2023秋

模式识别与深度学习 (14)

深度序列建模

左旺孟

综合楼712

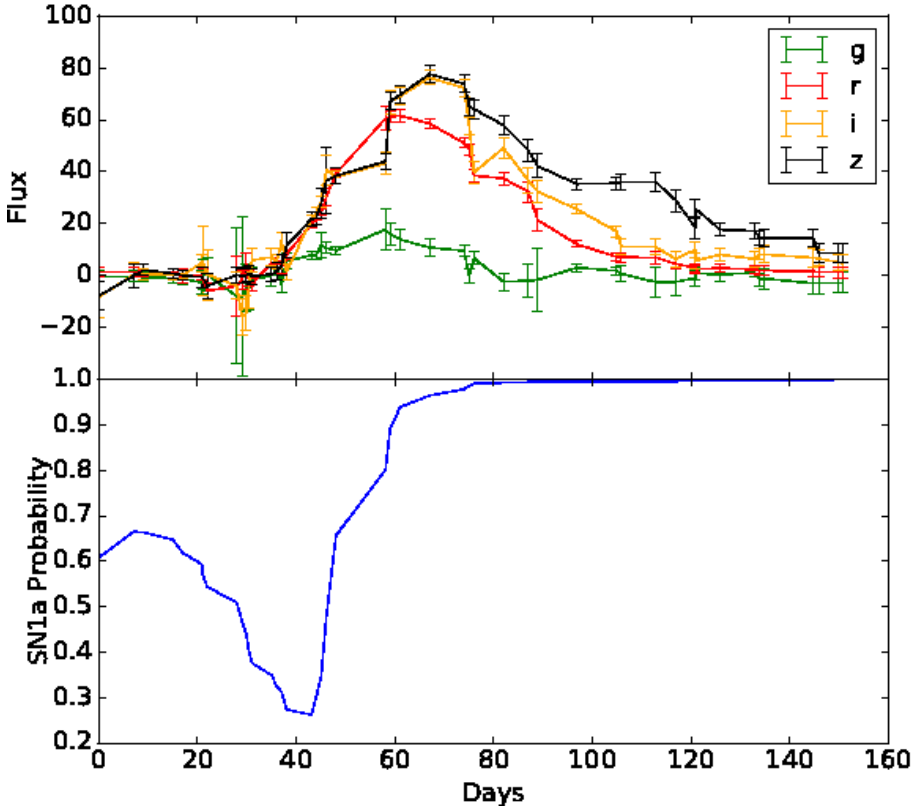
机器学习研究中心

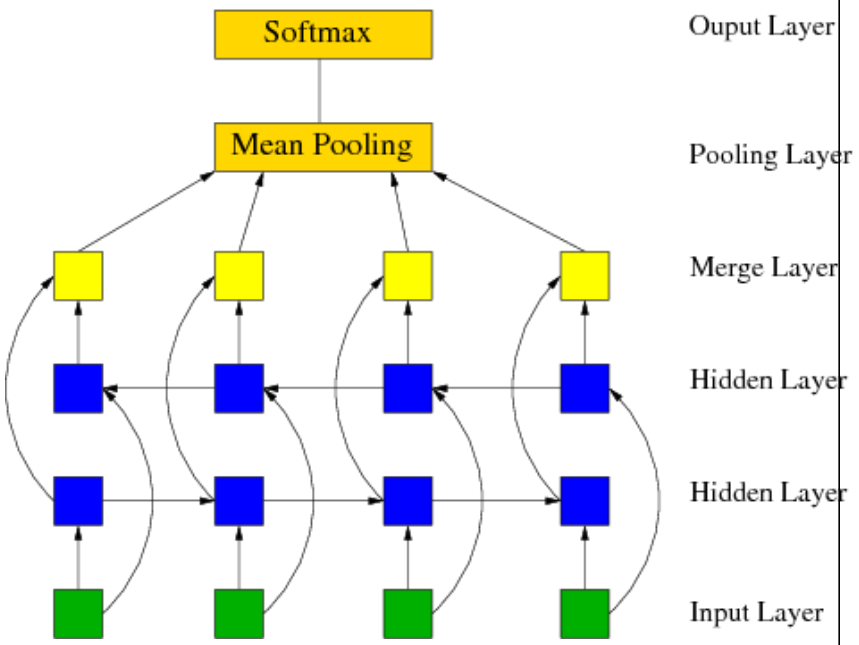
哈尔滨工业大学计算机学院

cswmzuo@gmail.com

13134506692

序列信号识别/分类


Flux
SN1a Probability
Days
g
r
i
z

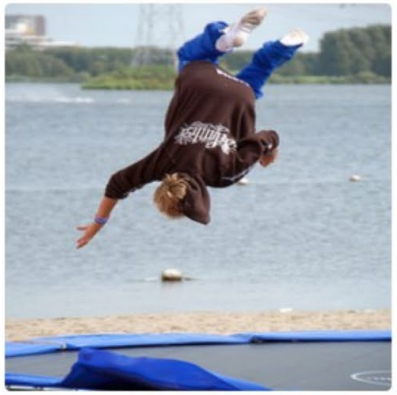

Softmax
Mean Pooling
Ouput Layer
Pooling Layer
Merge Layer
Hidden Layer
Hidden Layer
Input Layer

序列建模问题：图像->自然语言

"man in black shirt is playing guitar."

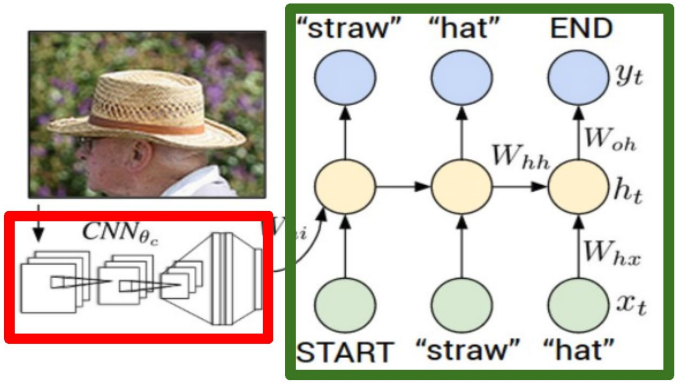
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."



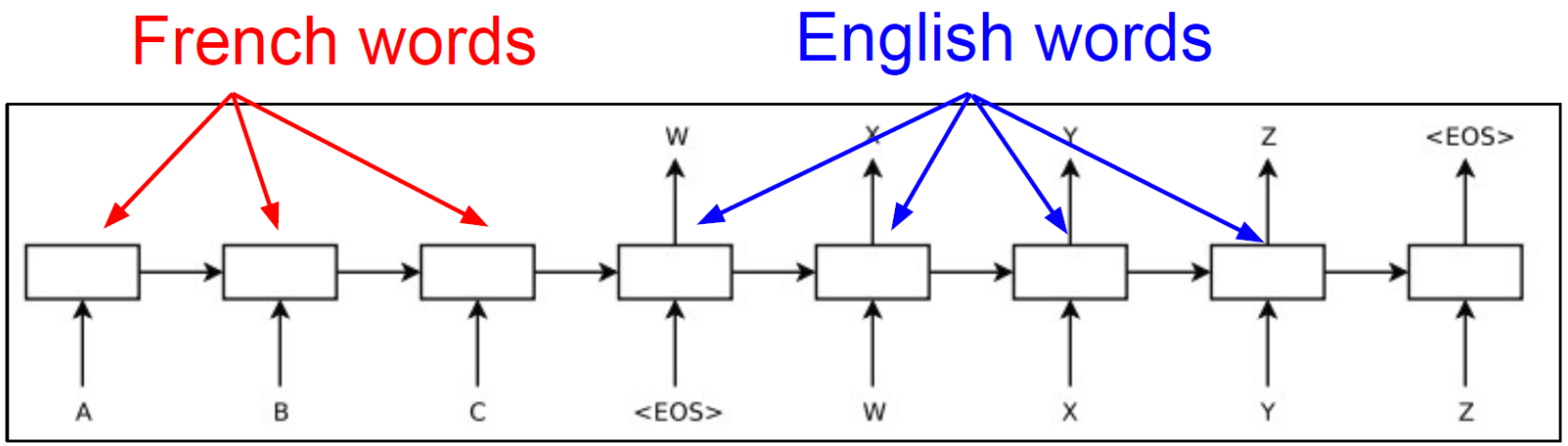
"boy is doing backflip on wakeboard."

