

# 《机器学习基础》



## 循环神经网络

# 自然语言处理模型(natural language processing)

---

►人工智能和语言学领域的一个分支，旨在使计算机能够理解、解释和产生人类语言的内容。

1. **语音识别**：将语音转换为文本。
2. **自然语言理解**：理解自然语言的意图、情感和语境。
3. **机器翻译**：将一种语言翻译成另一种语言。
4. **文本挖掘和文本分析**：从文本中提取有用信息和洞察。
5. **语言生成**：生成自然语言响应或文本。

# 自然语言处理模型(natural language processing)

---

► **自然语言处理模型** 一个重要的用法就是通过之前的信息来决策当前的问题。

(比如就像我们看电影，我们要根据电影之前的情节，才能理解现在的情节。)

例子1：有一朵云飘在（）

例子2：我从小生长在中国。。。我可以说一口流利的（）

# 自然语言处理模型(natural language processing)

---

- ▶ 词袋模型 (Bag of Words, BoW) : 文本 (如一段话、一篇文章或一个文档) 被表示为其词汇的无序集合, 不考虑语法和词序, 但保留词汇的多重性 (即词汇出现的频率)。
- ▶ One Hot 编码 (One-Hot Encoding) : 每个类别值都被转换成一个二进制向量, 除了表示该类别的一个位置是1之外, 其余位置都是0。这种编码方式因其表示的向量中只有一个元素是"热"的 (即1), 其他位置都是"冷"的 (即0), 因此得名 One Hot 编码。

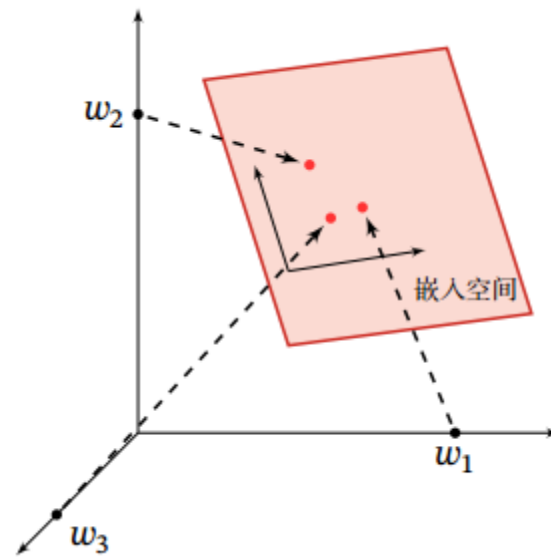
# 什么是Embedding

---

- ▶使用One-hot 方法编码的向量会很高维也很稀疏。假设我们在做自然语言处理（NLP）中遇到了一个包含2000个词的字典，当使用One-hot编码时，每一个词会被一个包含2000个整数的向量来表示，其中1999个数字是0，如果字典再大一点，这种方法的计算效率会大打折扣。

# 什么是Embedding

- ▶ 把高维的局部表示空间映射到一个非常低维的分布式表示空间，例如  $[w_1, 0, 0]$ ，映射到一个更低维的稠密空间上。
- ▶ 在用深度学习做 NLP 任务时，通常我们会在输入层，即字ID后接入一个 Embedding Layer，将正整数转换为具有固定大小的向量。



# 什么是Embedding

---

## ▶ *deep learning is very deep*

- ▶ 使用嵌入层embedding 的第一步是通过索引对该句子进行编码，这里我们给每一个不同的单词分配一个索引，上面的句子就会变成这样：
- ▶ 1、2、3、4、1
- ▶ 我们要决定每一个索引需要分配多少个“潜在因子”，这大体上意味着我们想要多长的向量，通常使用的情况是长度分配为32和50。

# 什么是Embedding

- ▶ 假设每个索引指定6个潜在因子。嵌入矩阵则为

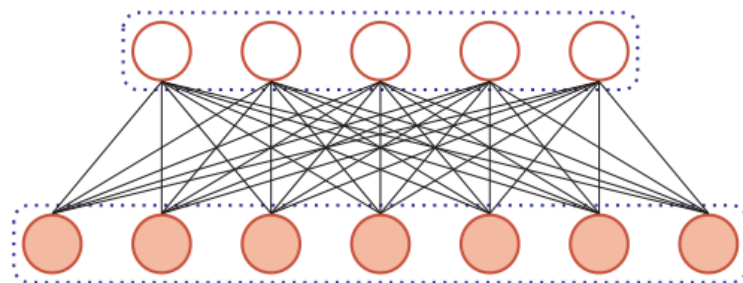
Indices	Latent Factors					
1	.32	.02	.48	.21	.56	.15
2	.65	.23	.41	.57	.03	.92
3	.45	.87	.89	.45	.12	.01
4	.65	.21	.25	.45	.78	.82

- ▶ 这可以使用嵌入矩阵而不是one-hot编码向量来保持每个向量更小。
- ▶ 单词“deep”用向量[.32, .02, .48, .21, .56, .15]来表达。
- ▶ 每一个单词都会被替换为用于查找嵌入矩阵中向量的索引。
- ▶ 大数据时也能有效计算。
- ▶ 可以探索在高维空间中哪些词语之间具有相似性

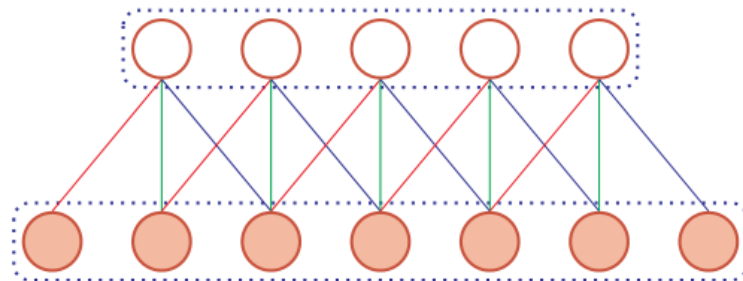


# 前馈网络

- ▶ 连接存在层与层之间，每层的节点之间是无连接的。（无循环）
- ▶ 输入和输出的维数都是固定的，不能任意改变。无法处理变长的序列数据。



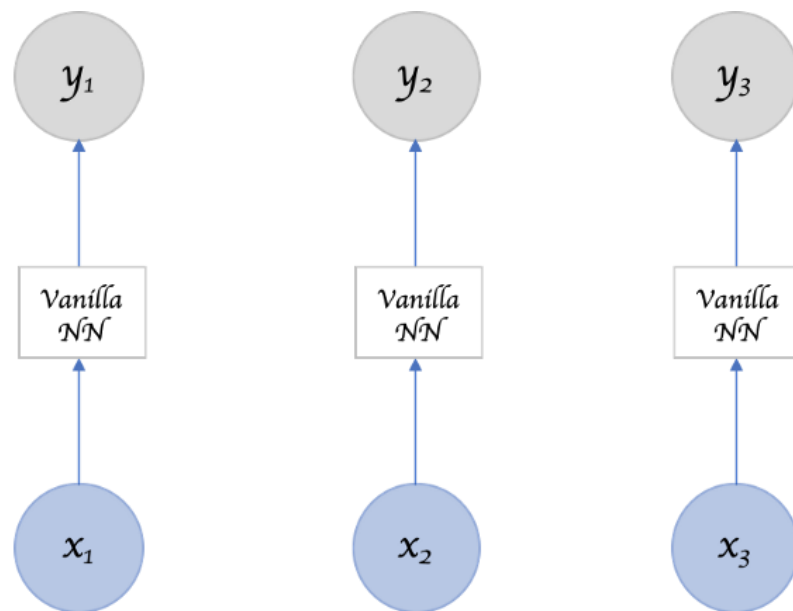
(a) 全连接层



(b) 卷积层

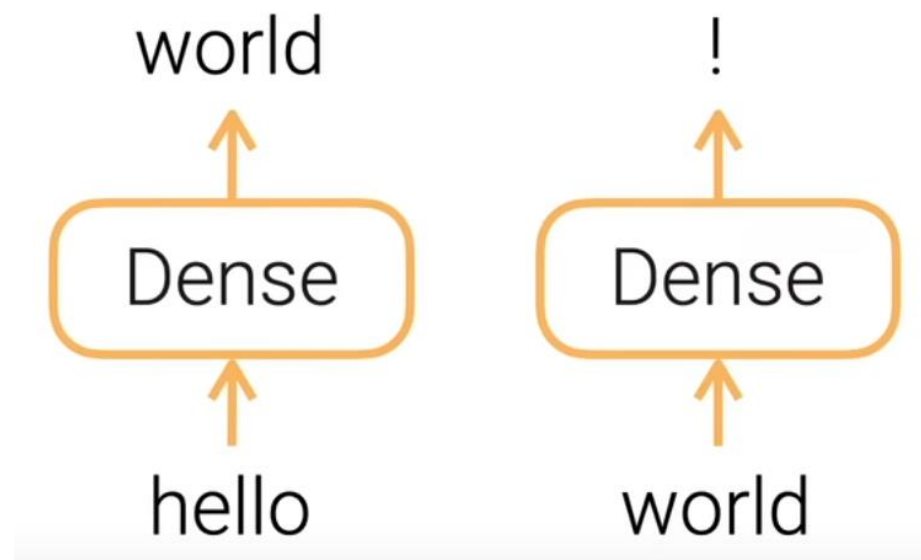
# 前馈网络

- 假设每次输入都是独立的，也就是说每次网络的输出只依赖于当前的输入。



# 自然语言处理模型(natural language processing)

- ▶人工智能和语言学领域的一个分支，旨在使计算机能够理解、解释和产生人类语言的内容。
- ▶如何预测下一个要输出的单词？
  - ▶hello → world      hello world → !
- ▶MLP、CNN无法很好的处理时序数据



