{\partial net_t}{\partial h_t} \left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W_{hh}}$$

梯度消失/爆炸

梯度消失/爆炸问题

- $\frac{\partial h_j}{\partial h_{j-1}}$ 是向量对向量的导数，即 Jacobian 矩阵

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- $$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \left\| W^T \right\| \left\| \mathrm{diag}\left[tanh'\left(h_{j-1}\right)\right]\right\| \leq \beta_W \beta_h$$

  其中 β_W 和 β_h 分别为 W 和 h 的 L_2 范数上界

- $$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \left(\beta_W \beta_h\right)^{t-k}$$

梯度消失/爆炸

梯度消失/爆炸问题

- $(\beta_W \beta_h)^{t-k}$ 为指数形式，会快速增长或快速下降
- 梯度爆炸导致训练困难，但可以通过 gradient clipping 解决
- 梯度消失导致模型只能学习到短时依赖，可以通过 Relu 激活函数稍加改进

$$\text{new weight} = \text{weight- learning rate} \times \text{gradient}$$

LSTM

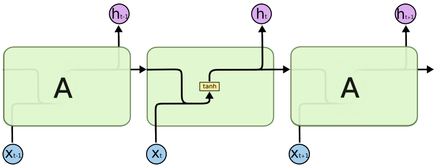


Figure: 隐层为单层的 vanilla RNN

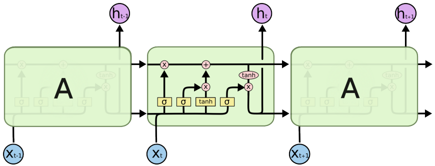


Figure: 链式的 LSTM

vanilla RNN

- 只能学习到短时依赖

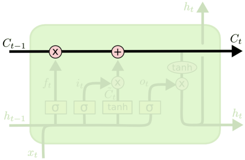
LSTM(Long-Short)

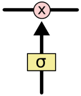
- 1997 年 Hochreiter 以及 Schmidhuber 提出
- 修改 RNN 单元结构，引入遗忘门、输入门、输出门
- 既可学习短时依赖又可学习长时依赖

重点概念

记忆单元 (cell state)

- LSTM 的核心结构
- 保存网络记忆的信息, 类似 RNN 中的 h_t
- 遗忘门和输入门所修改的对象，输出门的部分输入

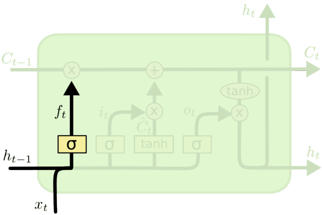




门 (gate)

- LSTM 的核心操作
- 由 sigmoid 和 pointwise multiplication 操作组成
- 决定信息可否通过

遗忘门

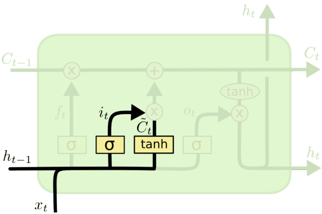


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

遗忘门 (forget gate)

- 决定记忆单元中哪些信息需要被遗忘
- 经过 sigmoid 后，向量 f_t 的各元素值被固定在 $0 \sim 1$ 之间。越接近 0，代表遗忘；越接近 1，代表记忆
- $C_{t-1} * f_t$ 得到的则是网络经过遗忘门后记住的信息

输入门

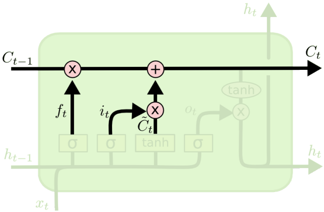


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

输入门 (input gate)

- 决定哪些新的信息需要被记忆
- 第一步，依据 h_{t-1} 和 x_t 决定哪些信息需要保存至状态单元中
- 第二步，依据 h_{t-1} 和 x_t 构建候选向量 $\tilde{C}_t$，得到当前输入产生的新信息

