递归神经网络

Recurrent Neural Network

CONTENT

递归神经网络

长短记忆网络

词嵌入表示

递归神经网络的应用

1 递归神经网络

Recurrent Neural Network: Sequence Problem

递归神经网络: 序列问题

- 翻译
- 语音识别
- 时序预测
- 语言生成

Why Recurrent?

为什么递归?

- 传统模型通常假设样本是iid
- 时序依赖数据需要一个机制来记住之前的信息
  - “我是中国人，我说?(中文?英文?)
- HMM模型训练复杂度过高, 参数数量过高

Recurrent Neural Network

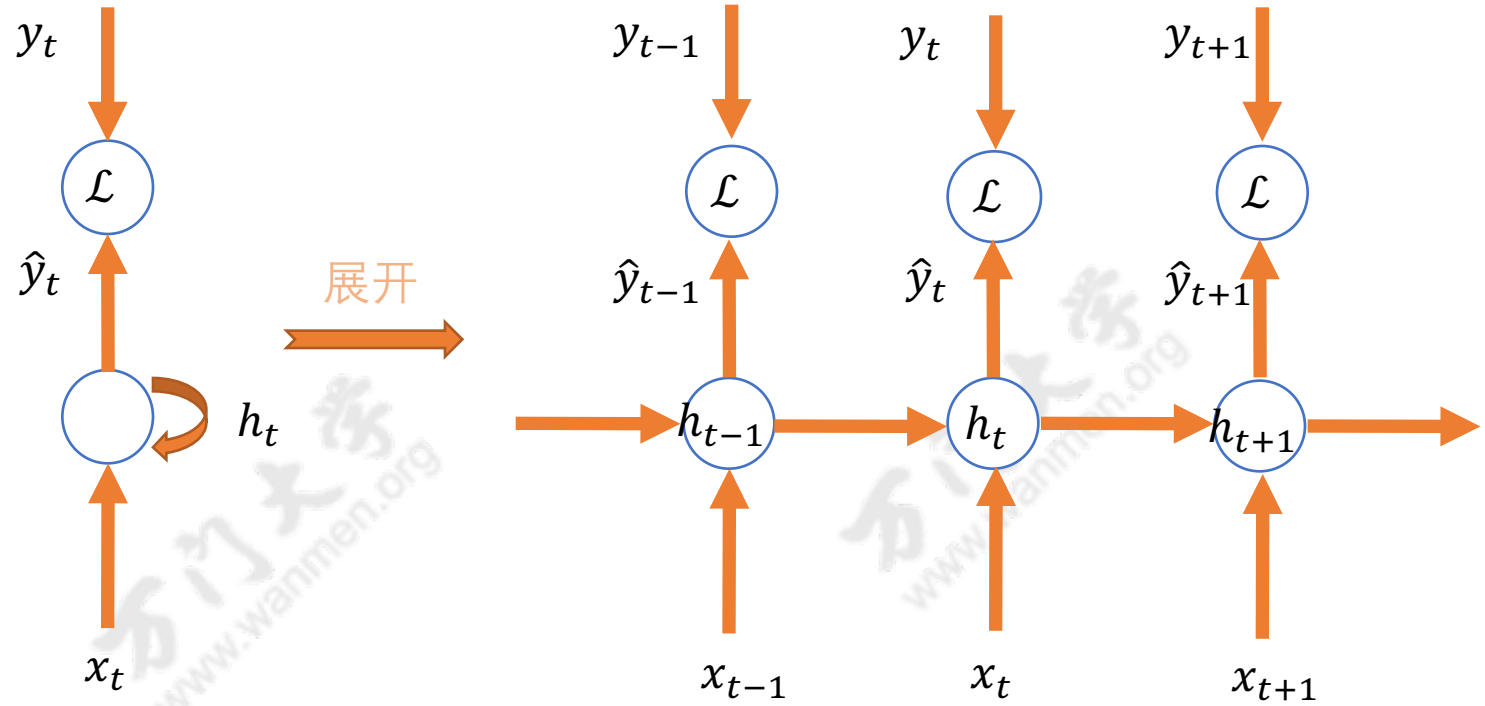
递归神经网络

$$h_t = f(Ux_t + Wh_{t-1} + b)$$

$$o_t = Vh_t + c$$

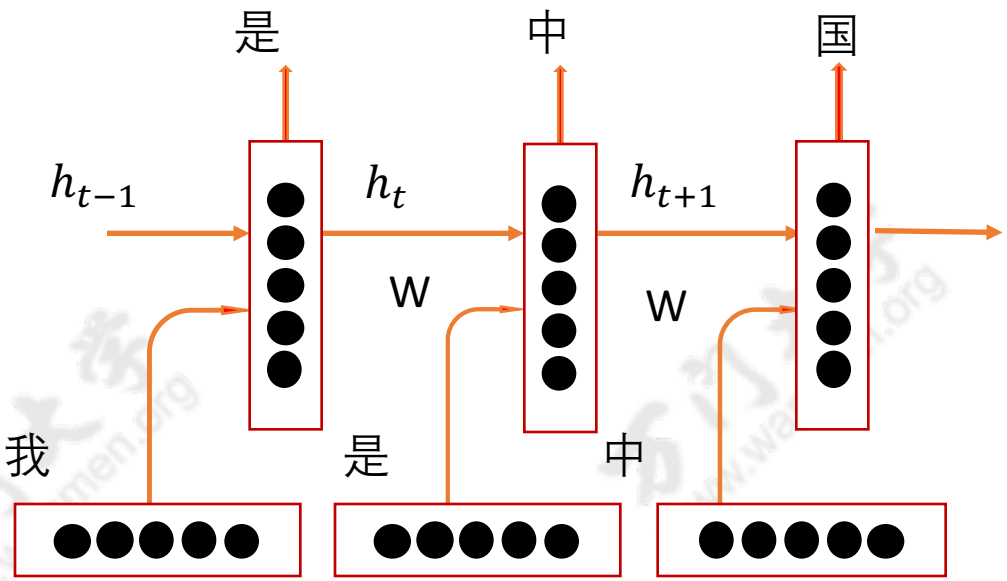
Recurrent Neural Network

递归神经网络



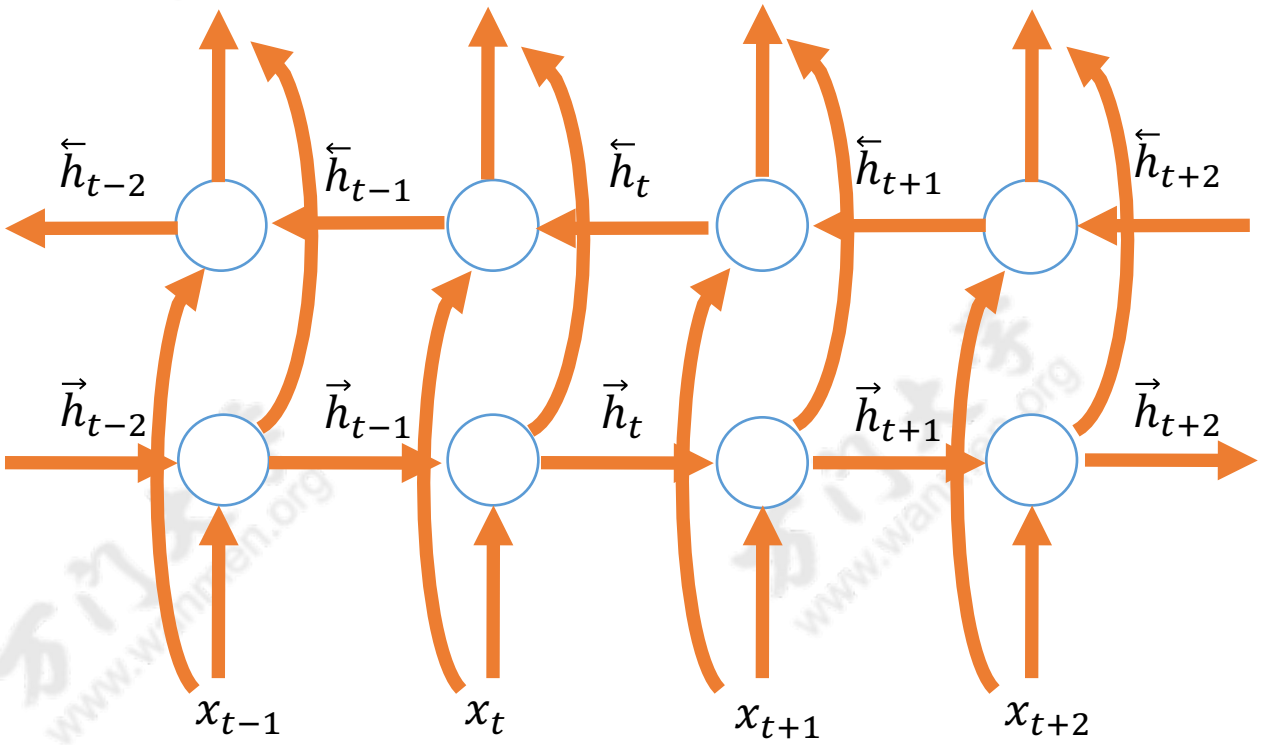
Recurrent Neural Network

递归神经网络



Bidirectional Recurrent Neural Network

双向递归神经网络

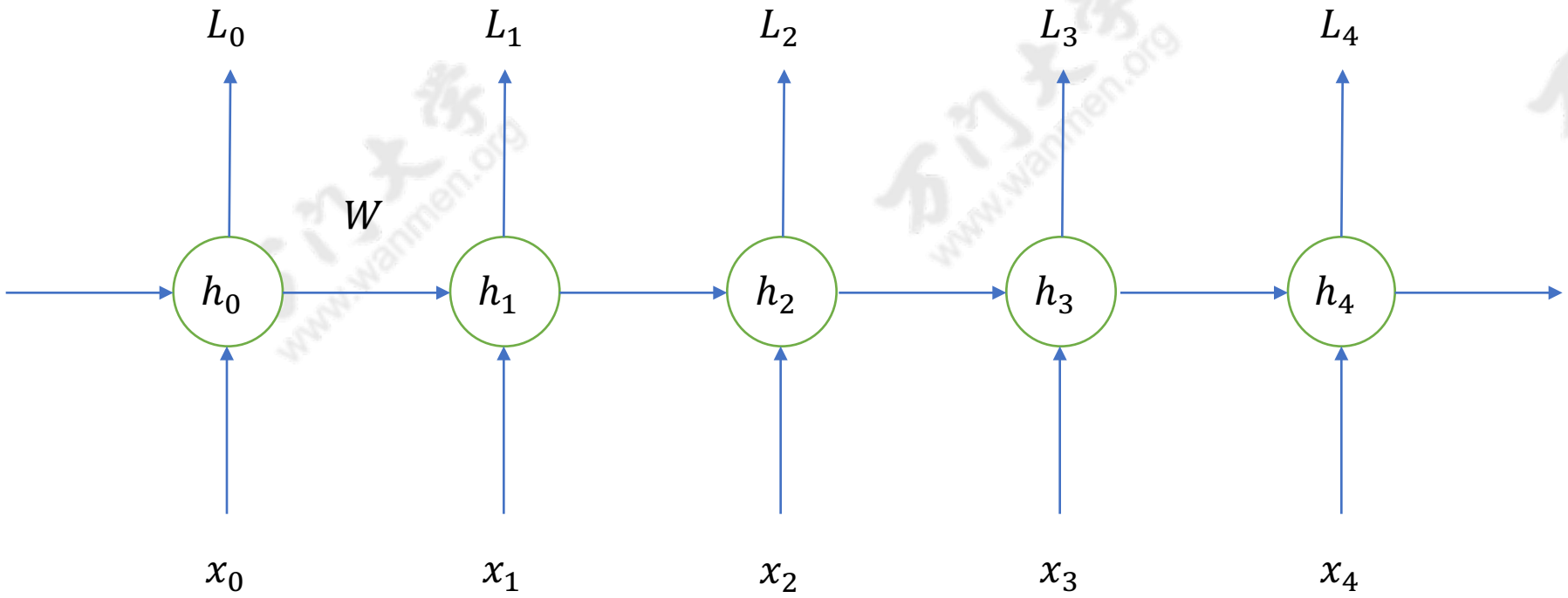


Backpropagation Through Time

沿时间反向传播

$$L_t(y_t, \hat{y}_t) = -y_t log\hat{y}_t$$

$$L(y, \hat{y}) = \sum_t L_t$$



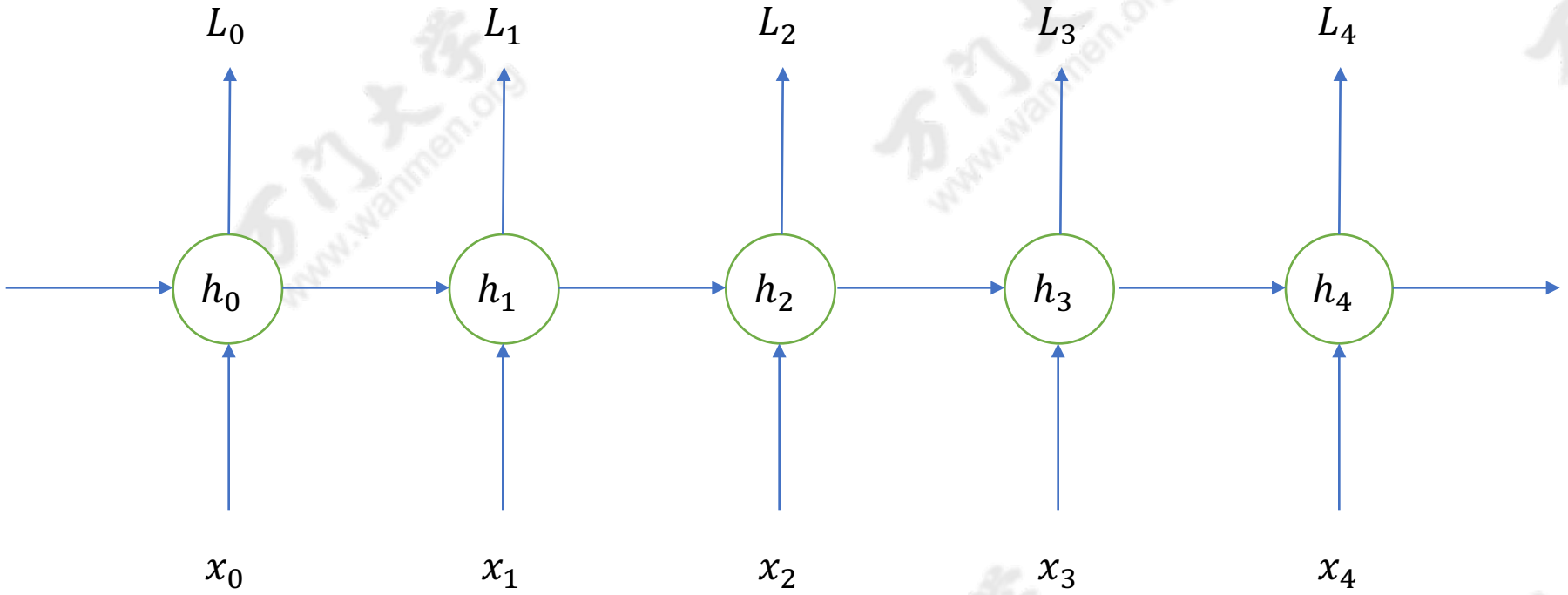
Backpropagation Through Time

沿时间反向传播

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W}$$

$$h_t = f(Ux_t + Wh_{t-1} + b)$$
$$o_t = Vh_t + c$$

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W} = \sum_{t=0}^{3} \frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial h_t}\frac{\partial h_t}{\partial W}$$



