序列模型

动手学深度学习 v2
李沐・AWS

序列数据

- 实际中很多数据是有时序结构的
- 电影的评价随时间变化而变化
  - 拿奖后评分上升，直到奖项被忘记
  - 看了很多好电影后，人们的期望变高
  - 季节性：贺岁片、暑期档
  - 导演、演员的负面报道导致评分变低

序列数据 - 更多例子

- 音乐、语言、文本、和视频都是连续的
  - 标题“狗咬人”远没有“人咬狗”那么令人惊讶
- 大地震发生后，很可能会有几次较小的余震
- 人的互动是连续的，从网上吵架可以看出
- 预测明天的股价要比填补昨天遗失的股价的更困难

统计工具

- 在时间 t 观察到 x_t，那么得到 T 个不独立的随机变量

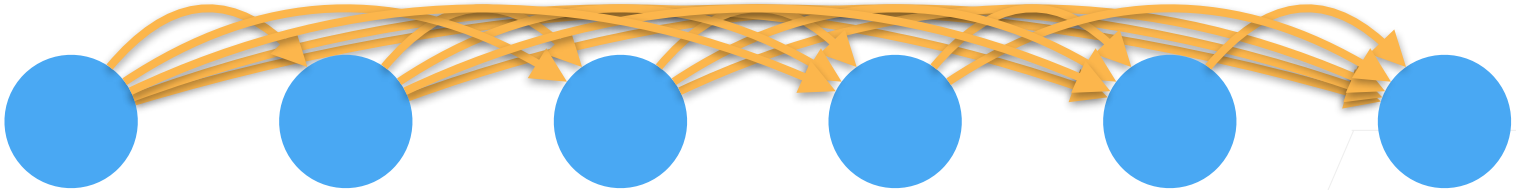
$$(x_1, \ldots x_T) \sim p(\mathbf{x})$$

- 使用条件概率展开

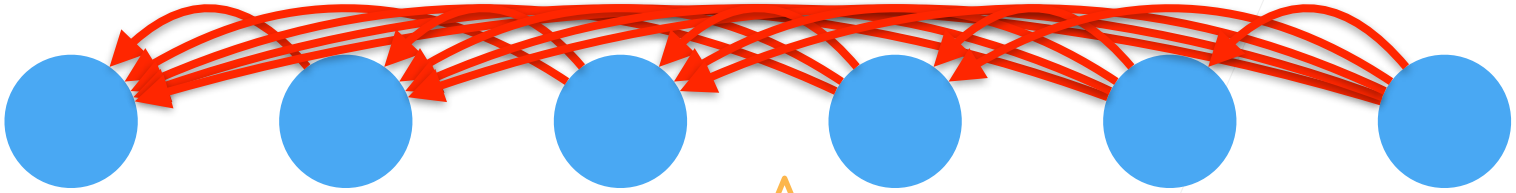
$$p(a, b) = p(a)p(b \mid a) = p(b)p(a \mid b)$$

统计工具

$$p(\mathbf{x}) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_1, \ldots x_{T-1})$$



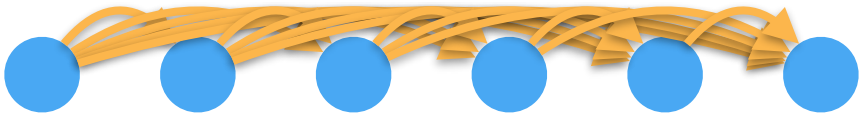
$$p(\mathbf{x}) = p(x_T) \cdot p(x_{T-1} \mid x_T) \cdot p(x_{T-2} \mid x_{T-1}, x_T) \cdot \ldots p(x_1 \mid x_2, \ldots x_T)$$



物理上不一定可行

序列模型

$$p(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \ldots p(x_T|x_1, \ldots x_{T-1})$$



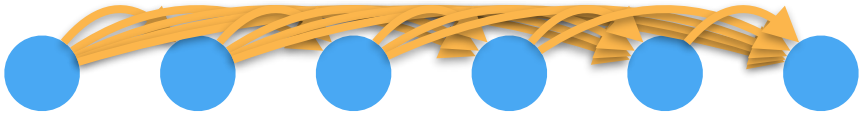
- 对条件概率建模

$$p(x_t|x_1, \ldots x_{t-1}) = p(x_t|f(x_1, \ldots x_{t-1}))$$

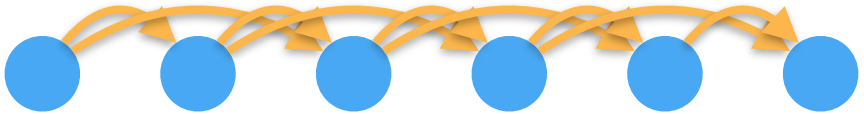
对见过的数据建模，也称自回归模型

方案 A - 马尔科夫假设

$$p(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \ldots p(x_T|x_1, \ldots x_{T-1})$$



- 假设当前当前数据只跟 τ 个过去数据点相关

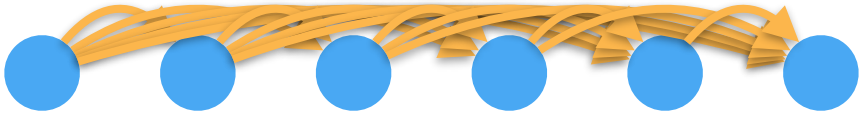


$$p(x_t|x_1, \ldots x_{t-1}) = p(x_t|x_{t-\tau}, \ldots x_{t-1}) = p(x_t|f(x_{t-\tau}, \ldots x_{t-1}))$$

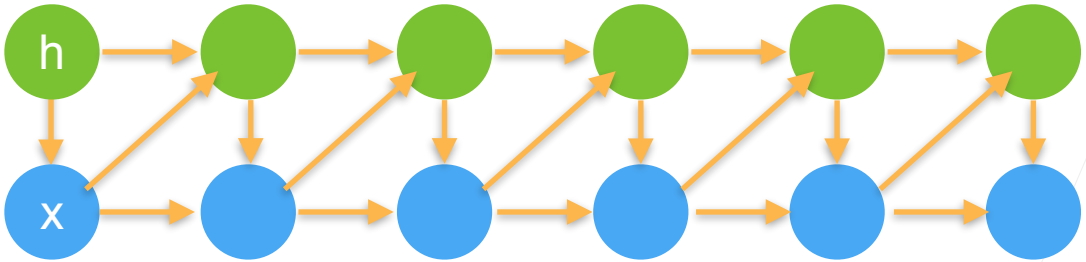
例如在过去数据上训练
一个MLP模型

方案 B - 潜变量模型

$$p(\mathbf{x}) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \ldots p(x_T|x_1, \ldots x_{T-1})$$



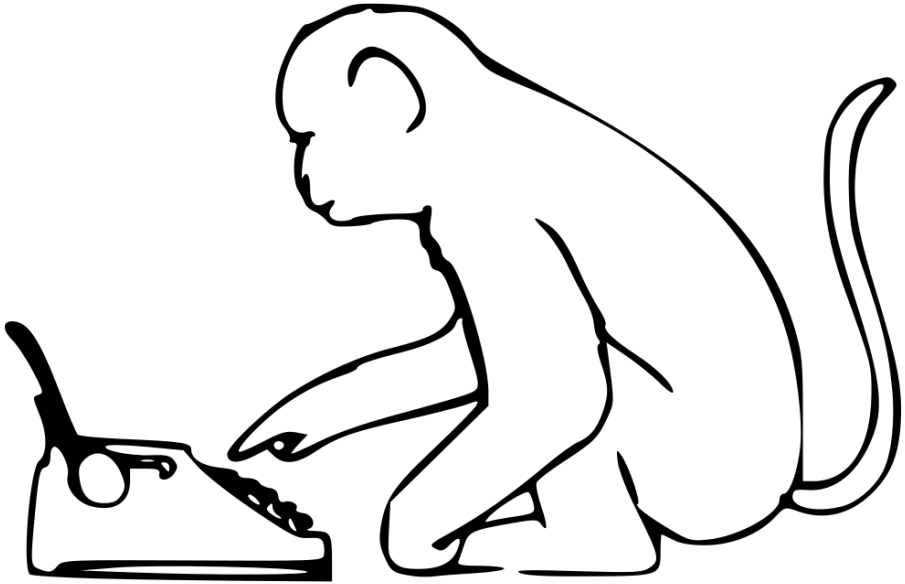
- 引人潜变量 h_t 来表示过去信息 $h_t = f(x_1, \ldots x_{t-1})$
  - 这样 $x_t = p(x_t|h_t)$



总结

- 时序模型中，当前数据跟之前观察到的数据相关
- 自回归模型使用自身过去数据来预测未来
- 马尔科夫模型假设当前只跟最近少数数据相关，从而简化模型
- 潜变量模型使用潜变量来概括历史信息

语言模型



语言模型

- 给定文本序列 $x_1, \ldots, x_T$，语言模型的目标是估计联合概率 $p(x_1, \ldots, x_T)$
- 它的应用包括
  - 做预训练模型（eg BERT，GPT-3）
  - 生成本文，给定前面几个词，不断的使用$x_t \sim p(x_t \mid x_1, \ldots, x_{t-1})$ 来生成后续文本
  - 判断多个序列中哪个更常见，e.g.“to recognize speech” vs“to wreck a nice beach