Recurrent Neural Network

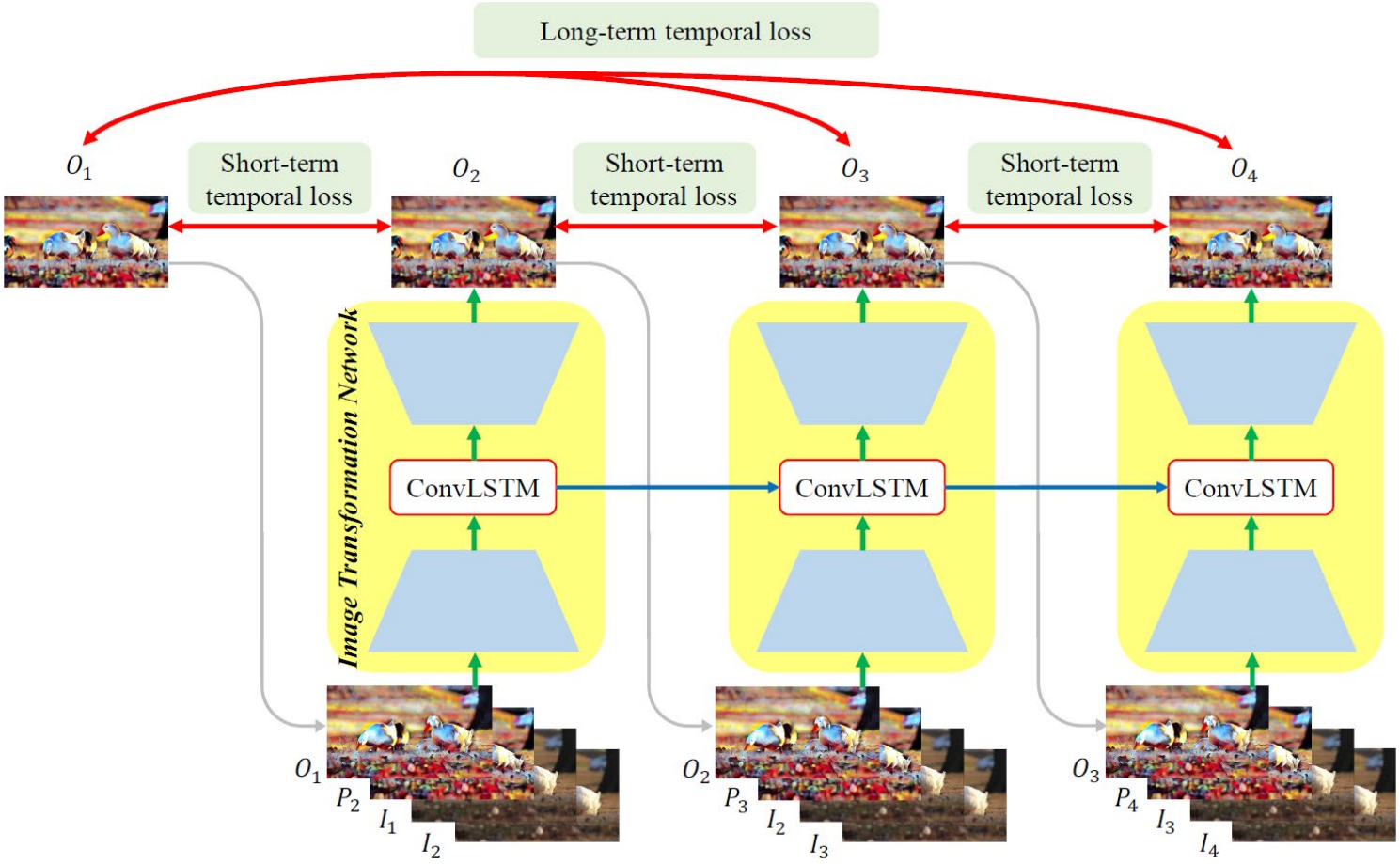


Convolutional Neural Network

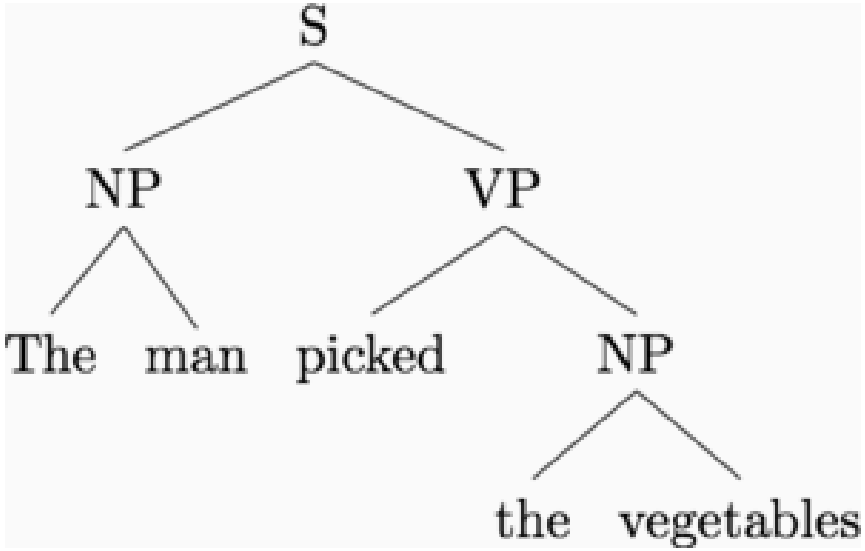
序列建模问题：机器翻译



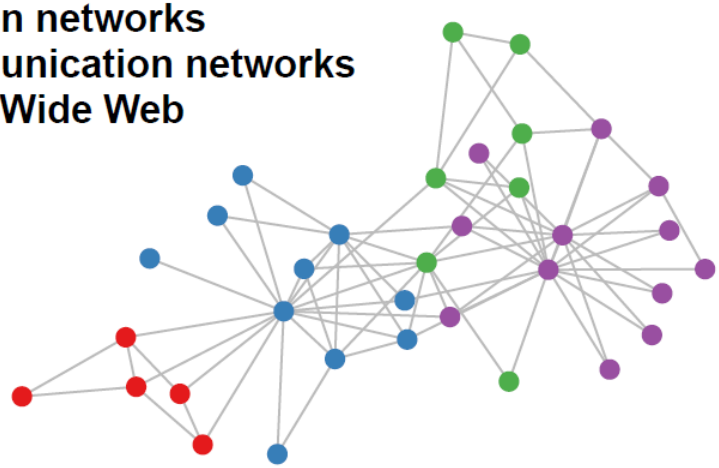
时间序列分析


Long-term temporal loss
O_1
Short-term temporal loss
O_2
Short-term temporal loss
O_3
Short-term temporal loss
O_4
Image Transformation Network
ConvLSTM
ConvLSTM
ConvLSTM
O_1
P_2
I_1
I_2
O_2
P_3
I_2
I_3
O_3
P_4
I_3
I_4

构建树和图



Social networks
Citation networks
Communication networks
World Wide Web



- 序列结构、树结构、图结构
- 输入、输出
- 结构化数据

序列建模

- 循环神经网络
- 递归神经网络
- 回声状态网络
- 记忆网络

循环神经网络

- **循环神经网络**（Recurrent NN）
- 双向RNN
- 序列到序列模型
- 长短期记忆（LSTM）、GRU

循环神经网络

- 循环神经网络
  - 用于处理序列$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)}$的神经网络
  - 处理长序列的能力
  - 处理变长序列的能力

  } 参数共享

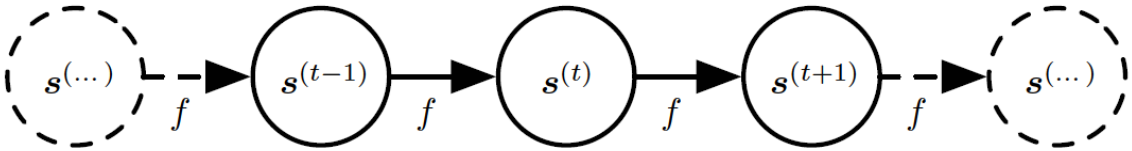
  - 通常在序列的小批量上进行
  - 时间t：序列中的位置

动态系统、计算图及其展开

- 计算图：一组计算结构的形式化表达
  - 输入、参数 -> 输出并计算损失
- 经典动态系统 $\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$
  - 展开（Unfolding）

$$\begin{aligned} \boldsymbol{s}^{(3)} &= f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) \\ &= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta}) \end{aligned}$$

  - 计算图展开



计算图及其展开

- 外部信号驱动的动态系统
  - 当前状态

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

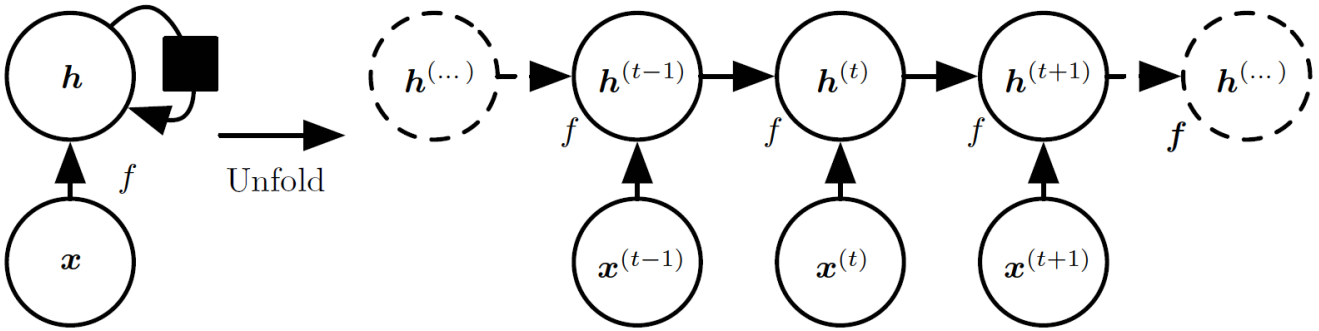
  - 可以包含整个过去系列的信息
  - 隐藏单元$\boldsymbol{h}$

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

  - 固定长度
  - 过去序列与任务有关的有损摘要
  - f: 转移函数

计算图及其展开

- 外部信号驱动的动态系统
  - 计算图及其展开

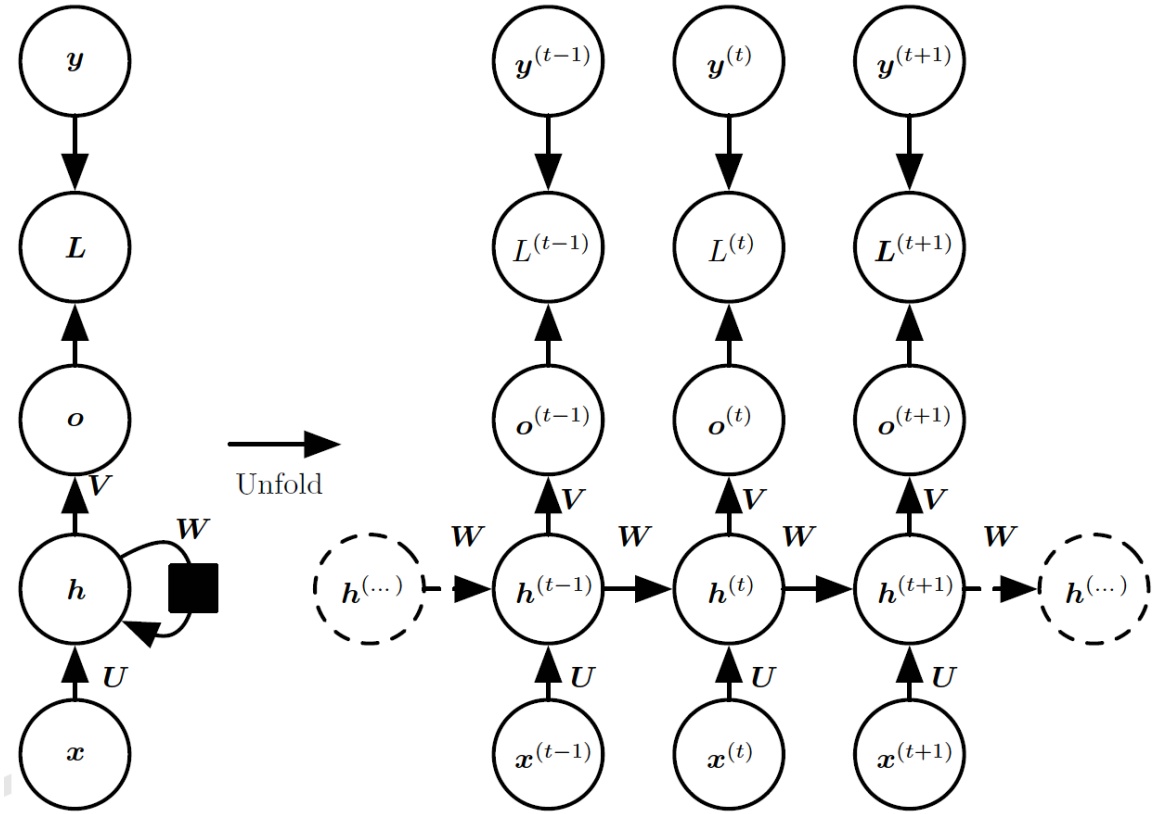


$$\boldsymbol{h}^{(t)} = g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)})$$
$$= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).$$

