第十章

序列建模：循环和递归网络

循环神经网络（RNN)

- 概念：是一类用于处理序列数据$x^{(1)}, \ldots, x^{(\tau)}$的神经网络
- 卷积网络：专门用于处理网格化数据 **X** （如一个图像）的神经网络

- 类比：卷积网络可以很容易地扩展到具有很大宽度和高度的图像，以及处理大小可变的图像；循环网络可以扩展到更长的序列，大多数循环网络也能处理可变长度的序列

- 从多层网络出发到循环网络的思想来源：**在模型的不同部分共享参数**，参数共享使得模型能够扩展到不同长度的样本）并进行泛化

- 案例：“I went to Nepal in 2009’’ 和 “In 2009, I went to Nepal.”
- 传统前馈网络下每个特征参数独立：需要分别学习句子每个位置的所有语言规则
- 循环神经网络参数共享：在几个时间步内共享相同的权重，不需要分别学习句子每个位置的所有语言规则。

循环神经网络（RNN)

- 参数共享的变种：**在序列上使用卷积**

- 卷积的输出是一个序列，其中输出中的每一项是相邻几项输入的函数
- 参数共享的概念体现在每个时间步中使用的相同卷积核
- 循环神经网络以不同的方式共享参数。输出的每一项是前一项的函数。输出的每一项对先前的输出应用相同的更新规则而产生。这种循环方式导致参数通过很深的计算图共享

- 本章相关定义：

- 为简单起见，我们说的 RNN 是指在序列上的操作，并且该序列在时刻 t（从 1 到 τ）包含向量 $x^{(t)}$
- 此外， 时间步索引不必是字面上现实世界中流逝的时间，有时，它仅表示序列中的位置

10.1 展开计算图

- 计算图：是形式化一组计算结构的方式，如那些涉及将输入和参数映射到输出和损失的计算

- 考虑动态系统的经典形式：

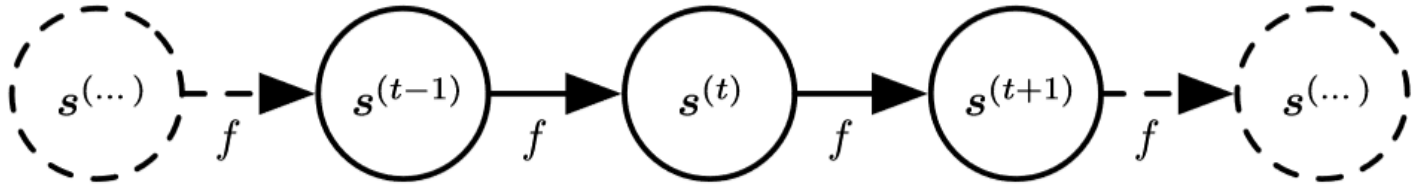
$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

- $\boldsymbol{s}$ 在时刻 t 的定义需要参考时刻 t−1 时同样的定义，因此上式是循环的

- 令t=3，可将上式展开，得到不涉及循环的表达：

$$\boldsymbol{s}^{(3)} = f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) = f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta}).$$

- 以上两个式子的展开计算图：



10.1 展开计算图

- 作为另一个例子，让我们考虑由外部信号 $\boldsymbol{x}^{(t)}$ 驱动的动态系统：

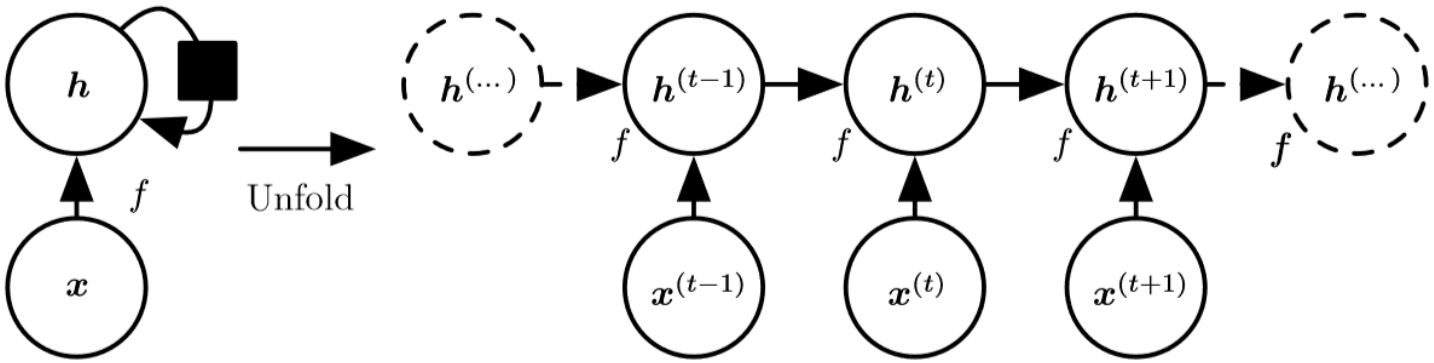
$$\boldsymbol{s}^{(t)} = f((\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- 在上式中，当前状态包含了整个过去序列的信息

- 循环神经网络可以通过许多不同的方式建立，对上式进行重写，很多循环神经网络使用下式或类似的公式定义隐藏单元的值：

$$\boldsymbol{h}^{(t)} = f((\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- 如下图所示，对应了上式的两种绘制方式。左图类比于生物神经网络，两个物理节点之间存在相互作用（黑色方块）；右图表示展开的计算图



10.1 展开计算图

- 两个不同的绘制方式用公式表达如下：

$$\boldsymbol{h}^{(t)} = \boldsymbol{g}^{(t)}\left(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right) = f((\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- 展开过程引入两个主要优点：
- 无论序列的长度，学成的模型始终具有相同的输入大小，是泛化的，而不是在每个时间步内独立的。
- 我们可以在每个时间步使用相同参数的相同转移函数 f。

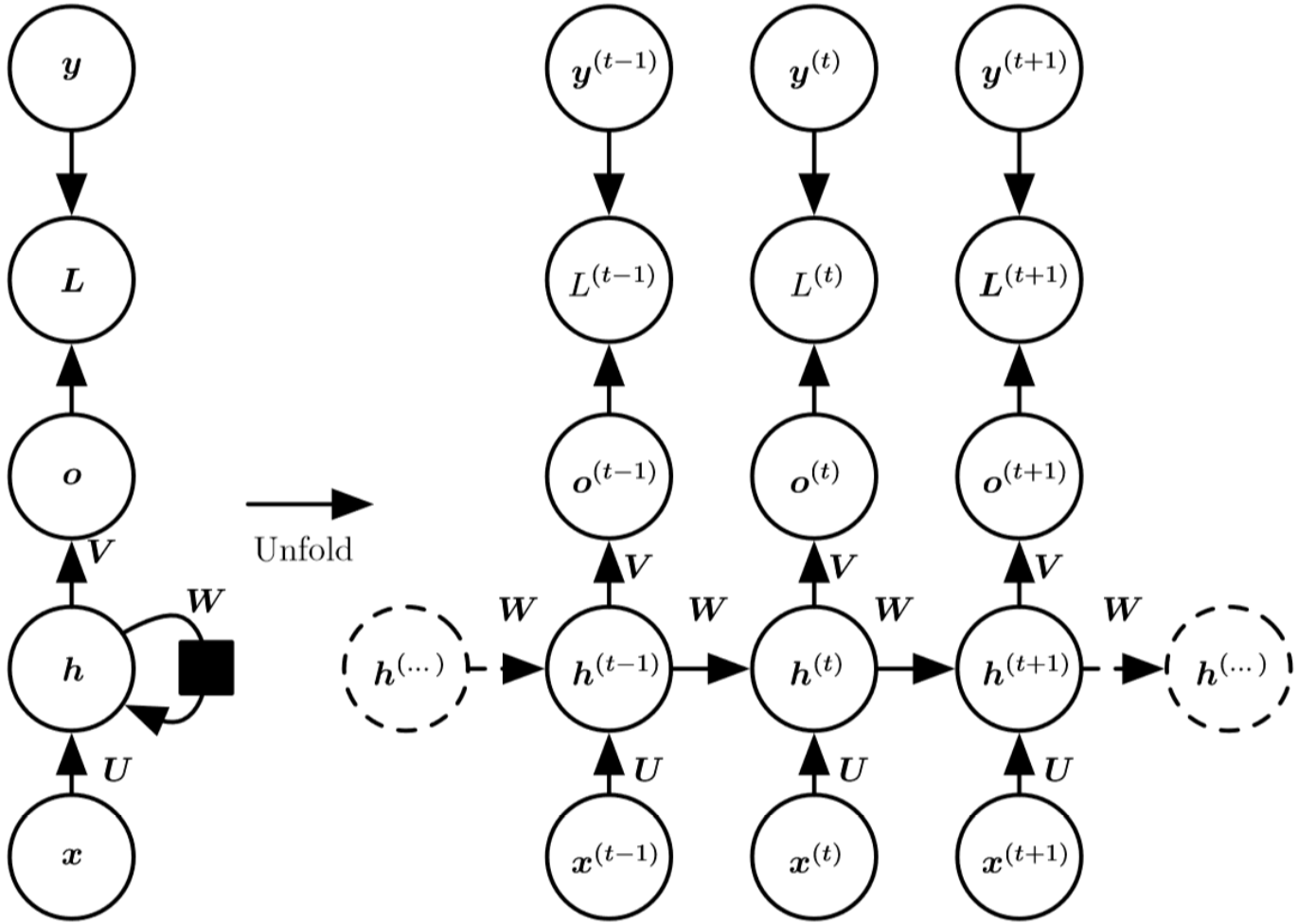
- 上述优点带来的结果：
- 这两个因素使得学习在所有时间步和所有序列长度上操作单一的模型 f 是可能的， 而不需要在所有可能时间步学习独立的模型 $\boldsymbol{g}^{(t)}$ 。学习单一的共享模型允许泛化到没有见过的序列长度