使用计数来建模

- 假设序列长度为2，我们预测

$$p(x, x') = p(x)p(x'|x) = \frac{n(x)}{n}\frac{n(x, x')}{n(x)}$$

  - 这里 n 是总词数，$n(x), n(x, x')$ 是单个单词和连续单词对的出现次数

- 很容易拓展到长为3的情况

$$p(x, x', x'') = p(x)p(x'|x)p(x''|x, x') = \frac{n(x)}{n}\frac{n(x, x')}{n(x)}\frac{n(x, x', x'')}{n(x, x')}$$

N元语法

- 当序列很长时，因为文本量不够大，很可能 $n(x_1, \ldots, x_T) \leq 1$
- 使用马尔科夫假设可以缓解这个问题
  - 一元语法：
$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3)p(x_4)$$
$$= \frac{n(x_1)}{n}\frac{n(x_2)}{n}\frac{n(x_3)}{n}\frac{n(x_4)}{n}$$
  - 二元语法：
$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)$$
$$= \frac{n(x_1)}{n}\frac{n(x_1, x_2)}{n(x_1)}\frac{n(x_2, x_3)}{n(x_2)}\frac{n(x_3, x_4)}{n(x_3)}$$
  - 三元语法：$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_2, x_3)$

总结

- 语言模型估计文本序列的联合概率
- 使用统计方法时常采用n元语法

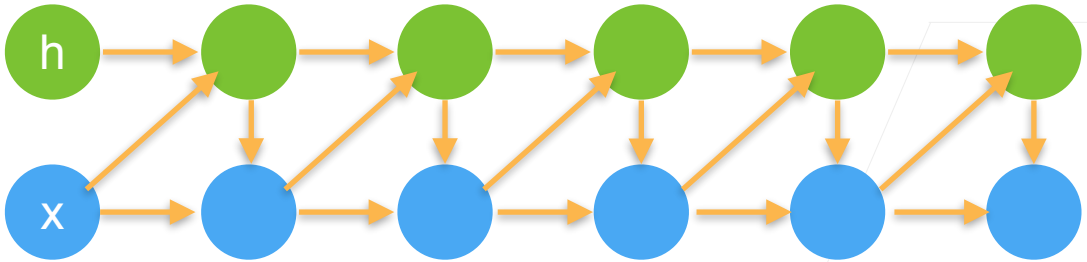
循环神经网络

潜变量自回归模型

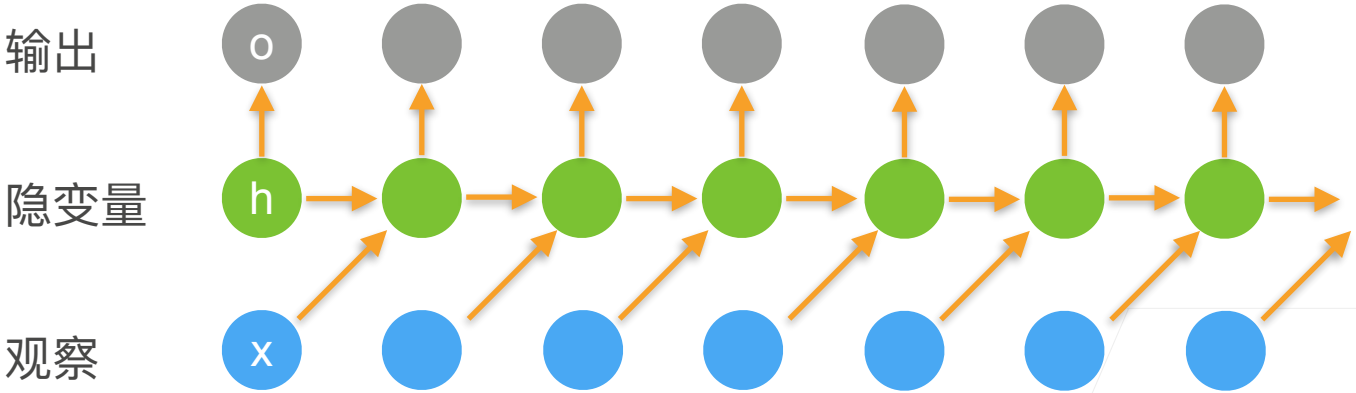
- 使用潜变量 h_t 总结过去信息

$$p(h_t \mid h_{t-1}, x_{t-1})$$

$$p(x_t \mid h_t, x_{t-1})$$



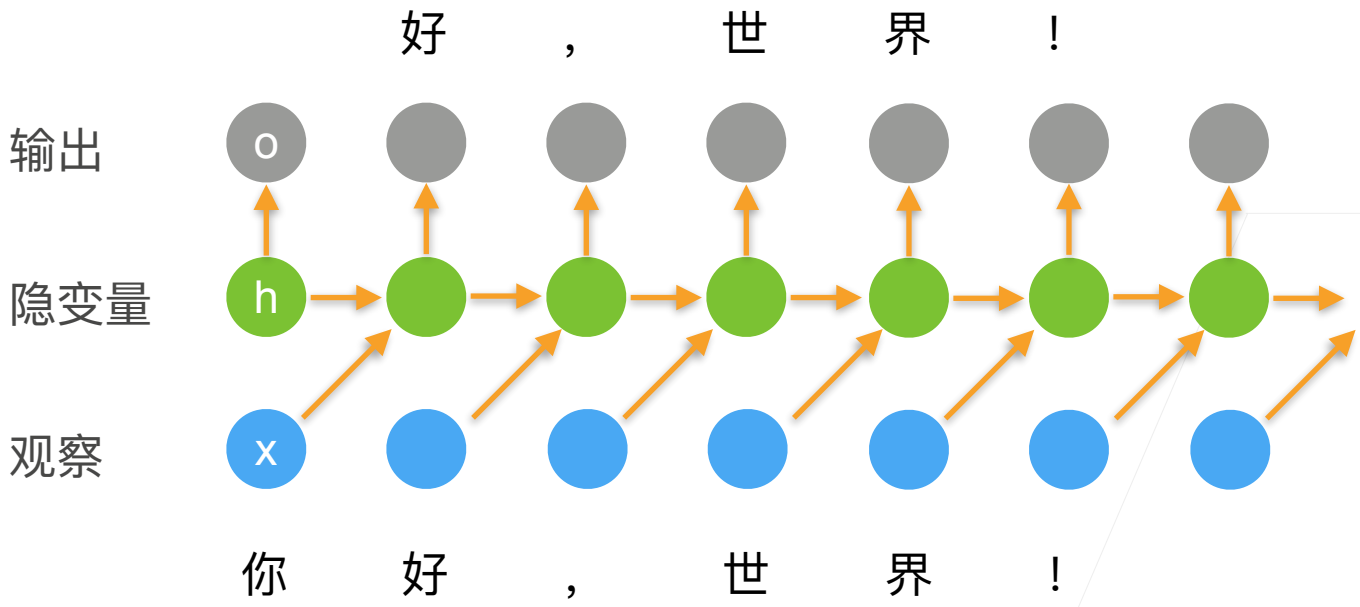
循环神经网络



- 更新隐藏状态： $\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$
- 输出： $\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$

去掉这一项就退化成MLP

使用循环神经网络的语言模型



困惑度（perplexity）

- 衡量一个语言模型的好坏可以用平均交叉熵

$$\pi = \frac{1}{n}\sum_{i=1}^{n} -\log p(x_t \mid x_{t-1}, \ldots)$$

p 是语言模型的预测概率，x_t 是真实词

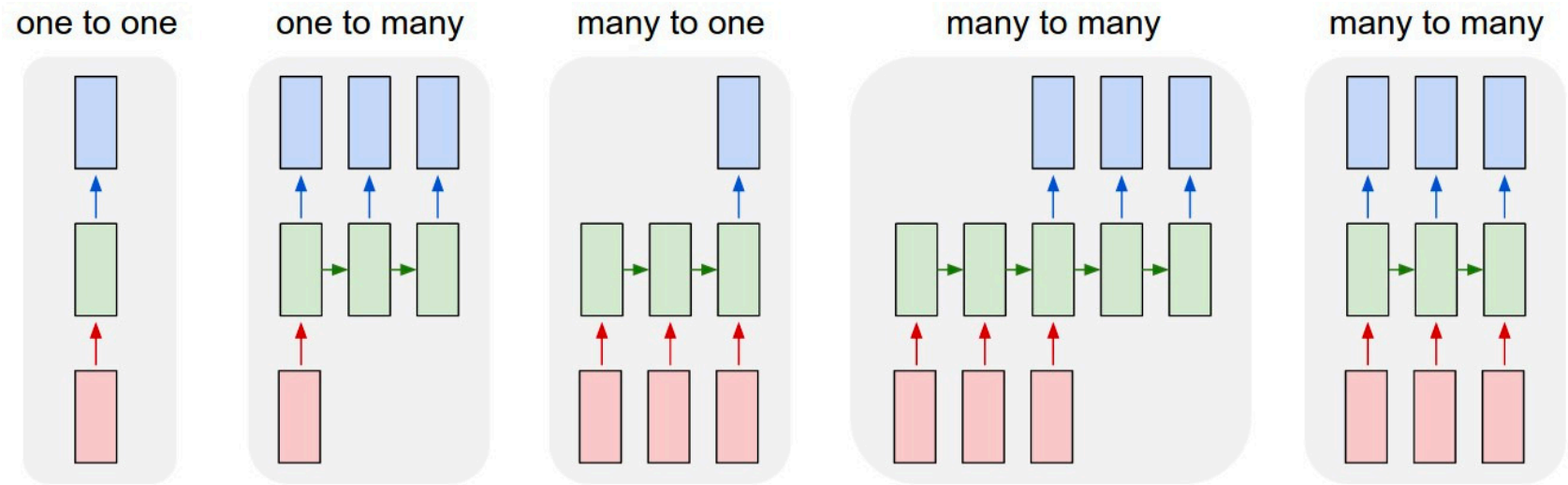
- 历史原因NLP使用困惑度 $\exp(\pi)$ 来衡量，是平均每次可能选项
  - 1表示完美，无穷大是最差情况