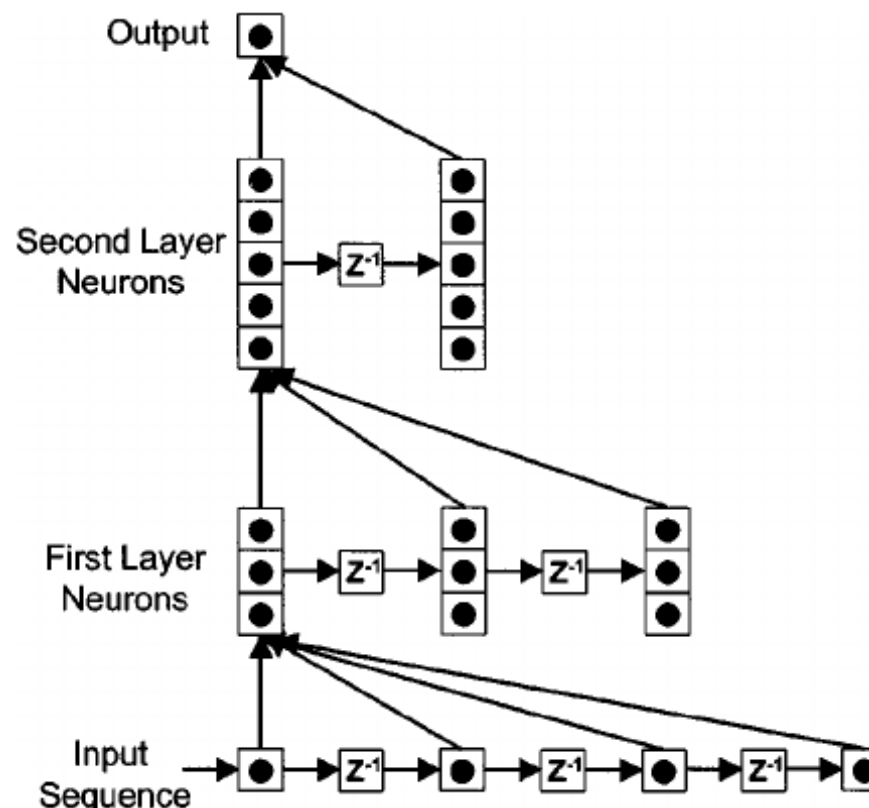
# 如何给网络增加记忆能力？

## ► 延时神经网络 (Time Delay Neural Network, TDNN)

- 建立一个额外的延时单元，用来存储网络的历史信息（可以包括输入、输出、隐状态等）

$$\mathbf{h}_t^{(l)} = f(\mathbf{h}_t^{(l-1)}, \mathbf{h}_{t-1}^{(l-1)}, \dots, \mathbf{h}_{t-K}^{(l-1)})$$

- 这样，前馈网络就具有了短期记忆的能力。



[https://www.researchgate.net/publication/12314435\\_Neural\\_system\\_identification\\_model\\_of\\_human\\_sound\\_localization](https://www.researchgate.net/publication/12314435_Neural_system_identification_model_of_human_sound_localization)

# 如何给网络增加记忆能力?

---

## ▶ 自回归模型 (Autoregressive Model, AR)

▶ 一类时间序列模型，用变量 $y_t$ 的历史信息来预测自己

$$y_t = w_0 + \sum_{k=1}^K w_k y_{t-k} + \epsilon_t$$

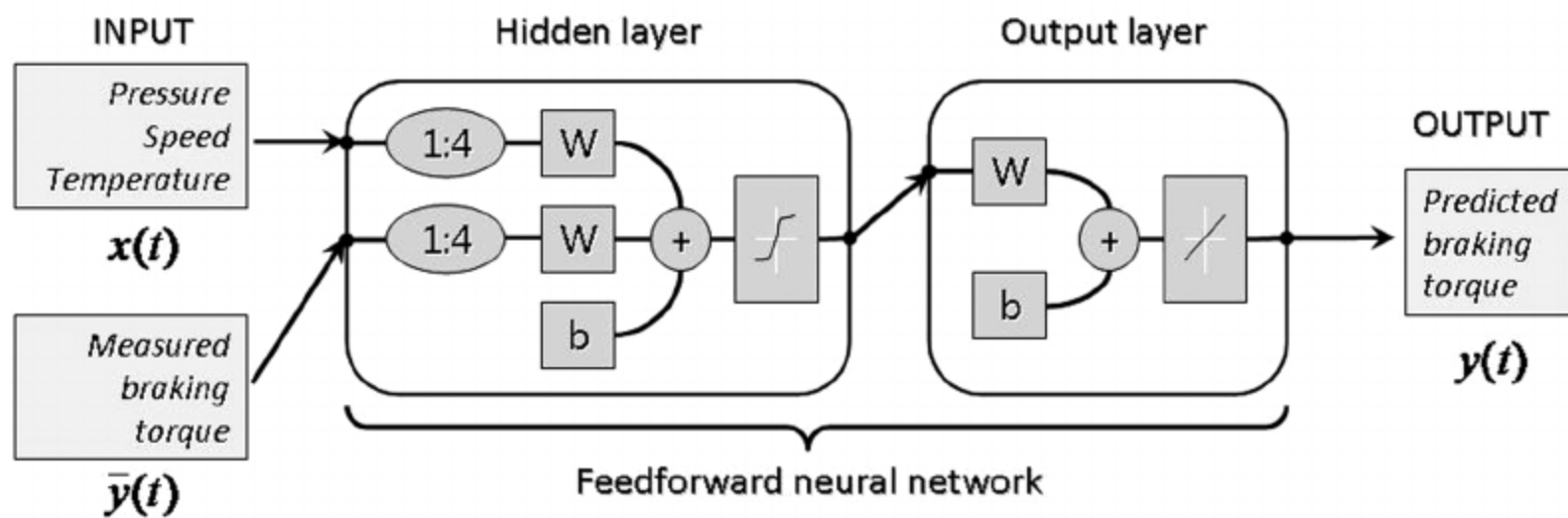
▶  $\epsilon_t \sim N(0, \sigma^2)$  为第 $t$ 个时刻的噪声

## ▶ 有外部输入的非线性自回归模型 (Nonlinear Autoregressive with Exogenous Inputs Model, NARX)

$$y_t = f(\mathbf{x}_t, \mathbf{x}_{t-1}, \dots, \mathbf{x}_{t-K_x}, y_{t-1}, y_{t-2}, \dots, y_{t-K_y})$$

▶ 其中  $f(\cdot)$  表示非线性函数，可以是一个前馈网络， $K_x$  和  $K_y$  为超参数。

# 非线性自回归模型



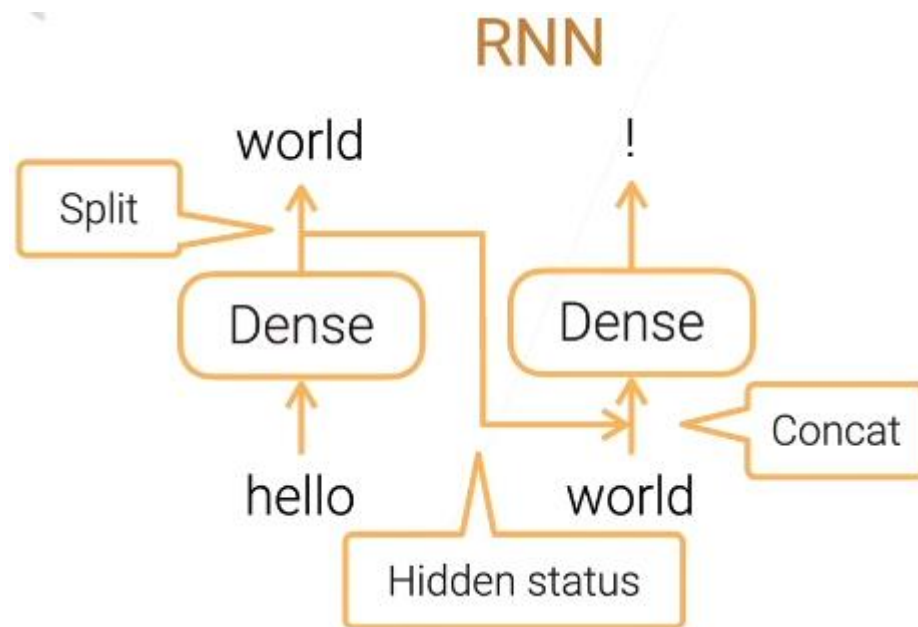
[https://www.researchgate.net/publication/234052442\\_Braking\\_torque\\_control\\_using\\_reccurent\\_neural\\_networks](https://www.researchgate.net/publication/234052442_Braking_torque_control_using_reccurent_neural_networks)

