序列建模：循环和递归网络

Xing Tang

May 10, 2017

目录

循环神经网络的基本概念

循环神经网络(recurrent neural network)或 RNN (Rumelhart et al., 1986c) 是一类用于处理序列数据的神经网络。

- 卷积神经网络
  - 专门用于处理网格化数据
  - 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
- 循环神经网络
  - 专门用于处理序列 $x(1),...,x(\tau)$ 的神经网络
  - 大多数循环网络也能处理可变长度的序列

循环神经网络的基本概念

循环神经网络的特点：

- 从多层网络出发到循环网络，在模型的不同部分共享参数
- 模型能够扩展到不同形式的样本(这里指不同长度的样本)并进行泛化
- 在序列上的操作，该序列在时刻t（从1到τ）包含向量x(t)
- 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

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。本节，我们对展开(unfolding)递归或循环计算得到的重复结构进行解释，这些重复结构通常对应于一个事件链。例如动态系统的经典形式：

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

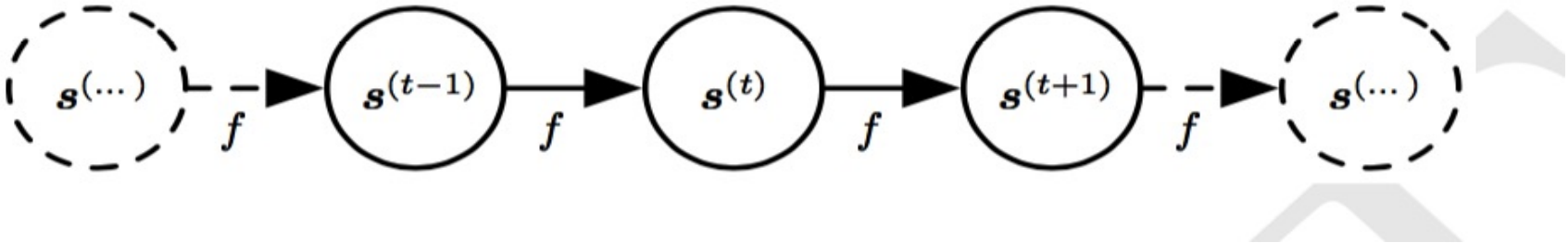
其中s称为系统的状态，s在时刻t的定义需要参考时刻t-1的同样的定义，因此公式（1）是循环的。

展开计算图

对有限时间步 τ，$\tau-1$ 次应用这个定义可以展开这个图。例如 $\tau = 3$, 我们对式 (1) 展开, 可以得到:

$$s^{(3)} = f(s^{(2)}; \theta) \tag{2}$$

$$s^{(3)} = f(f(s^{(1)}; \theta); \theta) \tag{3}$$



将式(1)描述的经典动态系统表示为展开的计算图。每个节点表示在某个时刻t的状态, 并且函数f将t处的状态映射到t+1处的状态。所有时间步都使用相同的参数(用于参数化f的相同 θ 值)。

展开计算图

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。为了表明状态是网络的隐藏单元，我们使用变量h代表状态，典型 RNN 会增加额外的架构特性，如读取状态信息 h 进行预测的 输出层。

$$\mathrm{h}^{(\mathrm{t})} = \mathrm{f}(\mathrm{h}^{(\mathrm{t}-1)}; \mathrm{x}^{(\mathrm{t})}, \theta) \tag{4}$$

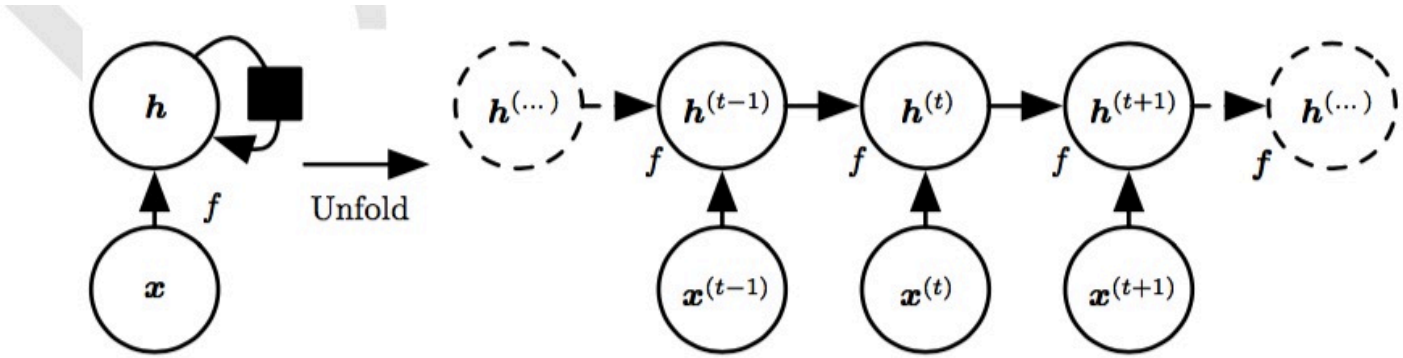


图 10.2: 没有输出的循环网络。此循环网络只处理来自输入 $\boldsymbol{x}$ 的信息，将其合并到经过时间向前传播的状态 $\boldsymbol{h}$。*(左)* 回路原理图。黑色方块表示单个时间步的延迟。*(右)* 同一网络被视为展开的计算图，其中每个节点现在与一个特定的时间实例相关联。

展开计算图

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，因为它指定的是从一种状态到另一种状态的转移，而不是在可变长度的历史状态上操作
- 我们可以在每个时间步使用相同参数的相同转移函数f

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(没有出现在训练集中)，并且估计模型所需的训练样本远远少于不带参数共享的模型。

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
- 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
- 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

