四川大學
SICHUAN UNIVERSITY

机器学习-第十一章 回复式神经网络和自注意力机制

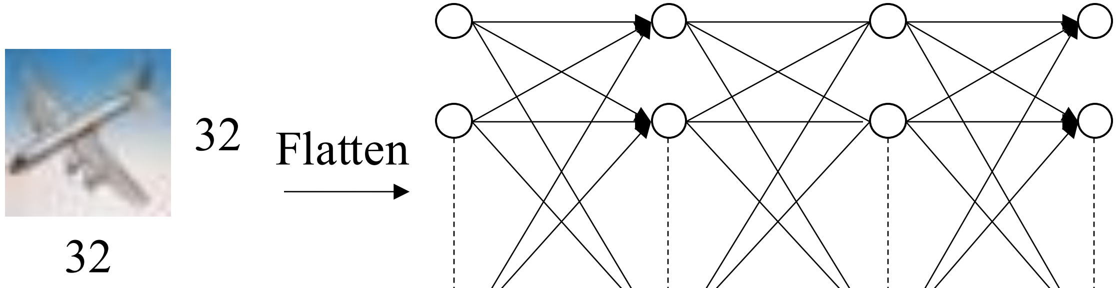
教师：胡俊杰 副教授

邮箱：hujunjie@scu.edu.cn

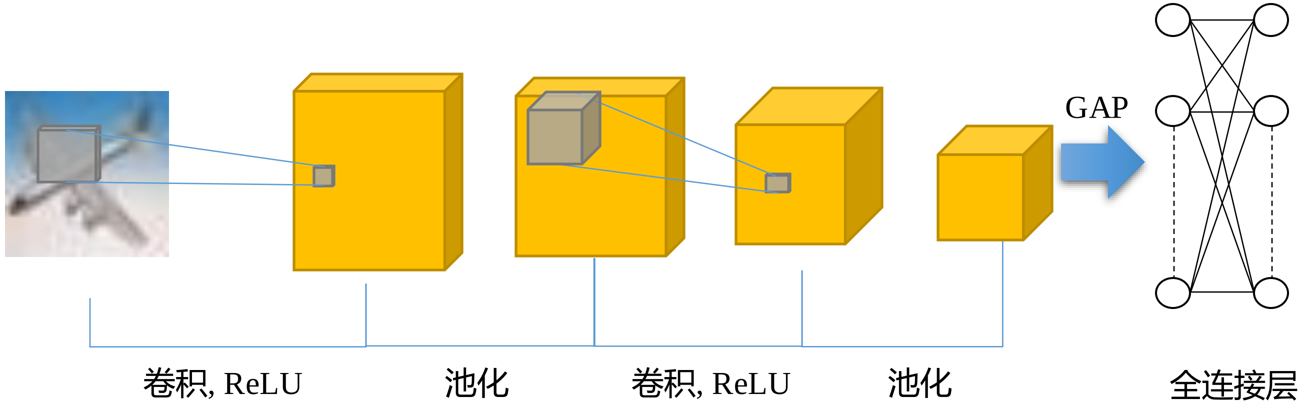
大纲

- 序列预测问题
- 回复式神经网络
- 门控机制
- 自注意力机制

序列预测问题



$$a^{l+1} = \phi(W^l a^l)$$



$$a_{n,m}^{l+1} = \phi\left(\sum_{i=1}^{I}\sum_{j=1}^{J} a_{n+j-1,m+i-1}^{l} \cdot W_{j,i}^{l}\right)$$

