

19 循环神经网络

概要

- 问题的根源: 为什么传统神经网络不擅长处理序列?
- RNN的诞生: 引入“隐状态”，实现对历史的记忆。
- 核心应用: 使用RNN构建语言模型。
- 严峻的挑战: 长期依赖与梯度消失/爆炸问题。
- 优雅的方案: 门控机制——GRU与LSTM的崛起。
 - 门控循环单元 (GRU)
 - 长短期记忆网络 (LSTM)

循环神经网络

为什么需要一种新的网络结构？

- 问题
 - 无隐状态的神经网络 (如 MLP)
 - 给定小批量样本 $X \in \mathbb{R}^{n \times d}$ ，隐藏层输出为
$$H = \phi(XW_{xh} + b_h)$$
 - 核心假设: 所有样本 (n 个) 是独立同分布的。模型一次性处理所有输入，样本之间无交互
 - 局限性: 无法处理具有内在顺序和依赖关系的数据，如文本。例如，预测“世界”需要先看到“你好”
- 需求
 - 序列处理的需求
 - 可变长度: 模型能处理任意长度的句子
 - 顺序依赖: 模型必须理解词语的顺序
 - 历史记忆: 模型需要一个“记忆”机制来存储之前看到的信息
 - 我们需要一个能够循环处理并维护一个动态记忆模型

无隐状态的神经网络

- 单隐藏层的多层感知机。设隐藏层的激活函数为 ϕ , 给定一个小批量样本 $\mathbf{X} \in \mathbb{R}^{n \times d}$, 其中批量大小为 n , 输入维度为 d , 则隐藏层的输出 $\mathbf{H} \in \mathbb{R}^{n \times h}$

$$\mathbf{H} = \phi(\mathbf{X}\mathbf{W}_{xh} + \mathbf{b}_h)$$

- 隐藏层权重参数 $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$, 偏置参数 $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$, 隐藏单元的数目为 h
- 接下来, 将隐藏变量 $\mathbf{H}$ 用作输出层的输入。输出层
 - $\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q$,
 - 其中, $\mathbf{O} \in \mathbb{R}^{n \times q}$ 是输出变量, $\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ 是权重参数, $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ 是输出层的偏置参数。如分类问题, 可用 $\text{softmax}(\mathbf{O})$ 来计算输出类别的概率分布
- 【注】全部 n 个样本都输入

引入“隐状态” — RNN的记忆核心

➤ 在时间步 t 有小批量输入 $\mathbf{X}_t \in \mathbb{R}^{n \times d}$

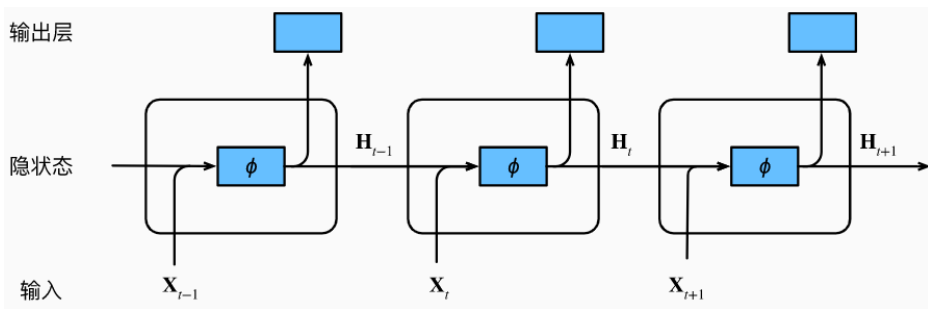
➤ 对于 n 个序列样本的小批量, $\mathbf{X}_t$ 的每一行对应于来自该序列的时间步 t 处的一个样本

