神经网络语言模型

北京市海淀区中关村东路95号
邮编：100190

电话：+86-10-82544688
邮件：cqzong@nlpr.ia.ac.cn

基于计数的N-元语言模型

$$
\begin{aligned}
& P(w_1 w_2 \cdots w_{t-1} w_n) \\
& = \prod_{t=1}^{n} P(w_t | w_{t-1} \cdots w_1) \\
& \approx \prod_{t=1}^{n} P(w_t | w_{t-1} \cdots w_{t-n+1})
\end{aligned}
$$

$$\downarrow$$

$$
P(w_t | w_{t-1} \cdots w_{t-n+1}) = \frac{count(w_t w_{t-1} \cdots w_{t-n+1})}{count(w_{t-1} \cdots w_{t-n+1})}
$$

基于计数的N-元语言模型

该 课程 很 枯燥 ， 大家 觉得 很 无聊 。

$$P(\text{无聊}|\text{很}) = \frac{count(\text{很 无聊})}{count(\text{很})}$$

问题①：数据稀疏

N-元组"很 无聊"未出现过，则回退

问题②：忽略语义相似性

"无聊"与"枯燥"虽语义相似，但无法共享信息

基于计数的N-元语言模型

该 课程 很 枯燥， 大家 觉得 很 无聊 。

$$P(\text{无聊}|\text{很}) = \frac{count(\text{很 无聊})}{count(\text{很})}$$

$P(\text{无聊}|$

vs.

$P(\text{枯燥}|$

问题①：数据稀疏

N-元组“很 无聊”未出现过，则回退

问题②：忽略语义相似性

“无聊”与“枯燥”虽语义相似，但无法共享信息

词语表示

- 典型方法：抽象符号（字符串）

该 课程 很 **枯燥** ， 大家 觉得 很 **无聊** 。

w_0=该 w_1=课程 w_2=很 w_3=枯燥 w_4=，
w_5=大家 w_6=觉得 w_7=很 w_8=无聊 w_9=。

- 等价表示方法：one-hot表示法

$|V|$

$$\begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}$$

所有词按照出现的顺序排序

每个词语将对应唯一的下标

枯燥	无聊
$\begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \mathbf{1} \\ 0 \end{bmatrix}$

词语表示

- 问题

枯燥　　无聊

$$\begin{bmatrix}0\\1\\0\\\vdots\\0\\0\\0\end{bmatrix}\qquad\begin{bmatrix}0\\0\\0\\\vdots\\0\\1\\0\end{bmatrix}$$

枯燥 $\otimes$ 无聊

$$\begin{bmatrix}0\\1\\0\\\vdots\\0\\0\\0\end{bmatrix}\times\begin{bmatrix}0\\0\\0\\\vdots\\0\\1\\0\end{bmatrix}=0$$

$\Rightarrow$ 任意两个词之间的相似度都为0！

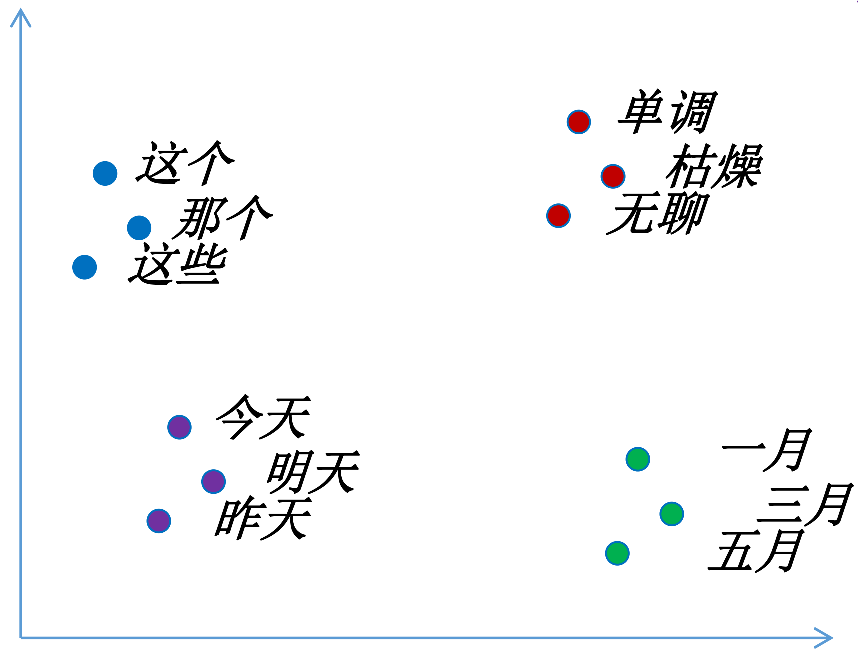
现实世界 VS. 认知世界

- 现实世界：物体相互独立地存在

现实世界 VS. 认知世界

- 认知世界：概念互相联系、语义连续分布

基于连续语义空间的词语表示



枯燥	无聊
$\begin{bmatrix}0.24\\0.15\\0.42\\0.51\\0.21\end{bmatrix}$	$\begin{bmatrix}0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix}$

低维、稠密的连续实数空间

基于分布式表示的N-元语言模型

很

$$\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\end{bmatrix}$$

$P(\text{无聊}|$ **vs.** $P(\text{枯燥}|$

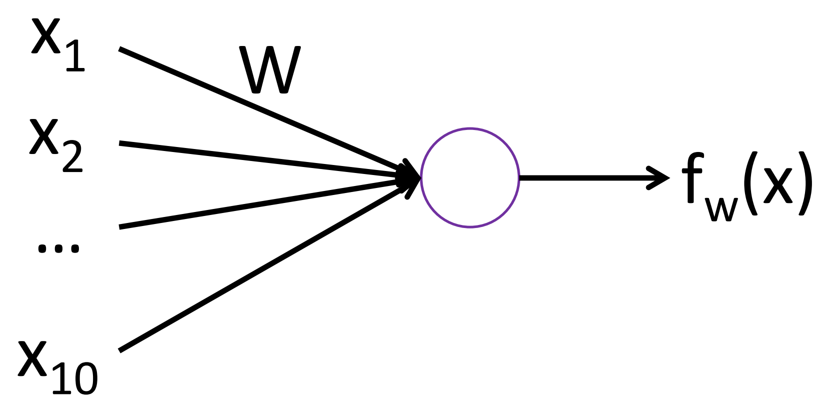
$$P\left(\begin{bmatrix}0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix} \middle| \begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\end{bmatrix}\right) \quad \textbf{vs.} \quad P\left(\begin{bmatrix}0.24\\0.15\\0.42\\0.51\\0.21\end{bmatrix} \middle| \begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\end{bmatrix}\right)$$

$$f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix}\right) \quad \textbf{vs.} \quad f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.24\\0.15\\0.42\\0.51\\0.21\end{bmatrix}\right)$$

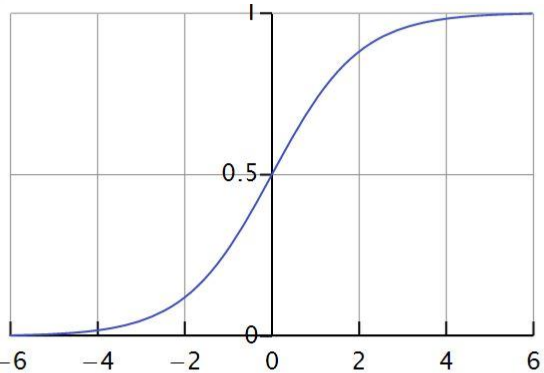
基于分布式表示的N-元语言模型

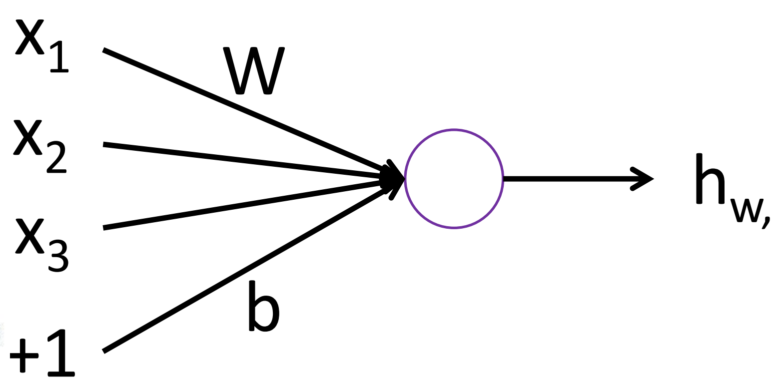
$P($无聊$|$

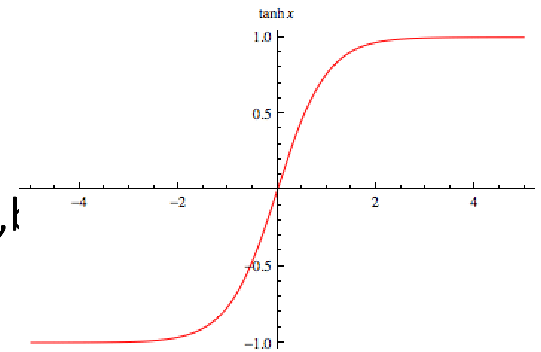
$$
f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix}\right)=f\left(\begin{bmatrix}w_1\times 0.01\\w_2\times 0.59\\w_3\times 0.18\\w_4\times 0.05\\w_5\times 0.47\\w_6\times 0.25\\w_7\times 0.12\\w_8\times 0.39\\w_9\times 0.46\\w_{10}\times 0.26\end{bmatrix}\right)=f(WX)
$$



神经元





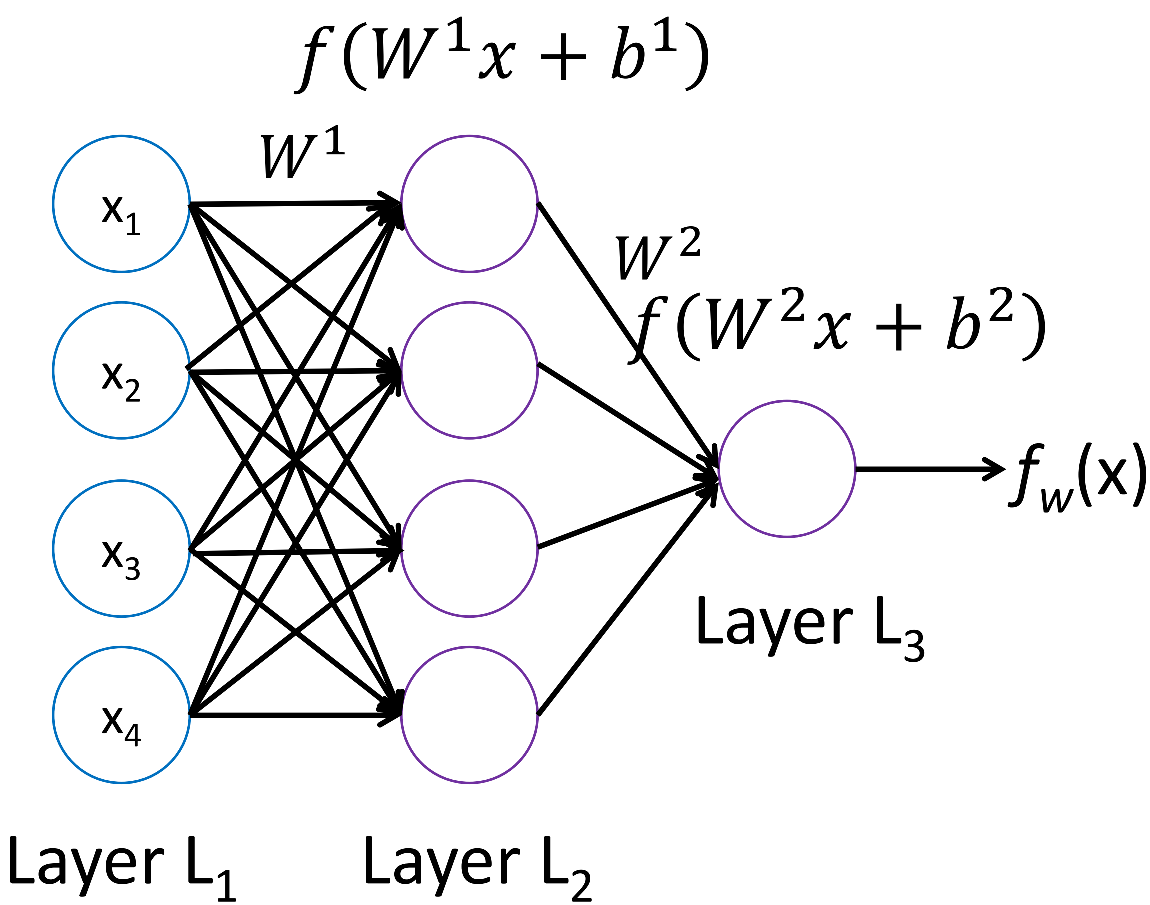


$$h_{W,b}(x) = f(W^T x + b)$$

f: 非线性激活函数

$$\begin{cases} f(z) = \dfrac{1}{1 + exp(-z)} & f'(z) = f(z)\big(1 - f(z)\big) \\ f(z) = tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}} & f'(z) = 1 - f^2(z) \end{cases}$$

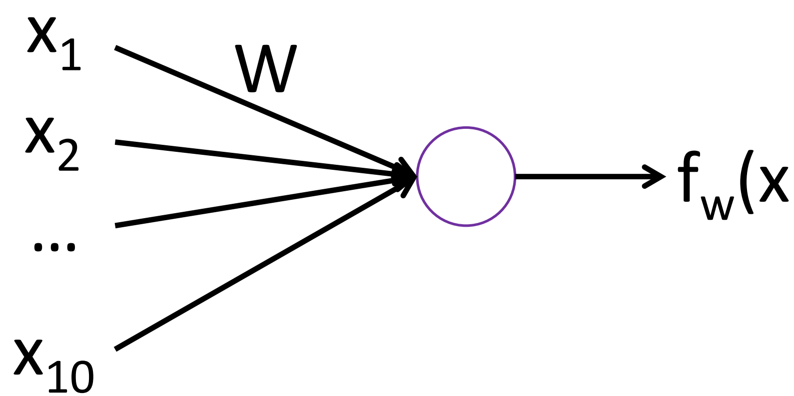
神经网络



基于分布式表示的N-元语言模型

$P(\text{无聊}|$ ***vs.*** $P(\text{枯燥}|$

$$f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix}\right)=f\left(\begin{bmatrix}w_1\times 0.01\\w_2\times 0.59\\w_3\times 0.18\\w_4\times 0.05\\w_5\times 0.47\\w_6\times 0.25\\w_7\times 0.12\\w_8\times 0.39\\w_9\times 0.46\\w_{10}\times 0.26\end{bmatrix}\right)=f(WX)$$



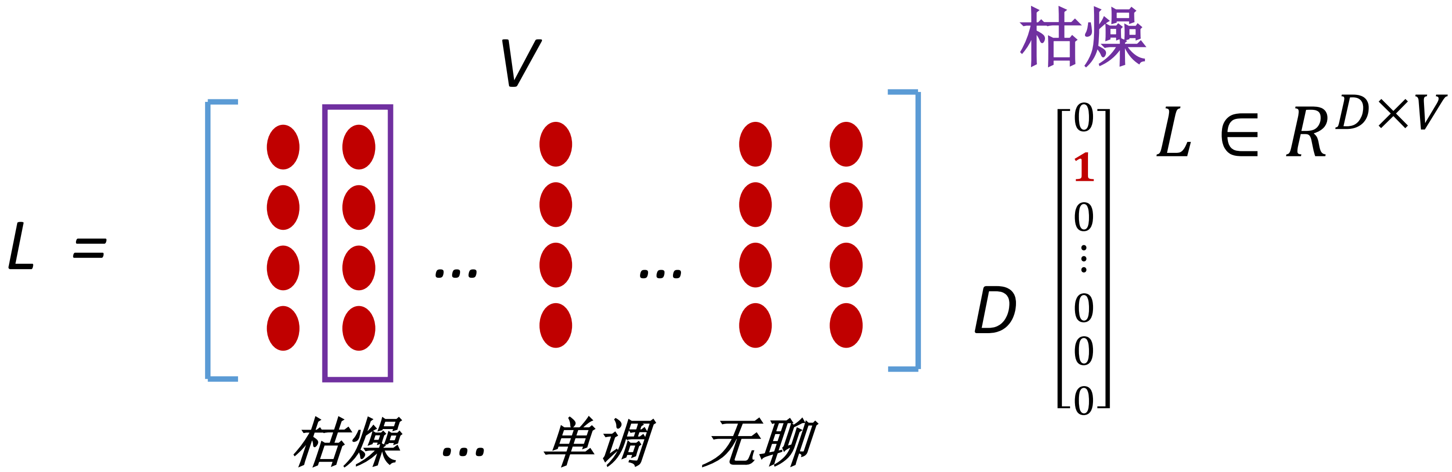
问题①：词向量

如何将每个词映射到实数向量空间中的一个点

问题②：*f*函数的设计

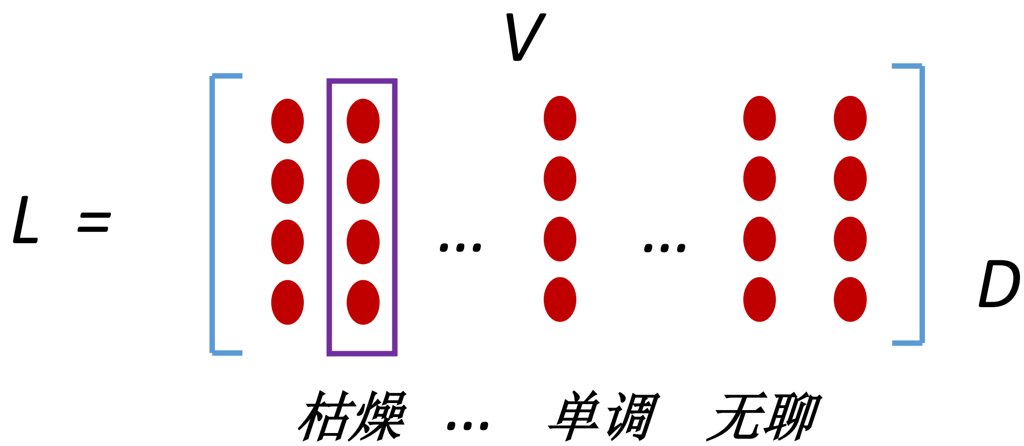
设计什么样的神经网络结构模拟函数*f*

词向量表示



- 通常称为look-up table
  - 我们可以对*L*右乘一个词的one-hot表示 *e* 得到该词的低维、稠密的实数向量表达： $x = Le$

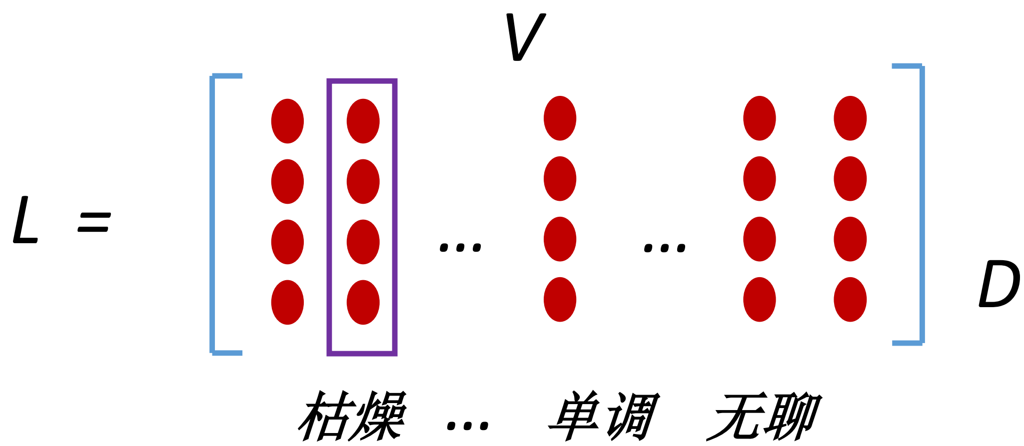
词向量表示



$$L \in R^{D \times V}$$

- 词表规模V和词向量维度D如何确定
  - V的确定：1,训练数据中所有词；2, 频率高于某个阈值的所有词；3, 前V个频率最高的词
  - D的确定：超参数，人工设定，一般从几十到几百

