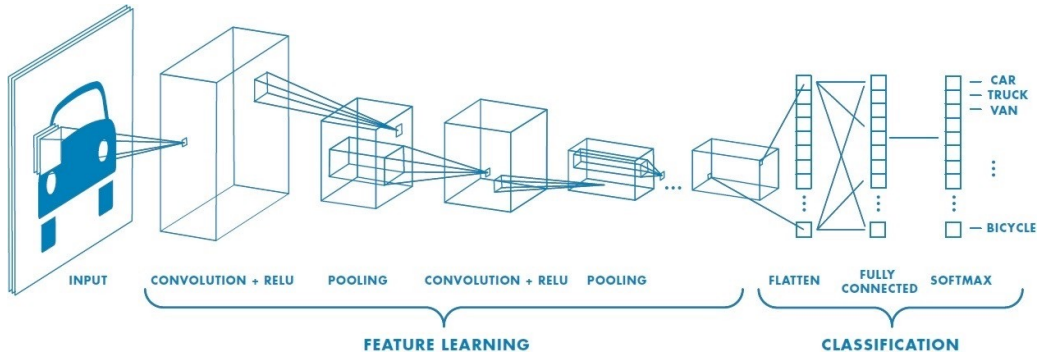
第九周 循环神经网络（RNN）和长短期记忆网络（LSTM）

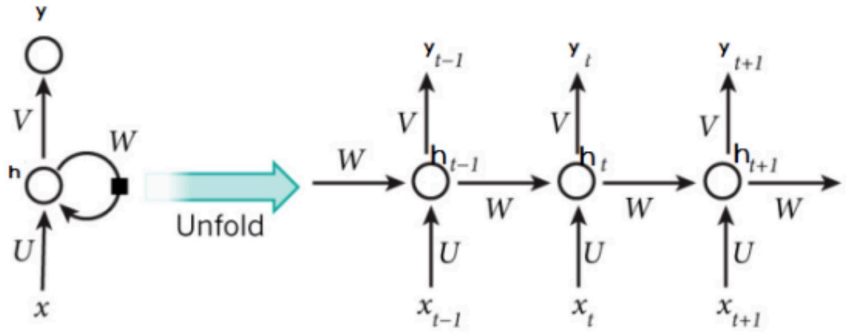
李泽榉，复旦大学 生物医学工程与技术创新学院

目录

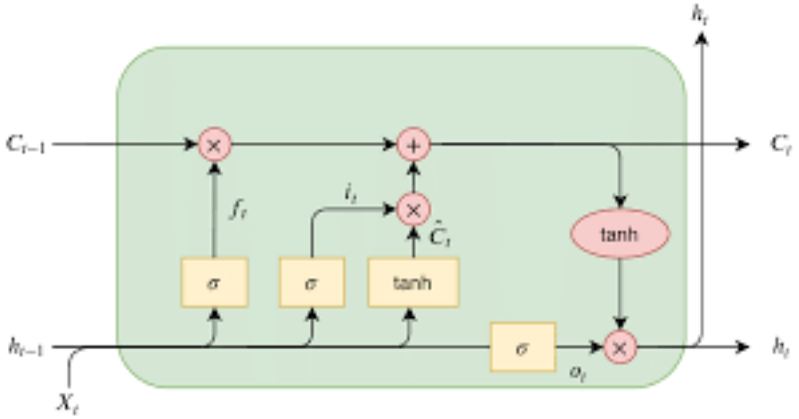
深度学习 – 新型网络结构



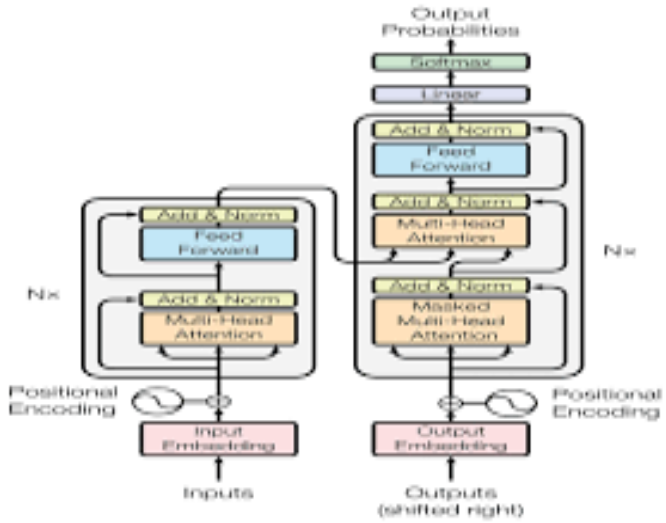
Convolutional Neural Networks



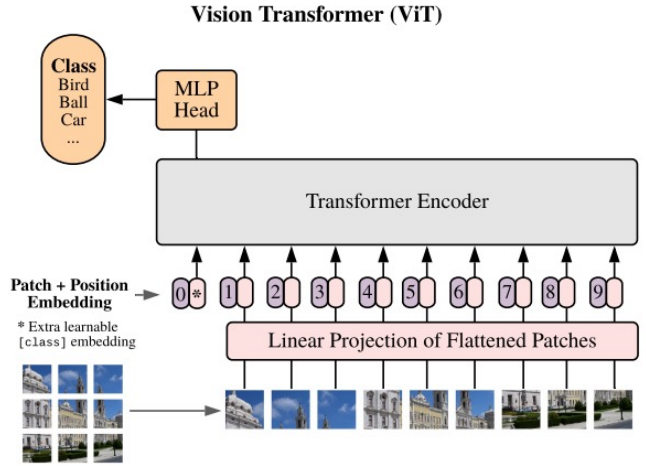
Recurrent Neural Networks



Long Short-Term Memory (LSTM)



Transformer



Vision Transformer (ViT)

循环神经网络（RNN）

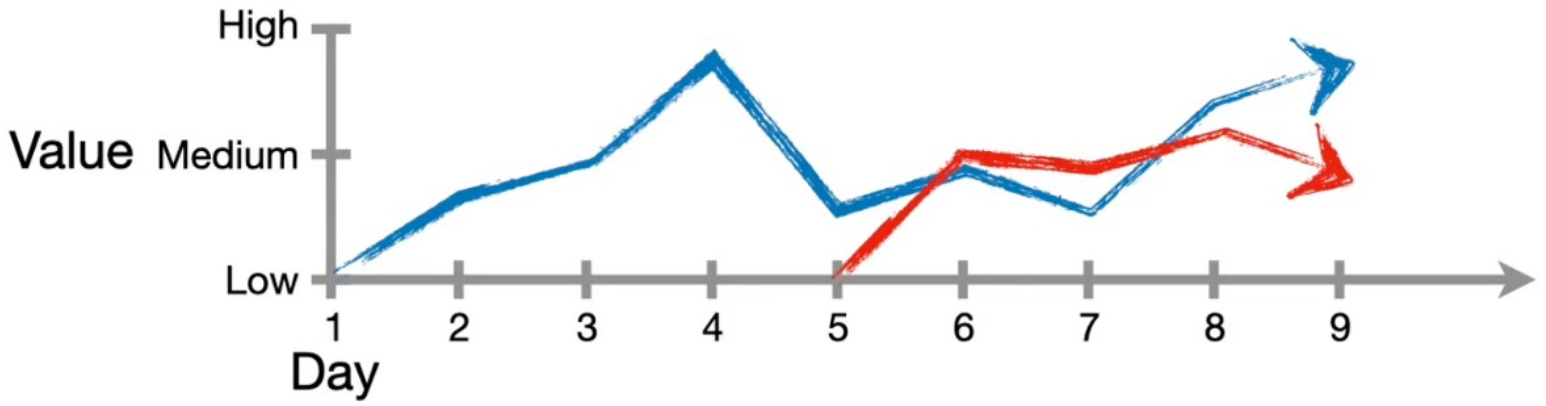
基本概念：专门用来处理序列数据的神经网络，通过内部状态记忆之前的信息，捕获序列数据中的动态特性

应用领域：自然语言处理、语音识别、时间序列分析

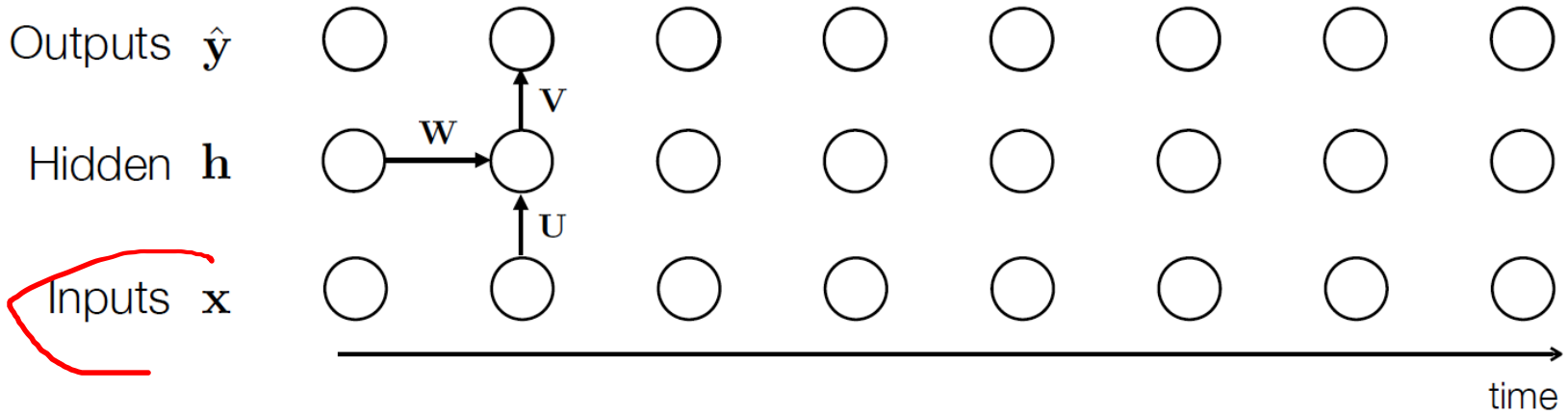
循环神经网络（RNN）

股价预测

我们想要使用循环神经网络来预测股票的价格



循环神经网络（RNN）



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

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}$$

$$\hat{\mathbf{y}}_t = \mathtt{softmax}(\mathbf{o}_t)$$

类比

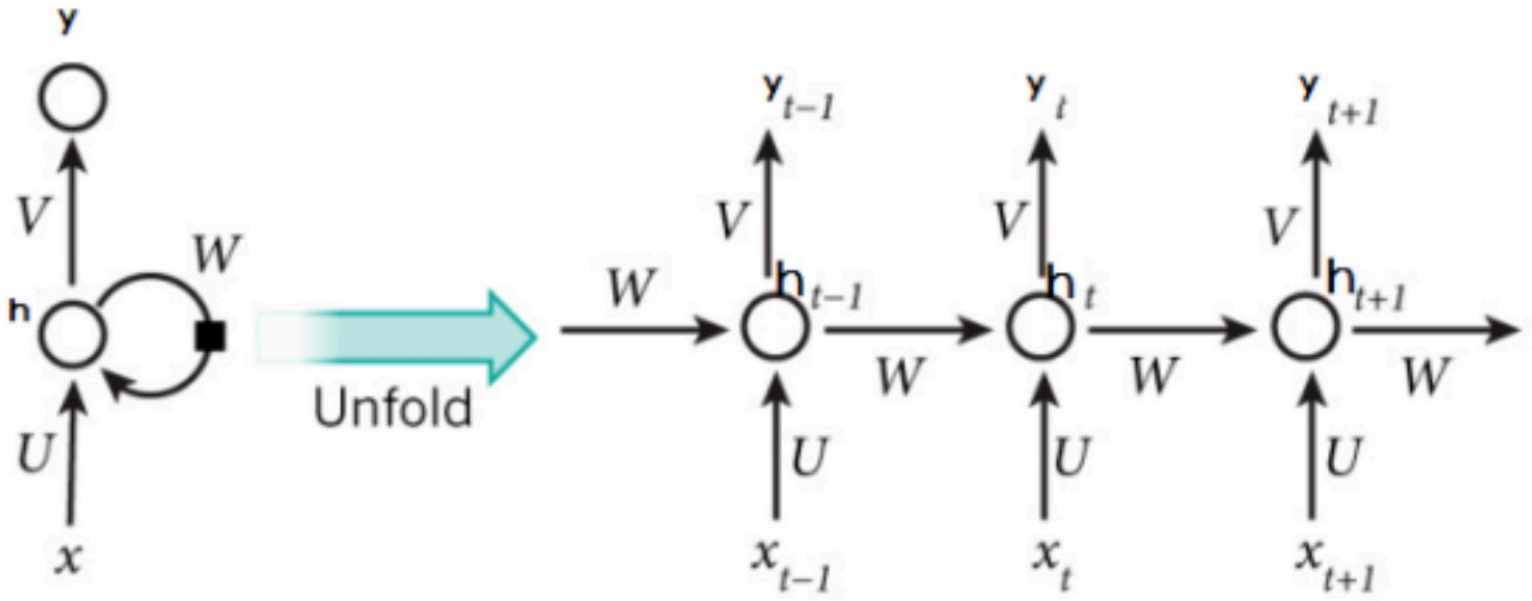
读小说时，看第5章脑中还记着前几章剧情

= h(t-1) 上一步隐藏状态

新剧情 x(t) + 旧记忆 h(t-1)
→ 新理解 h(t)

核心公式：h(t) = tanh(Wh · h(t-1) + Wx · x(t) + b)　其中 h(t-1) 是上一时刻隐藏状态，x(t) 是当前输入

循环神经网络（RNN）



时间展开 (Unrolling)

同一个RNN单元在每个时间步重复使用，共享同一套权重 W。

参数共享优势：
无论序列长100步还是1000步，参数量不变！

对比全连接网络：
100步序列需要100× 权重
RNN只需要一套 W

梯度沿时间反向传播 →
每步都乘同一个 Wh

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

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}$$

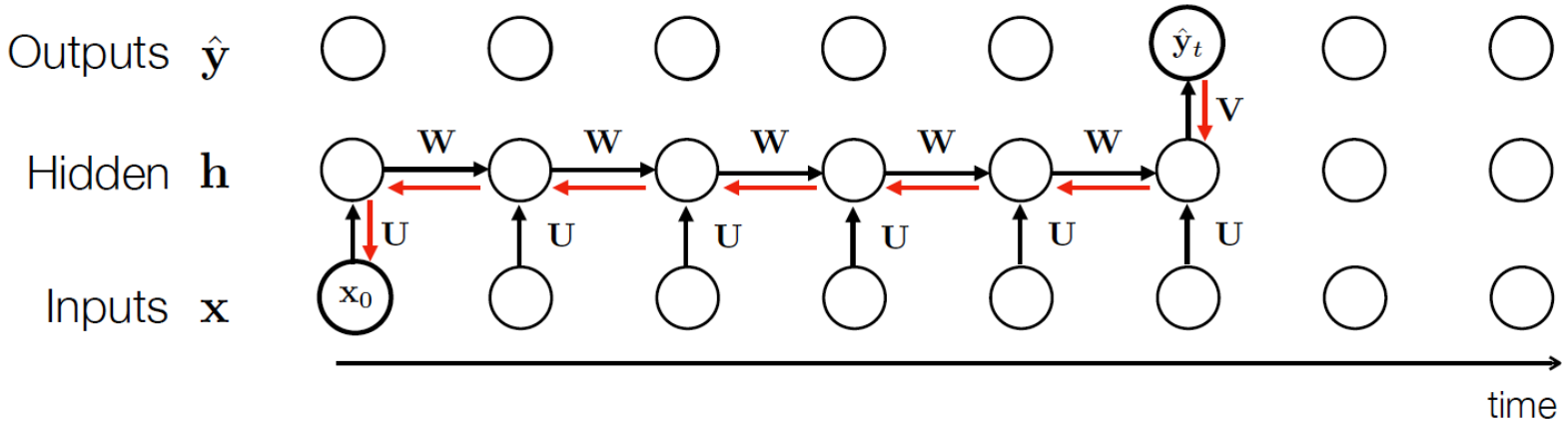
$$\hat{\mathbf{y}}_t = \mathtt{softmax}(\mathbf{o}_t)$$