10.2 循环神经网络

- 基于图展开和参数共享的思想，我们可以设计各种循环神经网络

- 计算循环网络(将 $\boldsymbol{x}$ 值的输入序列映射到输出值 $\boldsymbol{o}$ 的对应序列) 训练损失的计算图。损失 L 衡量每个 $\boldsymbol{o}$ 与相应的训练目标 $\boldsymbol{y}$ 的距离

- RNN输入到隐藏的连接由权重矩阵 $\boldsymbol{U}$ 参数化，隐藏到隐藏的循环连接由权重矩阵 $\boldsymbol{W}$ 参数化以及隐藏到输出的连接由权重矩阵 $\boldsymbol{V}$ 参数化

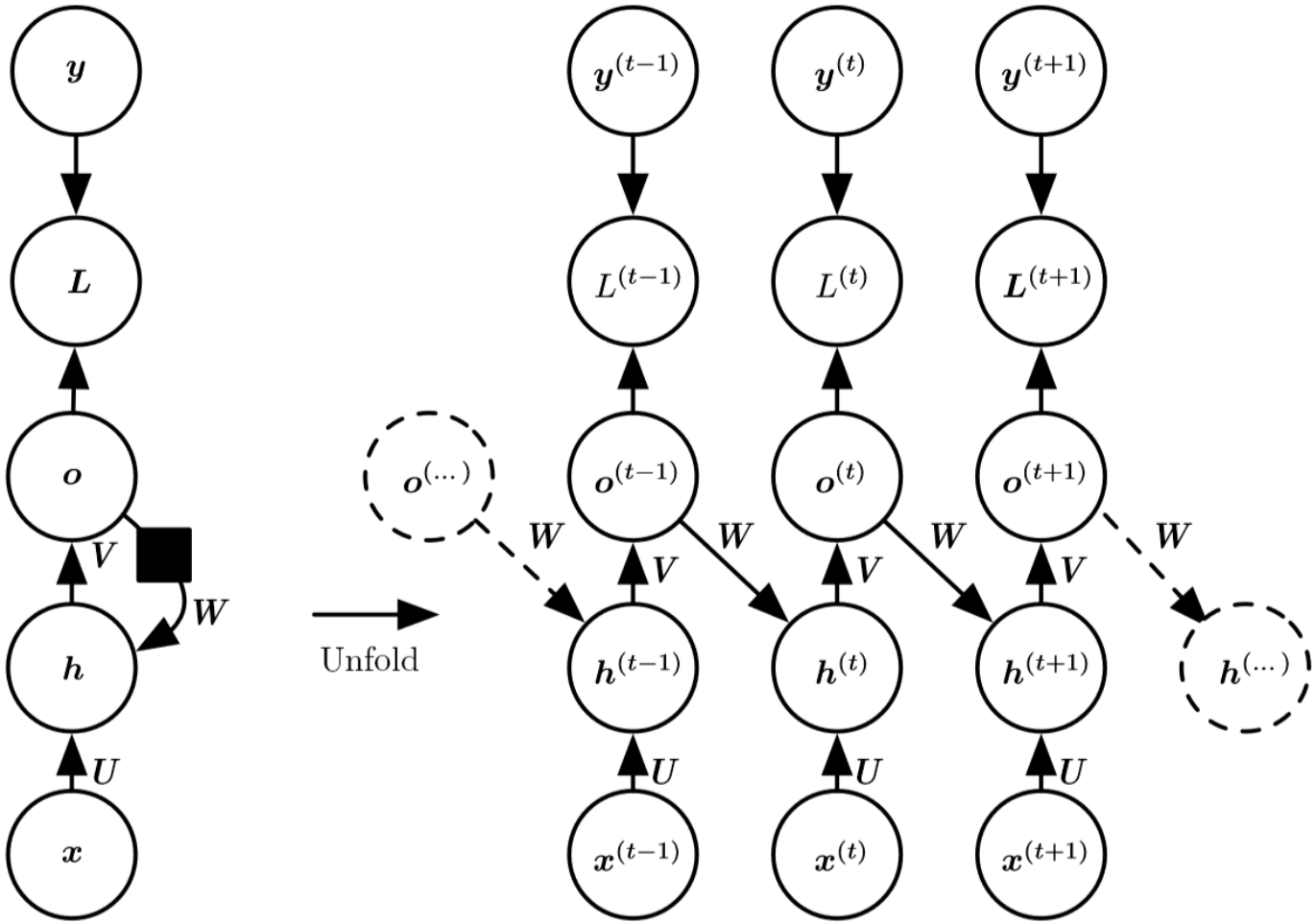


10.2 循环神经网络

- 此类 RNN 的唯一循环是从输出到隐藏层的反馈连接

- 这样 的 RNN 没有上图表示的 RNN 那样强大（只能表示更小的函数集合）。上图中的 RNN 可以选择将其想要的关于过去的任何信息放入隐藏表示 $\boldsymbol{h}$ 中并且将 $\boldsymbol{h}$ 传播到未来。该图中的 RNN 被 训练为将特定输出值放入 $\boldsymbol{o}$ 中，并且 $\boldsymbol{o}$ 是允许传播到未来的唯一信息

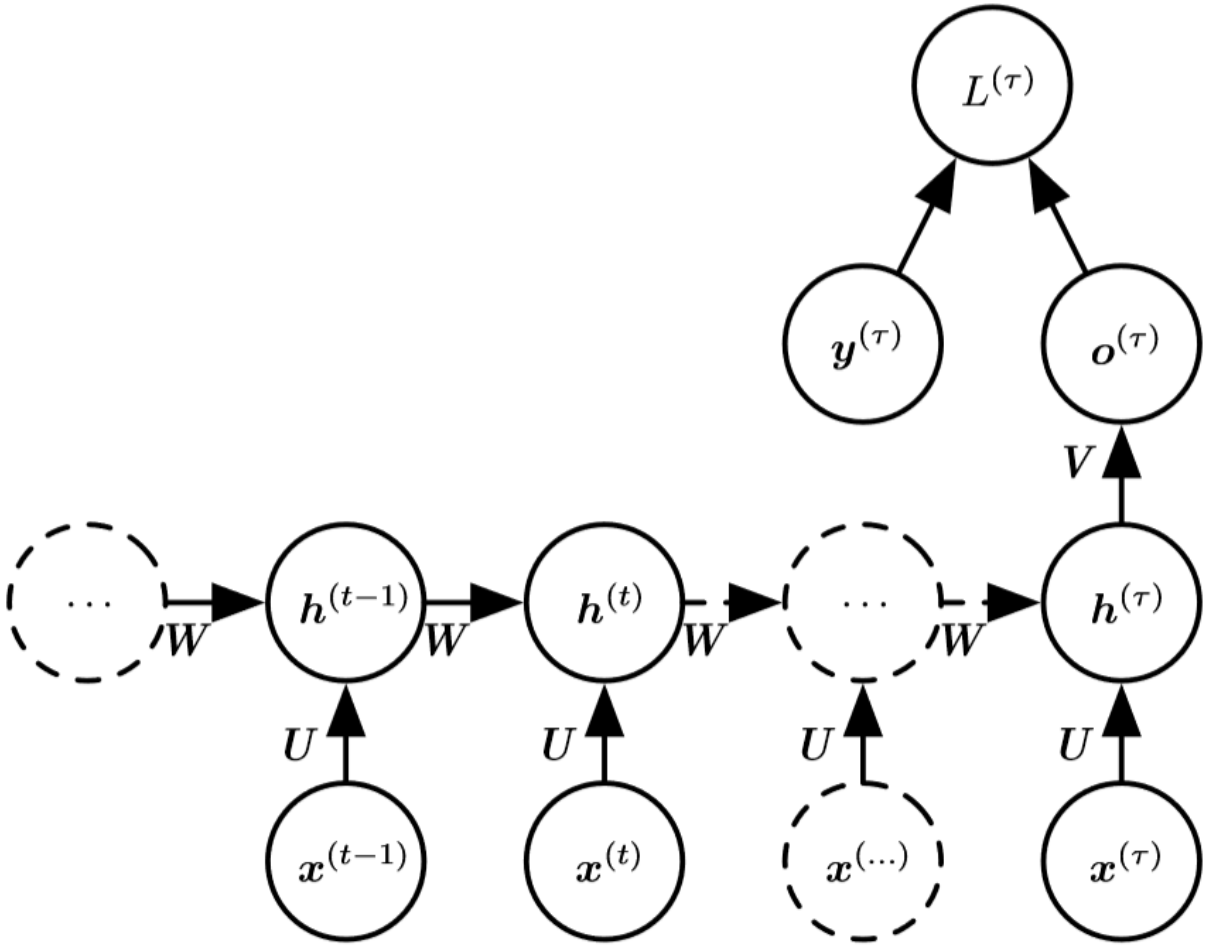
- $\boldsymbol{o}$ 通常缺乏过去的重要信息，除非它非常高维且内容丰富。这使得该图中的 RNN 不那么强大，但是它更容易训练



10.2 循环神经网络

- 关于时间展开的循环神经网络，在序列结束时具有单个输出。

这样的网络可以用于概括序列并产生用于进一步处理的固定大小的表示。在结束处可能存在目标（如此处所示），或者通过更下游模块的反向传播来获得输出$\boldsymbol{o}^{(t)}$上的梯度



10.2 循环神经网络

- 循环神经网络中一些重要的设计模式包括以下几种：

- 每个时间步都有输出，并且隐藏单元之间有循环连接的循环网络，如第一图所示。
- 每个时间步都产生一个输出，只有当前时刻的输出到下个时刻的隐藏单元之间有循环连接的循环网络，如第二图所示。
- 隐藏单元之间存在循环连接，但读取整个序列后产生单个输出的循环网络，如第三图所示。

- 对第一图中前向传播公式推导的前提假设：

- 图没有指定隐藏单元的激活函数，我们假设使用双曲正切激活函数
- 图中没有明确指定何种形式的输出和损失函数，我们假定输出是离散的，如用于预测词或字符的RNN，表示离散变量的常规方式是把输出 o 作为每个离散变量可能值的非标准化对数概率。