# 循环神经网络 ( Recurrent Neural Network , RNN )

- ▶ 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。

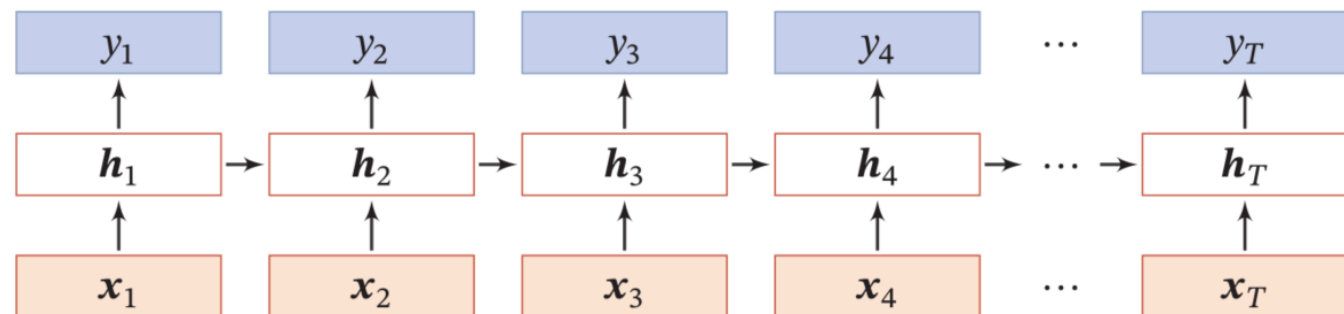
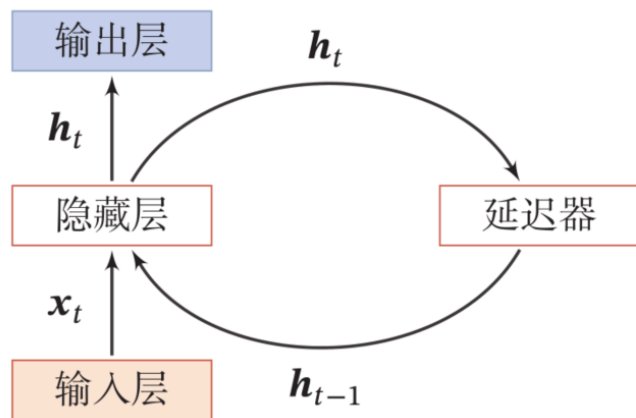
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

活性值  
状态



- ▶ 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- ▶ 循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上

# 按时间展开





# 简单循环网络 ( Simple Recurrent Network , SRN )

## ► 完全连接的循环神经网络

$$\begin{aligned} \mathbf{h}_t &= f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}), \\ \mathbf{y}_t &= \mathbf{V}\mathbf{h}_t, \end{aligned} \quad \mathbf{z}_t = \mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b},$$

## ► 一个完全连接的循环网络是任何非线性动力系统的近似器。

**定理 6.1** – 循环神经网络的通用近似定理 [Haykin, 2009]: 如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元, 它可以以任意的准确率去近似任何一个非线性动力系统

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \mathbf{x}_t), \quad (6.10)$$

$$\mathbf{y}_t = o(\mathbf{s}_t), \quad (6.11)$$

其中  $\mathbf{s}_t$  为每个时刻的隐状态,  $\mathbf{x}_t$  是外部输入,  $g(\cdot)$  是可测的状态转换函数,  $o(\cdot)$  是连续输出函数, 并且对状态空间的紧致性没有限制.

# 图灵完备

---

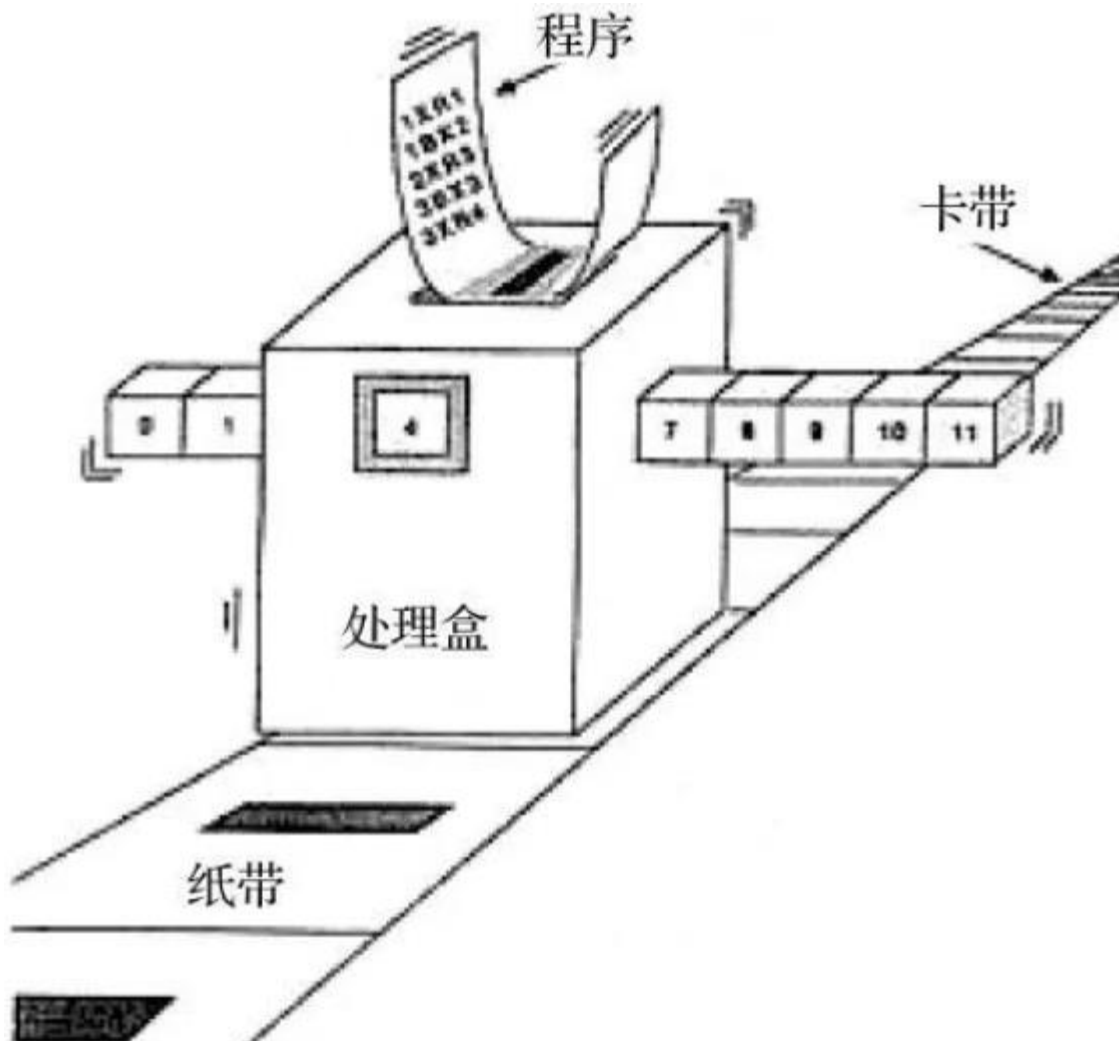
- ▶ 图灵完备 (Turing Completeness) 是指一种数据操作规则，比如一种计算机编程语言，可以实现图灵机的所有功能，解决所有的可计算问题。

**定理 6.2 – 图灵完备 [Siegelmann et al., 1991]:** 所有的图灵机都可以被一个由使用 Sigmoid 型激活函数的神经元构成的全连接循环网络来进行模拟。

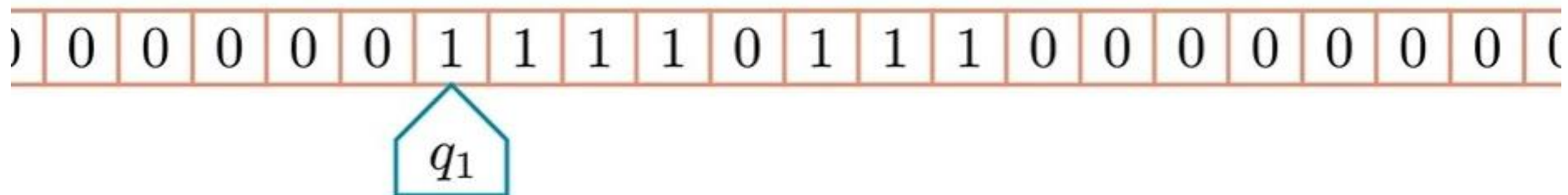
- ▶ 一个完全连接的循环神经网络可以近似解决所有的可计算问题。

# 图灵机

- ▶ 用纸笔进行数学运算的过程进行抽象，由一个虚拟的机器替代人类进行数学运算。
- ▶ 图灵机可以通过简单的读单元格、查指令集表、改变单元格状态、移动纸带这些非常简单的操作来进行非常复杂的数学运算。
- ▶ 图灵机为确定哪些问题是可以计算的提供了一个标准。如果一个问题可以用图灵机解决，那么这个问题就是可计算的；否则，它是不可计算的。

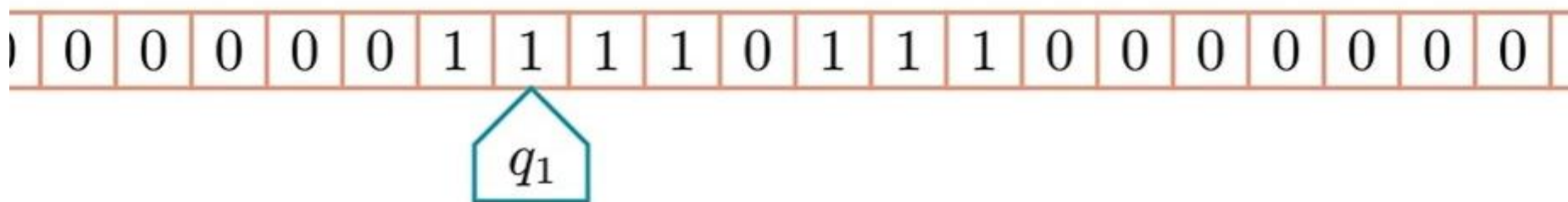


# 图灵机



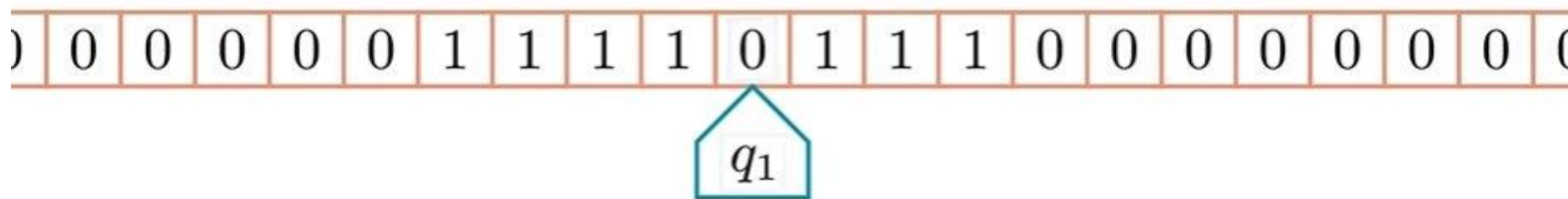
	0	1
$q_1$	1 $R$ $q_2$	1 $R$ $q_1$
$q_2$	0 $L$ $q_3$	1 $R$ $q_2$
$q_3$	0 $H$ $q_3$	0 $H$ $q_3$