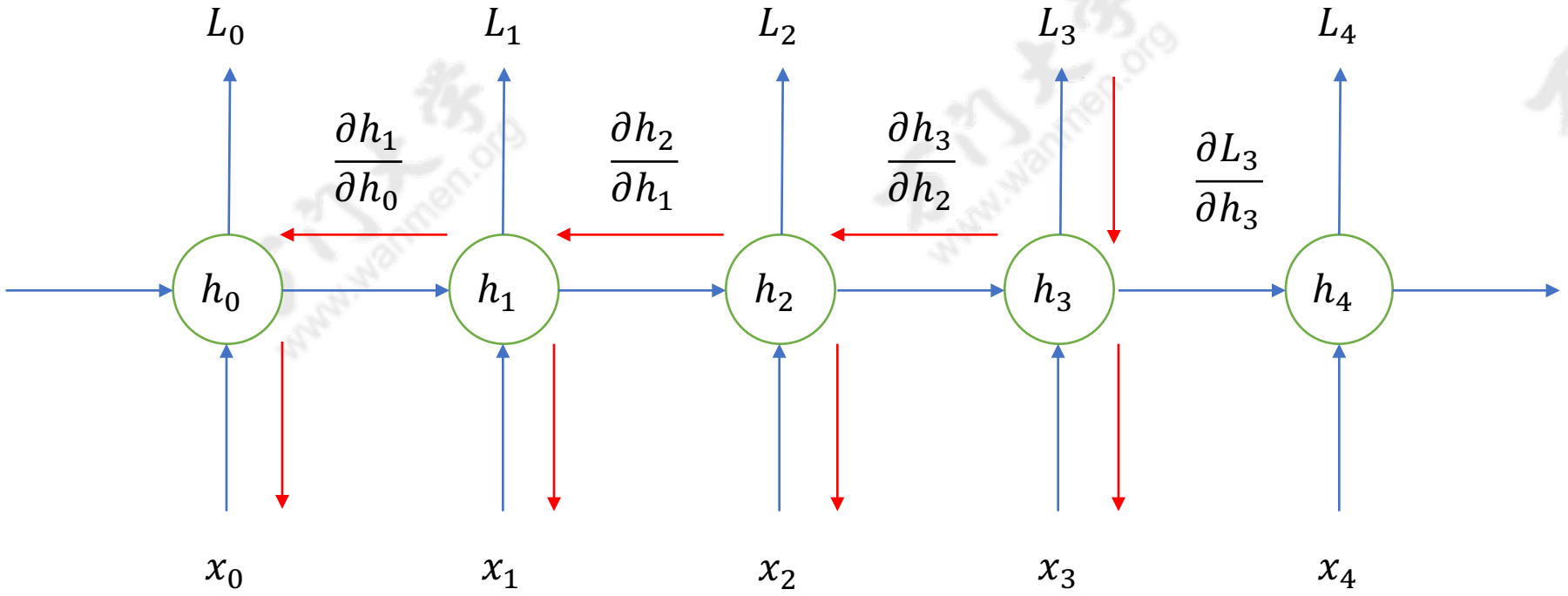
Backpropagation Through Time

沿时间反向传播

$$h_t = f(Ux_t + Wh_{t-1} + b)$$
$$o_t = Vh_t + c$$

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W} = \sum_{t=0}^{3} \frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial h_t}\frac{\partial h_t}{\partial W}$$



Gradient Vanish

梯度消失

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W} = \sum_{t=0}^{3}\frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial h_t}\frac{\partial h_t}{\partial W}$$

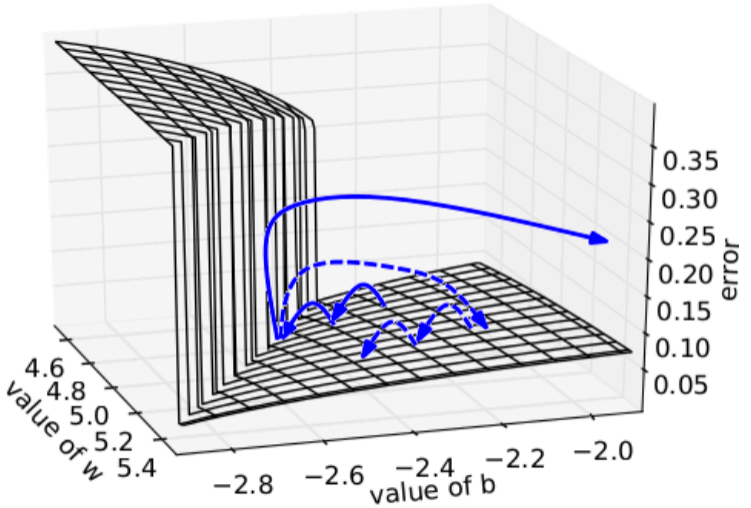
对$\frac{\partial h_3}{\partial h_k}$, 再次使用链式法则

$$\frac{\partial h_3}{\partial h_1} = \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$

$$\frac{\partial L_3}{\partial W} = \sum_{t=0}^{3}\frac{\partial L_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\left(\prod_{j=t+1}^{3}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_t}{\partial W}$$

Gradient Vanish

梯度消失



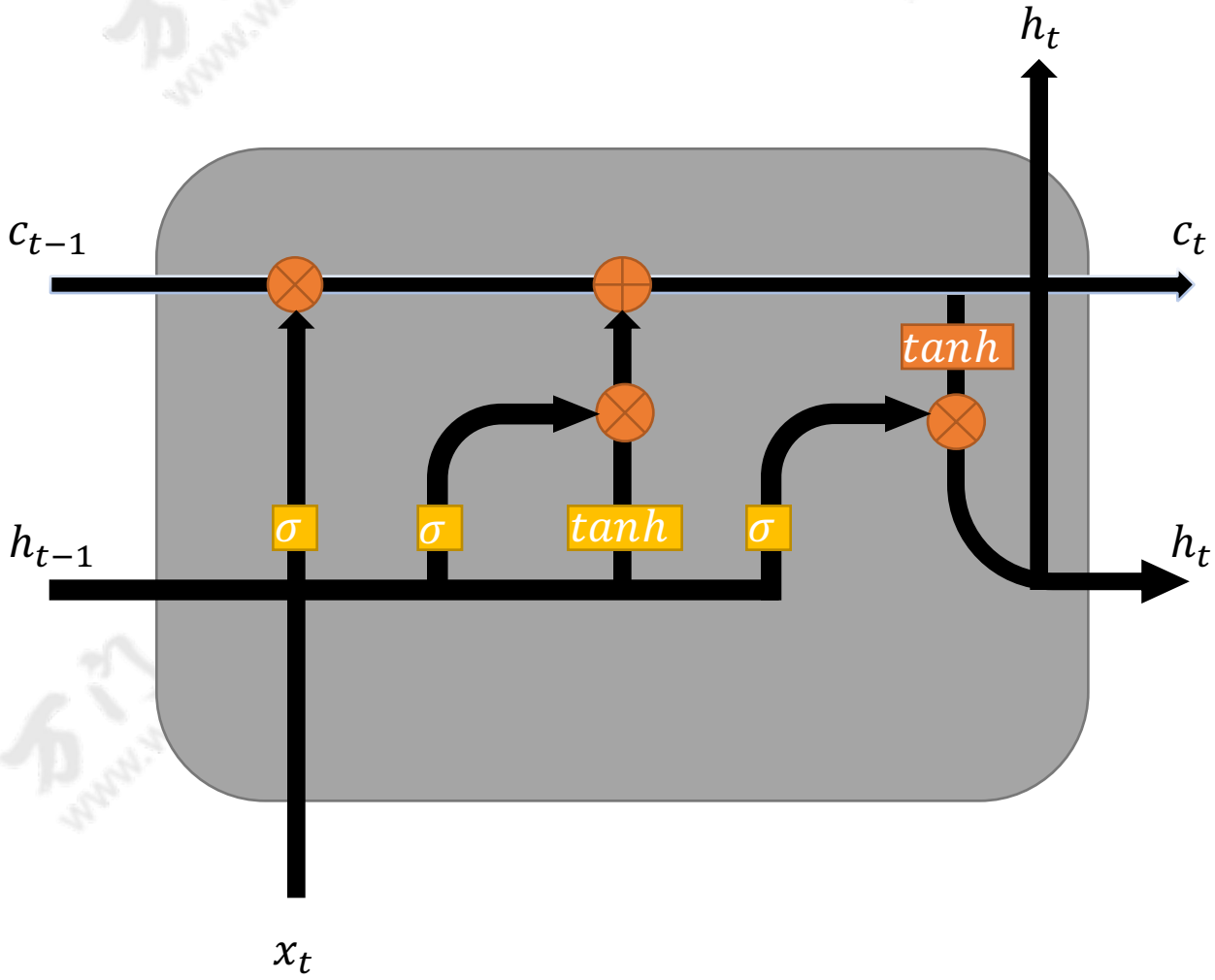
On the difficulty of training recurrent neural networks