➤ 时间步 t 隐状态 $\mathbf{H}_t \in \mathbb{R}^{n \times h}$, 通过循环计算来更新隐状态

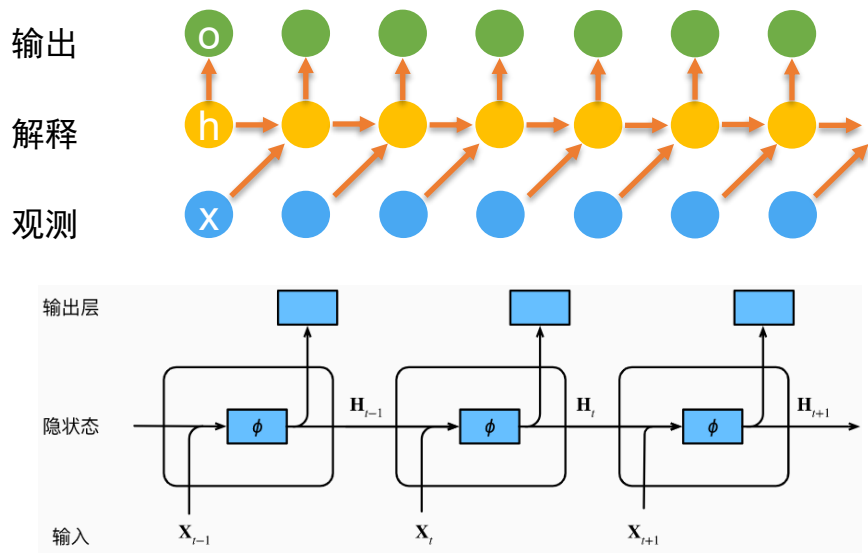
$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h)$$

➤ H_t 总结了从序列开始到时间步 t 的所有相关信息

➤ 权重 W_{xh} 和 W_{hh} 在所有时间步中是共享的, 这让模型学习通用的序列模式



循环神经网络

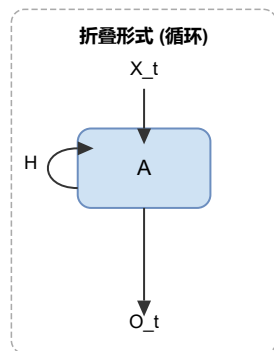


- 隐含状态更新 $\mathbf{H}_t = \phi(\mathbf{W}_{hx}\mathbf{X}_t + \mathbf{W}_{hh}\mathbf{H}_{t-1} + \mathbf{b}_h)$
- 输出更新 $\mathbf{O}_t = \mathbf{W}_{hq}\mathbf{H}_t + \mathbf{b}_q$
- 计算损失

RNN的折叠与展开视图

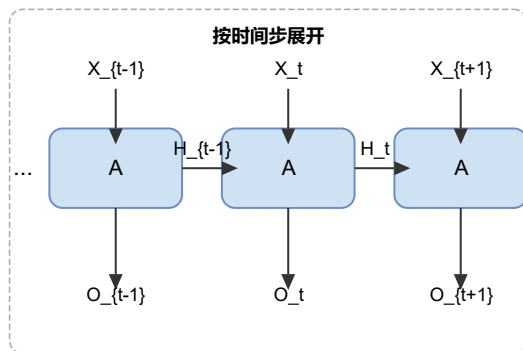
➤ 折叠视图 (循环表示)

- 一个RNN单元(A)，一个指向自身的循环箭头，表示隐状态在时间步之间的传递。
- 这种表示简洁地体现了其“循环”的本质。



➤ 按时间展开视图 (前馈网络表示)

- 将循环展开成一个链式结构，清晰地展示了信息在时间序列上的流动。
- 在时间步 t ，单元A接收当前输入 X_t 和来自上一步的隐状态 H_{t-1} ，计算出新的隐状态 H_t 并传递给下一步。



关键: 所有标记为“A”的单元都使用**完全相同**的权重 (W_{xh}, W_{hh})，这是权重共享的核心体现。

应用与挑战

基于循环神经网络的字符级语言模型

➤ 根据过去的和当前的词元预测下一个词元

➤ 输入: 你好, 世界!

➤ 标签 (目标输出): 好, 世界! <eos> (原始序列向左移一位)

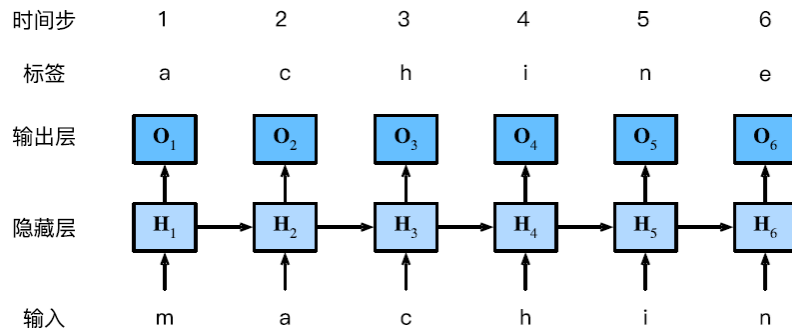
➤ 工作流程

➤ 时间步1: 输入“你”, RNN计算 H_1 , 并预测下一个词的概率分布 (目标是“好”)