词向量表示



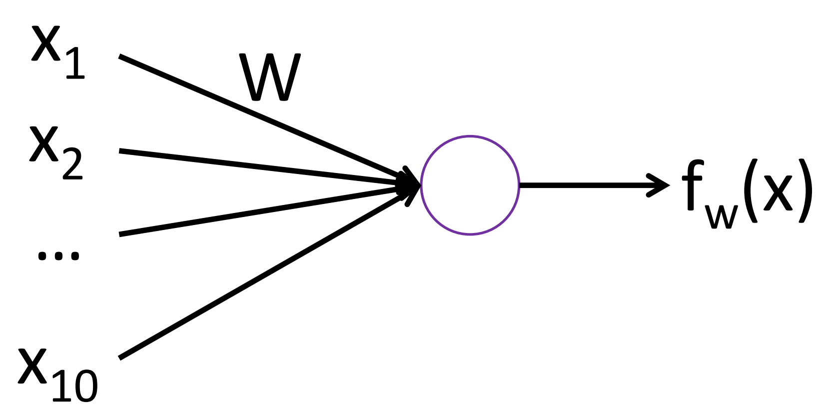
$$L \in R^{D \times V}$$

- 如何学习L
  - 通常先随机初始化，然后通过目标函数优化词的向量表达（e.g. 最大化语言模型似然度）

基于分布式表示的N-元语言模型

$P(\text{无聊}|$ **vs.** $P(\text{枯燥}|$

$$f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.25\\0.12\\0.39\\0.46\\0.26\end{bmatrix}\right)=f\left(\begin{bmatrix}w_1\times 0.01\\w_2\times 0.59\\w_3\times 0.18\\w_4\times 0.05\\w_5\times 0.47\\w_6\times 0.25\\w_7\times 0.12\\w_8\times 0.39\\w_9\times 0.46\\w_{10}\times 0.26\end{bmatrix}\right)=f(WX)$$



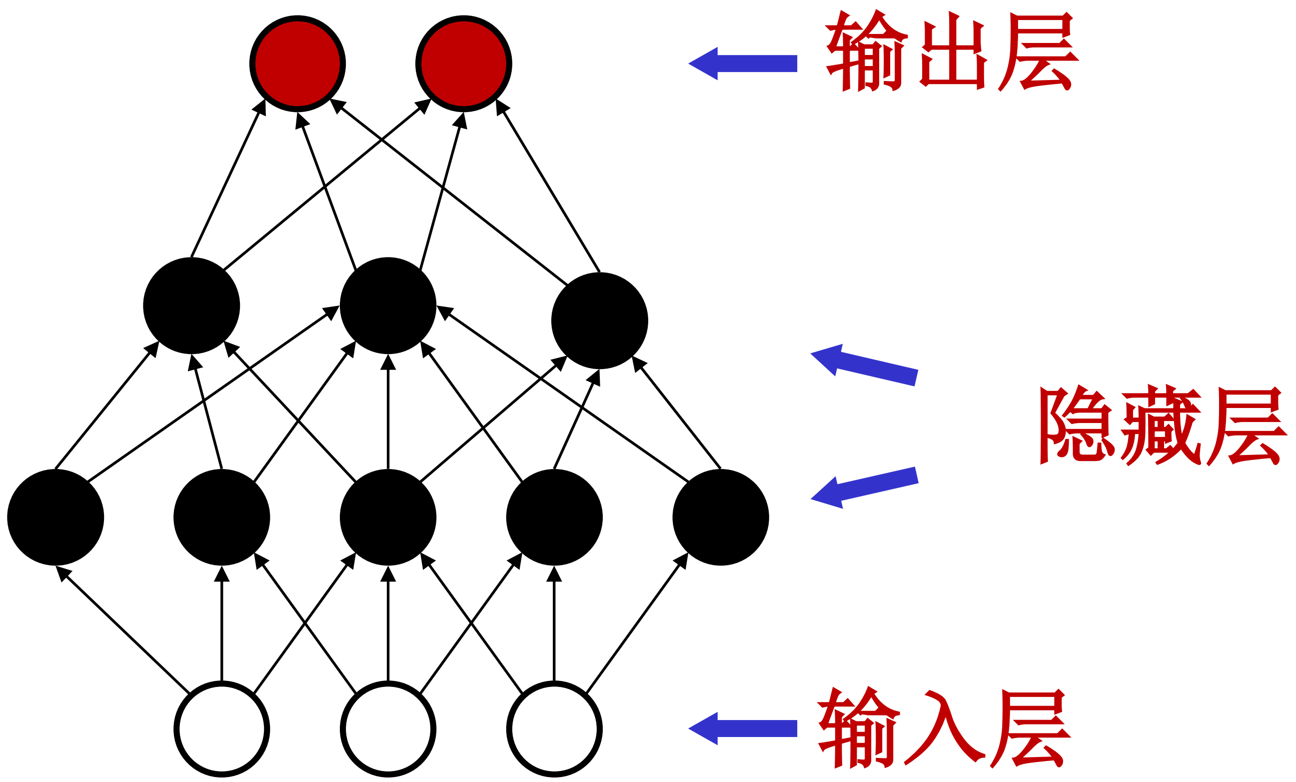
问题①：词向量

如何将每个词映射到实数向量空间中的一个点

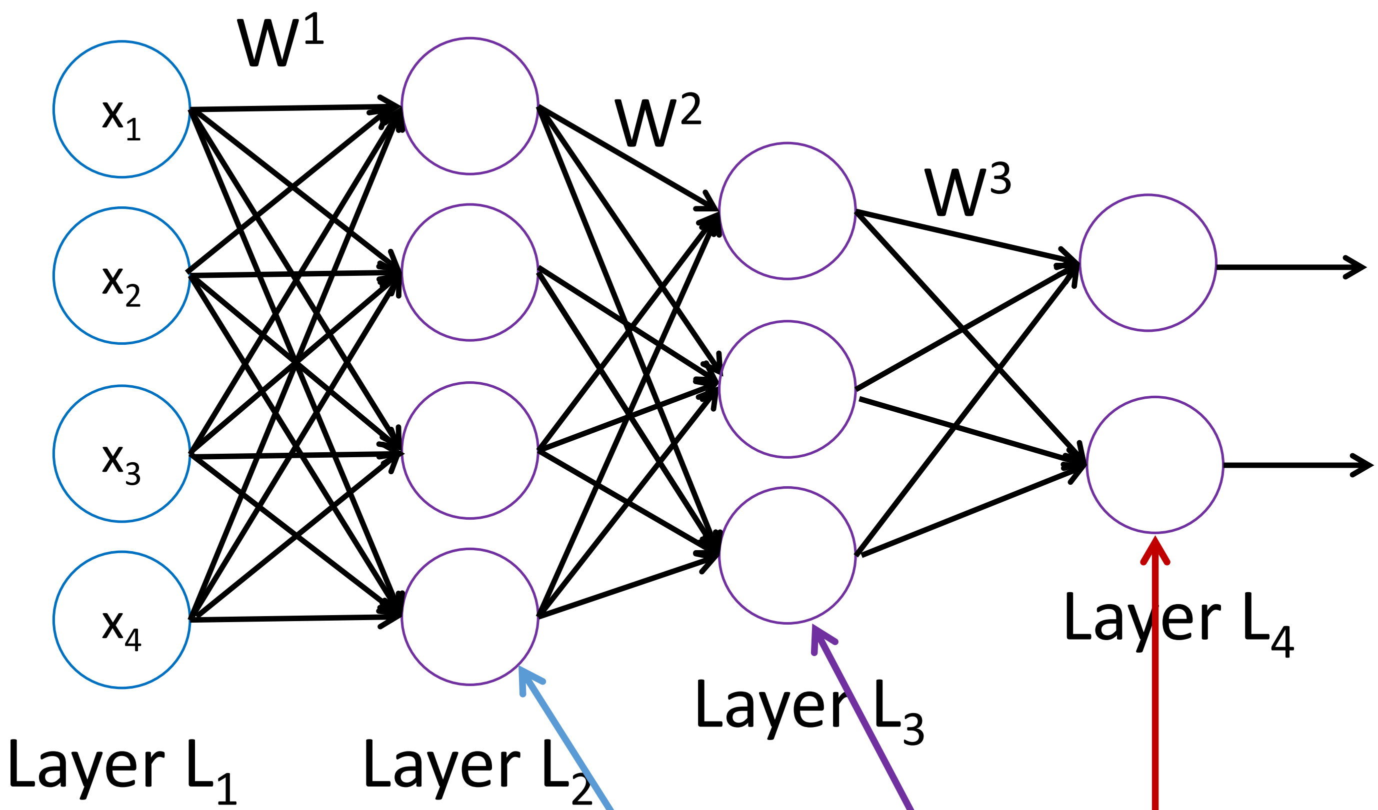
问题②：*f*函数的设计

设计什么样的神经网络结构模拟函数*f*

神经网络

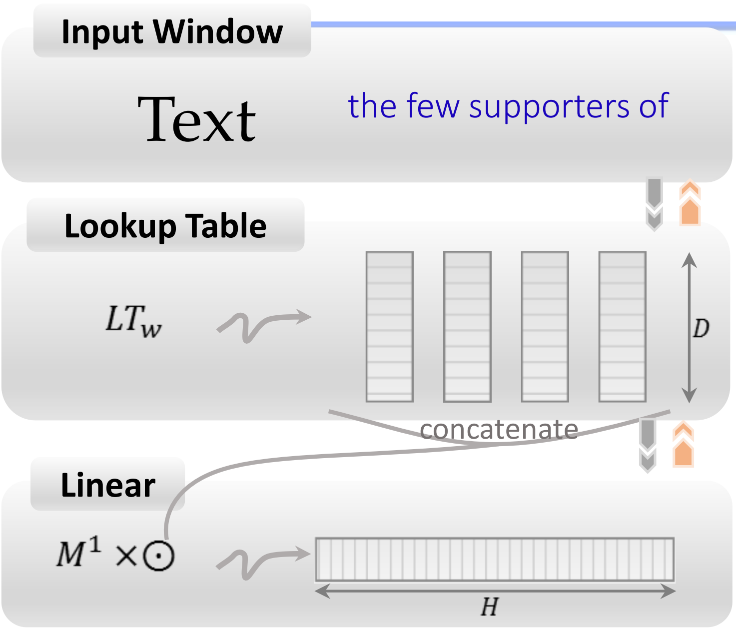


前馈神经网络



$$f\left(W^3\left(f\left(W^2\left(f\left(W^1x+b^1\right)\right)+b^2\right)\right)+b^3\right)$$

语言模型-前馈神经网络



$P(\textit{this}|\textit{the}, \textit{few}, \textit{supporters}, \textit{of})$

将每个词通过词向量矩阵L映射为低维实数向量

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4)

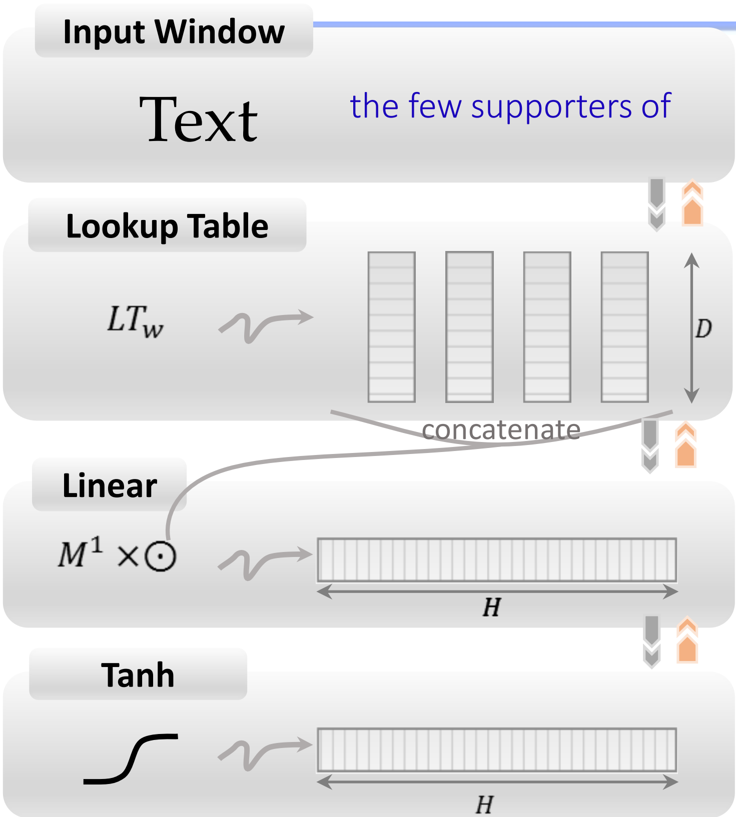
拼接所有词的向量，形成一个向量

the few supporters of
→ (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)

隐藏层:
线性映射+非线性变换

$$\overset{M^1}{\begin{pmatrix} 0.1 & 0 & 0.2 & 0.4 & 0.2 & 0.1 & 0 & 0.3 \\ 0.5 & 0.4 & 0.2 & 0 & 0.2 & 0.6 & 0 & 0.2 \end{pmatrix}} \times \overset{x}{\begin{pmatrix} 0.2 \\ 0.1 \\ 0.1 \\ 0.3 \\ 0.4 \\ 0.2 \\ 0.5 \\ 0.4 \end{pmatrix}} \xrightarrow{M^1 \times x} \begin{pmatrix} 0.3 \\ 0.8 \end{pmatrix}$$

语言模型-前馈神经网络



$P(\textit{this}|\textit{the}, \textit{few}, \textit{supporters}, \textit{of})$

将每个词通过词向量矩阵L映射为低维实数向量

the → (0.2, 0.1); *few* → (0.1, 0.3);
supporters → (0.4, 0.2); *of* → (0.5, 0.4)

拼接所有词的向量，形成一个向量

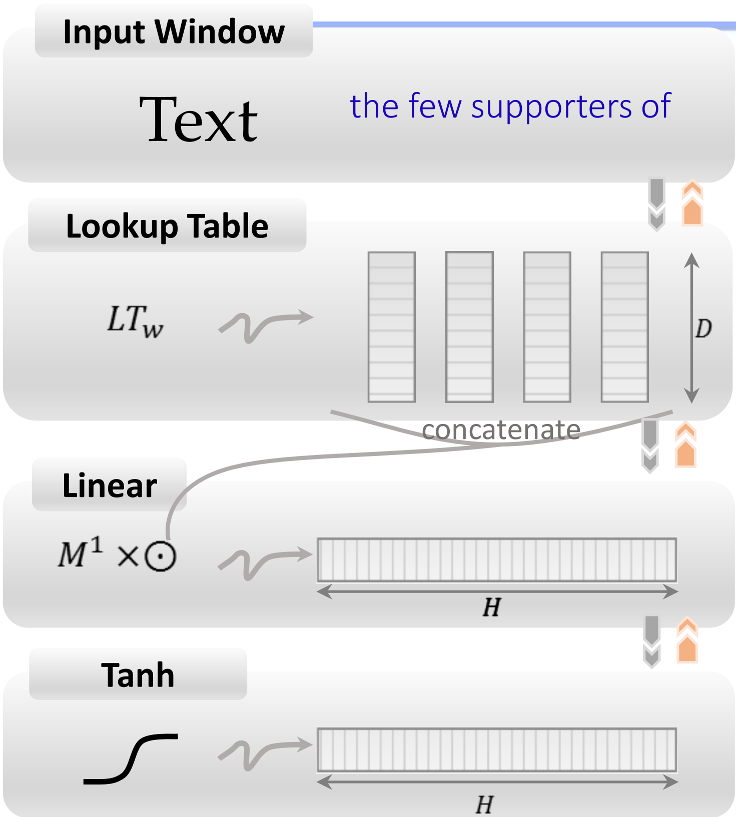
the few supporters of
→ (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)