梯度裁剪

- 迭代中计算这T个时间步上的梯度，在反向传播过程中产生长度为$O(T)$的矩阵乘法链，导致数值不稳定
- 梯度裁剪能有效预防梯度爆炸
  - 如果梯度长度超过θ，那么拖影回长度θ

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

更多的应用 RNNs



文本生成　文本分类　问答、机器翻译　Tag生成

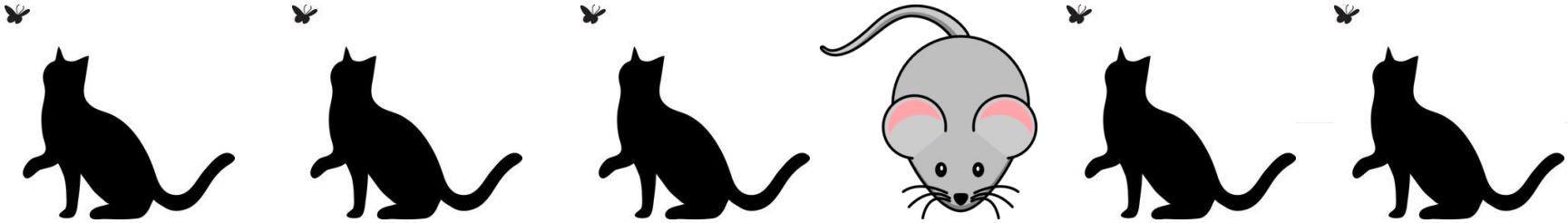
总结

- 循环神经网络的输出取决于当下输入和前一时间的隐变量
- 应用到语言模型中时，循环神经网络根据当前词预测下一次时刻词
- 通常使用困惑度来衡量语言模型的好坏

门控循环单元 GRU

关注一个序列

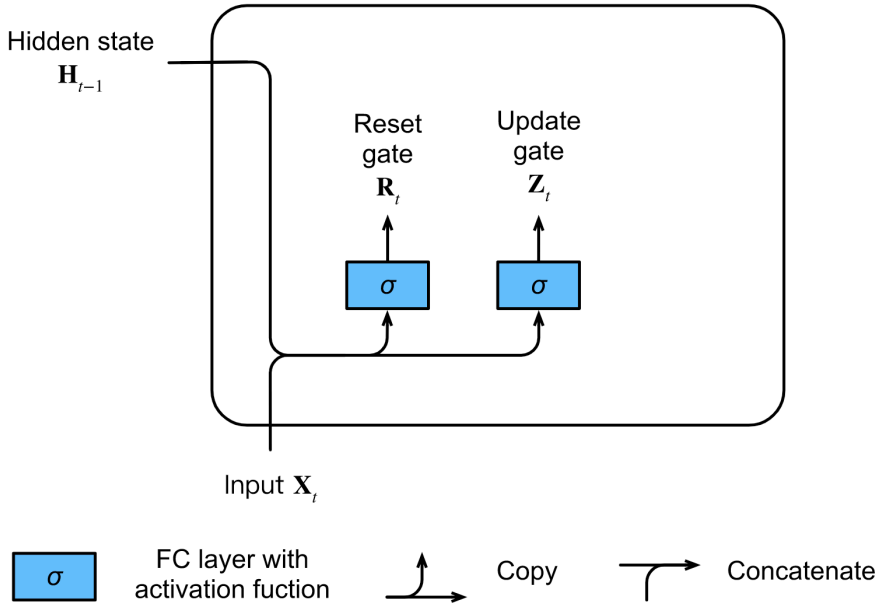
- 不是每个观察值都是同等重要



- 想只记住相关的观察需要：
  - 能关注的机制（更新门）
  - 能遗忘的机制（重置门）

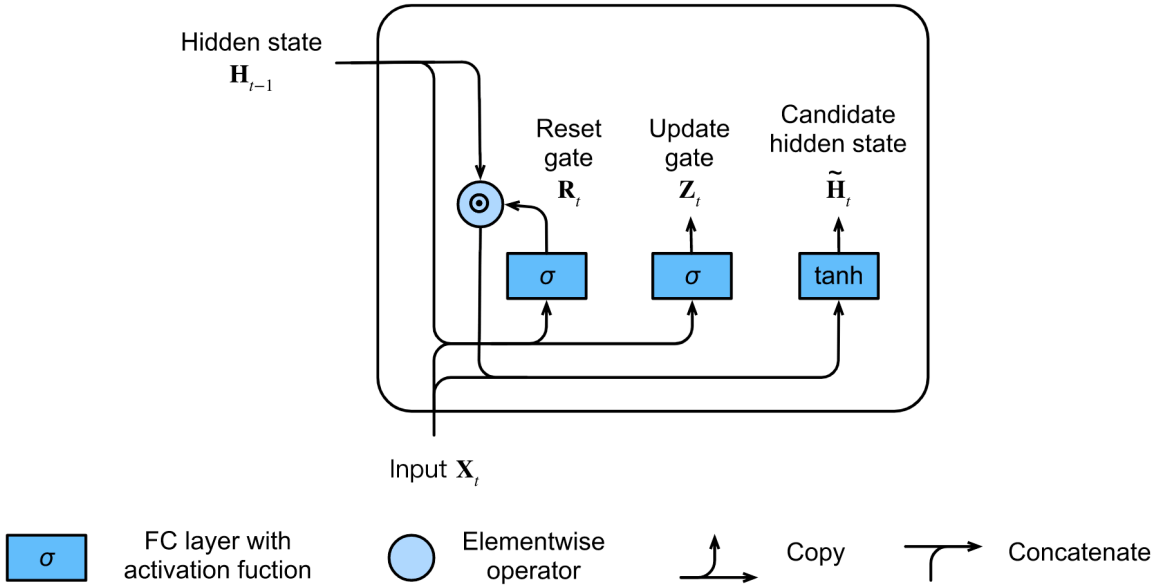
门

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$



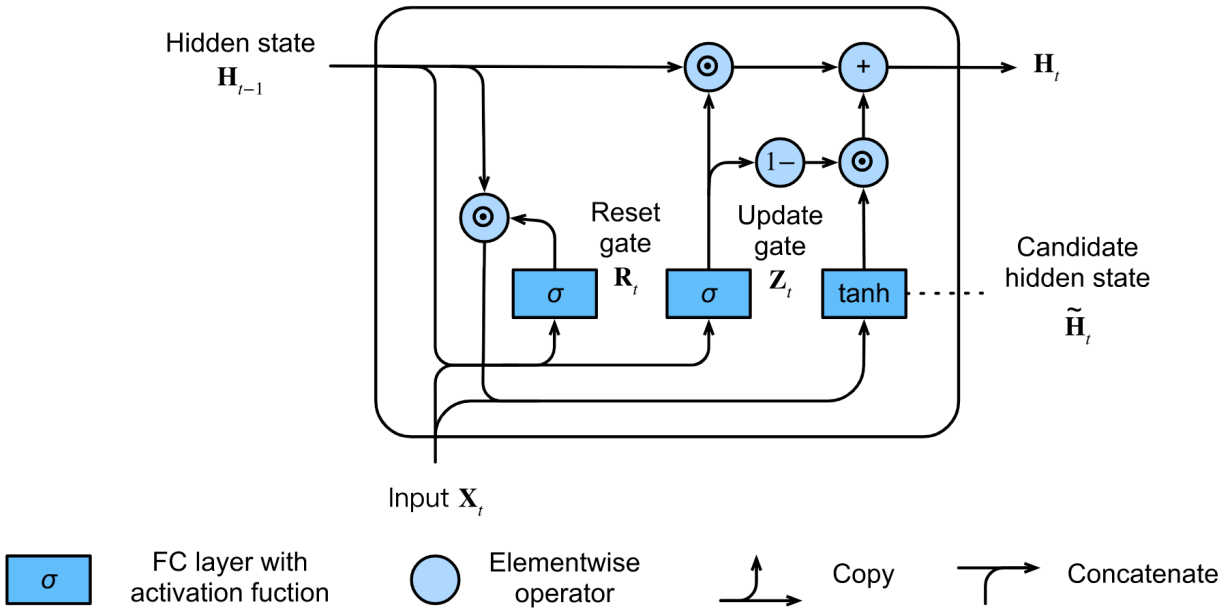
候选隐状态

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$



隐状态

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