➤ 时间步2: 输入“好”, RNN结合 H_1 计算 H_2 , 并预测下一个词的概率分布 (目标是“, ”)

➤ Bengio等首先提出使用神经网络进行语言建模

➤ 字符级语言模型(character-level language model), 将文本词元化为字符/非单词



评估语言模型 - 困惑度(perplexity)

➤ 交叉熵

- 一个观察者，他认为下一个词的概率分布是 Q (模型预测)，而真实分布是 P (真实的下一个词)
- 交叉熵 $H(P, Q)$: 可以想象为“该观察者在看到真实数据时的预期惊异程度”
- $H(P, Q) = -\sum_x P(x) \log Q(x)$ 。对于语言模型，它简化为对真实下一个词的负对数似然

➤ 直观度量: 困惑度 (Perplexity)

- 定义: 困惑度是交叉熵的指数形式

$$\text{Perplexity}(P, Q) = \exp(H(P, Q))$$

- 可以理解为模型在预测下一个词元时，平均面临的“有效选项数量”
- 完美模型: 困惑度为1，表示模型对下一个词的出现非常有信心且完全正确
- 基线模型: 如果模型预测是词表上的均匀分布（即瞎猜），困惑度等于词表大小。任何有用的模型都必须超越这个基线
- 困惑度越低，语言模型性能越好

长期依赖问题：RNN的阿喀琉斯之踵

- RNN的训练方法，截断通过时间反向传播 (BPTT)
 - 将RNN按时间展开成一个深层网络，然后使用反向传播计算梯度
 - 挑战: 梯度需要从最后一个时间步传播到第一个时间步
- 根本问题: 梯度消失与梯度爆炸
 - 在反向传播过程中，梯度计算涉及一长串雅可比矩阵的连乘

$$\frac{\partial L_t}{\partial H_k} = \frac{\partial L_t}{\partial H_t} \prod_{i=k+1}^t \frac{\partial H_i}{\partial H_{i-1}} \propto \frac{\partial L_t}{\partial H_t} (W_{hh}^T)^{t-k}$$

- 梯度消失: 如果权重矩阵 W_{hh} 的范数或激活函数的导数小于1，经过多次连乘后，梯度会呈指数级衰减至0。这使得模型无法学习到相距很远的时间步之间的依赖关系（长期依赖问题）
- 梯度爆炸: 如果权重矩阵范数大于1，梯度会呈指数级增长，导致训练不稳定，更新过大
- 初步解决方案，梯度裁剪 (Gradient Clipping)，有效预防梯度爆炸
 - 如果梯度的范数 g 超过一个预设的阈值 θ ，就将其投影回半径为 θ 的球上