Learning Long-Term Dependencies with Gradient Descent is Difficult

2 长短记忆网络

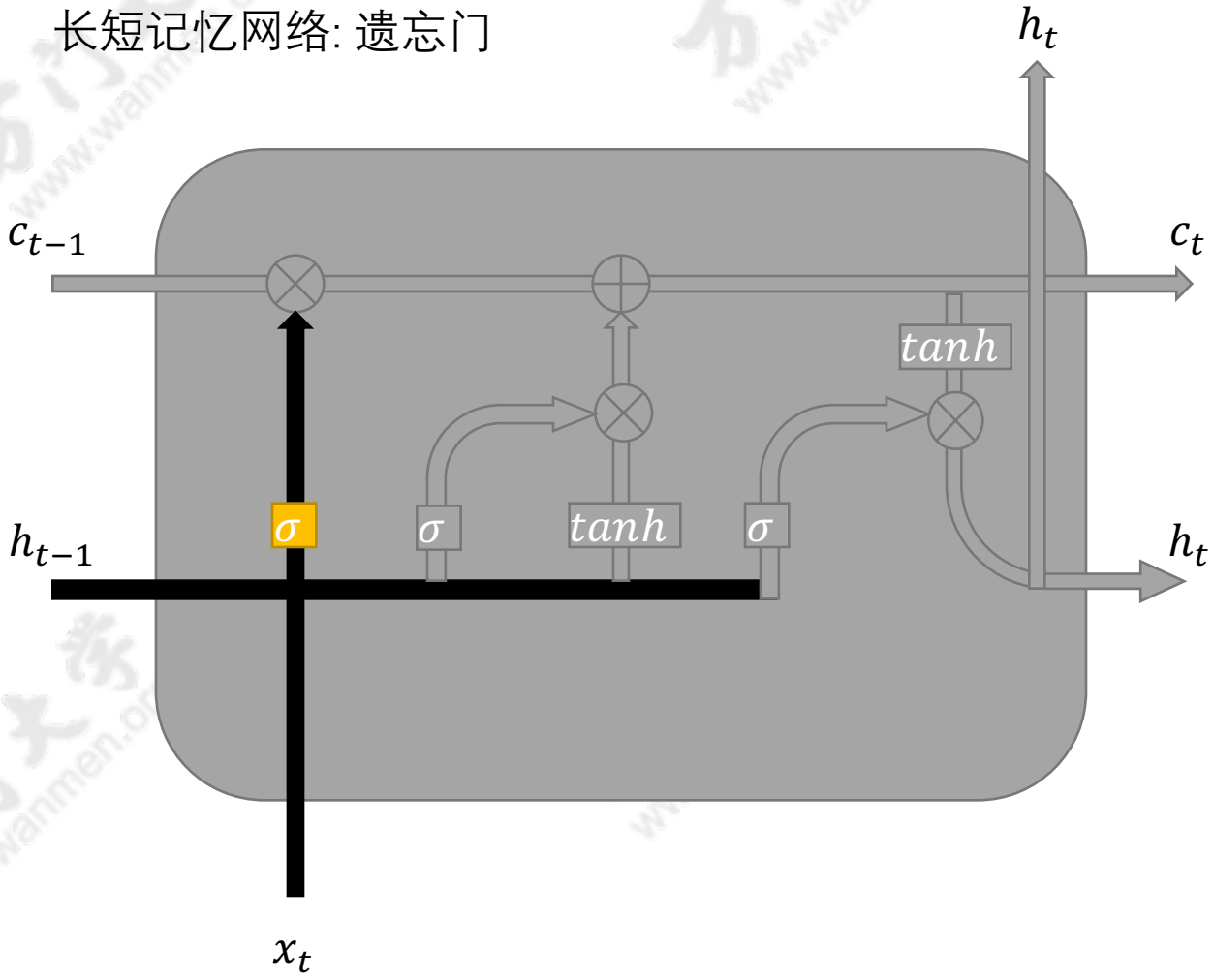
Long-short Term Memory Network

长短记忆网络



Long-short Term Memory Network: Forget Gate

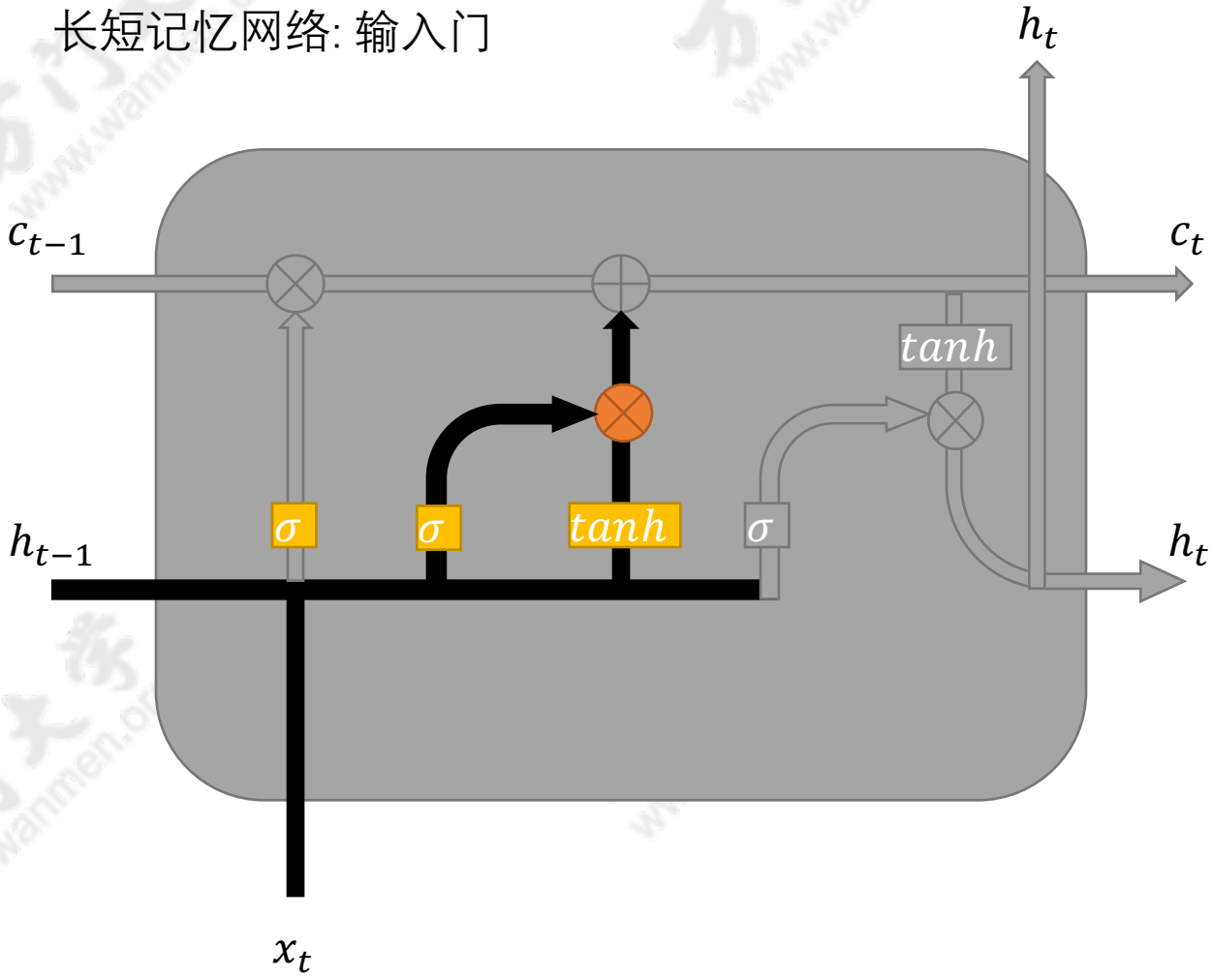
长短记忆网络: 遗忘门



$$f_t = \sigma(U^f x_t + W^f h_{t-1})$$

Long-short Term Memory Network: Input Gate

长短记忆网络: 输入门

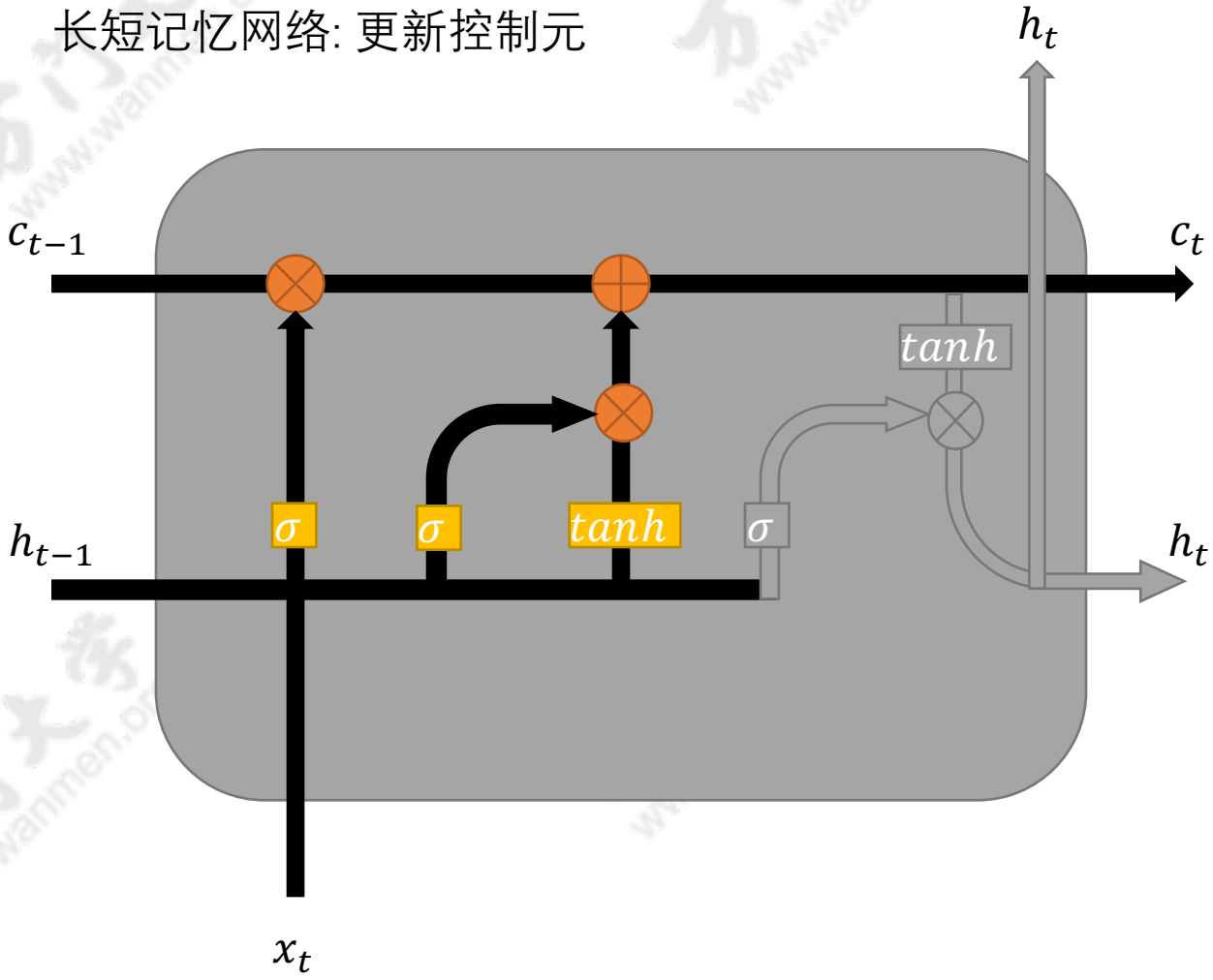


$$i_t = \sigma\left(U^i x_t + W^i h_{t-1}\right)$$

$$g_t = \tanh(U^g x_t + W^g h_{t-1})$$

Long-short Term Memory Network: Update Cell State

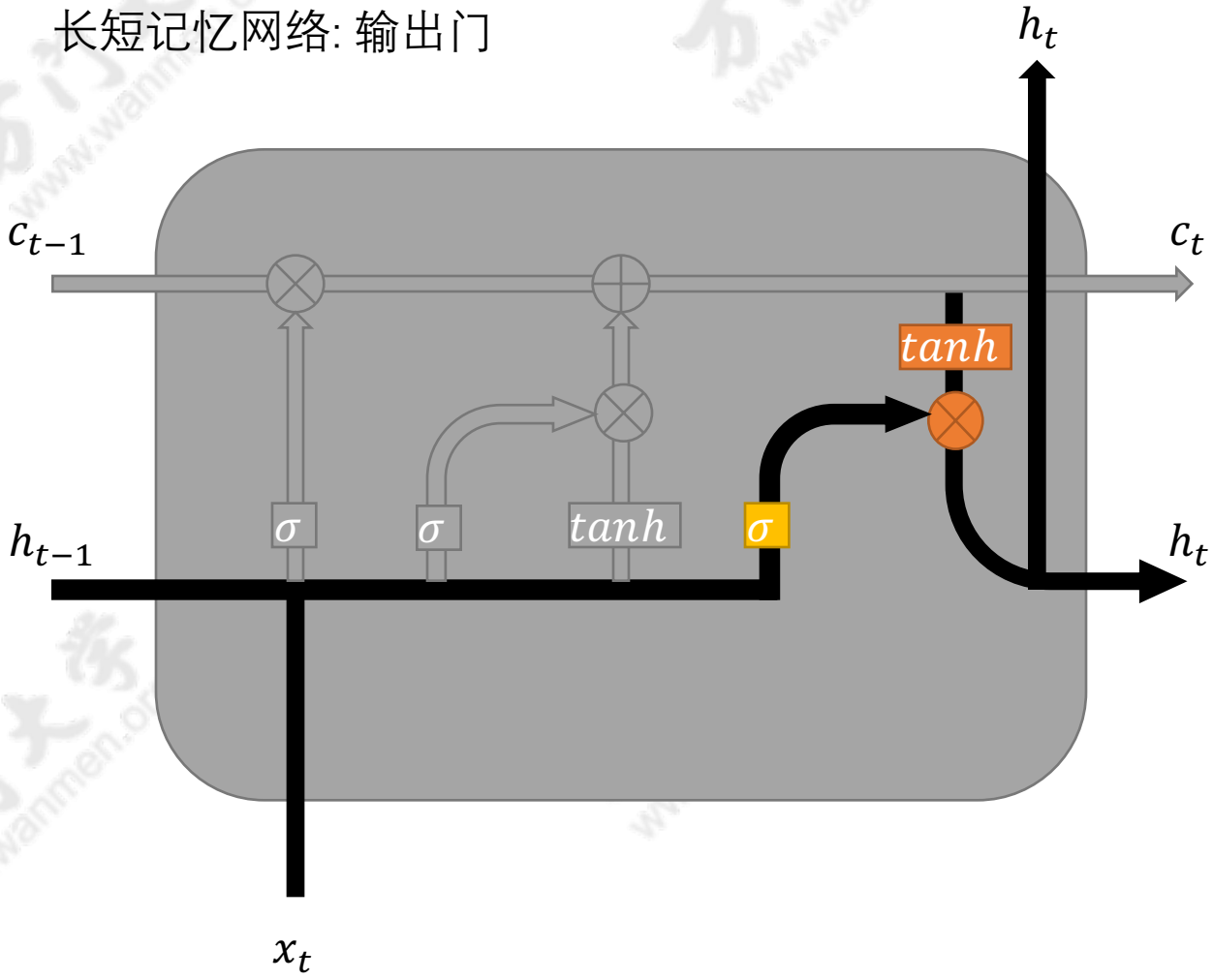
长短记忆网络: 更新控制元



$$c_t = c_{t-1} \circ f_t + g_t \circ i_t$$

Long-short Term Memory Network: Output Gate

长短记忆网络: 输出门

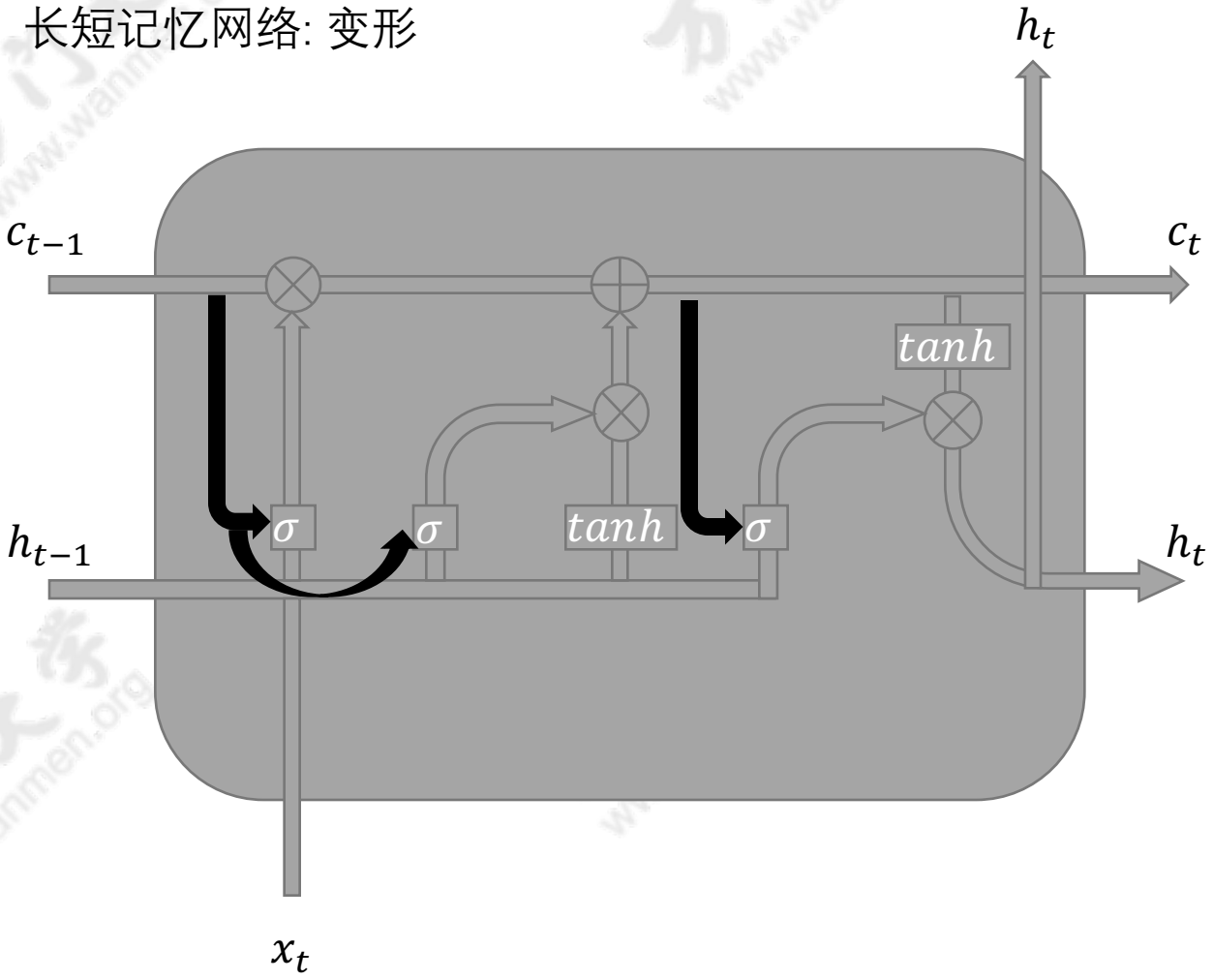


$$o_t = \sigma(U^o x_{t-1} + W^o h_{t-1})$$

$$h_t = o_t * \tanh(c_t)$$

Long-short Term Memory Network: Variants

长短记忆网络: 变形



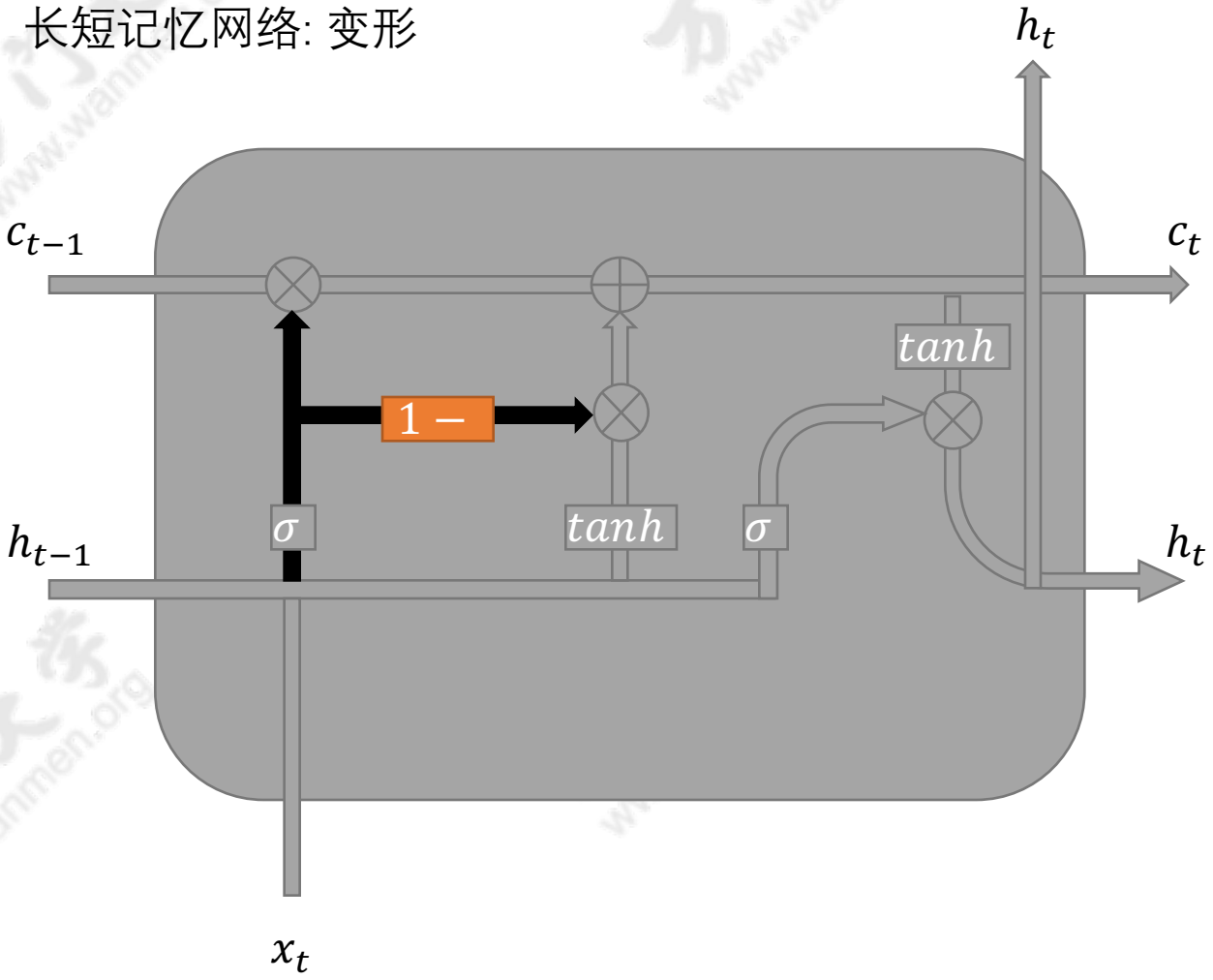
$$f_t = \sigma\left(U^f x_t + W^f h_{t-1} + V^f c_{t-1}\right)$$

$$i_t = \sigma\left(U^i x_t + W^i h_{t-1} + V^i c_{t-1}\right)$$

$$o_t = \sigma(U^o x_t + W^o h_{t-1} + V^o c_t)$$

Long-short Term Memory Network: Variants

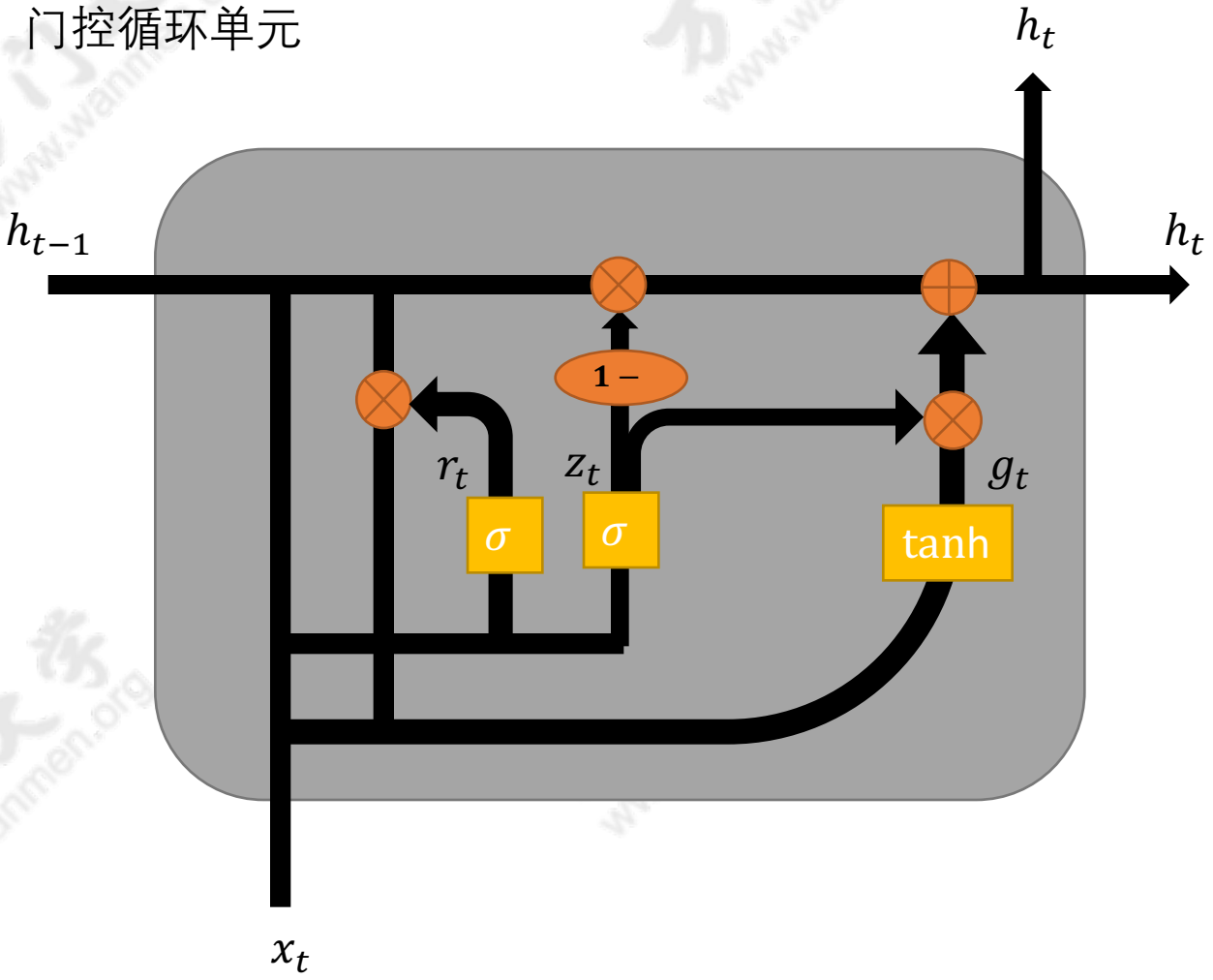
长短记忆网络: 变形



$$c_{t-1} = f_t * c_{t-1} + (1 - f_t) * g_t$$