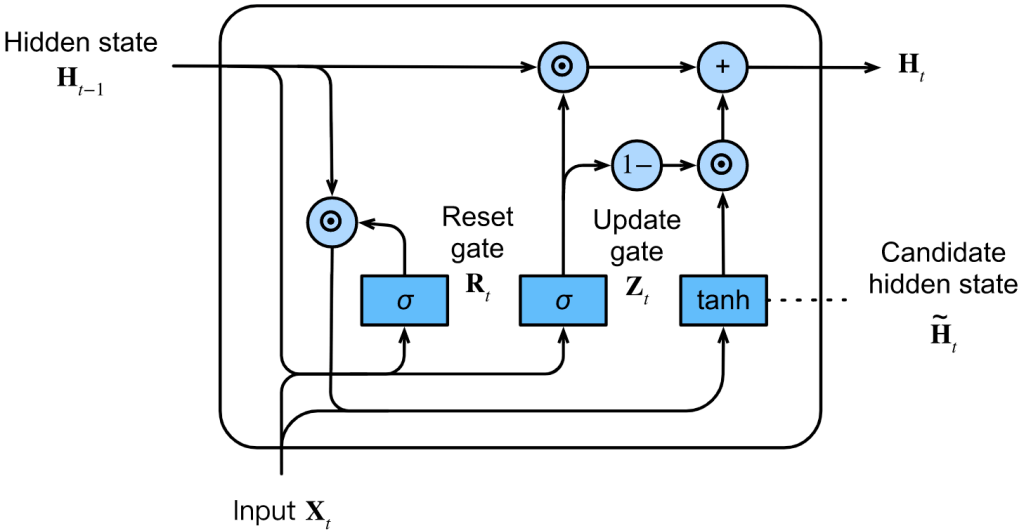
总结

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$

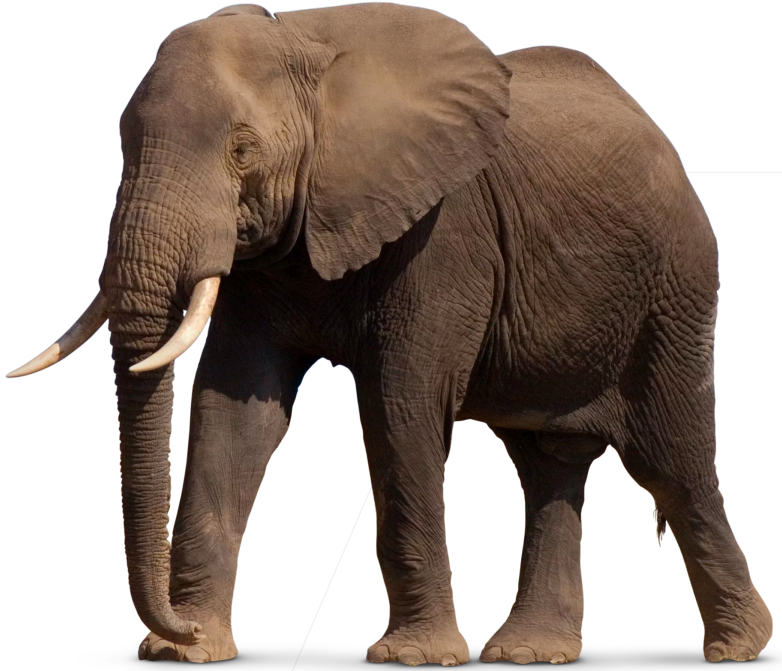
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



长短期记忆网络（LSTM）



长短期记忆网络

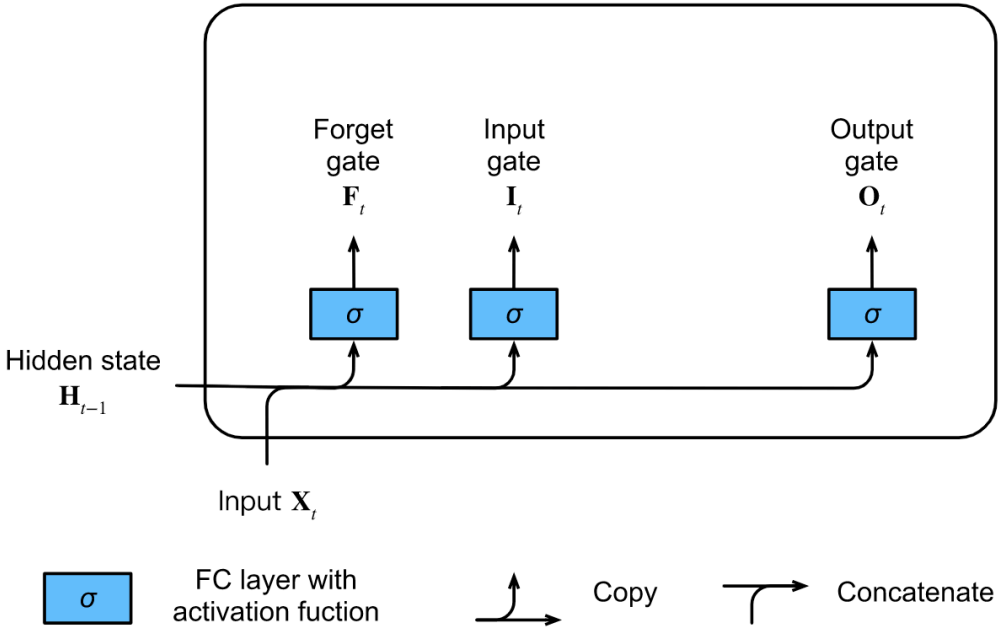
- 忘记门：将值朝0减少
- 输入门：决定不是忽略掉输入数据
- 输出门：决定是不是使用隐状态

门

$$\boldsymbol{I}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i)$$

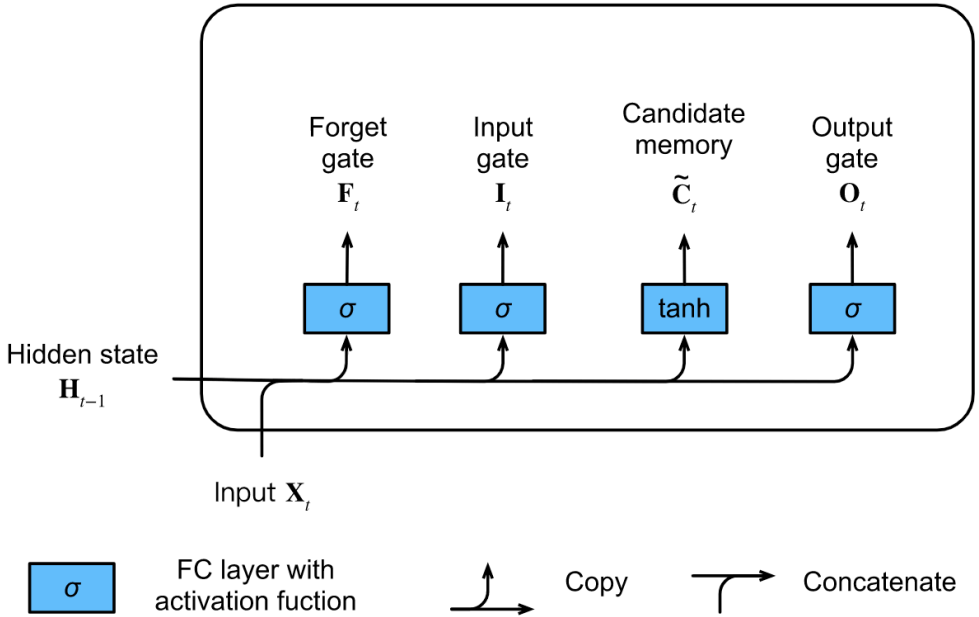
$$\boldsymbol{F}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$

$$\boldsymbol{O}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o)$$



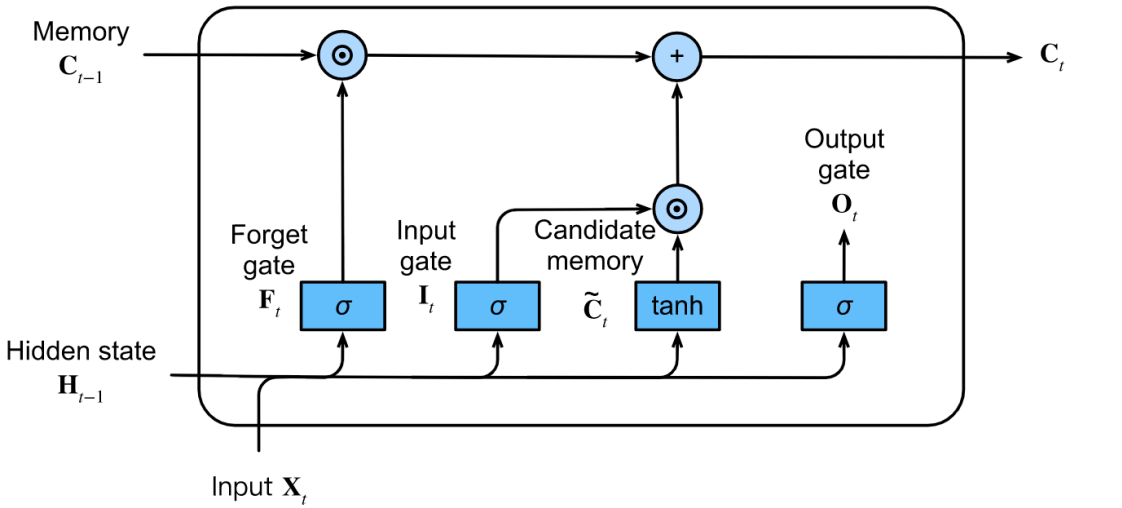
候选记忆单元

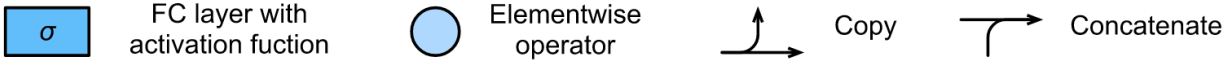
$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)$$