序列预测问题

输出：长度为m的英文序列

I love machine learning

深度神经网络模型

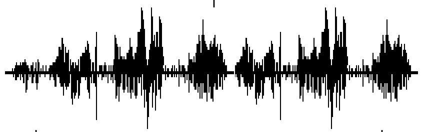
我爱机器学习

输入：长度为n的中文序列

输出：情绪类别

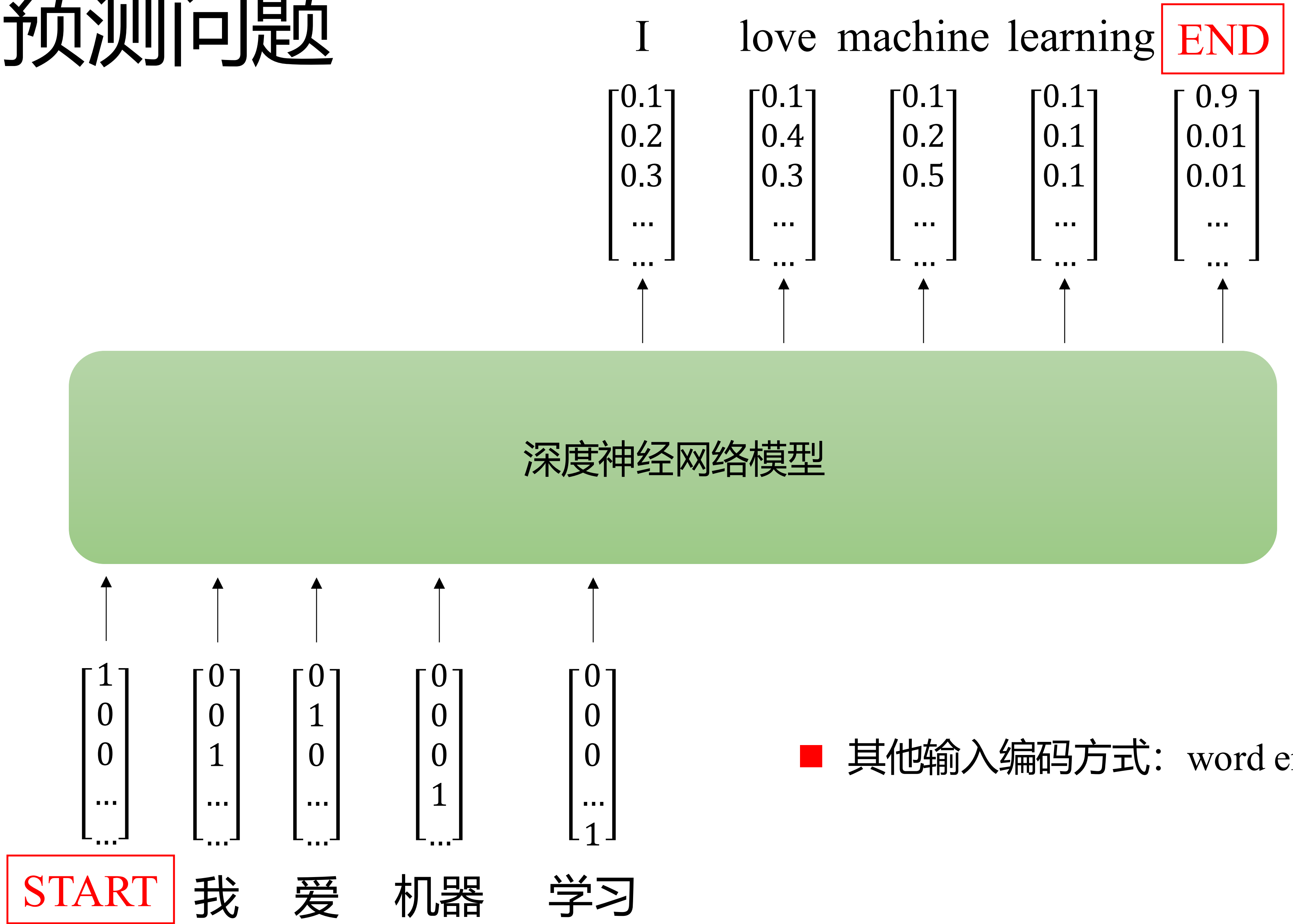
悲伤

深度神经网络模型



输入：长度为n的音频序列

序列预测问题

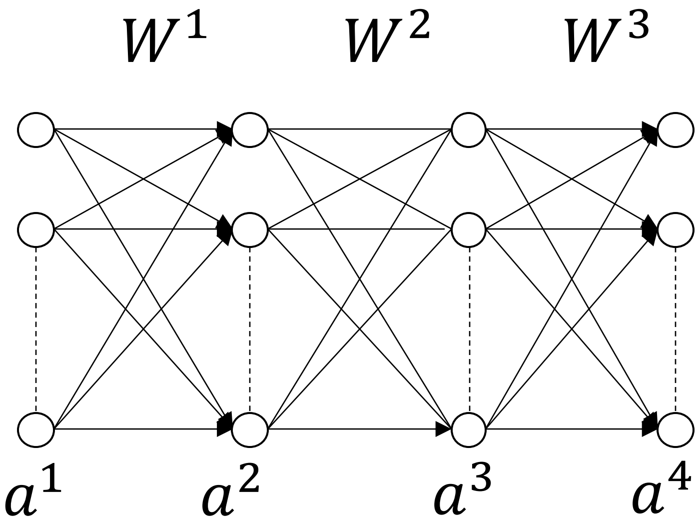


- 其他输入编码方式：word embedding

大纲

- 序列预测问题
- 回复式神经网络
- 门控机制
- 自注意力机制

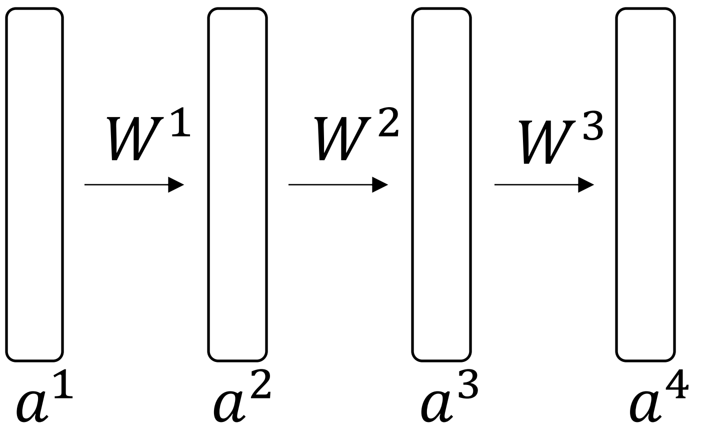
回复式神经网络



全连接神经网络

$$a^{l+1} = \phi(W^l a^l)$$

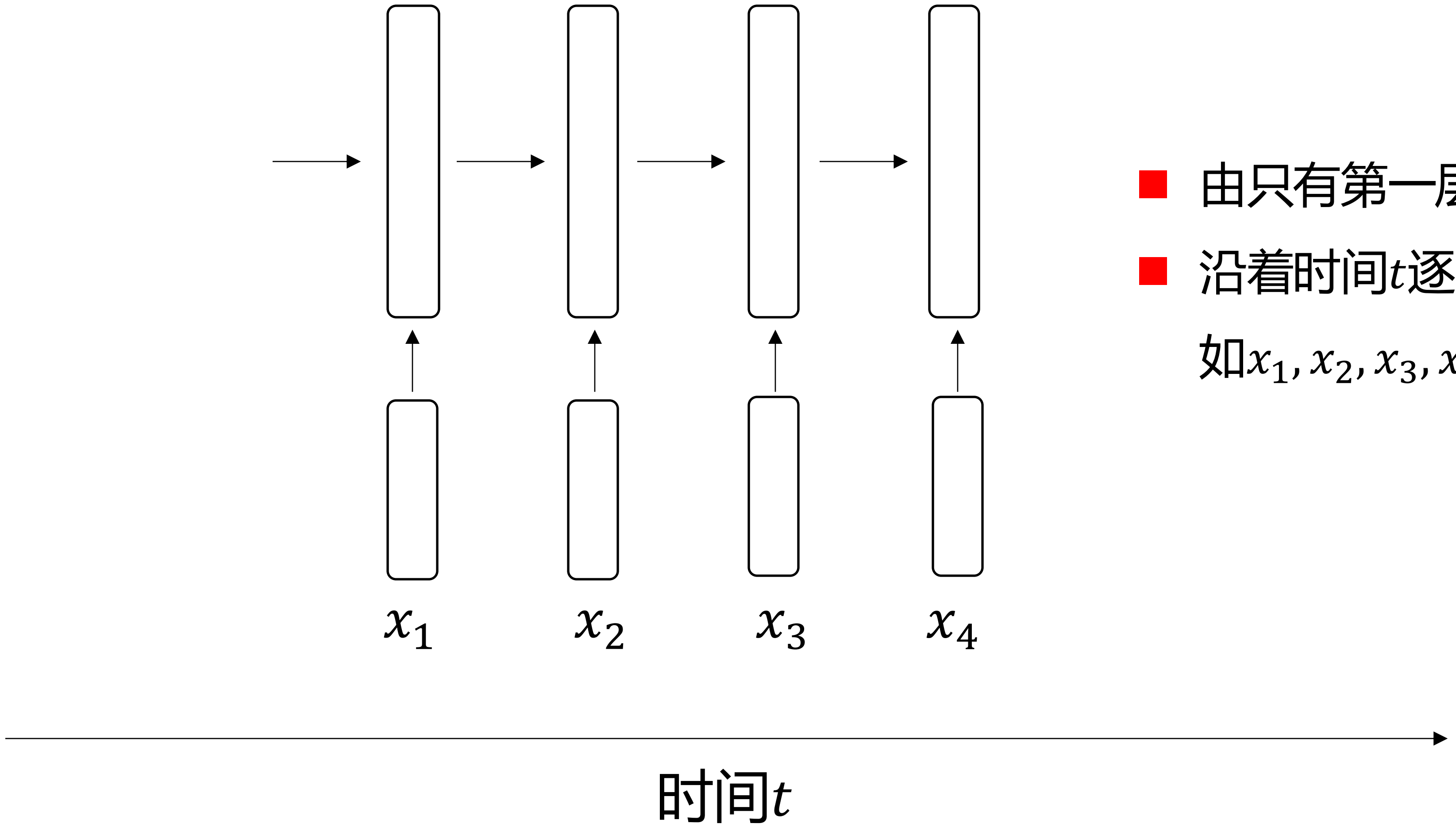
简化



全连接神经网络

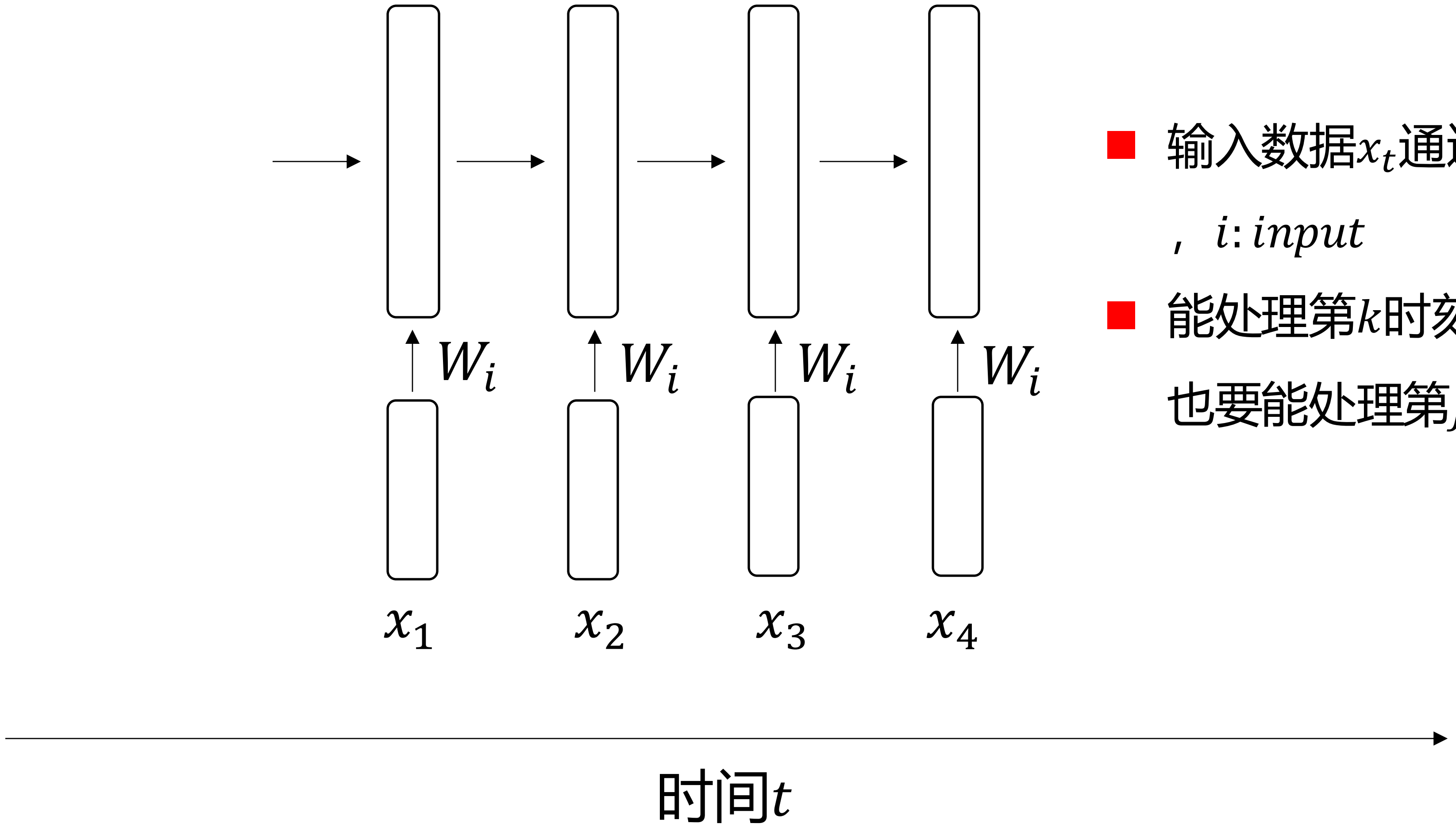
$$a^{l+1} = \phi(W^l a^l)$$

回复式神经网络



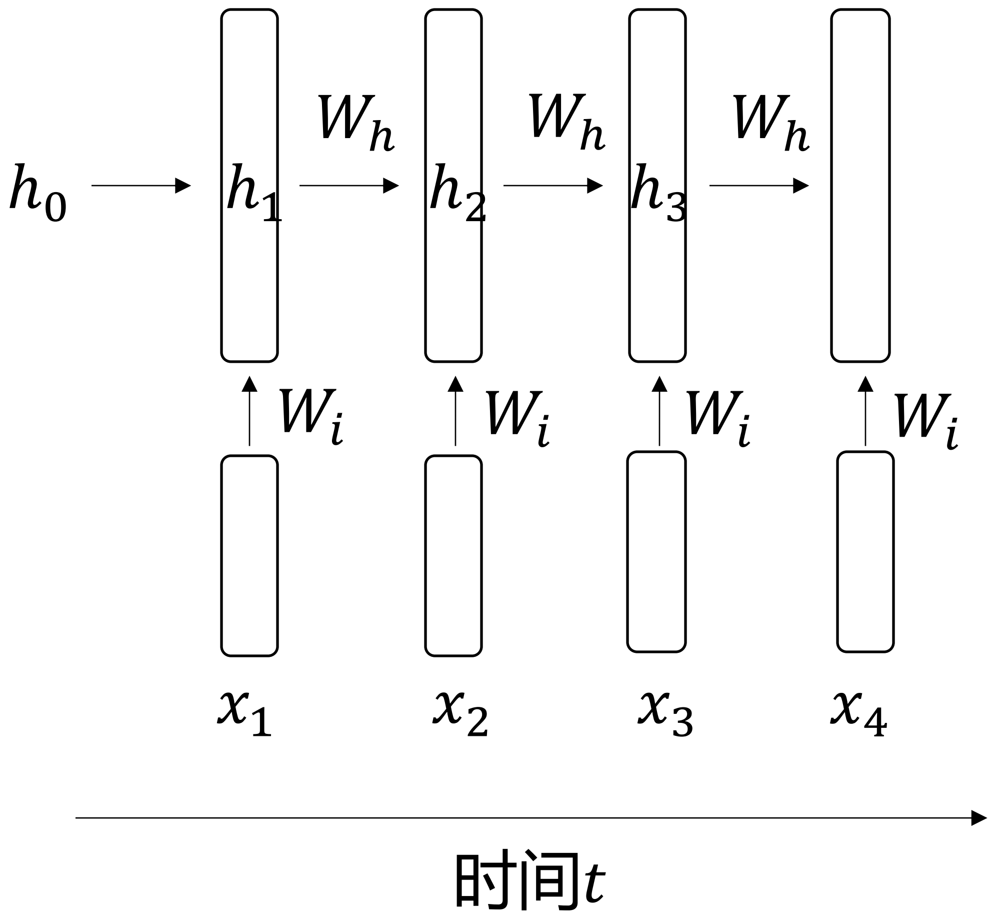
- 由只有第一层有输入变换为每层均有输入
- 沿着时间t逐个地向网络输入x_t， $x_t \in \mathbb{R}^d$，如x_1, x_2, x_3, x_4

回复式神经网络



- 输入数据x_t通过连接权矩阵W_i进入神经网络，$i: input$
- 能处理第k时刻输入x_k的连接权矩阵，应该也要能处理第j时刻的输入x_j：各时刻共享W_i