Gated Recurrent Unit

门控循环单元



$$z_{t-1} = \sigma(W^z h_{t-1} + U^z x_t)$$

$$r_t = \sigma(W^r h_{t-1} + U^r x_t)$$

$$g_t = \tanh(W^g \cdot (r_t * h_{t-1}) + U^g x_t)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * g_t$$

词嵌入表示

N-gram Model Recap

N-元模型回顾

- 句子: $x_1, \dots x_m$
- 贝叶斯展开:

$$P(x_1, x, \dots, x_m) = P(x_1)P(x_2|x_1) \dots P(x_m|x_1, x_2, \dots, x_{m-1})$$

- N-元模型:

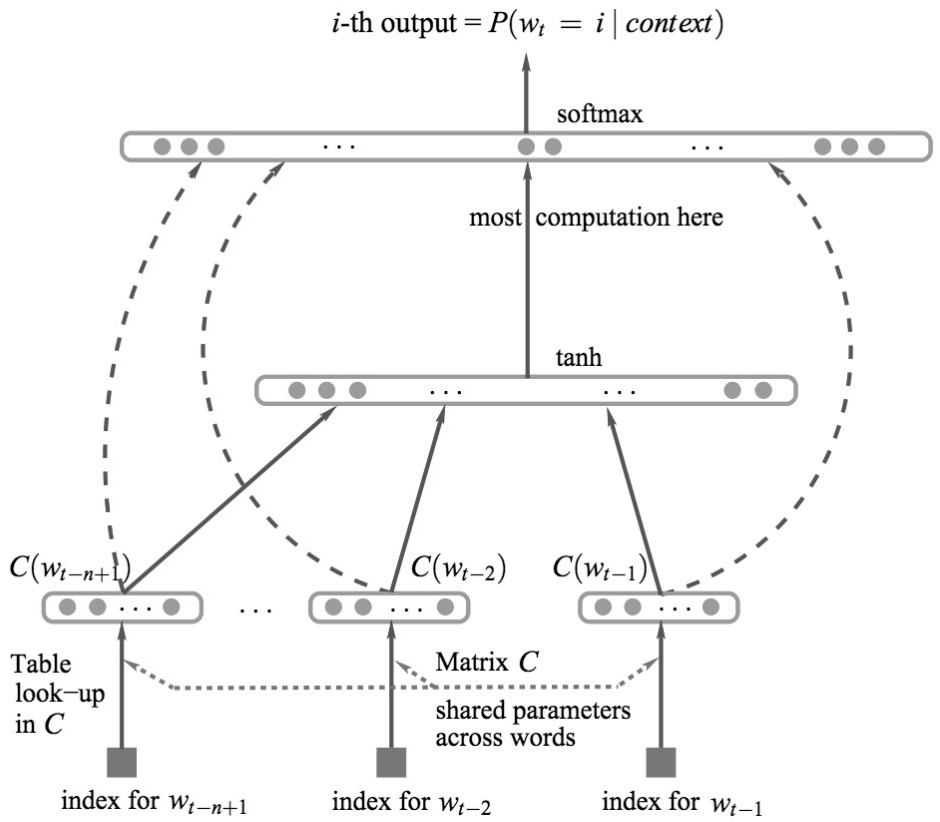
$$P(x_1, x_2, \dots, x_m) = P(x_1)P(x_2|x_1) \dots P(x_m|x_1, x_2, \dots, x_{m-1})$$

- 极大似然估计:

$$P\left(x_i \middle| x_{i-(n-1)}, \dots, x_{i-1}\right) = \frac{count(x_{i-(n-1)}, \dots, x_{i-1}, x_i)}{count(x_{i-(n-1)}, \dots, x_{i-1})}$$

Neural Language Model

神经语言模型



A Neural Probabilistic Language Model, Yoshua Bengio, etc

Neural Language Model

神经语言模型

- “一只狗在叫”出现一百次 v.s. “一只猫在叫”出现一次
- 对n-gram模型来说, 狗和猫的权重不一致
- 对神经概率语言模型来说:
  - 假定了相似词对于的词向量也相似
  - 概率函数关于词向量是光滑的

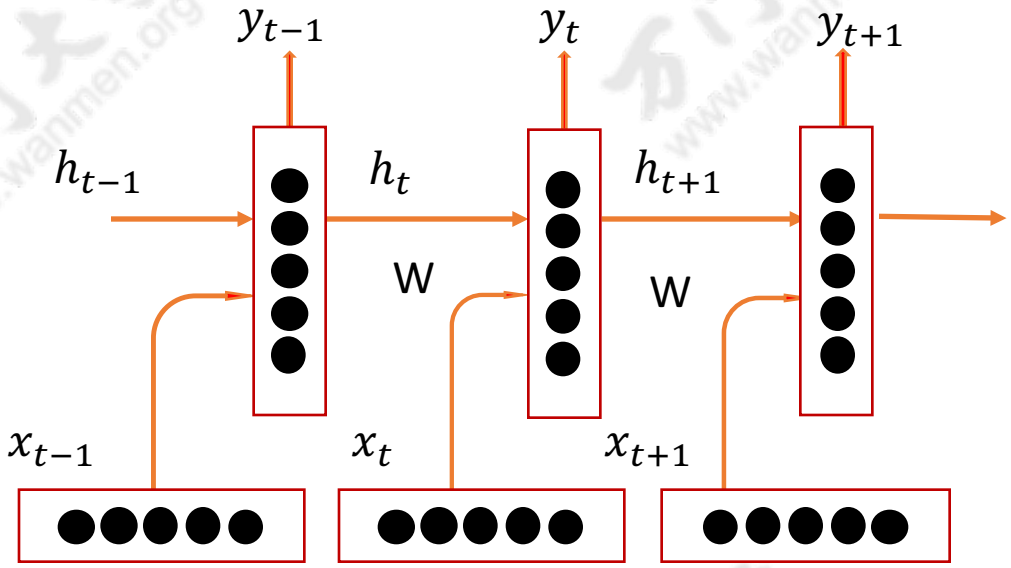
RNN Language Model

递归神经网络语言模型

- 句子: $x_1, \dots x_m$
- 时间t时刻:

$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}e(x_t)\right)$$
$$\hat{y}_t = softmax(W^{(s)}h_t)$$
$$\hat{P}\left(y_{t+1} = v_j \middle| x_t, \dots, x_1\right) = \hat{y}_{t,i}$$

- 结构



Word Embedding

词嵌入

- 独热表示(One-hot Encoding):