循环神经网络

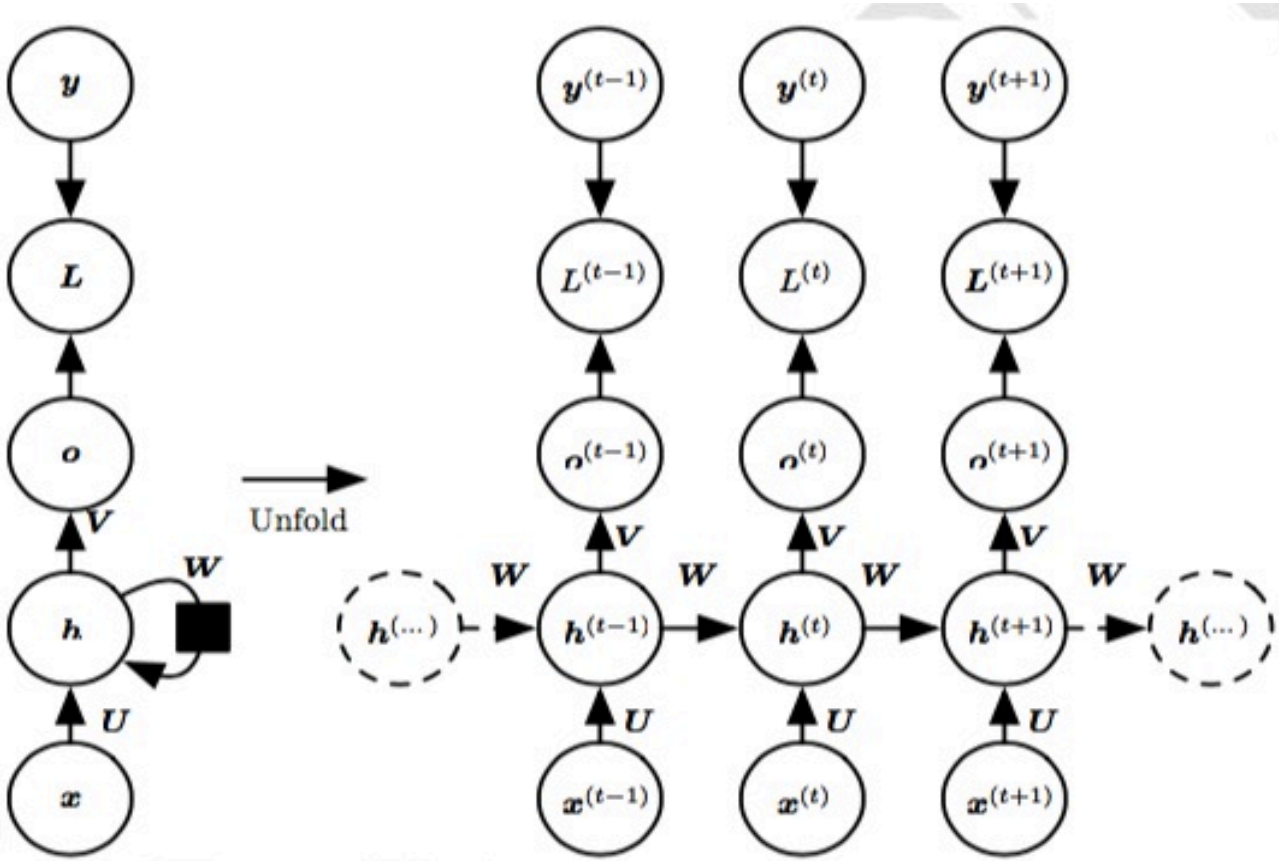


图 10.3: 计算循环网络(将 $\boldsymbol{x}$ 值的输入序列映射到输出值 $\boldsymbol{o}$ 的对应序列) 训练损失的计算图。损失 L 衡量每个 $\boldsymbol{o}$ 与相应的训练目标 $\boldsymbol{y}$ 的距离。当使用 softmax 输出时，我们假设 $\boldsymbol{o}$ 是未归一化的对数概率。损失 L 内部计算 $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{o})$，并将其与目标 $\boldsymbol{y}$ 比较。RNN输入到隐藏的连接由权重矩阵 $\boldsymbol{U}$ 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。式 (10.8) 定义了该模型中的前向传播。*(左)* 使用循环连接绘制的 RNN 和它的损失。*(右)* 同一网络被视为展开的计算图，其中每个节点现在与一个特定的时间实例相关联。

循环神经网络

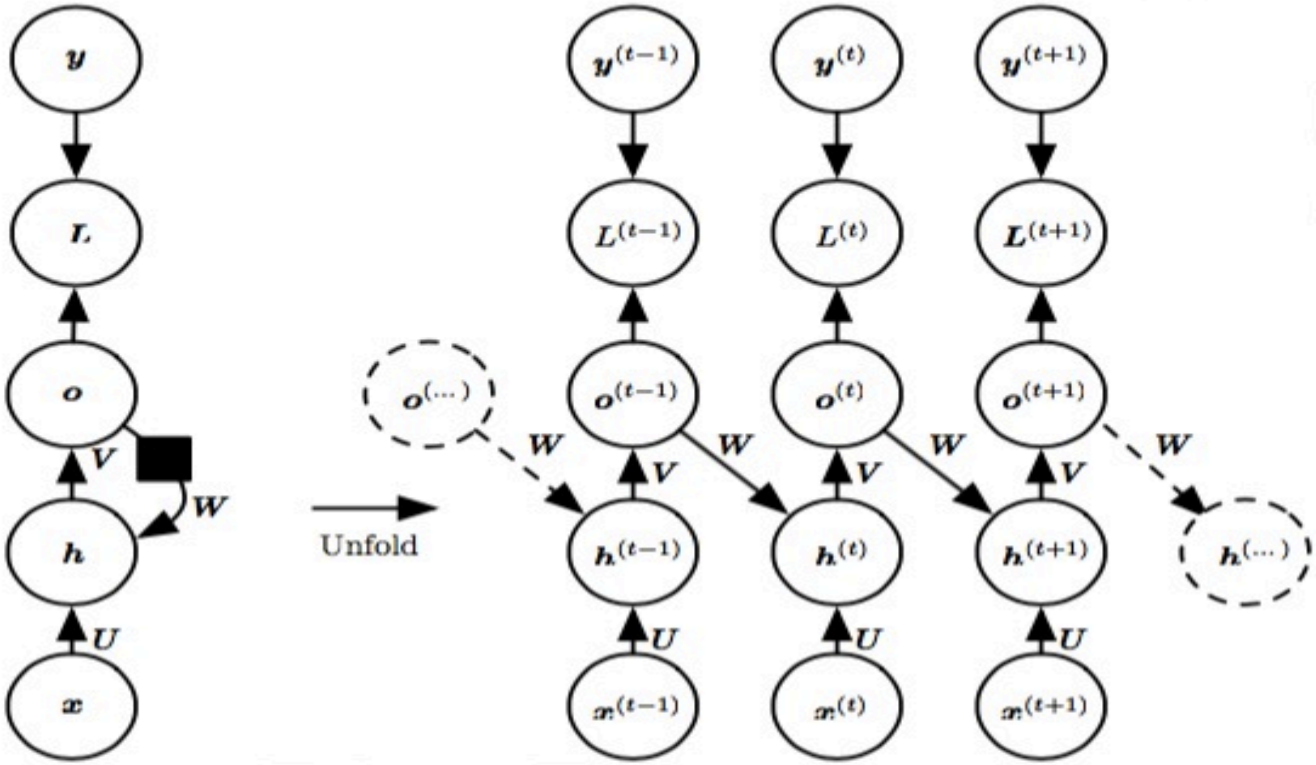


图 10.4: 此类 RNN 的唯一循环是从输出到隐藏层的反馈连接。在每个时间步 t，输入为 $\boldsymbol{x}_t$，隐藏层激活为 $\boldsymbol{h}^{(t)}$，输出为 $\boldsymbol{o}^{(t)}$，目标为 $\boldsymbol{y}^{(t)}$，损失为 $L^{(t)}$。*(左)* 回路原理图。*(右)* 展开的计算图。这样的 RNN 没有图 10.3 表示的 RNN 那样强大（只能表示更小的函数集合）。图 10.3 中的 RNN 可以选择将其想要的关于过去的任何信息放入隐藏表示 $\boldsymbol{h}$ 中并且将 $\boldsymbol{h}$ 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。此处没有从 $\boldsymbol{h}$ 前向传播的直接连接。之前的 $\boldsymbol{h}$ 仅通过产生的预测间接地连接到当前。$\boldsymbol{o}$ 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，因为每个时间步可以与其他时间步分离训练，允许训练期间更多的并行化，如第 10.2.1 节所述。