  - 为变长序列学习单一的共享模型
  - 没有考虑最终输出

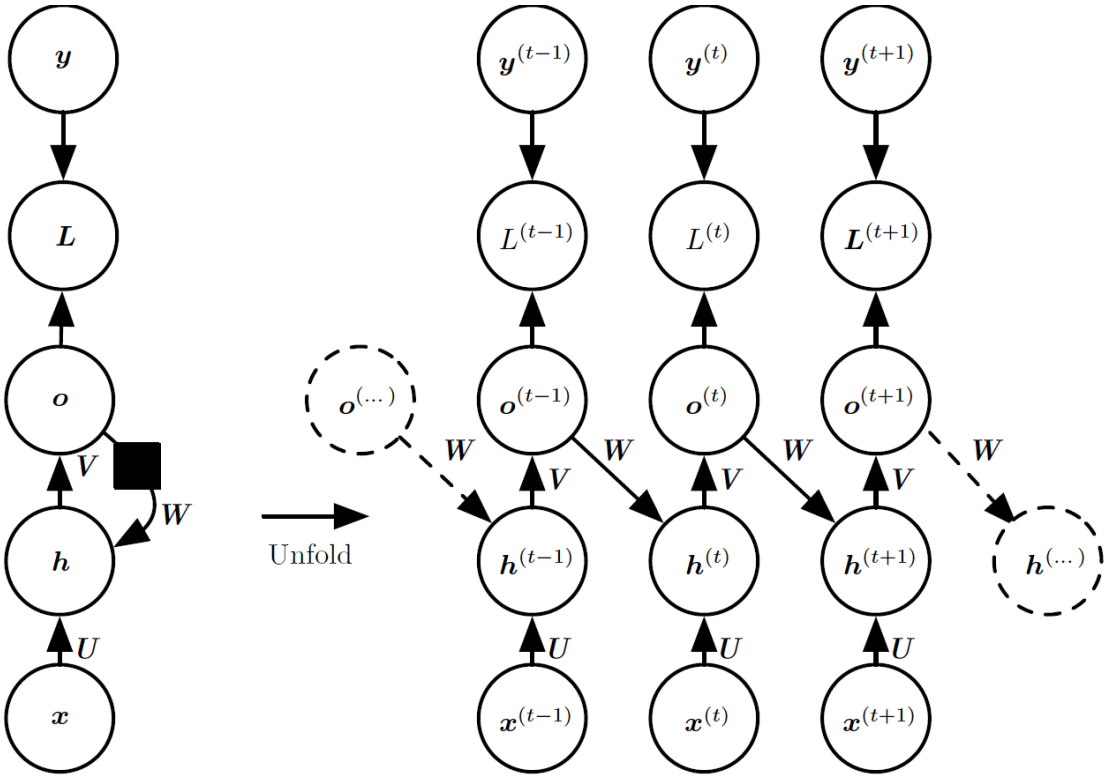
典型RNN设计模式1：考虑输出及损失

- 每个时间步都有输出，并且隐藏单元之间有循环连接的循环网络



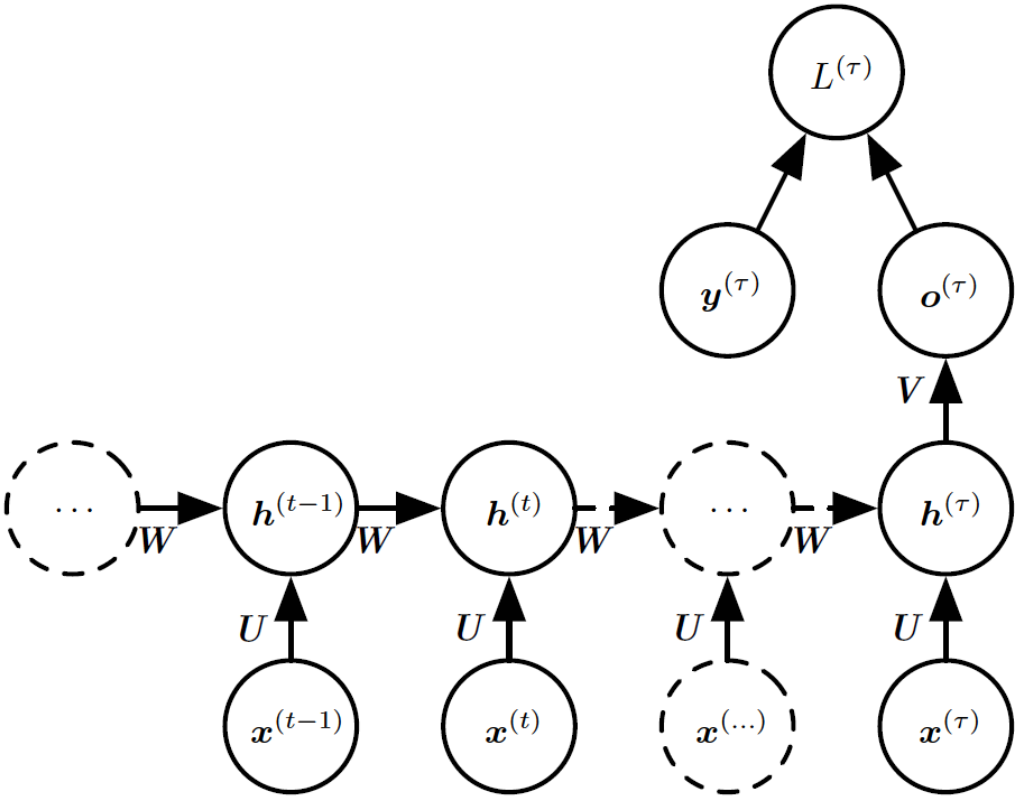
典型RNN设计模式2：考虑输出及损失

- 每个时间步都产生一个输出，只有当前时刻的输出到下个时刻的隐藏单元之间有循环连接的循环网络



典型RNN设计模式3：考虑输出及损失

- 隐藏单元之间存在循环连接，但读取整个序列后产生单个输出的循环网络



前向传播公式：以模式1为例

- 前向传播

$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}$$

- 损失函数

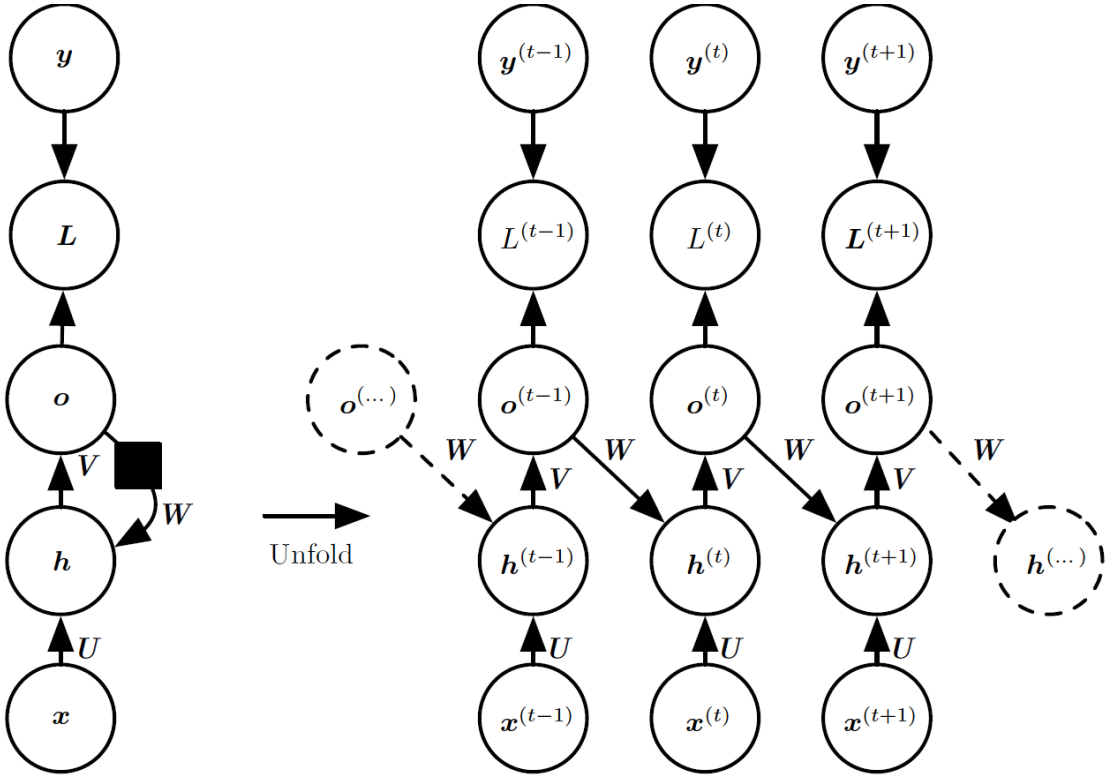
$$\begin{aligned}
&L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\
&= \sum_t L^{(t)} \\
&= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right),
\end{aligned}$$

- 梯度计算：通过时间反向传播（BPTT）
  - 计算复杂性： $\mathcal{O}(\tau)$
  - 存储复杂性： $\mathcal{O}(\tau)$

RNN训练

- 导师驱动过程
- 通过时间反向传播（BPTT）
- 作为有向图模型的循环网络
- 基于上下文的**RNN** 序列建模

RNN设计模式2



- 导师驱动过程：在时刻$t+1$ 接收真实值$y(t)$ 作为输入

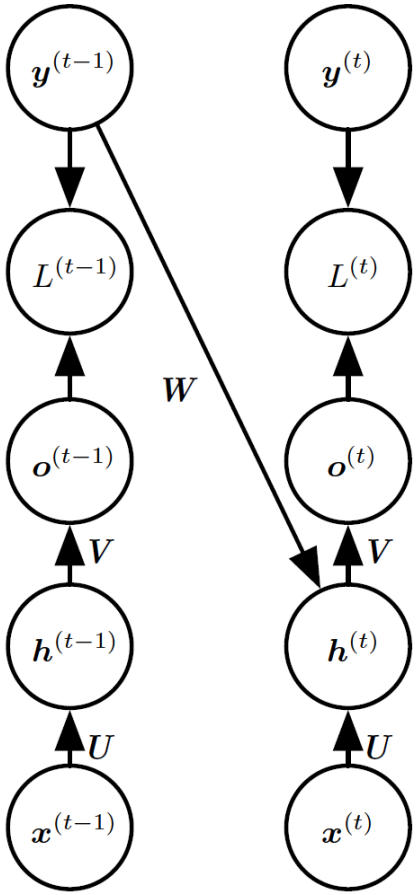
导师驱动过程

- 条件最大似然准则

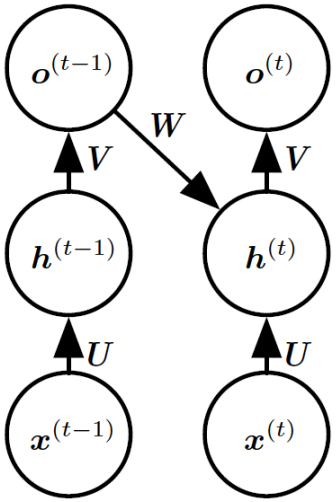
$$\log p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$
$$= \log p(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) + \log p(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$

- 避免通过时间反向传播

导师驱动过程



Train time



Test time

闭环模式训练到开环模式应用的不一致性

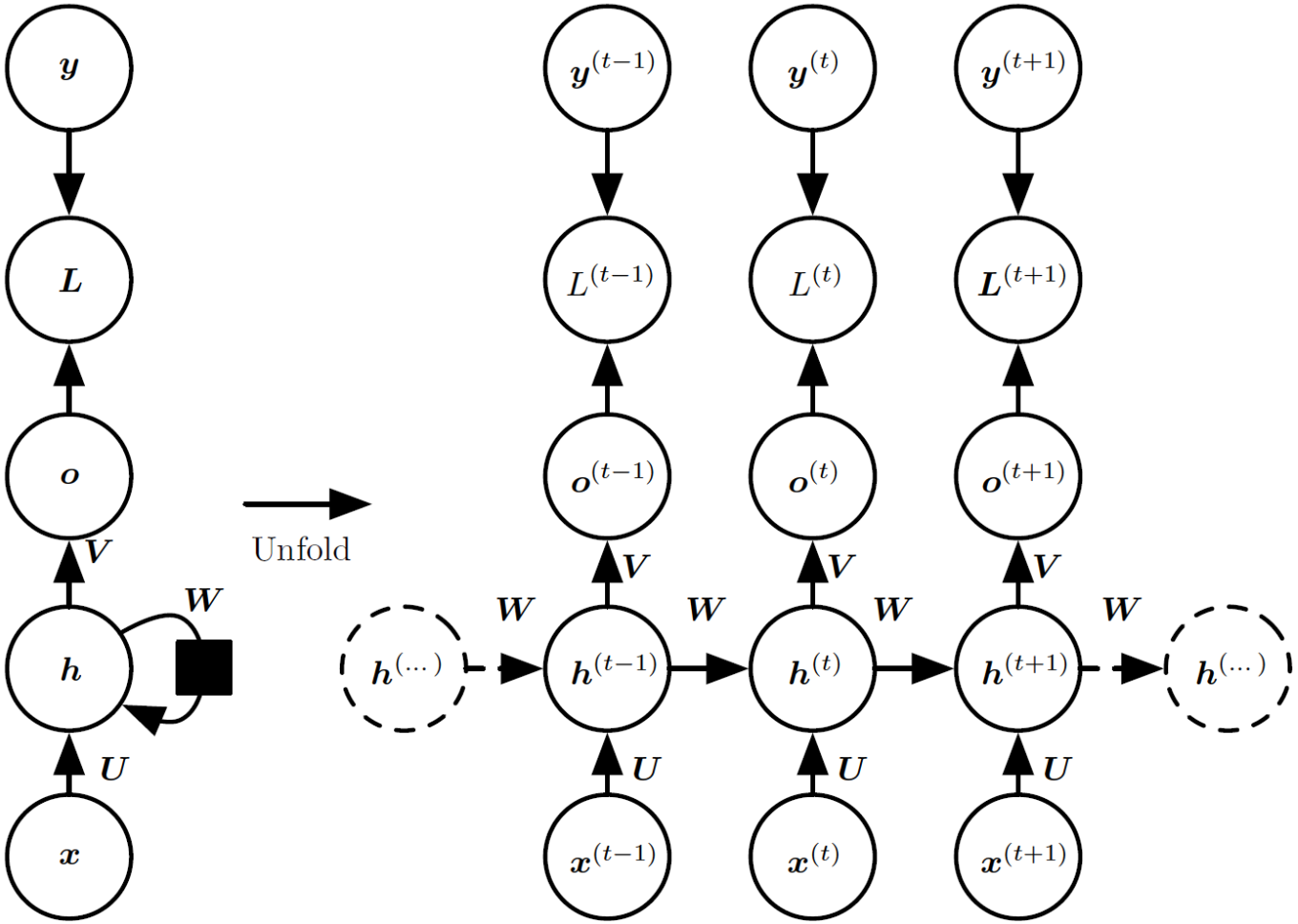
导师驱动过程：改进

- 同时使用导师驱动过程和自由运行的输入进行训练
- 随机选择生成值或真实的数据值作为输入
  - 结合课程学习：逐步增加使用生成值作为输入的概率

RNN训练

- 导师驱动过程
- 通过时间反向传播（BPTT）
- 作为有向图模型的循环网络
- 基于上下文的**RNN** 序列建模

BPTT：计算循环神经网络的梯度



BPTT：计算循环神经网络的梯度

$$
\begin{aligned}
&L\big(\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(\tau)}\},\{\boldsymbol{y}^{(1)},\ldots,\boldsymbol{y}^{(\tau)}\}\big)\\
&=\sum_t L^{(t)}\\
&=-\sum_t \log p_{\text{model}}\big(y^{(t)}\mid\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(t)}\}\big)
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b}+\boldsymbol{W}\boldsymbol{h}^{(t-1)}+\boldsymbol{U}\boldsymbol{x}^{(t)},\\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}),\\
\boldsymbol{o}^{(t)} &= \boldsymbol{c}+\boldsymbol{V}\boldsymbol{h}^{(t)},\\
\hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

- 参数：$\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{b}$ 和 $\boldsymbol{c}$
- 节点序列：$\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{o}^{(t)}$ 和 $L^{(t)}$

梯度计算: L -> o -> h -> x

- $\frac{\partial L}{\partial L^{(t)}} = 1$

- softmax 函数

$$(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}$$

- $\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= \boldsymbol{W}^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) \mathrm{diag}\left(1 - (\boldsymbol{h}^{(t+1)})^2\right) + \boldsymbol{V}^\top (\nabla_{\boldsymbol{o}^{(t)}} L),$$