10.2 循环神经网络

- 第一图中的前向传播公式:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \qquad (10.8)$$

$$\boldsymbol{h}^{(t)} = tanh\left(\boldsymbol{a}^{(t)}\right), \qquad (10.9)$$

$$\hat{\boldsymbol{y}}^{(t)} = softmax\left(\boldsymbol{o}^{(t)}\right), \qquad (10.11)$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \qquad (10.10)$$

- 其中的参数的偏置向量 $\boldsymbol{b}$ 和 $\boldsymbol{c}$ 连同权重矩阵 $\boldsymbol{U}$、$\boldsymbol{V}$ 和 $\boldsymbol{W}$，分别对应于输入到隐藏、 隐藏到输出和隐藏到隐藏的连接。这个循环网络将一个输入序列映射到相同长度的 输出序列。与 $\boldsymbol{x}$ 序列配对的 $\boldsymbol{y}$ 的总损失就是所有时间步的损失之和。例如，$\boldsymbol{L}^{(t)}$ 为 给定的 $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}$后 $\boldsymbol{y}^{(t)}$的负对数似然，则

$$\begin{aligned} & L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\ &= \sum_t L^{(t)} \\ &= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right) \end{aligned}$$

10.2 循环神经网络

- 第一图中前向传播相关分析:

- 关于各个参数计算这个损失函数的梯度是计算成本很高的操作
- 梯度计算涉及执行 一次前向传播（如在第一图展开图中从左到右的传播），接着是由右到左的反向传播。运行时间是 O(τ)，并且不能通过并行化来降低，因为前向传播图是固有循序的; 每个时间步只能一前一后地计算
- 前向传播中的各个状态必须保存，直到它们反向传播中被再次使用，因此内存代价也是 O(τ)。**应用于展开图且代价为 O(τ) 的反向传播算法称为通过时间反向传播（back-propagation through time, BPTT）**

- 结论：如第一图所示的隐藏单元之间存在循环的网络非常强大但训练代价也很大

10.2 循环神经网络

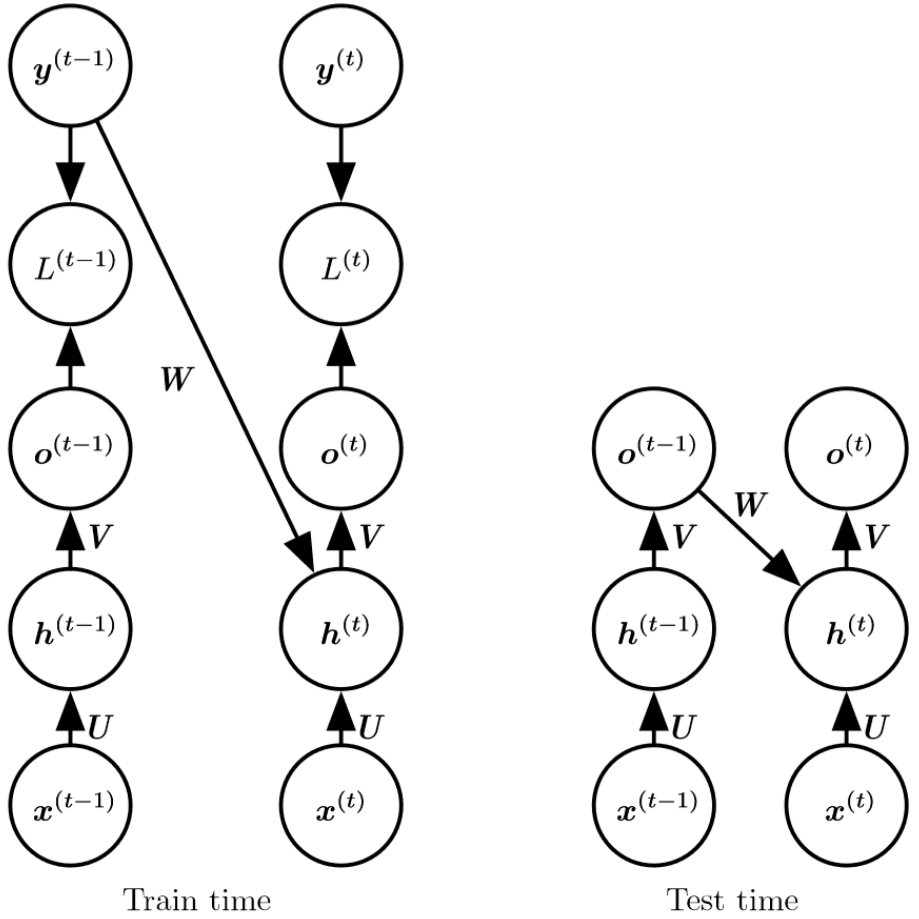
- **10.2.1 导师驱动过程和输出循环网络**

• 导师驱动过程：训练模型时，导师驱动过程不再使用最大似然准则，而在时刻 t + 1 接收 真实值 y(t) 作为输入：

$$\log p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$
$$= \log p(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) + \log p(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$$

• (左) 训练时，我们将训练集中正确的输出 $y^{(t)}$反馈到 $h^{(t+1)}$。

• (右) 当模型部署后，真正的输出通常是未知的。在这种情况下，我们用模型的输出 $o^{(t)}$近似正确的输出 $y^{(t)}$， 并反馈回模型



10.2 循环神经网络

- **10.2.2 计算循环神经网络的梯度**

- 基本思想：由反向传播计算得到的梯度，并结合任何通用的基于梯度的技术就可以训练 RNN

- 为了获得 BPTT 算法行为的一些直观理解，文章举例说明如何通过 BPTT 计算 上述RNN公式（式(10.8)和式(10.12)）的梯度：

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \qquad (10.8)$$

$$\boldsymbol{L}\left(\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\right\}, \left\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\right\}\right), \qquad (10.12)$$

- 计算图的节点包括参数 $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{b}$ 和 $\boldsymbol{c}$，以及以 t 为索引的节点序列 $\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t)}, \boldsymbol{o}^{(t)}$ 和 $\boldsymbol{L}^{(t)}$。对于每一个节点 **N**，我们需要基于**N**后面的节点的梯度，递归地计算梯度 $\nabla_N L$。

10.2 循环神经网络

- **10.2.2 计算循环神经网络的梯度**

- 从紧接着最终损失的节点开始，首先计算$\boldsymbol{L}^{(t)}$：

$$\frac{\partial L}{\partial L^{(t)}} = 1.$$

- 其次计算$\boldsymbol{o}^{(t)}$，假设损失是迄今为止给定了输入后的真实目标 $\boldsymbol{y}^{(t)}$ 的负对数似然，对于所有 i, t，关于时间步 t 输出的梯度 $\nabla_{o^{(t)}} L$ 如下：

- $(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial {o_i}^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial {o_i}^{(t)}} = {\hat{y}_i}^{(t)} - \mathbf{1}_{i,y^{(t)}}$

- 结果来源：$\hat{\boldsymbol{y}}^{(t)} = softmax\left(\boldsymbol{o}^{(t)}\right)$，对$\hat{\boldsymbol{y}}^{(t)}$做负对数似然求导(交叉熵)

实际上是对$softmax\left(\boldsymbol{o}^{(t)}\right)$求导，而$softmax$函数的定义为：

$$S_i = \frac{e_i}{\sum_j e_j}$$

$$\begin{aligned}
\frac{\partial Loss_i}{\partial_i} &= -\frac{\partial ln y_i}{\partial_i} \\
&= \frac{\partial(-ln\frac{e^i}{\sum_j e^j})}{\partial_i} \\
&= -\frac{1}{\frac{e^i}{\sum_j e^j}} \cdot \frac{\partial(\frac{e^i}{\sum_j e^j})}{\partial_i} \\
&= -\frac{\sum_j e^j}{e^i} \cdot \frac{\partial(1 - \frac{\sum_{j \neq i} e^j}{\sum_j e^j})}{\partial_i} \\
&= -\frac{\sum_j e^j}{e^i} \cdot (-\sum_{j \neq i} e^j) \cdot \frac{\partial(\frac{1}{\sum_j e^j})}{\partial_i} \\
&= \frac{\sum_j e^j \cdot \sum_{j \neq i} e^j}{e^i} \cdot \frac{-e^i}{(\sum_j e^j)^2} \\
&= \frac{\sum_{j \neq i} e^j}{\sum_j e^j} \\
&= -(1 - \frac{e^i}{\sum_j e^j}) \\
&= y_i - 1
\end{aligned}$$

10.2 循环神经网络

- **10.2.2 计算循环神经网络的梯度**

- 再次计算$\boldsymbol{h}^{(t)}$，

$$\nabla_c L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}}\right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L, \tag{10.22}$$

$$\nabla_b L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L, \tag{10.23}$$

- 然后，我们可以

$$\nabla_V L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_V o_i^{(t)} = \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \boldsymbol{h}^{(t)\top}, \tag{10.24}$$

$$\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{W^{(t)}} h_i^{(t)} \tag{10.25}$$

$$= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)\top}, \tag{10.26}$$

- 类似的，其他的

$$\nabla_U L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{U^{(t)}} h_i^{(t)} \tag{10.27}$$

$$= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)\top}, \tag{10.28}$$

：

具有 $\boldsymbol{o}^{(t)}$ 和 $\boldsymbol{h}^{(}$

$\boldsymbol{h}^{(t)}$(t < τ) 同时

10.2 循环神经网络

- **10.2.3 作为有向图模型的循环网络**

- 基本思想:

- 将整个序列 $\boldsymbol{y}$ 的联合分布分解为一系列单步的概率预测是捕获关于整个序列完整联合分布的一种方法

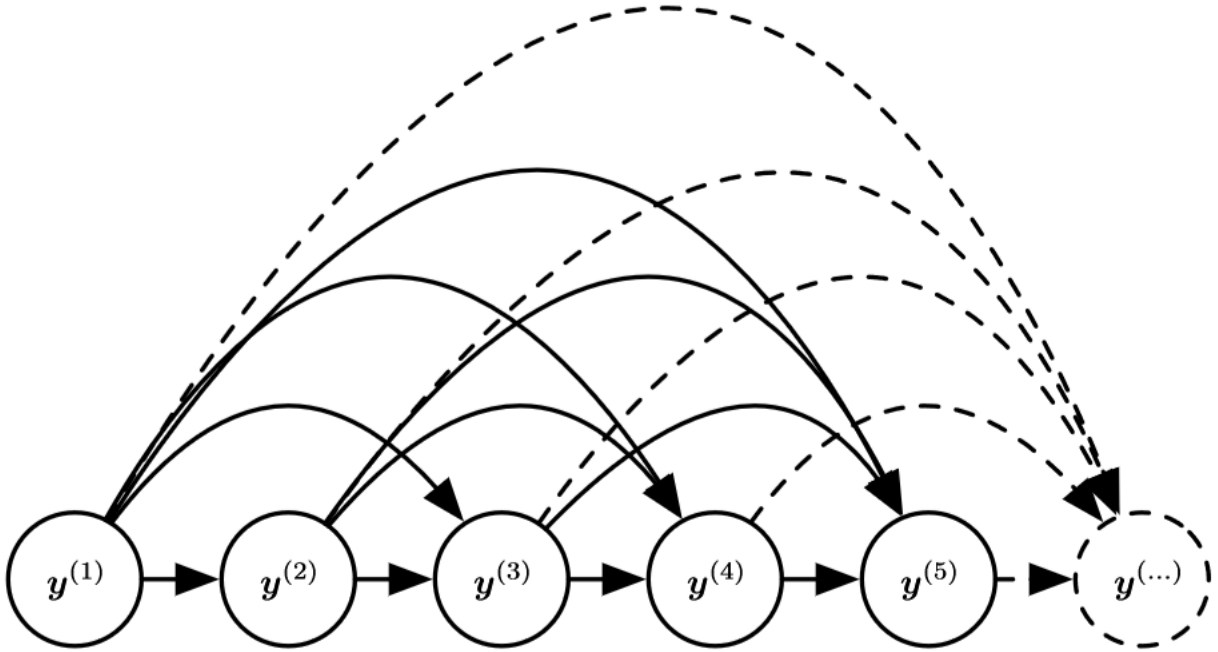
- 当我们不把过去的 $\boldsymbol{y}$ 值反馈给下一步作为预测的条件时，那么有向图模型不包含任何从过去 $\boldsymbol{y}^{(i)}$到当前 $\boldsymbol{y}^{(t)}$的边。在这种情况下，输出 $\boldsymbol{y}$ 与给定的 $\boldsymbol{x}$ 序列是条件独立的

- 当我们反馈真实的 $\boldsymbol{y}$ 值（不是它们的预测值，而是真正观测到或生成的值）给网络时，那么有向图模型包含所有从过去 $\boldsymbol{y}^{(i)}$到当前 $\boldsymbol{y}^{(t)}$ 的边

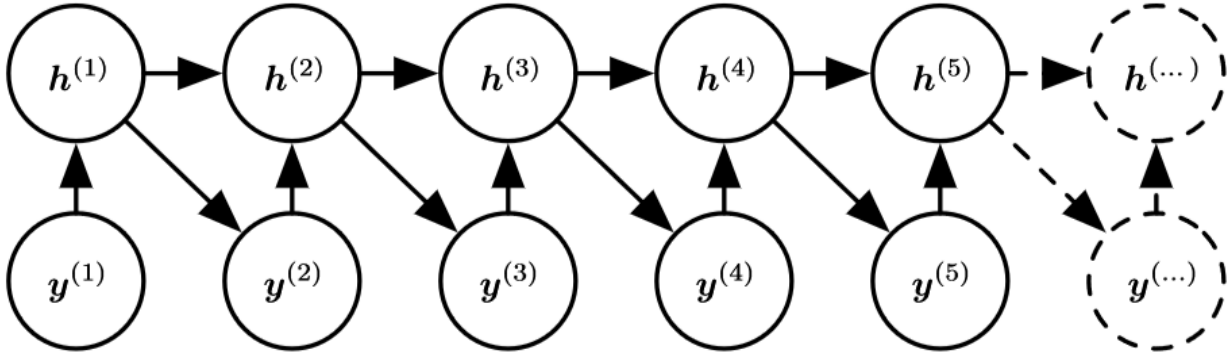
10.2 循环神经网络

- **10.2.3 作为有向图模型的循环网络**

- 全连接图模型:
- 给定先前的值，每个过去的观察值 $\boldsymbol{y}^{(i)}$ 可以影响一些 $\boldsymbol{y}^{(t)}(t > i)$ 的条件分布。当序列中每个元素的输入和参数的数目越来越多，根据此图直接参数化图模型可能是非常低效的



- 高效参数化全连接图模型:
- 序列中的每个阶段（对于 $\boldsymbol{h}^{(t)}$和 $\boldsymbol{y}^{(t)}$ ）使用相同的结构（每个 节点具有相同数量的输入），并且可以与其他阶段共享相同的参数



10.2 循环神经网络

- **10.2.3 作为有向图模型的循环网络**

- 在循环网络中使用的参数共享的前提是相同参数可用于不同时间步的假设，这意味着之前的时间步与下个时间步之间的关系并不依赖于 t，在这种情况下，可以采用模型采样进行优化

- 模型采样：简单地从每一时间步的条件分布采样
- 进行模型采样时，RNN 必须有某种机制来确定序列的长度。这可以通过多种方式实现：

- 在当输出是从词汇表获取的符号的情况下，我们可以添加一个对应于序列末端的特殊符号，当产生该符号时，采样过程停止
- 另一种选择是在模型中引入一个额外的 Bernoulli 输出，表示在每个时间步决定 继续生成或停止生成
- 确定序列长度 τ 的另一种方法是将一个额外的输出添加到模型并预测整数 τ 本 身。模型可以采出 τ 的值，然后采 τ 步有价值的数据

10.2 循环神经网络

- **10.2.4 基于上下文的RNN序列建模**

- 上一节描述了没有输入 $\boldsymbol{x}$ 时，关于随机变量序列 $\boldsymbol{y}^{(t)}$ 的RNN如何对应于有向图模型。而如式(10.8)所示的RNN包含一个输入序列$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(\tau)}$。一般情况下，RNN 允许将图模型的观点扩展到不仅代表 $\boldsymbol{y}$ 变量的联合分布也能表示给定 $\boldsymbol{x}$ 后 $\boldsymbol{y}$ 条件分布

- 之前，我们已经讨论了将 $t = 1, \ldots, \tau$ 的向量 $\boldsymbol{x}^{(t)}$ 序列作为输入的 RNN。另一 种选择是只使用单个向量 $\boldsymbol{x}$ 作为输入。当 $\boldsymbol{x}$ 是一个固定大小的向量时，我们可以简单地将其看作产生 $\boldsymbol{y}$ 序列 RNN 的额外输入。将额外输入提供到 RNN 的一些常见方法是：

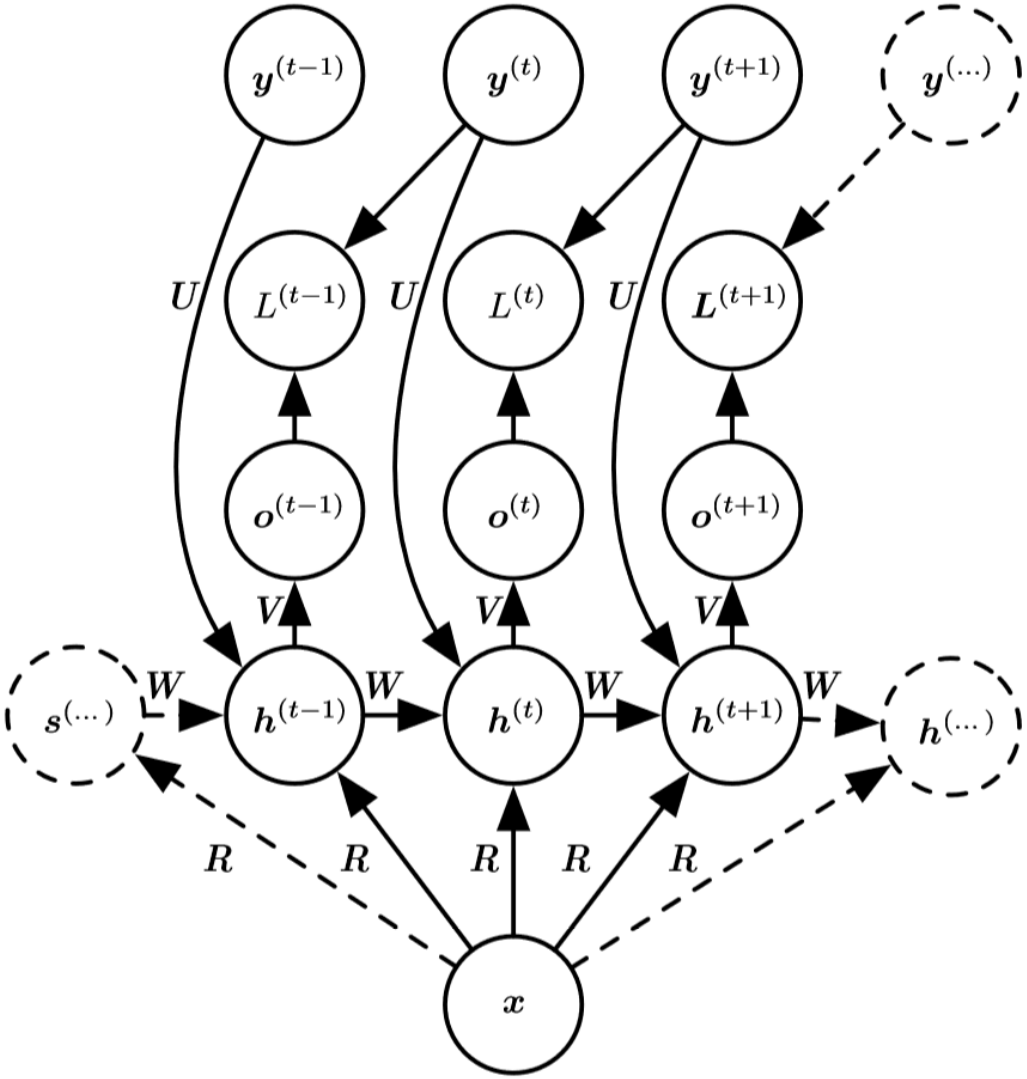
- 在每个时刻作为一个额外输入
- 作为初始状态 $\boldsymbol{h}^{(0)}$
- 结合两种方式