回复式神经网络



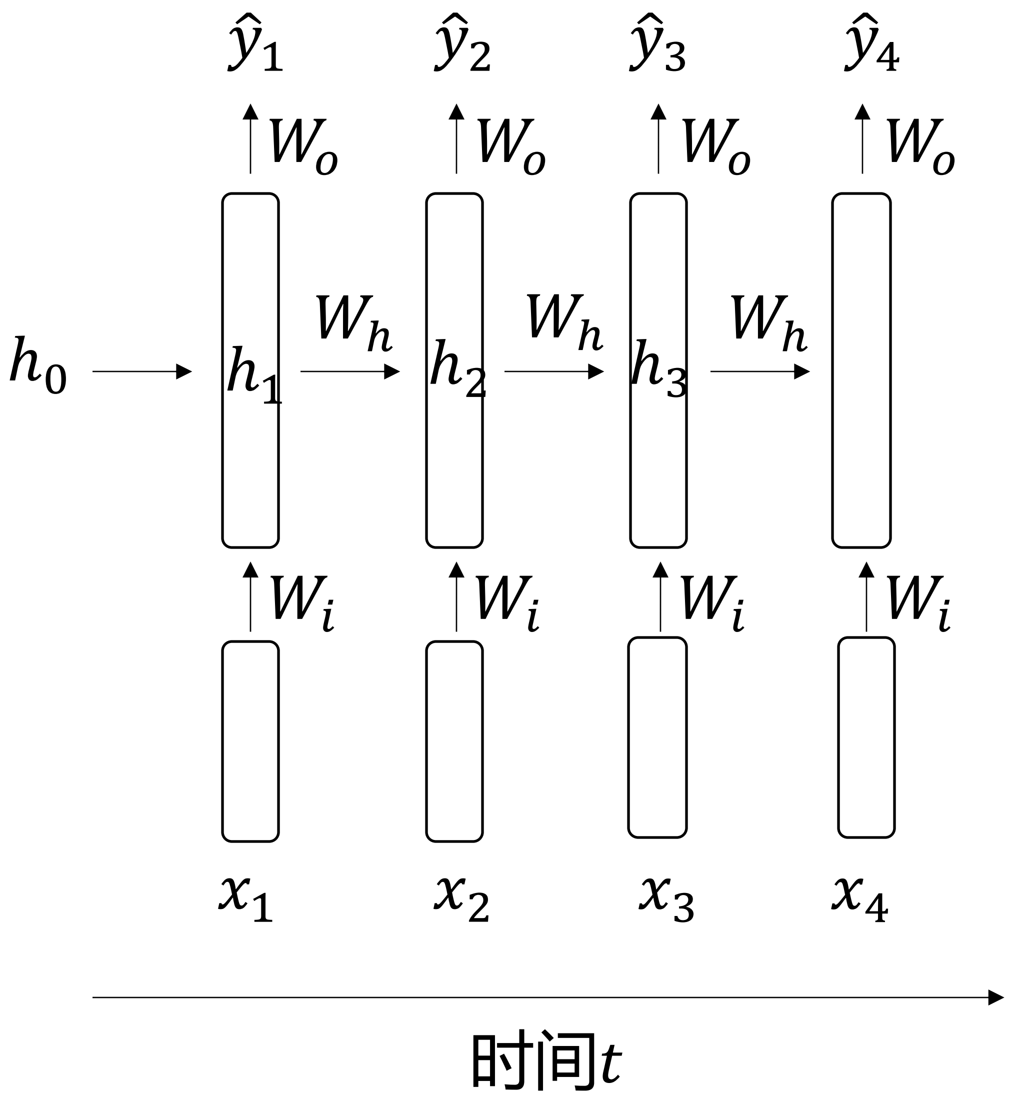
- 神经网络第t时刻的内部状态为h_t， $h_t \in \mathbb{R}^n$
- h_t依赖于上一时刻的状态h_{t-1}以及当前时刻的输入x_t

$$\boxed{h_t = \phi(W_i x_t + W_h h_{t-1})}$$

- ϕ为非线性激活函数
- 各时刻共享W_h
- $W_h \in \mathbb{R}^{n \times n}$， $W_i \in \mathbb{R}^{n \times d}$

输入x_t　　内部状态h_t　　输出?

回复式神经网络

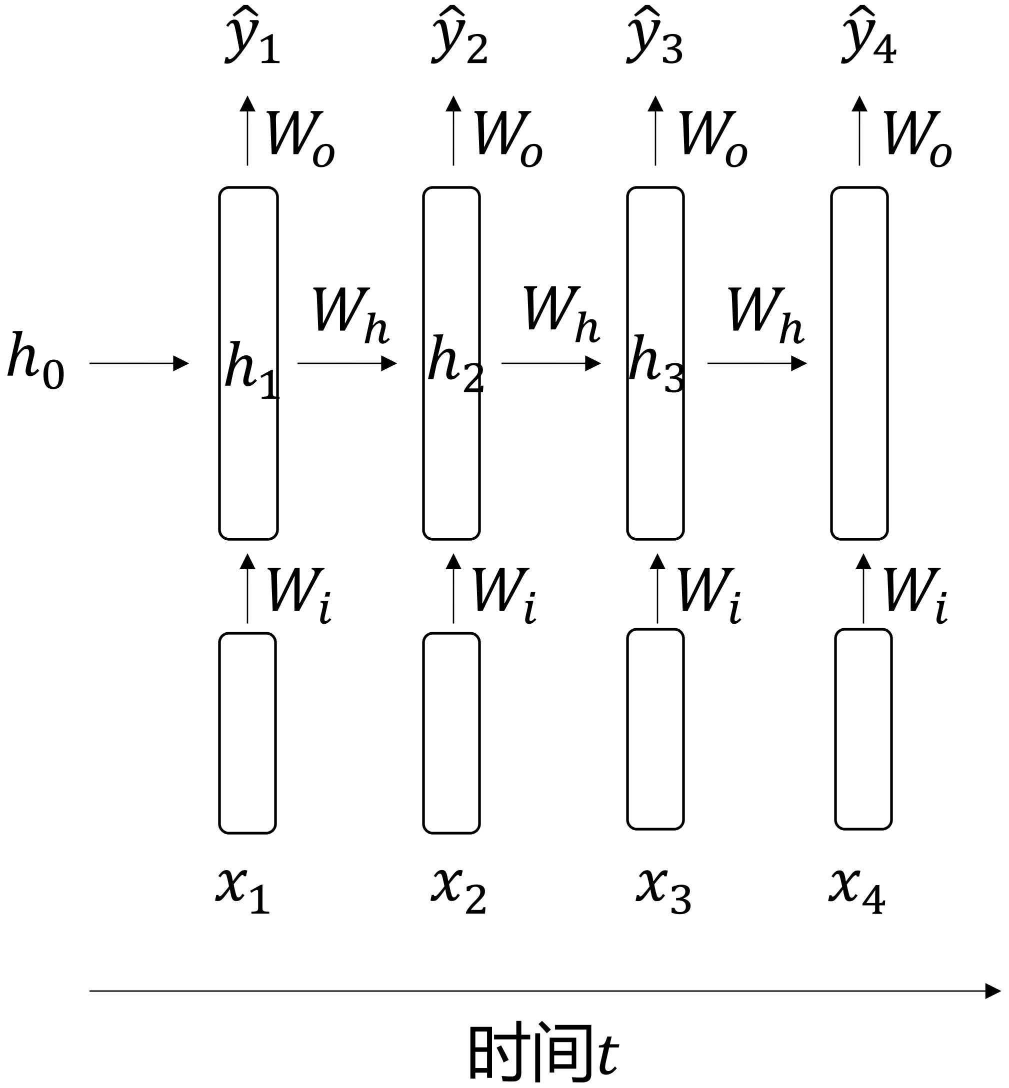


- h_t依赖于上一时刻的状态h_{t-1}以及当前时刻的输入x_t

$$\hat{y}_t = softmax(W_o h_t)$$

- $\hat{y}_t \in \mathbb{R}^{|V|}$，$W_o \in \mathbb{R}^{|V|\times n}$， $|V|$代表词典大小
- 各输出共享W_o
- 针对序列生成或序列分类问题，$\hat{y}_t$为可选

回复式神经网络



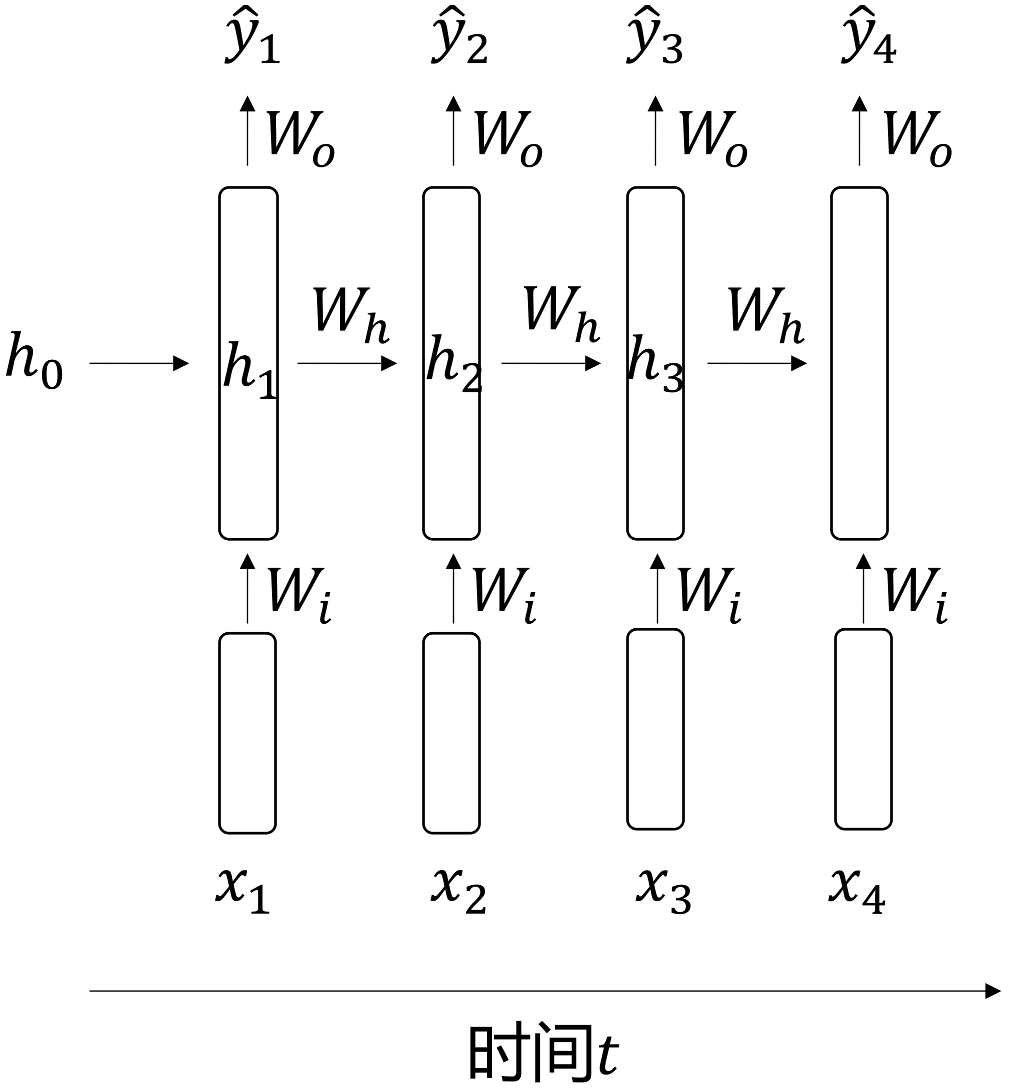
- Recurrent Neural Networks (RNNs)
- 回复式神经网络、循环神经网络……

$$\begin{cases} h_t = \phi(W_i x_t + W_h h_{t-1}) \\ \hat{y}_t = softmax(W_o h_t) \end{cases}$$