梯度计算: 参数

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

$$
\nabla_{\boldsymbol{c}} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}}\right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L,
$$

$$
\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L,
$$

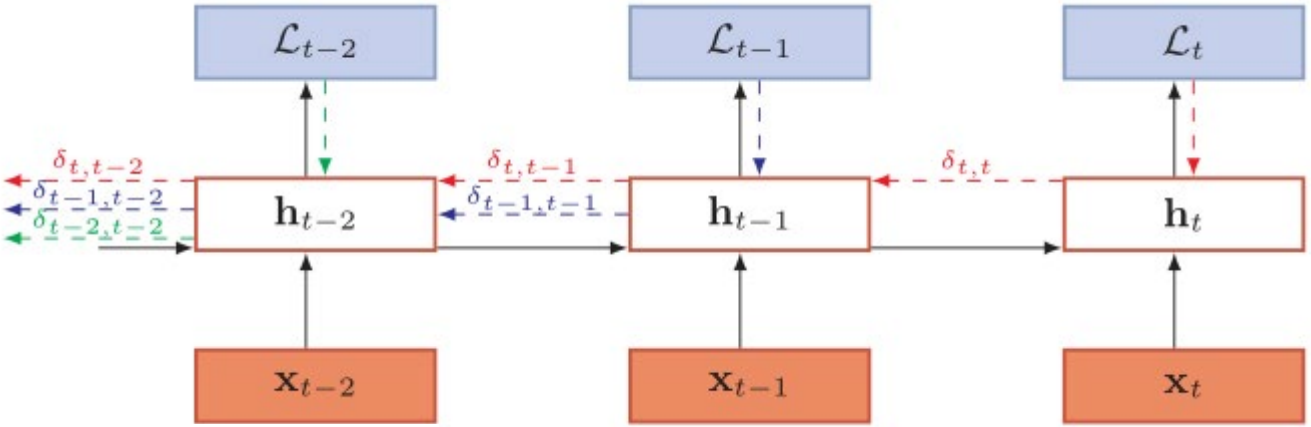
$$
\nabla_{\boldsymbol{V}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{\boldsymbol{V}} o_i^{(t)} = \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \boldsymbol{h}^{(t)\top},
$$

$$
\begin{aligned}
\nabla_{\boldsymbol{W}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \\
&= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)\top},
\end{aligned}
$$

$$
\begin{aligned}
\nabla_{\boldsymbol{U}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \\
&= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)\top},
\end{aligned}
$$

BPTT

$$\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(U\mathbf{h}_t + W\mathbf{x}_{t+1} + \mathbf{b})$$



$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T}\sum_{k=1}^{t} \delta_{t,k}\mathbf{h}_{k-1}^{\mathrm{T}}$$

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

$\delta_{t,k}$为第t时刻的损失对第k步隐藏神经元的净输出z_k的导数

RNN训练

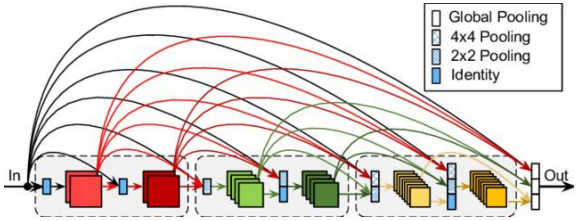
- 导师驱动过程
- 通过时间反向传播（BPTT）
- 作为有向图模型的循环网络
- 基于上下文的**RNN** 序列建模

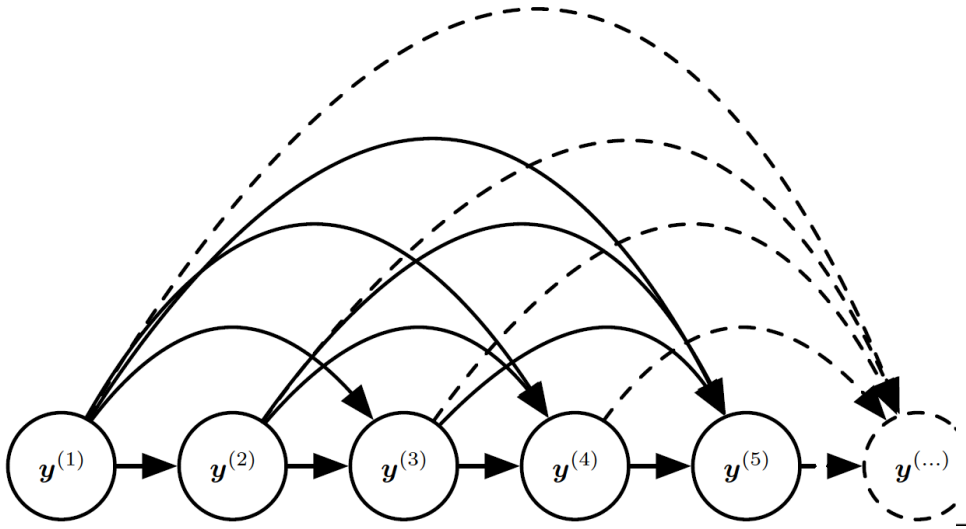
作为有向图模型的循环网络

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

$$\log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(t-1)})$$

- 有向图模型包含所有从过去$\boldsymbol{y}^{(i)}$ 到当前$\boldsymbol{y}^{(t)}$ 的边。





作为有向图模型的循环网络

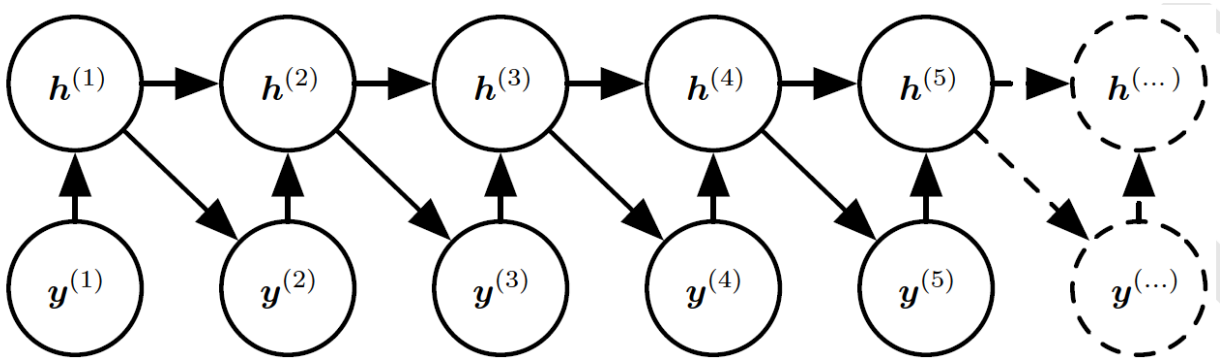
- 没有额外的输入**x**

$$P(\mathbb{Y}) = P(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\tau)}) = \prod_{t=1}^{\tau} P(\mathbf{y}^{(t)} \mid \mathbf{y}^{(t-1)}, \mathbf{y}^{(t-2)}, \ldots, \mathbf{y}^{(1)})$$

$$L = \sum_t L^{(t)}$$

$$L^{(t)} = -\log P(\mathrm{y}^{(t)} = y^{(t)} \mid y^{(t-1)}, y^{(t-2)}, \ldots, y^{(1)})$$