10.2 循环神经网络

- **10.2.4 基于上下文的RNN序列建模**

- 单个固定$\boldsymbol{x}$在每个时刻作为一个额外输入

- 输入 $\boldsymbol{x}$ 和每个隐藏单元向量 $\boldsymbol{h}^{(t)}$ 之间 的相互作用是通过新引入的权重矩阵 $\boldsymbol{R}$ 参数化的，这是只包含 $\boldsymbol{y}$ 序列的模型所没有 的。同样的乘积 $\boldsymbol{x}^{\top}\boldsymbol{R}$ 在每个时间步作为隐藏单元的一个额外输入



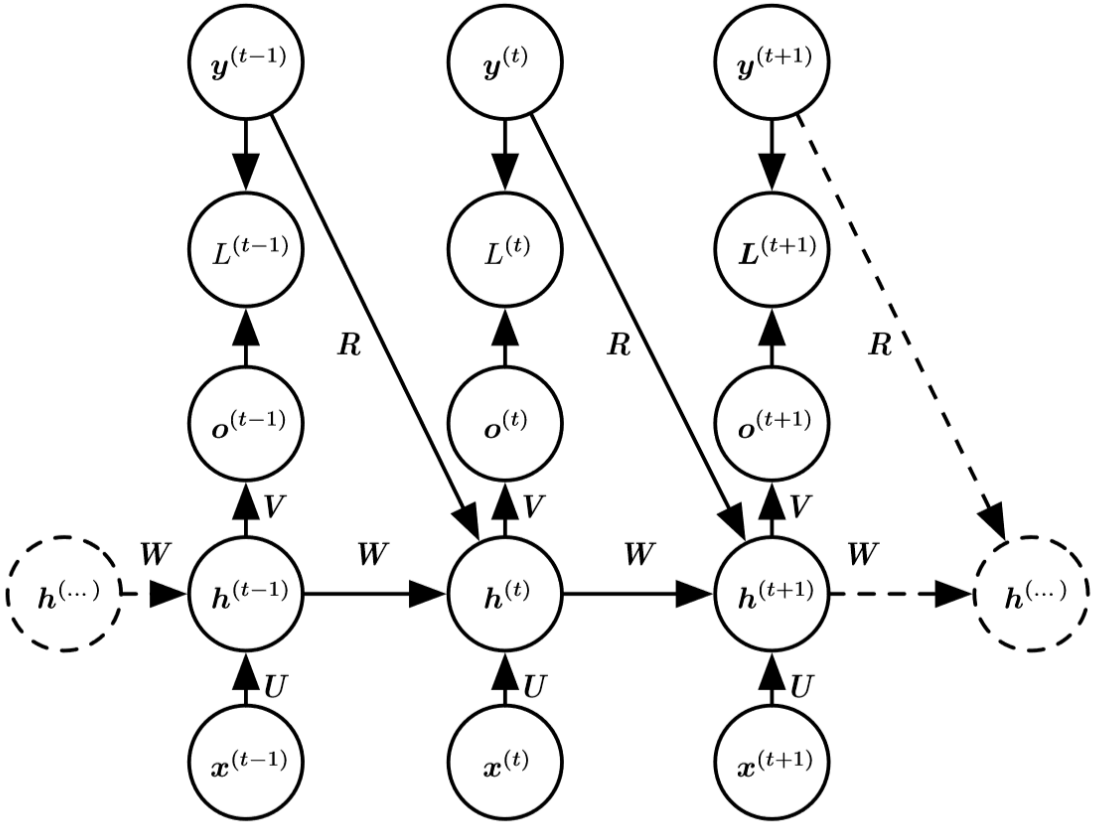
10.2 循环神经网络

- **10.2.4 基于上下文的RNN序列建模**

- 可变向量序列 $\boldsymbol{x}^{(t)}$作为额外输入

- 将可变长度的 $\boldsymbol{x}$ 值序列映射到相同长度的 $\boldsymbol{y}$ 值序列上分布的条件循环神经网络。

- PS：这种给定一个序列表示另一个序列分布的模型的还是有一个限制，就是这两个序列的长度必须是相同的。我们将在第10.4节描述如何消除这种限制



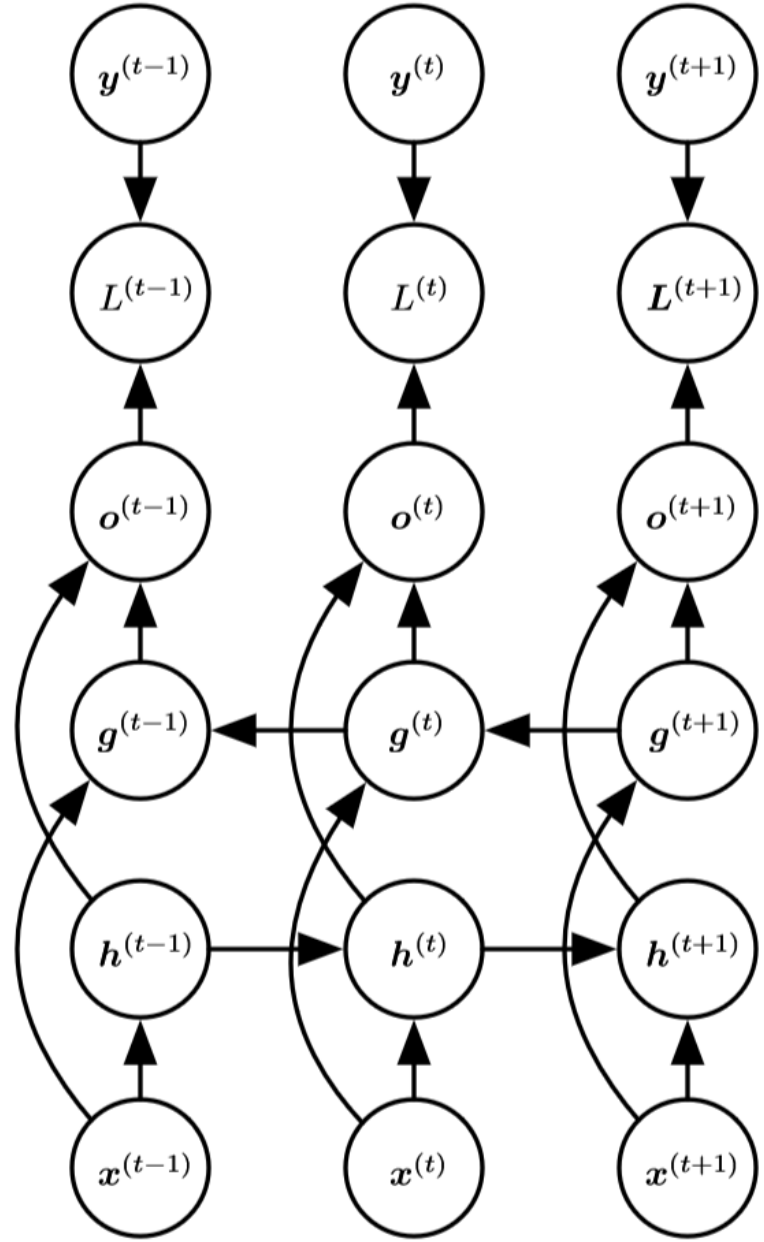
10.3 双向RNN

- 前面所讨论的：在时刻 t 的 状态只能从过去的序列$\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t-1)}$以及当前的输入 $\boldsymbol{x}^{(t)}$ 捕获信息，或者在 $\boldsymbol{y}$ 可用时，允许过去的 $\boldsymbol{y}$ 值信息影响当前状态的模型

- 现实区别：在许多应用中，我们要输出的 $\boldsymbol{y}^{(t)}$ 的预测可能依赖于整个输入序列。例如，在语音识别中，由于协同发音，当前声音作为音素的正确解释可能取决于未来几个音素，甚至潜在的可能取决于未来的几个词

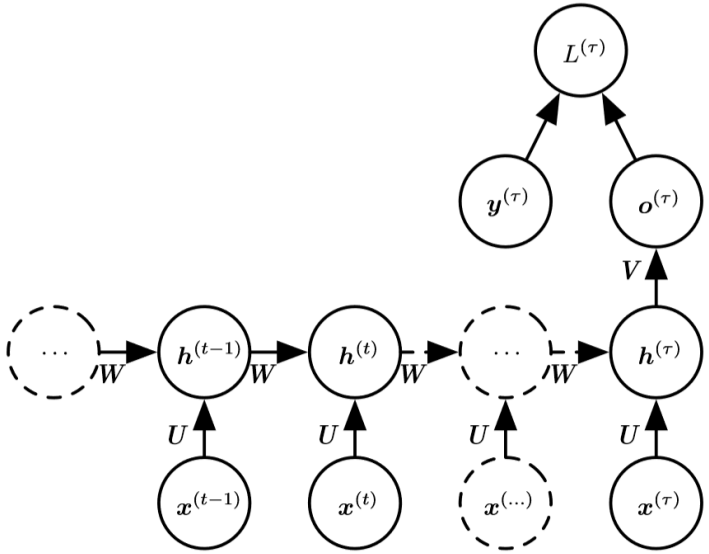
- 双向RNN：结合时间上从序列起点开始移动的 RNN 和另一个时间上从序列末尾开始移动的 RNN