# 图灵机



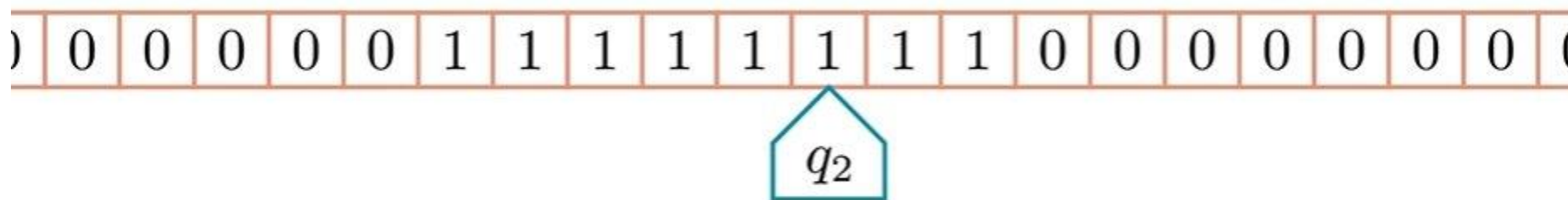
	0	1
$q_1$	1 $R$ $q_2$	1 $R$ $q_1$
$q_2$	0 $L$ $q_3$	1 $R$ $q_2$
$q_3$	0 $H$ $q_3$	0 $H$ $q_3$

# 图灵机



	0	1
$q_1$	1 $R$ $q_2$	1 $R$ $q_1$
$q_2$	0 $L$ $q_3$	1 $R$ $q_2$
$q_3$	0 $H$ $q_3$	0 $H$ $q_3$

# 图灵机



	0	1
$q_1$	1 $R$ $q_2$	1 $R$ $q_1$
$q_2$	0 $L$ $q_3$	1 $R$ $q_2$
$q_3$	0 $H$ $q_3$	0 $H$ $q_3$