- 隐藏单元作为解耦单元

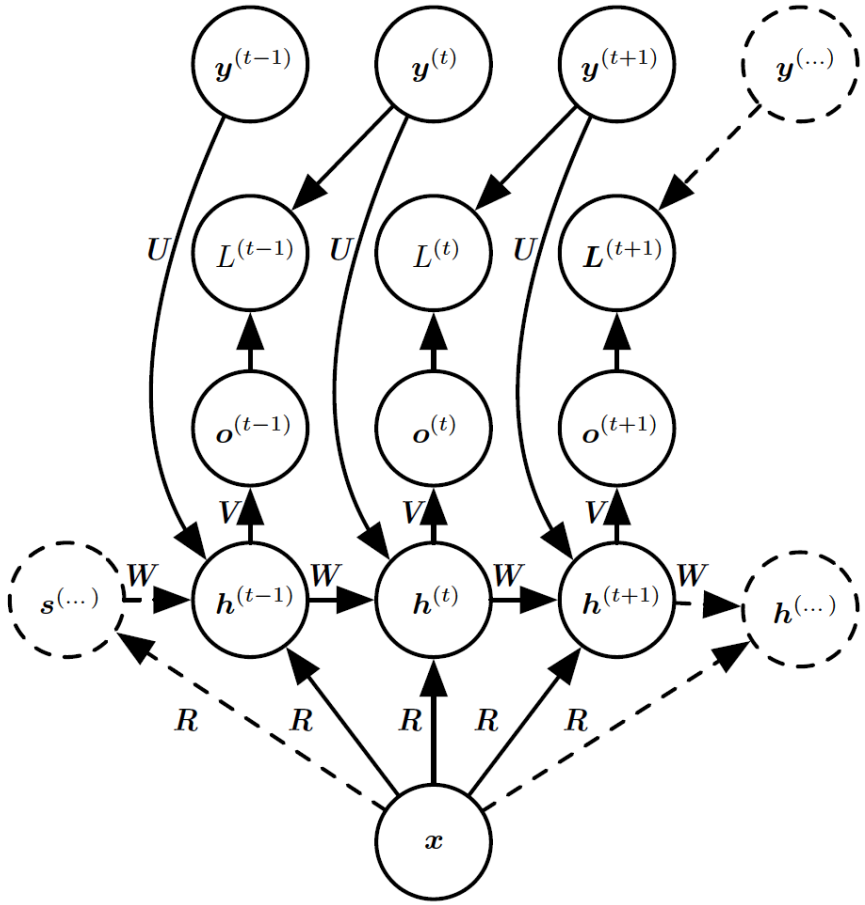


RNN训练

- 导师驱动过程
- 通过时间反向传播（BPTT）
- 作为有向图模型的循环网络
- 基于上下文的**RNN** 序列建模

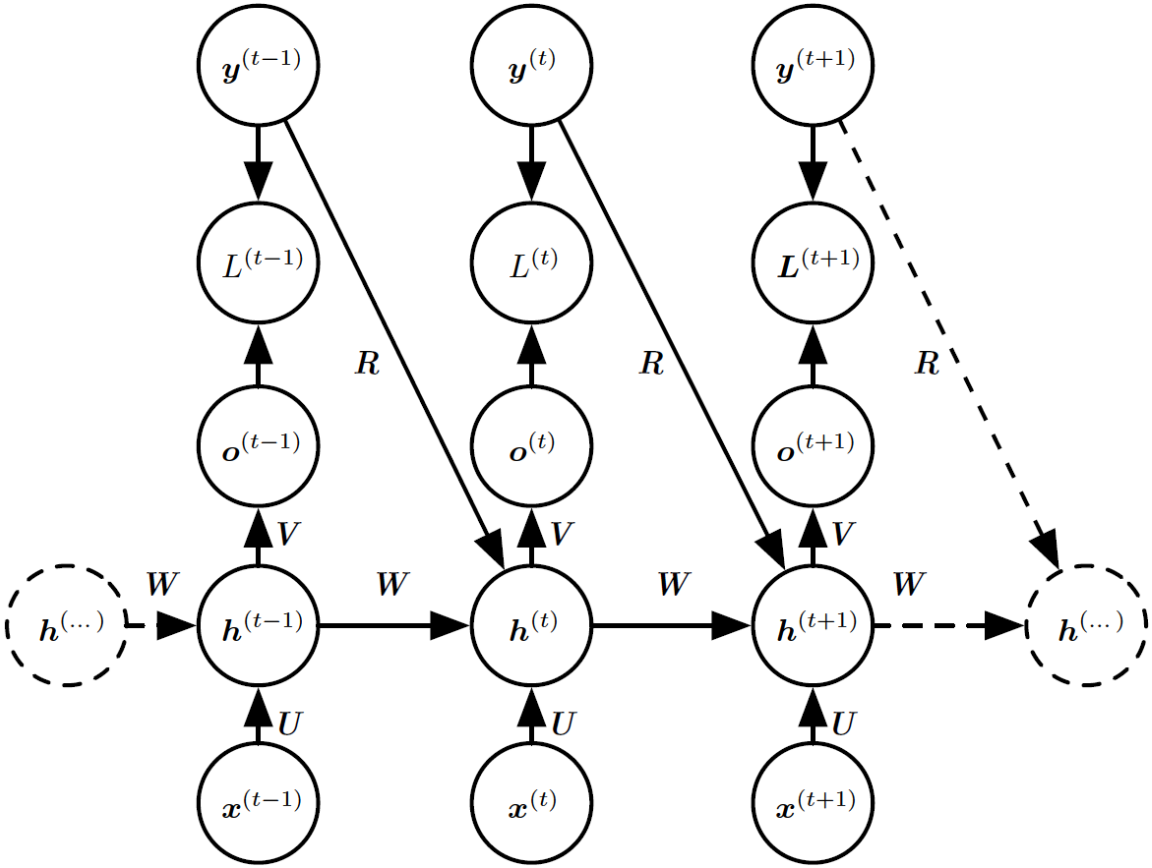
基于上下文的RNN 序列建模

- 只使用单个向量$\boldsymbol{x}$ 作为输入



基于上下文的RNN 序列建模

- 接收向量序列$\boldsymbol{x}^{(t)}$



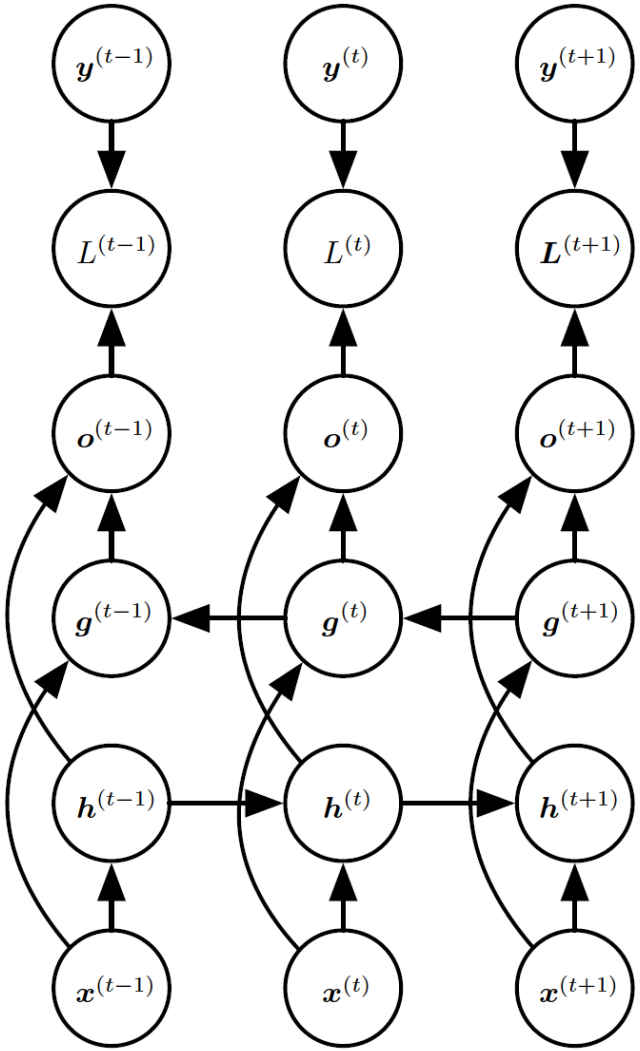
循环神经网络

- 循环神经网络（Recurrent NN）
- 双向RNN
- 序列到序列模型
- 长短期记忆（LSTM）、GRU

双向RNN

- 起因：要输出的$y^{(t)}$ 的预测可能依赖于整个输入序列
- 如：语音识别、图像自然语言描述、手写识别
- 双向RNN：结合时间上从序列起点开始移动的RNN 和另一个时间上从序列末尾开始移动的RNN

双向RNN



- 思考：二维
  - MRF/CRF

循环神经网络

- 循环神经网络（Recurrent NN）
- 双向RNN
- 序列到序列模型
- 长短期记忆（LSTM）、GRU

序列建模问题：机器翻译

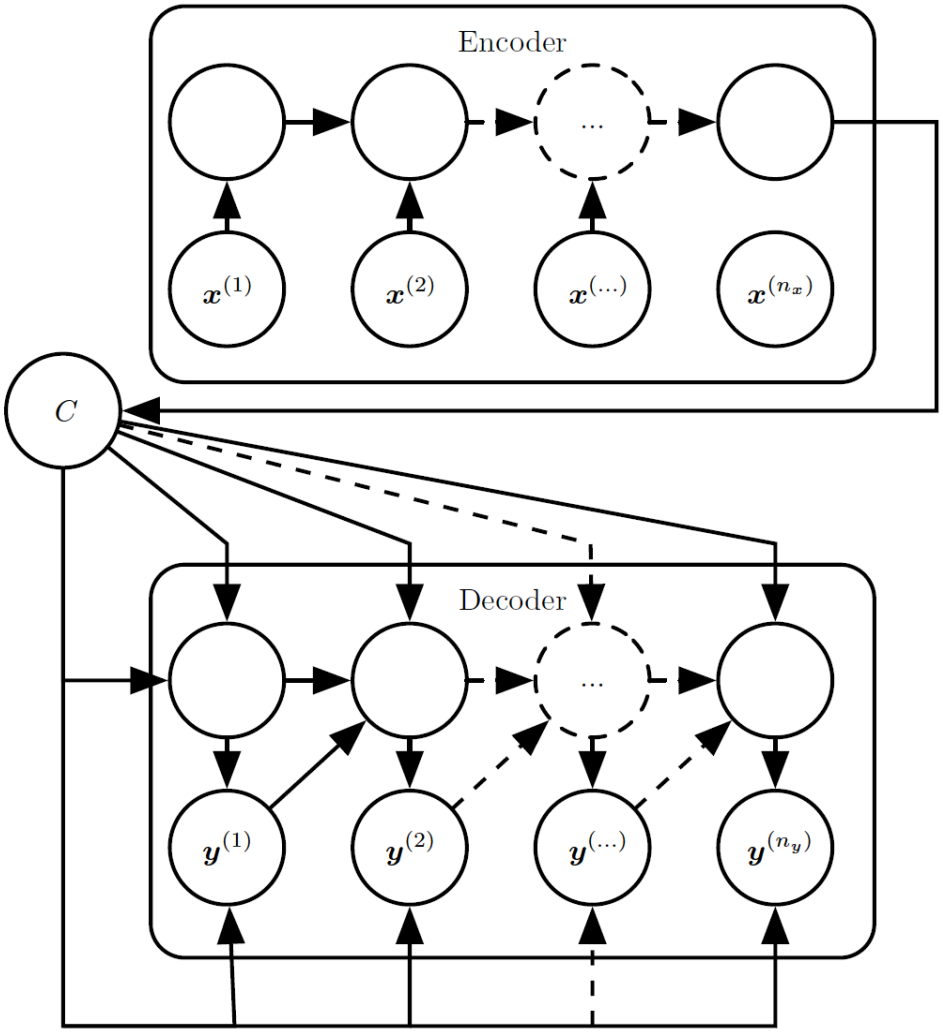
Input	Two field measurements for atmospheric fine particles were conducted in Baoan district of Shenzhen during the summer and winter in 2004.
Google	大气细颗粒两个现场测量在深圳市宝安区2004夏季和冬季期间进行。
Baidu	2004宝安区深圳夏季和冬季大气细颗粒物的两场测量。
Youdao	两个大气细粒子进行了实地测量在深圳宝安区2004年夏季和冬季。

序列到序列模型

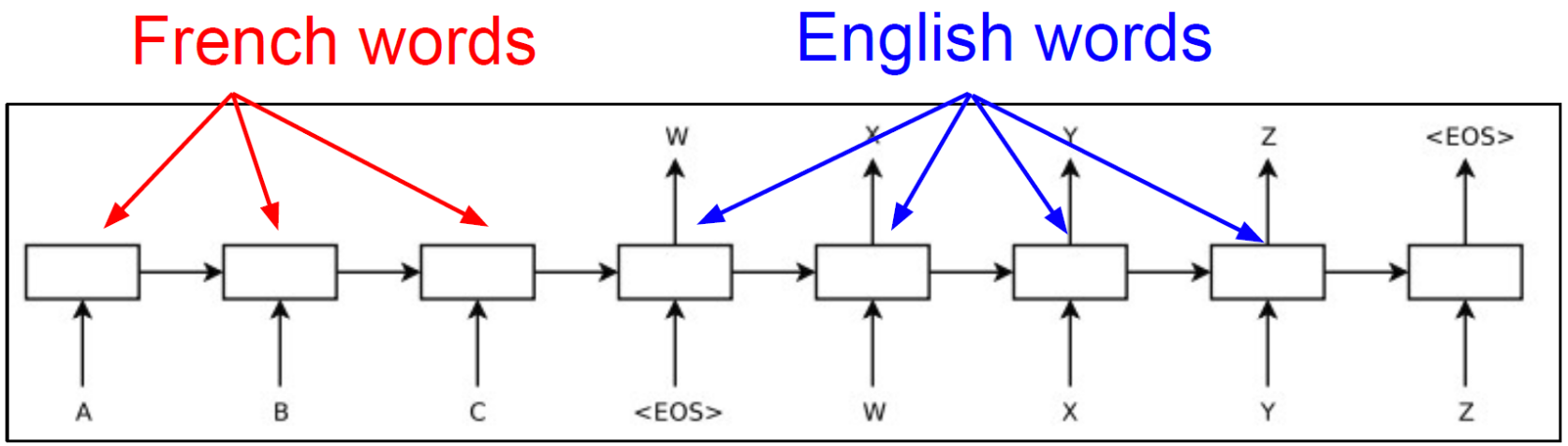
- 编码-解码或序列到序列架构
  - 编码器（encoder）或读取器(reader) 或输入(input) RNN 处理输入序列。编码器输出上下文 C（通常是最终隐藏状态的简单函数）。
  - 解码器（decoder）或写入器(writer) 或输出(output) RNN 则以固定长度的向量为条件产生输出序列 $\boldsymbol{Y} = (\boldsymbol{y}_{(1)}, \ldots, \boldsymbol{y}_{(ny)})$。
  - 共同训练以最大化

$$\log P(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n_y)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n_x)})$$

序列到序列模型


Encoder
...
$\boldsymbol{x}^{(1)}$
$\boldsymbol{x}^{(2)}$
$\boldsymbol{x}^{(\dots)}$
$\boldsymbol{x}^{(n_x)}$
C
Decoder
...
$\boldsymbol{y}^{(1)}$
$\boldsymbol{y}^{(2)}$
$\boldsymbol{y}^{(\dots)}$
$\boldsymbol{y}^{(n_y)}$

序列到序列模型：机器翻译

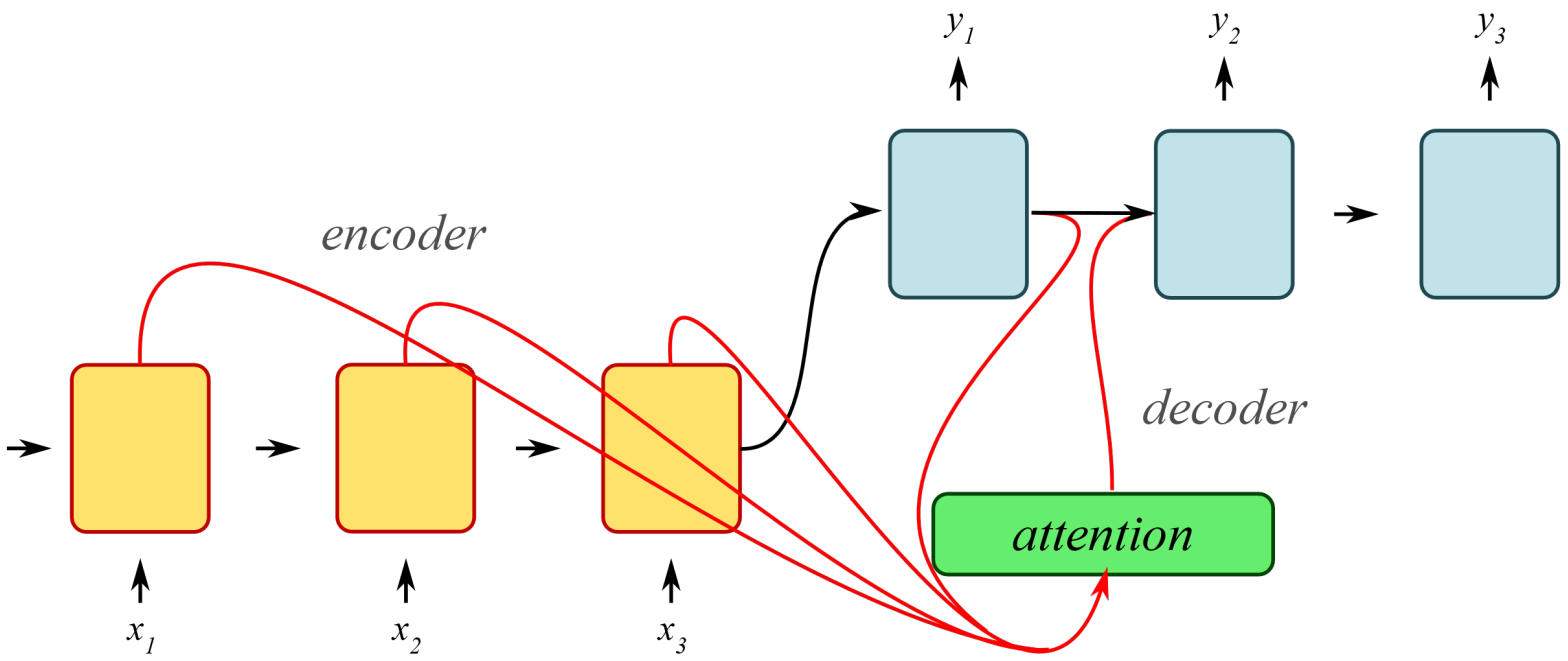


注意力机制：图像

Brushing teeth

Cutting trees

序列到序列模型：注意力机制



循环神经网络

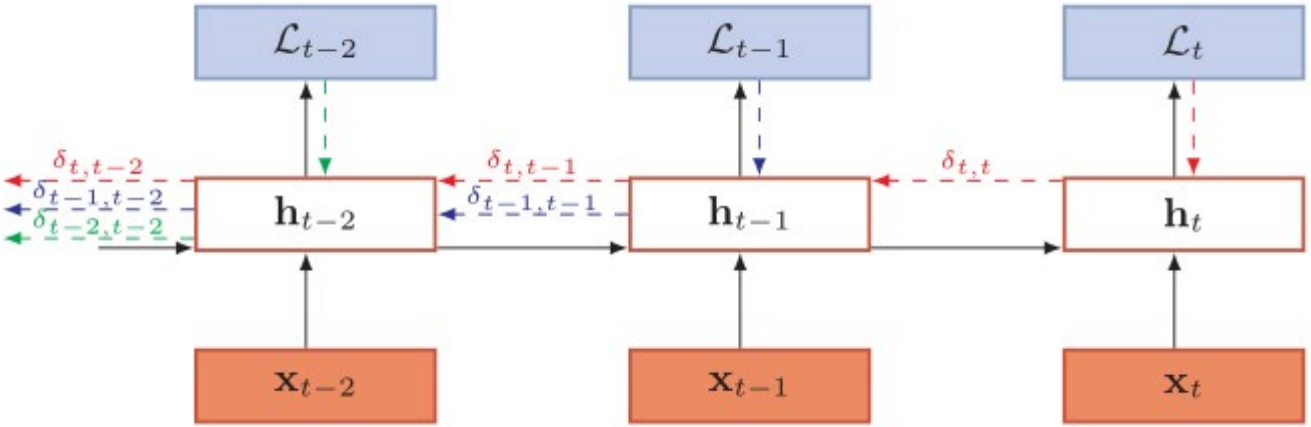
- 循环神经网络（Recurrent NN）
- 双向RNN
- 序列到序列模型
- 长短期记忆（LSTM）、GRU