目录

训练RNN

BPTT核心思想：把RNN沿时间展开，当成一个深层前馈网络，再用普通反向传播求梯度。

随时间反向传播（BP Through Time）



$$\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{x}_0} = \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_1}{\partial \mathbf{h}_0} \frac{\partial \mathbf{h}_0}{\partial \mathbf{x}_0}$$

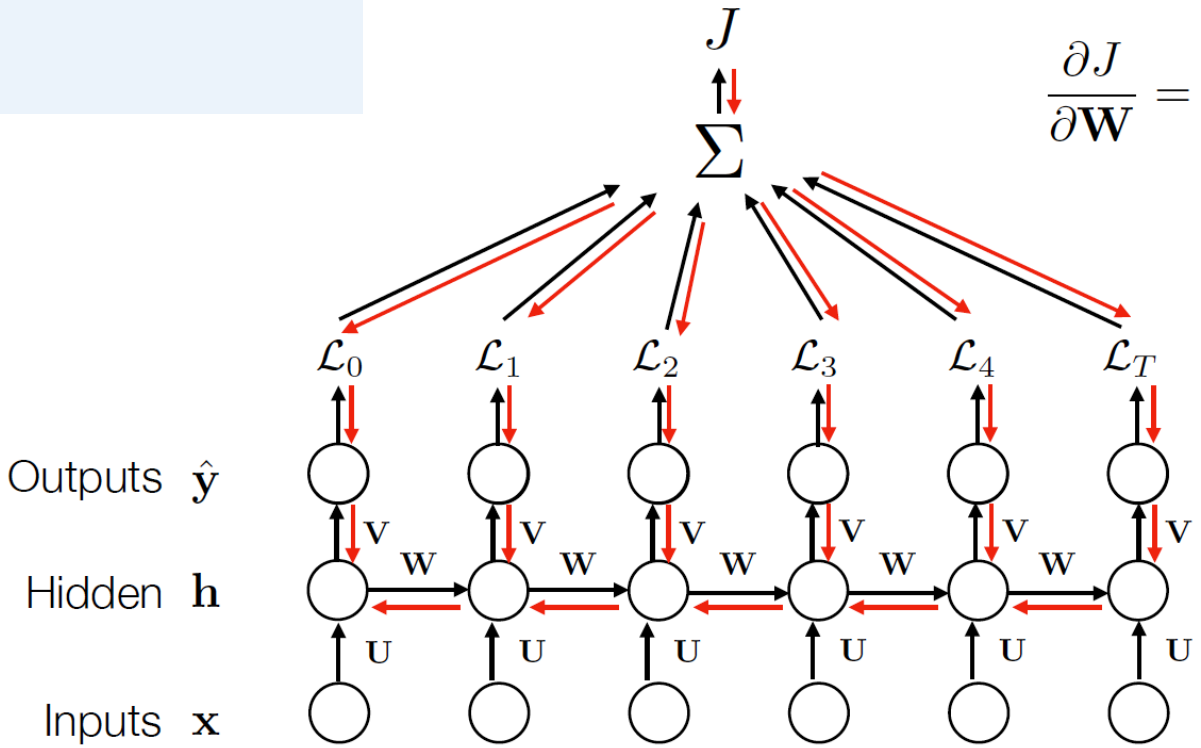
梯度计算: dL/dW = sum_t dLt/dW（累加每一时刻的梯度）| 类比：追责任链——每一步走错都要为最终失败负责！

训练RNN

BPTT四步流程：
① 前向传播计算每步输出
② 计算总损失 L = sum(Lt)
③ 从T时刻往0时刻反向传梯度
④ 累加各时刻的 dL/dW 更新参数

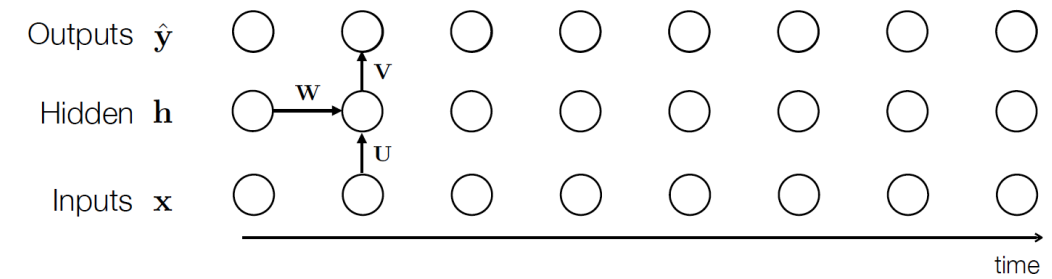
随时间反向传播（BP Through Time）

$$\frac{\partial J}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}_t(\hat{\mathbf{y}}_t, \mathbf{y}_t)}{\partial \mathbf{W}}$$



注意：序列越长，需要保存的中间状态越多（内存开销大）。实践中常用截断BPTT（每K步截断一次），用近似梯度换取效率。

训练RNN

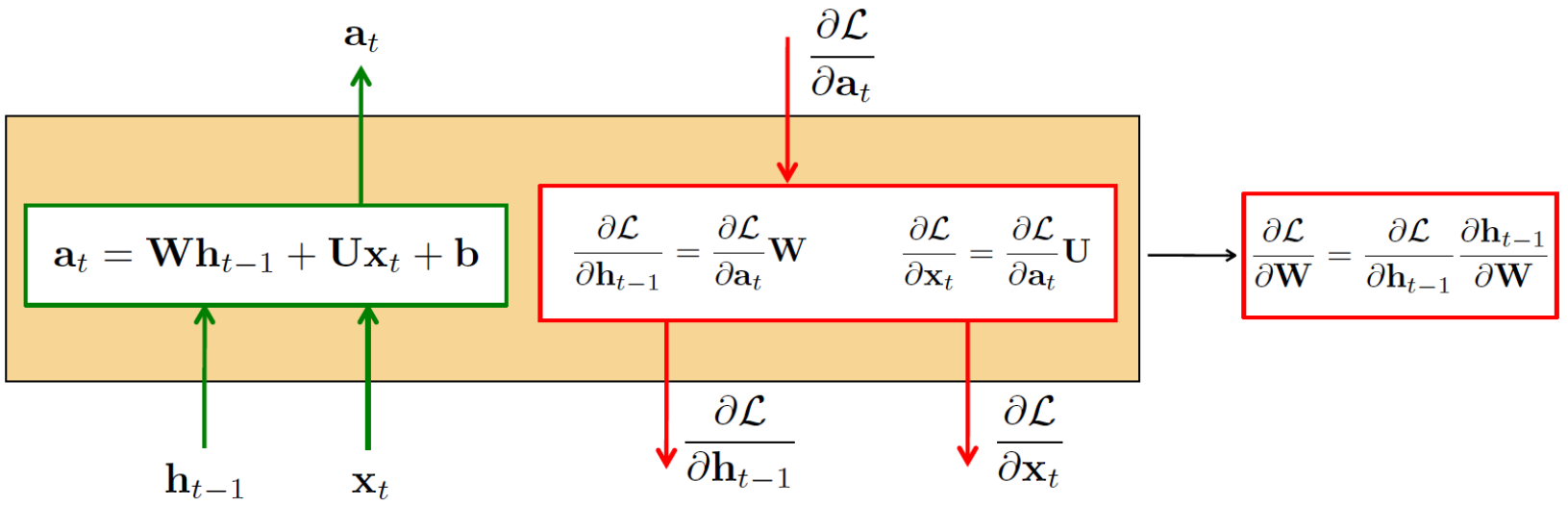


$$\mathbf{a}_t = \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}$$

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}$$

$$\hat{\mathbf{y}}_t = \texttt{softmax}(\mathbf{o}_t)$$

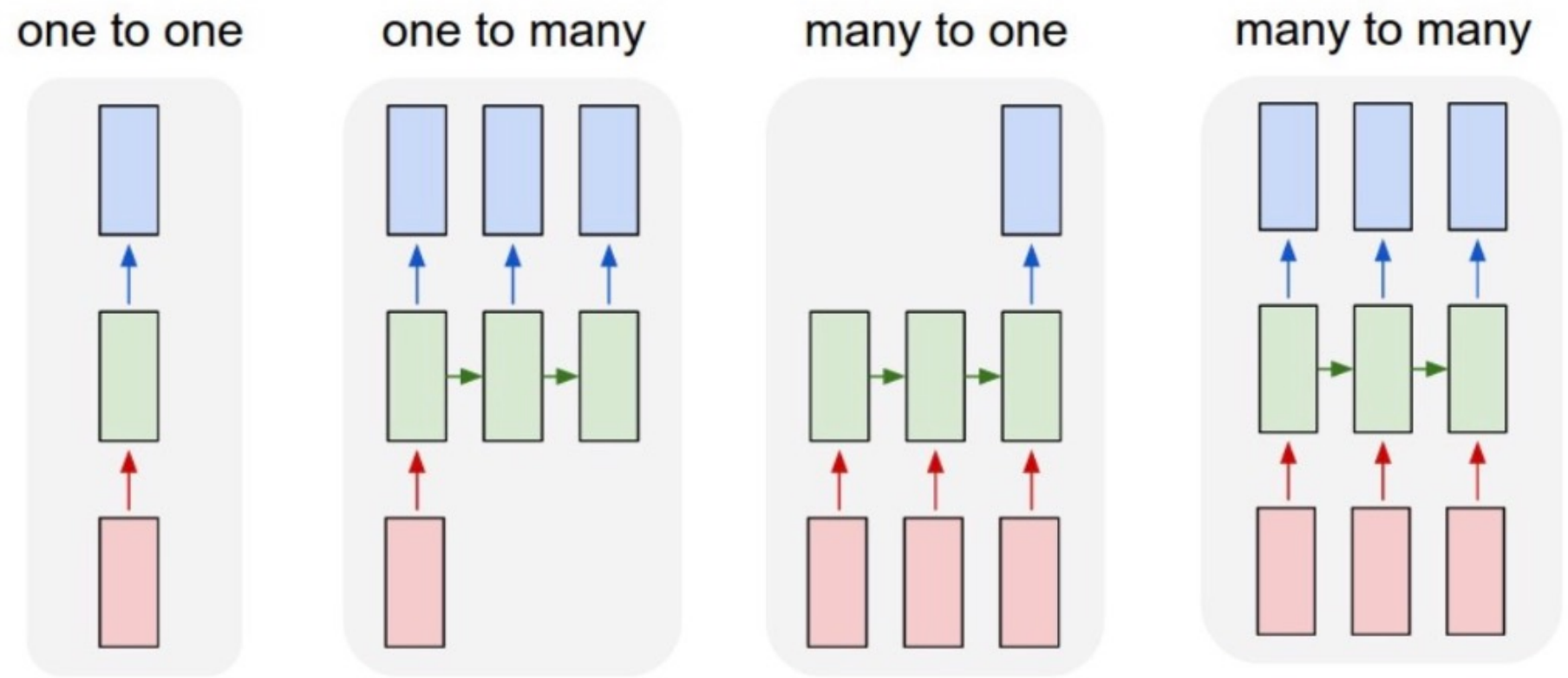


$$\frac{\partial J}{\partial \mathbf{W}} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}(\hat{\mathbf{y}}_t, \mathbf{y}_t)}{\partial \mathbf{W}}$$