$$\text{if } \|g\| > \theta, \text{ then } g \leftarrow \frac{\theta}{\|g\|} g$$

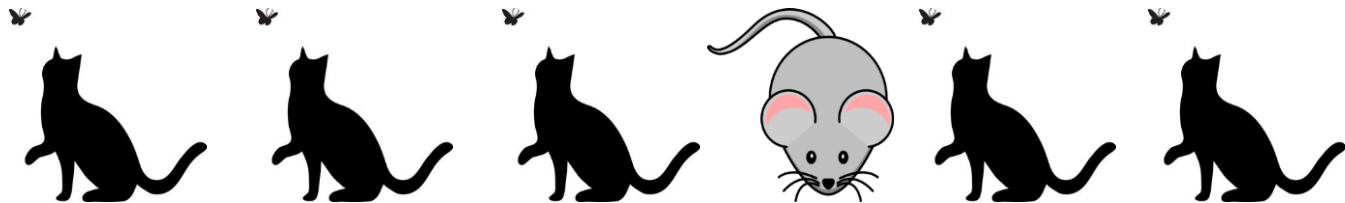
- 梯度裁剪治标不治本，它解决了梯度爆炸，但对更根本的梯度消失问题无能为力

引入“门”来智能控制信息流

- 与其让信息在RNN中被动地、粗暴地混合和衰减，不如设计一种机制，让模型能够主动地、有选择地决定
 - 哪些旧信息需要忘记？
 - 哪些新信息需要学习？
 - 哪些信息应该传递到下一步？
- 实使用Sigmoid激活函数作为“门”。其输出在 $(0, 1)$ 之间，看作控制阀门
 - 0: 完全关闭，信息无法通过
 - 1: 完全打开，信息无损通过
 - $(0, 1)$ 之间: 按比例通过部分信息

在一个序列的注意力

- 并非所有元素都具有同等意义



- 想只记住相关的元素【特别的元素】

- 能关注的机制(更新门)【0为与过去无关，1为只和过去有关】【更新的是候选的内容】

- 能忘记的机制(重置门)【0忘记，1记住】【生成候选内容，不一定会更新】

- 如何实现？

- 对应的控制变量

- 隐状态，以及候选隐状态！

门控循环单元(GRU)

第一个具体网络的RNN

GRU: LSTM的精简高效变体

➤核心组件:

- 更新门 (Update Gate) Z_t : 控制新状态中有多少是旧状态的副本
- 重置门 (Reset Gate) R_t : 控制在计算“候选隐状态”时，要忘记多少过去的隐状态

➤直觉

- 重置门决定了如何将新输入与之前的记忆结合，它控制着“短期记忆”的形成
- 更新门决定了最终的记忆在多大程度上是过去的记忆，在多大程度上是新形成的“短期记忆”，它控制着“长期记忆”的保留

门控循环单元

➤ 重置门 R_t ，新信息如何形成

➤ $R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r)$

➤ 控制着过去的信息 (H_{t-1}) 对新信息的生成 (候选隐状态 $\tilde{H}_t$) 的影响程度

➤ 候选记忆，存放在候选隐状态中，1：记住；0：不记住

➤ $\tilde{H}_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h)$

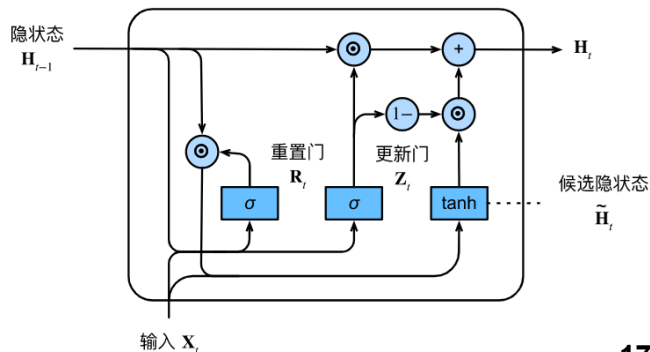
➤ 更新门 Z_t ，决定了新信息是否被采纳

➤ $Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z)$

➤ 控制着最终的隐状态 (H_t) 在多大程度上是对旧状态 (H_{t-1}) 的复制，以及在多大程度上采纳了新生成的信息 ($\tilde{H}_t$)

➤ 1：采纳新的记忆；0：完全用过去隐含状态

➤ $H_t = (1 - Z_t) \odot H_{t-1} + Z_t \odot \tilde{H}_t$



候选隐含状态

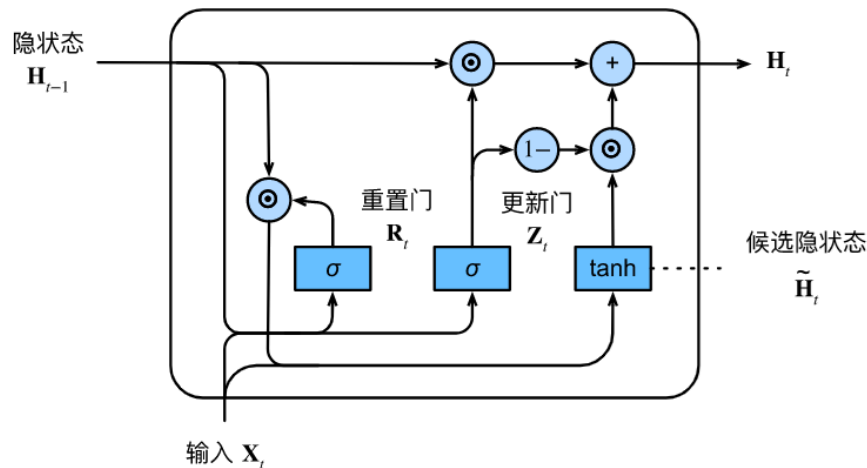
➤ 候选隐状态 (candidate hidden state) $\tilde{\mathbf{H}}_t \in \mathbb{R}^{n \times h}$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h)$$