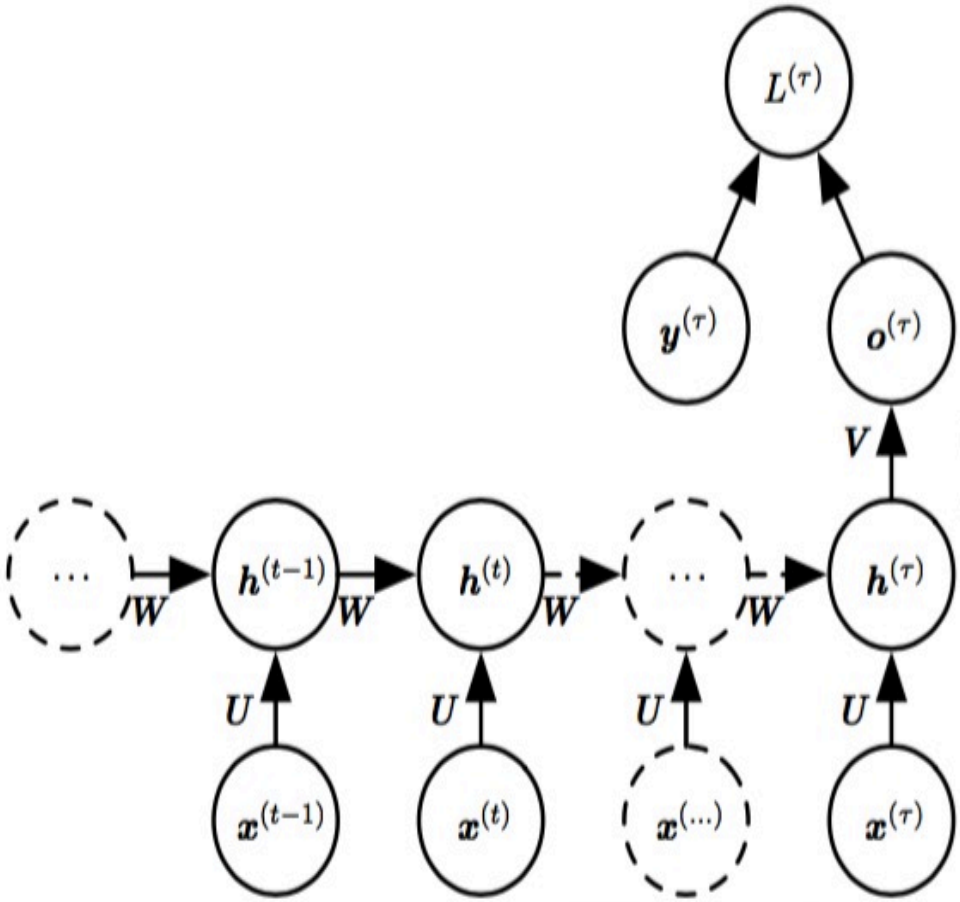


图 10.5: 关于时间展开的循环神经网络，在序列结束时具有单个输出。这样的网络可以用于概括序列并产生用于进一步处理的固定大小的表示。在结束处可能存在目标（如此处所示），或者通过更下游模块的反向传播来获得输出 $\boldsymbol{o}^{(t)}$ 上的梯度。

现在我们研究第一种网络的前向传播公式，假设使用双曲正切激活函数，输出是离散的,如用于预测词或字符的 RNN

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}, \tag{5}$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)}), \tag{6}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}, \tag{7}$$

$$\check{\mathbf{y}}^{(t)} = \mathrm{softmax}(\mathbf{o}^{(t)}), \tag{8}$$

其中的参数的偏置向量b和c连同权重矩阵U、V和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x序列配对的y的总损失就是所有时间步的损失之和。例如，L(t)为给定的x(1),...,x(t) 后y(t) 的负对数似然，则

$$2L(\{x^{(1)}\},...,\{x^{(\tau)}\},\{y^{(1)}\},...,\{y^{(\tau)}\}) \tag{9}$$

$$= \sum_t L^{(t)} \tag{10}$$

$$= -\sum_t \log p_{model}(y^{(t)}|\{x^{(1)}\},...,\{x^{(t)}\}) \tag{11}$$

梯度计算涉及执行一次前向传播，接着是由右到左的反向传播。运行时间是O(τ)。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(BPTT)。因此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。

导师驱动过程和输出循环网络

由输出反馈到模型而产生循环连接的模型可用导师驱动过程进行训练。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

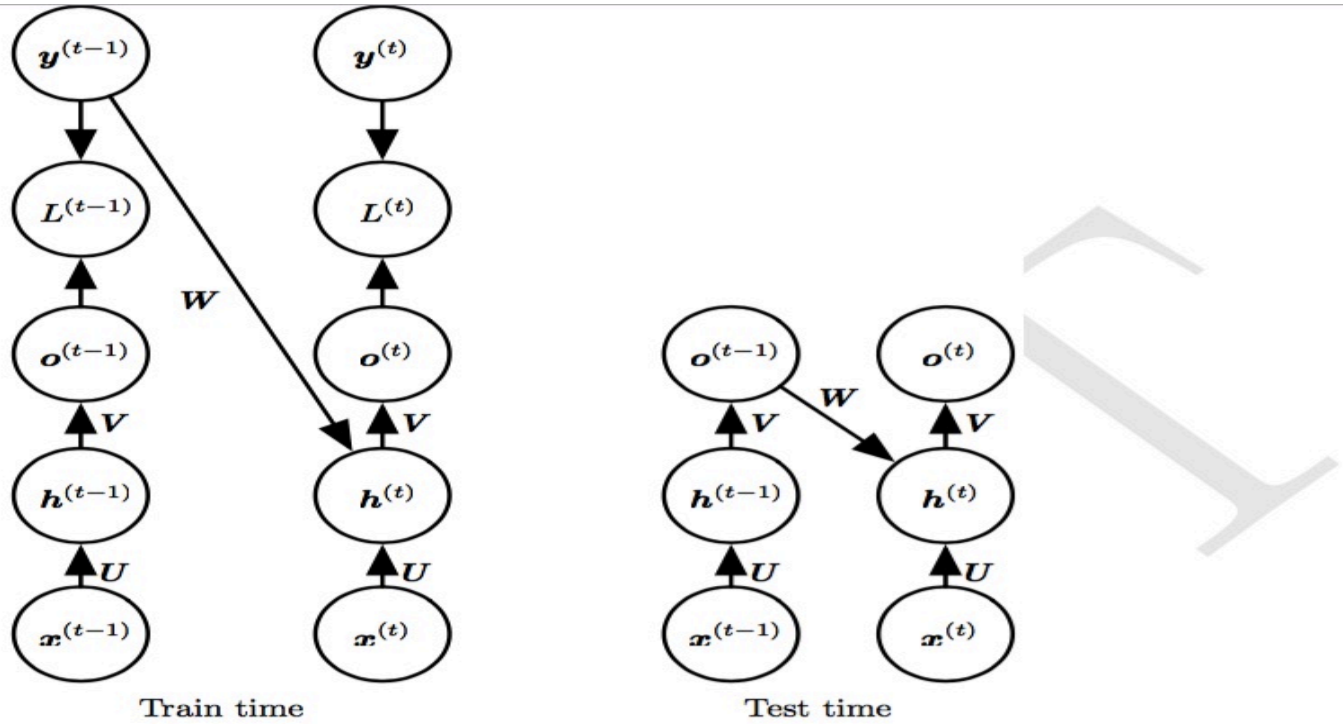


图 10.6: 导师驱动过程的示意图。导师驱动过程是一种训练技术，适用于输出与下一时间步的隐藏状态存在连接的 RNN。*(左)* 训练时，我们将训练集中正确的输出 $\boldsymbol{y}^{(t)}$ 反馈到 $\boldsymbol{h}^{(t+1)}$。*(右)* 当模型部署后，真正的输出通常是未知的。在这种情况下，我们用模型的输出 $\boldsymbol{o}^{(t)}$ 近似正确的输出 $\boldsymbol{y}^{(t)}$，并反馈回模型。