记忆单元

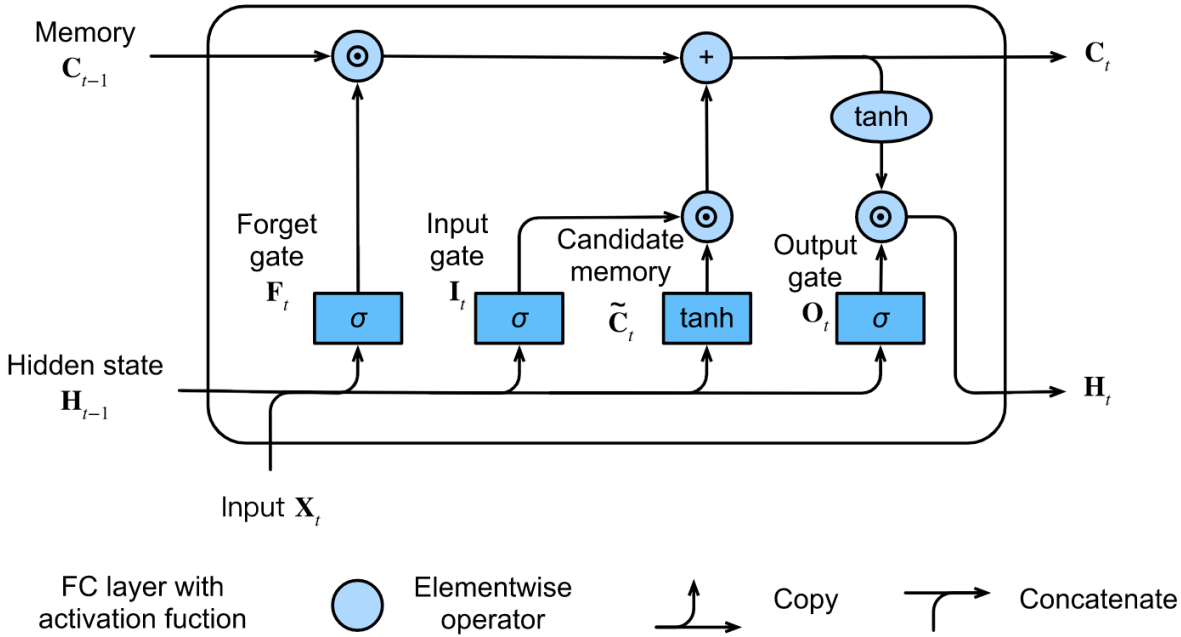
$$\boldsymbol{C}_t = \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t$$





隐状态

$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot \tanh(\boldsymbol{C}_t)$$



总结

$$\boldsymbol{I}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i)$$

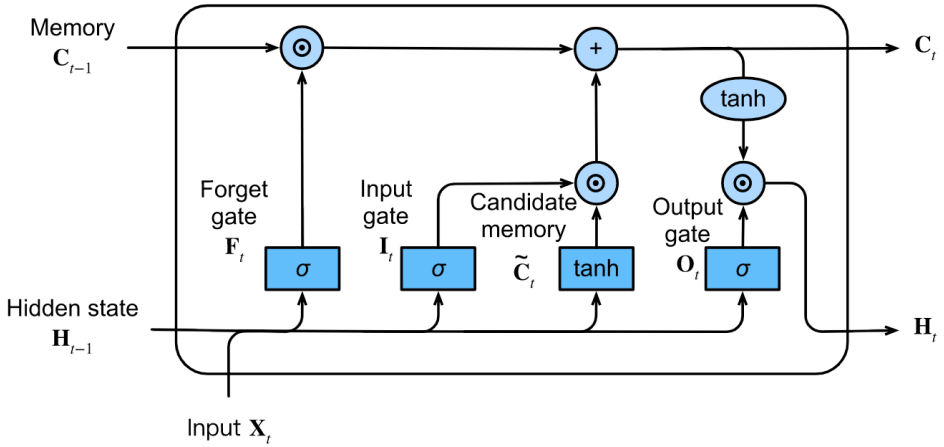
$$\boldsymbol{F}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$

$$\boldsymbol{O}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o)$$

$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)$$

$$\boldsymbol{C}_t = \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t$$

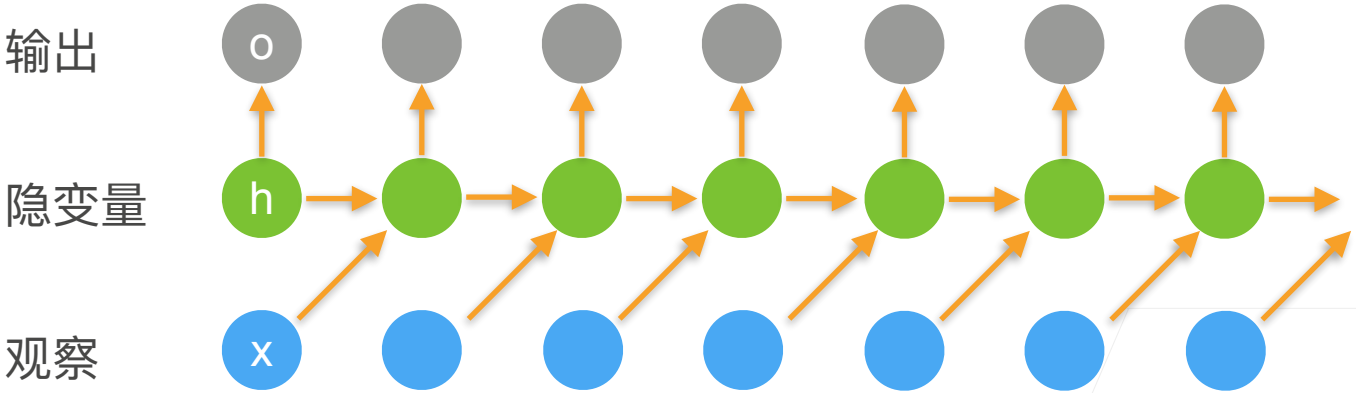
$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot \tanh(\boldsymbol{C}_t)$$



深度循环神经网络

李沐 · AWS

回顾：循环神经网络

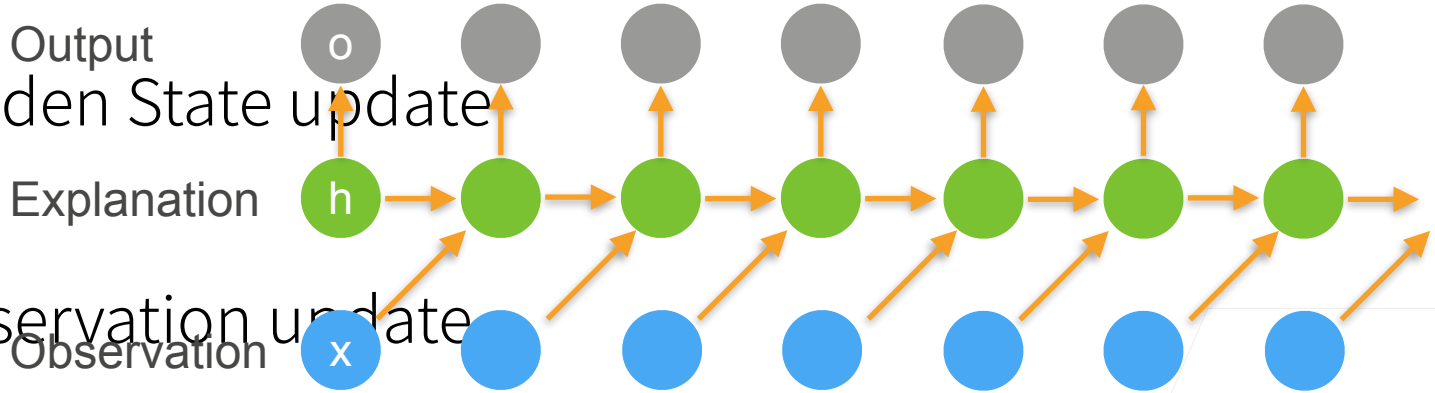


- 更新隐藏状态： $\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$
- 输出： $\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$