长短期记忆

- 长期依赖
- 启发式解决方案
- GRU
- LSTM

BPTT

$$\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(U\mathbf{h}_t + W\mathbf{x}_{t+1} + \mathbf{b})$$

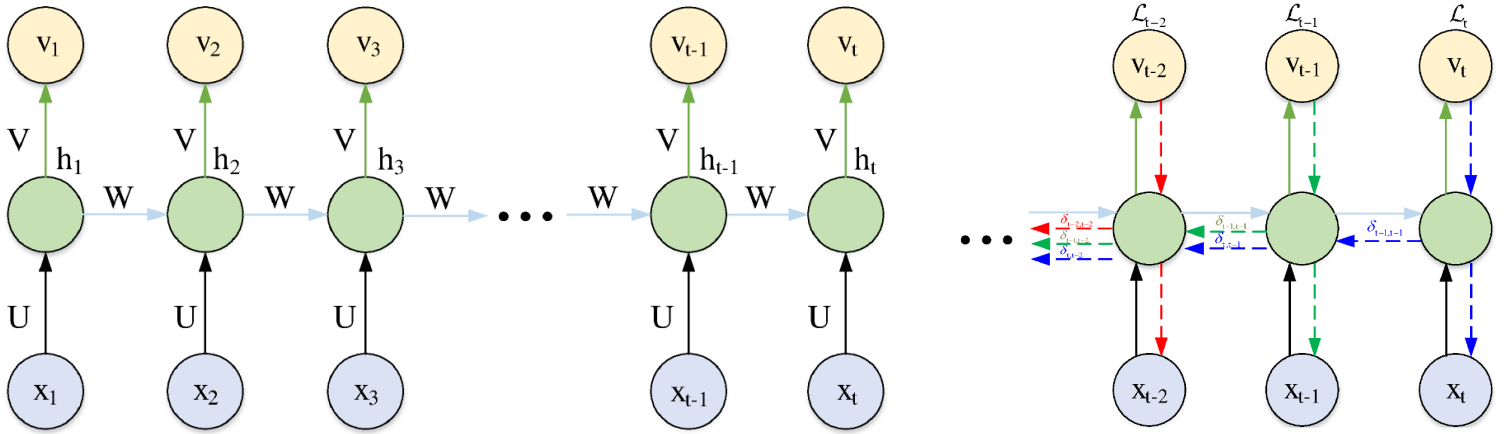


$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T}\sum_{k=1}^{t} \delta_{t,k}\mathbf{h}_{k-1}^{\mathrm{T}}$$

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

$\delta_{t,k}$为第t时刻的损失对第k步隐藏神经元的净输出z_k的导数

长期依赖（Long-Term Denpendency）



- 梯度计算

$$\nabla_{h^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{o^{(t)}} L)$$
$$= \boldsymbol{W}^{\top} (\nabla_{h^{(t+1)}} L) \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t+1)})^2\right) + \boldsymbol{V}^{\top} (\nabla_{o^{(t)}} L),$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)}$$
$$= \sum_t \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) (\nabla_{h^{(t)}} L) \boldsymbol{h}^{(t-1)^{\top}}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \left(\prod_{i=k}^{t-1} \operatorname{Diag}\left(f'\left(\mathbf{z}_i\right)\right) W^T \right) \delta_{t,t} \mathbf{h}_{k-1}^T$$

长期依赖（Long-Term Denpendency）

- 假设 $\gamma \approx \left\| \text{Diag}\left(f'\left(\mathbf{z}_i\right)\right) W^T \right\|$

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \gamma^{t-k} \delta_{t,t} \mathbf{h}_{k-1}^T$$

  - 当 $\gamma > 1$，$t-k \to \infty$ 时，$\gamma^{t-k} \to \infty$, 此时会产生**梯度爆炸**
  - 当 $\gamma < 1$，$t-k \to \infty$ 时，$\gamma^{t-k} \to 0$，从而出现和前馈神经网络类似的梯度消失问题
  - 当 $t-k$ 较大时，时刻 t 损失函数 $\mathcal{L}_t$ 产生的梯度无法对 $t-k$ 时刻之前的参数 W 产生影响
  - 长期依赖：当间隔 k 较大时，网络无法对长时间间隔的数据依赖关系进行建模

长短期记忆

- 长期依赖
- 启发式解决方案
- GRU
- LSTM

缓解梯度爆炸

- 截断梯度 （Gradient clipping）
  - 是当参数的梯度大于一定阈值时，就将其截断为一个较小的数值
  - 方式1：在参数更新之前，逐元素地截断Mini-batch产生的参数梯度
  - 方式2：在参数更新之前，整体约束参数梯度大小（不改变梯度方向）

$$\mathbf{g} = \begin{cases} \frac{\mathbf{g}v}{\|\mathbf{g}\|}, & \text{if } \|\mathbf{g}\| > v \\ \mathbf{g}, & \text{else} \end{cases}$$

  - 实际应用中，两种方式性能表现类似

缓解梯度消失

- 时间维度的跳跃连接
  - 直接构造从t时刻单元到$t+d$时刻单元的连接

- 渗漏单元
  - 令 $W=\mathrm{I}$，$f'(\mathbf{z}_i)=\mathbf{1}$

$$\mathbf{h}_t=\mathbf{h}_{t-1}+\mathbf{g}(\mathbf{x}_t,\phi)$$

  - 丢失了神经元上存在的非线性激活性质，降低了网络的拟合能力

缓解梯度消失：渗漏单元

- 记忆容量（Memory Capacity）问题
  - 随着$\mathbf{h}_t$不断累积存储过去的输出状态，会发生“饱和”现象
- **渗漏单元**（Leaky Unit）

$$\mathbf{h}_t = \mu \mathbf{h}_{t-1} + (1-\mu)\,\mathbf{g}\left(\mathbf{x}_t, \mathbf{h}_{t-1}, \phi\right)$$

  - 当μ接近于1时，神经网络能够记住过去很长一段时间的信息；
  - 当μ接近于0时，关于过去的信息会被快速丢弃
  - 超参数μ：可以预设，也通过数据驱动的方式学习

长短期记忆

- 长期依赖
- 启发式解决方案
- GRU

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

- LSTM

门控循环单元（GRU）

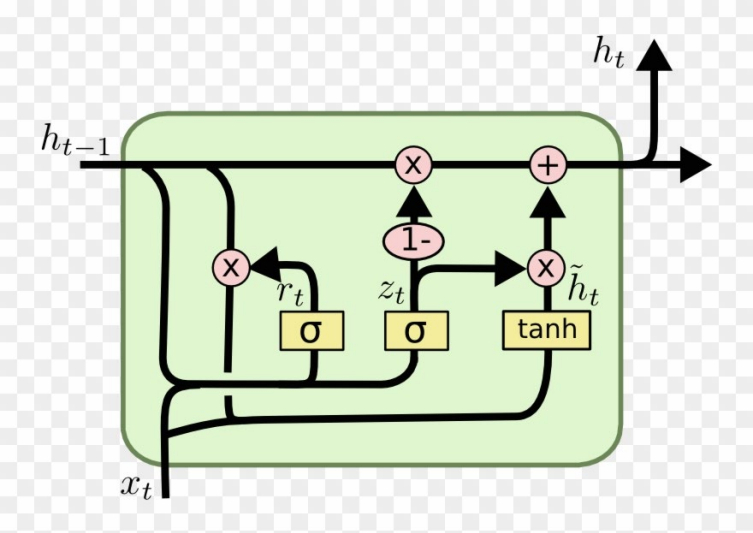
- 引入门控机制：由神经网络学会决定何时清除状态

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t,$$

$$\tilde{\mathbf{h}}_t = \tanh\left(W_h \mathbf{x}_t + U_h \left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right) + b_h\right),$$

$$\mathbf{z}_t = \delta\left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} + b_z\right),$$

$$\mathbf{r}_t = \delta\left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} + b_r\right),$$



- $\mathbf{z}_t$: 更新门（Update Gate）
- $\mathbf{r}_t$: 重置门（Reset Gate）

解释

- 当$\mathbf{z}_t = 0$ 时，当前状态$\mathbf{h}_t$ 和历史状态$\mathbf{h}_{t-1}$ 只存在非线性关系
- 当$\mathbf{z}_t = 0$，$\mathbf{r}_t = 1$，GRU 网络则退化为简单循环神经网络
- 当$\mathbf{z}_t = 0$，$\mathbf{r}_t = 0$， GRU 网络退化为传统的前馈神经网络
- 当$\mathbf{z}_t = 1$ 时，当前时刻的隐藏层输出$\mathbf{h}_{t+1}$ 等于上一时刻的隐藏层输出$\mathbf{h}_t$，而与当前输入$\mathbf{x}_t$ 无关

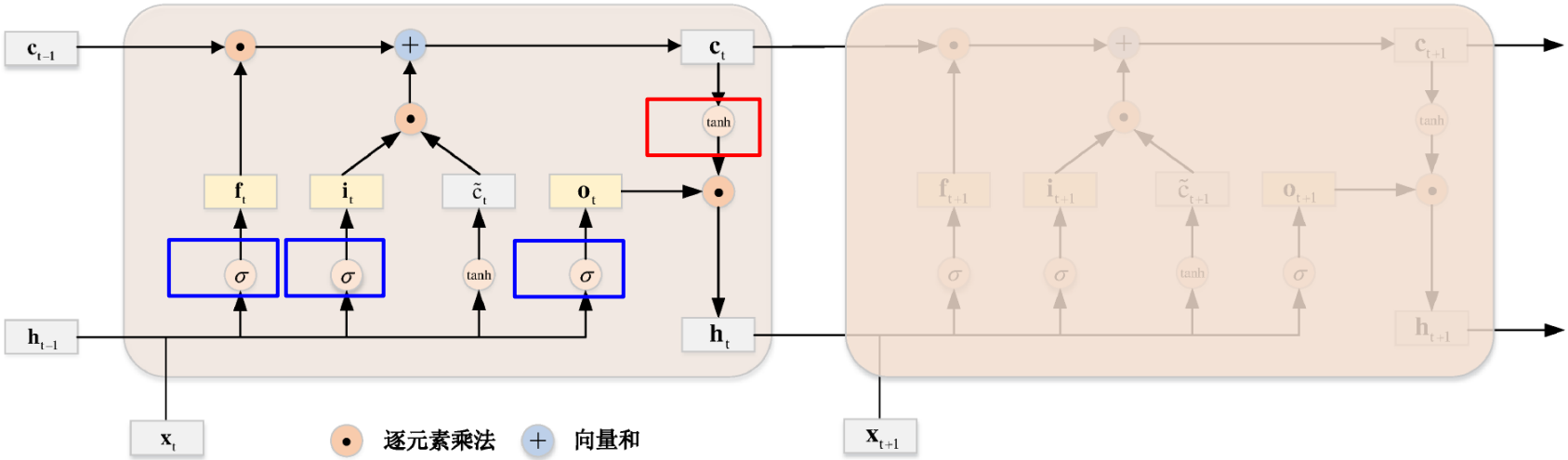
长短期记忆

- 长期依赖
- 启发式解决方案
- GRU
- LSTM

S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997

长短期记忆（long short-term memory）

- 新的记忆单元$\mathbf{c}_t$，用于控制信息的线性传递
  - 候选内部状态 $\tilde{\mathbf{c}}_t$
- 三个门组件
  - 输入门（Input Gate）$\mathbf{i}_t$，
  - 遗忘门（Forget Gate）$\mathbf{f}_t$
  - 输出门（Output Gate）$\mathbf{o}_t$