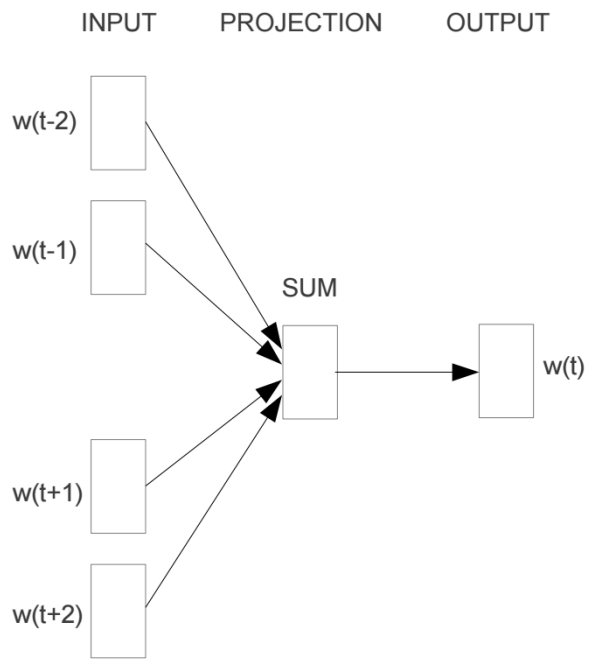
国王	0	1	0	0	0
王后	0	0	0	1	0

- 分布式表示(Distributed Representation):

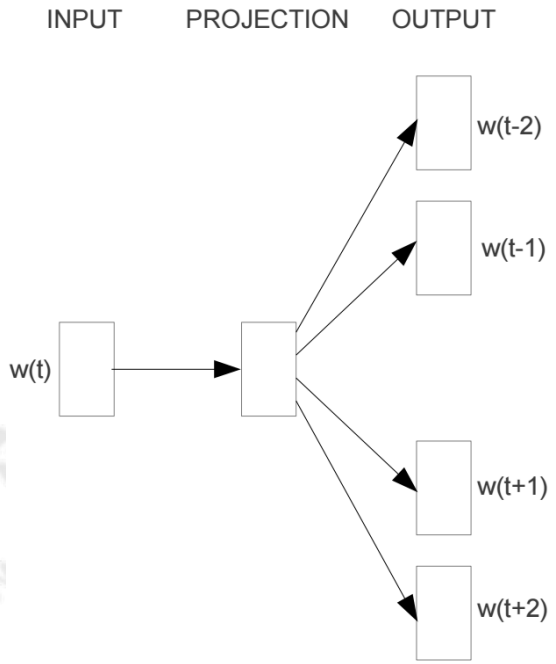
国王	4.3	1.0	0.6	3.2	-0.7
王后	0.25	0.5	-1.2	0.9	1.2

Word Embedding: Skip-Gram & CBOW

词嵌入: Skip-Gram & CBOW



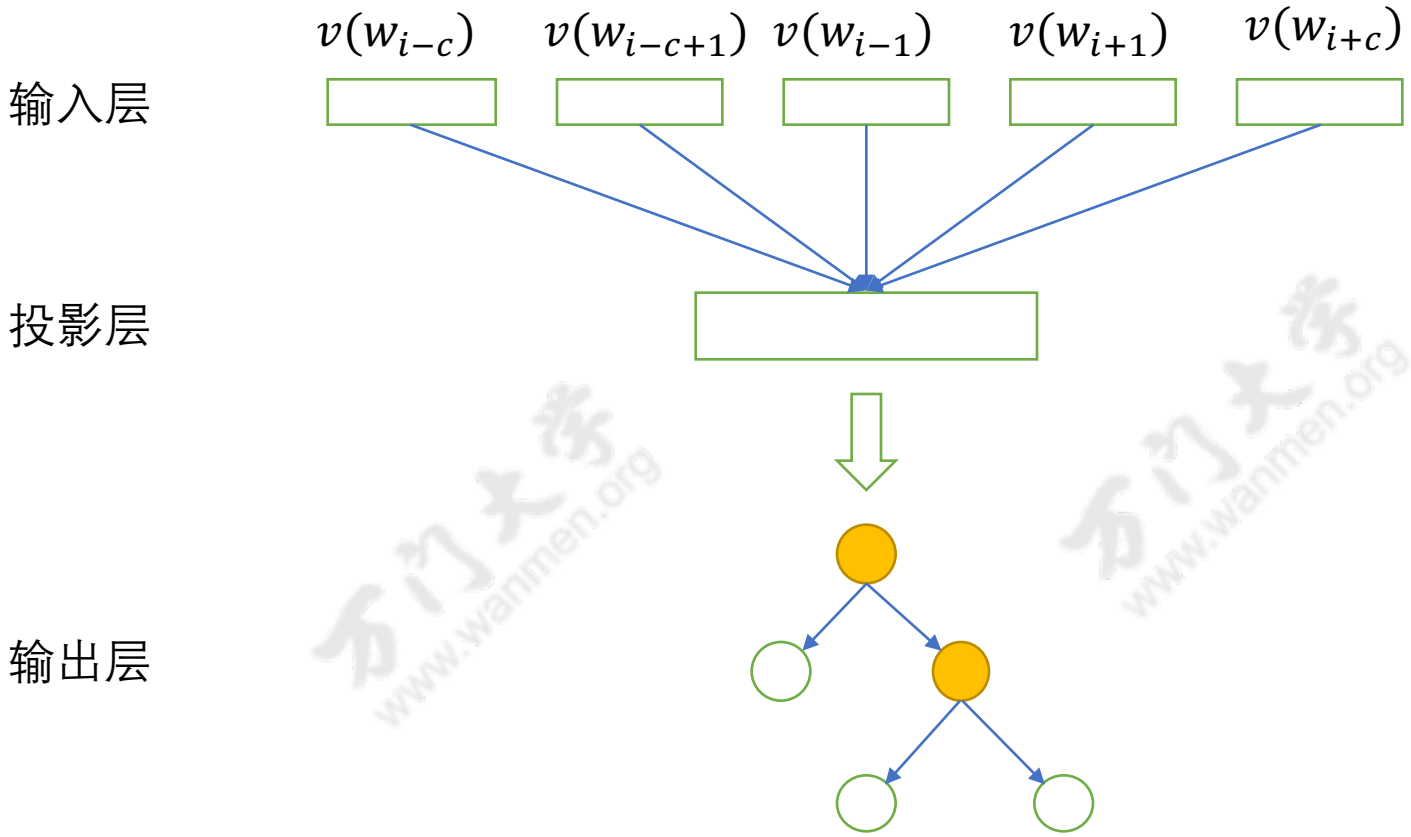
CBOW



Skip-gram

CBOW: Hierarchical Softmax

连续词袋模型: 分层Softmax



$$X_w = \sum_{\substack{j=i-c \\ j \neq i}}^{i+c} v(w_j)$$

Huffman树

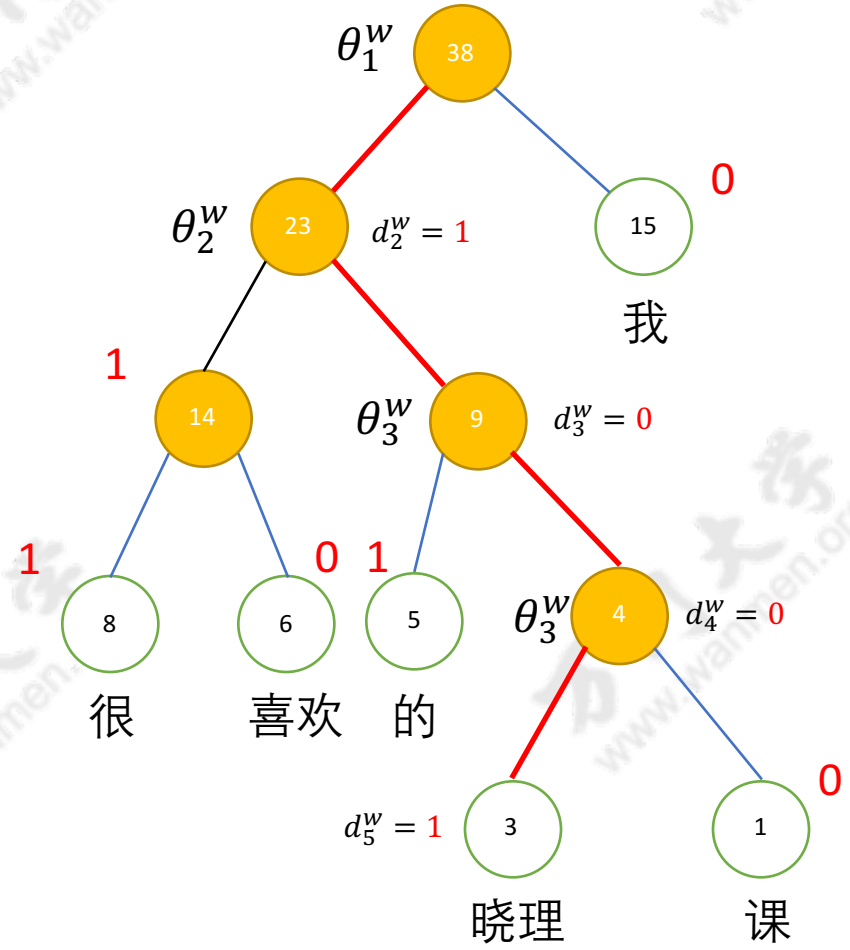
CBOW: Hierarchical Softmax

连续词袋模型: 分层Softmax

- p^w: 从根结点出发达到w对应的叶子结点的路径
- n^w: 路径p^w中包含的结点的个数
- $p_1^w, p_2^w, \ldots, p_{n^w}^w$: 路径$p^w$中的$n^w$结点, p_1^w为根结点, $p_{n^w}^w$为词w对应的结点
- $d_2^w, d_3^w, \ldots, p_{n^w}^w \in \{0,1\}$: 词$w$的Huffman编码, 由$l^w - 1$位编码构成, d_j^w表示路径p^w中第j个结点对应的编码
- $\theta_1^w, \theta_2^w, \ldots, \theta_{n^w}^w \in R^m$: 路径$p^w$中非叶子结点对应的向量

CBOW: Hierarchical Softmax

连续词袋模型: 分层Softmax



- w='晓理'
- $n^w = 5$
- $d_2^w, d_3^w, d_4^w, d_5^w$为1,0,0,1, '晓理'的Huffman编码
- 每一次分支都视为一次二分类
- $Label(p_i^w) = 1 - d_i^w, i = 2,3,\dots,l^w$
- 分为正类概率: $\sigma(X_w^T\theta) = \frac{1}{1+e^{-X_w^T\theta}}$

CBOW: Hierarchical Softmax

连续词袋模型: 分层Softmax

- 计算$p(w_i|w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c})$
- 第i次分支概率: $p(d_i^w|X_w,\theta_{i-1}^w)=\sigma(X_w^T\theta_{i-1}^w)^{1-d_i^w}(1-\sigma(X_w^T\theta_{i-1}^w))^{d_i^w}$
- $p(w_i|w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c})=\prod_{j=2}^{n^w}p(\,d_i^w|X_w,\theta_{j-1}^w)$
- 例如: $p(\text{晓理}|w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c})=\prod_{j=2}^{5}p\,(d_i^w|X_w,\theta_{j-1}^w)$
- 目标函数: $\mathcal{L}=\sum_{w\in C}\log(p(w|\,w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c})$

CBOW: Hierarchical Softmax

连续词袋模型: 分层Softmax

$$\mathcal{L} = \sum_{w \in C} log \prod_{j=2}^{n^w} \left\{ \sigma\left(X_w^T \theta_{j-1}^w\right)^{1-d_j^w} \left(1 - \sigma\left(X_w^T \theta_{j-1}^w\right)\right)^{d_j^w} \right\}$$

$$\mathcal{L}(w,j) = \left(1 - d_j^w\right) \cdot \log\left(\sigma\left(X_w^T \theta_{j-1}^w\right)\right) + d_j^w \cdot \log\left(1 - \sigma\left(X_w^T \theta_{j-1}^w\right)\right)$$

$$\frac{\partial \mathcal{L}(w,j)}{\partial \theta_{j-1}^w} = \left[1 - d_j^w - \sigma\left(X_w^T \theta_{j-1}^w\right)\right] X_w$$

$$\frac{\partial \mathcal{L}(w,j)}{\partial X_w} = \left[1 - d_j^w - \sigma\left(X_w^T \theta_{j-1}^w\right)\right] \theta_{j-1}^w$$

$$v(\tilde{w}) = v(\tilde{w}) + \lambda \sum_{j=2}^{n^w} \frac{\partial \mathcal{L}(w,j)}{\partial X_w}, \qquad \tilde{w} \in Context(w)$$

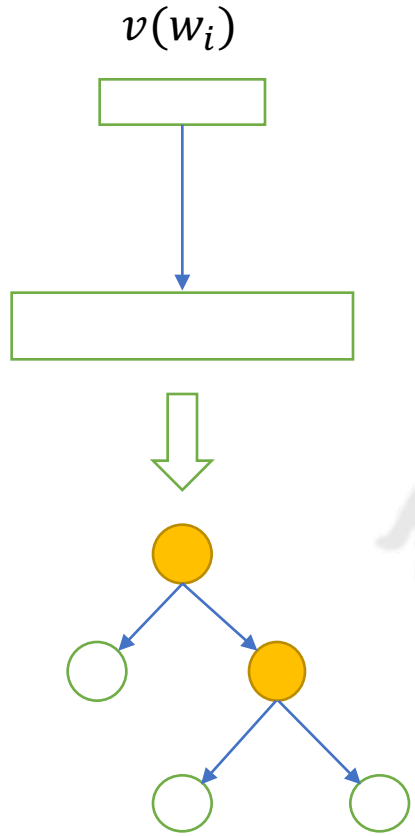
Skip-gram: Hierarchical Softmax

Skip-gram: 分层Softmax

输入层

投影层

输出层



$v(w_i)$

Huffman树

Skip-gram: Hierarchical Softmax

Skip-gram: 分层Softmax

- 计算$p(w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c}|w_i)$
- 第j次分支概率: $p\left(d_j^u\middle|v(w),\theta_{j-1}^u\right)=\sigma\left(v(w)^T\theta_{j-1}^u\right)^{1-d_j^u}\left(1-\sigma\left(v(w)^T\theta_{j-1}^u\right)\right)^{d_j^u}$
- $p(w_{i-c},\dots,w_{i-1},w_{i+1},\dots,w_{i+c}|w_i)=\prod_{j=2}^{n^u}p(\,d_i^u|v(w),\theta_{j-1}^u)$
- 目标函数: $\mathcal{L}=\sum_{w\in C}\log\prod_{u\in context(w)}\prod_{j=2}^{n^u}\left\{\sigma\left(v(w)^T\theta_{j-1}^u\right)^{1-d_j^u}\left(1-\sigma\left(v(w)^T\theta_{j-1}^u\right)\right)^{d_j^u}\right\}$