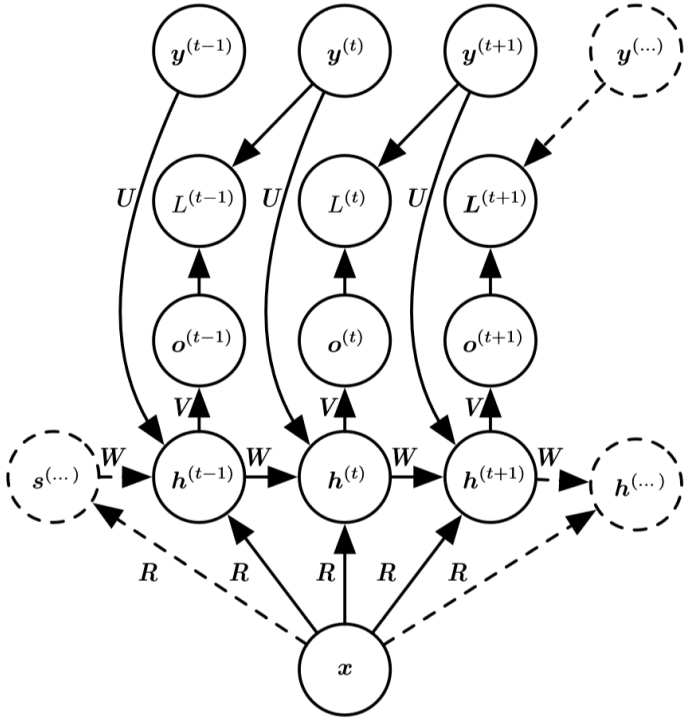
- 拓展：这个想法可以自然地扩展到 2 维输入，如图像，由四个 RNN 组成，每一个沿 着四个方向中的一个计算：上、下、左、右。相比卷积网络，应用于图像的 RNN 计算成本通常更高，但允许同一特征图的特征之间存在长期横向的相互作用

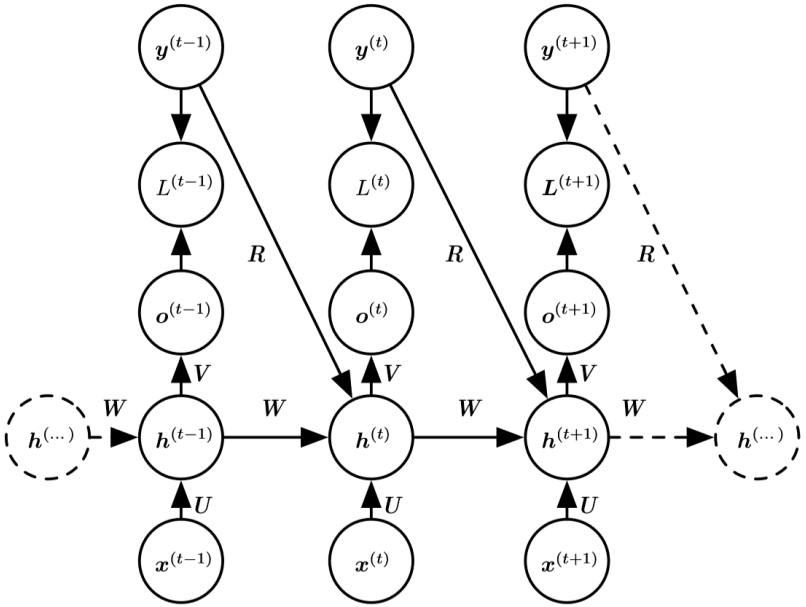


10.4 基于编码-解码的序列到序列架构

- 思想来源







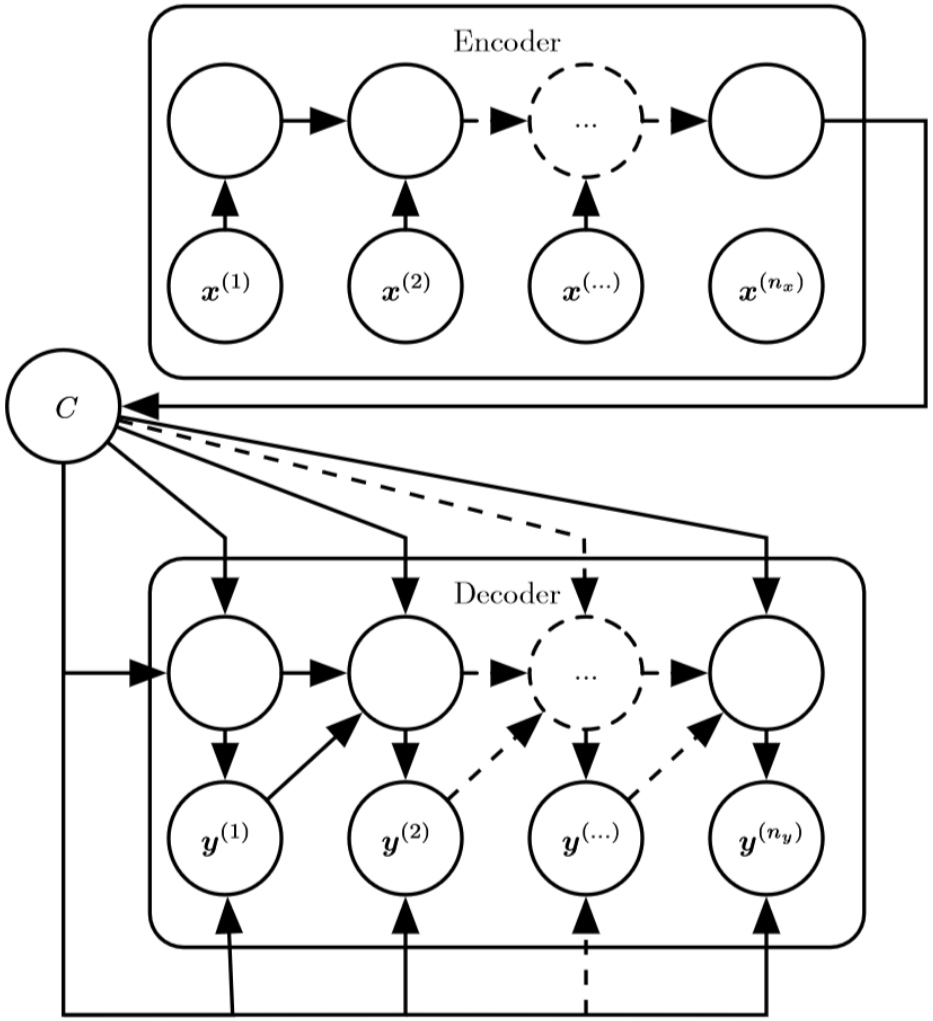
- 上述三张图片对应的RNN功能分别是：输入一个序列输出固定大小的向量、将固定大小的向量映射成一个序列、将一个输入序列映射到等长的输出序列
- 编码：图一到图二
- 解码：图二到图三

10.4 基于编码-解码的序列到序列架构

- 正如我们所见，向量到序列 RNN 至少有两种接受输入的方法。输入可以被提供为 RNN 的初始状态，或连接到每个时间步中的隐藏单元。这两种方式也可以结合。

- 这里并不强制要求编码器与解码器的隐藏层具有相同的大小。

- 此架构的一个明显不足是，编码器 RNN 输出的上下文 C 的维度太小而难以适当地概括一个长序列。这种现象由 Bahdanau et al. (2015) 在机器翻译中观察到。他们提出让 C 成为可变长度的序列，而不是一个固定大小的向量。此外，他们还引入 了将序列 C 的元素和输出序列的元素相关联的**注意力机制（attention mechanism）**。读者可在第12.4.5.1节了解更多细节



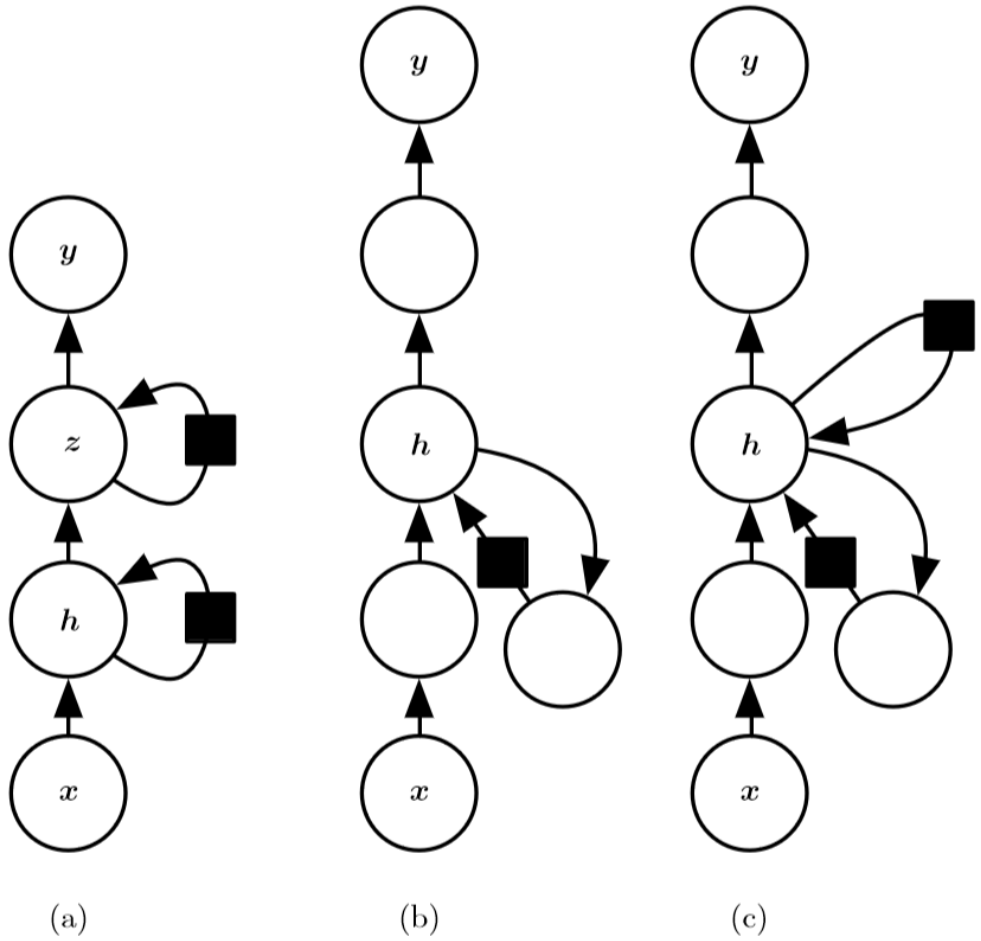
10.5 深度循环网络

- 大多数 RNN 中的计算可以分解成三块参数及其相关的变换：

1. 从输入到隐藏状态、2. 从前一隐藏状态到下一隐藏状态
3. 从隐藏状态到输出

- 思想：能通过深度 MLP 内单个层（包含以上三个步骤）来表示的变换称为浅变换，考虑引入深度（多层变化）对结果往往有利

- (a) 隐藏循环状态可 以被分解为具有层次的组
- (b) 可以向输入到隐藏，隐藏到隐藏以及隐藏到输出的部分引入更深的 计算 (如 MLP)。这可以延长链接不同时间步的最短路径（缺点：路径加深边长，优化困难）。
- (c) 可以引入跳跃连接来缓解路径延长 的效应。