记忆更新

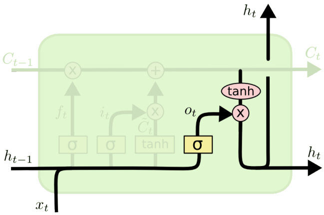


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

记忆单元的更新

- 依据遗忘门和输入门更新记忆单元 (cell state)
- 经过遗忘门后记忆单元内的信息为 $f_t * C_{t-1}$
- 经过输入门后需要增加的信息为 $i_t * \tilde{C}_t$

输出门



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

输出门 (output gate)

- 决定输出的信息是什么
- 输出的信息基于记忆单元，但需要依据输入再一次对其进行过滤
- 第一步，依据 h_{t-1} 和 x_t，决定细胞单元中哪些信息被输出
- 第二步，利用 tanh，把记忆单元中现有的信息其值压缩在 $-1 \sim 1$ 之间，再乘以输出门 o_t，得到最终的输出信息

LSTM

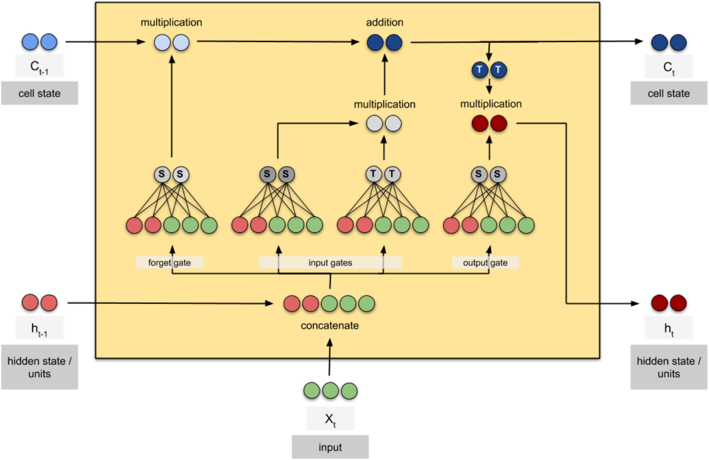


Figure: LSTM 单元结构图

LSTM

梯度消失/爆炸

LSTM 为什么可以解决梯度消失/爆炸问题?

- 回顾：vanilla RNN 中导致梯度消失/爆炸的原因——导数的连乘
- 其实 LSTM 的梯度反传中也有连乘项 $\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$
- 在 vanilla RNN 中:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=1}^{t-k} Wtanh'\left(Wh_{t-j}\right) = W^{t-k} \prod_{j=1}^{t-k} tanh'\left(Wh_{t-j}\right)$$

- 在 LSTM 中:

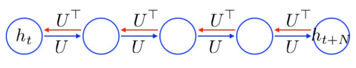
$$\frac{\partial h_t}{\partial h_k} = \prod_{j=1}^{t-k} tanh\left(V_{k+j}\right)$$

- 结论：LSTM 也存在梯度消失/爆炸，但是梯度减小、增大的幅度更小
- 这方面的讨论可见文献 Bayer, Justin Simon. Learning Sequence Representations.

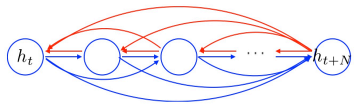
梯度消失/爆炸

另一种解释

- BPTT:



- 在 vanilla RNN 中，梯度严格的按照所有的中间节点流动的
- 为了避免梯度消失/爆炸，在梯度反传中创造自适应的短链接



- 通过门创造自适应的短链接，如 LSTM 中信息更新

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

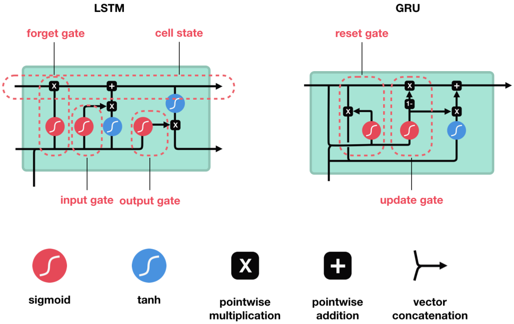
GRU

LSTM 的缺点

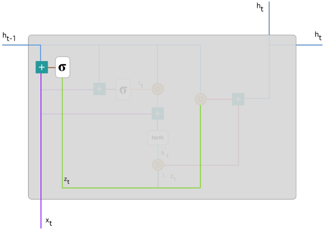
- 参数多

GRU(Gated Recurrent Unit)