➤ 符号 $\odot$ 是Hadamard积 (按元素乘积) 运算符

➤ 重置门 $\mathbf{R}_t$ 值越大，隐含状态保留越多

➤ 保留到候选隐含状态中



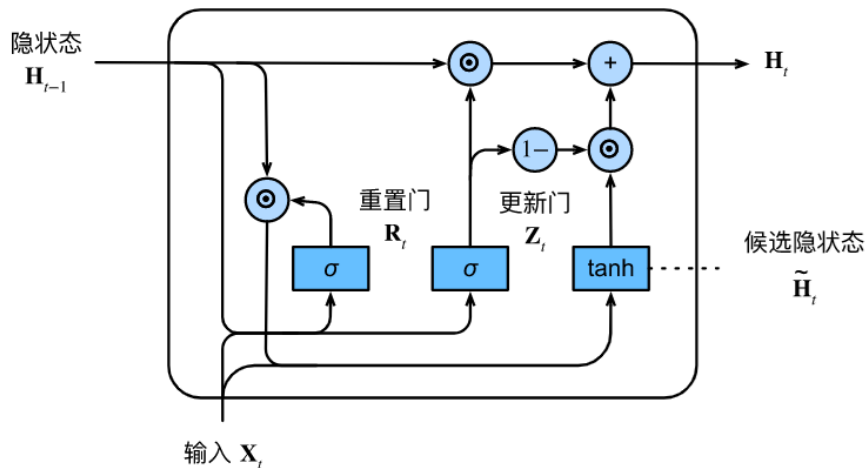
隐含状态

- $\mathbf{H}_{t-1}$ 和 $\tilde{\mathbf{H}}_t$ 之间进行按元素的凸组合更新隐含状态

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$$

- 隐含状态，融合输入信息和隐含状态的候选隐状态，加权平均
 - 权重由更新门控制 $\mathbf{Z}_t$ ，0为与过去隐含状态无关，1为只和过去隐含状态有关
 - 更新门数值越大，隐含状态保留越多
- 隐状态，与其相关状态包含

- 过去隐状态
- 候选隐状态
 - 过去隐状态
 - 输入



门控循环单元(GRU)总结

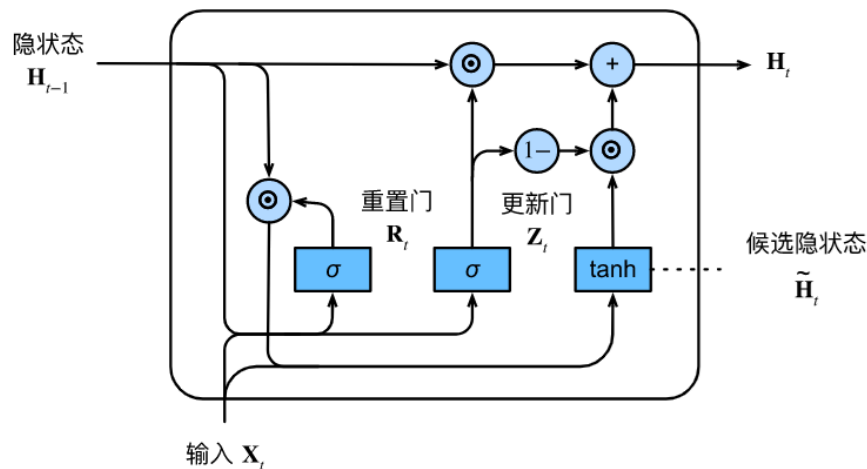
$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r)$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z)$$