- x_t：外部输入，h_t：内部输入

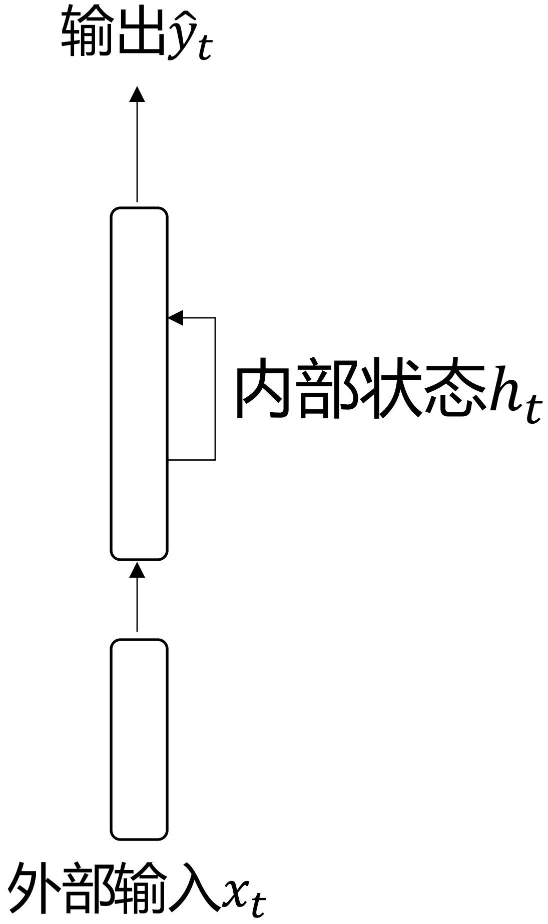
以上公式省略了偏置项b

回复式神经网络



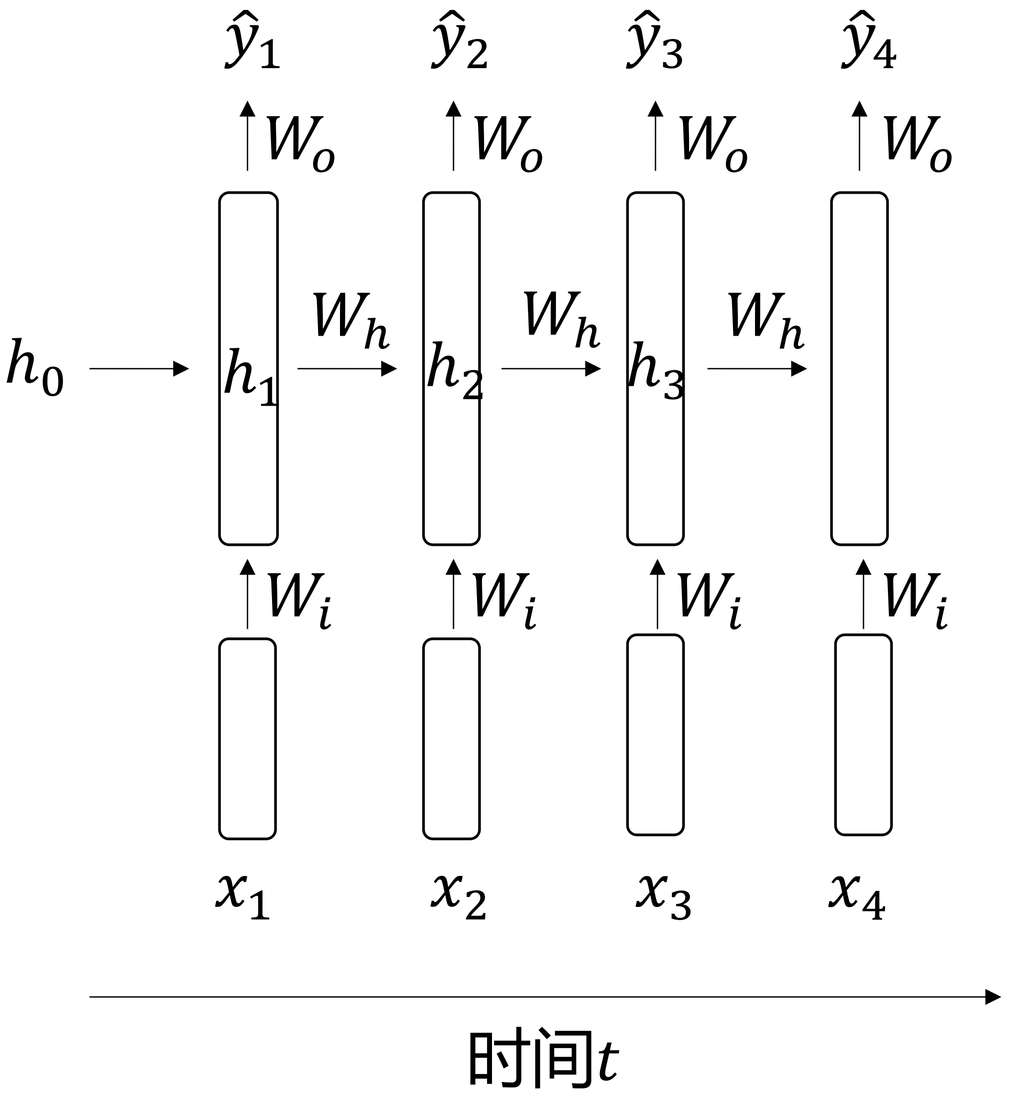
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
W_o W_o W_o W_o
h_0 h_1 W_h h_2 W_h h_3 W_h
W_i W_i W_i W_i
x_1 x_2 x_3 x_4
时间t

简化



输出$\hat{y}_t$
内部状态h_t
外部输入x_t

回复式神经网络



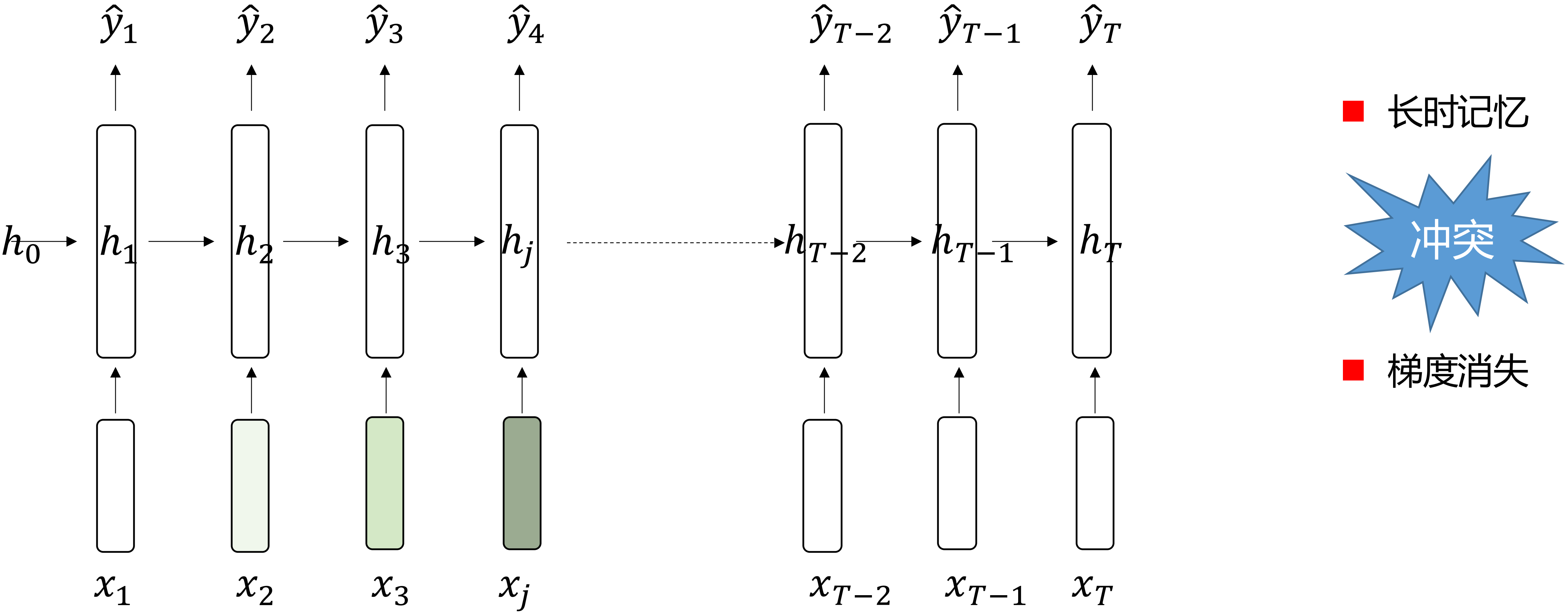
- 第t时刻的代价函数J_t

$$J_t = -\sum_{j=1}^{|V|} y_{t,j} \ln \hat{y}_{t,j}$$

- 全体时刻的代价函数J

$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} \ln \hat{y}_{t,j}$$

回复式神经网络



- 长时记忆
- 梯度消失

回复式神经网络

$$\frac{\partial J_t}{\partial W_h} = \sum_{k=1}^{t} \frac{\partial J_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W_h}$$

$$\frac{\partial J}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial W_h}$$

$$h_t = \phi(W_i x_t + W_h h_{t-1})$$

$$\begin{cases} net_t = W_i x_t + W_h h_{t-1} \\ h_t = \phi(net_t) \end{cases}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

$$\frac{\partial h_j}{\partial h_{j-1}} = \begin{bmatrix} \frac{\partial h_j}{\partial h_{j-1,1}} \\ \cdots \\ \frac{\partial h_j}{\partial h_{j-1,n}} \end{bmatrix} = \begin{bmatrix} \frac{\partial h_{j,1}}{\partial h_{j-1,1}} & \cdots & \frac{\partial h_{j,n}}{\partial h_{j-1,1}} \\ \cdots & \cdots & \cdots \\ \frac{\partial h_{j,1}}{\partial h_{j-1,n}} & \cdots & \frac{\partial h_{j,n}}{\partial h_{j-1,n}} \end{bmatrix} = \begin{bmatrix} W_{11}\dot{\phi}_1 & \cdots & W_{n1}\dot{\phi}_n \\ \cdots & \cdots & \cdots \\ W_{1n}\dot{\phi}_1 & \cdots & W_{nn}\dot{\phi}_n \end{bmatrix}$$

$$= \begin{bmatrix} W_{11} & \cdots & W_{n1} \\ \cdots & \cdots & \cdots \\ W_{1n} & \cdots & W_{nn} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 & & \\ & \cdots & \\ & & \dot{\phi}_n \end{bmatrix} = W^T \mathrm{diag}[\dot{\phi}(net_t)]$$

回复式神经网络

$$\frac{\partial J}{\partial w_h} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial w_h}$$

$$h_t = \phi(U_i x_t + W_h h_{t-1})$$

$$\begin{cases} net_t = U_i x_t + W_h h_{t-1} \\ h_t = \phi(net_t) \end{cases}$$

$$\frac{\partial J_t}{\partial W_h} = \sum_{k=1}^{t} \frac{\partial J_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W_h}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W_h^T \mathrm{diag}[\dot{\phi}(net_t)]$$

$$\begin{aligned} \left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| &= \left\| W_h^T \mathrm{diag}[\dot{\phi}(net_t)] \right\| \\ &\leq \left\| W_h^T \right\| \left\| \mathrm{diag}[\dot{\phi}(net_t)] \right\| \\ &\leq \beta_w \beta_h \end{aligned}$$

定理

$$\|AB\| \leq \|A\| \cdot \|B\|$$

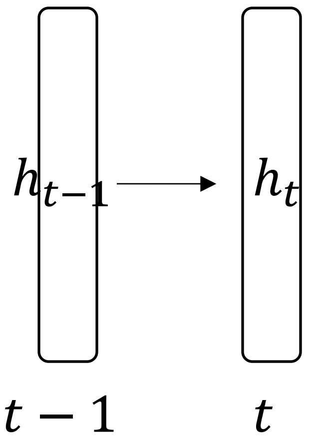
$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_w \beta_h)^{t-k}$$