长短期记忆（LSTM）

- 计算过程
  - 更新门组件

$$\mathbf{i}_t = \delta\left(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + b_i\right),$$

$$\mathbf{f}_t = \delta\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + b_f\right),$$

$$\mathbf{o}_t = \delta\left(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + b_o\right).$$

  - 候选内部状态更新

$$\tilde{\mathbf{c}}_t = \mathbf{tanh}\left(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + b_c\right)$$

  - 记忆单元和隐藏单元更新

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{tanh}\left(\mathbf{c}_t\right),$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t,$$

$$\tilde{\mathbf{h}}_t = \tanh\left(W_h\mathbf{x}_t + U_h\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right) + b_h\right),$$

$$\mathbf{z}_t = \delta\left(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1} + b_z\right),$$

$$\mathbf{r}_t = \delta\left(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1} + b_r\right),$$

长短期记忆（LSTM）

- 通过前一时刻的输出状态$\mathbf{h}_{t-1}$和当前时刻输入$\mathbf{x}_t$计算当前时刻三个门的输出$\mathbf{f}_t$，$\mathbf{i}_t$和$\mathbf{o}_t$；
- 计算当前时刻候选内部状态$\tilde{\mathbf{c}}_t$，同时结合上一时刻的记忆单元输出$\mathbf{c}_{t-1}$和$\mathbf{f}_t$，计算当前时刻的记忆单元输出$\mathbf{c}_t$。
- 结合输出门$\mathbf{o}_t$，计算当前时刻隐藏单元的最终输出$\mathbf{h}_t$。

门机制解释

- 当$\mathbf{f}_t = 0$，$\mathbf{i}_t = 1$ 时，记忆单元将历史信息清空，并将候选内部状态 $\tilde{\mathbf{c}}_t$ 写入；

- 当$\mathbf{f}_t = 1$，$\mathbf{i}_t = 0$ 时，记忆单元将复制上一时刻的内容，不写入新的信息。

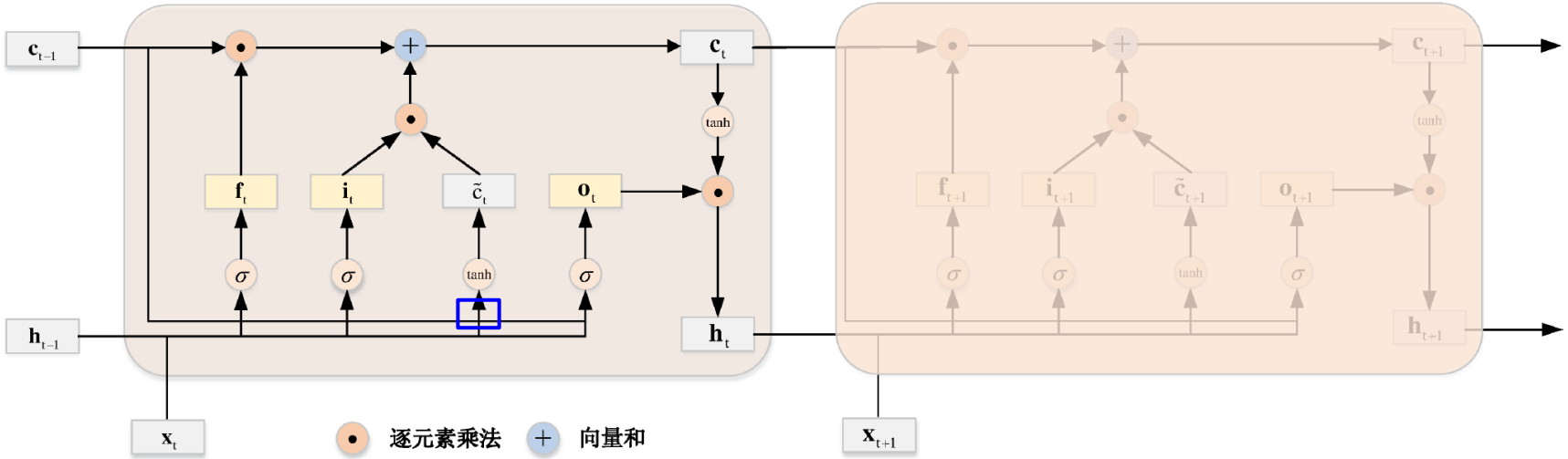
- 外部的RNN 循环+内部的LSTM 细胞循环（自环）

变体1：带有peephole 连接的LSTM

$$\mathbf{i}_t = \delta\left(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c}_{t-1} + b_i\right),$$

$$\mathbf{f}_t = \delta\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_f\mathbf{c}_{t-1} + b_f\right),$$

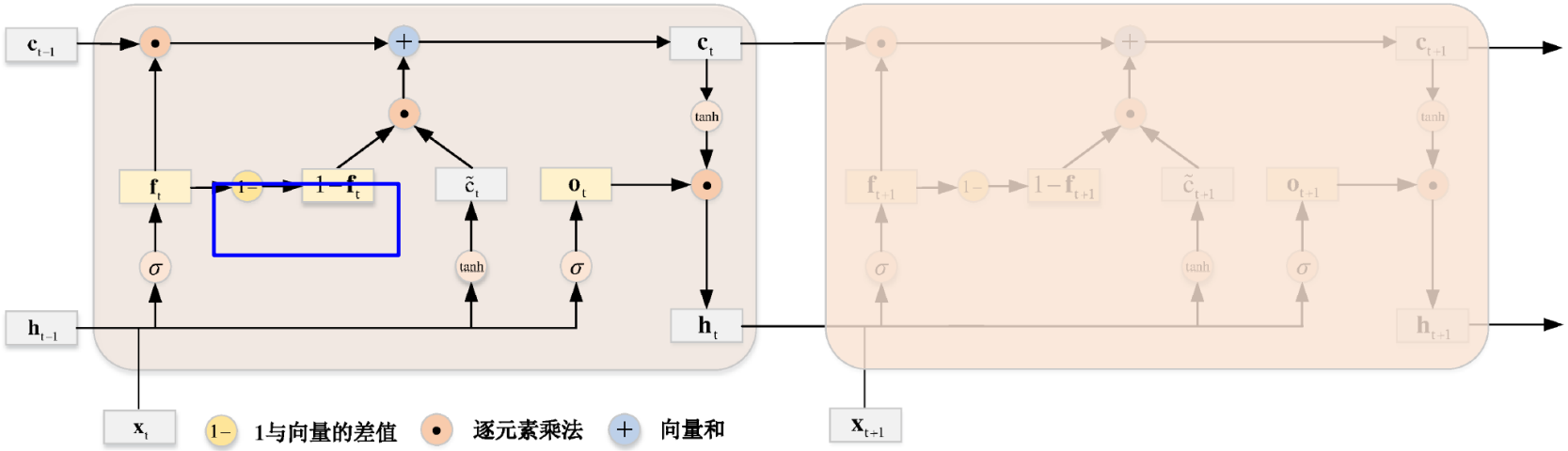
$$\mathbf{o}_t = \delta\left(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c}_{t-1} + b_o\right).$$



变体2：耦合输入门和遗忘门的LSTM

$$\mathbf{i}_t = 1 - \mathbf{f}_t$$

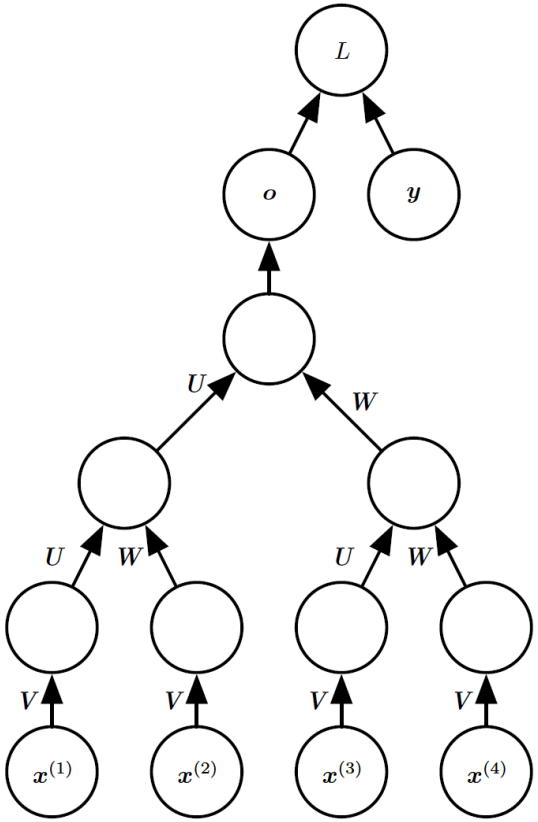
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \tilde{\mathbf{c}}_t$$



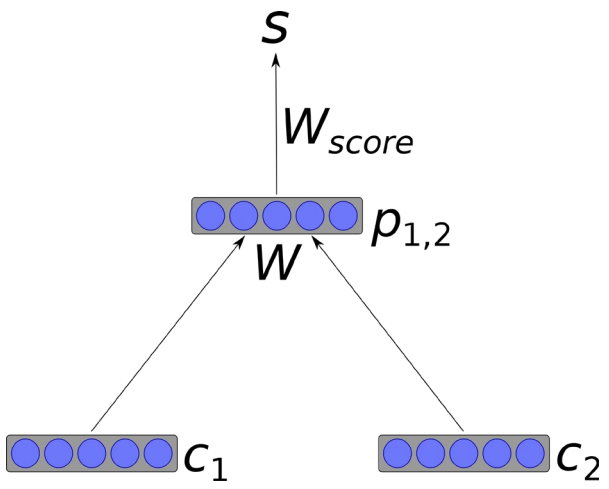
序列建模

- 循环神经网络
- 递归神经网络
- 记忆网络
- 图神经网络

递归神经网络



递归网络将循环网络的链状计算图推广到树状计算图



递归网络基本单元

$$p_{1,2} = \tanh(W[c_1; c_2])$$

给定树结构，网络深度可从O(T)降至O(logT)

递归神经网络

- 如何以最佳的方式构造树
  - 使用不依赖于数据的树结构
  - 借鉴外部方法选择适当的树结构（语法树）
  - 自行发现和推断适合于任意给定输入的树结构（层次聚类）

序列建模

- 循环神经网络
- 递归神经网络
- 记忆网络
- 图神经网络

知识的种类与表达

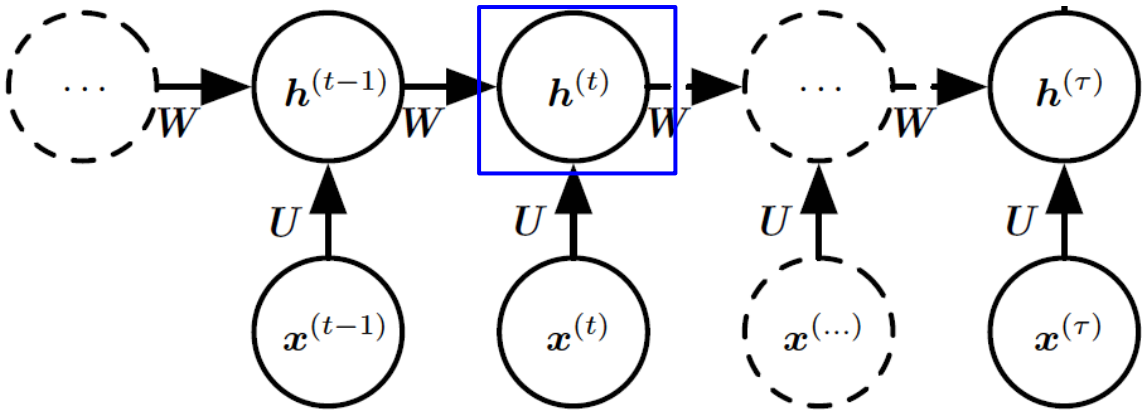
- 隐性知识：隐含的、潜意识的并且难以用语言表达
  - 如：怎么行走或狗与猫的样子有什么不同
- 明确的、可陈述的以及可以相对简单地使用词语表达
  - 常识性的知识：猫是一种动物
  - 具体的事实：与销售团队会议在141室于下午3:00 开始