前向传播

反向传播

循环神经网络（RNN）



四种结构

1:1
普通前馈网络

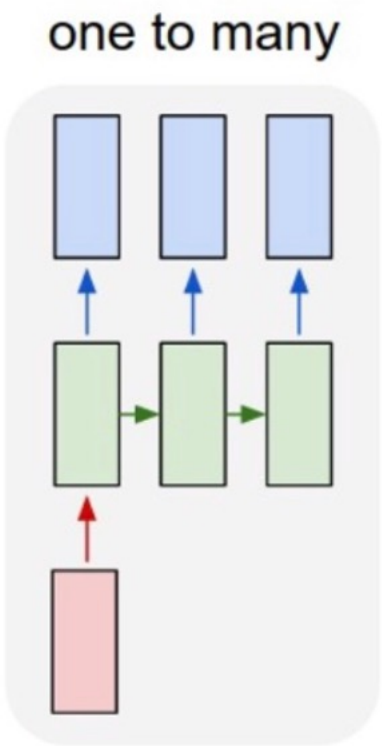
1:N
图像描述
音乐生成

N:1
情感分析
垃圾邮件分类

N:N
词性标注
机器翻译

口诀:单进多出=说故事 | 多进单出=做判断 | 多进多出(同步)=逐字翻译 | 多进多出(异步)=整句翻译

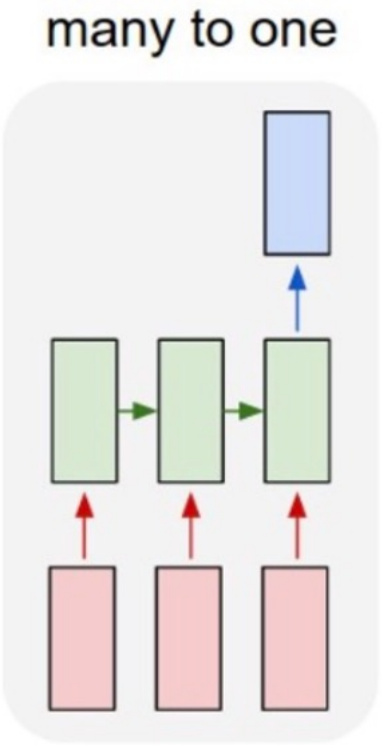
循环神经网络（RNN）



特点: 单个输入通过RNN生成一系列输出，可以连续生成多个输出。

适用任务: 能够从单个数据点生成序列，可以适用于音乐生成和图像描述等任务

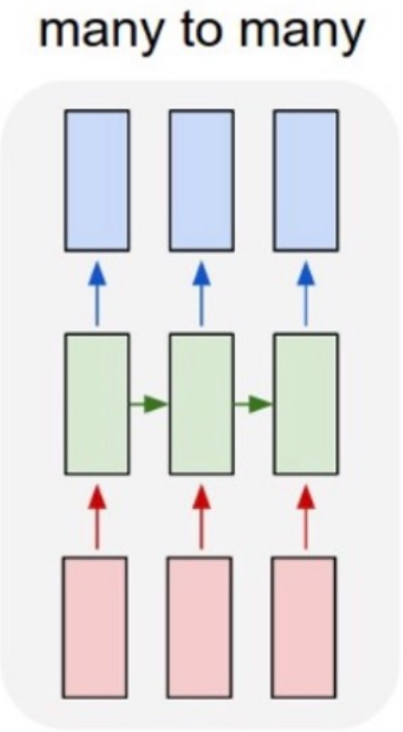
循环神经网络（RNN）



特点: 序列输入转化为单个输出，经过RNN处理后输出一个结果

适用任务: 需要理解整个序列后做出判断或分类的任务，例如情感分析和垃圾邮件分类检测

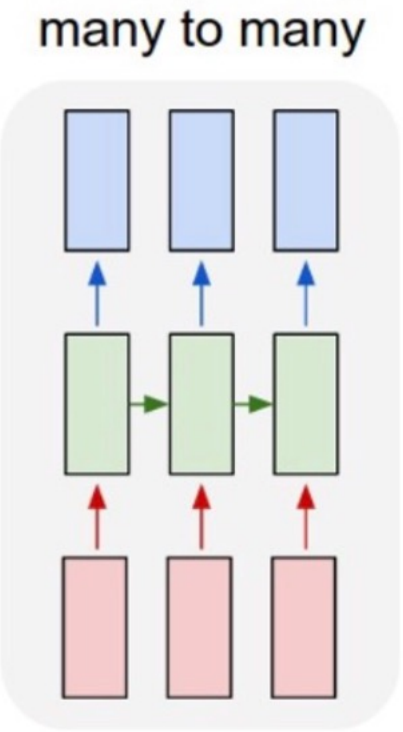
循环神经网络（RNN）



特点: (同步类型)序列输入和序列输出，输入和输出长度相同

适用任务: 在每个时间步都有输入和输出，适用于需要逐步处理的复杂任务，如词性标注

循环神经网络（RNN）

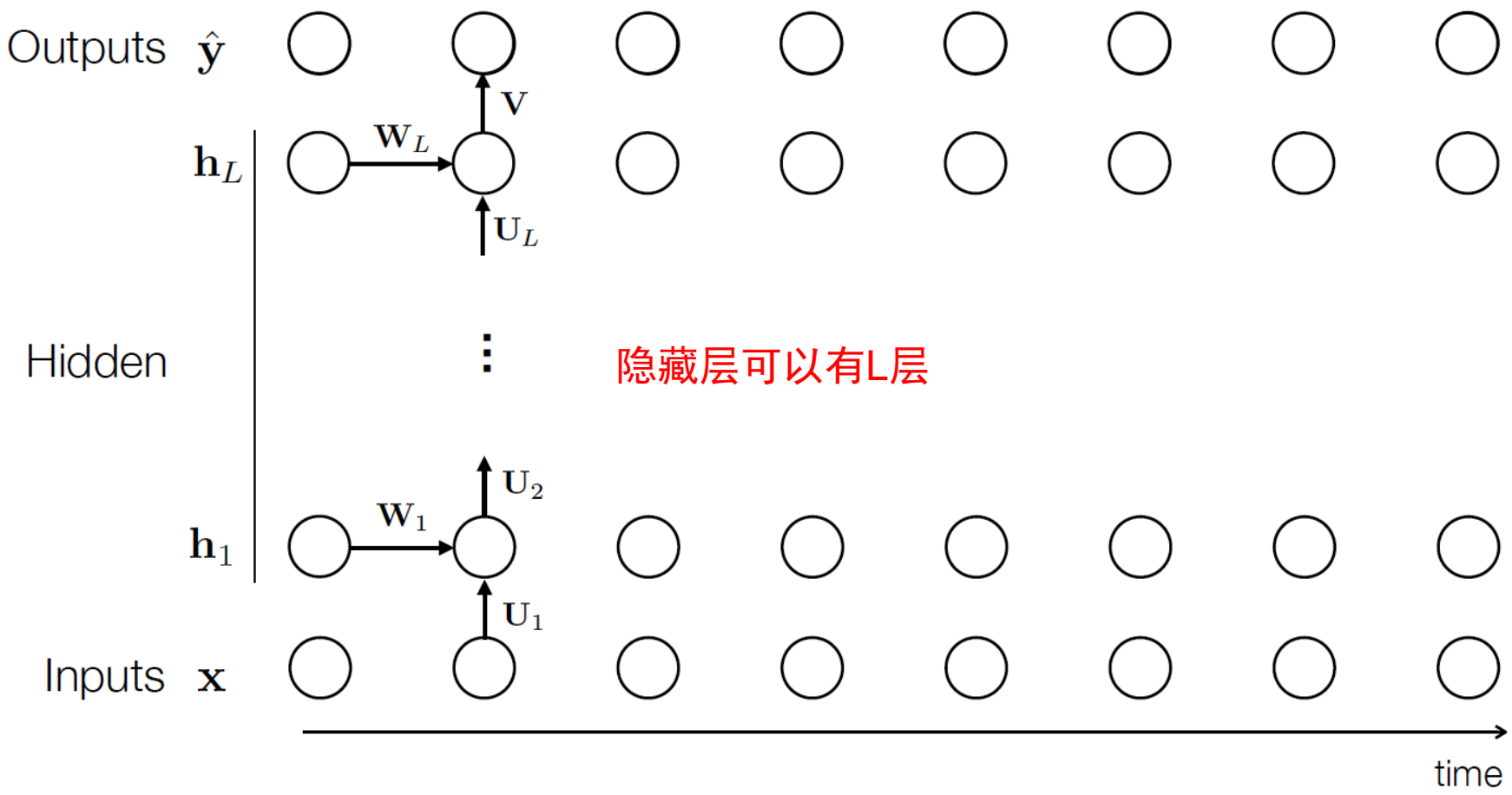


特点: (异步类型)序列输入和序列输出，输入和输出长度不相同

适用任务: 用来处理输入输出长度不匹配的情况，允许网络结构更加灵活地处理信息，如机器翻译和语音识别任务

深度RNN网络：Deep RNN

Deep RNN：叠加多层隐藏层，下层输出作为上层输入。第 l 层：h(l,t) = tanh(W(l)·h(l,t-1) + U(l)·h(l-1,t))　优势：更强表达力，能学习更抽象的时序特征。实践建议：从2层开始，层数越多梯度越难训练。

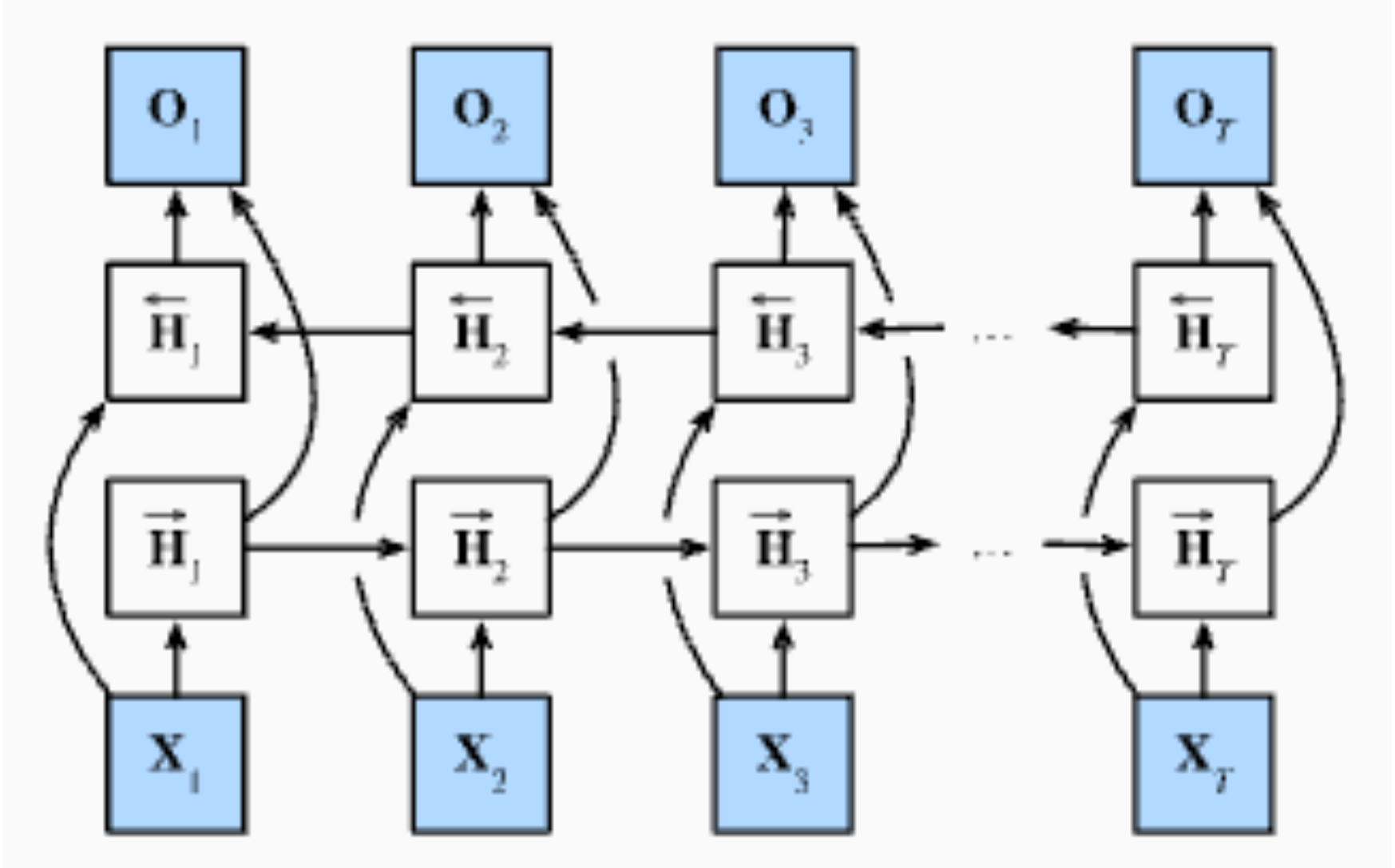


层数
越多
↓
表达力
更强

但训练
更慢！

建议
2-4层

双向RNN网络：Bi-RNN



为什么需要双向？

单向RNN只能看「过去」

例：
「我去了___，
吃了正宗小龙虾」
不看后文无法
判断是哪个城市！

公式：
前向 h_f(t)
后向 h_b(t)
合并 y(t) = V·[h_f; h_b]

适用：情感分析
命名实体识别

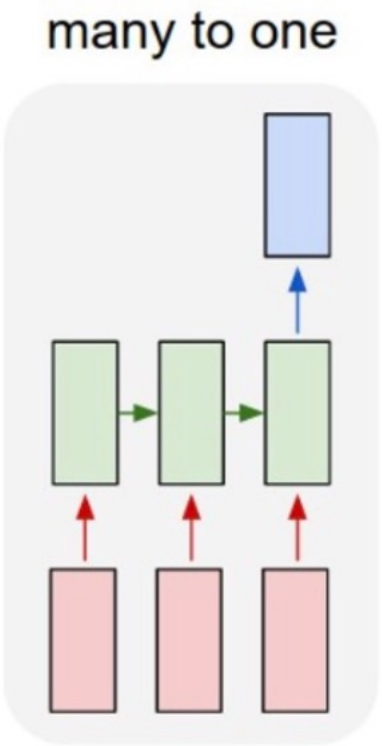
不适用：
实时语音转写
（需要未来输入）

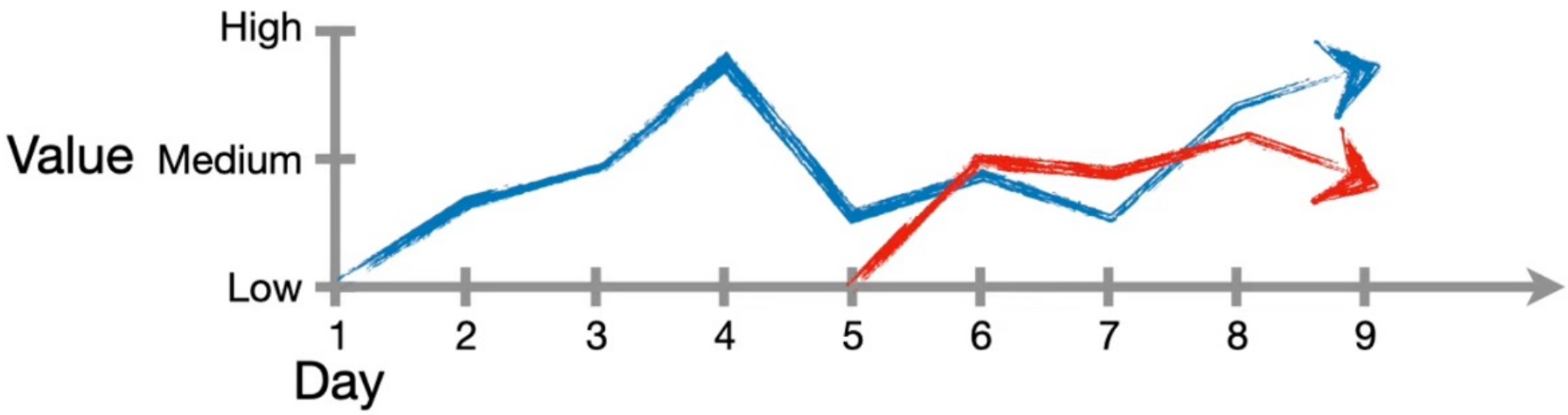
目录

循环神经网络（RNN）的应用

输入特征(每日):
- 开盘/收盘/最高/最低价
- 成交量、技术指标(MA,RSI)