隐藏层:
线性映射+非线性变换

$M^1 \times x$ $\qquad$ $tanh(M^1 \times x)$

$\begin{pmatrix} 0.3 \\ 0.8 \end{pmatrix}$ → $\begin{pmatrix} 0.1 \\ 0.3 \end{pmatrix}$

语言模型-前馈神经网络



$P(\textbf{this}|the, few, supporters, of)$

将每个词通过词向量矩阵L映射为低维实数向量

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4)

拼接所有词的向量，形成一个向量

the few supporters of
→ (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)

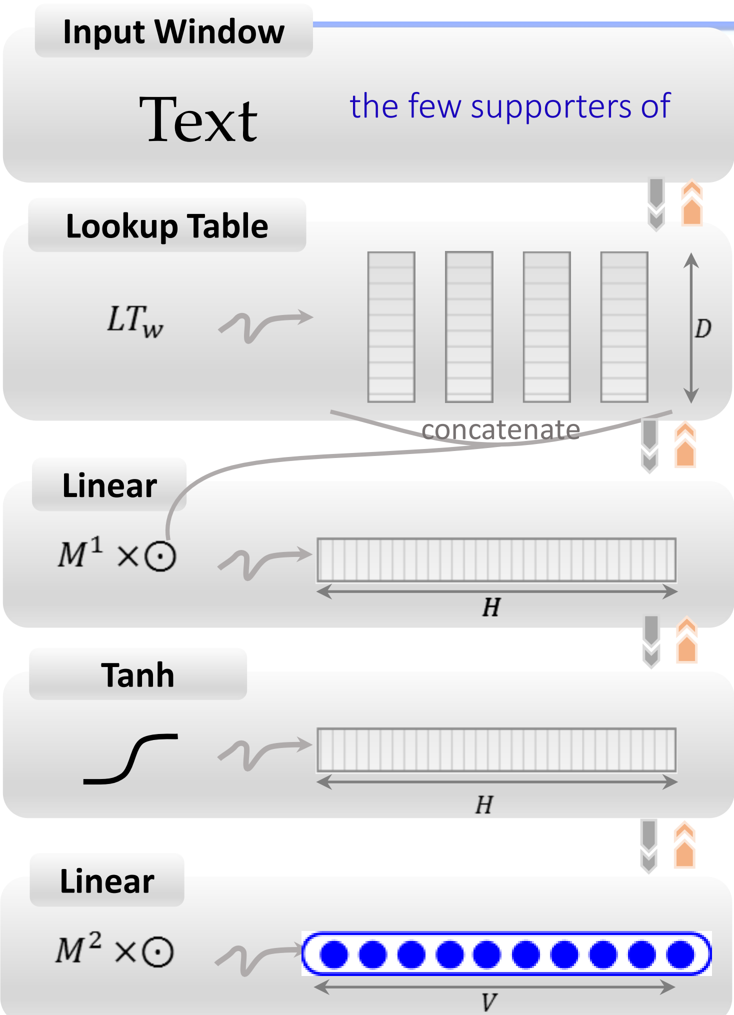
$h \in R^2$

隐藏层:
线性映射+非线性变换

$$V = \{the, few, supporters, of, this\}$$

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4); this → (0.3, 0.2)

语言模型-前馈神经网络



$P(\textbf{this} \mid the, few, supporters, of)$

将每个词通过词向量矩阵L映射为低维实数向量

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4)

拼接所有词的向量，形成一个向量

the few supporters of
→ (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)

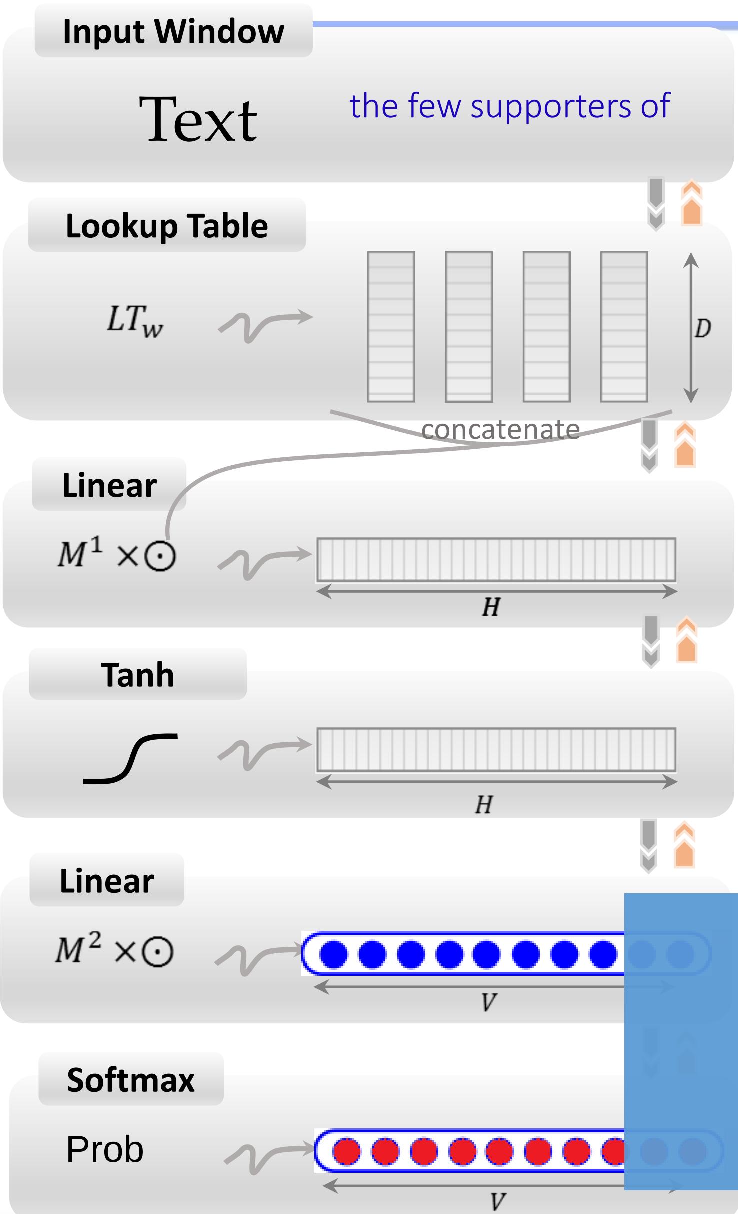
$h \in R^2$

隐藏层:
线性映射+非线性变换

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4); this → (0.3, 0.2)

$$\overset{M^2}{\begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.4 & 0.2 \\ 0.5 & 0.4 \\ 0.3 & 0.2 \end{pmatrix}} \times \overset{h}{\begin{pmatrix} 0.1 \\ 0.3 \end{pmatrix}} \Rightarrow \begin{pmatrix} 0.12 \\ 0.23 \\ 0.34 \\ 0.55 \\ 0.13 \end{pmatrix}$$

语言模型-前馈神经网络



$P(\textbf{this}|\textbf{the}, \textbf{few}, \textbf{supporters}, \textbf{of})$

将每个词通过词向量矩阵L映射为低维实数向量

the → (0.2, 0.1); few → (0.1, 0.3);
supporters → (0.4, 0.2); of → (0.5, 0.4)

拼接所有词的向量，形成一个向量

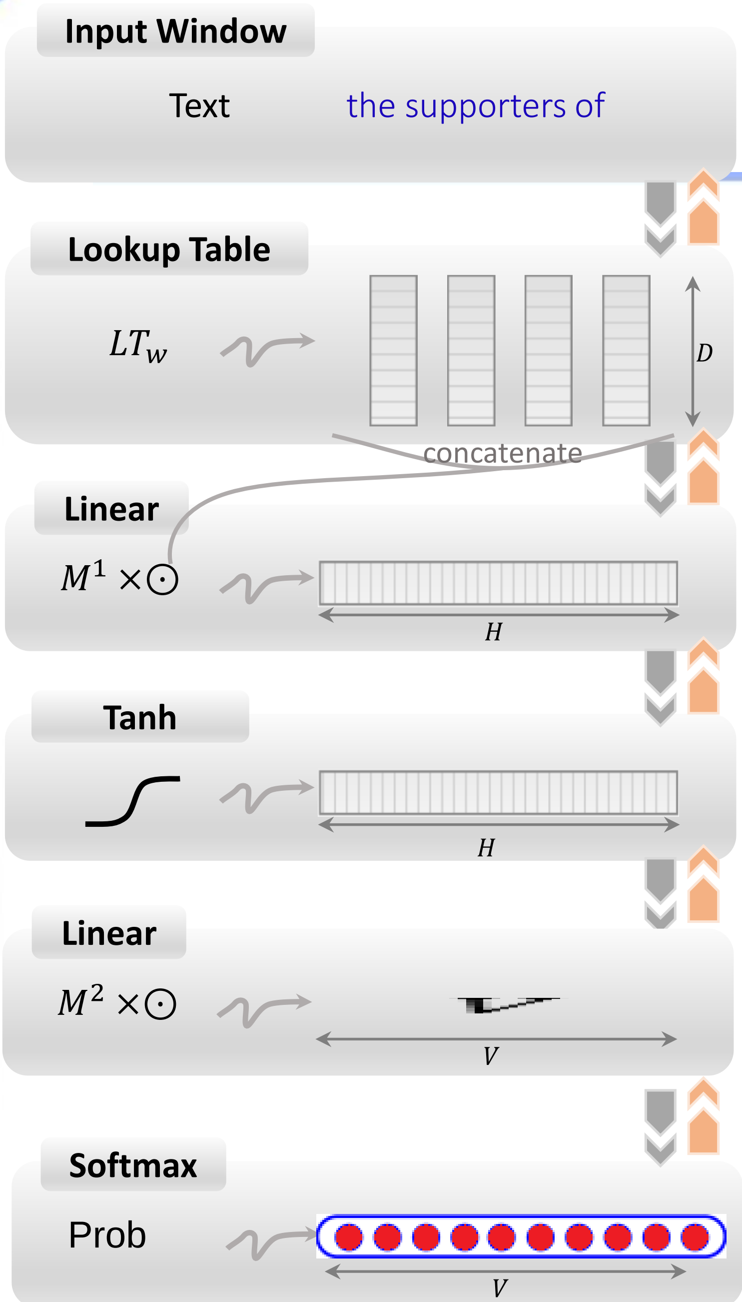
the few supporters of
→ (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)

$h \in R^2$

隐藏层:
线性映射+非线性变换

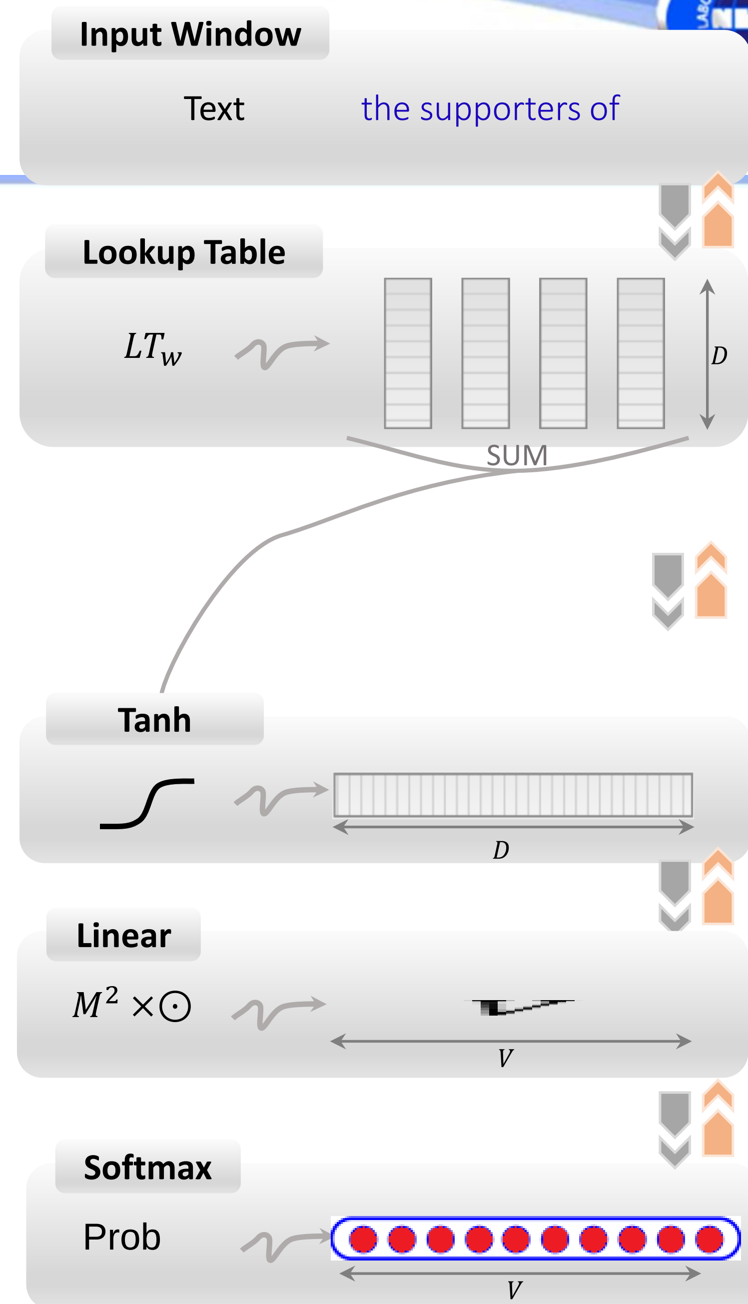
⋮

$$softmax\begin{pmatrix} h \cdot the \\ h \cdot few \\ h \cdot supporters \\ h \cdot of \\ h \cdot this \end{pmatrix} = softmax\begin{pmatrix} \mathbf{0.12} \\ \mathbf{0.23} \\ \mathbf{0.34} \\ \mathbf{0.55} \\ \mathbf{0.13} \end{pmatrix}$$



Input Window
Text
the supporters of
Lookup Table
LT_W
D
concatenate
Linear
$M^1 \times \odot$
H
Tanh
H
Linear
$M^2 \times \odot$
V
Softmax
Prob
V

VS.



Input Window
Text
the supporters of
Lookup Table
LT_W
D
SUM
Tanh
D
Linear
$M^2 \times \odot$
V
Softmax
Prob
V