➤ $\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h)$

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$$

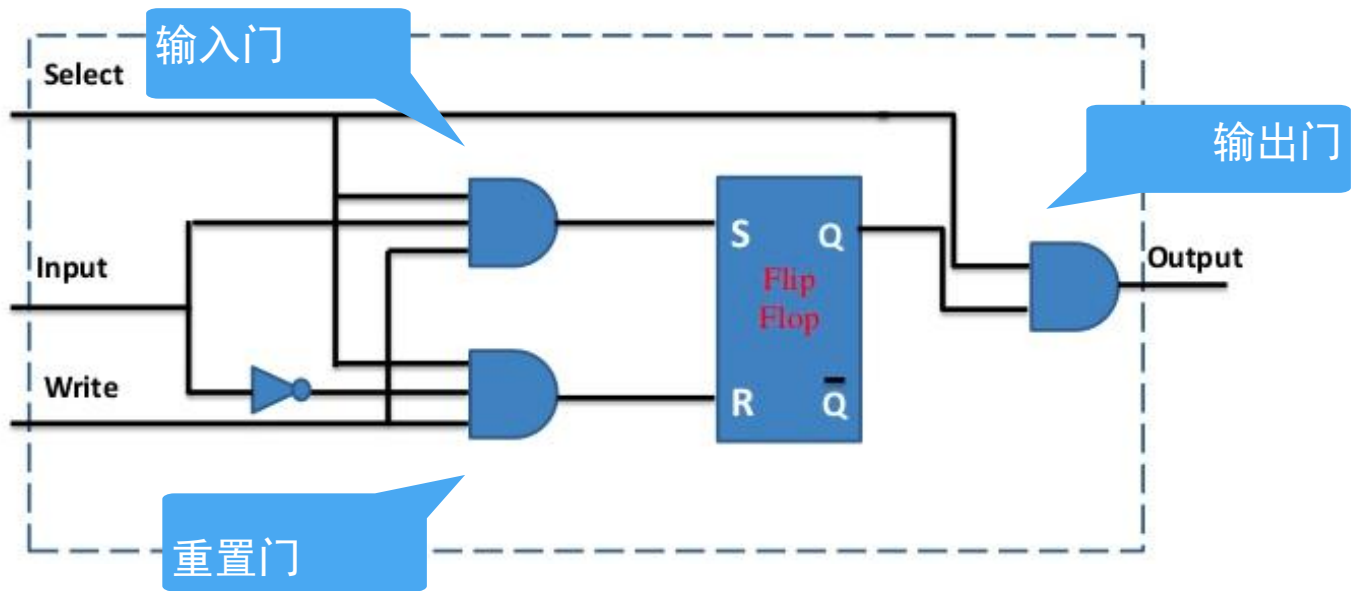
$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$



长短期记忆(LSTM)

电门联想

- ▶ 长短期记忆网络引入记忆元(memory cell), 或简称为单元(cell)



长短期记忆(LSTM)