- $\beta_w \beta_h < 1$，梯度消失，无法解决长时依赖
- $\beta_w \beta_h > 1$，梯度爆炸，网络无法训练

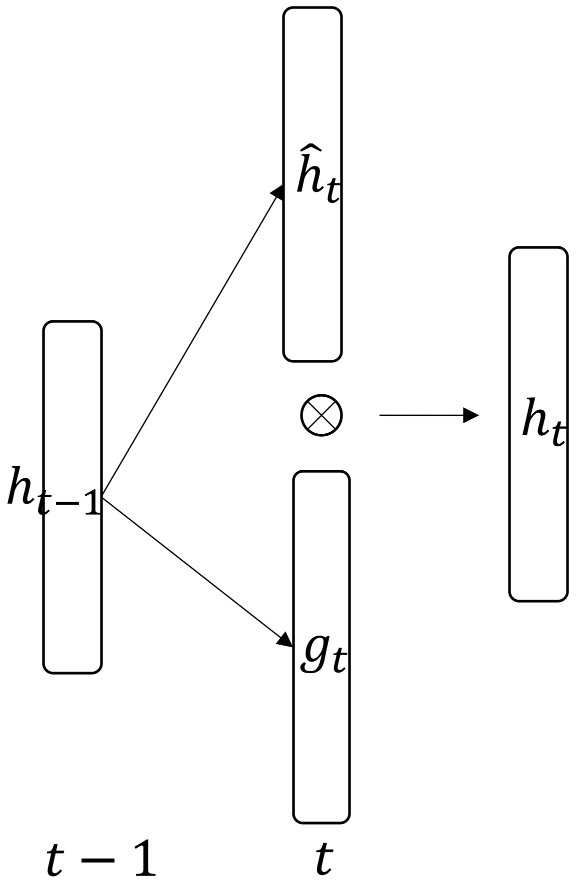
大纲

- 序列预测问题
- 回复式神经网络
- 门控机制
- 自注意力机制

门控机制



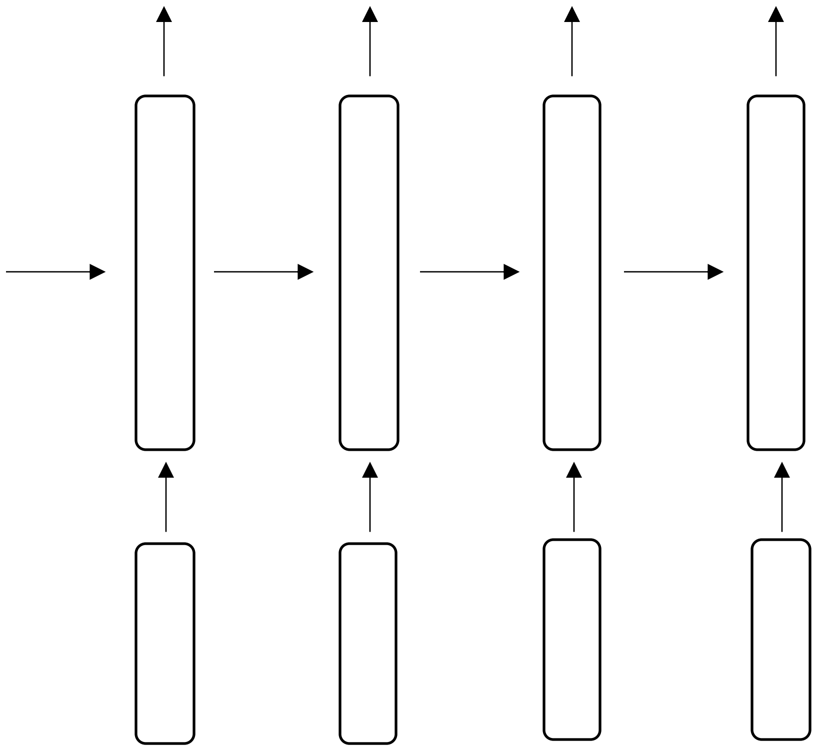
$$h_t = \phi(W_i x_t + W_h h_{t-1})$$



$$\begin{cases} \hat{h}_t = \phi(W_i x_t + W_h h_{t-1}) \\ g_t = \sigma\left(U_g x_t + W_g h_{t-1}\right) \\ h_t = \hat{h}_t \circ g_t \end{cases}$$

- $\sigma(x) = \frac{1}{1+e^{-x}}$, $\sigma(x) \in (0,1)$
- g_t为可调整的门（gate），通过$\hat{h}_t$与g_t对位元素相乘，实现控制$\hat{h}_t$的前向流动

Cell



时间t

- 引入Cell存储知识，实现记忆机制
- c_t代表t时刻的记忆
  - 上一时刻的记忆c_{t-1}对c_t的贡献
  - 当前时刻输入x_t对记忆c_t的贡献
  - c_t的输出

- 使用gate控制c_t

Cell

- 引入Cell存储知识，实现记忆机制
- c_t代表t时刻的记忆
  - 上一时刻的记忆c_{t-1}对c_t的贡献
  - 当前时刻输入x_t对记忆c_t的贡献
  - c_t的输出

- 遗忘门（forget gate）

$$f_t = \sigma\left(U_f x_t + W_f h_{t-1}\right)$$

- 输入信息

$$z_t = \phi(U_z x_t + W_z h_{t-1})$$

- 输入门（input gate）

$$i_t = \sigma(U_i x_t + W_i h_{t-1})$$

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

Cell

- 引入Cell存储知识，实现记忆机制
- c_t代表t时刻的记忆
  - 上一时刻的记忆c_{t-1}对c_t的贡献
  - 当前时刻输入x_t对记忆c_t的贡献
  - c_t的输出h_t

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

- 输出门（output gate）

$$o_t = \sigma(U_o x_t + W_o h_{t-1})$$

$$h_t = o_t \circ \phi(c_t)$$

门控机制

- 输入信息： $z_t = \phi(U_z x_t + W_z h_{t-1})$
- 输入门：$i_t = \sigma(U_i x_t + W_i h_{t-1})$
- 遗忘门：$f_t = \sigma\left(U_f x_t + W_f h_{t-1}\right)$
- 输出门：$o_t = \sigma(U_o x_t + W_o h_{t-1})$

- Cell更新：$c_t = f_t \circ c_{t-1} + i_t \circ z_t$

- Cell输出：$h_t = o_t \circ \phi(c_t)$

加入peephole连接 $\longrightarrow$

- 输入信息： $z_t = \phi(U_z x_t + W_z h_{t-1})$
- 输入门：$i_t = \sigma(U_i x_t + W_i h_{t-1} + P_i c_{t-1})$
- 遗忘门：$f_t = \sigma\left(U_f x_t + W_f h_{t-1} + P_f c_{t-1}\right)$
- 输出门：$o_t = \sigma(U_o x_t + W_o h_{t-1} + P_o c_t)$

- Cell更新：$c_t = f_t \circ c_{t-1} + i_t \circ z_t$

- Cell输出：$h_t = o_t \circ \phi(c_t)$

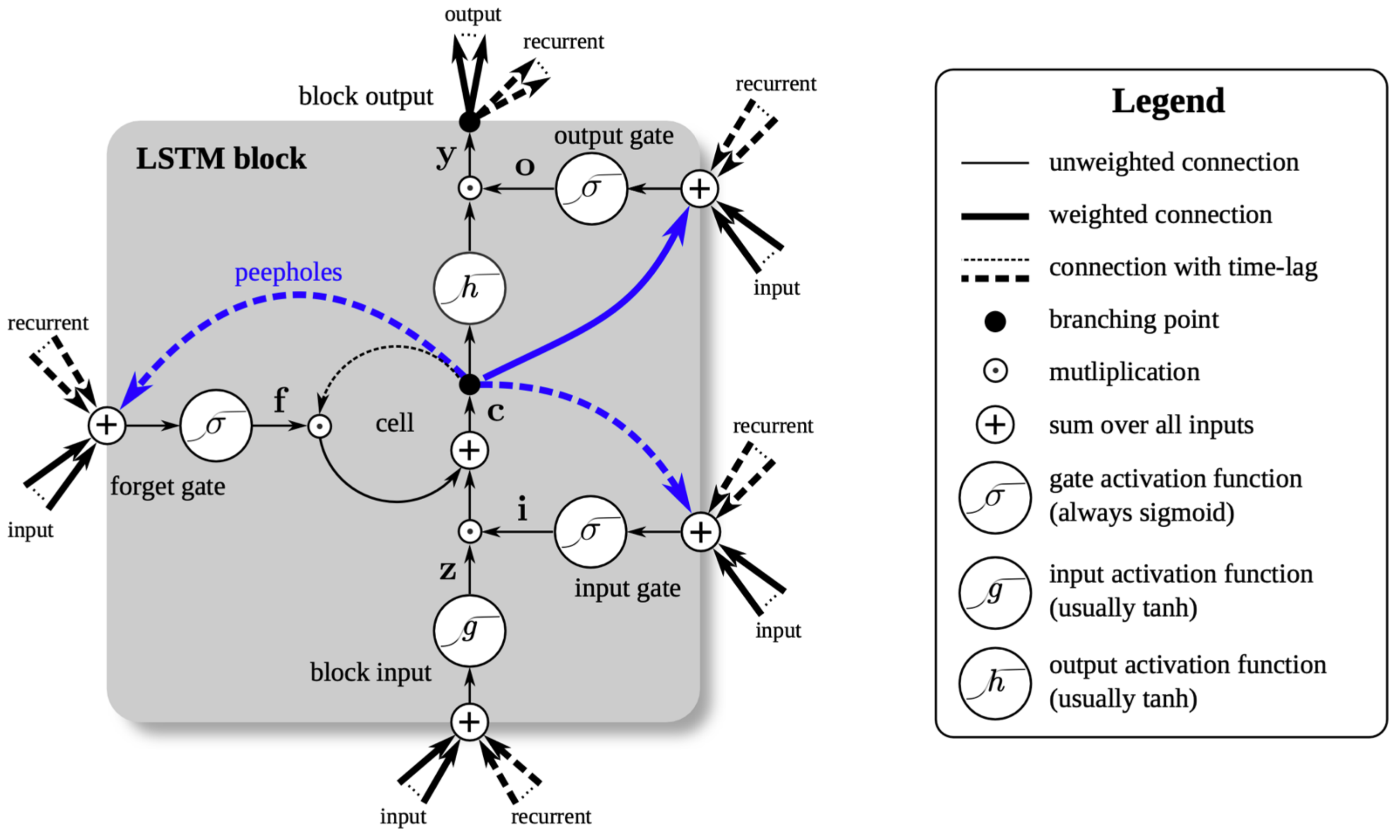
LSTM: Long Short-Term Memory

门控机制

- 输入信息：$z_t = \phi(U_z x_t + W_z h_{t-1})$
- 输入门：$i_t = \sigma(U_i x_t + W_i h_{t-1} + P_i c_{t-1})$
- 遗忘门：$f_t = \sigma\big(U_f x_t + W_f h_{t-1} + P_f c_{t-1}\big)$
- 输出门：$o_t = \sigma(U_o x_t + W_o h_{t-1} + P_o c_t)$