时间步 t = 每一天

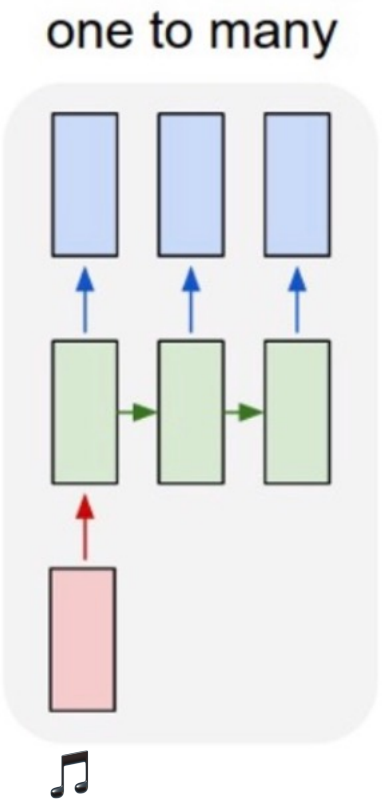




RNN处理: h(t) = 对过去价格走势的理解 → y(t) = 预测明天收盘价　注意: 股市受政策/情绪影响，纯技术面RNN有局限！

循环神经网络（RNN）的应用

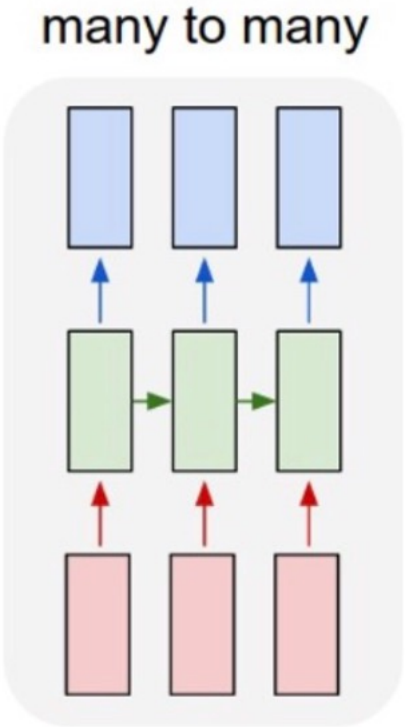
音乐生成

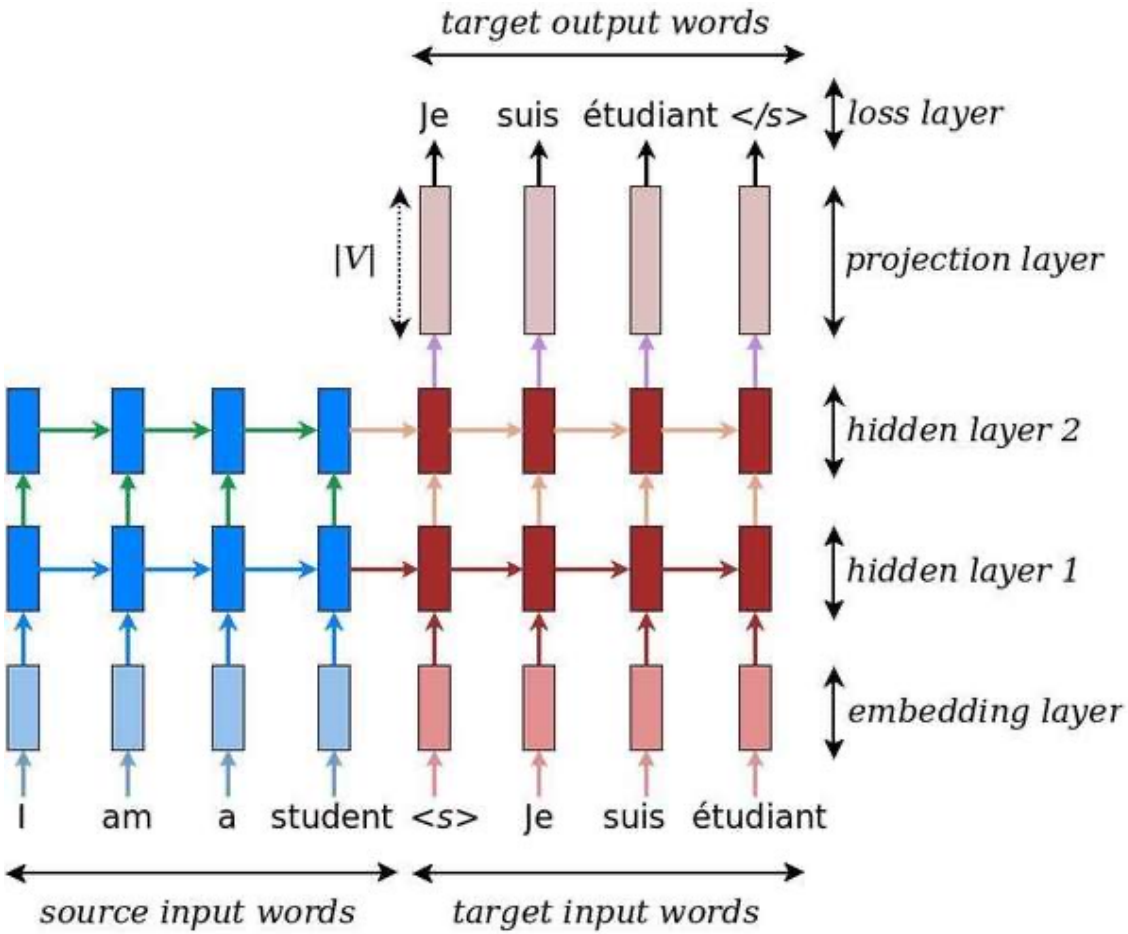


输入一个音乐风格或者一个音符，
生成一个音乐片段

循环神经网络（RNN）的应用

Seq2Seq架构:
Encoder RNN 读入源语言
→ 压缩为上下文向量 c
Decoder RNN 以 c 为起点
→ 逐词生成目标语言





例:
「我爱你」
↓ Encoder
上下文向量 c
↓ Decoder
「I love you」

瓶颈问题:
所有信息压缩
在一个向量里
→ 长句会丢失细节

这就是后来
Attention机制
诞生的原因！

目录

RNN的缺点：梯度消失/爆炸问题

RNN的缺点：梯度消失问题

核心问题：在训练RNN时，长序列的梯度通过时间反向传播会导致梯度数值逐渐减小至接近零，使得网络权重难以更新，尤其是对于序列前端的信息，导致模型难以学习到输入序列中时间间隔较长的依赖关系，即模型的“记忆”能力受限。

RNN的缺点：梯度消失问题

数学解释：RNN的梯度是通过链式法则计算得到的，涉及多个较小的梯度乘积。在多个时间步上累乘这些小数值会导致总梯度迅速衰减。

$$\frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{x}_0} = \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_1}{\partial \mathbf{h}_0} \frac{\partial \mathbf{h}_0}{\partial \mathbf{x}_0}$$

RNN的缺点：梯度消失问题的影响

学习障碍：权重更新受阻，特别是影响网络的早期层（对应于序列的开始部分），使得模型不能有效学习序列的全局特征。

- **序列处理限制**：这种梯度问题限制了RNN处理长序列的能力，因为序列越长，梯度消失的问题越严重。

RNN的缺点：梯度爆炸问题

- **核心问题**：在RNN的训练过程中，当梯度的值在反向传播过程中急剧增大，超过了处理能力，会导致网络权重更新过大，从而使得学习过程不稳定，甚至导致模型数值计算上的溢出。
- **影响**：梯度爆炸常常导致模型训练过程中出现数值不稳定，例如权重更新过大，使得模型无法收敛，表现为损失函数值波动极大或变为NaN（不是一个数字）。

RNN的缺点：梯度爆炸问题

数学解释：与梯度消失类似，梯度爆炸也是通过链式法则计算得到的，但是当梯度的累积乘积异常大时，就会出现爆炸。

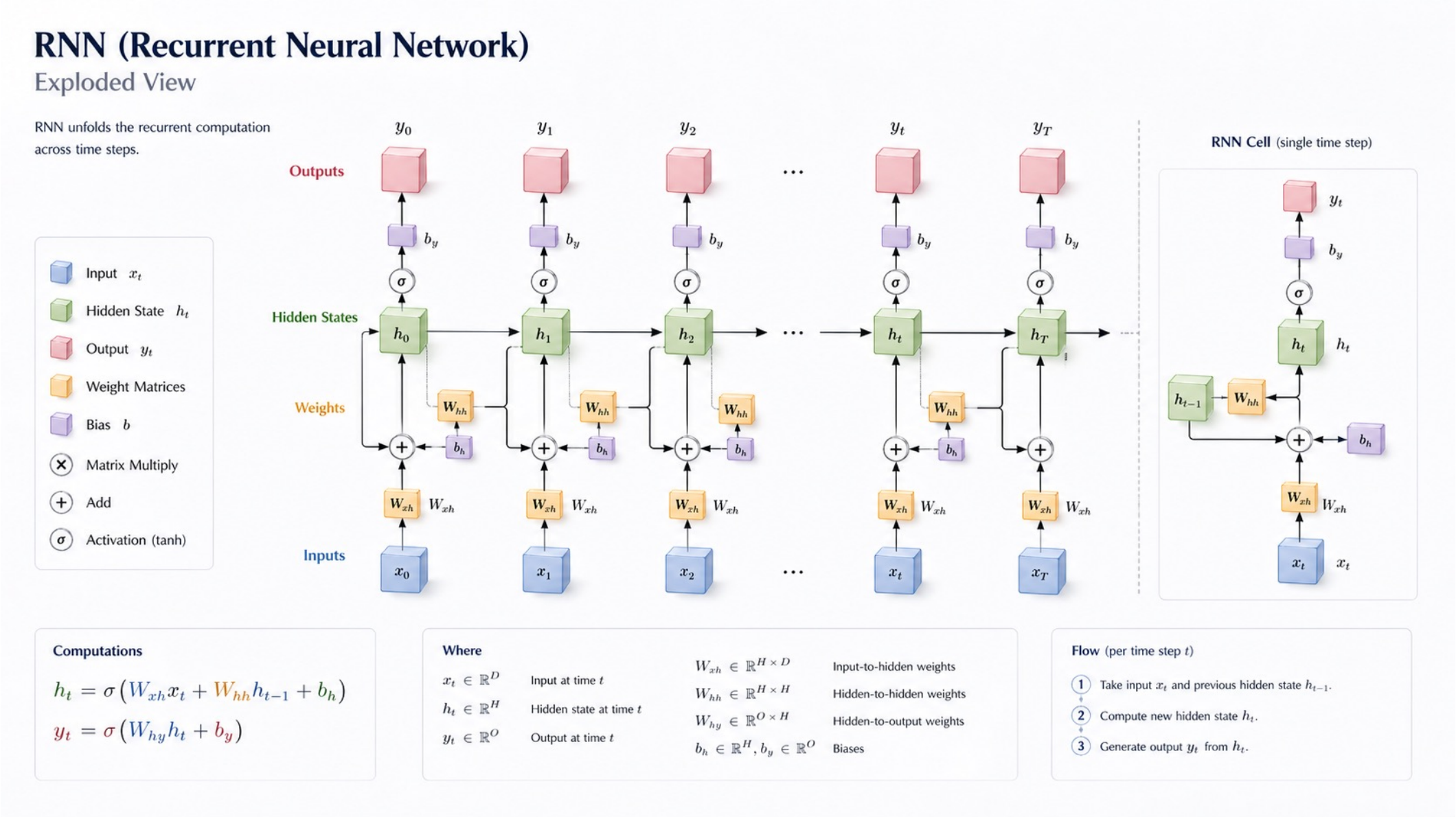
循环神经网络（RNN）：需要特殊训练算法

解决梯度消失/爆炸问题

梯度裁剪：设定阈值限制梯度的最大值，避免梯度爆炸，间接减轻梯度消失的影响

短序列训练：通过将长序列切分为较短的片段训练

循环神经网络（RNN）：具体实现



练习时间

1、什么是RNN？它与传统神经网络的主要区别是什么？

RNN（循环神经网络）是一种专门用来处理序列数据的神经网络，通过内部状态记忆之前的信息，捕获序列数据中的动态特性

2、RNN的主要优点和缺点是什么？

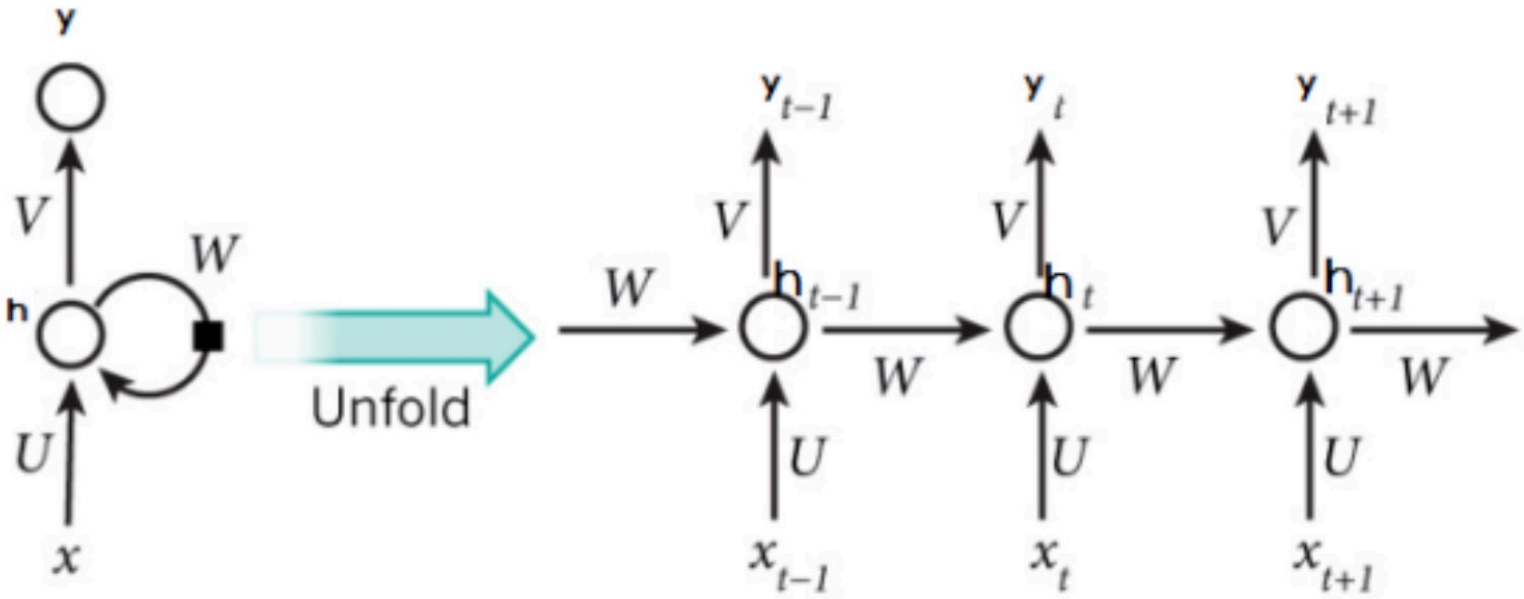
优点：能够处理长序列数据，适合时间序列、文本等任务。
缺点：训练过程中容易出现梯度消失或梯度爆炸问题，对长距离的序列信息记忆能力较差