如何得到更多的非线性？

Plan A - Nonlinearity in the units

- Hidden State update
- Observation update



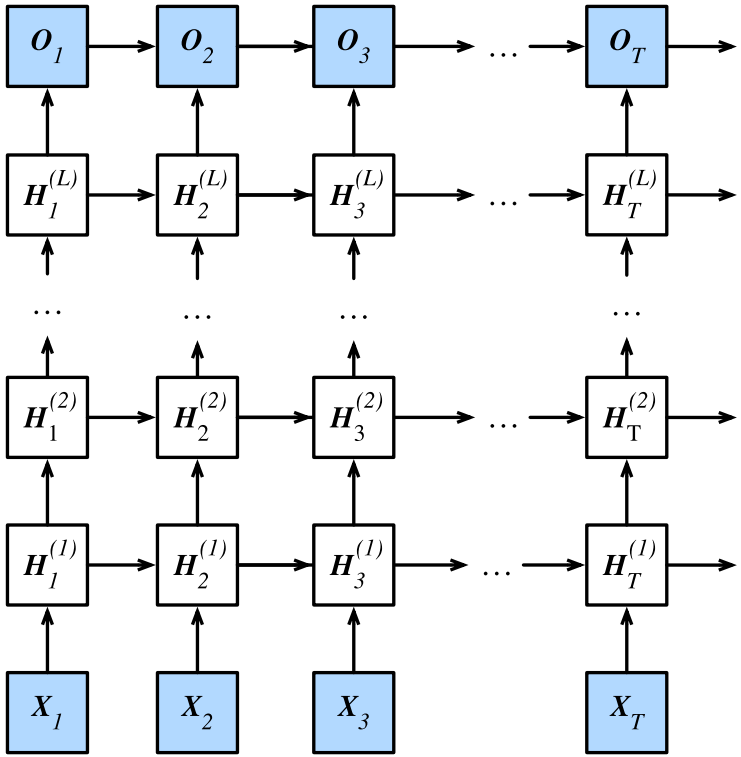
$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$$

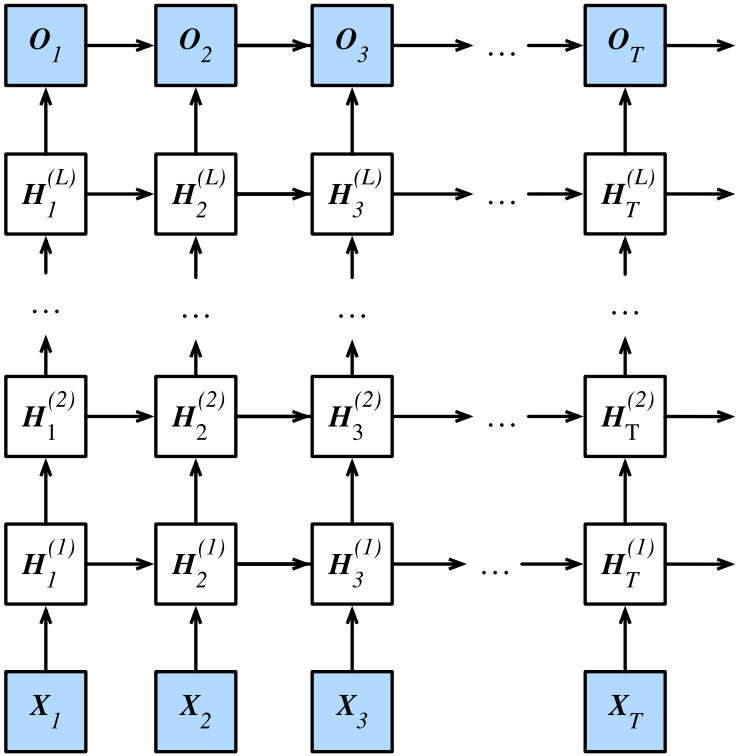
Replace with MLP?

更深

- 浅 RNN
  - 输入
  - 隐层
  - 输出
- 深 RNN
  - 输入
  - 隐层
  - 隐层
  - …
  - 输出



更深



$$\mathbf{H}_t = f(\mathbf{H}_{t-1}, \mathbf{X}_t)$$
$$\mathbf{O}_t = g(\mathbf{H}_t)$$

.

$$\mathbf{H}_t^1 = f_1(\mathbf{H}_{t-1}^1, \mathbf{X}_t)$$
$$\mathbf{H}_t^j = f_j(\mathbf{H}_{t-1}^j, \mathbf{H}_t^{j-1})$$
$$\mathbf{O}_t = g(\mathbf{H}_t^L)$$

总结

- 深度循环神经网络使用多个隐藏层来获得更多的非线性性

双向循环神经网络


JOHN COLTRANE BOTH
DIRECTIONS AT ONCE
THE LOST ALBUM
impulse!
JOHN COLTRANE BOTH
DIRECTIONS AT ONCE
THE LOST ALBUM
impulse!

未来很重要

```
I am _____
I am _____ very hungry,
I am _____ very hungry, I could eat half a pig.
```

未来很重要

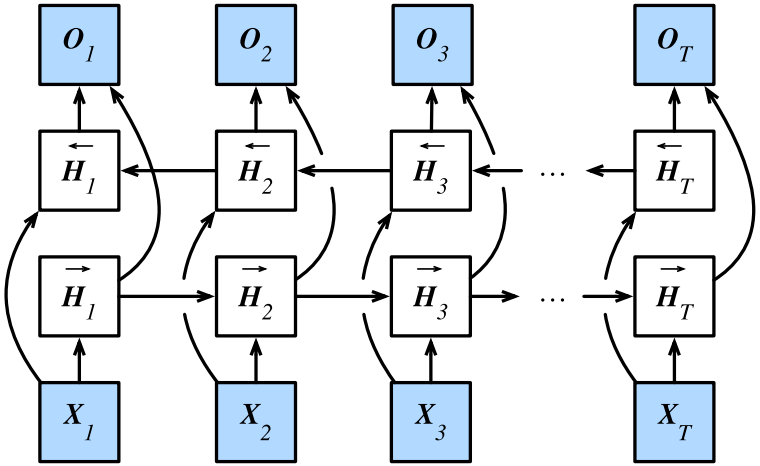
```
I am happy.
I am not   very hungry,
I am very  very hungry, I could eat half a pig.
```

未来很重要

```
I am happy.
I am not   very hungry,
I am very  very hungry, I could eat half a pig.
```

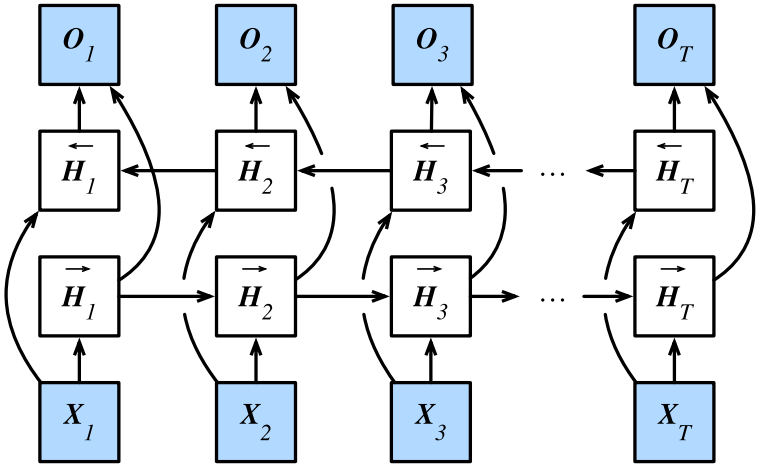
- 取决于过去和未来的上下文，可以填很不一样的词
- 目前为止RNN只看过去
- 在填空的时候，我们也可以看未来

双向 RNN



- 一个前向RNN隐层
- 一个方向RNN隐层
- 合并两个隐状态得到输出

双向 RNN



$$\overrightarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(f)} + \overrightarrow{\mathbf{H}}_{t-1} \mathbf{W}_{hh}^{(f)} + \mathbf{b}_h^{(f)}),$$

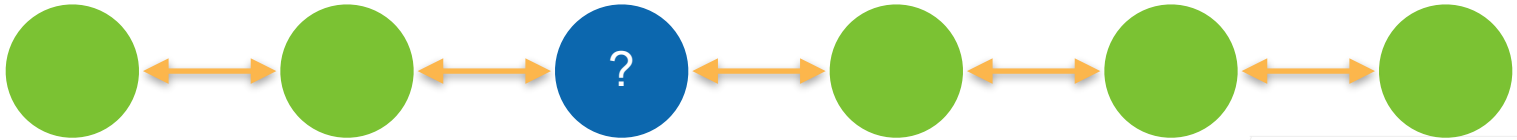
$$\overleftarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(b)} + \overleftarrow{\mathbf{H}}_{t+1} \mathbf{W}_{hh}^{(b)} + \mathbf{b}_h^{(b)}),$$

$$\mathbf{H}_t = [\overrightarrow{\mathbf{H}}_t, \overleftarrow{\mathbf{H}}_t]$$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