- Cell更新：$c_t = f_t \circ c_{t-1} + i_t \circ z_t$

- Cell输出：$h_t = o_t \circ \phi(c_t)$

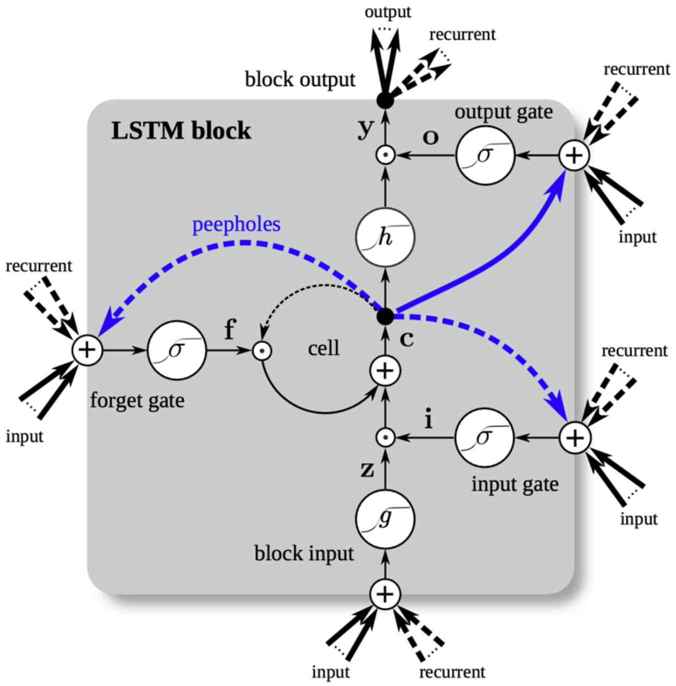


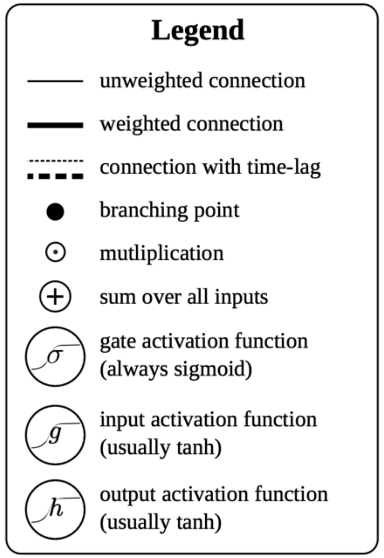
[*] Greff K, Srivastava R K, Koutník J, et al. LSTM: A search space odyssey[J]. IEEE transactions on neural networks and learning systems, 2016, 28(10): 2222-2232.

大纲

- 序列预测问题
- 回复式神经网络
- 门控机制
- 自注意力机制

自注意力机制

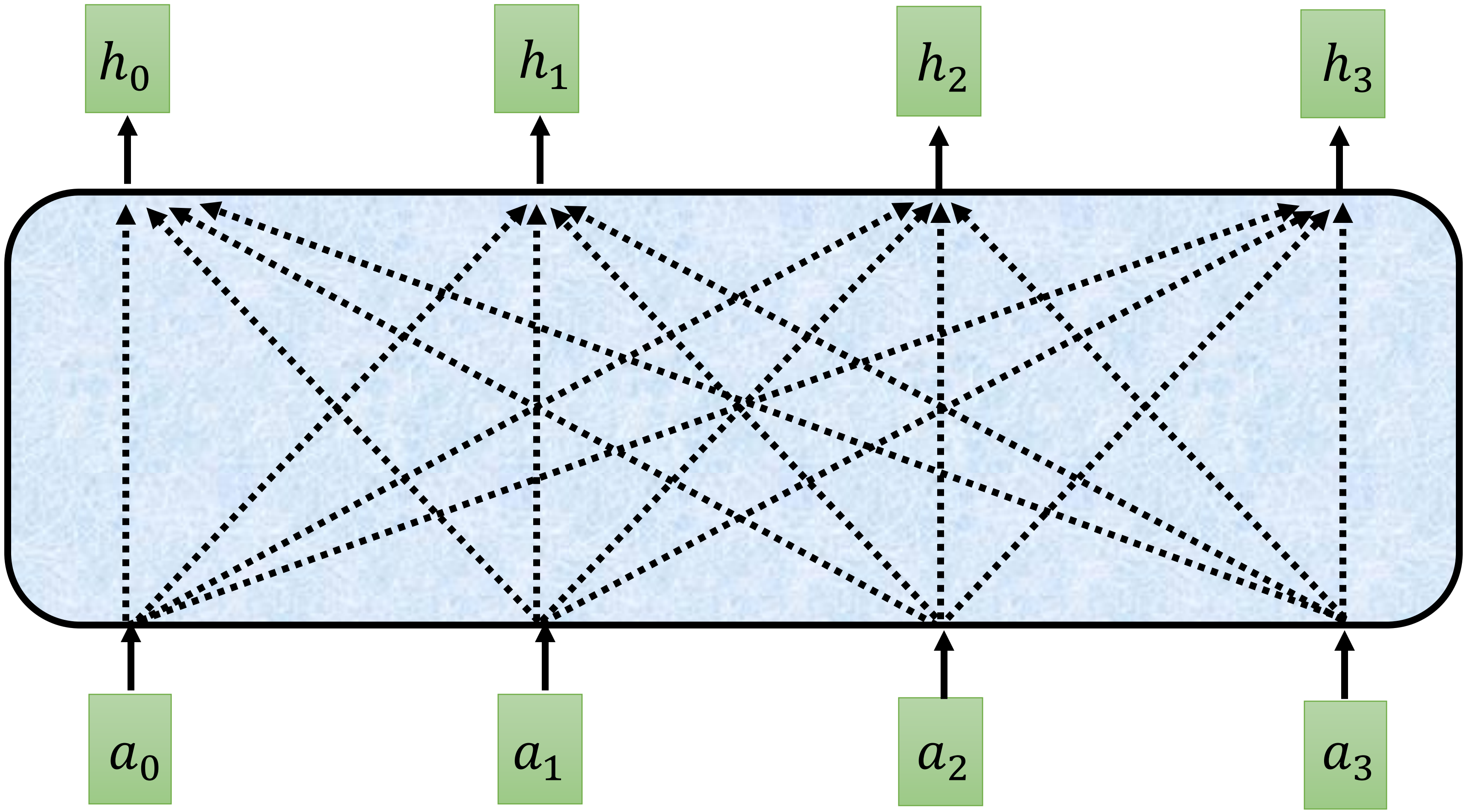




RNNs的局限

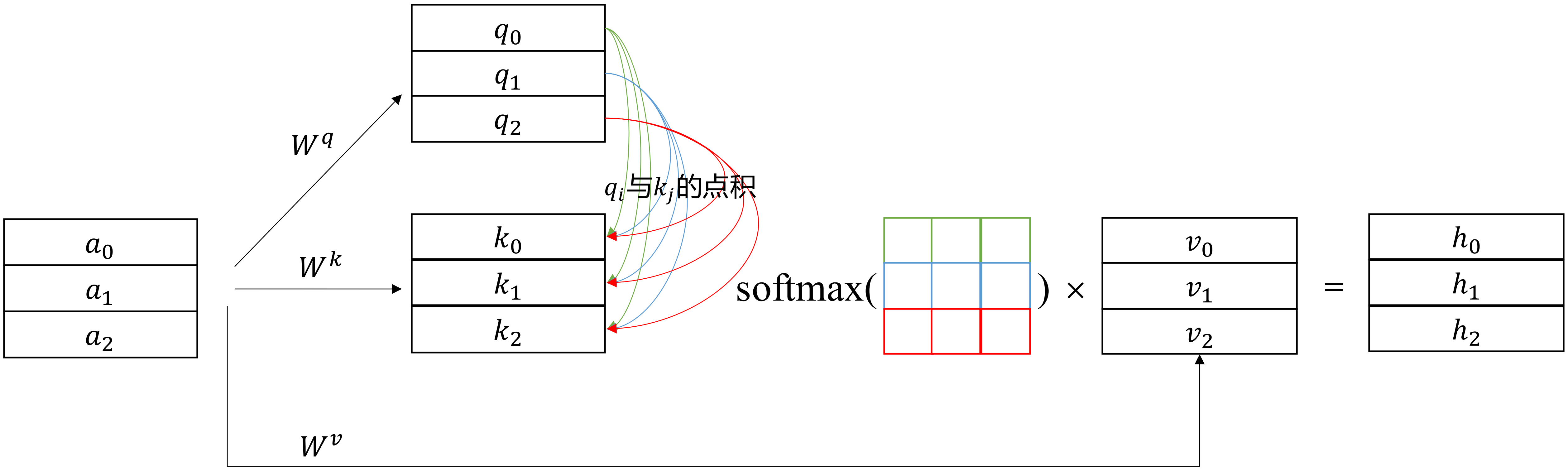
- 长时依赖仍限制RNNs的应用
- 由于时间依赖，RNNs无法并行

自注意力机制



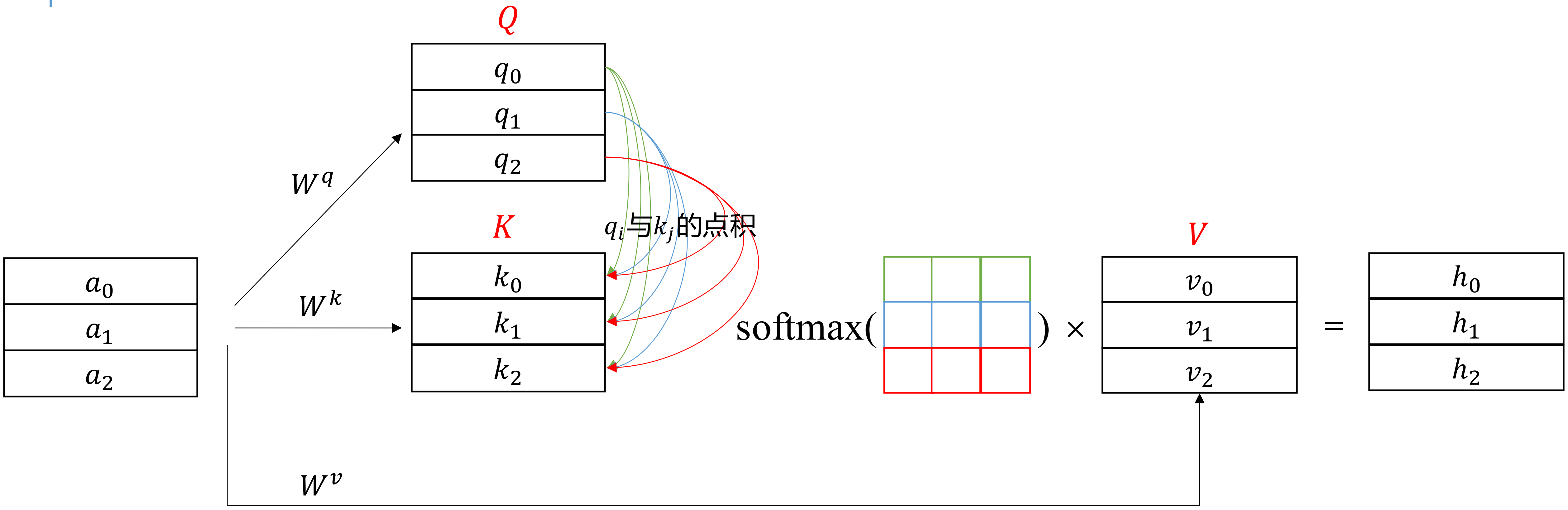
- a_0, a_1, a_2, a_3为序列输入
- RNNs逐步（局部）处理a_0至a_3，带来长时依赖问题
- 通过模型直接学习a_0, a_1, a_2, a_3之间的全局信息