3、什么是RNN的“时间步”（time step）？它在RNN中的作用是什么？

时间步是RNN处理序列数据的基本单位。每个时间步对应序列中的一个元素，RNN通过逐步处理每个时间步的输入并更新隐藏状态，从而捕捉序列中的信息。

练习时间

4、看图推导RNN的前向传播公式



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

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}$$

$$\hat{\mathbf{y}}_t = \mathtt{softmax}(\mathbf{o}_t)$$

练习时间

5、假设RNN的隐藏状态h_t的梯度为$\frac{\partial L}{\partial h_t}$，请分析为什么在长序列中梯度会消失或爆炸。

梯度消失：由于RNN的梯度是通过时间反向传播（BPTT）计算的，梯度在每一步都会乘以权重矩阵 $\boldsymbol{W}$。如果 $\boldsymbol{W}$ 的特征值小于1，梯度会指数级减小，导致梯度消失。
梯度爆炸：如果 $\boldsymbol{W}$ 的特征值大于1，梯度会指数级增大，导致梯度爆炸。

练习时间

2. （5 分）循环神经网络 (RNN) 特别适合处理以下哪种类型的数据？

A. 静态图像数据
B. 独立的表格数据
C. 序列数据（如文本、时间序列）
D. 无序的特征集合

练习时间

2. （5 分）循环神经网络 (RNN) 特别适合处理以下哪种类型的数据？

A. 静态图像数据
B. 独立的表格数据
C. 序列数据（如文本、时间序列）
D. 无序的特征集合

答案：C

练习时间

3. （5 分）一个团队正在使用循环神经网络（RNN）来执行词性标注（Part-of-Speech Tagging）任务。该任务的目标是为句子中的每一个单词分配一个正确的词性标签（如名词、动词、形容词等）。

当模型处理句子 "The cat sat" 中的第三个单词 "sat" 时，它在计算该单词的词性标签（动词）时，主要依赖于哪两部分信息？

A. 当前输入的单词 "sat" 和从处理 "cat" 时传递过来的隐藏状态（hidden state）
B. 整个句子 "The cat sat" 的全局信息和目标词性标签 "动词"
C. 仅当前输入的单词 "sat" 和一个随机初始化的向量
D. 当前输入的单词 "sat" 和句子中下一个单词的信息（如果存在）

练习时间

3. （5 分）一个团队正在使用循环神经网络（RNN）来执行词性标注（Part-of-Speech Tagging）任务。该任务的目标是为句子中的每一个单词分配一个正确的词性标签（如名词、动词、形容词等）。

当模型处理句子 "The cat sat" 中的第三个单词 "sat" 时，它在计算该单词的词性标签（动词）时，主要依赖于哪两部分信息？

A. 当前输入的单词 "sat" 和从处理 "cat" 时传递过来的隐藏状态（hidden state）

B. 整个句子 "The cat sat" 的全局信息和目标词性标签 "动词"

C. 仅当前输入的单词 "sat" 和一个随机初始化的向量

D. 当前输入的单词 "sat" 和句子中下一个单词的信息（如果存在）

答案：A

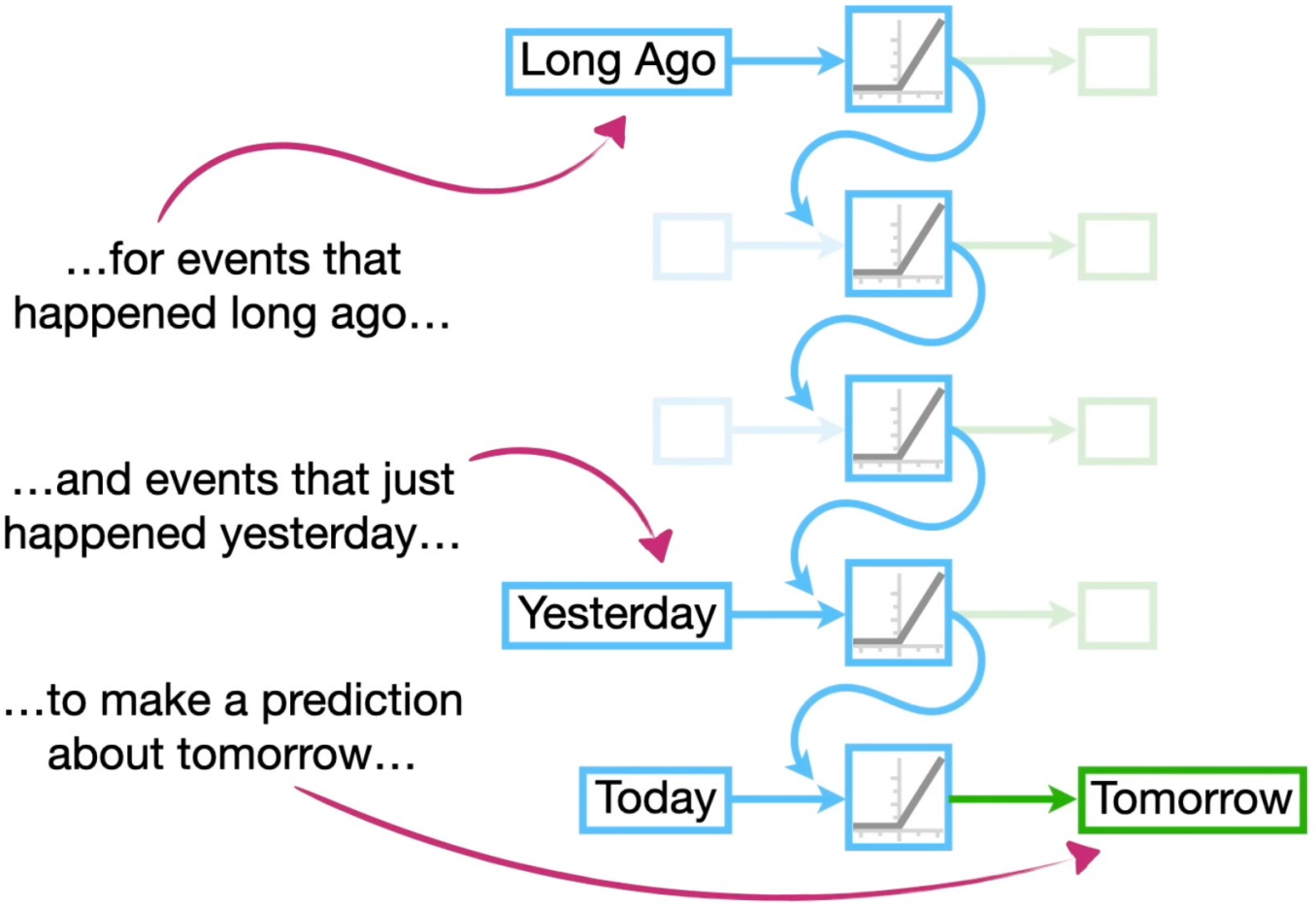
目录

长短时记忆神经网络（LSTM）

很久以前的信息

昨天的信息

今天的信息



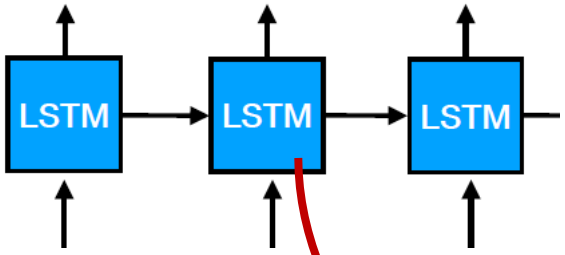
为什么需要LSTM?

RNN的记忆会随时间衰减:

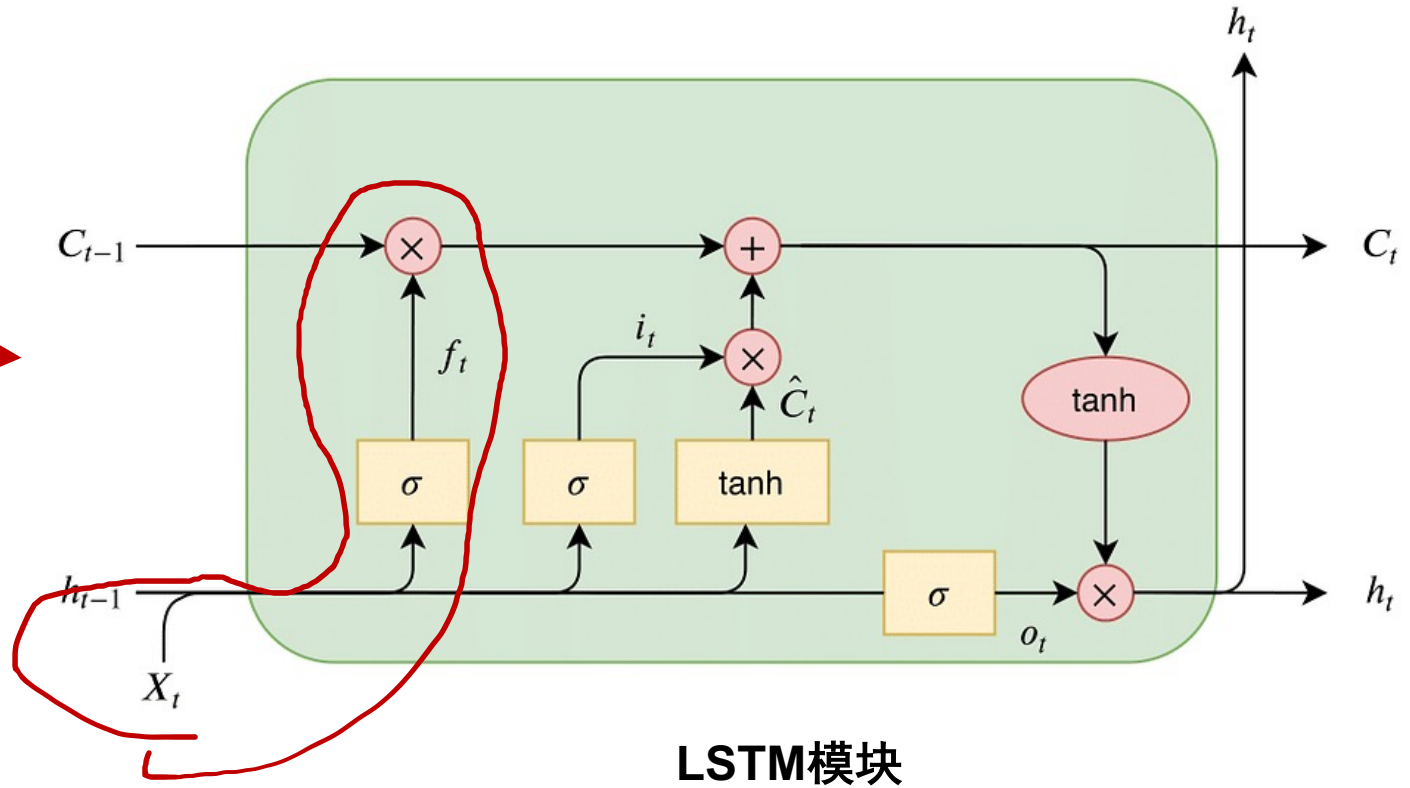
- 10步前 → 梯度 ×Wh^10 ≈ 0
- 100步前 → 信息几乎丢失

LSTM引入
细胞状态 c(t)
作为「长期记忆」
可跨数百步
无损传递信息

长短时记忆神经网络（LSTM）



一个LSTM网络包含很多LSTM模块，每一个模块里面的操作如我们前面所讲：



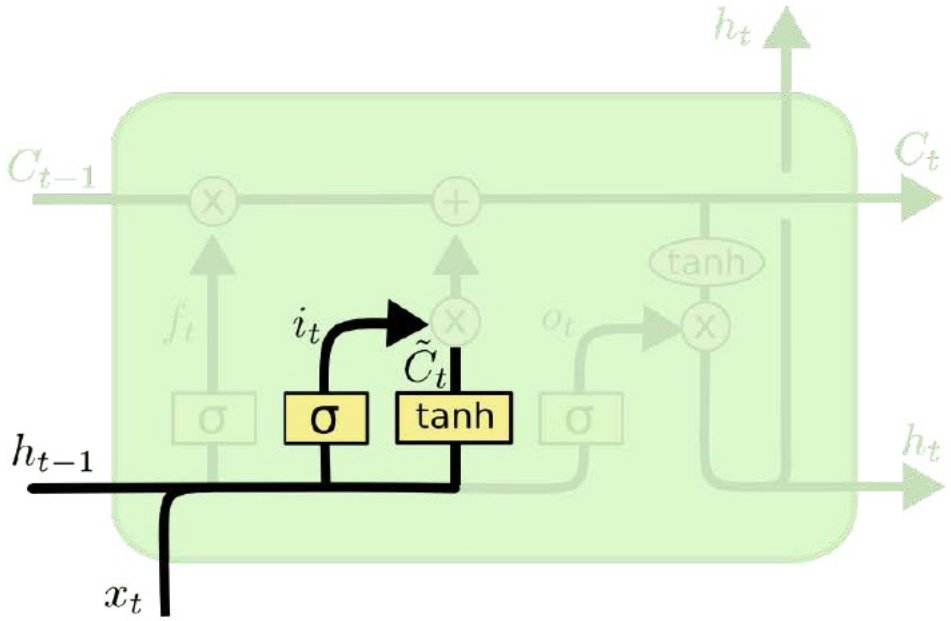
LSTM模块

https://www.youtube.com/watch?v=YCzL96nL7j0

目录

长短时记忆神经网络（LSTM）

输入门：



which indices to write to

公式：

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

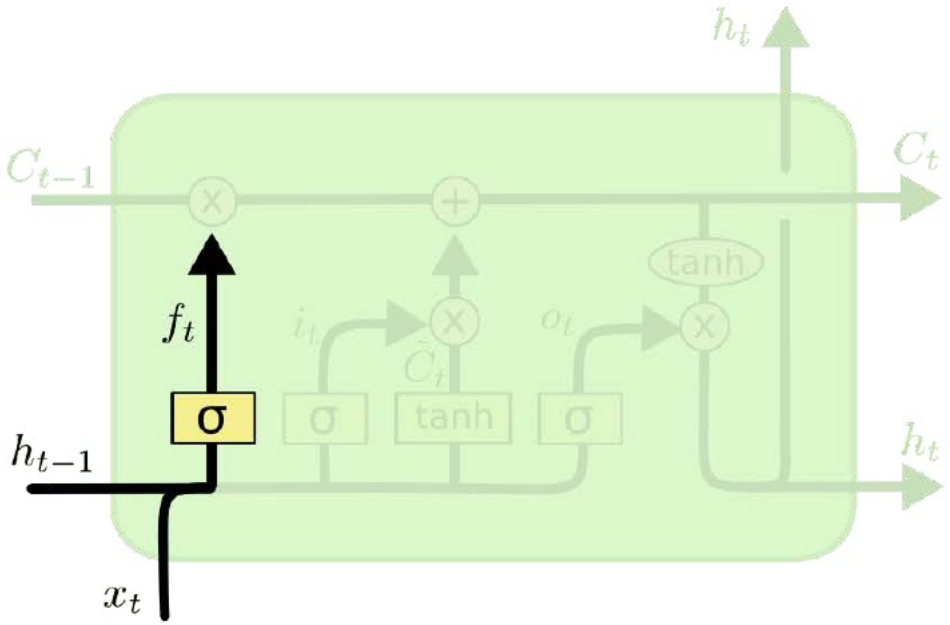
what to write to those indices

输入门公式:
i(t) = sigmoid(Wi·x(t) + Ui·h(t-1) + bi)
c~(t) = tanh(Wc·x(t) + Uc·h(t-1) + bc)
c(t) = f(t)⊙c(t-1) + i(t)⊙c~(t)