词语、概念和概念间的关系

记忆网络

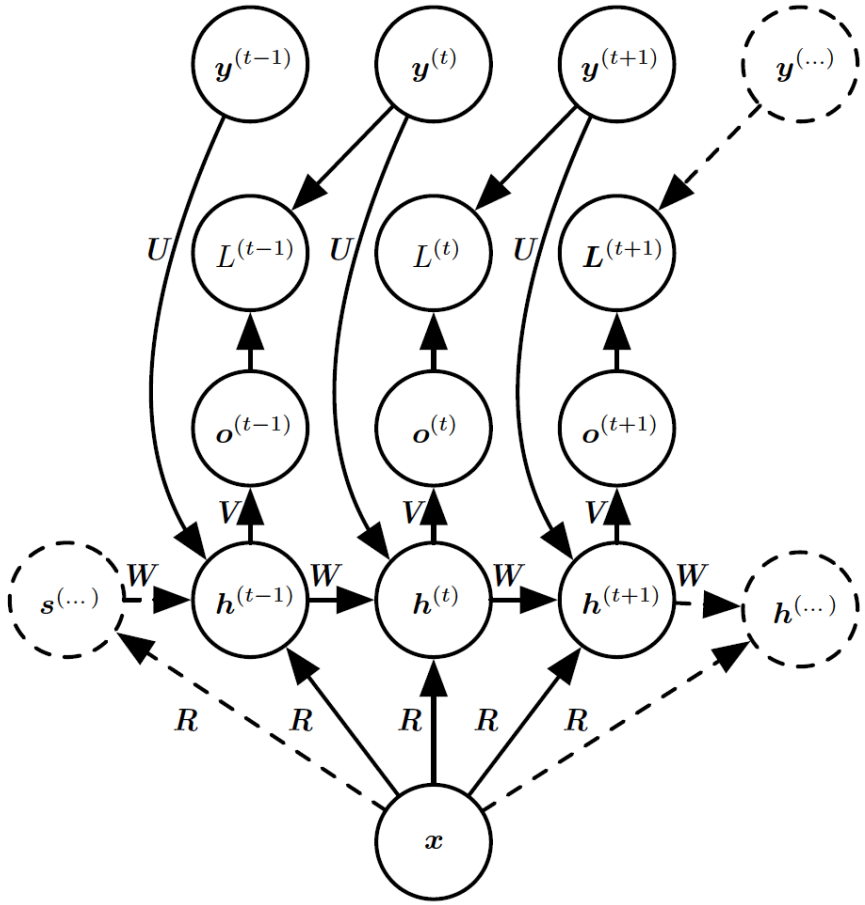
- 神经网络擅长存储隐性知识，但很难记住事实
  - 缺乏工作存储系统：外显记忆组件

- 如果在神经网络中引入外部知识？



Revisit: 基于上下文的RNN 序列建模

- 只使用单个向量$\boldsymbol{x}$ 作为输入



记忆网络

- 记忆网络：引入记忆单元
  - 需要监督信号指示他们如何使用自己的记忆单元

    Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916* .

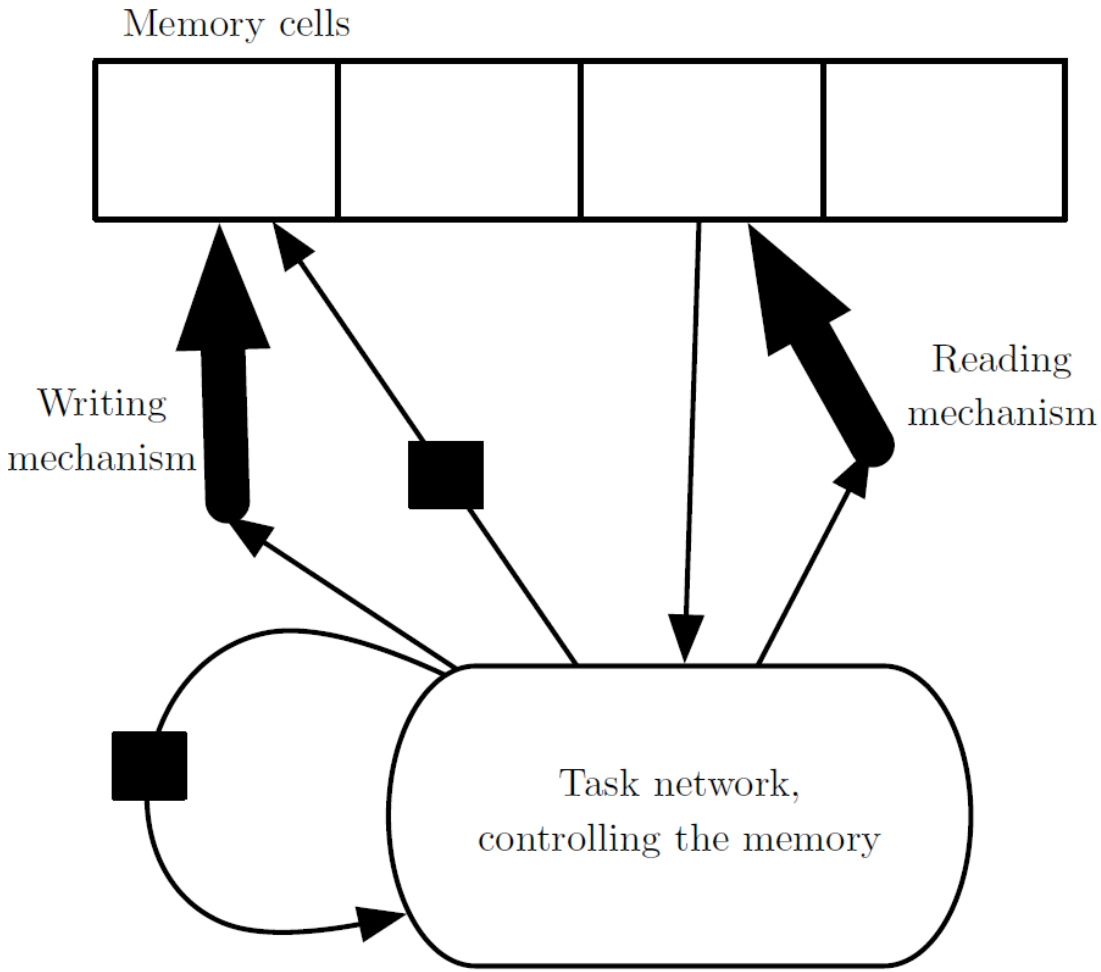
- 神经网络图灵机：不需要明确的监督指示而能学习从记忆单元读写任意内容

  Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. arXiv:1410.5401.
- 基于内容的软注意机制：端到端训练

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In ICLR'2015, arXiv:1409.0473 .

具有外显记忆的神经网络

- 记忆网络
- 神经网络图灵机 (NTM)



具有外显记忆的神经网络

- 记忆单元的写入和读取：
  - 避免整数寻址
  - 以概率形式同时从多个记忆单元写入或读取
    - 读取时，采取许多单元的加权平均值
    - 写入时，同时修改多个单元

- 使用向量值的记忆单元
  - 基于内容的寻址(content-based addressing)

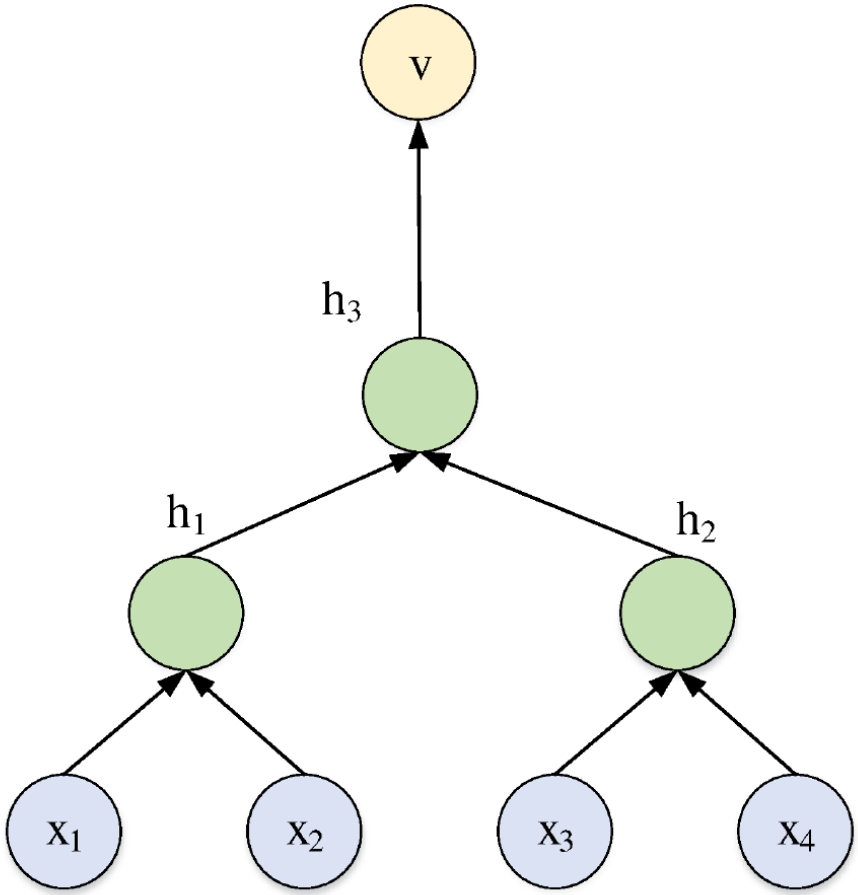
    检索一首副歌歌词中带有’We all live in a yellow submarine’ 的歌

序列建模

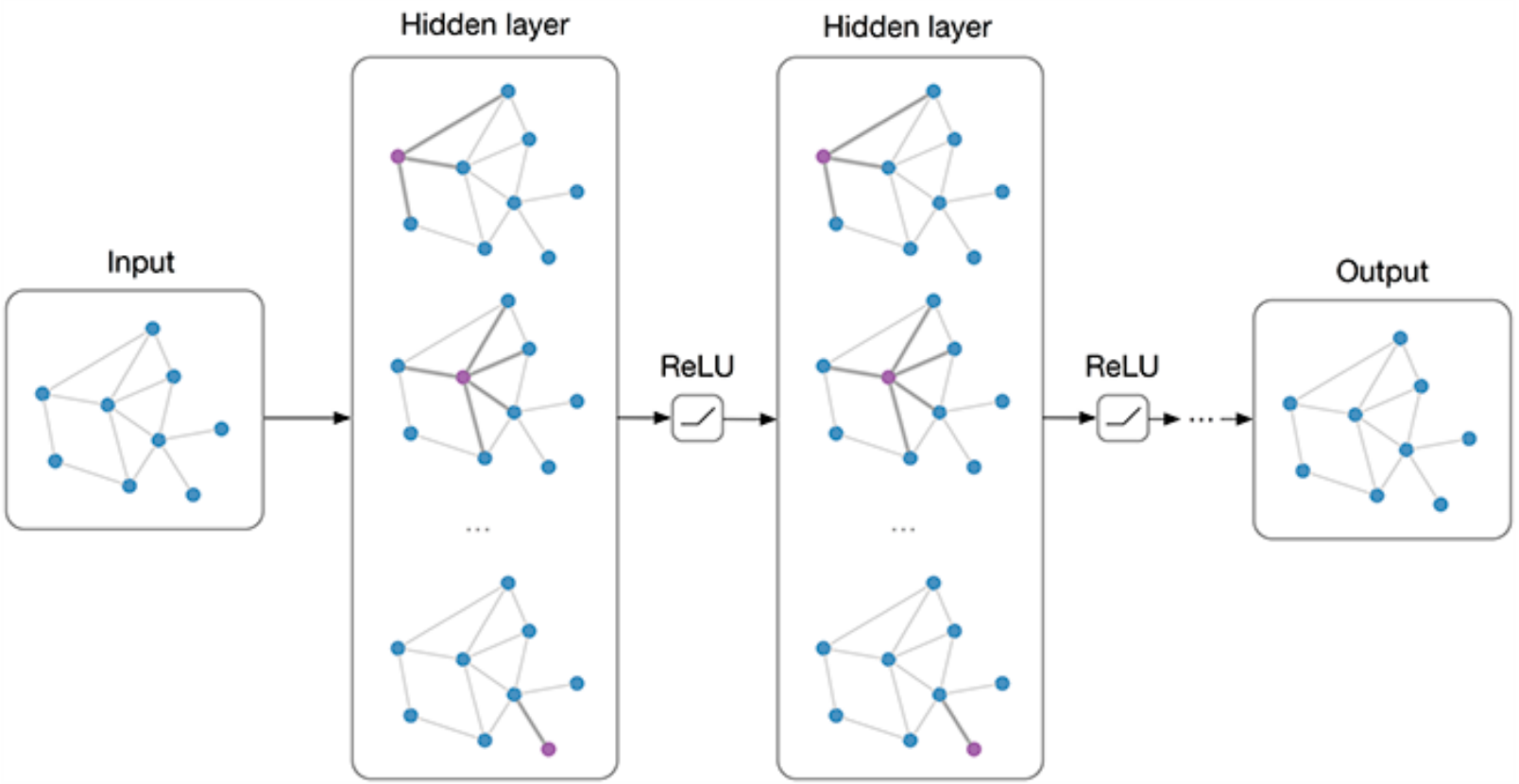
- 循环神经网络
- 递归神经网络
- 记忆网络
- 图神经网络

递归神经网络

- 当输入数据为更为复杂的结构化数据
  - 节点分类
  - 关系预测



图神经网络



T. N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)

总结

- 循环神经网络
- 梯度膨胀/消失
- LSTM/GRU