推理

- 训练：



- 推理：

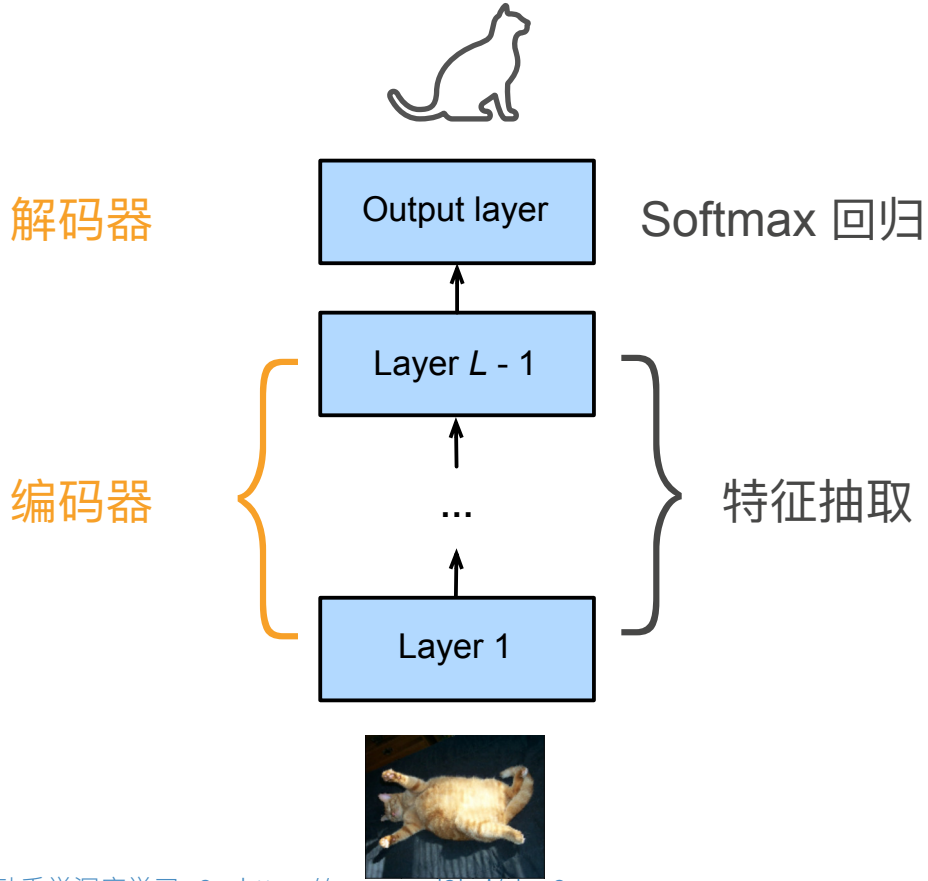
总结

- 双向循环神经网络通过反向更新的隐藏层来利用方向时间信息
- 通常用来对序列抽取特征、填空，而不是预测未来

编码器-解码器

动手学深度学习 v2
李沐・AWS

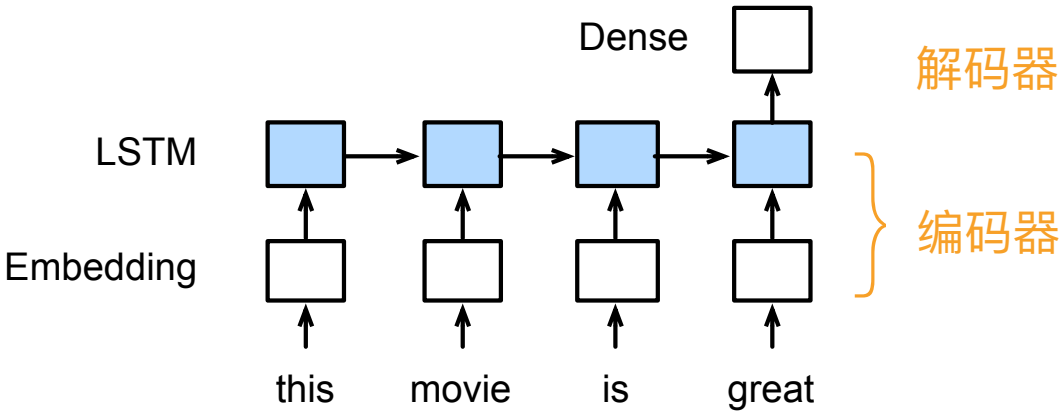
重新考察CNN



- 编码器：将输入编程成中间表达形式（特征）
- 解码器：将中间表示解码成输出

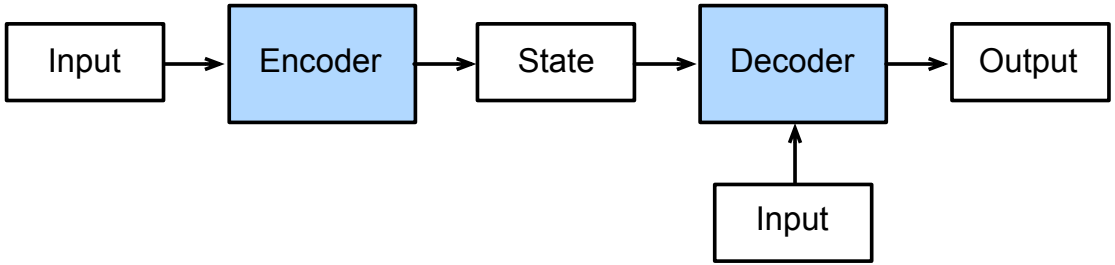
重新考察RNN

- 编码器：将文本表示成向量
- 解码器：向量表示成输出



编码器-解码器架构

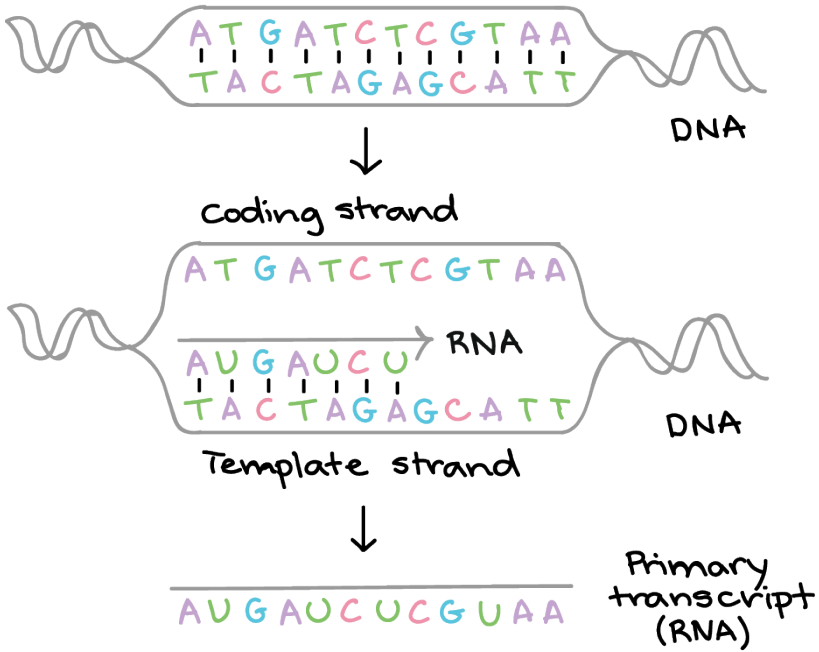
- 一个模型被分为两块：
  - 编码器处理输出
  - 解码器生成输出



总结

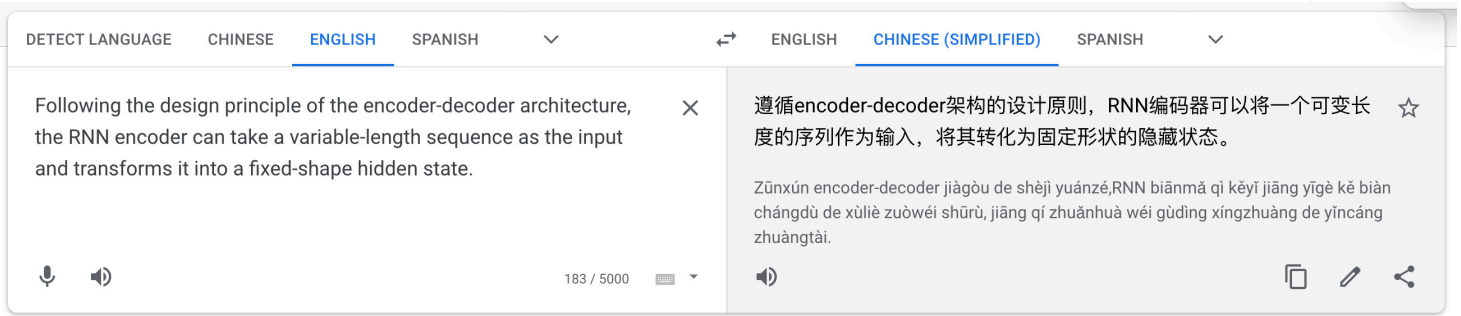
- 使用编码器-解码器架构的模型，编码器负责表示输入，解码器负责输出

序列到序列学习（seq2seq）

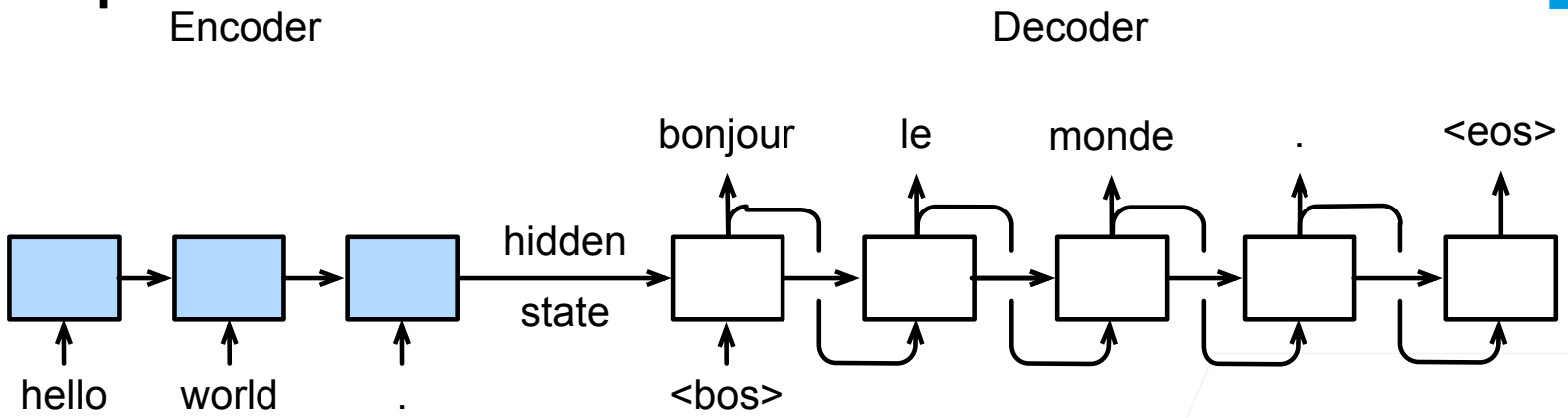


机器翻译

- 给定一个源语言的句子，自动翻译成目标语言
- 这两个句子可以有不同的长度

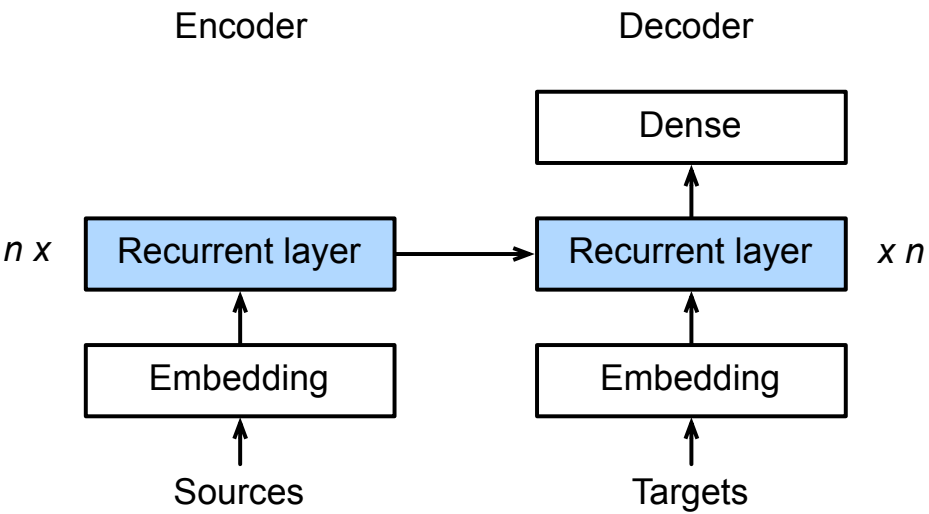


Seq2seq



- 编码器是一个RNN，读取输入句子
  - 可以是双向
- 解码器使用另外一个RNN来输出

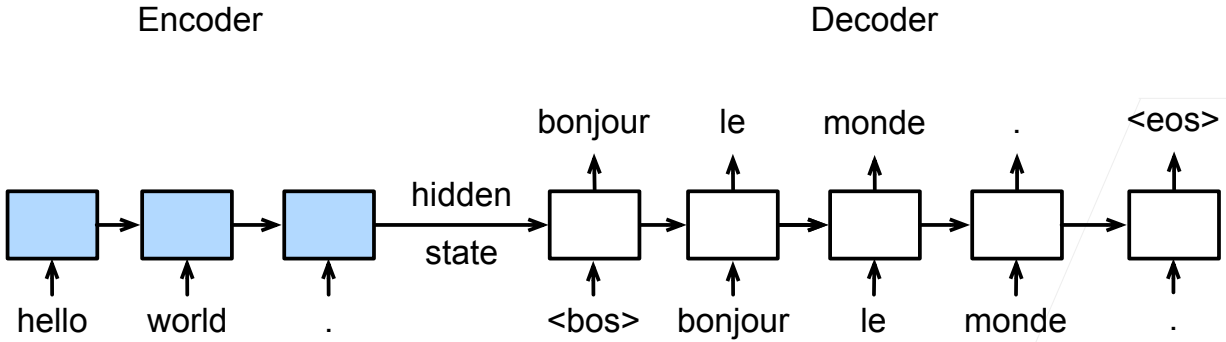
编码器-解码器细节



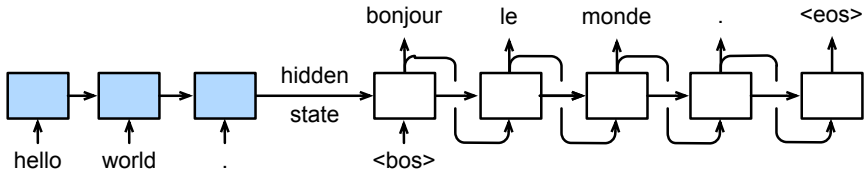
- 编码器是没有输出蹭到RNN
- 编码器最后时间步的隐状态用作解码器的初始隐状态

训练

- 训练时解码器使用目标句子作为输入



推理：



衡量生成序列的好坏的BLEU

- p_n 是预测中所有 n-gram 的精度
  - 标签序列$A\ B\ C\ D\ E\ F$和预测序列$A\ B\ B\ C\ D$，有

    $p_1 = 4/5,\ p_2 = 3/4,\ p_3 = 1/3,\ p_4 = 0$
- BLEU定义