类比记忆:
输入门 = 课堂上决定"哪些笔记值得记下来"
- i(t) 控制写入强度 (0=不记, 1=全记)
- c~(t) 是候选内容（新知识）
- 最终写入 = 旧记忆保留 + 新内容写入

长短时记忆神经网络（LSTM）

遗忘门：



公式：

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

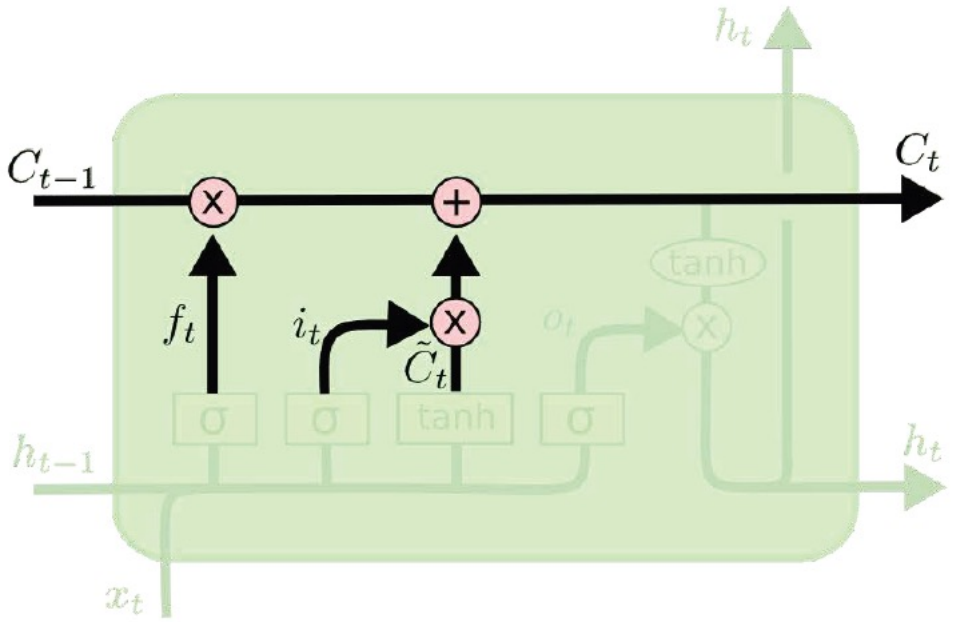
遗忘门公式:
f(t) = sigmoid(Wf·x(t) + Uf·h(t-1) + bf)

输出范围: f(t) ∈ (0, 1)
f(t) → 0: 完全遗忘旧记忆
f(t) → 1: 完全保留旧记忆

类比记忆:
遗忘门 = 期末复习时决定"哪些旧知识可以忘掉"
例:学到新语法规则时，过时的规则记忆被遗忘门压制。
RNN没有这个机制，所以会被早期无关信息"拖累"！

长短时记忆神经网络（LSTM）

遗忘和更新：



细胞状态更新公式:
c(t) = f(t) ⊙ c(t-1) + i(t) ⊙ c~(t)

⊙ = 逐元素乘法（Hadamard积）
第一项: 遗忘门 × 旧记忆（选择性保留）
第二项: 输入门 × 候选值（选择性写入）

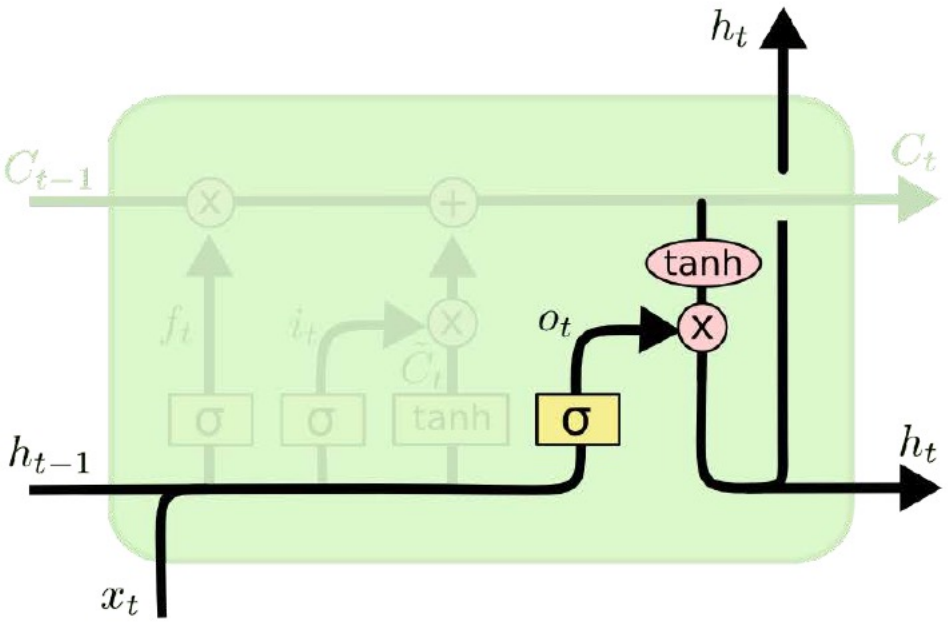
核心亮点：梯度高速公路！
c(t) 通过加法连接，梯度可以直接反向流通，不像tanh激活函数会压缩梯度，这正是LSTM解决梯度消失的关键。

执行遗忘和更新：

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

长短时记忆神经网络（LSTM）

输出门：



输出门公式:
o(t) = sigmoid(Wo·x(t) + Uo·h(t-1) + bo)
h(t) = o(t) ⊙ tanh(c(t))

o(t): 控制"读取"多少细胞记忆
tanh(c(t)): 将细胞状态压缩到(-1,1)
h(t): 最终隐藏状态（也是本步输出）

公式：

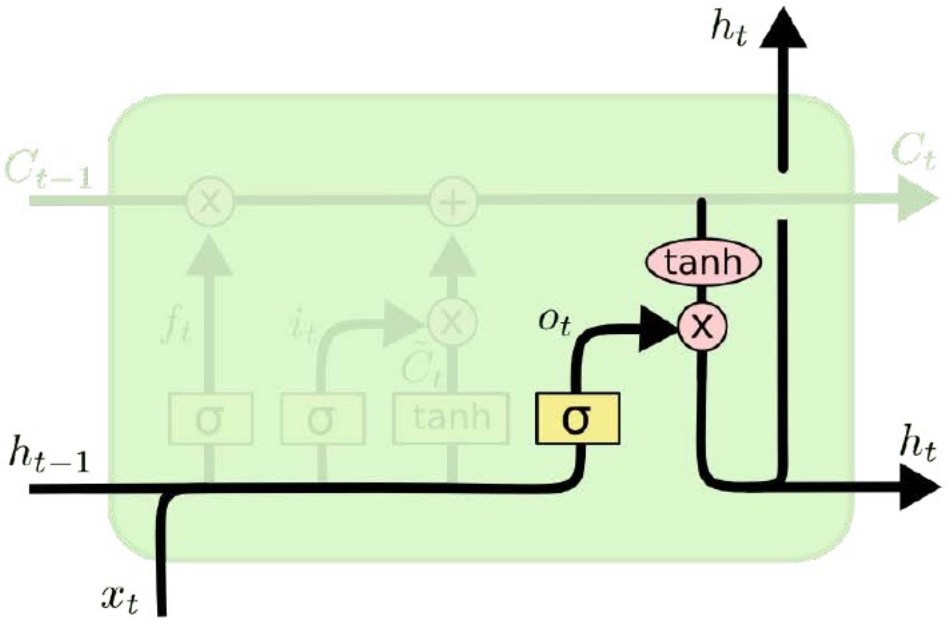
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

类比：
输出门 = 考试时决定"把脑中哪些知识写到答卷上"
记事本(c)存了很多，但只输出当前问题需要的部分。

长短时记忆神经网络（LSTM）

LSTM为什么在长序列数据的处理比RNN更优秀？



LSTM vs RNN 三大优势

① 解决梯度消失/爆炸
细胞状态用加法更新，
梯度可无损反向传播

② 捕捉长期依赖
遗忘门选择性保留
数百步之前的信息

③ 信息流可控性
三个门独立学习何时
遗忘 / 写入 / 输出

代价: 参数量约为
RNN的 4 倍

长短时记忆神经网络（LSTM）

LSTM为什么在长序列数据的处理比RNN更优秀？

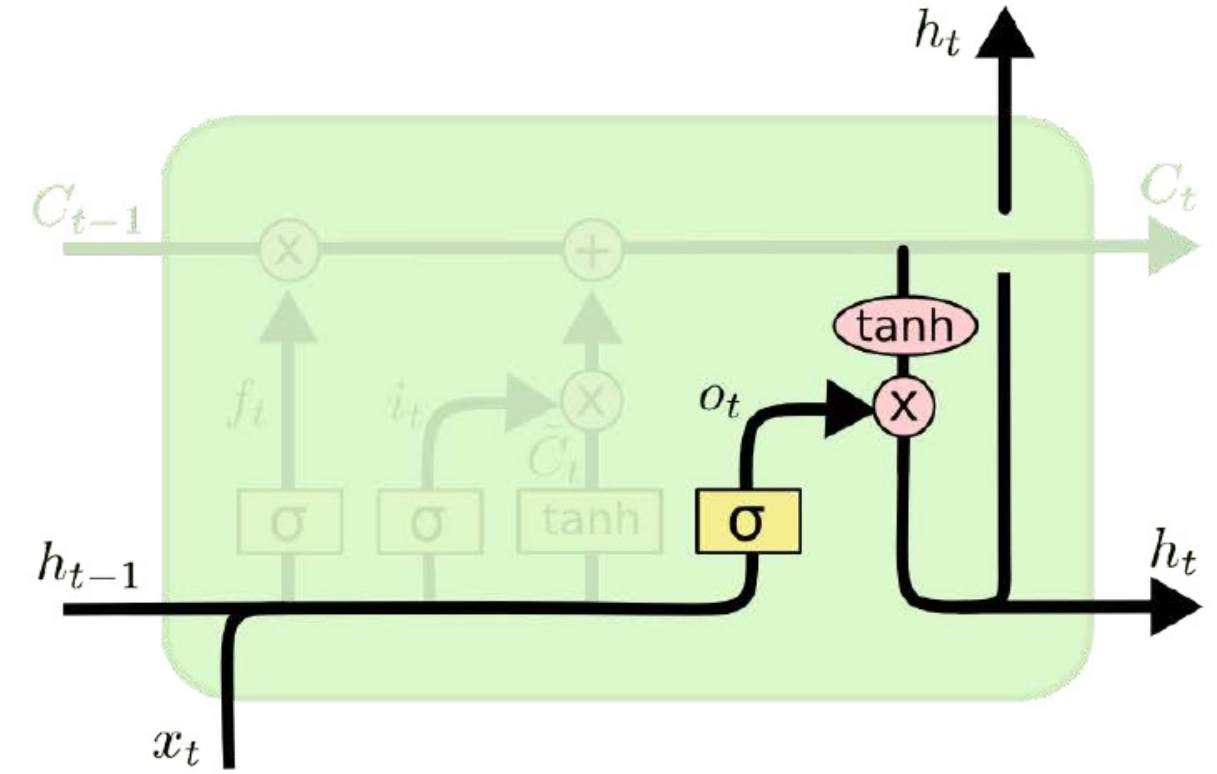
为什么LSTM能避免梯度消失？

关键：细胞状态 c(t) 的更新是加法：
c(t) = f(t)⊙c(t-1) + i(t)⊙c~(t)

梯度反向传播时：
∂c(t)/∂c(t-1) = f(t) (只需乘以遗忘门值)