语言模型-前馈神经网络

- 问题

仅对小窗口的历史信息建模

例如5-gram语言模型，仅考虑前面4个词的历史信息

$$P(w_t|w_{t-1}\cdots w_{t-n+1})$$

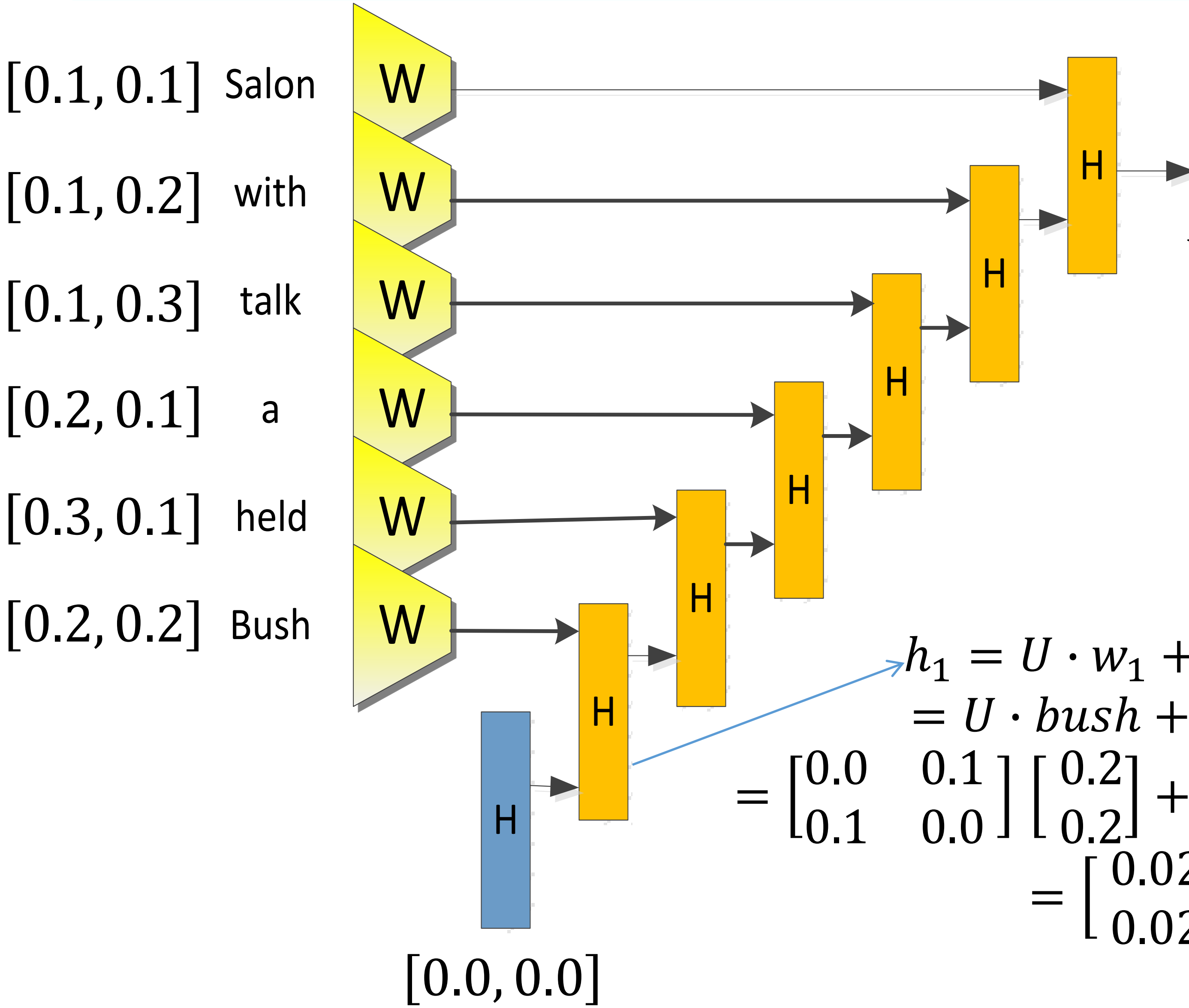
↓

能否对所有的历史信息进行建模

即第t个词的语言模型概率依赖于所有前t-1个词

$$P(w_t|w_{t-1}\cdots w_2 w_1)$$

语言模型-循环神经网络

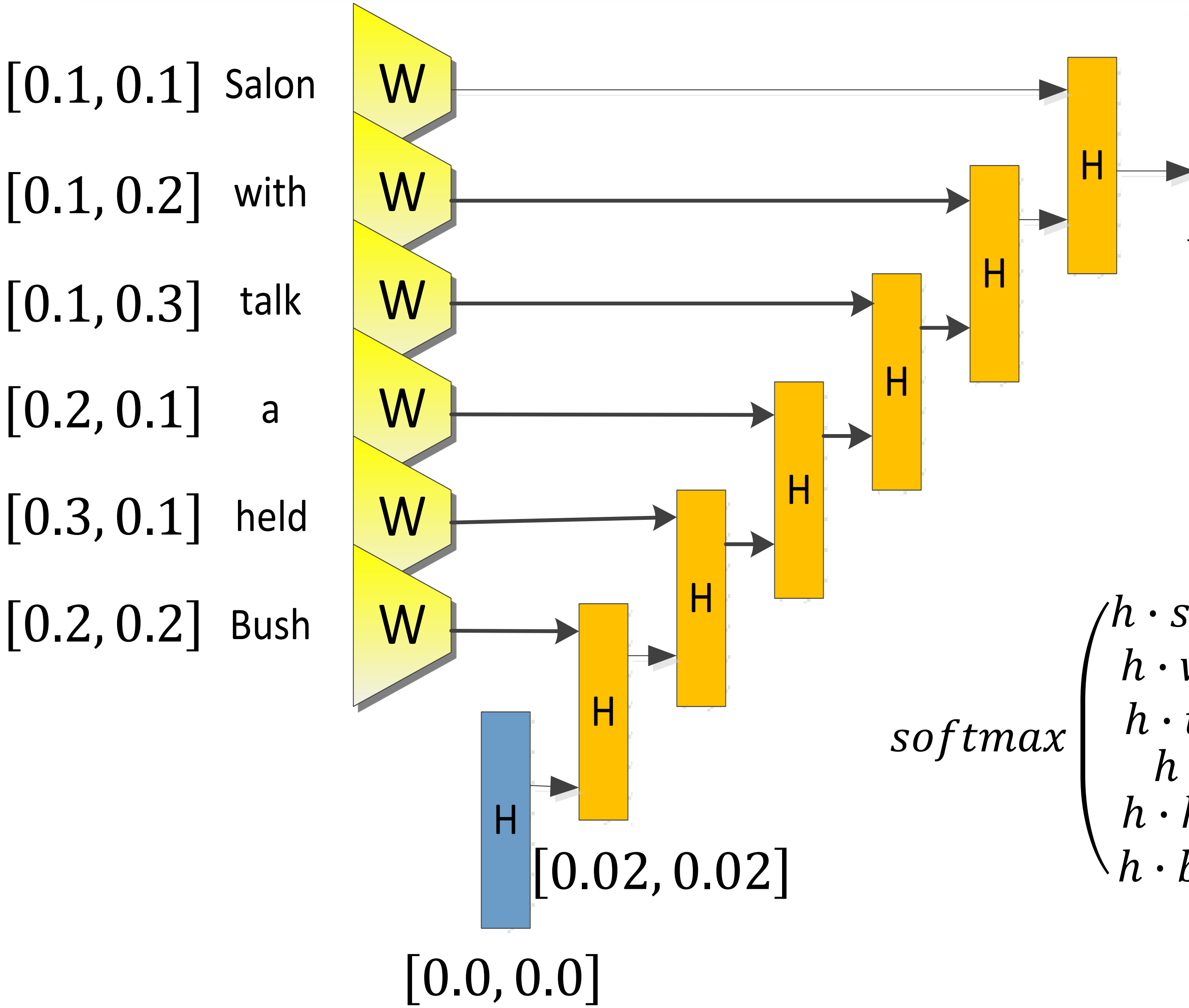


$$U = \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

$$\begin{aligned} h_1 &= U \cdot w_1 + W \cdot h_0 \\ &= U \cdot bush + W \cdot h_0 \\ &= \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0.02 \\ 0.02 \end{bmatrix} \end{aligned}$$

语言模型-循环神经网络

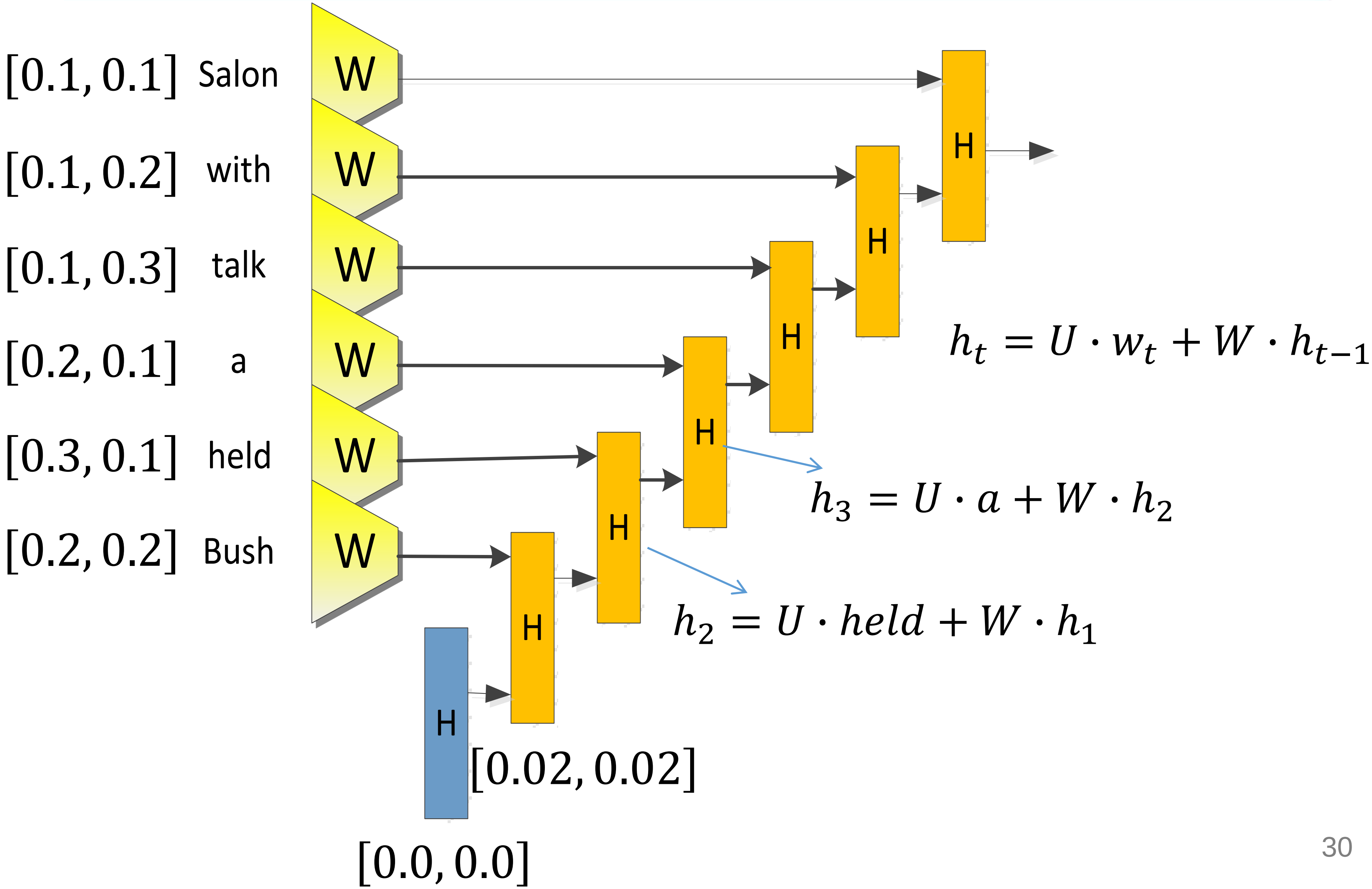


$$U = \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

$$softmax\begin{pmatrix} h \cdot salon \\ h \cdot with \\ h \cdot talk \\ h \cdot a \\ h \cdot held \\ h \cdot bush \end{pmatrix}$$

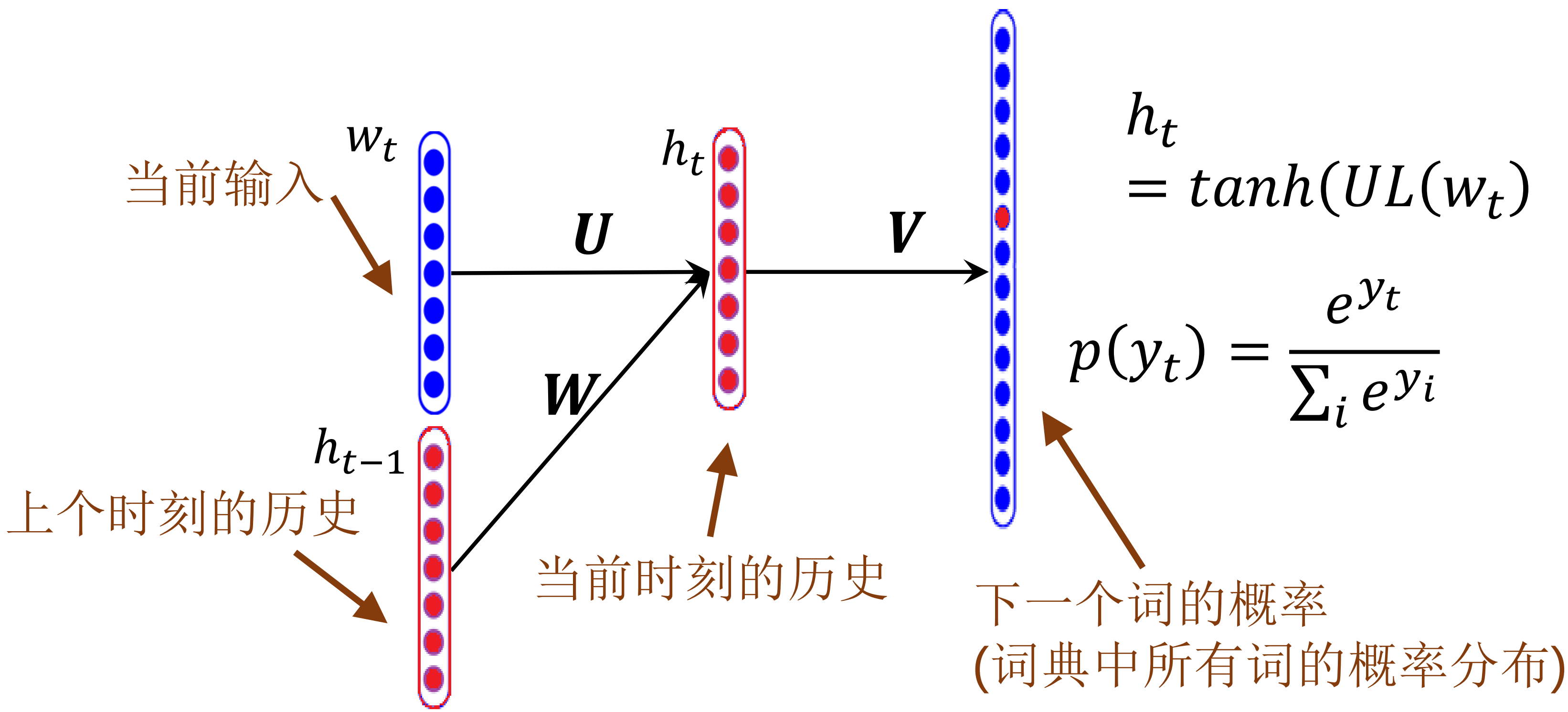
语言模型-循环神经网络



$[0.1, 0.1]$ Salon
$[0.1, 0.2]$ with
$[0.1, 0.3]$ talk
$[0.2, 0.1]$ a
$[0.3, 0.1]$ held
$[0.2, 0.2]$ Bush
W
H
$[0.02, 0.02]$
$[0.0, 0.0]$
$h_t = U \cdot w_t + W \cdot h_{t-1}$
$h_3 = U \cdot a + W \cdot h_2$
$h_2 = U \cdot held + W \cdot h_1$

语言模型-循环神经网络

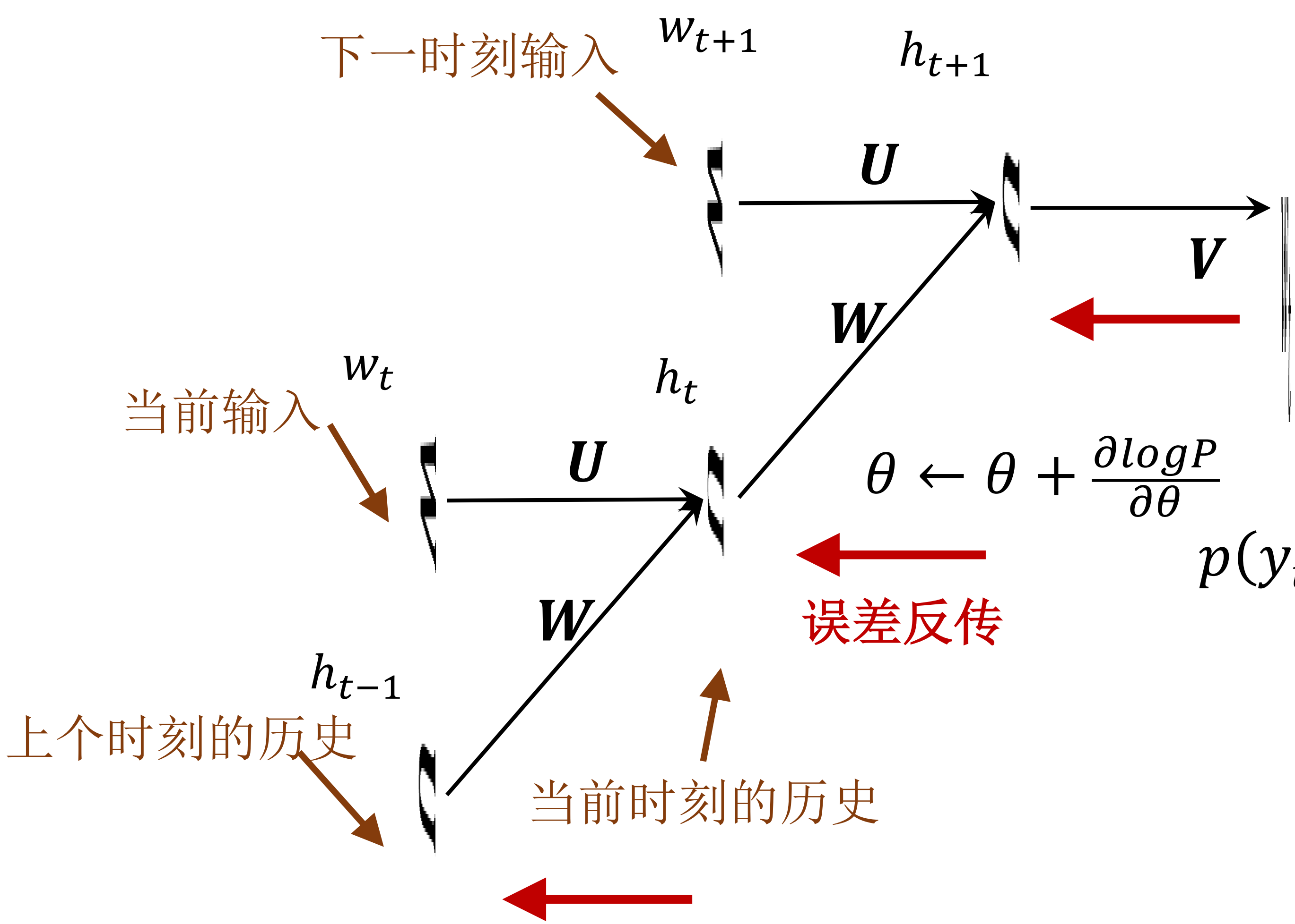
- 输入: $t-1$时刻历史 h_{t-1} 与 t时刻输入 w_t
- 输出: t时刻历史 h_t 与 下个时刻$t+1$输入 的概率