导师驱动过程和输出循环网络

- 优点是可以并行化，加快训练速度
- 因为缺乏隐藏到隐藏的循环连接，表示能力下降
- 训练期间该网络看到的输入与测试时看到的会有很大的不同

计算循环神经网络的梯度

为了获得BPTT算法行为的一些直观理解，我们举例说明如何通过BPTT计算RNN的梯度。计算图的节点包括参数U, V, W, b和c，以及以t为索引的节点序列x(t), h(t), o(t)和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。我们从紧接着最终损失的节点开始递归：

$$\frac{\partial L}{\partial L^{(t)}} = 1 \tag{12}$$

我们假设输出o(t)作为softmax函数的参数，对于所有i, t，关于时间步t输出的梯度如下：

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \check{y}_i^{(t)} - 1_{i,y^{(t)}} \tag{13}$$

计算循环神经网络的梯度

在最后时间步 τ，h(τ)只有o(τ)作为后续节点，因此这个梯度很简单：

$$\nabla_{h(\tau)} L = V^T \nabla_{o(\tau)} L \tag{14}$$

然后，我们可以从时刻t= τ −1到t=1反向迭代，通过时间反向传播梯度，梯度计算公式如下：

$$\begin{aligned} \nabla_{h(t)} L &= \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}^T\right)(\partial h^{(t+1)} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^T (\nabla_{o(t)} L) && (15) \\ &= (\partial h^{(t+1)} L)\,\mathrm{diag}\left(1 - (h^{(t+1)})^2\right) + V^T \nabla_{o(t)} L && (16) \end{aligned}$$

计算循环神经网络的梯度

参数的梯度可以由下式给出：

$$\nabla_c L = \sum_t \left(\frac{\partial o^{(t)}}{\partial c}\right)^T \nabla_{o^{(t)}} L = \sum_t \nabla_{o^{(t)}} L \tag{17}$$

$$\nabla_b L = \sum_t \left(\frac{\partial h^{(t)}}{\partial b^{(t)}}\right)^T \nabla_{h^{(t)}} L = \sum_t \mathrm{diag}\left(1 - (h^{(t)})^2\right) \nabla_{h^{(t)}} L \tag{18}$$

$$\nabla_V L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_V o_i^{(t)} = \sum_t (\nabla_{o^{(t)}} L) h^{(t)T} \tag{19}$$

$$\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{W^{(t)}} h_i^{(t)} \tag{20}$$

$$= \sum_t \mathrm{diag}\left(1 - (h^{(t)})^2\right) (\nabla_{h^{(t)}} L) h^{(t-1)T} \tag{21}$$

$$\nabla_U L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{U^{(t)}} h_i^{(t)} \tag{22}$$

$$= \sum \mathrm{diag}\left(1 - (h^{(t)})^2\right) (\nabla_{h^{(t)}} x^{(t)T}) \tag{23}$$

双向 RNN

- 目前为止我们考虑的所有循环神经网络有一个“因果”结构，意味着在时刻t的状态只能从过去的序列x(1), ..., x(t−1)以及当前的输入x(t)捕获信息
- 在许多应用中，我们要输出的y(t)的预测可能依赖于整个输入序列，如语音识别、手写识别

双向 RNN

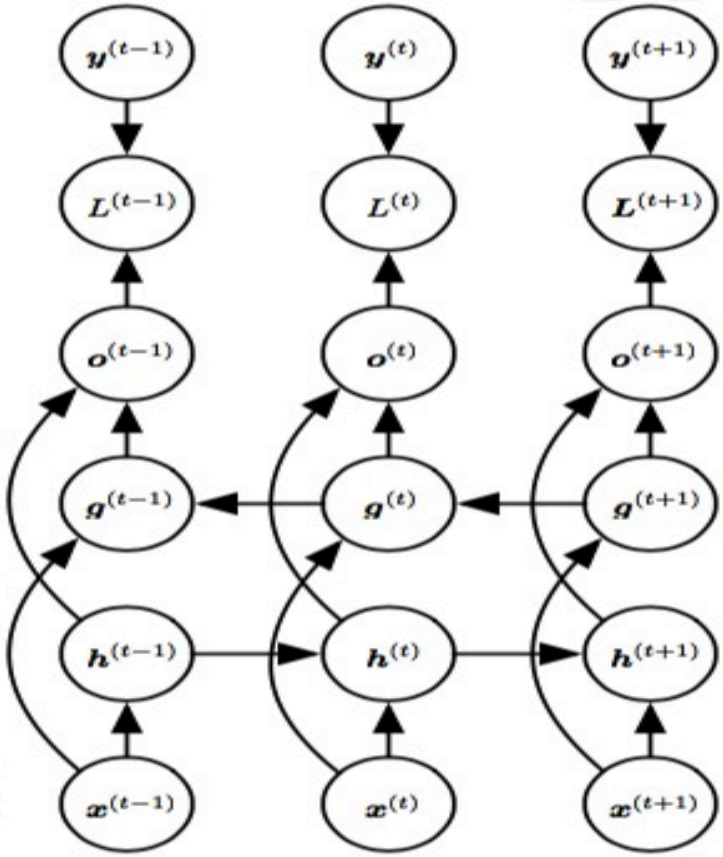


图 10.11：典型的双向循环神经网络中的计算，意图学习将输入序列 $\boldsymbol{x}$ 映射到目标序列 $\boldsymbol{y}$（在每个步骤 t 具有损失 $L^{(t)}$）。循环性 $\boldsymbol{h}$ 在时间上向前传播信息（向右），而循环性 $\boldsymbol{g}$ 在时间上向后传播信息（向左）。因此在每个点 t，输出单元 $\boldsymbol{o}^{(t)}$ 可以受益于输入 $\boldsymbol{h}^{(t)}$ 中关于过去的相关概要以及输入 $\boldsymbol{g}^{(t)}$ 中关于未来的相关概要。

基于编码-解码的序列到序列架构

这在许多场景中都有应用,如语音识别、机器翻译或问答,其中训练集的输入和输出序列的长度通常不相同,我们经常将RNN的输入称为“上下文”。我们希望产生此上下文的表示C。这个上下文C可能是一个概括输入序列 X=(x(1),...,x(nx)) 的向量或者向量序列。