Skip-gram: Hierarchical Softmax

Skip-gram: 分层Softmax

$$\mathcal{L} = \sum_{w \in C} log \prod_{w \in Context(w)} \prod_{j=2}^{n^u} \left\{ \sigma\left(v(w)^T \theta_{j-1}^u\right)^{1-d_j^u} \left(1 - \sigma\left(v(w)^T \theta_{j-1}^u\right)\right)^{d_j^u} \right\}$$

$$\mathcal{L}(w,u,j) = \left(1 - d_j^u\right) \cdot \log\left(\sigma\left(v(w)^T \theta_{j-1}^u\right)\right) + d_j^u \cdot \log\left(1 - \sigma\left(v(w)^T \theta_{j-1}^u\right)\right)$$

$$\frac{\partial \mathcal{L}(w,u,j)}{\partial \theta_{j-1}^u} = \left[1 - d_j^w - \sigma\left(v(w)^T \theta_{j-1}^u\right)\right] v(w)$$

$$\frac{\partial \mathcal{L}(w,u,j)}{\partial v(w)} = \left[1 - d_j^u - \sigma\left(v(w)^T \theta_{j-1}^u\right)\right] \theta_{j-1}^u$$

$$v(w) = v(w) + \sum_{u \in Context(w)} \sum_{j=2}^{l^u} \frac{\partial \mathcal{L}(w,u,j)}{\partial v(w)}, \qquad \widetilde{w} \in Context(w)$$

CBOW: Negative Sampling

连续词袋模型: 负采样

- 给定w的上下文, 预测w, w为正样本, 其余词为负样本
- $L^w(\tilde{w}) = \begin{cases} 1, \tilde{w} = w \\ 0, \tilde{w} \neq w \end{cases}$
- $p(u|\tilde{w}) = [\sigma(v(\tilde{w})^T\theta^u)]^{L^w(u)} \cdot [1 - \sigma(v(\tilde{w})^T\theta^u)]^{1-L^w(u)}$
- 目标函数: $\mathcal{L} = \sum_{w \in C} \log \prod_{\tilde{w} \in \{w\} \cup NEG^{\tilde{w}}(w)} \prod_{j=2}^{n^u} \{[\sigma(v(\tilde{w})^T\theta^u)]^{L^w(u)} [1 - \sigma(v(\tilde{w})^T\theta^u)]^{1-L^w(u)}\}$
- $\frac{\partial \mathcal{L}(w,\tilde{w},u)}{\partial \theta^u} = [L^w(u) - \sigma(v(\tilde{w})^T\theta^u]v(\tilde{w})$
- $\frac{\partial \mathcal{L}(w,\tilde{w},u)}{\partial v(\tilde{w})} = [L^w(u) - \sigma(v(\tilde{w})^T\theta^u]\theta^u$
- $v(\tilde{w}) = v(\tilde{w}) + \sum_{u \in \{w\} \cup NEG(w)} \frac{\partial \mathcal{L}(w,u)}{\partial X_w}, \ \tilde{w} \in Context(w)$

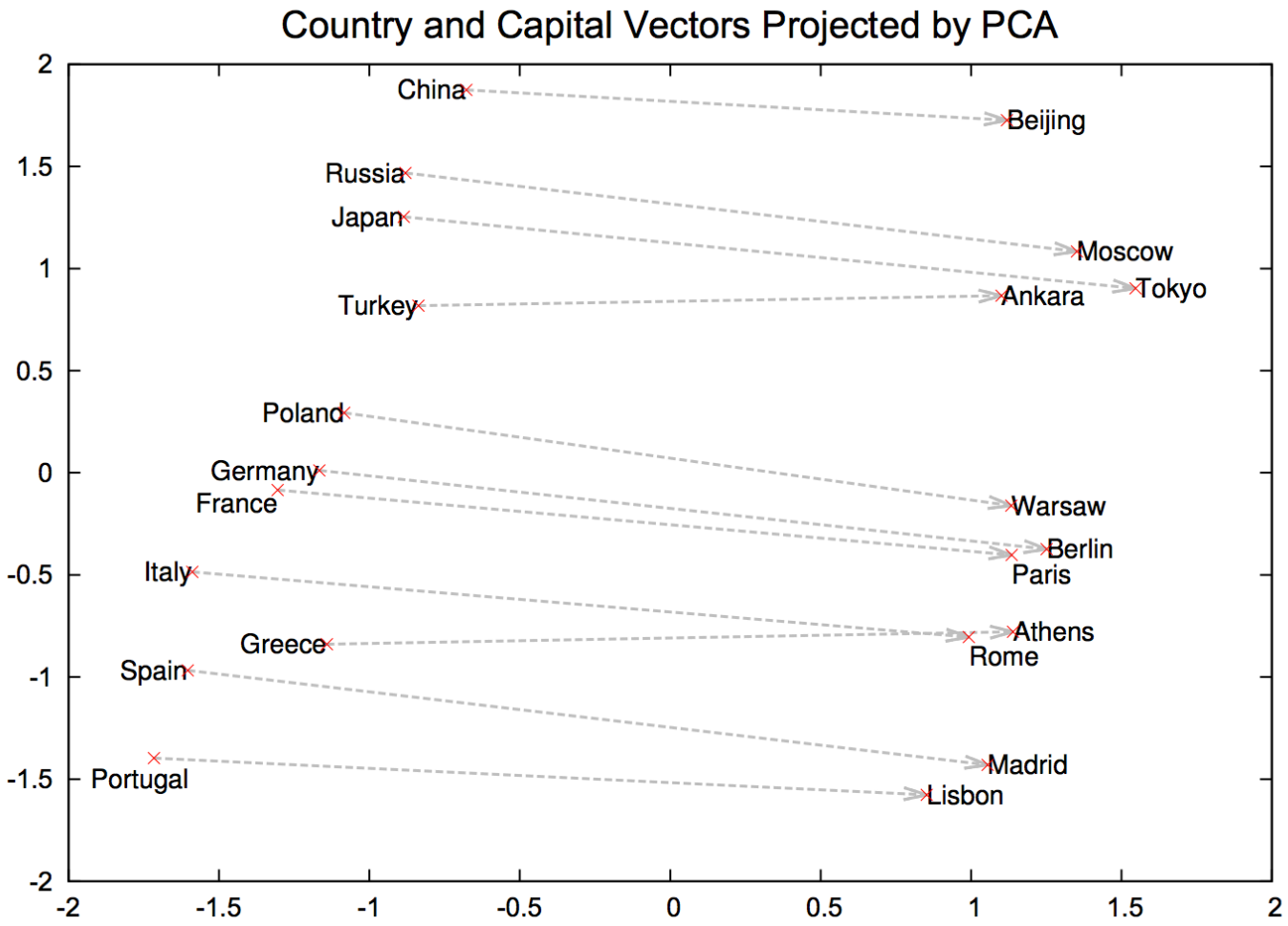
Skip-gram: Negative Sampling

Skip-gram: 负采样

- 给定w的上下文, 预测w, w为正样本, 其余词为负样本
- $L^w(\tilde{w}) = \begin{cases} 1, \tilde{w} = w \\ 0, \tilde{w} \neq w \end{cases}$
- $p(u|Context(w)) = [\sigma(X_w^T\theta^u)]^{L^w(\tilde{w})} \cdot [1 - \sigma(X_w^T\theta^u)]^{1-L^w(\tilde{w})}$
- 目标函数: $\mathcal{L} = \sum_{w \in C} \log \prod_{u \in \{w\} \cup NEG(w)} \prod_{j=2}^{n^u} \left\{ [\sigma(X_w^T\theta^u)]^{1-d_j^u} [1 - \sigma(X_w^T\theta^u)]^{d_j^u} \right\}$
- $\frac{\partial \mathcal{L}(w,u)}{\partial \theta^u} = [L^w(u) - \sigma(X_w^T\theta^u)]X_w$
- $\frac{\partial \mathcal{L}(w,u)}{\partial X_w} = [L^w(u) - \sigma(X_w^T\theta^u]\theta^u$
- $v(\tilde{w}) = v(\tilde{w}) + \sum_{u \in \{w\} \cup NEG(w)} \frac{\partial \mathcal{L}(w,u)}{\partial X_w}, \ \tilde{w} \in Context(w)$