自注意力机制



- a_t为序列输入， $a_t \in \mathbb{R}^{1\times n}$
- W^q, W^k, W^v为可学习矩阵
- $W^q \in \mathbb{R}^{n\times m}, W^k \in \mathbb{R}^{n\times m}, W^v \in \mathbb{R}^{n\times d}$
- $q_t \in \mathbb{R}^{1\times m}, k_t \in \mathbb{R}^{1\times m}, v_t \in \mathbb{R}^{1\times d}$

自注意力机制

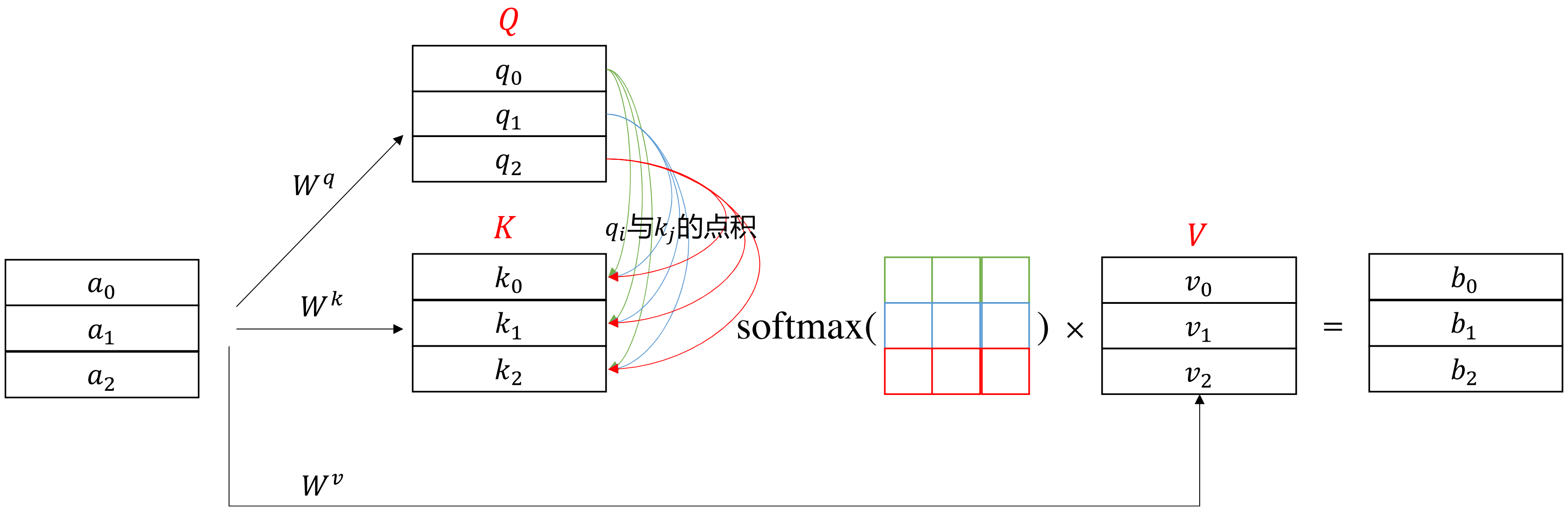


$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{n}}\right)V$$

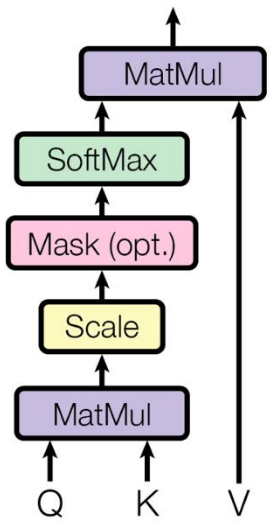
$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- n代表a_t的维度， n 过大时， 将导致QK^T方差过大，softmax归一化后的数值分布差异将过大，影响计算的梯度强度

自注意力机制



Scaled Dot-Product Attention



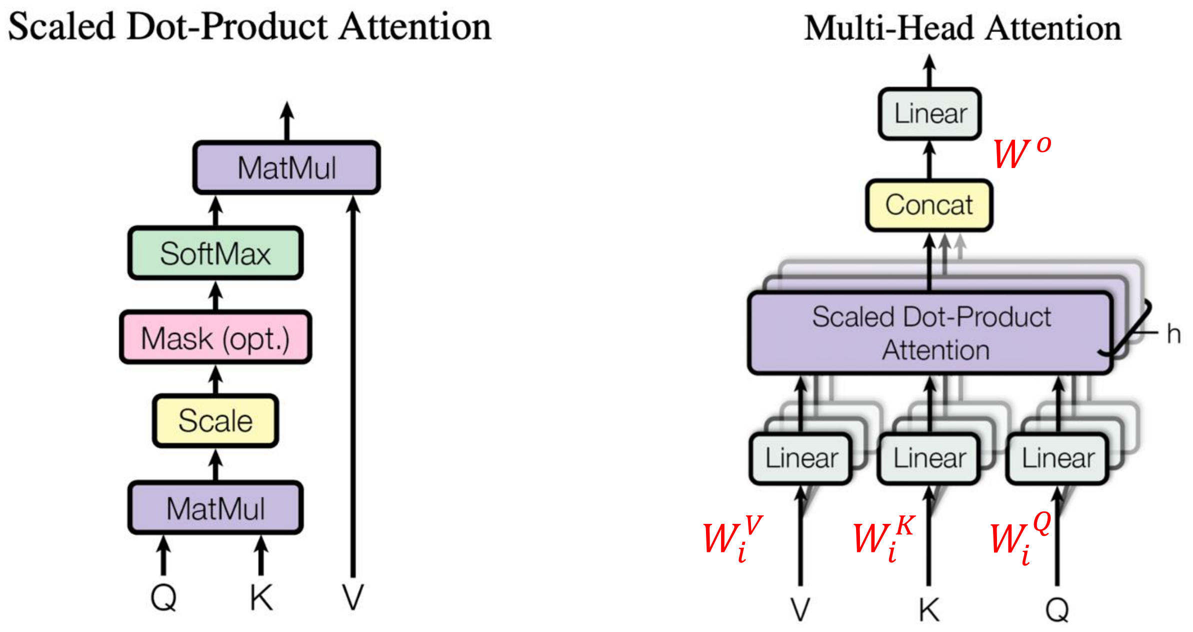
Attention is **all you need**

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

… to attend to **all** positions in the decoder up to and including that position. **We need** to prevent … **We** implement this inside of scaled dot-product **attention** by masking out (setting to −∞) …

Save Cite Cited by 178597 Related articles All 73 versions

自注意力机制

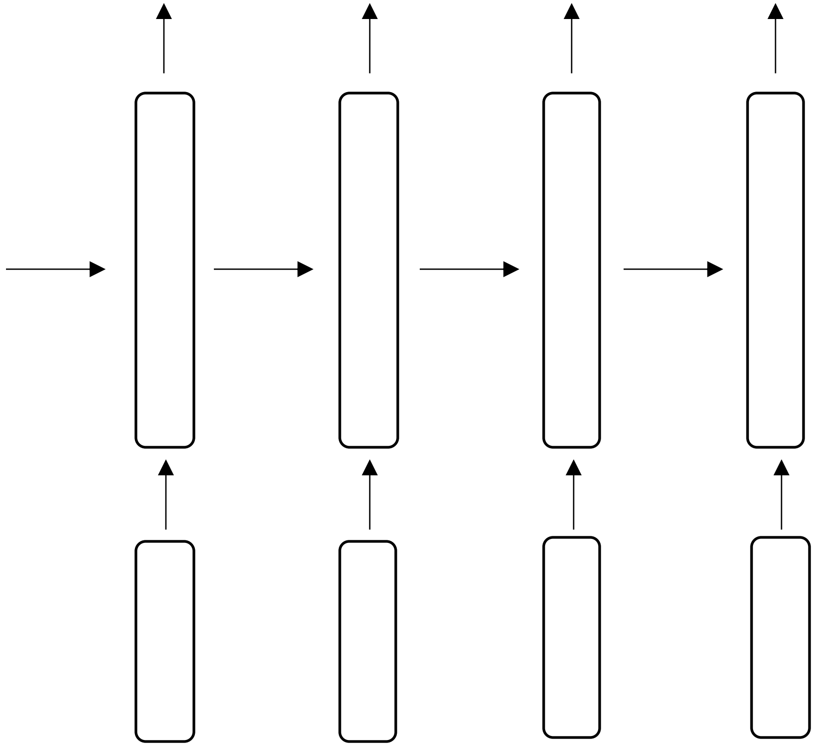


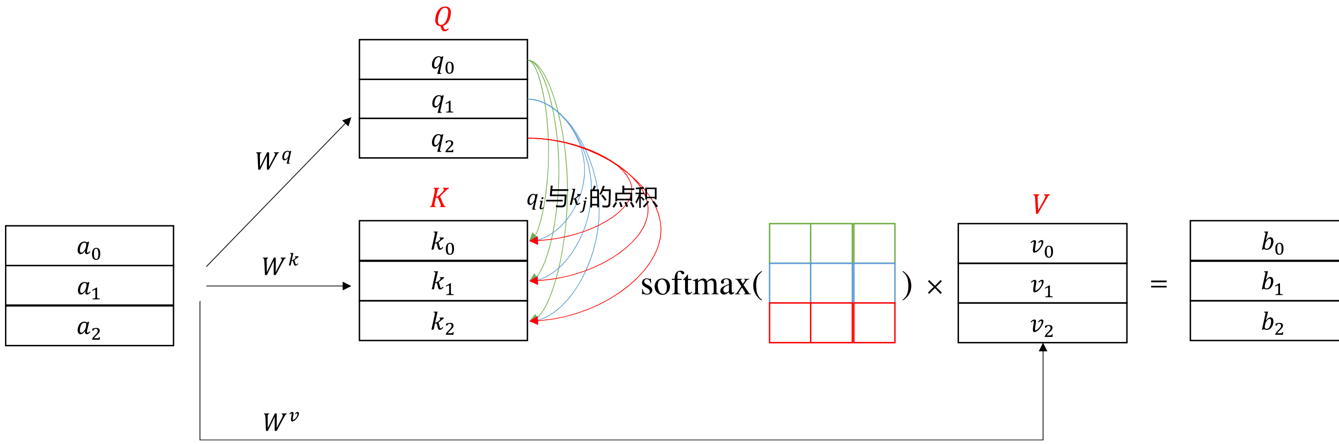
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1,\ldots,\mathrm{head}_h)W^O$$

$$\mathrm{head_i} = \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

- h组并行的自注意力计算，增加特征的多样性（联想CNN的通道数）
- h组输出Concat后由W^o进行映射

自注意力机制





- RNNs的序列输入特性隐含地包含了位置信息

- 自注意力机制未考虑位置关系
- 如何在自注意力机制中加入位置信息?