# 图灵完备

---

- ▶ 图灵完备 (Turing Completeness) 是指一种数据操作规则，比如一种计算机编程语言，可以实现图灵机的所有功能，解决所有的可计算问题。

**定理 6.2 – 图灵完备 [Siegelmann et al., 1991]:** 所有的图灵机都可以被一个由使用 Sigmoid 型激活函数的神经元构成的全连接循环网络来进行模拟。

- ▶ 一个完全连接的循环神经网络可以近似解决所有的可计算问题。





应用到机器学习

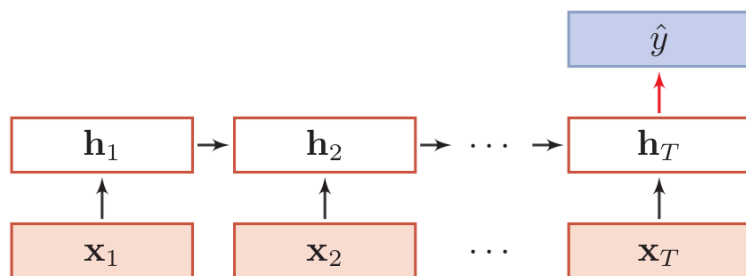
# 应用到机器学习

---

- ▶ 序列到类别模式
- ▶ 同步的序列到序列模式
- ▶ 异步的序列到序列模式

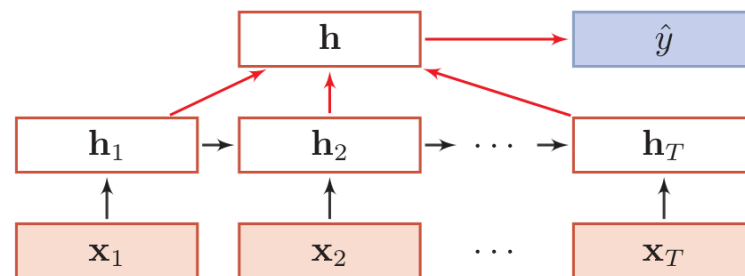
# 应用到机器学习

## ► 序列到类别



(a) 正常模式

$$\hat{y} = g(\mathbf{h}_T),$$

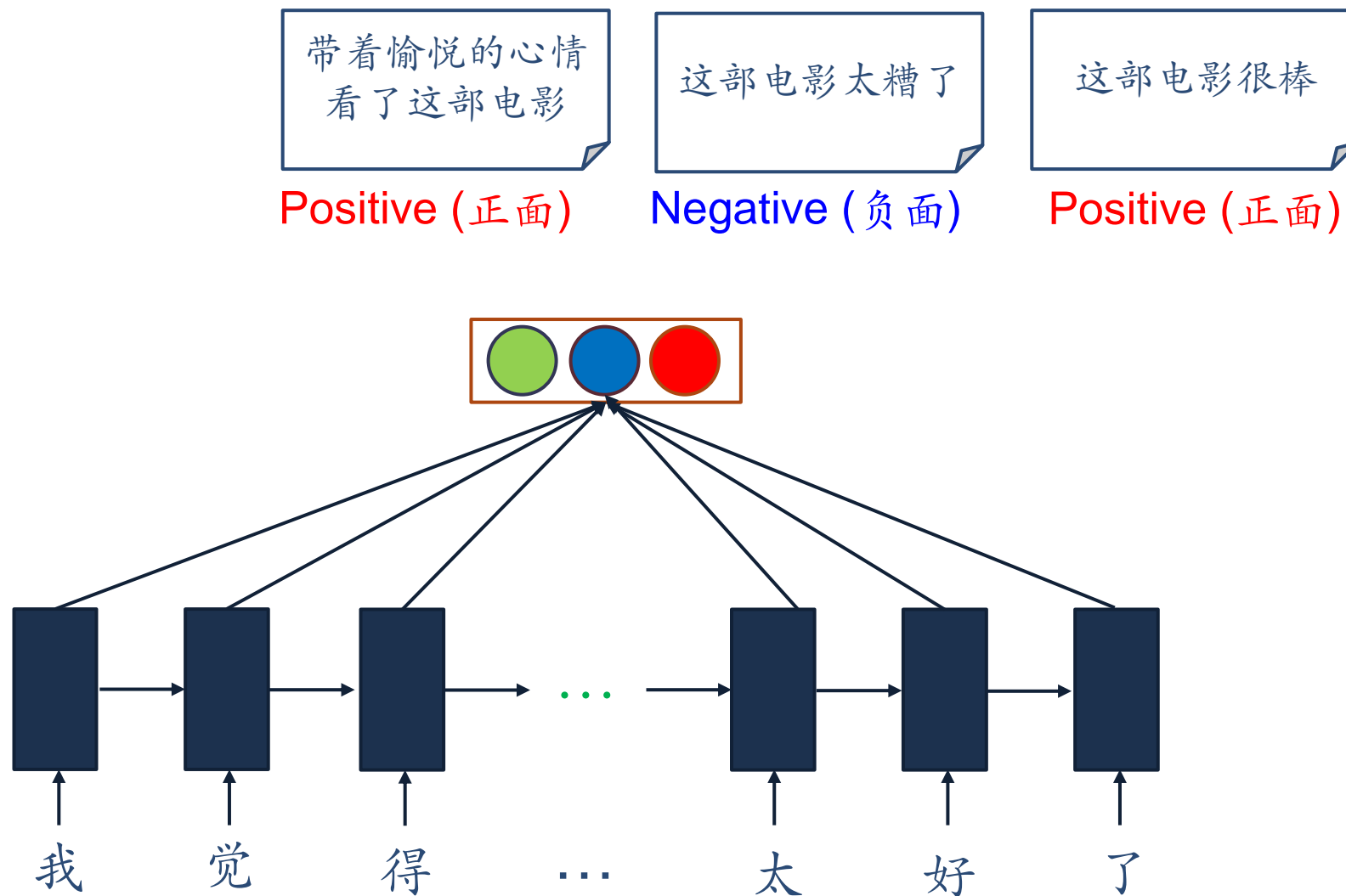


(b) 按时间进行平均采样模式

$$\hat{y} = g\left(\frac{1}{T} \sum_{t=1}^T \mathbf{h}_t\right).$$

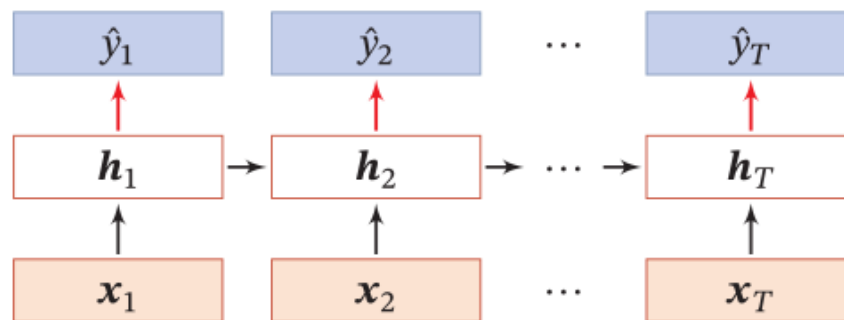
# 序列到类别

## ► 情感分类



# 应用到机器学习

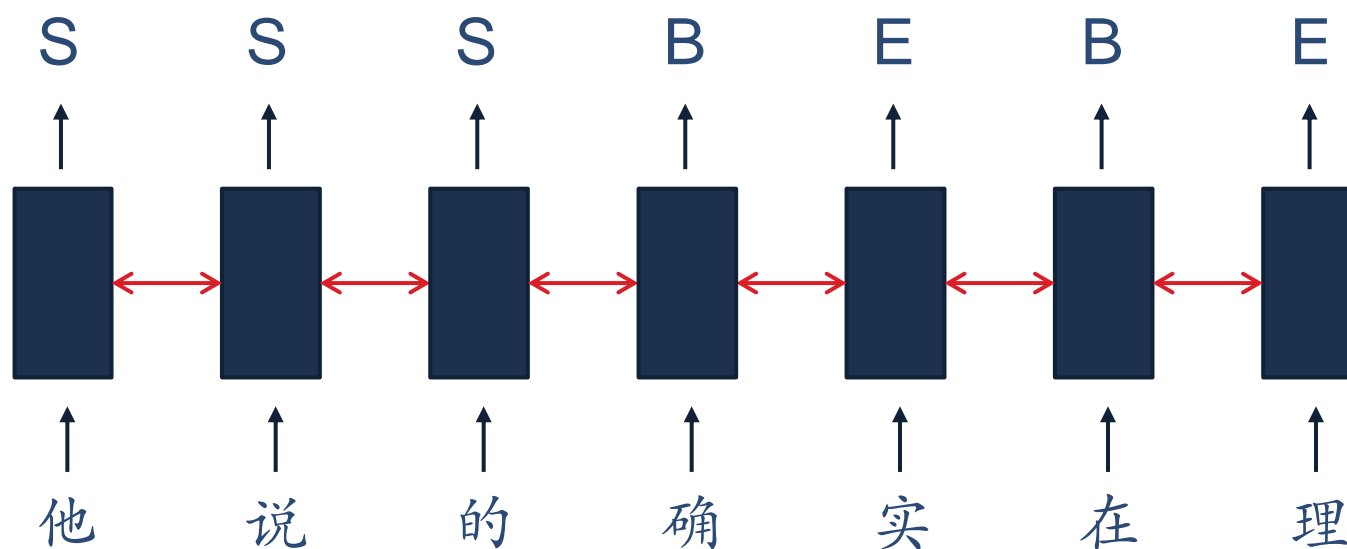
## ► 同步的序列到序列模式



$$\hat{y}_t = g(\mathbf{h}_t), \quad \forall t \in [1, T].$$

# 同步的序列到序列模式

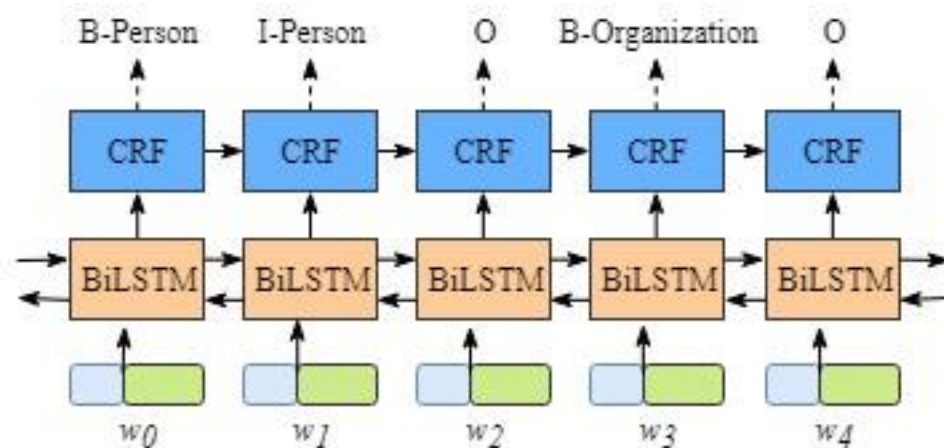
## ► 中文分词



# 同步的序列到序列模式

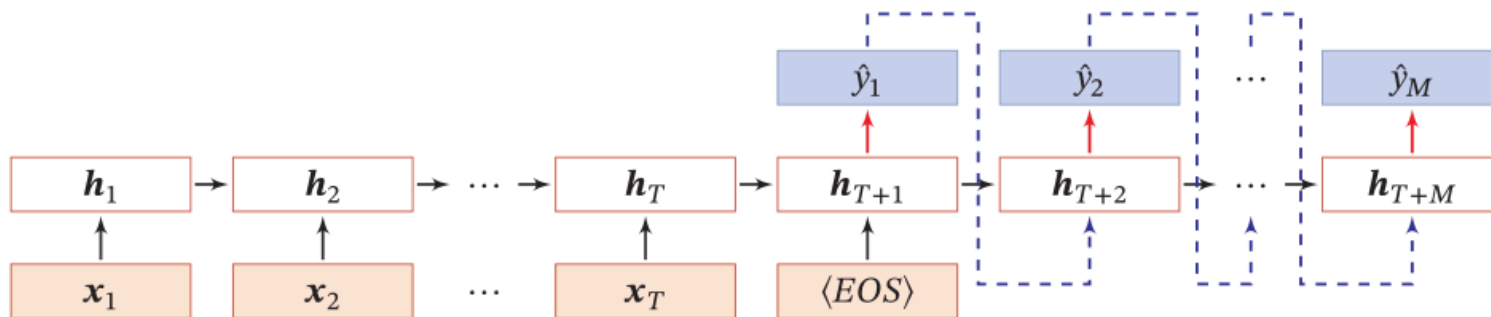
- ▶ 信息抽取(Information Extraction, IE)
- ▶ 从无结构的文本中抽取结构化的信息，形成知识

小米创始人雷军表示，该公司2015年营收达到780亿元人民币，较2014年的743亿元人民币增长了5%。



# 应用到机器学习

## ► 异步的序列到序列模式



$$\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \mathbf{x}_t), \quad \forall t \in [1, T]$$

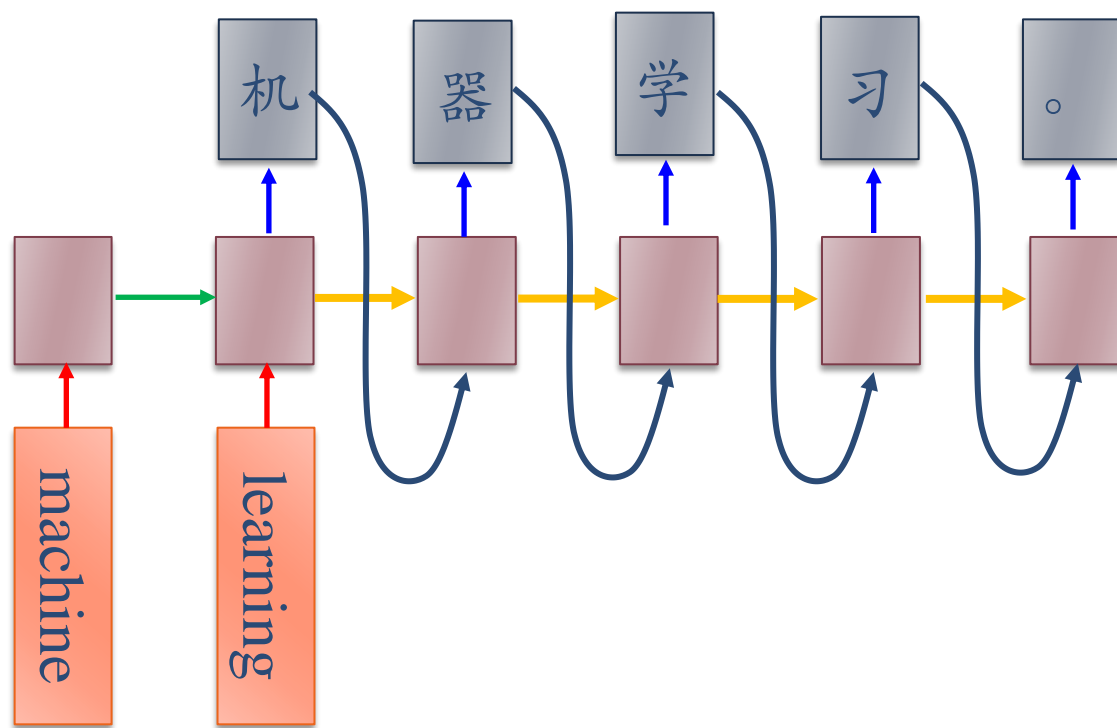
$$\mathbf{h}_{T+t} = f_2(\mathbf{h}_{T+t-1}, \hat{\mathbf{y}}_{t-1}), \quad \forall t \in [1, M]$$

$$\hat{\mathbf{y}}_t = g(\mathbf{h}_{T+t}), \quad \forall t \in [1, M]$$



# 异步的序列到序列模式

## ► 机器翻译



# 参数学习

## ▶ 机器学习

▶ 给定一个训练样本 $(x, y)$ ，其中

▶  $x = (x_1, \dots, x_T)$  为长度是  $T$  的输入序列，

▶  $y = (y_1, \dots, y_T)$  是长度为  $T$  的标签序列。

## ▶ 时刻 $t$ 的瞬时损失函数为

$$\mathcal{L}_t = \mathcal{L}(\mathbf{y}_t, g(\mathbf{h}_t)),$$

## ▶ 总损失函数

$$\mathcal{L} = \sum_{t=1}^T \mathcal{L}_t.$$

## ▶ 梯度

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \sum_{t=1}^T \frac{\partial \mathcal{L}_t}{\partial \mathbf{U}}, \quad \longleftarrow \quad \frac{\partial \mathcal{L}_t}{\partial u_{ij}} = \sum_{k=1}^t \frac{\partial^+ \mathbf{z}_k}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k},$$