- 当 f(t)≈1：梯度几乎无损传回早期时刻
- 不像RNN中连乘tanh'，梯度快速趋零

口诀：加法传梯度，门控管大小！



RNN的梯度消失/爆炸的问题成功解决

长短时记忆神经网络（LSTM）

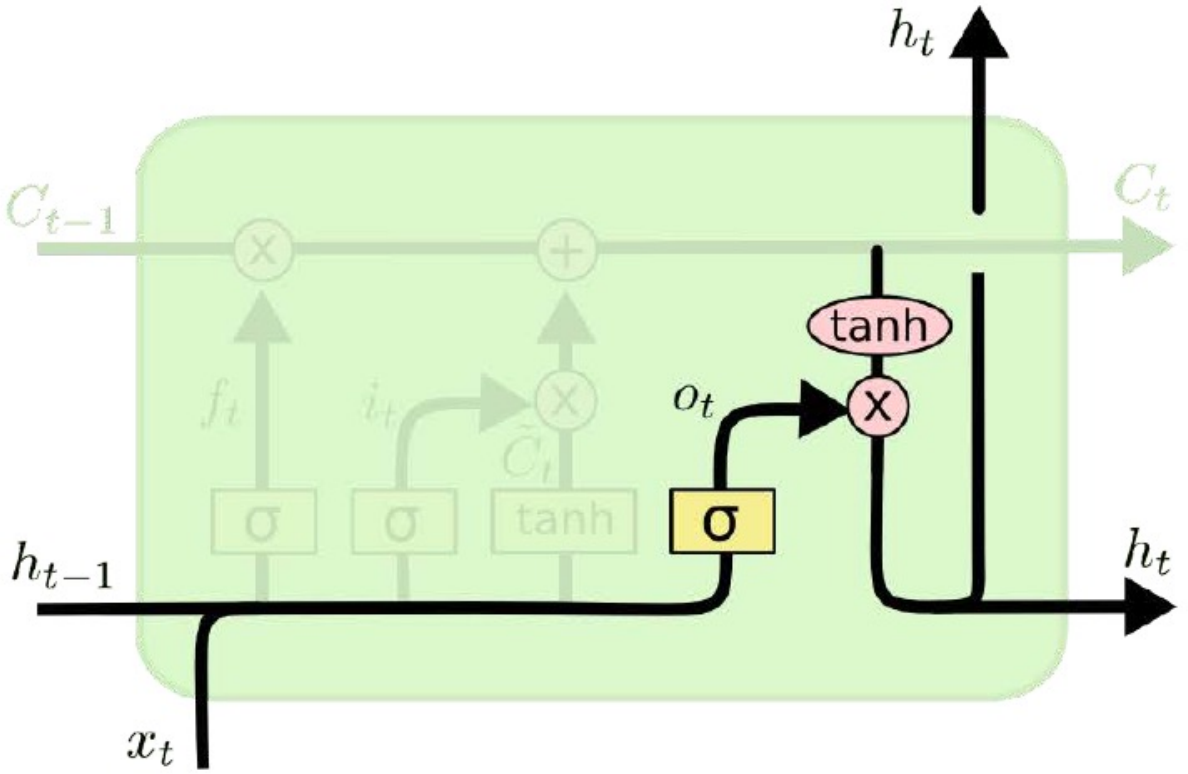
LSTM为什么在长序列数据的处理比RNN更优秀？

长期依赖：LSTM能记住多远？

经典例子：
"那只在院子里追着松鼠跑的猫，__饿了。"
（空格距"猫"已有10个词，RNN会遗忘"猫"是主语）

LSTM如何做到？
- 遗忘门主动决定保留重要语义
- 细胞状态c就像一条"记忆高速公路"
  可以把"猫"的信息无损保存到句末

实验结果：
LSTM在需要500+步以上长期依赖的任务中，准确率比RNN高30%以上！



长短时记忆神经网络（LSTM）

LSTM为什么在长序列数据的处理比RNN更优秀？

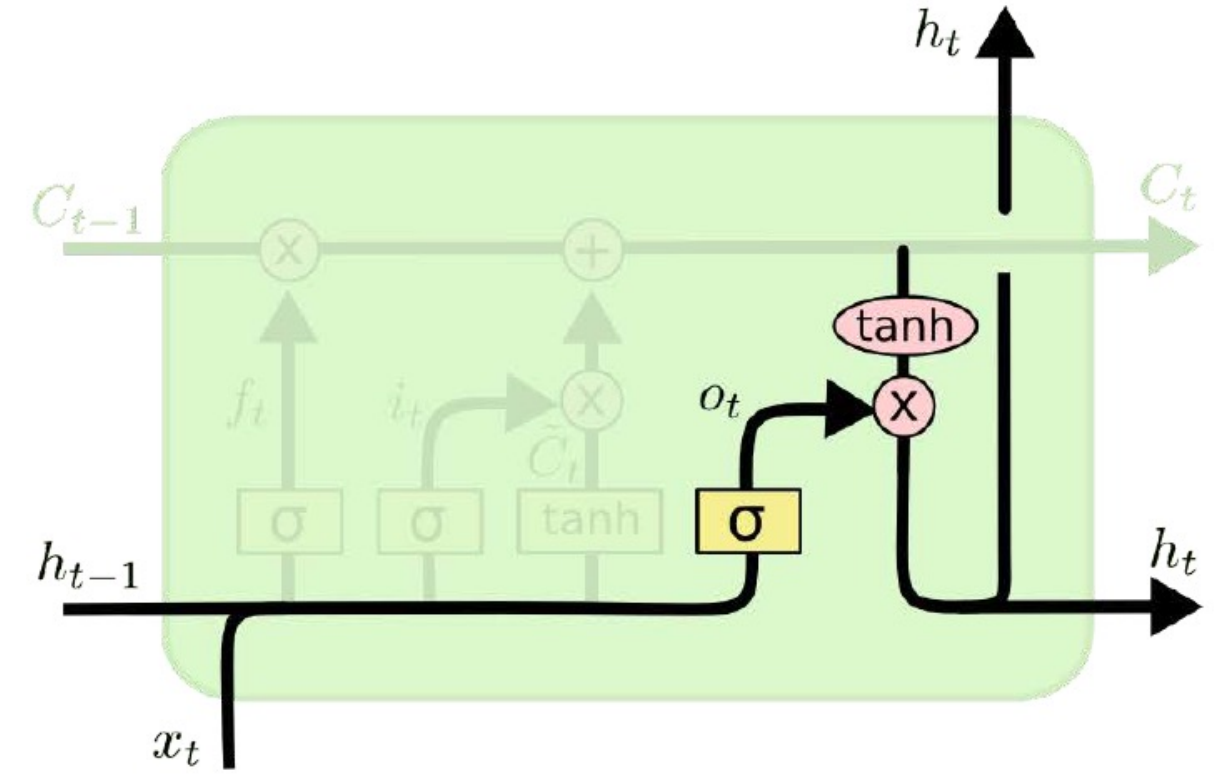
信息流可控性：三门协作

不同任务，门的行为不同：

情感分析（关键词决定情感）：
→ 遗忘门关注"否定词"出现时大幅重置
→ 输入门对"糟糕""完美"等词赋予高权重

机器翻译（保留全句结构）：
→ 遗忘门维持较高值（f≈0.8~0.9）
→ 逐步积累语法结构信息

RNN没有这种"选择性"，只能对所有信息一视同仁地加权叠加。



长短时记忆神经网络（LSTM）

LSTM的训练：整体流程

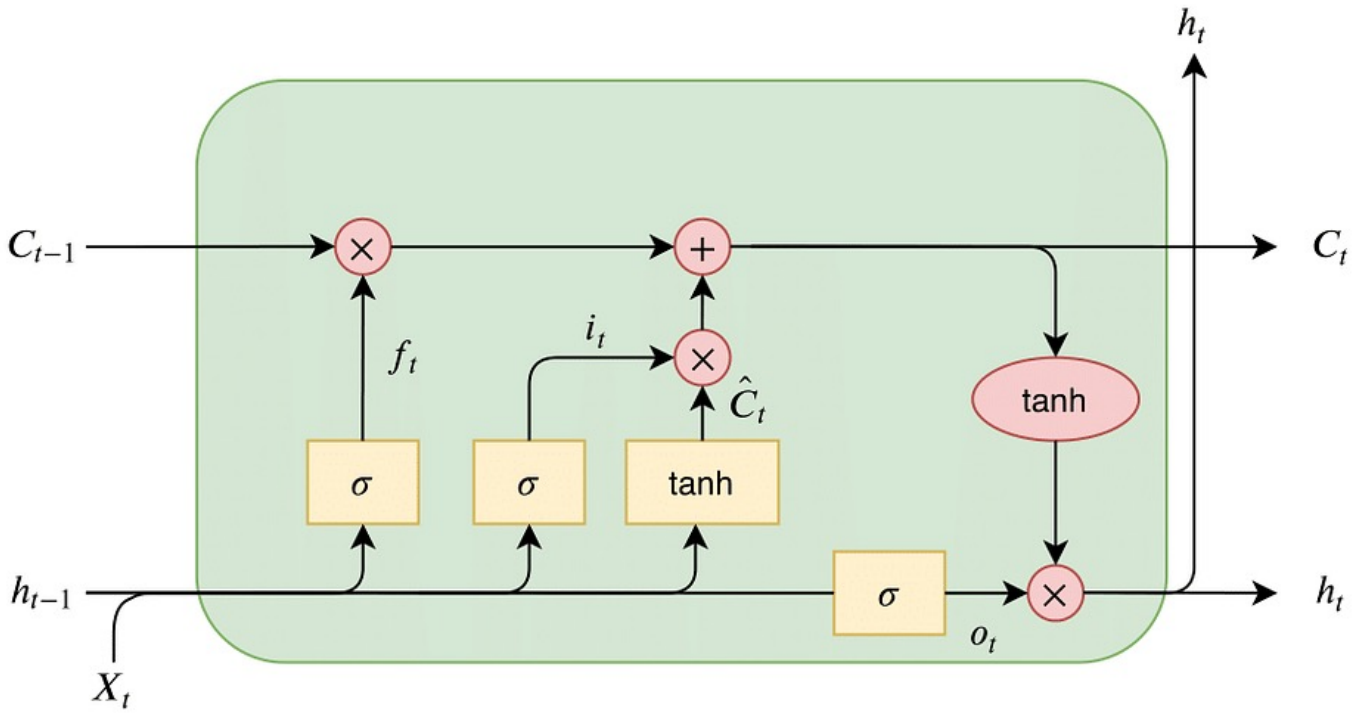
训练整体流程：

① 前向传播
逐时间步计算 h(t), c(t), y(t)

② 计算损失
L = 1/T · Σ Loss(y(t), label(t))

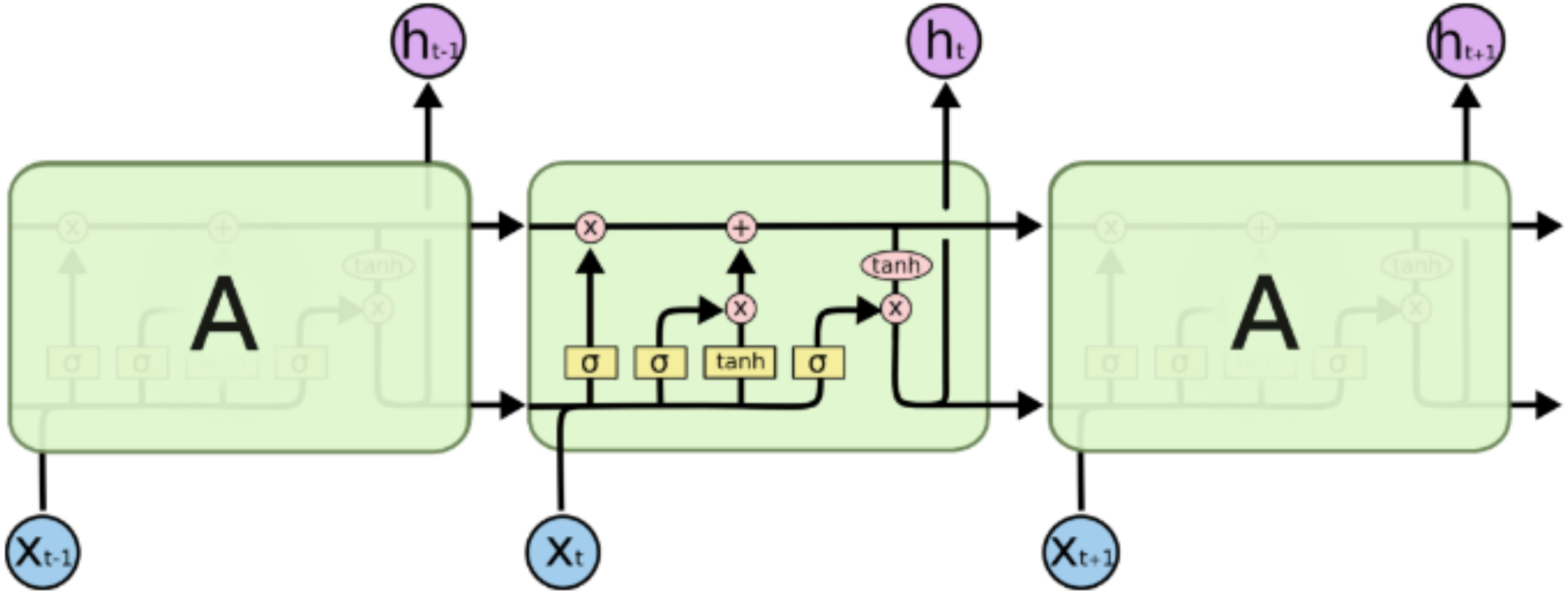
③ BPTT反向传播
梯度沿时间反向流，
同时也通过门流入c和h

④ 参数更新
Adam/SGD更新所有W, U, b
(遗忘/输入/输出门各自独立更新)



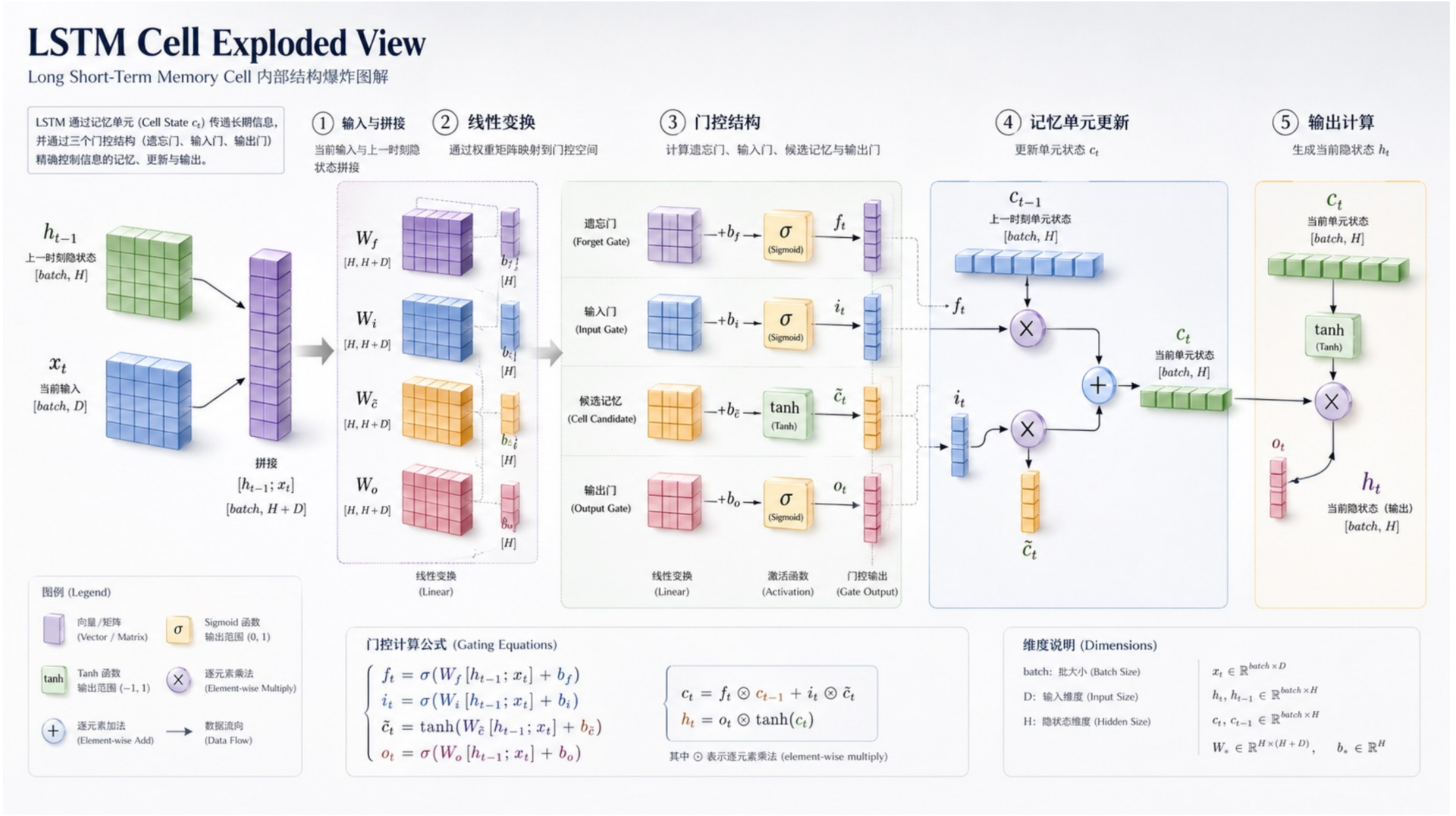
长短时记忆神经网络（LSTM）

LSTM的训练：前向传播 -> 计算损失 -> 反向传播



图解要点: 前向→损失→BPTT梯度反传→更新W (4套权重矩阵:Wf,Wi,Wc,Wo) | LSTM参数量 ≈ 4 × RNN参数量, 训练更慢, 但效果更好