➤ 忘记门 F_t

➤ 重置单元的内容

➤ 输入门 I_t

➤ 决定是否应忽略输入数据

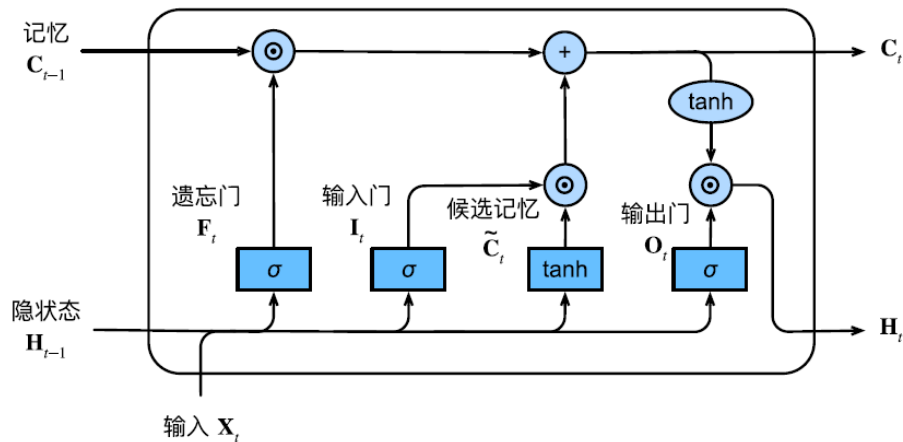
➤ 输出门 O_t

➤ 决定是不是使用隐状态

➤ 机制

➤ 隐状态

➤ 记忆和候选记忆【可理解为没有归一化的隐状态，因此这三个变量有交叉】

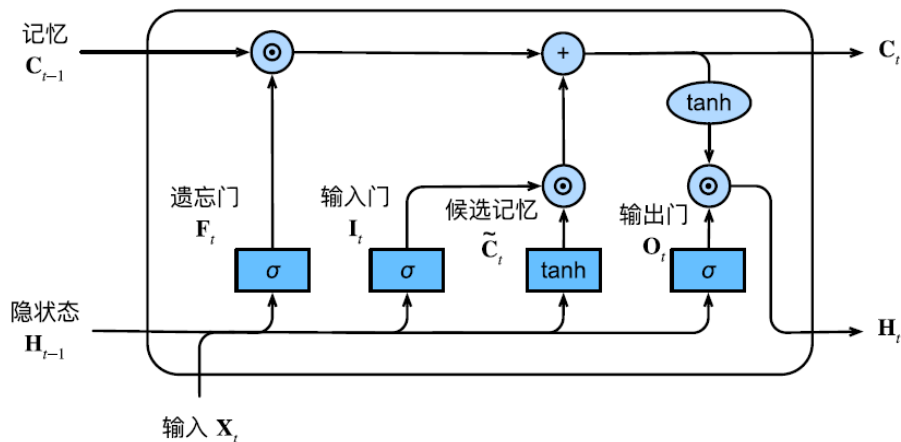


输入门、遗忘门和输出门

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$

$$\text{➤ } F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$

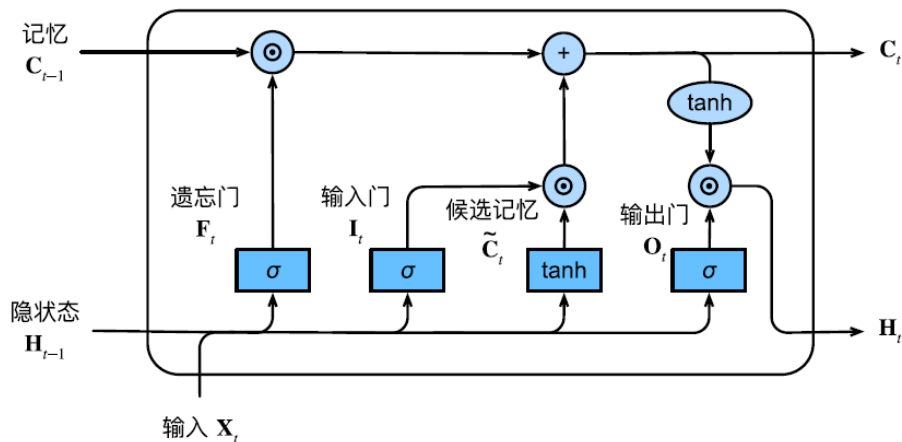


候选记忆元

➤ 候选记忆元 (candidate memory cell) $\tilde{\mathbf{C}}_t \in \mathbb{R}^{n \times h}$ 。函数的值范围为 $(-1,1)$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c),$$