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

$$\mathbf{y}_t = \mathbf{V}\mathbf{h}_t,$$

$$\mathbf{z}_k = \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_k + \mathbf{b}$$







# 梯度消失/爆炸

► 梯度

► 其中

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

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{x}_k^T,$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k}.$$

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

由于梯度爆炸或消失问题，实际上只能学习到短周期的依赖关系。这就是所谓的长程依赖问题。

# 梯度

---

## ▶ 反向传播的长程依赖问题

- ▶ 需要在内存中保留很多中间值
- ▶ 梯度消失或梯度爆炸

## ▶ 裁剪梯度以防止发散

$$\mathbf{g} \leftarrow \min \left( 1, \frac{\theta}{\|\mathbf{g}\|} \right) \mathbf{g}$$

## ▶ 重新缩放到最大尺寸为 $\theta$ 的梯度

# 长程依赖问题

---

▶ 循环神经网络在时间维度上非常深！

▶ 梯度消失或梯度爆炸

▶ 如何改进？

▶ 梯度爆炸问题

▶ 权重衰减

▶ 梯度截断

▶ 梯度消失问题

▶ 改进模型



# 长程依赖问题

---

## ► 改进方法

### ► 循环边改为线性依赖关系

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

### ► 增加非线性

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

# 长程依赖问题

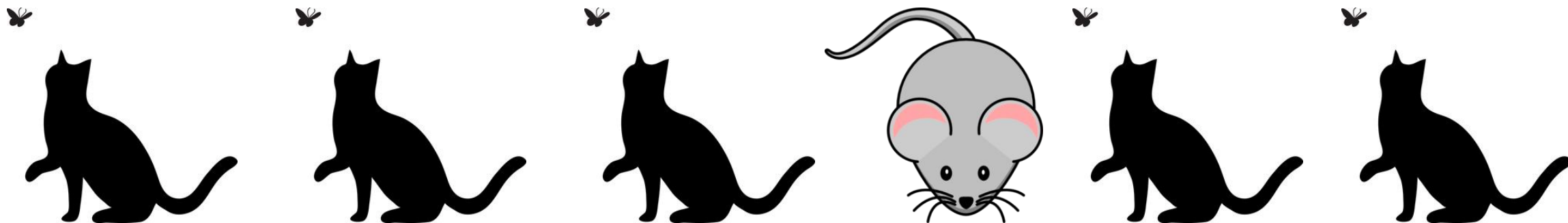
---

1. **长短期记忆网络 (LSTM)** : 通过引入门控机制, LSTM可以学习控制信息的流动, 从而更好地捕捉长期依赖关系。
2. **门控循环单元 (GRU)** : GRU是LSTM的一个变种, 它将LSTM中的遗忘门和输入门合并为一个更新门, 简化了模型结构。
3. **深度循环网络 (Deep RNN)** : 通过堆叠多个RNN层, 可以增加模型的容量, 帮助捕捉更高层次的依赖关系。
4. **注意力机制 (Attention Mechanism)** : 注意力机制允许模型在每个时间步动态地聚焦于序列中的不同部分, 从而更好地处理长期依赖。



# 长短期记忆神经网络 门控循环单元

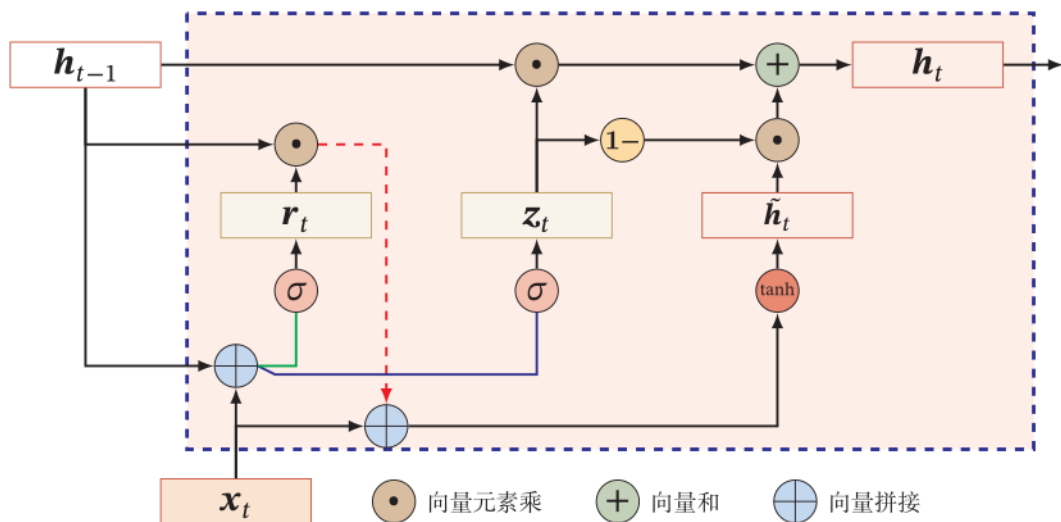
### ► 并非所有元素都具有同等意义



### ► 只记住相关的元素

- 需要注意的机制（更新门）
- 需要忘记的机制（重置门）

# 门控循环单元 GRU



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

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot g(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta),$$

重置门

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r),$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z),$$

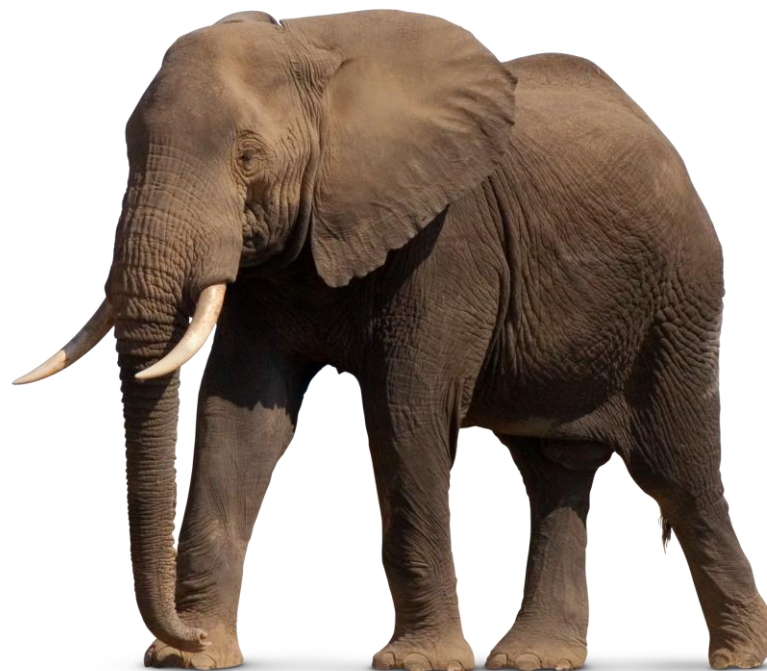
更新门

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

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

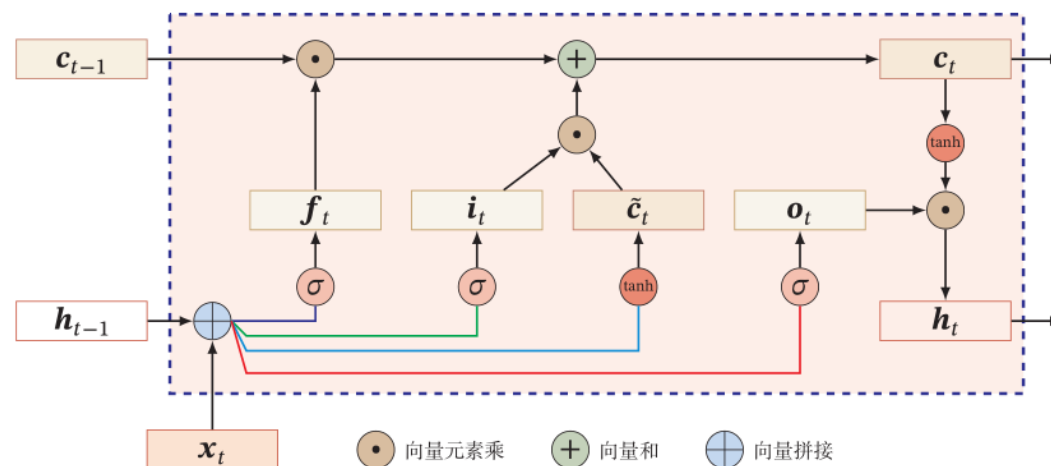
# 长短期记忆神经网络 (Long Short-Term Memory, LSTM )

---



# 长短期记忆神经网络 (Long Short-Term Memory, LSTM)

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



$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o),$$

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

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

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

# 长短期记忆 (LSTM)

---

- 遗忘门
  - 将每个值尽可能收缩为零
- 输入门
  - 决定是否应忽略输入数据
- 输出门
  - 决定隐含状态是否用于 LSTM 生成的输出
- 隐含状态
- 隐含状态和候选记忆