自注意力机制

0: 0000
1: 0001
2: 0010
3: 0011
4: 0100
5: 0101
6: 0110
7: 0111
8: 1000
9: 1001
10: 1010

- 二进制的位置编码
- 如何使用浮点型向量编码位置?

- 论文给出的位置编码方案:

$$P = \begin{bmatrix} & \sin\left(\frac{t}{f_1}\right) & \\ & \cos\left(\frac{t}{f_1}\right) & \\ & \sin\left(\frac{t}{f_2}\right) & \\ & \cos\left(\frac{t}{f_2}\right) & \\ \cdots & \cdots & \cdots \\ & \sin\left(\frac{t}{f_{\frac{d}{2}}}\right) & \\ & \cos\left(\frac{t}{f_{\frac{d}{2}}}\right) & \end{bmatrix}$$

- 以上列出了第t个位置的向量编码，向量维度为d
- 偶数维度使用sin，奇数维度使用cos
- f_i代表第i维的频率

自注意力机制

$$P=\begin{bmatrix} & \boxed{\begin{matrix}\sin\left(\frac{t}{f_1}\right)\\ \cos\left(\frac{t}{f_1}\right)\end{matrix}} & \\ & \sin\left(\frac{t}{f_2}\right) & \\ & \cos\left(\frac{t}{f_2}\right) & \\ \cdots & \cdots & \cdots \\ & \sin\left(\frac{t}{f_{\frac{d}{2}}}\right) & \\ & \cos\left(\frac{t}{f_{\frac{d}{2}}}\right) & \end{bmatrix}$$

即要求存在 $\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}\begin{bmatrix}\sin\left(\frac{t}{f_1}\right)\\ \cos\left(\frac{t}{f_1}\right)\end{bmatrix}=\begin{bmatrix}\sin\left(\frac{t+k}{f_1}\right)\\ \cos\left(\frac{t+k}{f_1}\right)\end{bmatrix}$

三角函数和差化积

$\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$

$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$

$$\begin{bmatrix} u_1\sin\left(\frac{t}{f_1}\right)+v_1\cos\left(\frac{t}{f_1}\right) \\ u_2\sin\left(\frac{t}{f_1}\right)+v_2\cos\left(\frac{t}{f_1}\right)\end{bmatrix}=\begin{bmatrix}\cos\left(\frac{k}{f_1}\right)\sin\left(\frac{t}{f_1}\right)+\sin\left(\frac{k}{f_1}\right)\cos\left(\frac{t}{f_1}\right) \\ \cos\left(\frac{k}{f_1}\right)\cos\left(\frac{t}{f_1}\right)-\sin\left(\frac{k}{f_1}\right)\sin\left(\frac{t}{f_1}\right)\end{bmatrix}$$

显然，$u_1=\cos\left(\frac{k}{f_1}\right), v_1=\sin\left(\frac{k}{f_1}\right), u_2=-\sin\left(\frac{k}{f_1}\right), v_2=\cos\left(\frac{k}{f_1}\right)$

- u_1,u_2,v_1,v_2与时间t无关

- 记p_t为位置t的编码，$p_t\in\mathbb{R}^{d\times 1}$
- 存在线性变换$M_k\in\mathbb{R}^{d\times d}$，使得$M_k p_t=p_{t+k}$

自注意力机制

- 使用余弦相似度度量p_t与p_{t+k}之间的距离
- 位置编码之间的余弦相似度仅与位置差 k有关，而与原始位置t无关

$$\left[\sin\left(\frac{t}{f_i}\right) \quad \cos\left(\frac{t}{f_i}\right)\right]\begin{bmatrix}\sin\left(\dfrac{t+k}{f_i}\right)\\ \cos\left(\dfrac{t+k}{f_i}\right)\end{bmatrix}$$

$$= \sin\left(\frac{t}{f_i}\right)\left[\sin\left(\frac{t}{f_i}\right)\cos\left(\frac{k}{f_i}\right) + \cos\left(\frac{t}{f_i}\right)\sin\left(\frac{k}{f_i}\right)\right] + \cos\left(\frac{t}{f_i}\right)\left[\cos\left(\frac{k}{f_i}\right)\cos\left(\frac{t}{f_i}\right) - \sin\left(\frac{k}{f_i}\right)\sin\left(\frac{t}{f_i}\right)\right]$$

$$= \sin^2\left(\frac{t}{f_i}\right)\cos\left(\frac{k}{f_i}\right) + \sin\left(\frac{t}{f_i}\right)\sin\left(\frac{k}{f_i}\right)\cos\left(\frac{t}{f_i}\right) + \cos^2\left(\frac{t}{f_i}\right)\cos\left(\frac{k}{f_i}\right)$$

$$- \sin\left(\frac{k}{f_i}\right)\sin\left(\frac{t}{f_i}\right)\cos\left(\frac{t}{f_i}\right)$$

$$= \cos\left(\frac{k}{f_i}\right)$$

- 论文中$f_i = 10000^{\frac{2i}{d}}$，其中$d$为词向量的长度

Attention is all you need

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

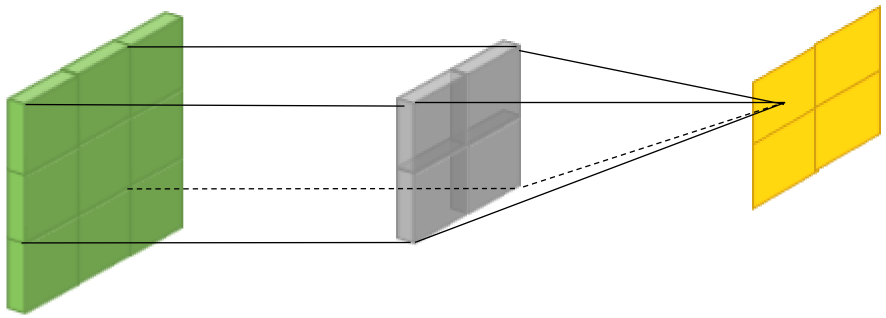
… to attend to **all** positions in the decoder up to and including that position. **We need** to prevent … **We** implement this inside of scaled dot-product **attention** by masking out (setting to −∞) …

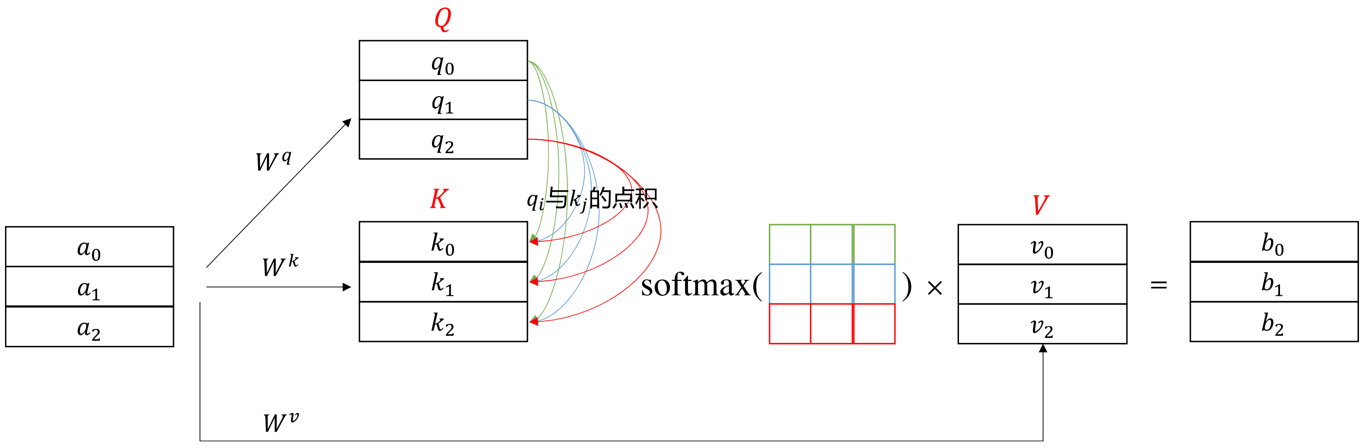
☆ Save 99 Cite Cited by 178597 Related articles All 73 versions »

自注意力机制

Vision Transformers

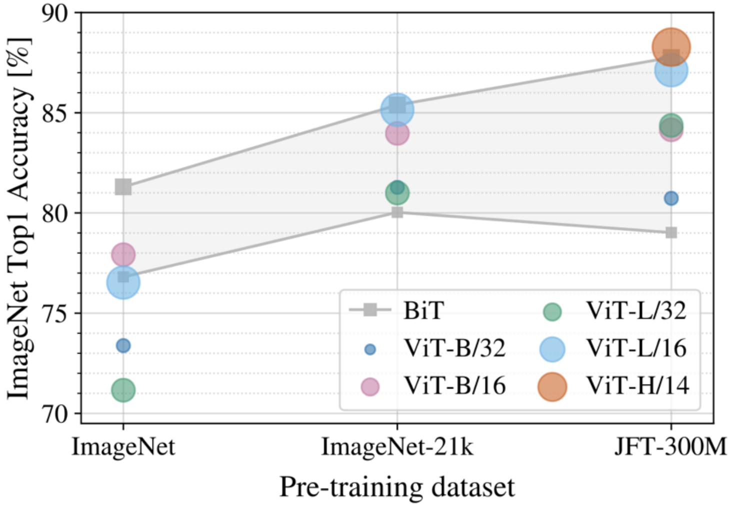
自注意力机制

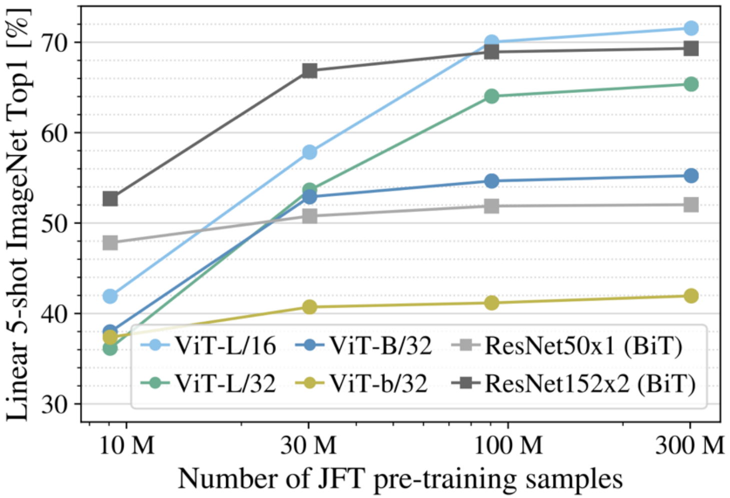




- 通过卷积核学习提取局部特征
- 通过自注意力机制学习提取全局特征

自注意力机制





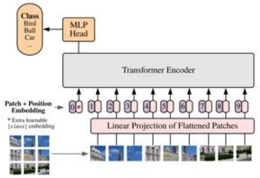
- 数据量较小时，CNNs占优
- 数据量较大时，ViT占优

Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

大数据

大模型



大算力

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

谢 谢！