- 编码器RNN处理输入序列，编码器输出上下文C
- 解码器RNN则以固定长度的向量为条件产生输出序列 Y=(y(1),...,y(ny))

基于编码-解码的序列到序列架构

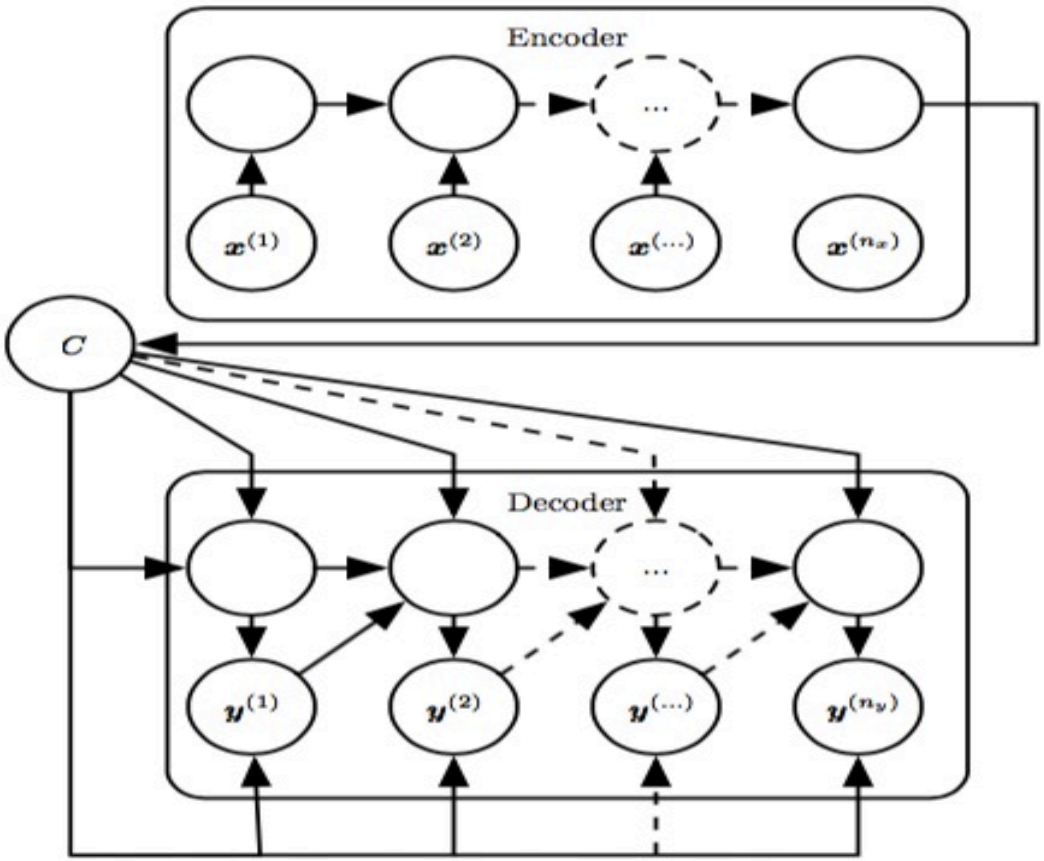


图 10.12: 在给定输入序列 $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n_x)})$ 的情况下学习生成输出序列 $(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n_y)})$ 的编码器-解码器或序列到序列的 RNN 架构的示例。它由读取输入序列的编码器 RNN 以及生成输出序列（或计算给定输出序列的概率）的解码器 RNN 组成。编码器 RNN 的最终隐藏状态用于计算一般为固定大小的上下文变量 C，C 表示输入序列的语义概要并且作为解码器 RNN 的输入。

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:

- 从输入到隐藏状态
- 从前一隐藏状态到下一隐藏状态
- 从隐藏状态到输出

这三个块都与单个权重矩阵相关联。换句话说,当网络被展开时,每个块对应一个浅的变换。

深度循环网络

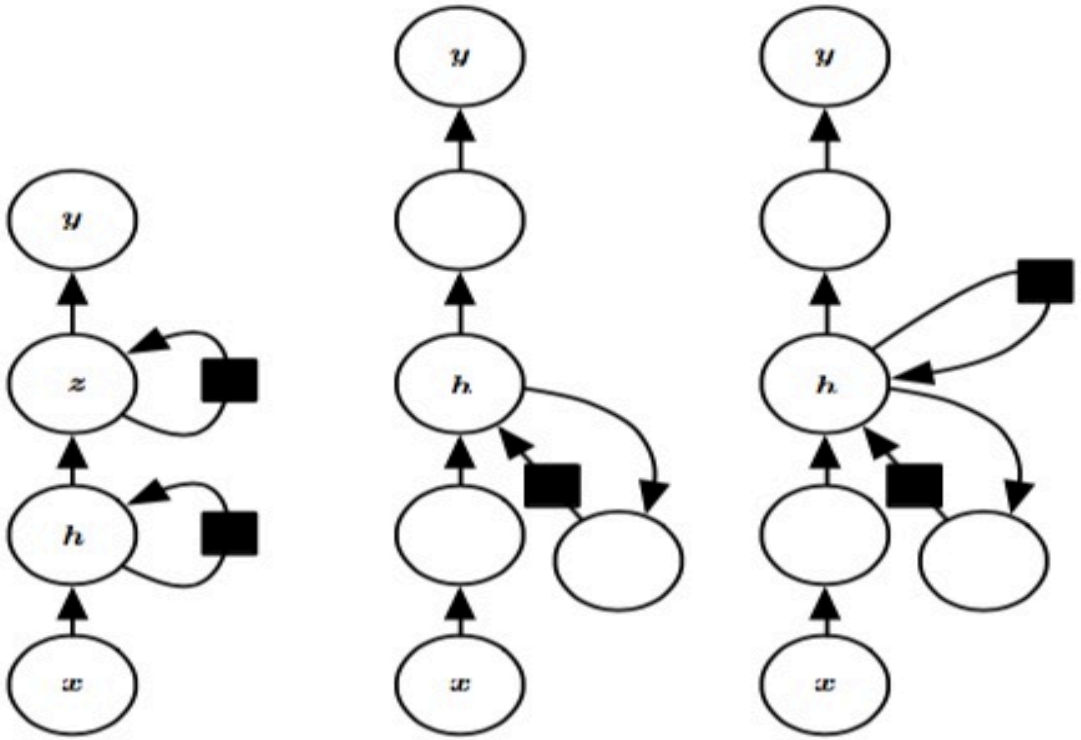


图 10.13: 循环神经网络可以通过许多方式变得更深 (Pascanu *et al.*, 2014a)。(a) 隐藏循环状态可以被分解为具有层次的组。(b) 可以向输入到隐藏，隐藏到隐藏以及隐藏到输出的部分引入更深的计算 (如 MLP)。这可以延长链接不同时间步的最短路径。(c) 可以引入跳跃连接来缓解路径延长的效应。