# LSTM的各种变体

---

## ▶ 没有遗忘门

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

## ▶ 耦合输入门和遗忘门

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

## ▶ peephole连接

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1} + \mathbf{b}_f),$$

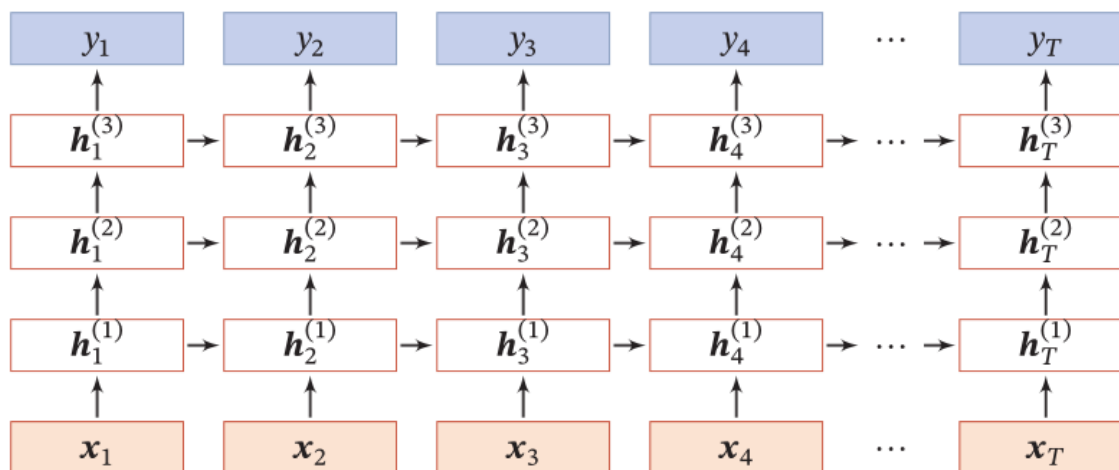
$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t + \mathbf{b}_o),$$



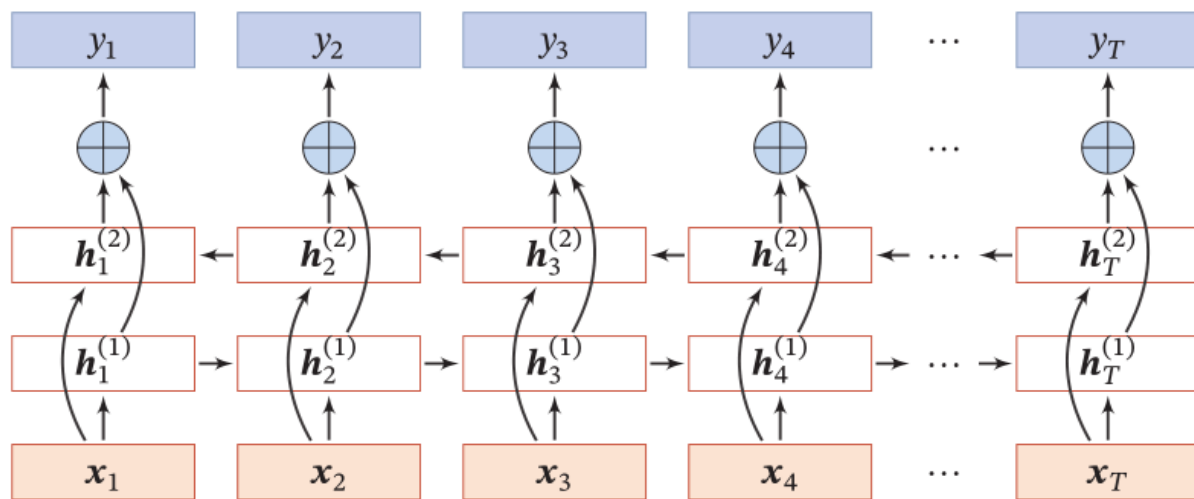
## 深层循环神经网络

# 堆叠循环神经网络

---



# 双向循环神经网络

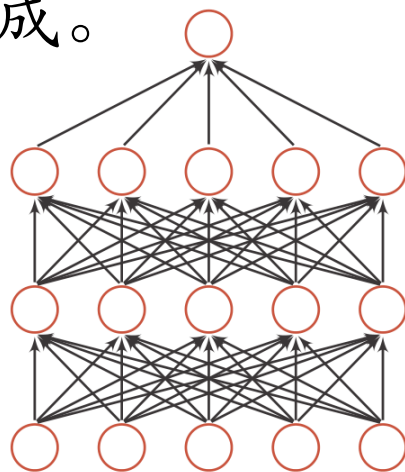


# 丢弃法 (Dropout Method)

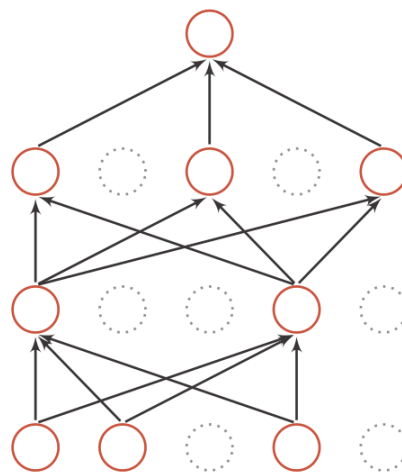
- ▶ 对于一个神经层  $y = f(Wx + b)$ ，引入一个丢弃函数  $d(\cdot)$  使得  $y = f(Wd(x) + b)$ 。

$$d(\mathbf{x}) = \begin{cases} \mathbf{m} \odot \mathbf{x} & \text{当训练阶段时} \\ p\mathbf{x} & \text{当测试阶段时} \end{cases}$$

- ▶ 其中  $m \in \{0,1\}^d$  是丢弃掩码 (dropout mask)，通过以概率为  $p$  的贝努力分布随机生成。



(a) 标准网络



(b) Dropout 后的网络

# Dropout意义

---

## ▶ 集成学习的解释

- ▶ 每做一次丢弃，相当于从原始的网络中采样得到一个子网络。如果一个神经网络有n个神经元，那么总共可以采样出 $2^n$ 个子网络。

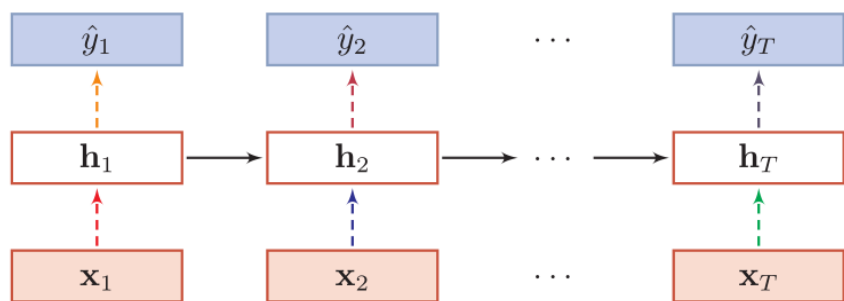
## ▶ 贝叶斯学习的解释

$$\begin{aligned}\mathbb{E}_{q(\theta)}[y] &= \int_{\theta} f(\mathbf{x}, \theta) q(\theta) d\theta \\ &\approx \frac{1}{M} \sum_{m=1}^M f(\mathbf{x}, \theta_m),\end{aligned}$$

- ▶ 其中 $f(\mathbf{x}, \theta_m)$ 为第m次应用丢弃方法后的网络。

# 循环神经网络上的丢弃法

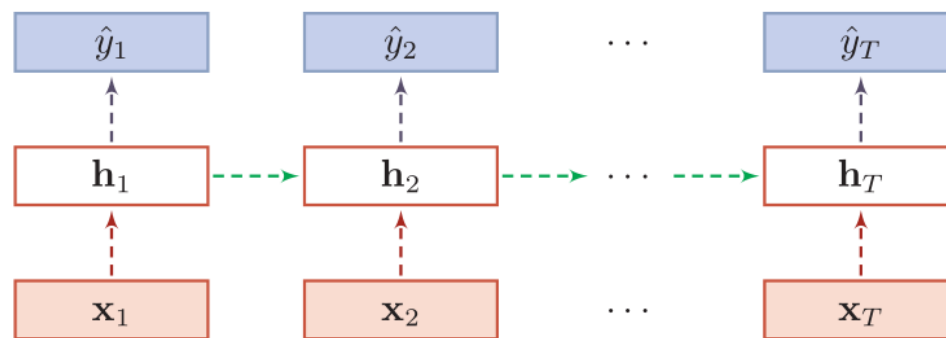
- 当在循环神经网络上应用丢弃法，不能直接对每个时刻的隐状态进行随机丢弃，这样会损害循环网络在时间维度上记忆能力。



虚线边表示进行随机丢弃，不同的颜色表示不同的丢弃掩码。

## 变分Dropout

- 根据贝叶斯学习的解释，丢弃法是一种对参数 $\theta$ 的采样。
- 每次采样的参数需要在每个时刻保持不变。因此，在对循环神经网络上使用丢弃法时，需要对参数矩阵的每个元素进行随机丢弃，并在所有时刻都使用相同的丢弃掩码。