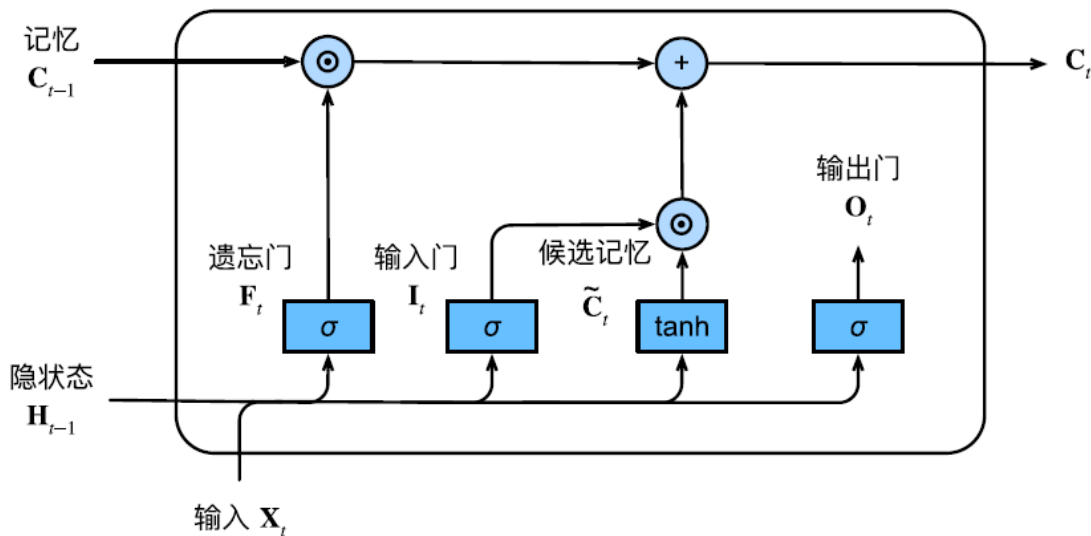
➤ 其中 $\mathbf{W}_{xc} \in \mathbb{R}^{d \times h}$ 和 $\mathbf{W}_{hc} \in \mathbb{R}^{h \times h}$ 是权重参数, $\mathbf{b}_c \in \mathbb{R}^{1 \times h}$ 是偏置参数。



记忆元

- 门控循环单元中有机制控制输入和遗忘 (或跳过)。在LSTM中, 输入门 I_t 控制采用多少来自 $\tilde{C}_t$ 的新数据, 而遗忘门 F_t 控制保留多少过去的记忆元 $C_{t-1} \in \mathbb{R}^{n \times h}$ 的内容:

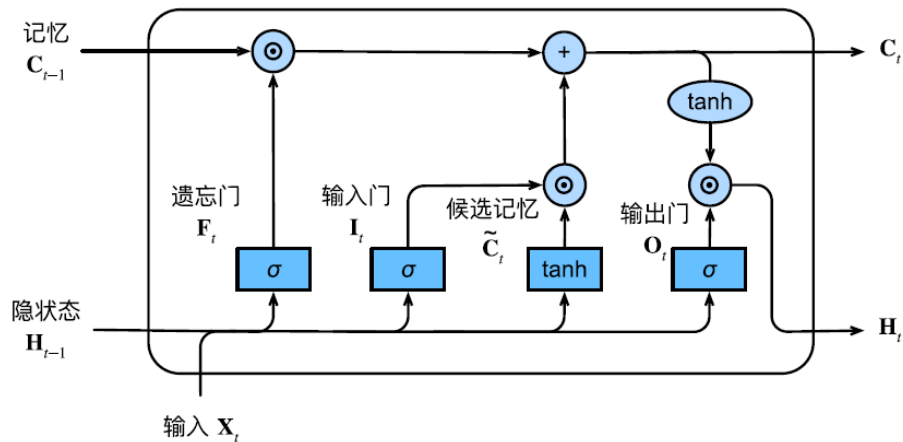
$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$



隐状态

➤ 隐状态 $\mathbf{H}_t \in \mathbb{R}^{n \times h}$

$$\mathbf{H}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$



长短期记忆(LSTM)总结

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$

$$\tilde{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)$$

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$

$$H_t = O_t \odot \tanh(C_t)$$

➤ 遗忘门和输入门，输出门

➤ 生成信息用于参数权重控制

➤ 长短期记忆网络的隐藏层输出包括“隐状态”和“记忆元”。只有隐状态会传递到输出层，而记忆元完全属于内部信息。

回顾与展望

- RNN: 通过隐状态和权重共享处理序列，但受限于长期依赖问题（梯度消失）
- GRU & LSTM: 通过门控机制解决了长期依赖问题，能够学习长距离依赖关系
 - LSTM: 引入独立的记忆元 C_t 和三个门（遗忘、输入、输出），设计精巧，功能强大
 - GRU: 将记忆元和隐状态合并，使用两个门（重置、更新），是LSTM的有效简化版，计算效率更高
- 在算法谱系中的位置:
 - RNN是序列建模的基石
 - LSTM和GRU是RNN最重要的改进，至今仍是处理序列问题的强大工具
 - 未来展望: 注意力机制 (Attention) 和 Transformer 架构的出现，为序列建模提供了新的范式，尤其在NLP领域取得了巨大成功。但理解RNN及其门控变体，是理解这些更前沿模型演进逻辑的关键