$$h_t = tanh(UL(w_t))$$

$$p(y_t) = \frac{e^{y_t}}{\sum_i e^{y_i}}$$

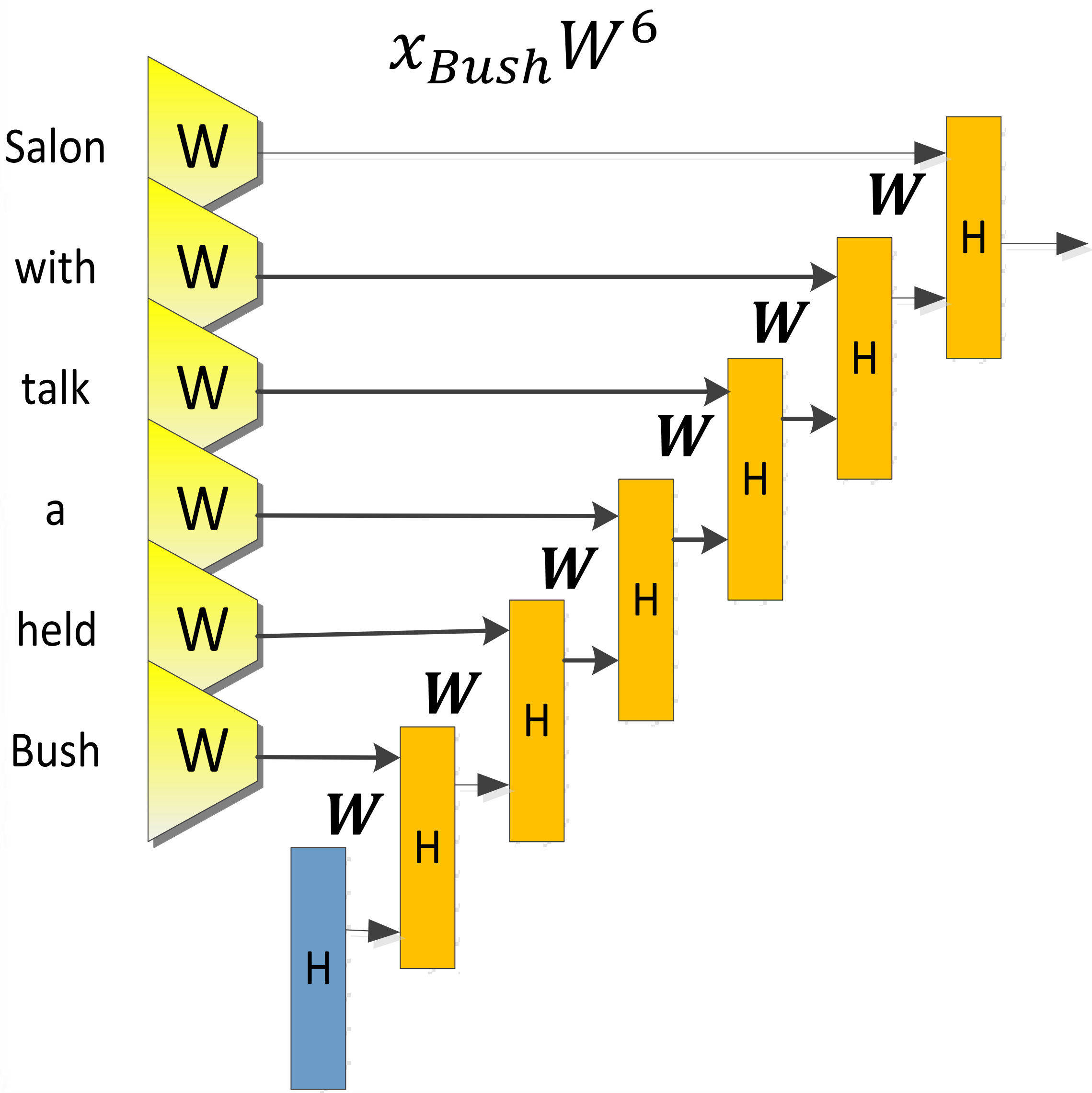
语言模型-循环神经网络



$$\theta \leftarrow \theta + \frac{\partial logP}{\partial \theta}$$

$$p(y_{t+1}) = \frac{e^{y_{t+1}}}{\sum_i e^{y_i}}$$

语言模型-循环神经网络



- **问题**

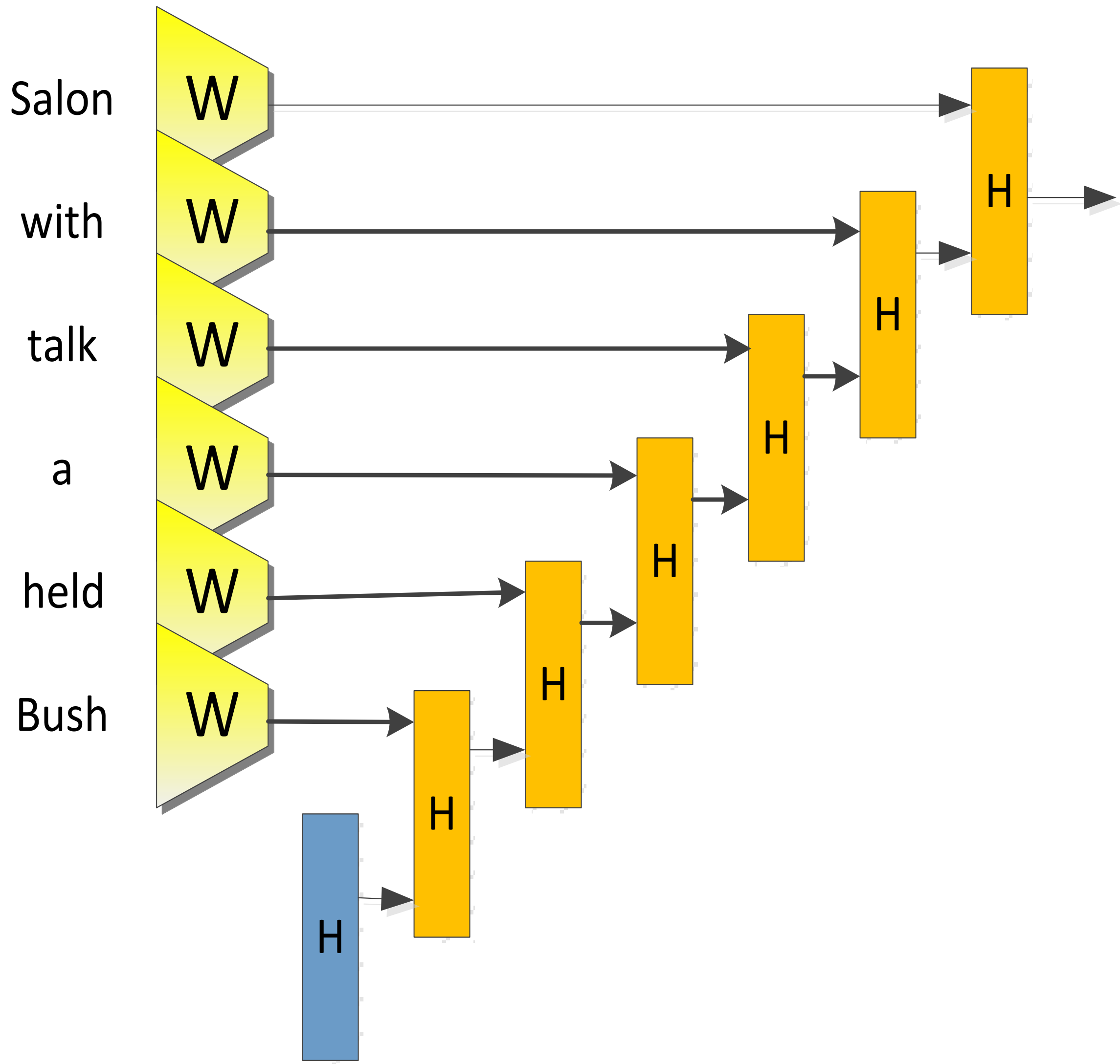
梯度消失和爆炸

参数W经过多次传递后，易发生梯度消失（<1）或爆炸（>1）

有选择地保留和遗忘

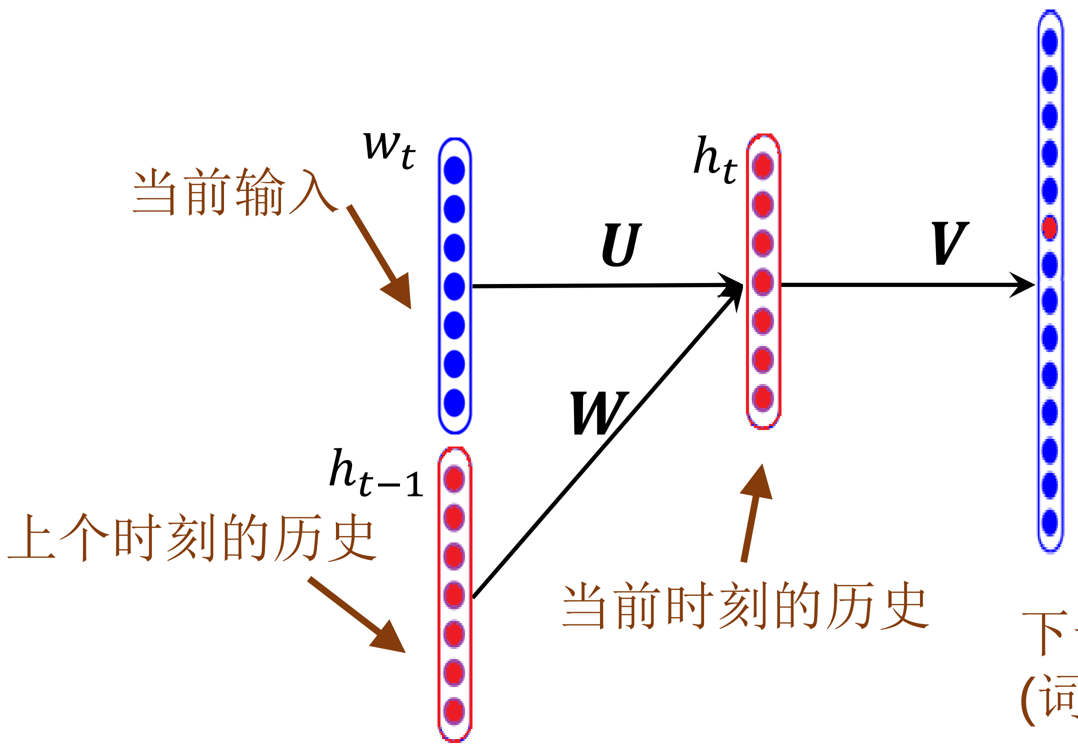
通过某种策略有选择地保留或者遗忘t时刻的信息

语言模型-长短时记忆网络(LSTM)



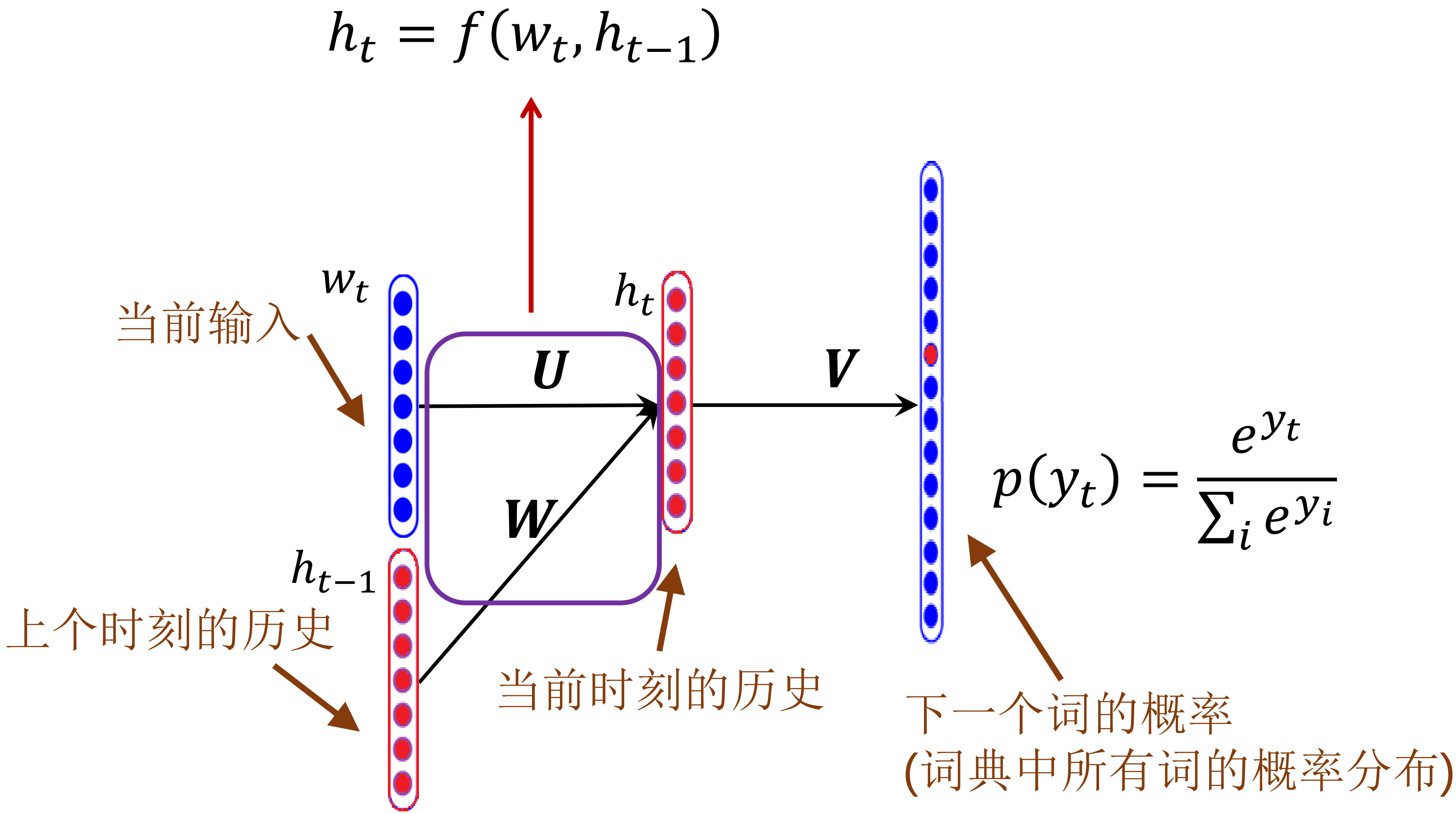
循环神经网络

$$h_t = tanh(UL(w_t) + Wh_{t-1})$$



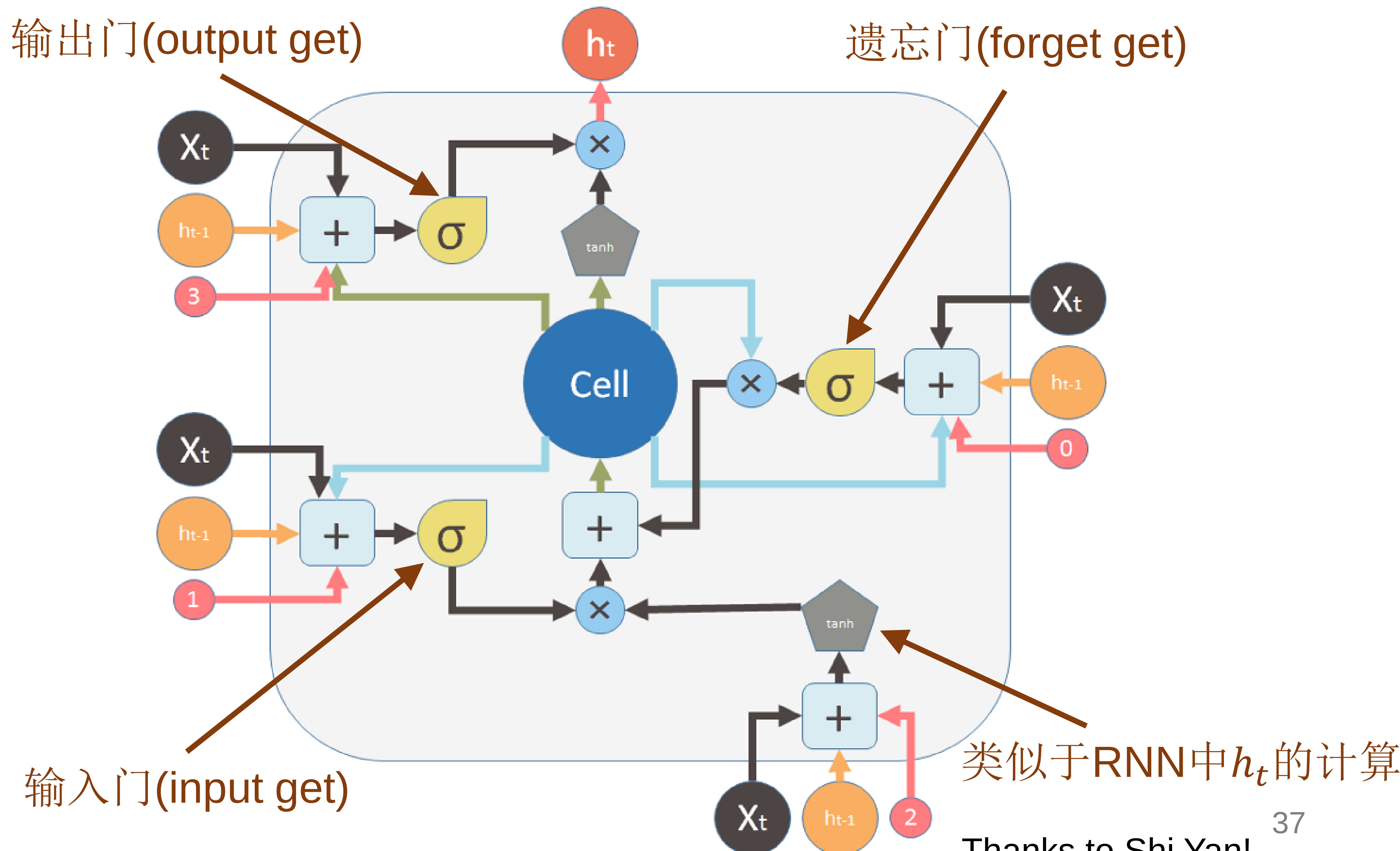
$$p(y_t) = \frac{e^{y_t}}{\sum_i e^{y_i}}$$

LSTM


$h_t = f(w_t, h_{t-1})$
w_t
当前输入
h_t
U
V
W
h_{t-1}
上个时刻的历史
当前时刻的历史
$p(y_t) = \frac{e^{y_t}}{\sum_i e^{y_i}}$
下一个词的概率
(词典中所有词的概率分布)