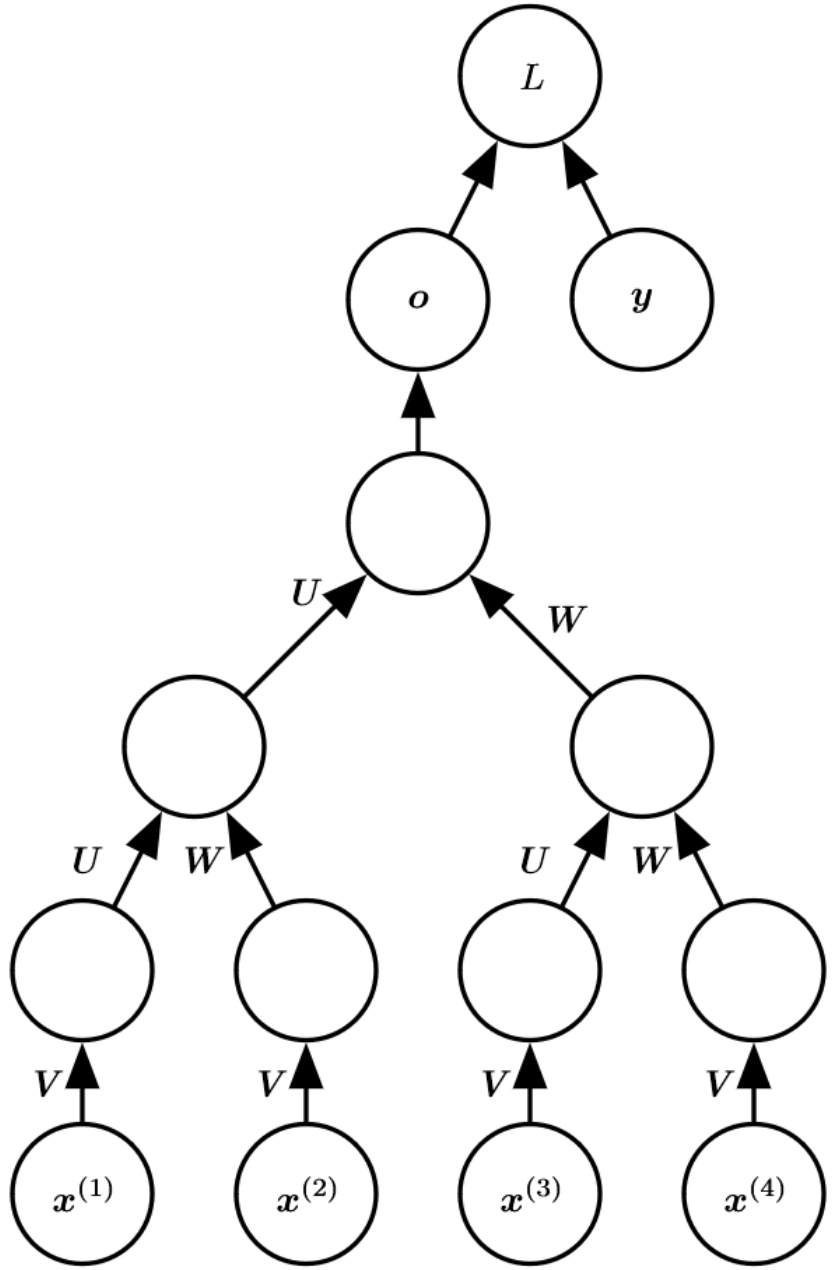


10.6 递归神经网络

- 特征：递归神经网络代表循环网络的另一个扩展，它被构造为深的树状结构而不 是RNN的链状结构，因此是不同类型的计算图

- 递归网络已成功地应用于输入是**数据结构**的神经网络(Frasconi et al., 1997, 1998)，如**自然语言处理** (Socher et al., 2011a,c, 2013a) 和**计算机视觉** (Socher et al., 2011b)

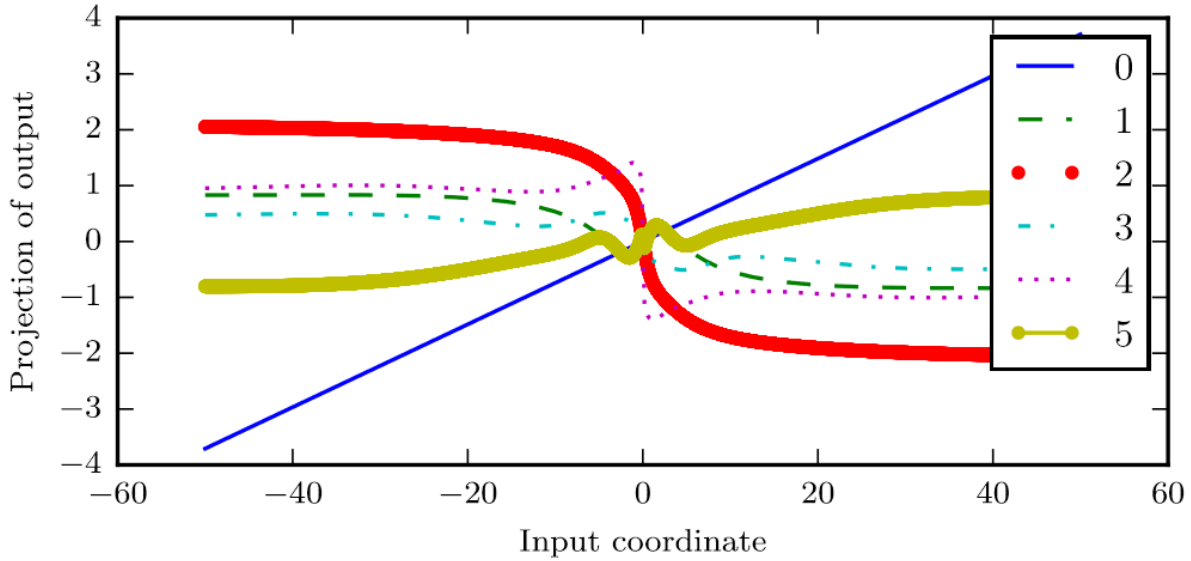
- 递归网络的明显优势：对于具有相同长度 τ 的序列，深度（通过非线性操作的组合数量来衡量）可以急剧地从 τ 减小为 $O(\log\tau)$，这可能有助于解决依赖。一个悬而未决的问题是如何以最佳的方式构造树。一种选择是使用不依赖于 数据的树结构，如平衡二叉树



10.7 长期依赖的挑战

- **描述：梯度消失与梯度爆炸**
- 学习循环网络长期依赖的数学挑战在第8.2.5节中引入。根本问题是，经过许 阶段传播后的梯度倾向于消失（大部分情况）或爆炸（很少，但对优化过程影响很 大）。即使我们假设循环网络是参数稳定的（可存储记忆，且梯度不爆炸），但长期依赖的困难来自比短期相互作用指数小的权重（涉及许多 Jacobian 相乘）

- 循环网络涉及相同函数的多次组合，每个时间步一次。这些组合可以导致极端非线性行为



10.7 长期依赖的挑战

- **分析：梯度消失与梯度爆炸**
- 特别地，循环神经网络所使用的函数组合有点像矩阵乘法。我们可以认为，循环联系

$$h^{(t)} = W^{\top} h^{(t-1)}$$

- 是一个非常简单的、缺少非线性激活函数和输入 x 的循环神经网络。它可以被简化为

$$h^{(t)} = (W^{t})^{\top} h^{(0)}$$

- 而当 W 符合下列形式的特征分解

$$W = Q \Lambda Q^{\top}$$

- 其中 Q 正交，循环性可进一步简化为

$$h^{(t)} = Q^{\top} \Lambda^{t} Q h^{(0)}$$

- 特征值提升到 t 次后，导致幅值不到一的特征值衰减到零（**梯度消失**），而幅值大于一的就会激增（**梯度爆炸**）。任何不与最大特征向量对齐的 $h^{(0)}$ 的部分将最终被丢弃

10.8 回声状态网络

- 定义：回声状态网络，也被称为储层计算

- 来源：从 $\boldsymbol{h}^{(t-1)}$到 $\boldsymbol{h}^{(t)}$的循环权重映射以及从 $\boldsymbol{x}^{(t)}$ 到 $\boldsymbol{h}^{(t)}$ 的输入权重映射是循环网络中最难学习的参数。研究者提出避免这种困难的方法是设定循环隐藏单元，使其能很好地捕捉过去输入历史，并且只学习输出权重

- 思想：类似于核机器，定义一个容器，能够包含足够的讯息并贯穿于整个循环过程，而不需要额外学习，比如它们将任意长度的序列（到时刻 t 的输入历史）映射为一个长度固定的向量（循环状态 $\boldsymbol{h}^{(t)}$ ）

- 方法：最初的想法是使状态到状态转换函数的 Jacobian 矩阵的特征值接近 1。 循环网络的一个重要特征就是 Jacobian 矩阵的特征值谱 $\boldsymbol{J}^{(t)} = \frac{\partial s^{(t)}}{\partial s^{(t-1)}}$。特别重要的是 $\boldsymbol{J}^{(t)}$ 的谱半径（spectral radius），定义为**特征值的最大绝对值**。 Jacobian 矩阵的特征值接近 1，不容易在n步循环之后出现出现**急剧增大或者减小**的情况，进而$\boldsymbol{J}^{(t)}$ 不容易跟随 t 改变而改变