递归神经网络

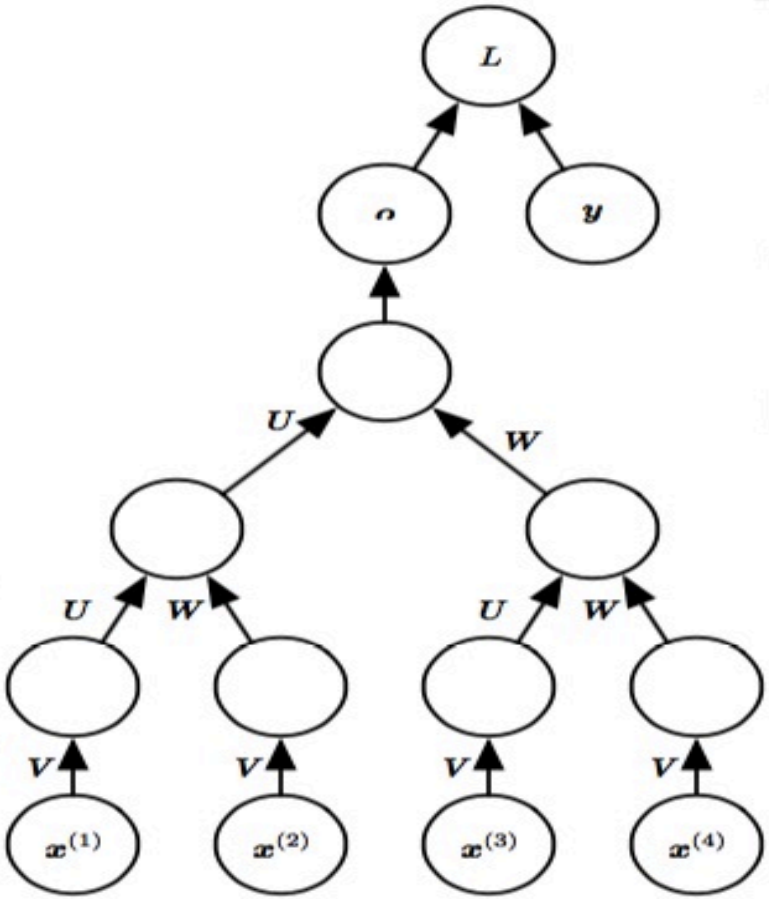


图 10.14: 递归网络将循环网络的链状计算图推广到树状计算图。可变大小的序列 $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ 可以通过固定的参数集合（权重矩阵 $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$）映射到固定大小的表示（输出 $\boldsymbol{o}$）。该图展示了监督学习的情况，其中提供了一些与整个序列相关的目标 $\boldsymbol{y}$。

长期依赖的挑战

学习循环网络长期依赖的数学挑战：

- 梯度倾向于消失(大部分情况)
- 梯度爆炸(很少情况)

许多资料提供了更深层次的讨论 (Hochreiter, 1991a; Doya, 1993; Bengio et al., 1994a; Pascanu et al., 2013a)。实验表明,当我们增加了需要捕获的依赖关系的跨度, 基于梯度的优化变得越来越困难,SGD 在长度仅为10或20的序列上成功训练传统RNN的概率迅速变为 0。

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。

- 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得到粗时间尺度
- 渗漏单元，我们对某些u值应用更新 $\mu(t) \leftarrow \alpha\,\mu(t-1)+(1-\alpha)u(t)$ 累积一个滑动平均值 $\mu(t)$，其中 α 是一个从 $\mu(t-1)$ 到 $\mu(t)$ 线性自连接的权重，使用权重接近1的线性自连接是确保该单元可以访问过去值的不同方式。线性自连接通过调节实值 α 更平滑灵活地调整这种效果，而不是调整整数值的跳跃长度。
- 删除连接，处理长期依赖另一种方法是在多个时间尺度组织RNN状态的想法，这个想法与之前讨论的时间维度上的跳跃连接不同，因为它涉及主动删除长度为一的连接并用更长的连接替换它们。

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（LSTM）和基于门控循环单元（GRU）的网络。

- 渗漏单元一样，门控RNN想法也是基于生成通过时间的路径，其中导数既不消失也不发生爆炸
- 渗漏单元允许网络在较长持续时间内积累信息，一旦该信息被使用，门控RNN能够有效的遗忘旧的状态

长短期记忆 LSTM

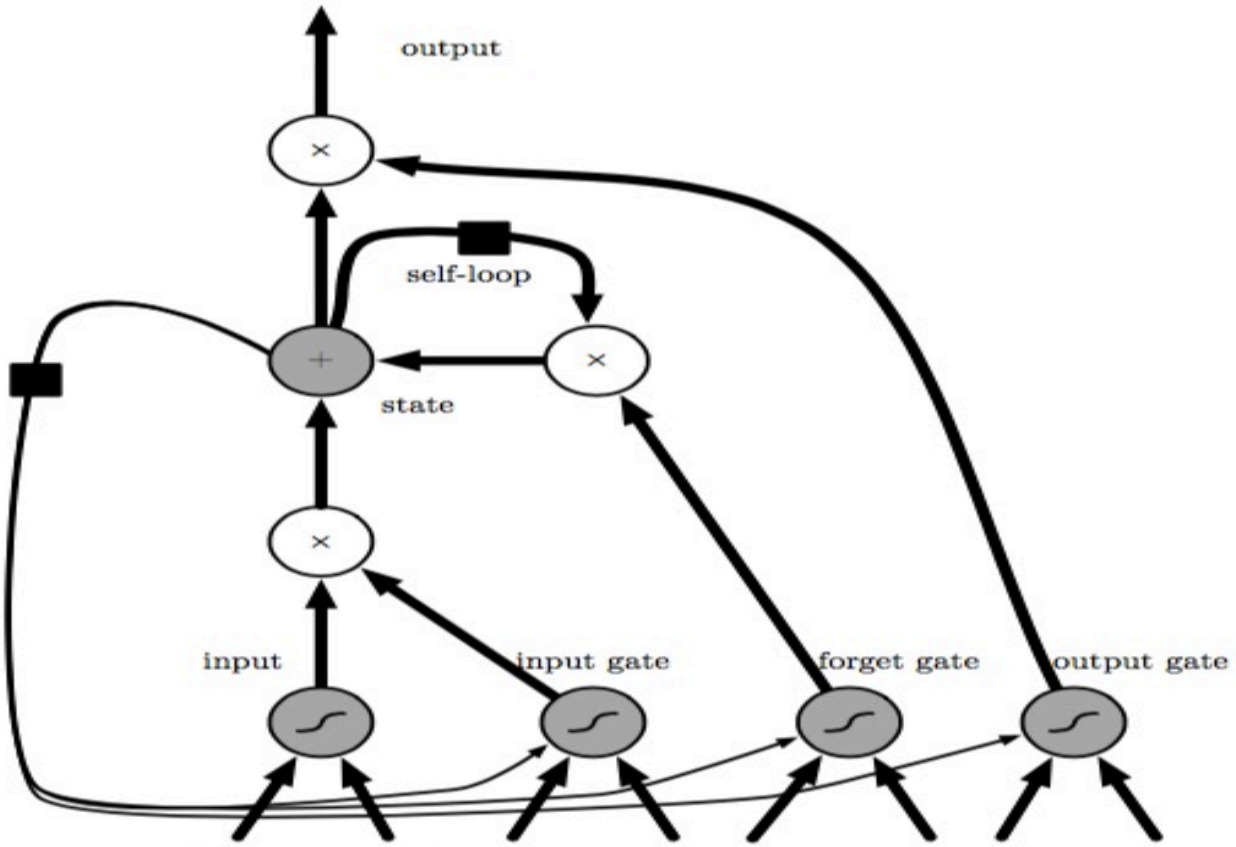


图 10.16: LSTM 循环网络“细胞” 的框图。细胞彼此循环连接，代替一般循环网络中普通的隐藏单元。这里使用常规的人工神经元计算输入特征。如果 sigmoid 输入门允许，它的值可以累加到状态。状态单元具有线性自循环，其权重由遗忘门控制。细胞的输出可以被输出门关闭。所有门控单元都具有 sigmoid 非线性，而输入单元可具有任意的压缩非线性。状态单元也可以用作门控单元的额外输入。黑色方块表示单个时间步的延迟。