$$\exp\left(\min\left(0, 1 - \frac{\text{len}_{\text{label}}}{\text{len}_{\text{pred}}}\right)\right)\prod_{n=1}^{k} p_n^{1/2^n}$$

惩罚过短的预测

长匹配有高权重

总结

- Seq2seq从一个句子生成另一个句子
- 编码器和解码器都是RNN
- 将编码器最后时间隐状态来初始解码器隐状态来完成信息传递
- 常用BLEU来衡量生成序列的好坏

束搜索

贪心搜索

- 在seq2seq中我们使用了贪心搜索来预测序列
  - 将当前时刻预测概率最大的词输出
- 但贪心很可能不是最优的：

贪心： $0.5\times0.4\times0.4\times0.6=0.048$

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

很好的选项：$0.5\times0.3\times0.6\times0.6=0.054$

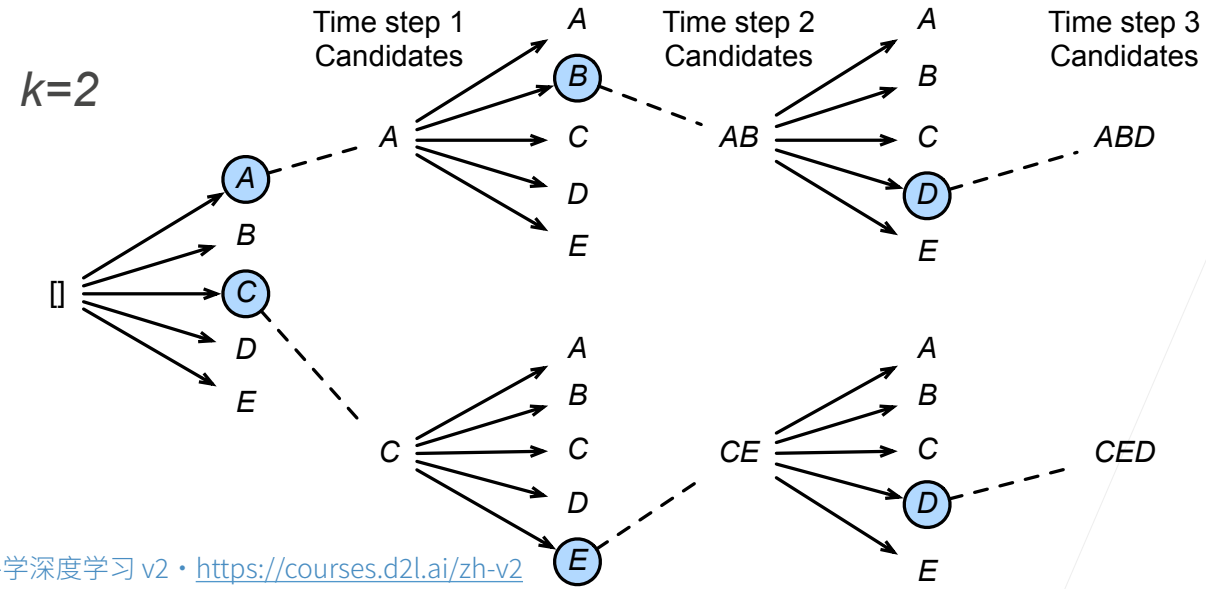
Time step	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	0.3	0.2	0.1
<eos>	0.1	0.2	0.1	0.6

穷举搜索

- 最优算法：对所有可能的序列，计算它的概率，然后选取最好的那个
- 如果输出字典大小为 n，序列最长为 T，那么我们需要考察 n^T 个序列
  - $n = 10000, \quad T = 10: \quad n^T = 10^{40}$
  - 计算上不可行

束搜索

- 保存最好的 k 个候选
- 在每个时刻，对每个候选新加一项（n 种可能），在 kn 个选项中选出最好的 k 个



束搜索

- 时间复杂度 $O(knT)$
  - $k = 5, \quad n = 10000, \quad T = 10: \quad knT = 5 \times 10^5$
- 每个候选的最终分数是：

$$\frac{1}{L^\alpha} \log p(y_1, \ldots, y_L) = \frac{1}{L^\alpha} \sum_{t'=1}^{L} \log p(y_{t'} \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c})$$

  - 通常 $\alpha = 0.75$

总结

- 束搜索在每次搜索时保存 k 个最好的候选
  - $k = 1$ 时是贪心搜索
  - $k = n$ 时是穷举搜索