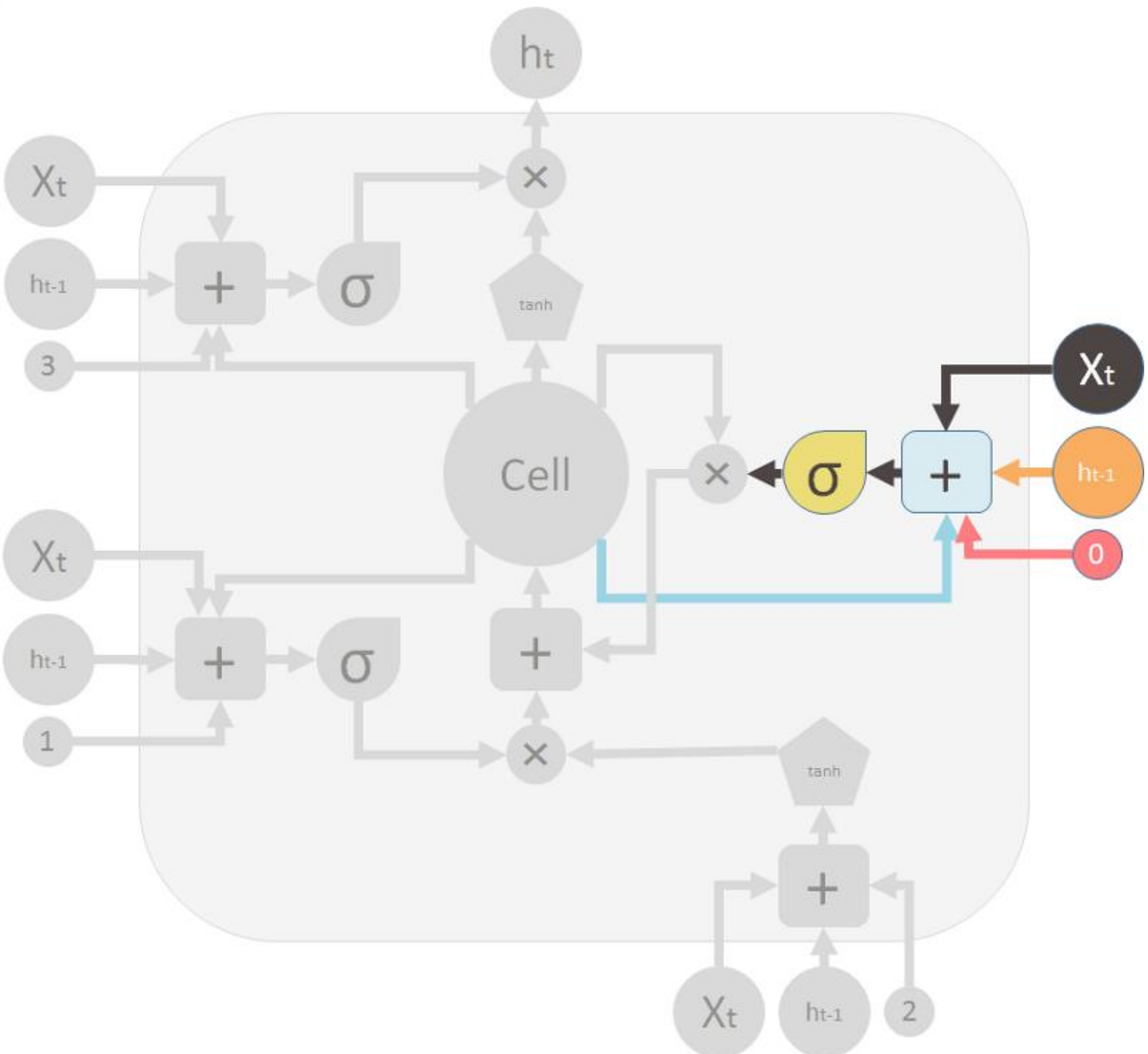
LSTM

$$h_t = f(w_t, h_{t-1})$$



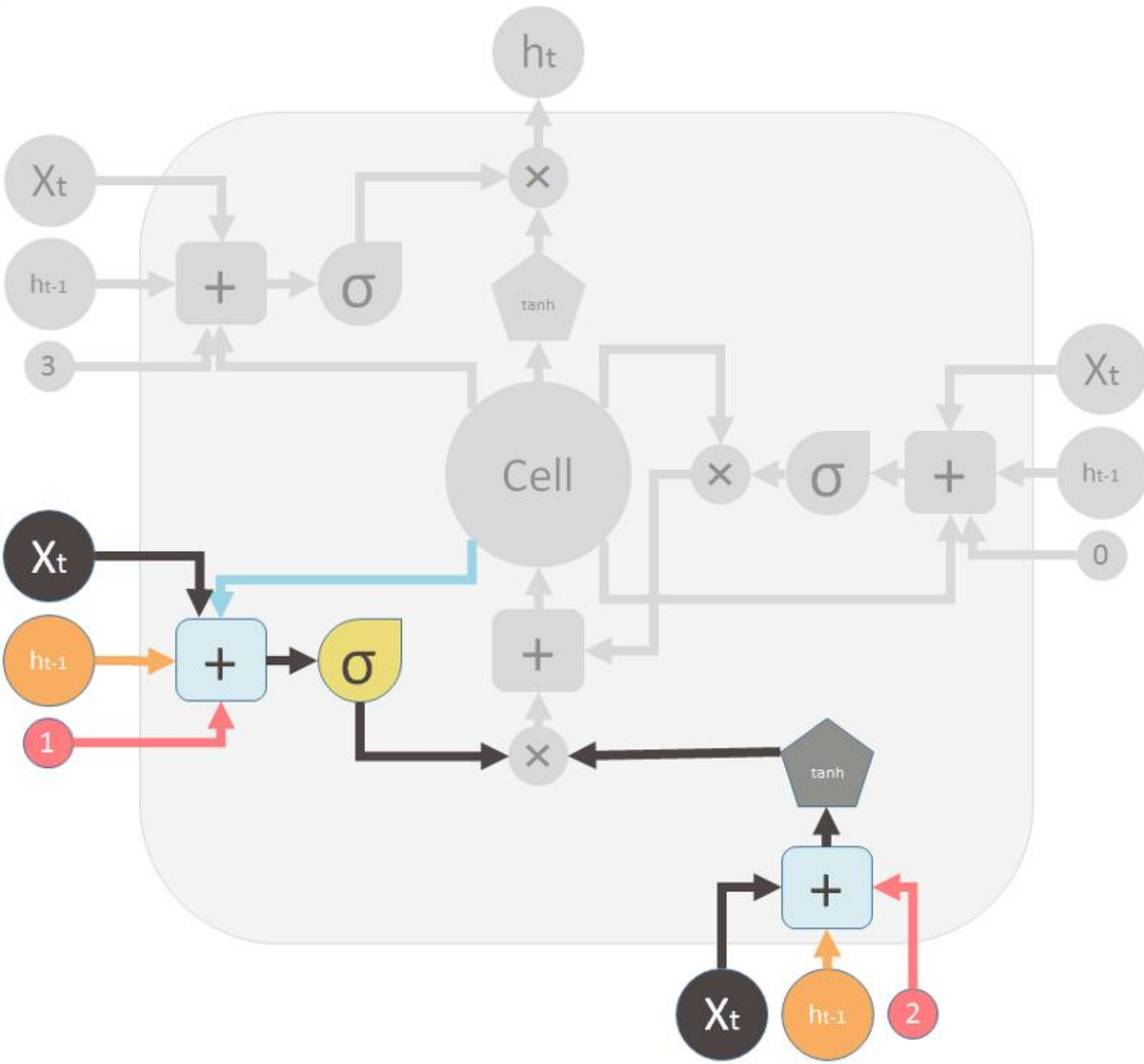
Thanks to Shi Yan!