控制线性自环权重的遗忘门的计算公式：

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \tag{25}$$

状态更新公式如下：

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{26}$$

外部输入门和遗忘门类似，不过有独立的参数：

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{27}$$

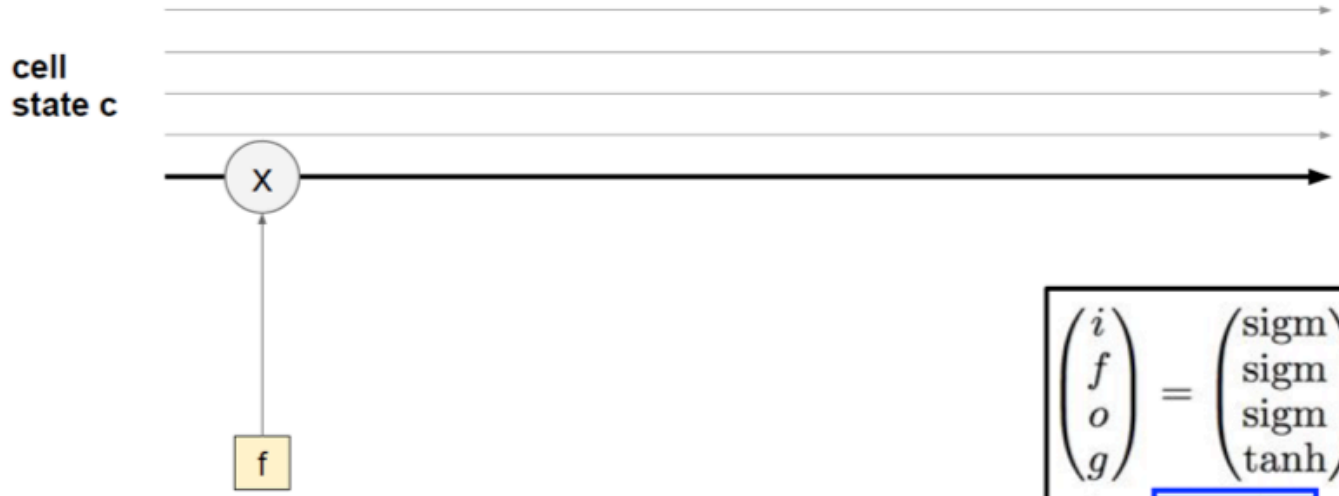
LSTM细胞的输出是由输出门控制：

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)} \tag{28}$$

$$q_i^{(t)} = \sigma\left(b_i^q + \sum_j U_{i,j}^q x_j^{(t)} + \sum_j W_{i,j}^q h_j^{(t-1)}\right) \tag{29}$$

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



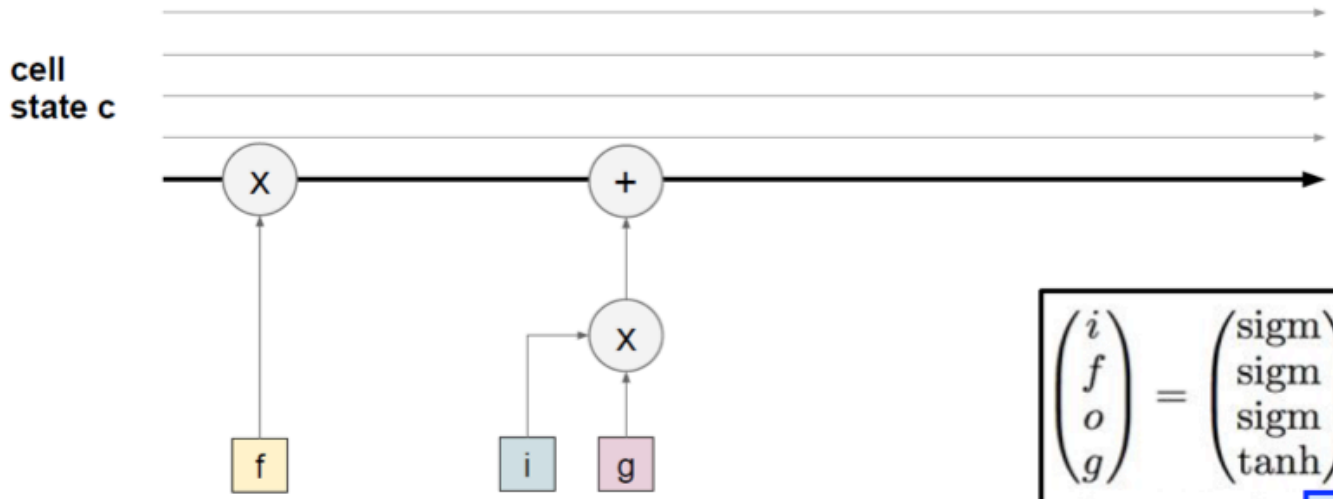
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

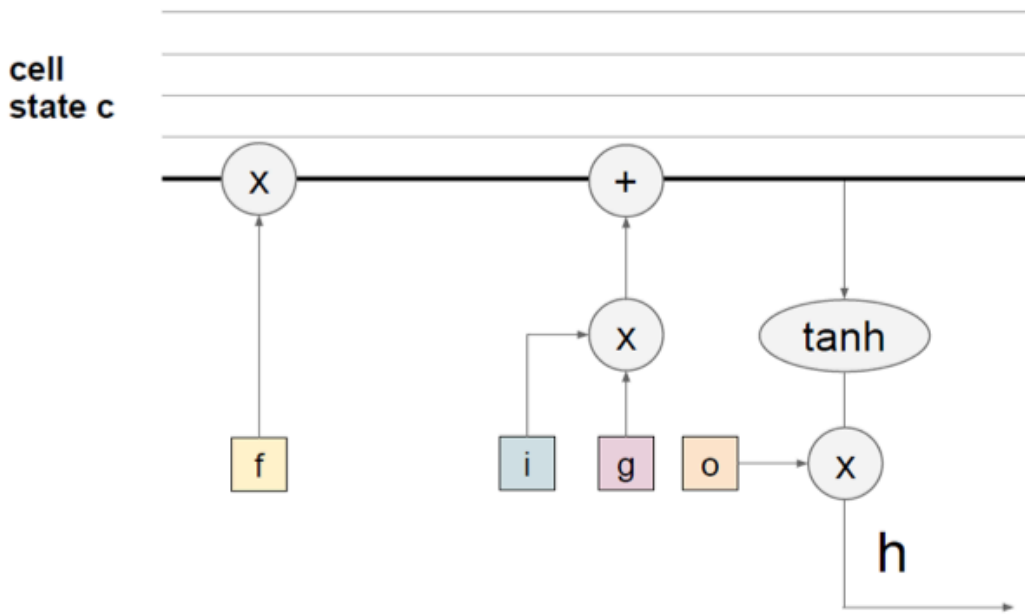
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 10 - 72 8 Feb 2016