长短时记忆神经网络（LSTM）：具体实现



目录

长短时记忆神经网络（LSTM）的应用

股票预测　　**情感分析**

机器翻译　　**词性标注**

习题

1、什么是LSTM？它与普通RNN的主要区别是什么？

LSTM（长短期记忆网络）是一种特殊的RNN，通过引入细胞状态和门机制（遗忘门、输入门、输出门）来解决普通RNN的梯度消失问题，能够更好地捕捉长距离依赖。

2、为什么LSTM能解决RNN的梯度消失问题？

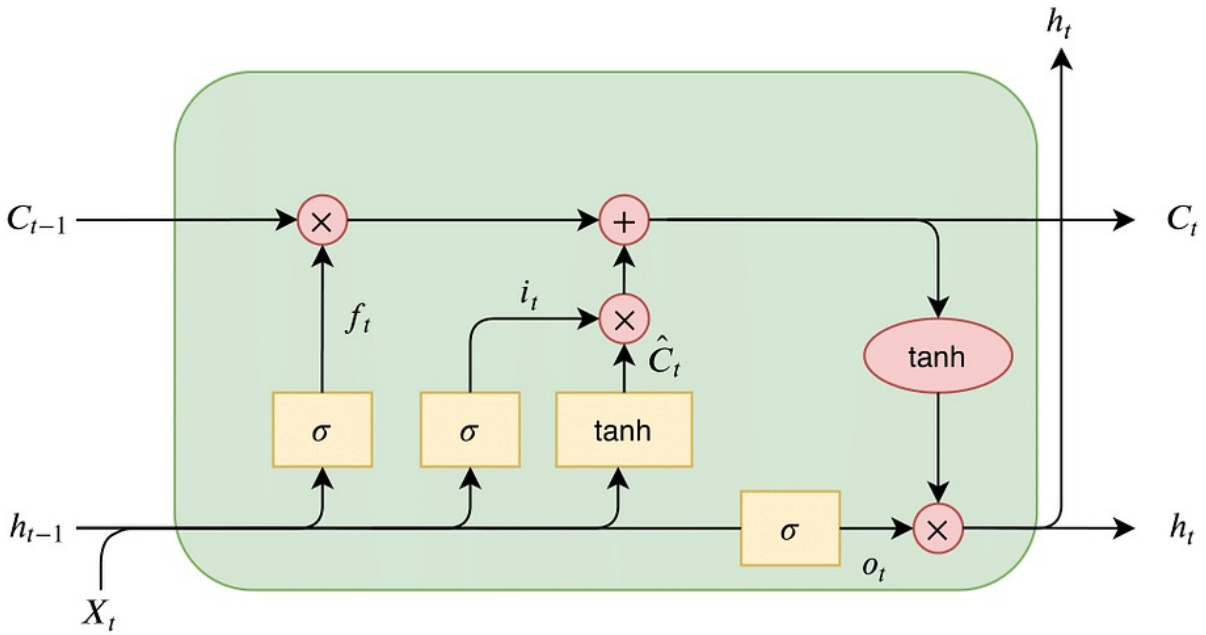
LSTM通过细胞状态传递信息，细胞状态的梯度可以通过时间步直接传播，而不需要经过非线性变换。门机制（遗忘门、输入门、输出门）控制信息的流动，使得梯度在传播过程中不会迅速衰减，从而缓解梯度消失问题。

3、LSTM中的“门机制”包括哪些门？它们的作用分别是什么？

输入门：决定哪些新信息被加入到细胞状态。
遗忘门：决定哪些旧信息应从细胞状态中移除。
输出门：决定哪些信息从细胞状态转移到隐藏状态，用于输出。

习题

4、看图推导LSTM的前向传播公式



- 遗忘门：$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- 输入门：$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- 输出门：$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
- 候选细胞状态：$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- 细胞状态：$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$
- 隐藏状态：$h_t = o_t \cdot \tanh(C_t)$

里程碑文献·序列建模演化史

1986 Rumelhart et al. — BPTT，奠定RNN训练基础

1997 Hochreiter & Schmidhuber — LSTM（Neural Computation）

2014 Cho et al. — GRU + Encoder-Decoder（arXiv:1406.1078）

2014 Bahdanau et al. — Attention机制（arXiv:1409.0473）

2015 Sutskever et al. — Sequence to Sequence（NeurIPS）

2017 Vaswani et al. — "Attention is All You Need"，Transformer

2018 BERT / GPT-2 — 预训练大模型时代开启

2023+ Mamba / RWKV — 状态空间模型，挑战Transformer

GRU：门控循环单元（Cho et al., 2014）

设计初衷：保留LSTM的门控思想，减少参数量，加速训练

GRU 只有两个门（LSTM有三个）：

- 重置门 $r(t) = \sigma(W_r \cdot [h(t-1), x(t)])$　决定「忘记多少过去」
- 更新门 $z(t) = \sigma(W_z \cdot [h(t-1), x(t)])$　决定「保留多少旧状态」

候选状态：$\tilde{h}(t) = \tanh(W \cdot [r(t) \odot h(t-1), x(t)])$

输出状态：$h(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot \tilde{h}(t)$

关键区别：GRU 无独立细胞状态 c(t)，隐状态 h(t) 同时承担长短期记忆

参数量：约为相同规模LSTM的 3/4，训练速度更快

实验结论（Chung et al., arXiv:1412.3555）：GRU与LSTM性能相当，

在数据量较少或序列较短时 GRU 往往更优

GRU vs LSTM — 如何选择？

对比维度	LSTM	GRU
门的数量	3个（输入/遗忘/输出）	2个（重置/更新）
记忆机制	独立细胞状态 c(t) + 隐状态 h(t)	仅隐状态 h(t)
参数量	较多（~4×隐层维度²）	较少（~3×隐层维度²）
训练速度	相对较慢	更快（约快25-30%）
适用场景	长序列、复杂时序、大数据	短序列、小数据集、资源受限
论文出处	Hochreiter & Schmidhuber 1997	Cho et al. 2014

实践建议：先用GRU快速验证，若性能不足再换LSTM。两者在大多数任务上差异在1%以内（Chung et al., 2014）。

Attention机制 — 打破信息瓶颈

Seq2Seq 的瓶颈问题

- 所有输入信息压缩在一个固定长度向量 c

→ 长句信息严重丢失

Bahdanau（2014）的解决方案

- Encoder 生成每步隐状态 $h_1, h_2, ..., hT$
- Decoder 每步计算注意力权重：
- $\alpha(t,i) = \text{softmax}(\text{score}(s(t), h_i))$
- 上下文：$c(t) = \Sigma_i \alpha(t,i) \cdot h_i$

→ **动态聚焦输入的不同部分！**

直觉类比

翻译「猫坐在垫子上」时：

- 生成「cat」时注意「猫」
- 生成「mat」时注意「垫子」

不再被迫把整句话塞进一个向量！

深远影响

- 机器翻译 BLEU 分数大幅提升
- 产生可解释的对齐权重可视化
- 直接启发了 Transformer（Vaswani et al., 2017）
- ChatGPT 等 LLM 的技术根源

从RNN到Transformer：序列建模的演化之路

RNN
（1986）
时序建模
有梯度消失

→

LSTM / GRU
（1997/2014）
门控机制
长期记忆

→

Attention
（2014）
动态聚焦
打破瓶颈

→

Transformer
（2017）
并行计算
全局建模

为什么 Transformer 取代 LSTM 成为 NLP 主流？

- 并行化：RNN/LSTM 必须顺序计算，Transformer 所有时间步同时计算 → GPU利用率提升100x
- 长程依赖：Self-Attention 直接连接任意两个位置，路径长度 O(1)，LSTM 路径 O(n)
- 但LSTM并未消亡：边缘设备推理、流式处理、小数据场景仍是RNN/LSTM的主场

LSTM/RNN 前沿应用领域（2020-2024）

医疗健康

- 电子病历（EHR）序列预测：LSTM预测ICU患者死亡风险，AUC可达0.87+
- 心电图（ECG）异常检测：双向LSTM实时识别心律失常
- 临床NLP：BERT/BiLSTM提取医学报告中的实体关系

自然语言处理（过渡时期）

- ELMo（2018）：双向LSTM生成上下文词向量，开启预训练语言模型先河
- 至今仍用于资源受限语言的情感分析、命名实体识别

工业与物联网

- 预测性维护：LSTM检测传感器时序数据中的设备故障前兆
- 能耗预测：LSTM与CNN混合模型优化建筑能耗管理

综述推荐：LSTM Networks: A Comprehensive Survey（MDPI AI, 2025）

课堂讨论：LSTM / RNN 还活着吗?

Transformer 的优势

- 全局注意力：一步捕获所有位置关系
- 并行计算：训练速度快 100x 以上
- 可扩展：参数量越大效果越好
- GPT / BERT / LLaMA 均基于此架构

LSTM / RNN 仍然有价值的场景

- 在线/流式推理：逐步接收数据（Transformer需完整序列）
- 边缘设备：内存 < 1GB 时RNN更省资源
- 时间序列小数据：过拟合风险低
- 强化学习：部分环境需顺序决策

新兴挑战者

Mamba（2023, Gu & Dao）

- 状态空间模型 + 选择性机制
- 线性复杂度，胜过同规模Transformer
- 被视为LSTM精神的现代复兴

RWKV（2023）

- 将Transformer的并行性与RNN的顺序推理结合
- 可作语言模型，参数达14B

思考题：

学了LSTM，为什么还值得深入？

→ 它是理解注意力和Transformer的必经之路！

动手实践：PyTorch 实现 LSTM / GRU

LSTM 时序分类（单向）

```
import torch
import torch.nn as nn

class LSTMClassifier(nn.Module):
    def __init__(self, input_size, hidden_size,
                 num_layers, num_classes):
        super().__init__()
        self.lstm = nn.LSTM(
            input_size, hidden_size,
            num_layers, batch_first=True
        )
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        # x: (batch, seq_len, input_size)
        out, (h_n, c_n) = self.lstm(x)
        # 取最后时间步的隐状态
        return self.fc(out[:, -1, :])

model = LSTMClassifier(28, 128, 2, 10)
```

GRU 替换只需一行

```
class GRUClassifier(nn.Module):
    def __init__(self, input_size, hidden_size,
                 num_layers, num_classes):
        super().__init__()
        # 只改这一行！
        self.gru = nn.GRU(
            input_size, hidden_size,
            num_layers, batch_first=True
        )
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        out, h_n = self.gru(x)
        # GRU 没有 c_n ！
        return self.fc(out[:, -1, :])

model = GRUClassifier(28, 128, 2, 10)
```

关键参数：hidden_size（记忆容量）| num_layers（深度）| batch_first=True（推荐开启）
双向LSTM：bidirectional=True → 输出维度×2，可捕获未来上下文

课堂思考题

①梯度爆炸时，遗忘门如何影响训练稳定性?

②GRU 无输出门，如何控制「对外暴露什么」？

③LSTM 如果去掉遗忘门会怎样？三个门缺一不可吗?

④双向LSTM（BiLSTM）比单向有什么优势？适合哪些任务?

⑤什么情况下堆叠多层LSTM比单层更好？有哪些风险?

请先独立思考 💭

后面每页给出参考答案

① 梯度爆炸时，遗忘门如何影响训练稳定性?

【根本原因】BPTT中梯度连乘传播，权重矩阵最大特征值>1时，梯度随时间步指数增大。

【遗忘门的双重作用】
- 遗忘门接近0时：细胞状态被清零，切断梯度回传路径，间接缓解爆炸
- 遗忘门饱和（接近1）时：梯度几乎不衰减，爆炸风险依然存在

【实践解法】梯度裁剪（Gradient Clipping）
当 $\|g\| > \theta$ 时令 $g \leftarrow g \cdot \theta / \|g\|$，把梯度范数限制在阈值内

结论：遗忘门是缓解梯度消失的关键工具，但对爆炸防护有限，需配合梯度裁剪使用。

② GRU 无输出门，如何控制「对外暴露什么」？

【LSTM的方式】输出门 $o(t) = \sigma(W_o \cdot [h(t-1), x(t)])$，隐状态 $h(t) = o(t) \odot \tanh(c(t))$
→ 输出门过滤细胞状态，$h(t) \neq c(t)$

【GRU的方式】没有独立细胞状态，隐状态 $h(t)$ 直接就是对外输出。
控制权交给「更新门」$z(t)$：
$h(t) = (1-z(t)) \odot h(t-1) + z(t) \odot \tilde{h}(t)$

- $z(t)$ 接近 0 → 保留旧状态，几乎不暴露新信息
- $z(t)$ 接近 1 → 完全替换为候选状态，充分暴露新信息

本质：GRU 用「是否更新」代替了「是否输出」，表达能力略弱，但参数更少、训练更快。

③ LSTM 如果去掉遗忘门会怎样？三个门缺一不可吗？

三个门的分工

- 遗忘门：决定丢弃细胞状态中哪些旧信息
- 输入门：决定写入哪些新信息进入细胞状态
- 输出门：决定对外暴露细胞状态中哪些信息作为 h(t)

去掉遗忘门会怎样？

- 细胞状态只会累加，永远无法清除过时信息
- 长序列中 c(t) 会无限增长，导致梯度爆炸或信息污染
- 实验验证（Jozefowicz et al., 2015）：遗忘门是三个门中最重要的

GRU 的启示：

GRU 去掉了输出门，将遗忘+输入合并为更新门，性能依然相当 → 输出门是三个门中最可省的。

④ 双向LSTM（BiLSTM）比单向有什么优势？适合哪些任务？

核心思想

用两个独立的LSTM分别从左→右、右→左处理序列，最后将两个隐状态拼接：

$h(t) = [\, h\rightarrow(t) \,;\, h\leftarrow(t) \,]$，输出维度×2

优势：同时获取过去和未来的上下文

- 例：「他去了银行——取钱还是河边？」单向LSTM读到「银行」时看不到后面，BiLSTM可以
- 表示能力更强，同等参数下分类准确率通常高2-5%

适合 vs 不适合的任务

✓ 适合：情感分析、命名实体识别、词性标注（整句已知）

✗ 不适合：语言生成、实时预测（未来数据不可见）

⑤ 什么情况下堆叠多层LSTM比单层更好？有哪些风险？

堆叠LSTM的原理

第 l 层的输出 h^l(t) 作为第 l+1 层的输入，逐层提取更抽象的时序特征。

适合加深的情况

- 任务复杂度高，需要捕捉多粒度时序规律（如语音识别：音素→词→句）
- 数据量足够大，能支撑更多参数不过拟合
- 单层LSTM已到性能瓶颈，加宽（增大hidden_size）也没有改善

风险与应对

- 过拟合：层数越多参数越多 → 配合层间 Dropout 缓解
- 训练变慢：层数翻倍则计算量翻倍，且无法并行
- 实践建议：先试2层，再试3-4层；超过4层极少有明显收益

谢谢！