Word Vectors: Visualization

词向量: 可视化

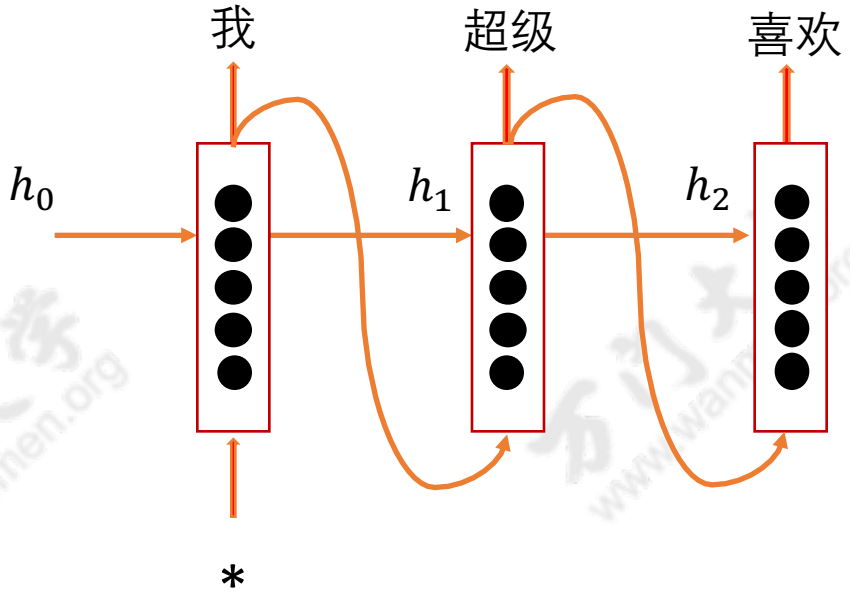


Distributed Representations of Words and Phrases and their Compositionality

4 递归神经网络的应用

RNN Applications: Text Generation

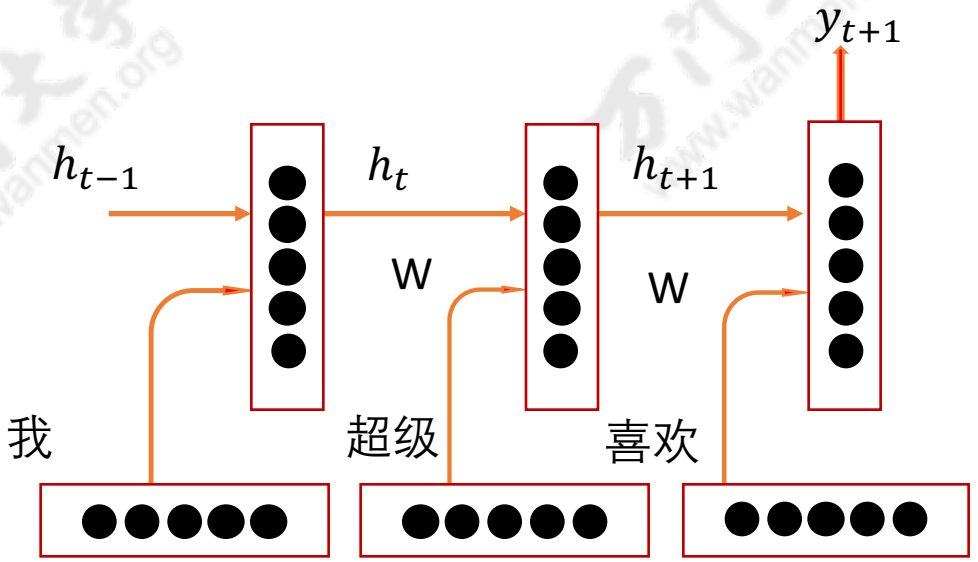
递归神经网络的应用: 文本生成



RNN Applications: Sentiment Analysis

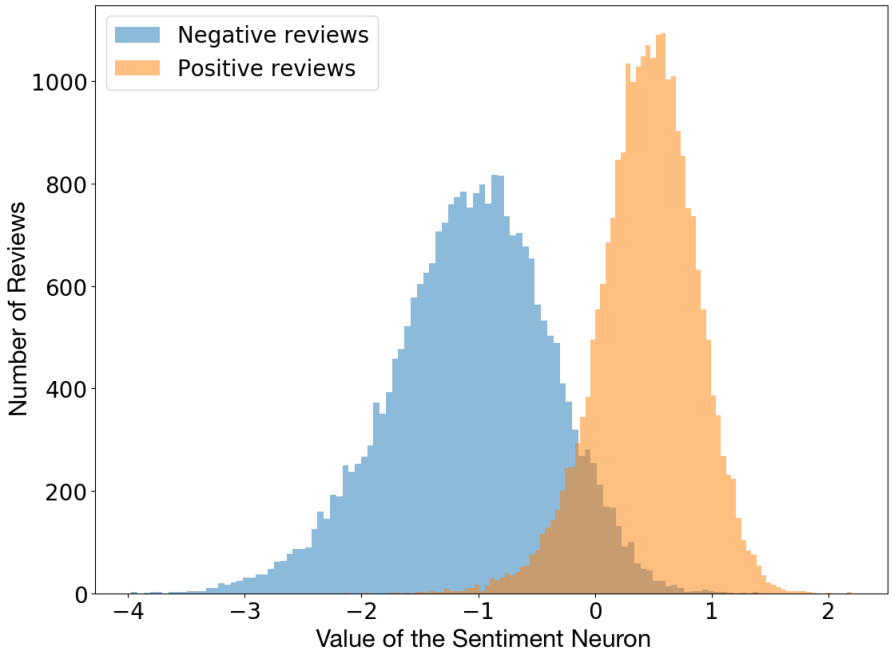
递归神经网络的应用: 情绪分析

- “我超级喜欢晓理的课”
- “这部电影拍得太糟糕了”
- “总体来说, 小说的前半部分比较拖沓, 但后部分非常出彩, 强烈推荐”
- …



RNN Applications: Sentiment Neuron

递归神经网络的应用: 情绪神经元

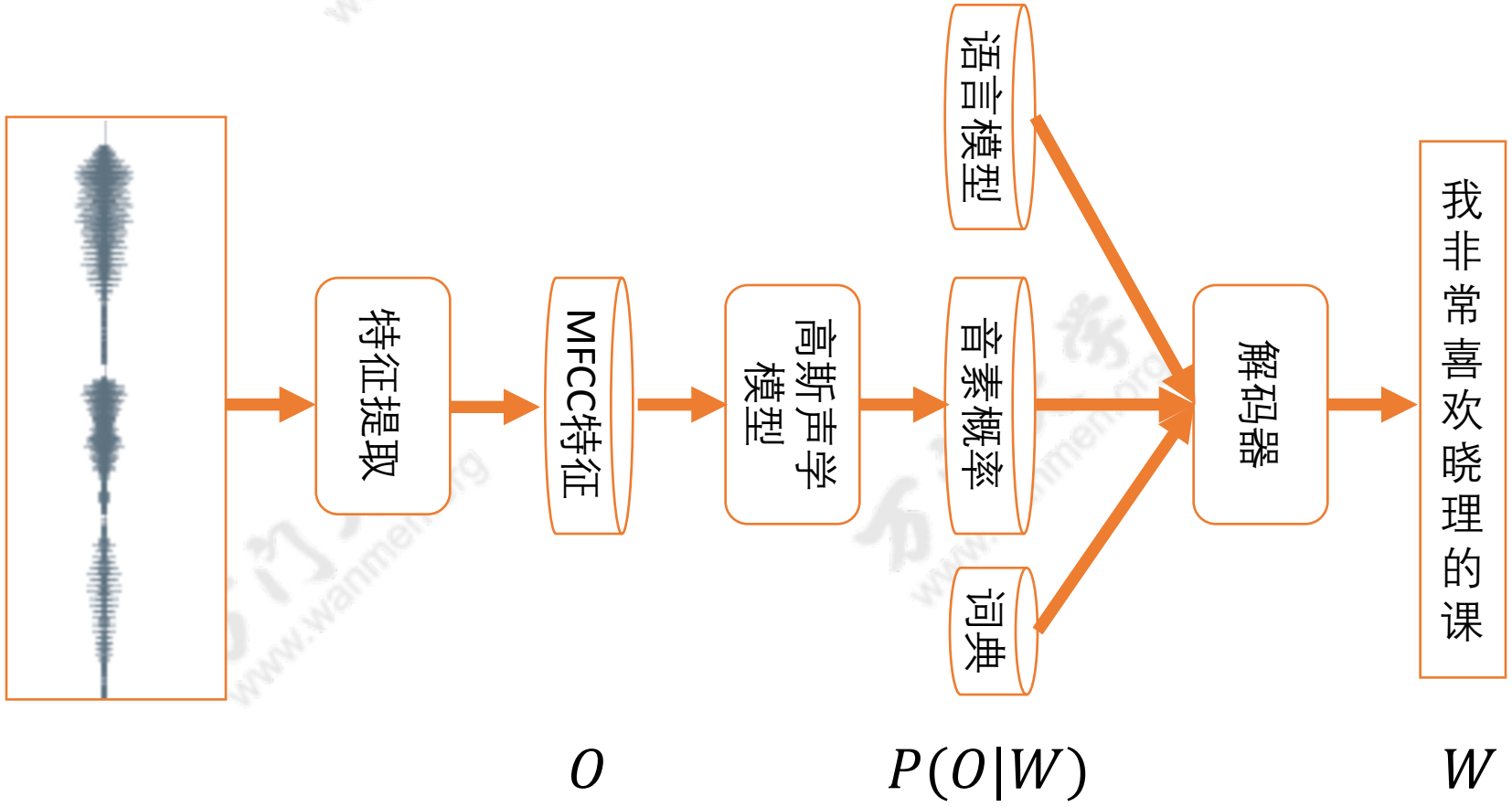


This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

Learning to Generate Reviews and Discovering Sentiment

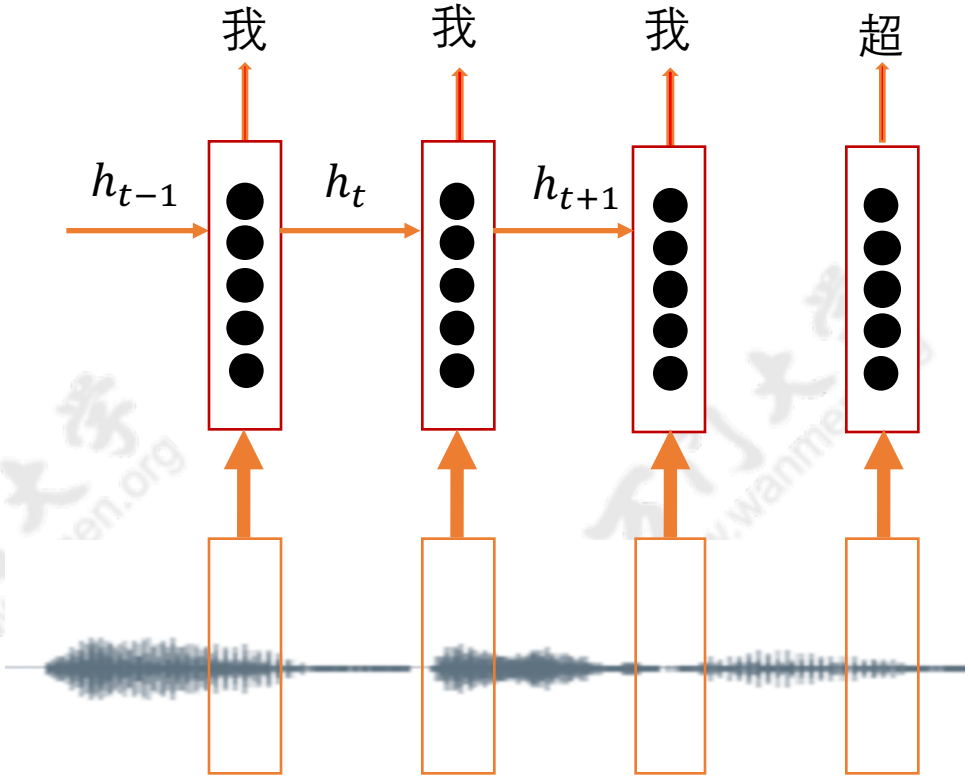
RNN Applications: Speech Recognition

递归神经网络的应用: 语音识别



RNN Applications: Speech Recognition

递归神经网络的应用: 语音识别

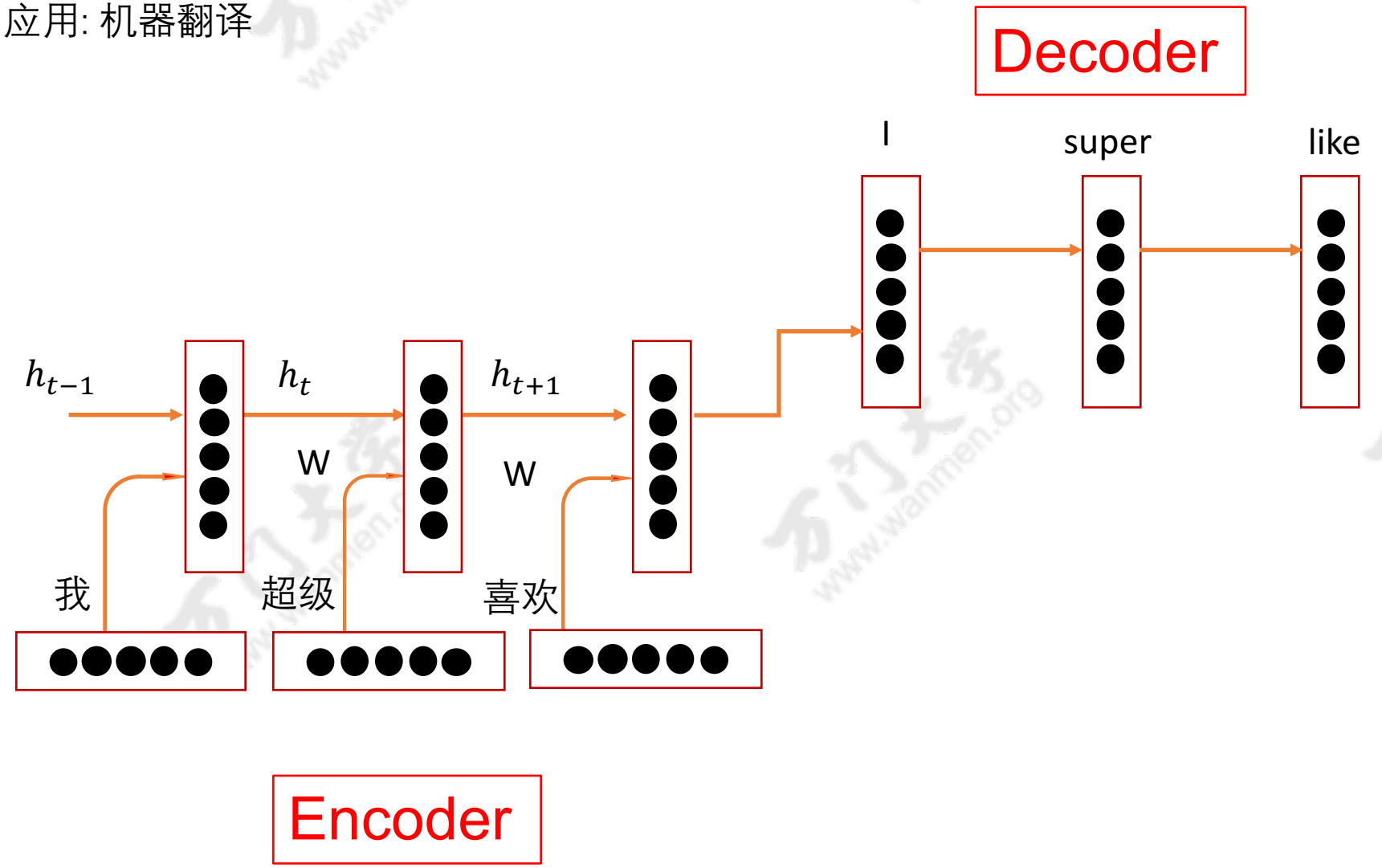


Speech Recognition: CTC

语音识别:

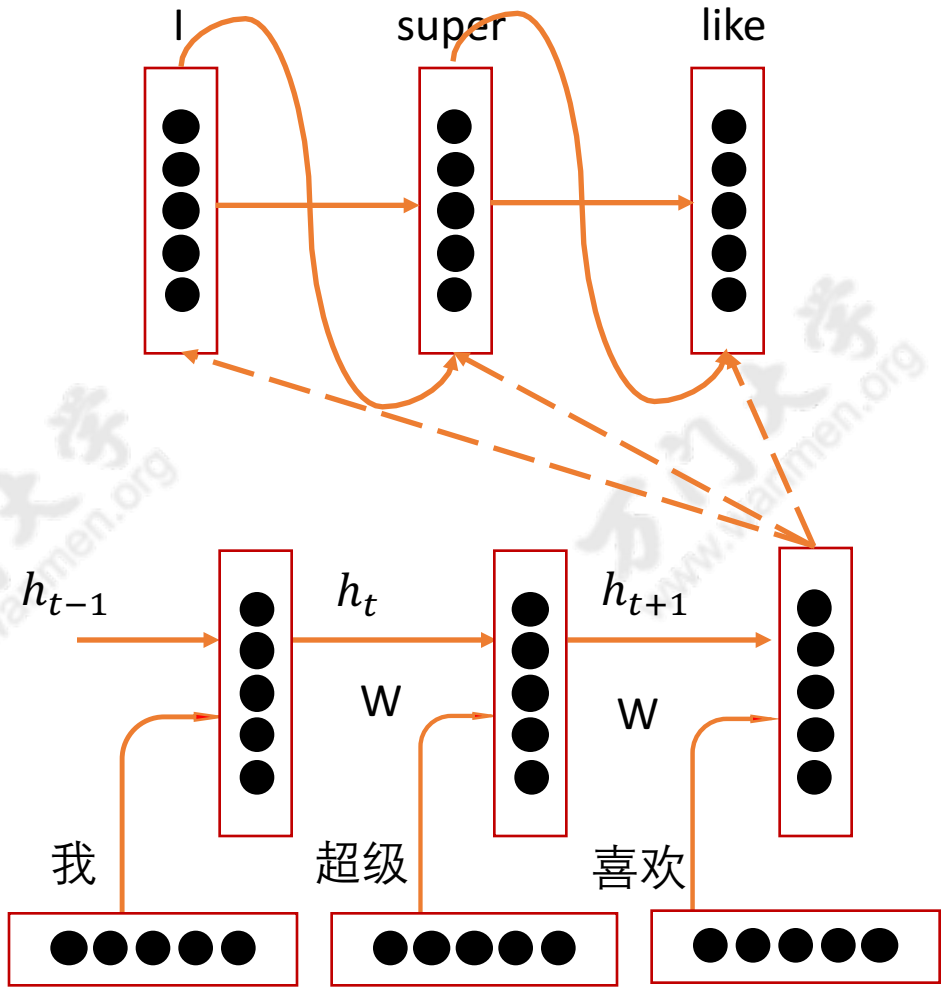
RNN Applications: Machine Translation

递归神经网络的应用: 机器翻译



RNN Applications: Machine Translation

递归神经网络的应用: 机器翻译



RNN Applications: Machine Translation

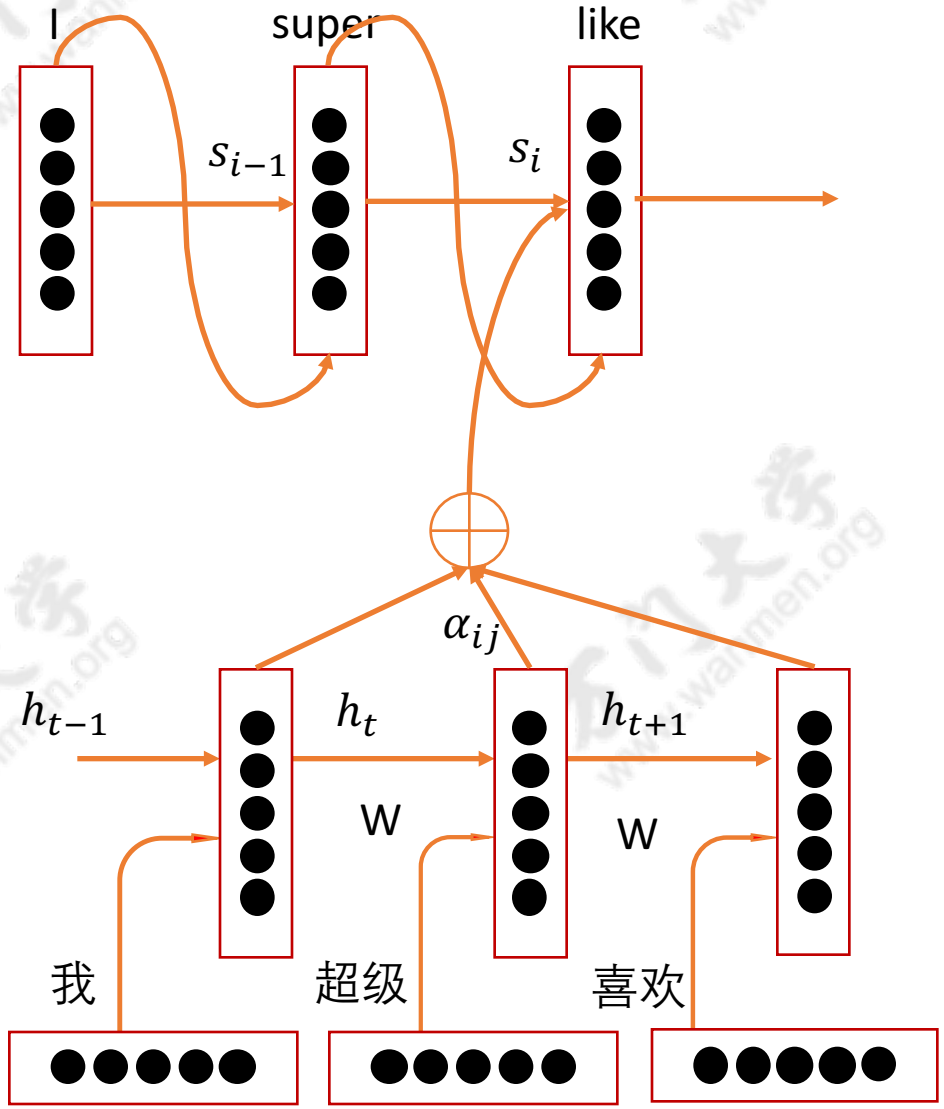
递归神经网络的应用: 机器翻译

- 解码器只依赖与编码器的最后一个hidden state的向量, 编码器只等价于一个sentence embedding

Sequence to Sequence Learning with Neural Networks

RNN Applications: Attention

递归神经网络的应用: 注意力机制



alignment model

$$e_{ij} = a(s_{i-1}, h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

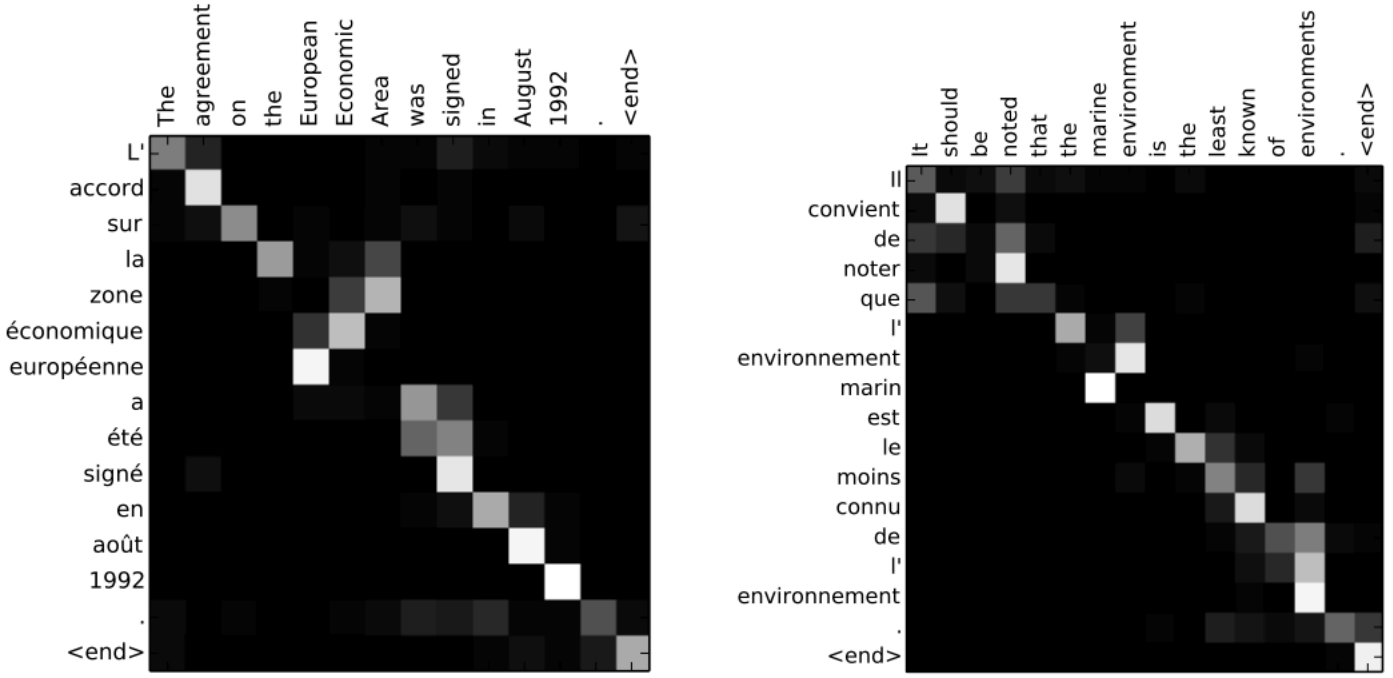
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

Neural machine translation by jointly learning to align and translate

RNN Applications: Attention

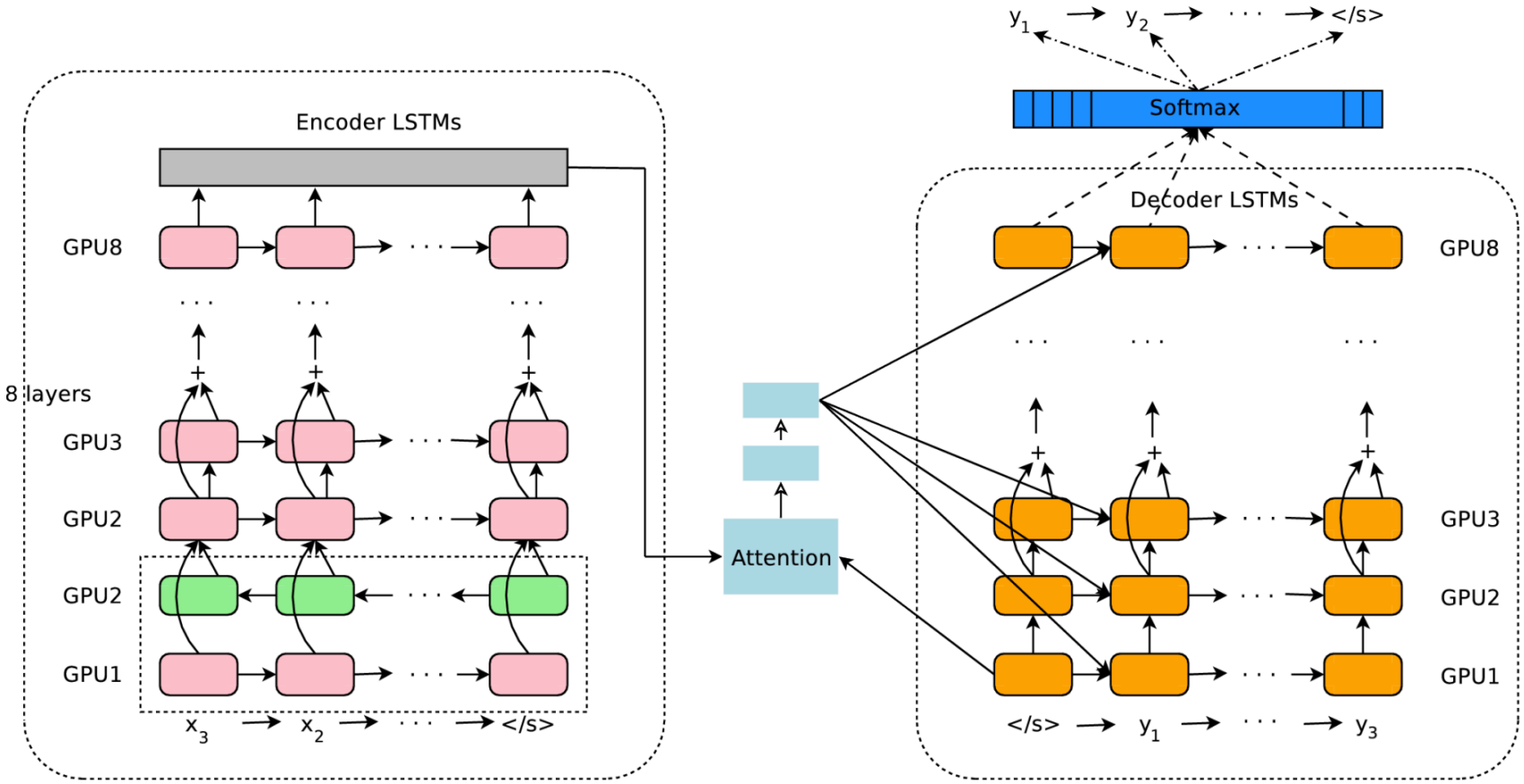
递归神经网络的应用: 注意力机制



Neural machine translation by jointly learning to align and translate

RNN Applications: Machine Translation

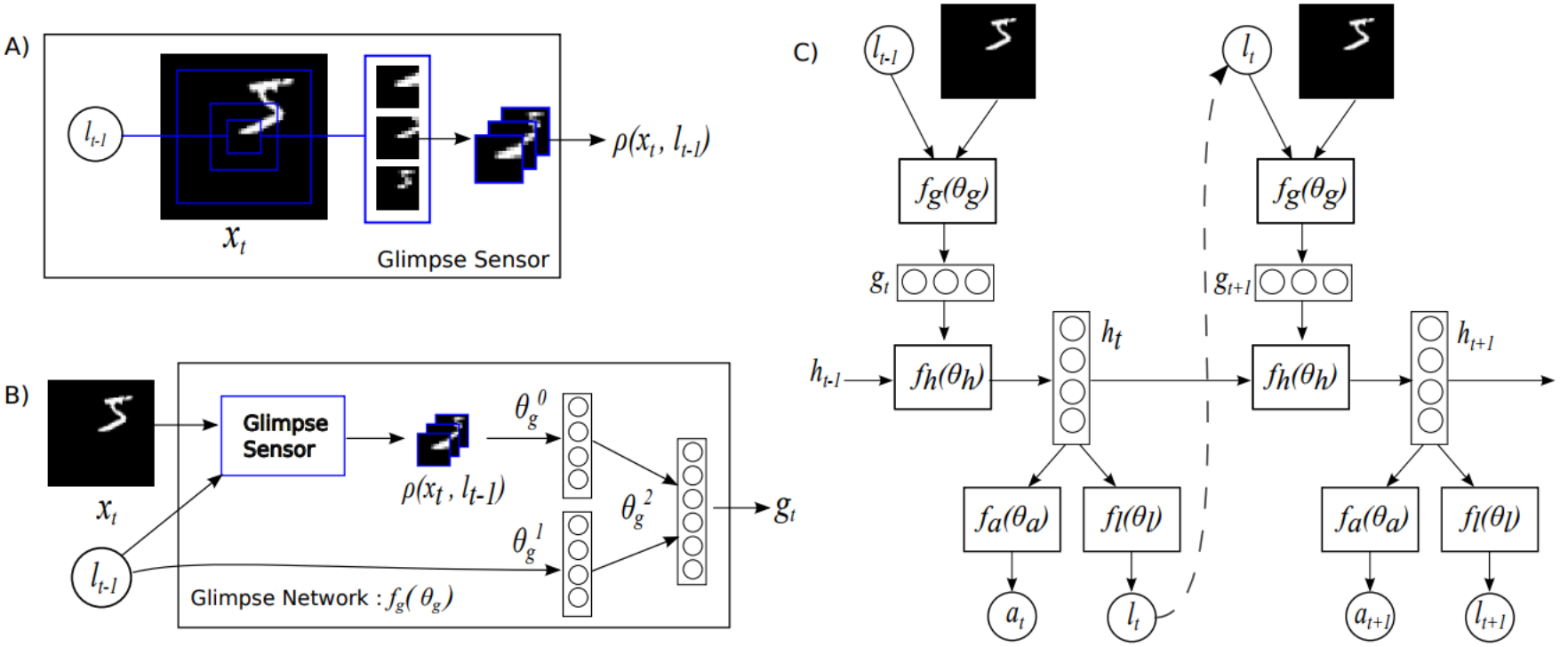
递归神经网络的应用: 机器翻译



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

RNN Applications: Visual Attention

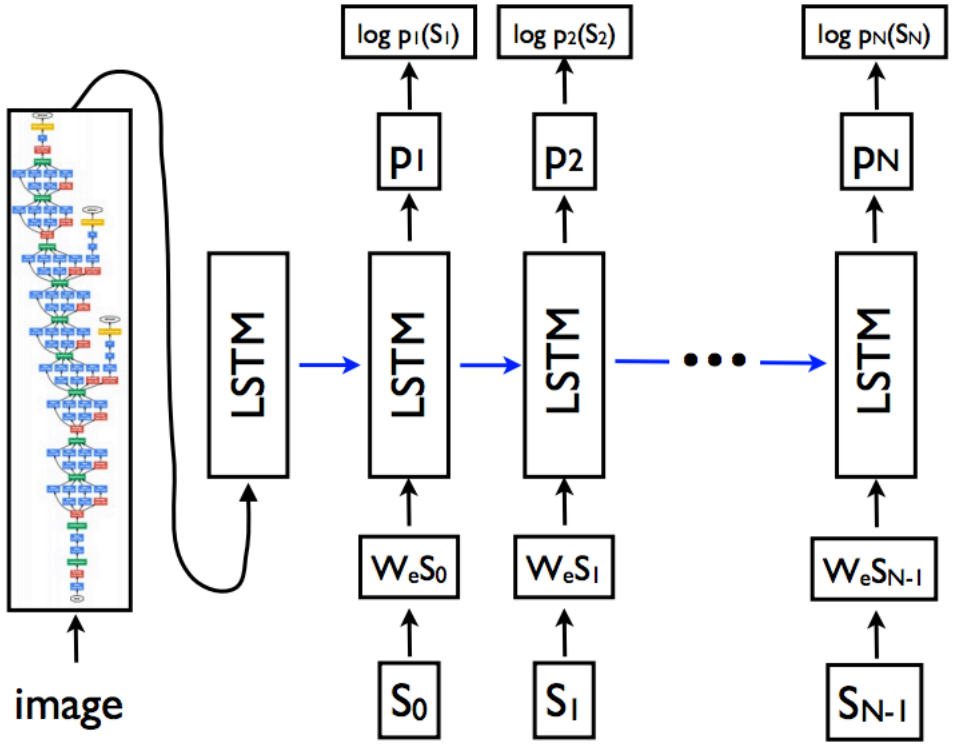
递归神经网络的应用: 视觉注意力机制



Recurrent Models of Visual Attention

RNN Applications: Image Caption

递归神经网络的应用: 视觉注意力机制



Show and Tell: A Neural Image Caption Generator

RNN Applications: Visual Attention

递归神经网络的应用: 视觉注意力机制

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

THANKS.