LSTM



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1}$$

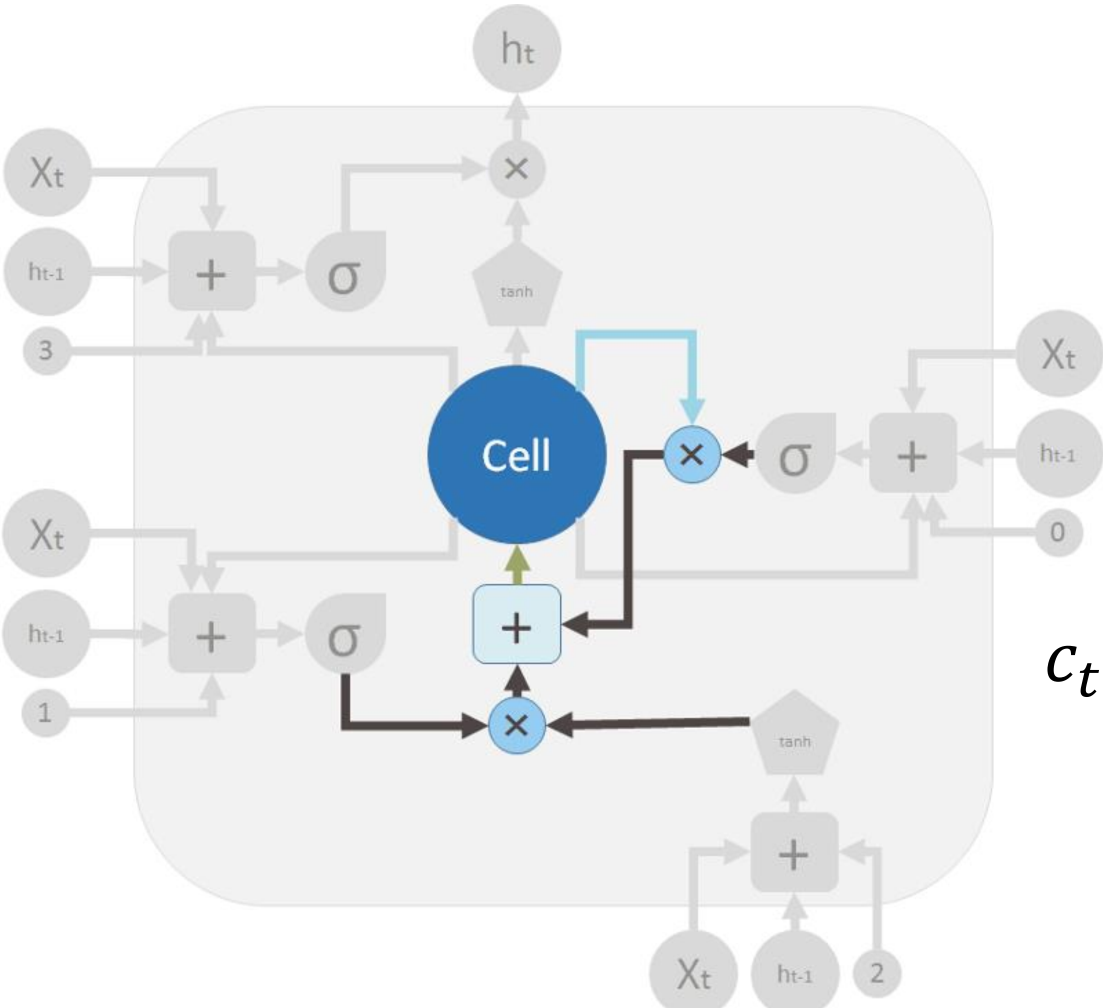
LSTM



$$i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1}$$

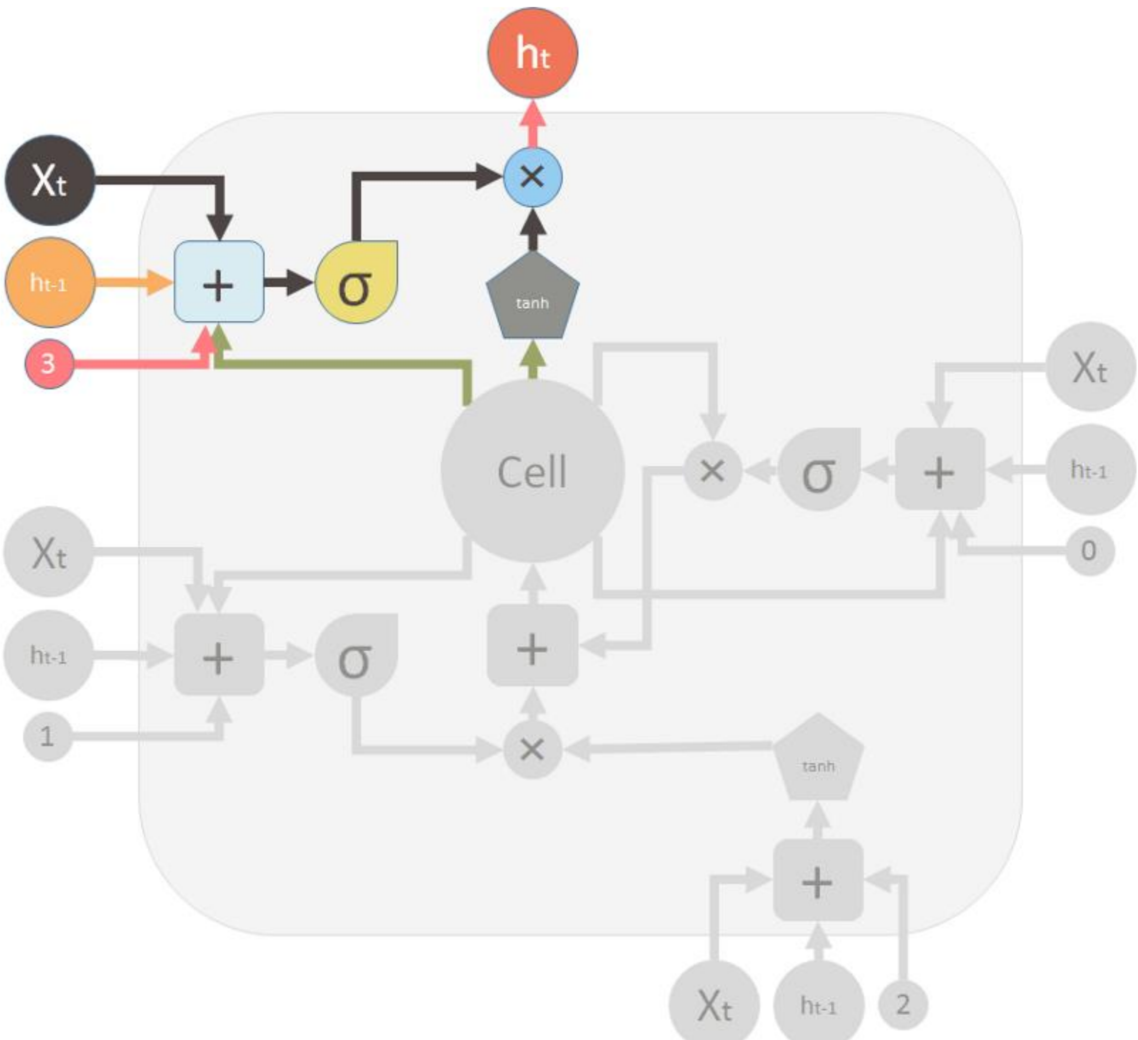
$$\tilde{h}_t = tanh(W_x x_t$$

LSTM



$$c_t = f_t c_{t-1} + i_t \tilde{h}_t$$

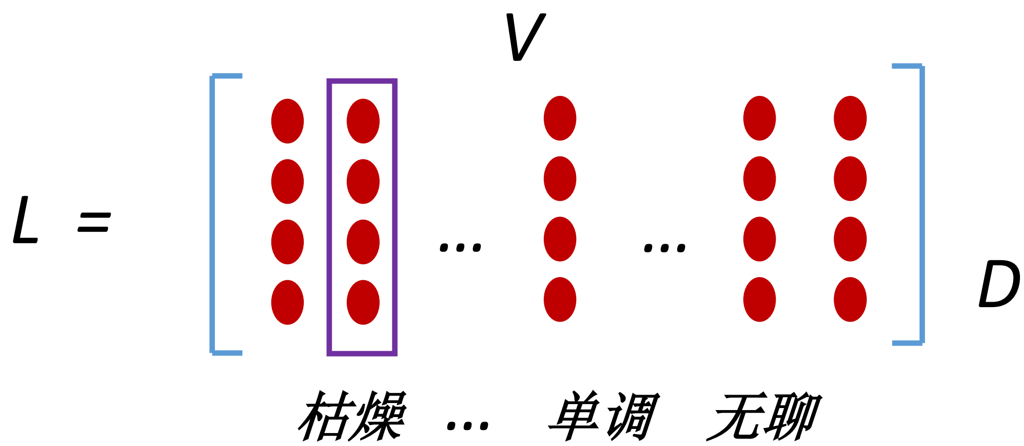
LSTM



$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1}$$

$$h_t = o_t tanh(c_t)$$

词向量规模V

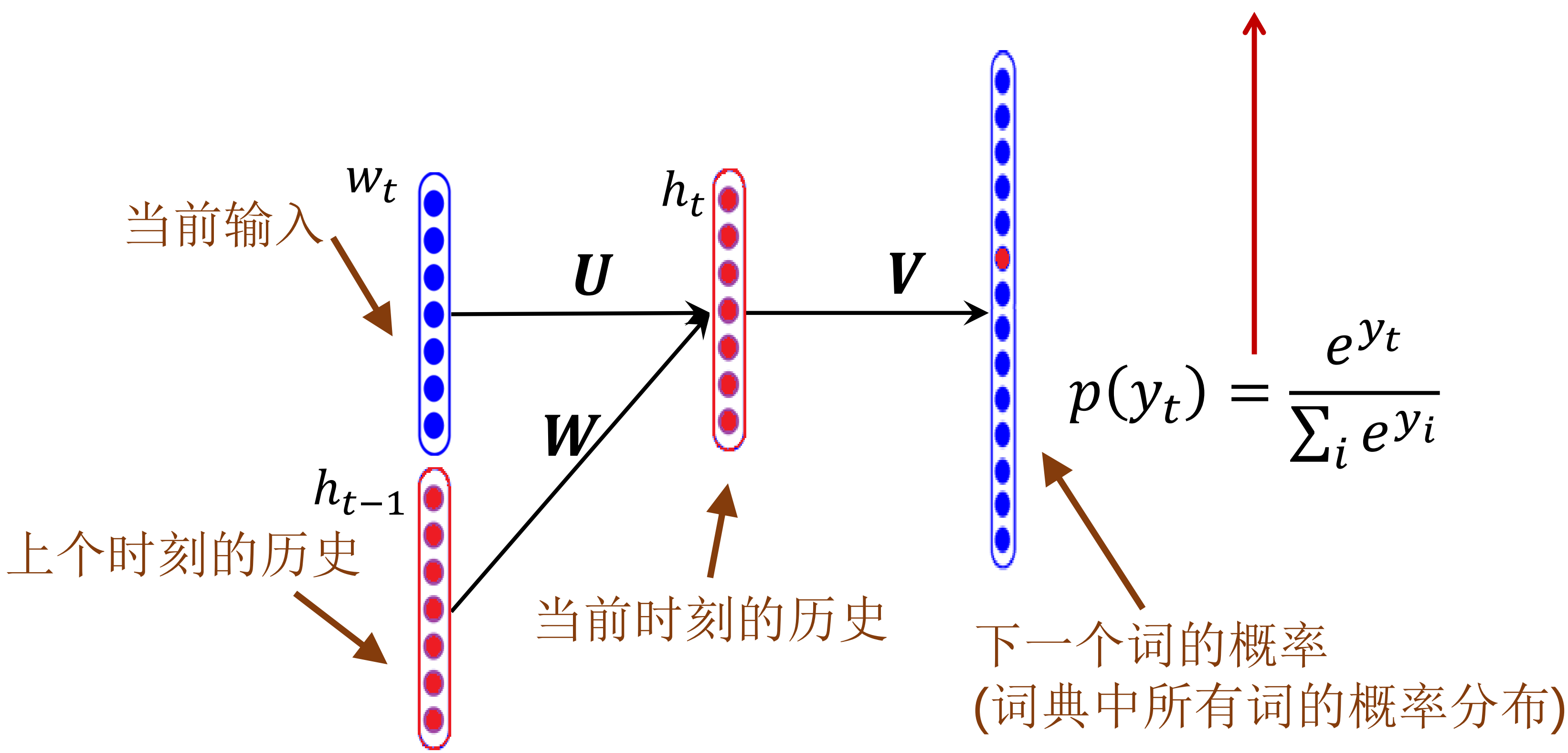


$L \in R^{D \times V}$

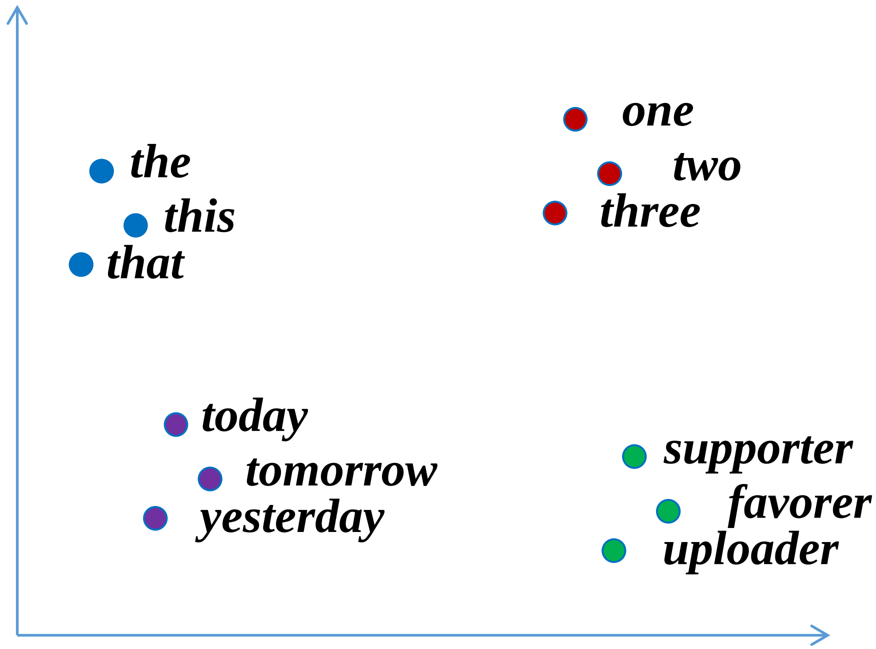
- 词表规模V的确定
  - V的确定：1,训练数据中所有词；2, 频率高于某个阈值的所有词；**3, 前V个频率最高的词**
  - **e.g. V=50000, V=80000**

词向量规模V

决定了模型的复杂度 ← $\sum_i e^{y_i}$遍历词表中的所有词



词向量分布



低维、稠密的实数向量空间

在低维、稠密的实数向量空间中，相似的词聚集在一起，在相同的历史上下文中具有相似的概率分布！

开源工具

1, NNlm, 前馈神经网络语言模型（feed-forward n-gram neural language model），http://nlg.isi.edu/software/nplm/

2, RNNlm, 循环神经网络语言模型（recurrent neural language model），http://rnnlm.org/;

2, LSTMlm, LSTM语言模型（recurrent neural language model with LSTM unit），https://www-i6.informatik.rwth-aachen.de/web/Software/rwthlm.php

3, LSTM反向传播算法，http://arunmallya.github.io/writeups/nn/lstm/index.html#/

4, Google Word2Vec, http://code.google.com/p/word2vec/

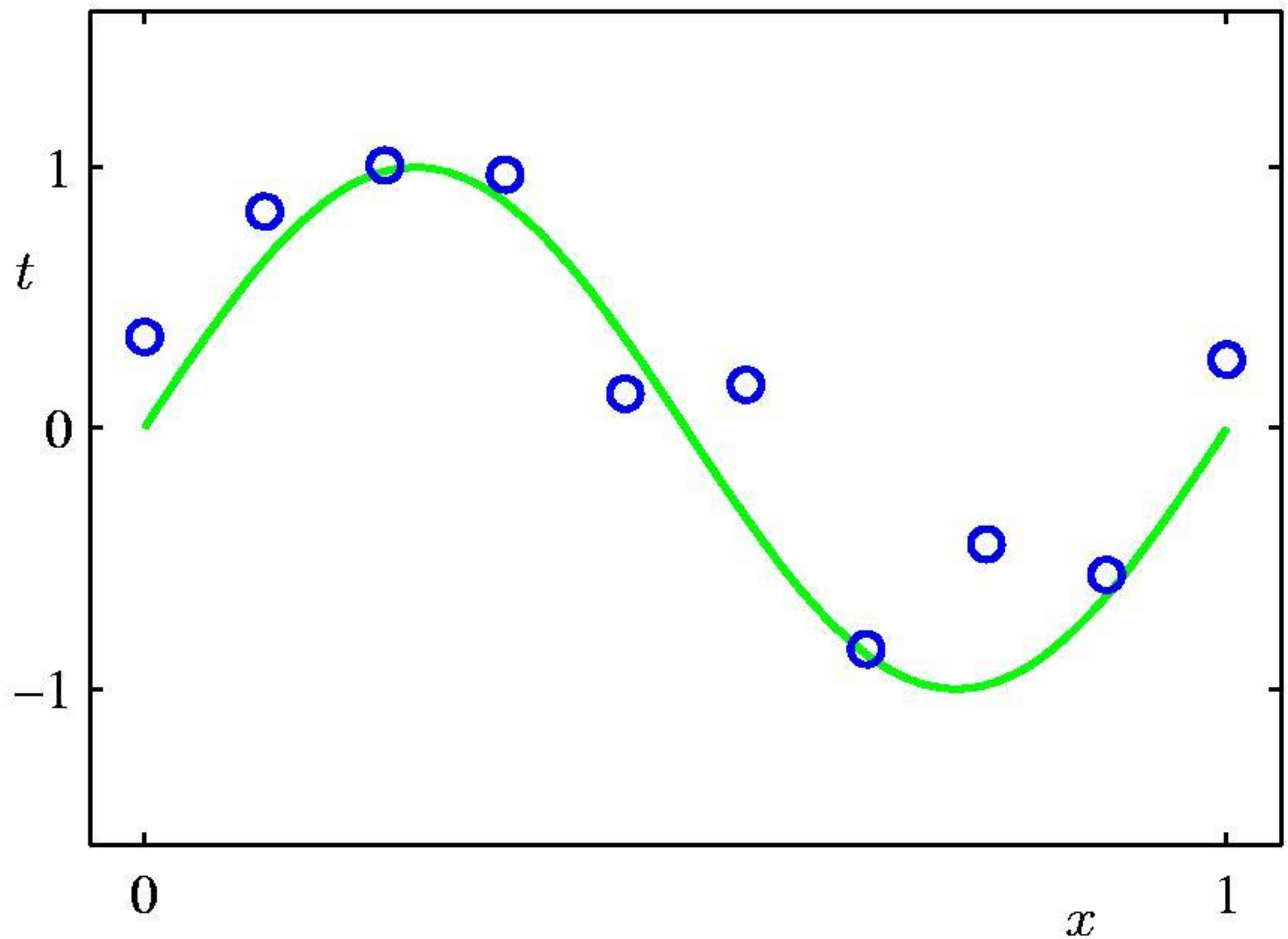
… …

谢谢!

Q&A

参数学习：反向传播算法

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$$



参数学习：反向传播算法

$$\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$$

$$h_{W,b}(x)=f(W^T x+b)$$

如何学习每一层的参数 W, b?

定义损失函数

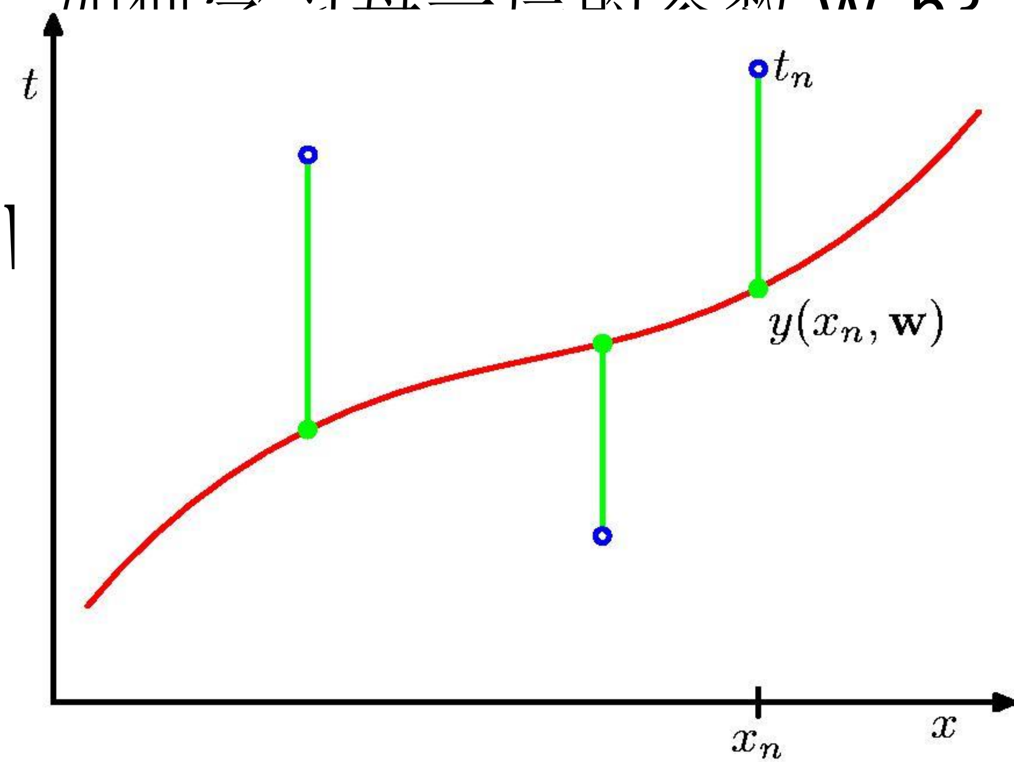
$$J(W,b;x,y)=\frac{1}{2}\left\|h_{W,b}(x)-y\right\|^2$$

参数学习：反向传播算法

$$\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$$

$$h_{W,b}(x)=f(W^Tx+b)$$

如何学习每一层的参数W, b?



参数学习：反向传播算法

$$\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$$

$$h_{W,b}(x)=f(W^Tx+b)$$

如何学习每一层的参数 W, b?

定义损失函数

$$J(W,b;x,y)=\frac{1}{2}\left\|h_{W,b}(x)-y\right\|^2$$

$$J(W,b)=\left[\frac{1}{m}\sum_{k=1}^{m}J(W,b;x^{(k)},y^{(k)})\right]$$

$$+\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{j=1}^{s_l}\sum_{i=1}^{s_{l+1}}\left({W_{ij}^{(l)}}^2\right)$$

参数学习：反向传播算法

$$W, b = argmin\, J(W, b)$$

Stochastic Gradient Descent Algorithm (随机梯度下降)

$$W_{ij}^{(l)} := W_{ij}^{(l)} - \alpha \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)}$$

$$b_{i}^{(l)} := b_{i}^{(l)} - \alpha \boxed{\frac{\partial}{\partial b_{i}^{(l)}} J(W,b)}$$

参数学习：反向传播算法

$$J(W,b) = \left[\frac{1}{m}\sum_{k=1}^{m} J\left(W,b;x^{(k)},y^{(k)}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{j=1}^{s_l}\sum_{i=1}^{s_{l+1}}\left({W_{ij}^{(l)}}^2\right)$$

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \frac{1}{m}\sum_{k=1}^{m} \boxed{\frac{\partial}{\partial W_{ij}^{(l)}} J\left(W,b;x^{(k)},y^{(k)}\right)} + \lambda W_{ij}^{(l)}$$

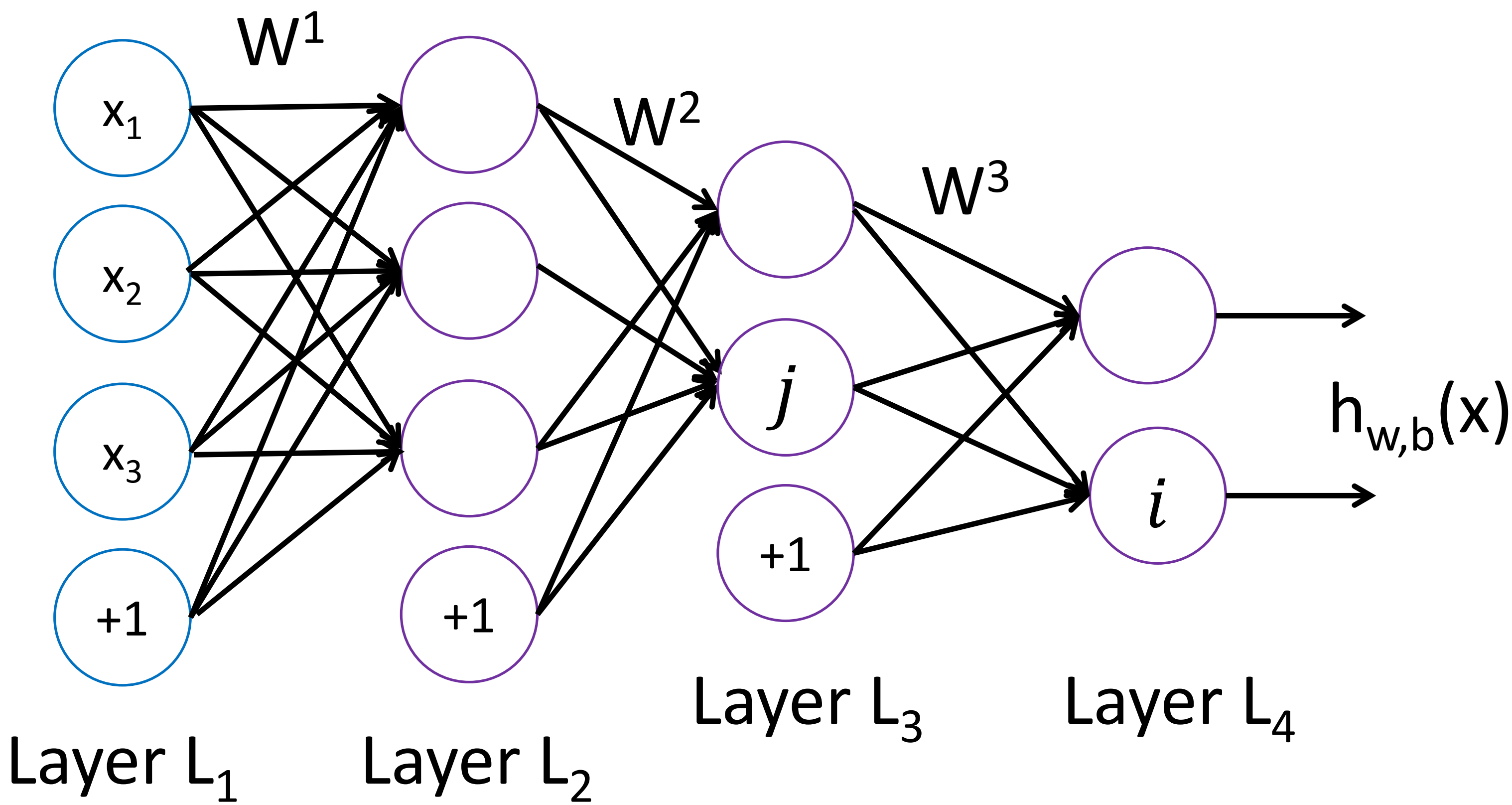
$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \frac{1}{m}\sum_{k=1}^{m} \boxed{\frac{\partial}{\partial b_{i}^{(l)}} J\left(W,b;x^{(k)},y^{(k)}\right)}$$

参数学习：反向传播算法

$$\frac{\partial}{\partial W_{ij}^{(l)}} J\left(W, b; x^{(k)}, y^{(k)}\right)$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J\left(W, b; x^{(k)}, y^{(k)}\right)$$