相同颜色表示使用相同的丢弃掩码



## 扩展到图结构



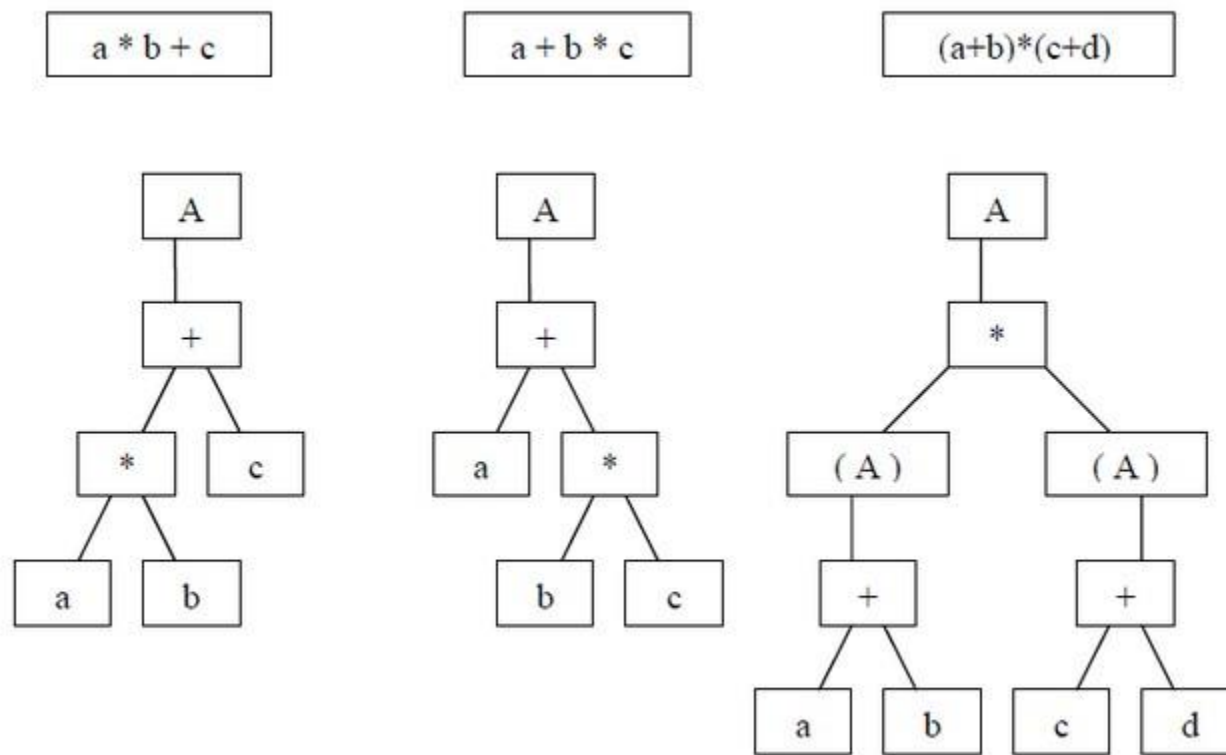
# 扩展到图结构

---



# 树结构

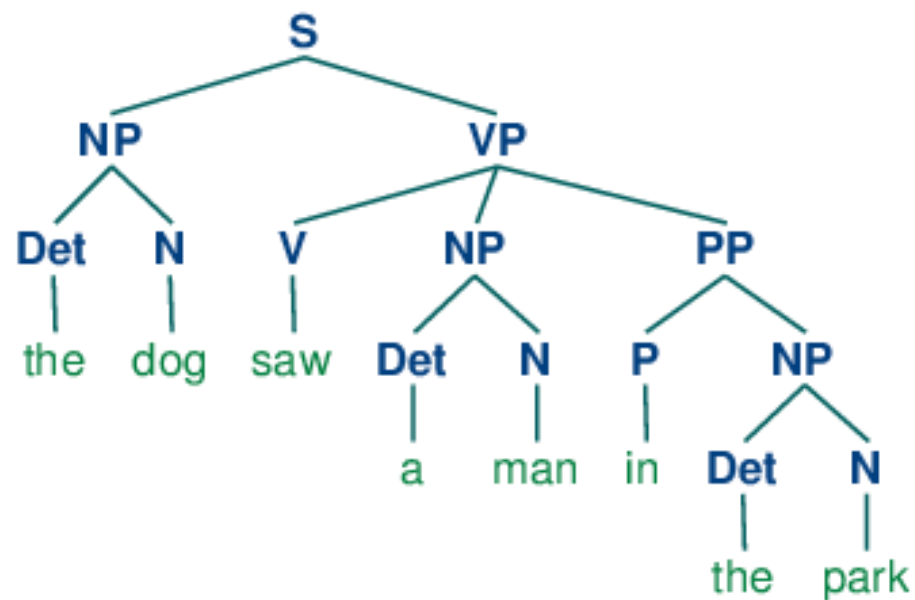
## ▶ 程序语言的句法结构



# 树结构

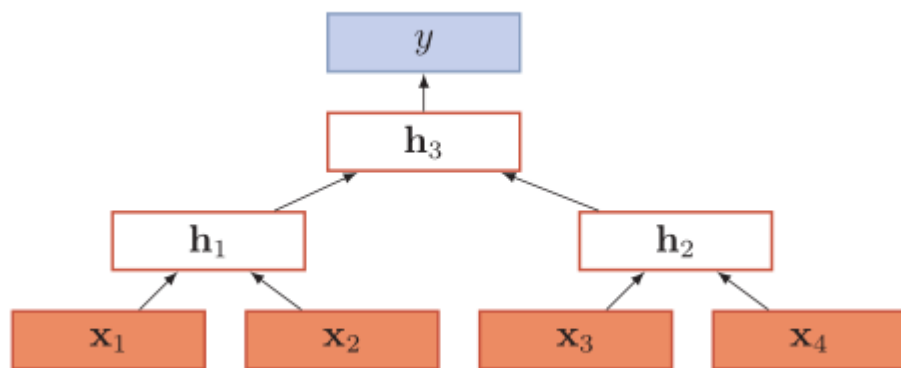
---

## ► 自然语言的句法结构



# 递归神经网络Recursive Neural Network

► 递归神经网络实在一个有向图无循环图上共享一个组合函数



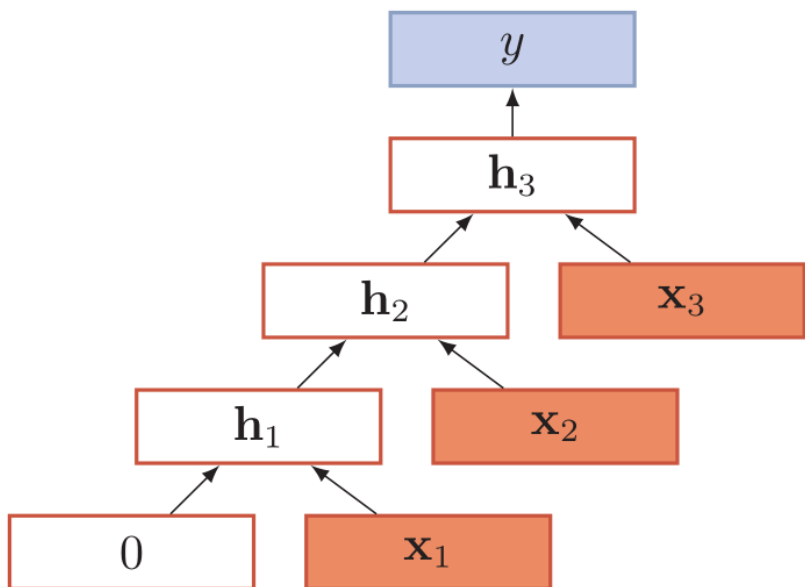
$$\mathbf{h}_i = f(\mathbf{h}_{\pi_i})$$

$$\begin{aligned}\mathbf{h}_1 &= f\left(W \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \mathbf{b}\right), \\ \mathbf{h}_2 &= f\left(W \begin{bmatrix} \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \mathbf{b}\right), \\ \mathbf{h}_3 &= f\left(W \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \mathbf{b}\right),\end{aligned}$$

# 递归神经网络

---

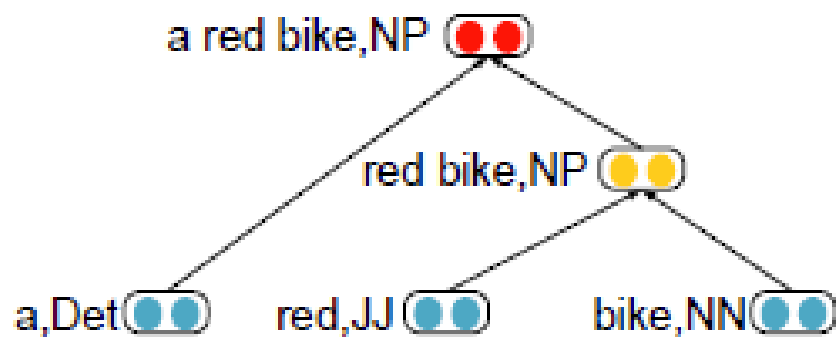
## ► 退化为循环神经网络



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

# 递归神经网络

给定一个语法树,

$$p_2 \rightarrow ap_1,$$
$$p_1 \rightarrow bc.$$


$$p_1 = f\left(W \begin{bmatrix} b \\ c \end{bmatrix}\right),$$

$$p_2 = f\left(W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

# 图网络

---

- ▶ 在实际应用中，很多数据是图结构的，比如知识图谱、社交网络、分子网络等。而前馈网络和循环网络很难处理图结构的数据。

# 图网络

---

► 对于一个任意的图结构  $G(V, E)$

► 更新函数

$$\mathbf{m}_t^{(v)} = \sum_{u \in N(v)} f(\mathbf{h}_{t-1}^{(v)}, \mathbf{h}_{t-1}^{(u)}, \mathbf{e}^{(u,v)})$$

$$\mathbf{h}_t^{(v)} = g(\mathbf{h}_{t-1}^{(v)}, \mathbf{m}_t^{(v)})$$

► 读出函数

$$\mathbf{y}_t = g(\{\mathbf{h}_T^{(v)} | v \in \mathcal{V}\})$$



# 图网络

<https://arxiv.org/pdf/1806.01261.pdf>