- 2014 年 Cho et al 提出
- 可达到与 LSTM 相当的效果
- 但训练参数少，更容易训练
- 相比 LSTM 而言，少了一个门，由重置门和更新门组成
- 没有 LSTM 中的记忆单元 (cell state)，由隐层状态传递信息



更新门

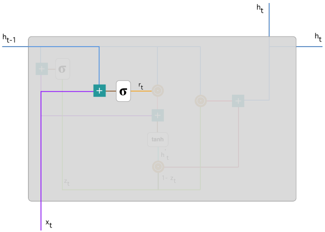


$$z_t = \sigma \left(W_z \cdot [h_{t-1}, x_t] \right)$$

更新门 (update gate)

- 作用相当于 LSTM 的遗忘门和输入门
- 决定过去时刻的信息 h_{t-1} 和新的信息有多少可以传递至新的时刻
- 经过 sigmoid 后其取值在 $0 \sim 1$ 之间

重置门

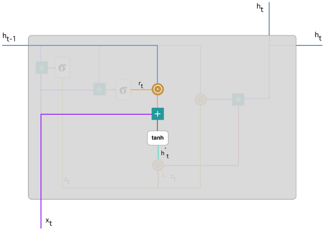


$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

重置门 (reset gate)

- 决定过去时刻的信息 h_{t-1} 有多少需要被遗忘
- 经过 sigmoid 后其取值也在 $0 \sim 1$ 之间

记忆上下文

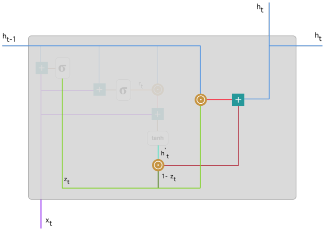


$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

当前记忆上下文 (current memory content)

- 利用重置门构建当前的记忆上下文
- $r_t * h_{t-1}$ 代表丢弃了一部分信息的历史信息
- 利用输入 x_t 在历史信息中又添加了新的信息
- 两部分共同构成当前时刻的记忆上下文

最终记忆



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

当前时刻最终的记忆 (final memory at current time step)

- 最终的记忆信息由更新门决定
- $z_t * h_{t-1}$ 得到历史信息，z_t 代表对历史信息的选择性遗忘
- $(1 - z_t) * \tilde{h}_t$ 得到新信息，$1 - z_t$ 代表对当前记忆上下文中的信息选择性记忆。若 z_t 越接近 1，则历史信息所占比重越大，遗忘越少。
- 两部分共同构成最终的记忆信息 h_t

GRU

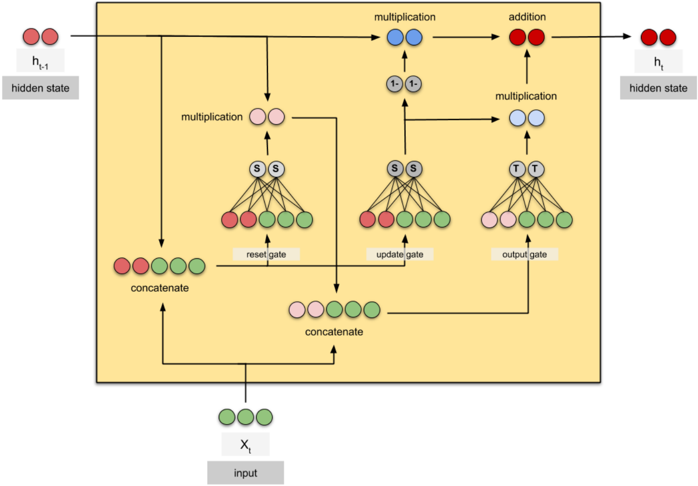


Figure: GRU 单元结构图

GRU

梯度消失/爆炸

GRU 如何解决梯度消失/爆炸问题的?

- 原理与 LSTM 相同
- 过长的依赖依然学习不到
- 通过门创造自适应的短链接:

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