参数学习：反向传播算法



$$\frac{\partial}{\partial W_{ij}^{(3)}} J(W,b;x,y) \Rightarrow \frac{\partial}{\partial W_{ij}^{(2)}} J(W,b;x,y) \Rightarrow$$

参数学习：反向传播算法

$$\frac{\partial}{\partial W_{ij}^{(3)}} J(W, b; x, y) = \frac{\partial}{\partial W_{ij}^{(3)}} \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2$$

$$= \frac{\partial}{\partial W_{ij}^{(3)}} \frac{1}{2} \left\| f\left(z^{(4)}\right) - y \right\|^2 = -\left(y_i - f\left(z_i^{(4)}\right)\right) \frac{\partial}{\partial W_{ij}^{(3)}} f\left(z_i^{(4)}\right)$$

$$= \boxed{-\left(y_i - f\left(z_i^{(4)}\right)\right) f'\left(z_i^{(4)}\right)} f\left(z_j^{(3)}\right)$$

$$z_i^{(4)} = W_{i\cdot}^{(3)} f\left(z^{(3)}\right) + b^{(3)} = \sum_k W_{ik}^{(3)} f\left(z_k^{(3)}\right) + b^{(3)}$$

参数学习：反向传播算法

$$\frac{\partial}{\partial W_{ij}^{(2)}} J(W,b;x,y) = \frac{\partial}{\partial W_{ij}^{(2)}} \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2 \qquad z_k^{(3)} = W_{k\cdot}^{(2)} f\left(z^{(2)}\right) + b^{(2)}$$

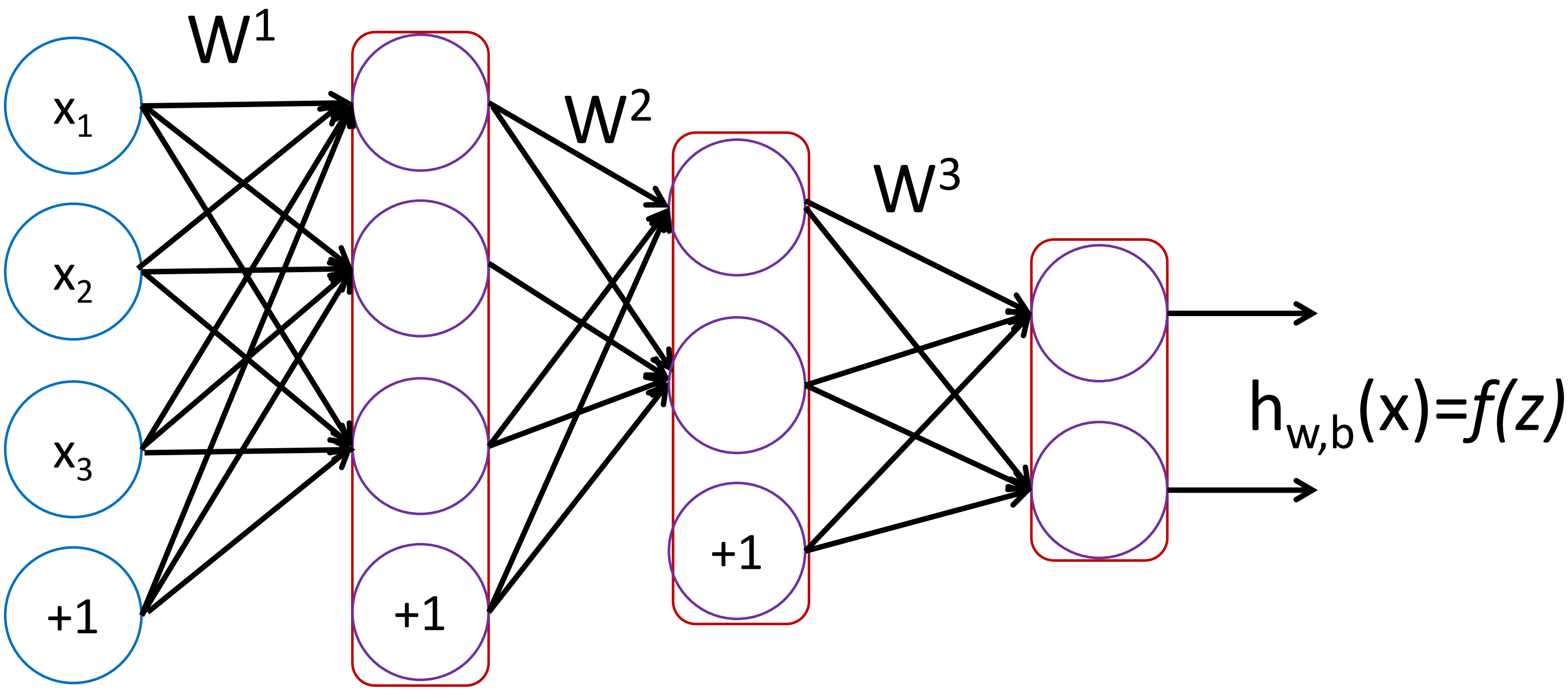
$$= \frac{\partial}{\partial W_{ij}^{(2)}} \frac{1}{2}\left\| f\left(z^{(4)}\right) - y \right\|^2 = -\left(y - f\left(z^{(4)}\right)\right) \frac{\partial}{\partial W_{ij}^{(2)}} f\left(z^{(4)}\right)$$

$$= -\left(y - f\left(z^{(4)}\right)\right) f'\left(z^{(4)}\right) \sum_k \frac{\partial f\left(z^{(4)}\right)}{\partial f\left(z_k^{(3)}\right)} \frac{\partial f\left(z_k^{(3)}\right)}{\partial W_{ij}^{(2)}}$$

$$= \left(-\left(y - f\left(z^{(4)}\right)\right) f'\left(z^{(4)}\right)\right)^T W_{\cdot i}^{(3)} f'\left(z_i^{(3)}\right) f\left(z_j^{(2)}\right)$$

$$= \boxed{\left(\sum_k \boxed{-\left(y_k - f\left(z_k^{(4)}\right)\right) f'\left(z_k^{(4)}\right)} W_{ki}^{(3)}\right) f'\left(z_i^{(3)}\right)} f\left(z_j^{(2)}\right)$$

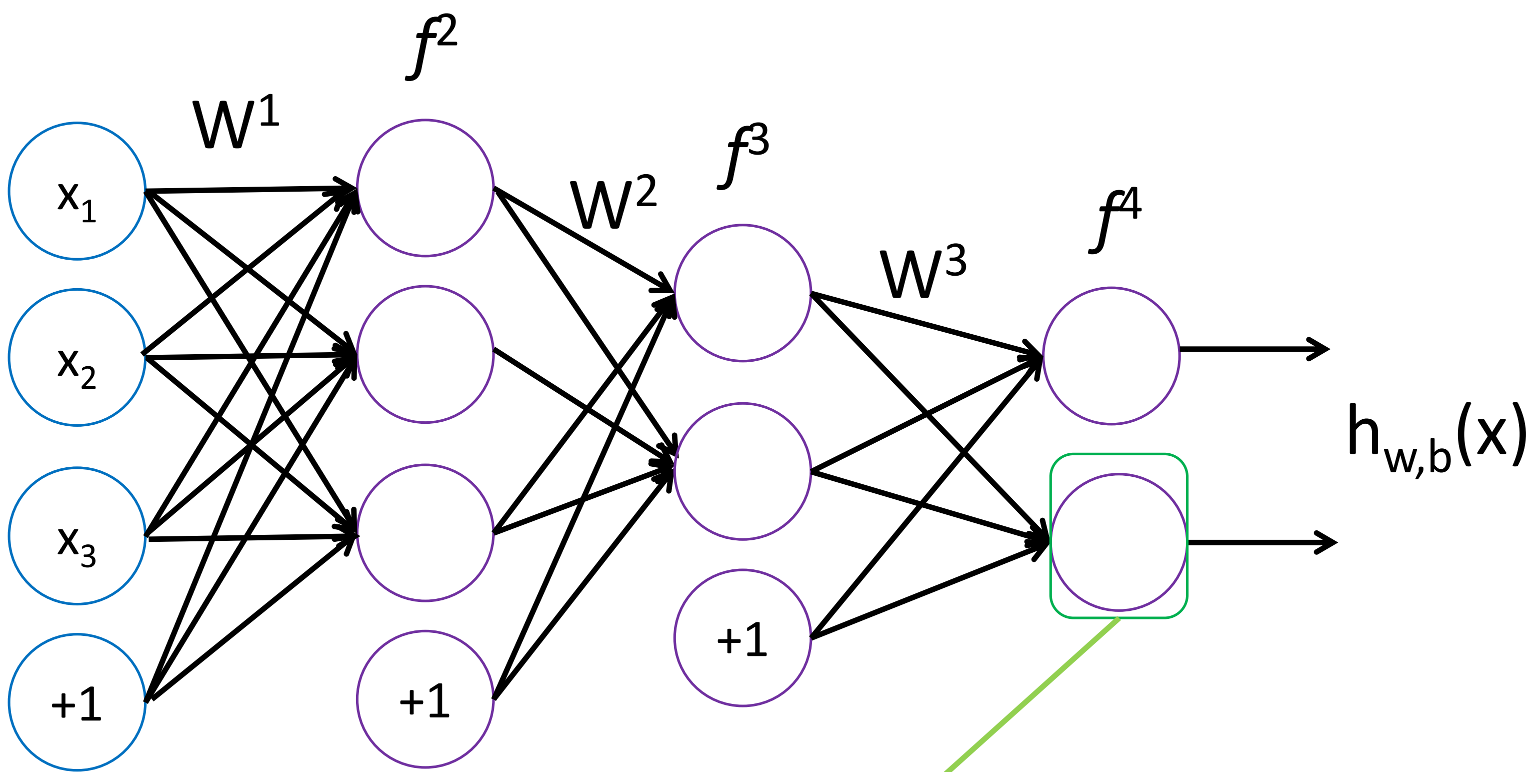
参数学习：反向传播算法



1, Forward

计算每个神经元的输出

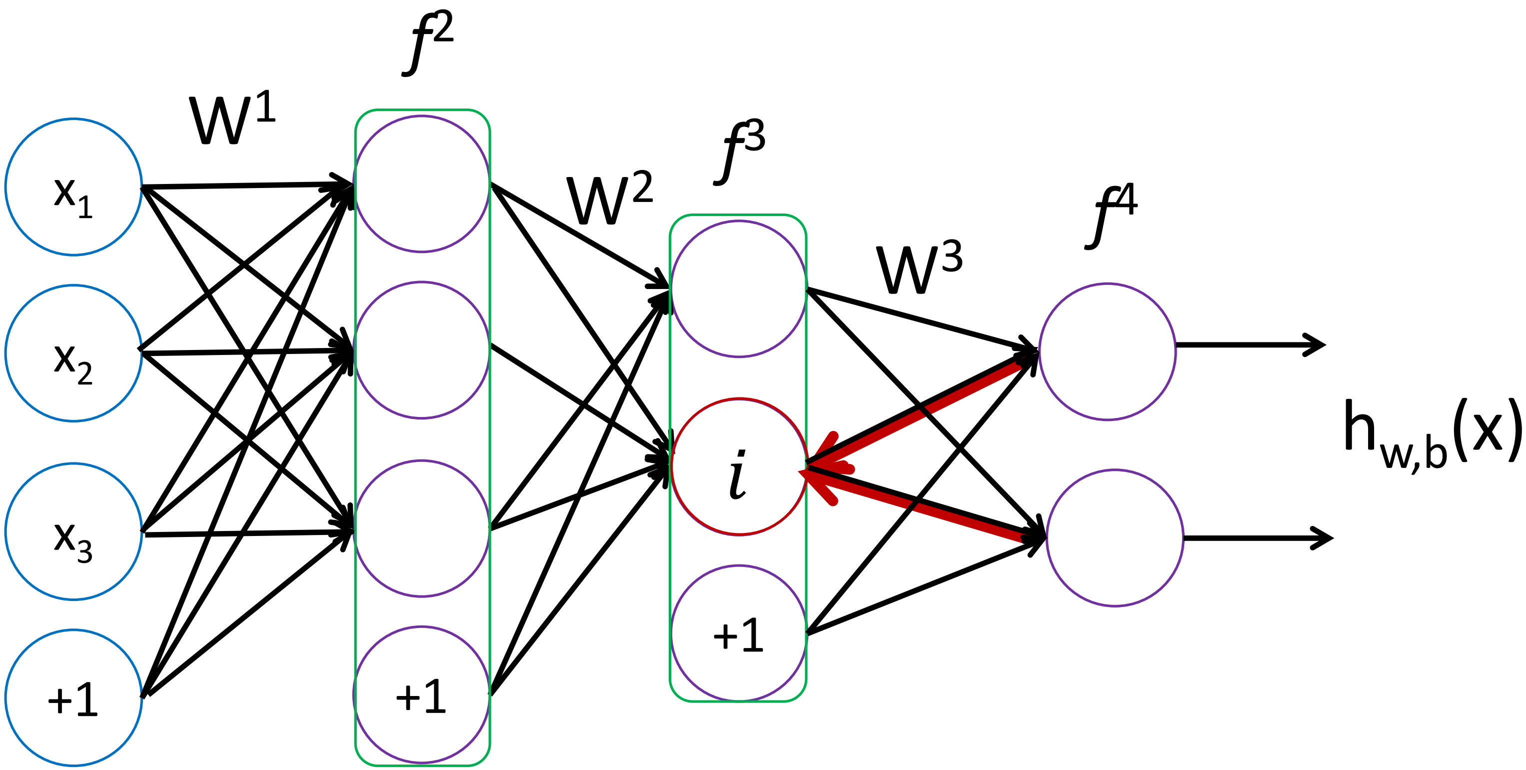
参数学习：反向传播算法



2, 计算输出层每个神经元的梯度

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2 = -\left(y_i - f\left(z_i^{(n_l)}\right)\right) \cdot f'\left(z_i^{(n_l)}\right)$$

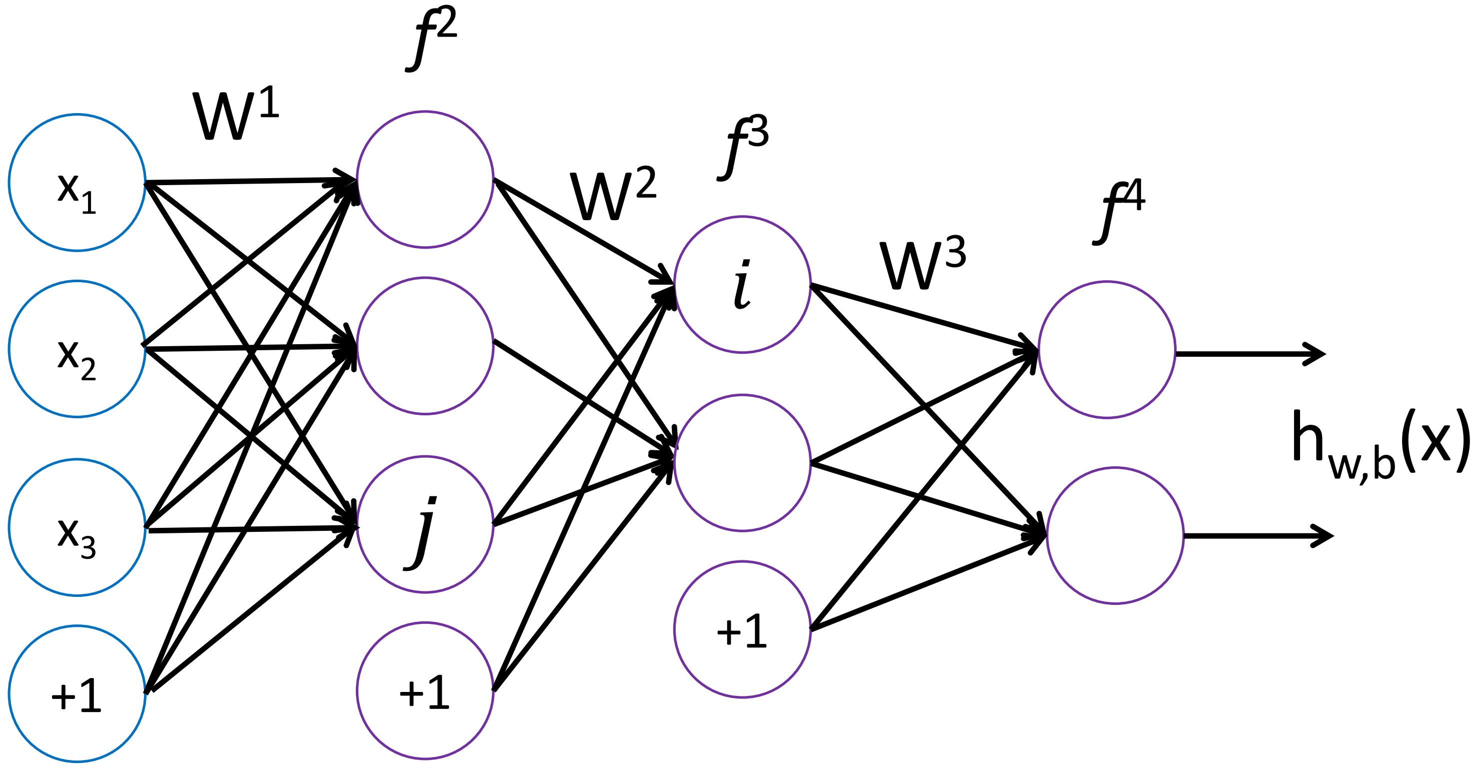
参数学习：反向传播算法



3, 反向计算每个神经元的梯度 $\delta_i^{(l)}$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_i^{(l+1)} \right) \cdot f'\left(z_i^{(l)}\right)$$

参数学习：反向传播算法



4, 计算连接权重和偏差的梯度

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = f\left(z_j^{(l)}\right) \cdot \delta_i^{(l+1)} \qquad \frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}$$