- 回声状态网络的策略：是简单地固定权重使其具有一定的谱半径如 3，其中信息通过时间前向传播，但会由于饱和非线性单元（如 tanh）的稳定作用而不会爆

10.9 渗漏单元和其他多时间尺度的策略

- 处理长期依赖的一种方法是设计工作在多个时间尺度的模型，使模型的某些部 分在细粒度时间尺度上操作并能处理小细节，而其他部分在粗时间尺度上操作并能 把遥远过去的信息更有效地传递过来。存在多种同时构建粗细时间尺度的策略。

- **10.9.1 时间维度的跳跃连接**
- 增加从遥远过去的变量到目前变量的直接连接是得到粗时间尺度的一种方法
- 不过由于梯度可能关于时间步数呈指数消失或爆炸，这种方法使得导数指数减小的速度与 $\frac{\tau}{d}$ 相关而不是 τ， 但梯度仍可能成 t 指数爆炸。这只是允许学习算法捕获更长的依赖性，并不能完全解决问题

- **10.9.2 渗漏单元和一系列不同时间尺度**
- 对某些 v 值应用更新 $\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1-\alpha)v^{(t)}$ 累积一个滑动平均值 $\mu^{(t)}$， 其中 α 是一个从 $\mu^{(t-1)}$ 到 $\mu^{(t)}$线性自连接的例子。当 α 接近 1 时，滑动平均值能记 住过去很长一段时间的信息，而当 α 接近 0，关于过去的信息被迅速丢弃

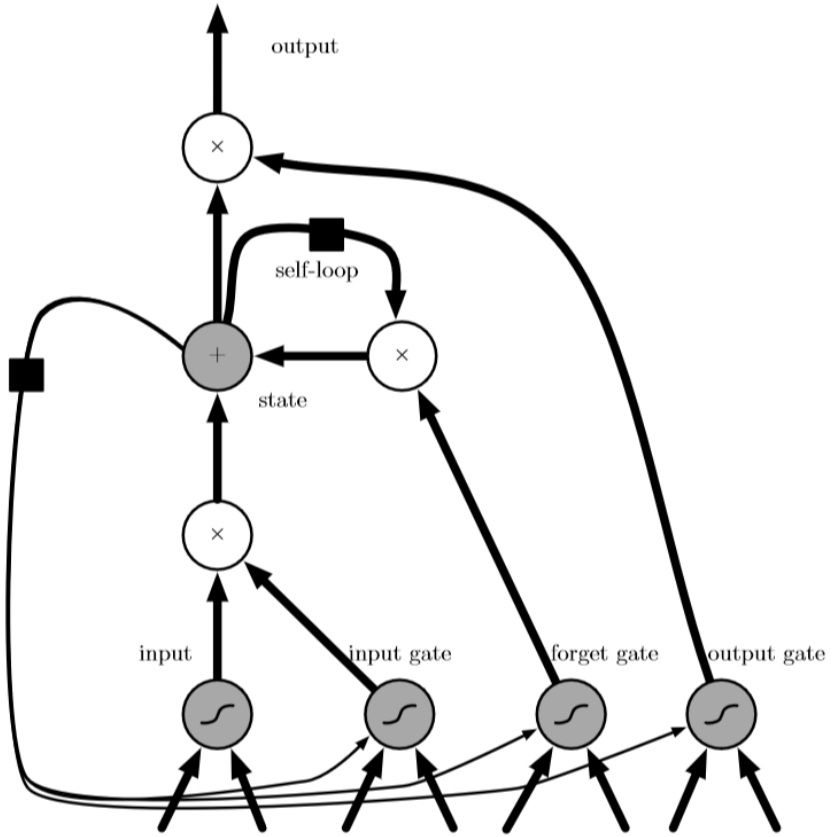
10.9 渗漏单元和其他多时间尺度的策略

- **10.9.3 删除连接**
- 主动删除长度为一的连接并用更长的连接替换它们。以这种方式修改的单元被迫在长时间尺度上运作
- 对比：这种方式是替换边，而之前描述的通过时间跳跃连接是添加边

- **其它**
- 强制一组循环单元在不同时间尺度上运作:
- 一种选择是使循环单元变成渗漏单元，但不同的单元组关联不同的固定时间尺度（α选取不同）
- 另一种选择是使显式且离散的更新发生在不同的时间，不同的单元组有不同的频率。

10.10 长短期记忆和其他门控 RNN

- 实际应用中最有效的序列模型称为**门控 RNN**，包括**基于长短期记忆**（long short-term memory）和**基于门控循环单元**（gated recurrent unit）的网络
- **基于长短期记忆-LSTM**
- 核心思想：引入自循环的巧妙构思，以产生**梯度长时间持续流动**的路径，其中一个关键扩展是使自循环的权重视上下文而定，而不是固定的。
- 应用：无约束手写、语音识别 、手写生成 、机器翻译 、为图像生成标题和解析。
- **基于门控循环单元-GRU**
- 与 LSTM 的主要区别是，单个门控单元同时控制遗忘因子和更新状态单元的决定

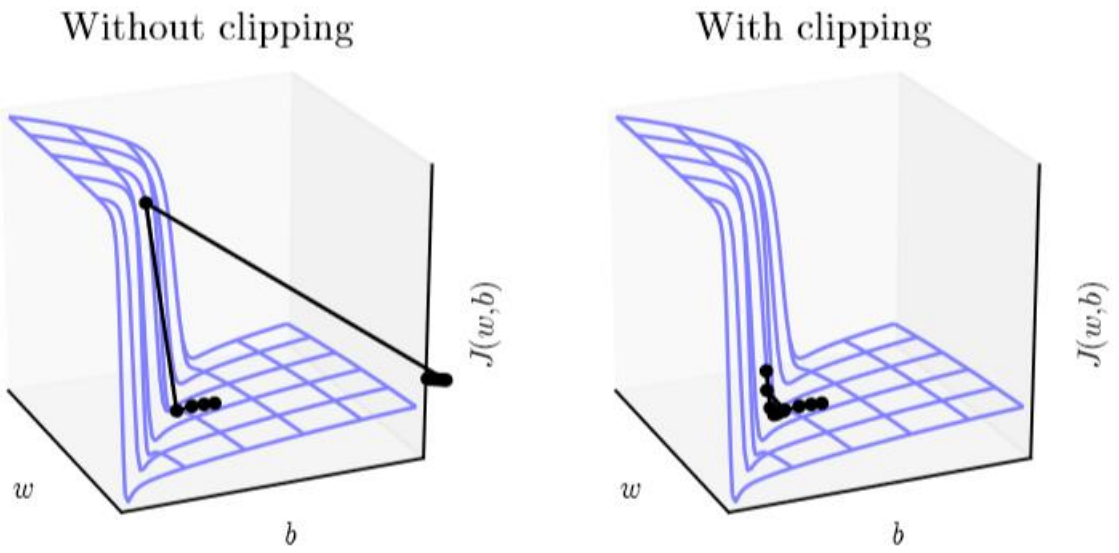


10.11 优化长期依赖

- **10.11.1 阶段梯度**

- 两种方法:
- 一种选择是在参数更新之前，逐元素地截断小批量产生的参数梯度
- 另一种是在参数更新之前截断梯度 $\boldsymbol{g}$ 的范数 $\|\boldsymbol{g}\|$:
- $$if \ \|g\| > v$$
- $$g \leftarrow \frac{gv}{\|g\|}$$

- 结论：后一方法具有的优点是保证了每 个步骤仍然是在梯度方向上的，但实验表明两种形式类似



10.11 优化长期依赖

- **10.11.2 引导信息流正则化**

- 来源：梯度截断有助于处理爆炸的梯度，但它无助于消失的梯度
- 正则化引导“信息流”，通过惩罚机制控制权重参数的过度增长或下降

- 目标： 形式上，我们要使：
- $(\nabla_{h^{(t)}} L)\ \frac{\partial h^{(t)}}{\partial h^{(t-1)}}$ 与$\nabla_{h^{(t)}} L$ 一样大
- 在这个目标下，Pascanu et al. (2013a) 提出以下正则项：

$$\Omega = \sum_t \left(\frac{\left\| (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \right\|}{\|\nabla_{\boldsymbol{h}^{(t)}} L\|} - 1 \right)^2 .$$

- 这种方法的一个主要弱点是，在处理数据冗余的任务时如语言模型，它并不 像 LSTM 一样有效。

10.12 外显记忆

- 来源：智能需要知识并且可以通过学习获取知识。然而知识种类繁多，有些知识是隐含的、潜意识的并且难以用语言表达——比如怎么行走或狗与猫的样子有什么不同。其他知识可以是明确的、可陈述的以及可以相对简单地使用词语表达——每天常识性的知识，如 ‘‘猫是一种动物’’，或者为实现自己当前目标所需知道的非常具体的事实，如 ‘‘与销售团队会议在 141 室于下午 3:00 开始’’。

- 问题描述：神经网络擅长存储隐性知识，但是他们很难记住事实。

- 解决方法：记忆网络
- 记忆网络中，包括一组可以通过寻址机制来访问的记忆单元，通过寻址实现对信息的复现。 。Graves et al. (2014) 引入的神经网络图灵机 （NTM），不需要明确的监督指示采取哪些行动而能学习从记忆单元读写任意内容

- 新问题：整数地址的函数很难优化
- 解决办法： NTM 实际同时从多个记忆单元写入或读取。读取时，它们采取许多单元的加权平均值。写入时，他们对 多个单元修改不同的数值。

10.12 外显记忆

- 记忆单元的形式:
- 通常包含向量，而不是标量，原因有二:
- 1 对于目标系数聚集的小单元，读取向量有助于减少部分访问成本
- 2 读取向量允许基于内容的寻址，简单的例子就是通过一句歌词来查找相应歌曲

- 图中可以看到与存储器耦接的 “任务神经网络”。虽然这一任务神经网络可以是前馈或循环的，但整个系统也可以看成是一个循环网络。