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

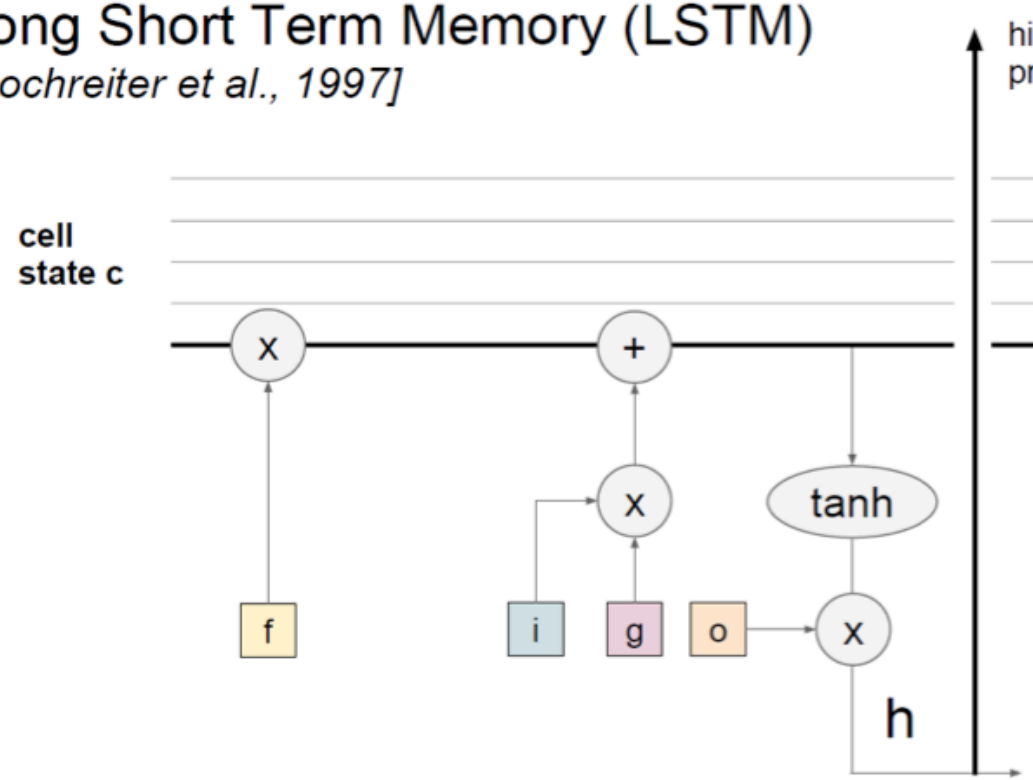
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 10 - 73 8 Feb 2016

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



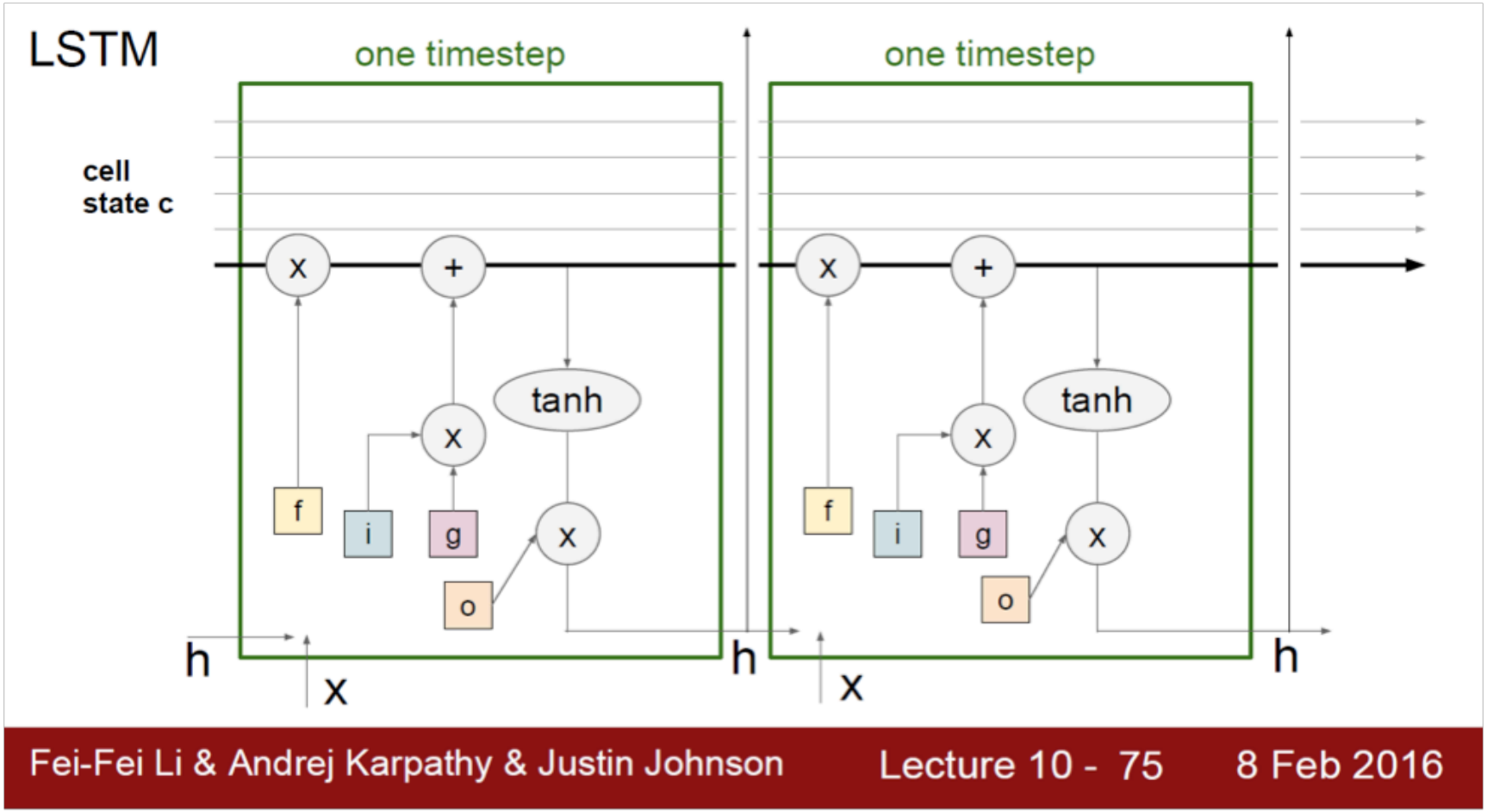
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

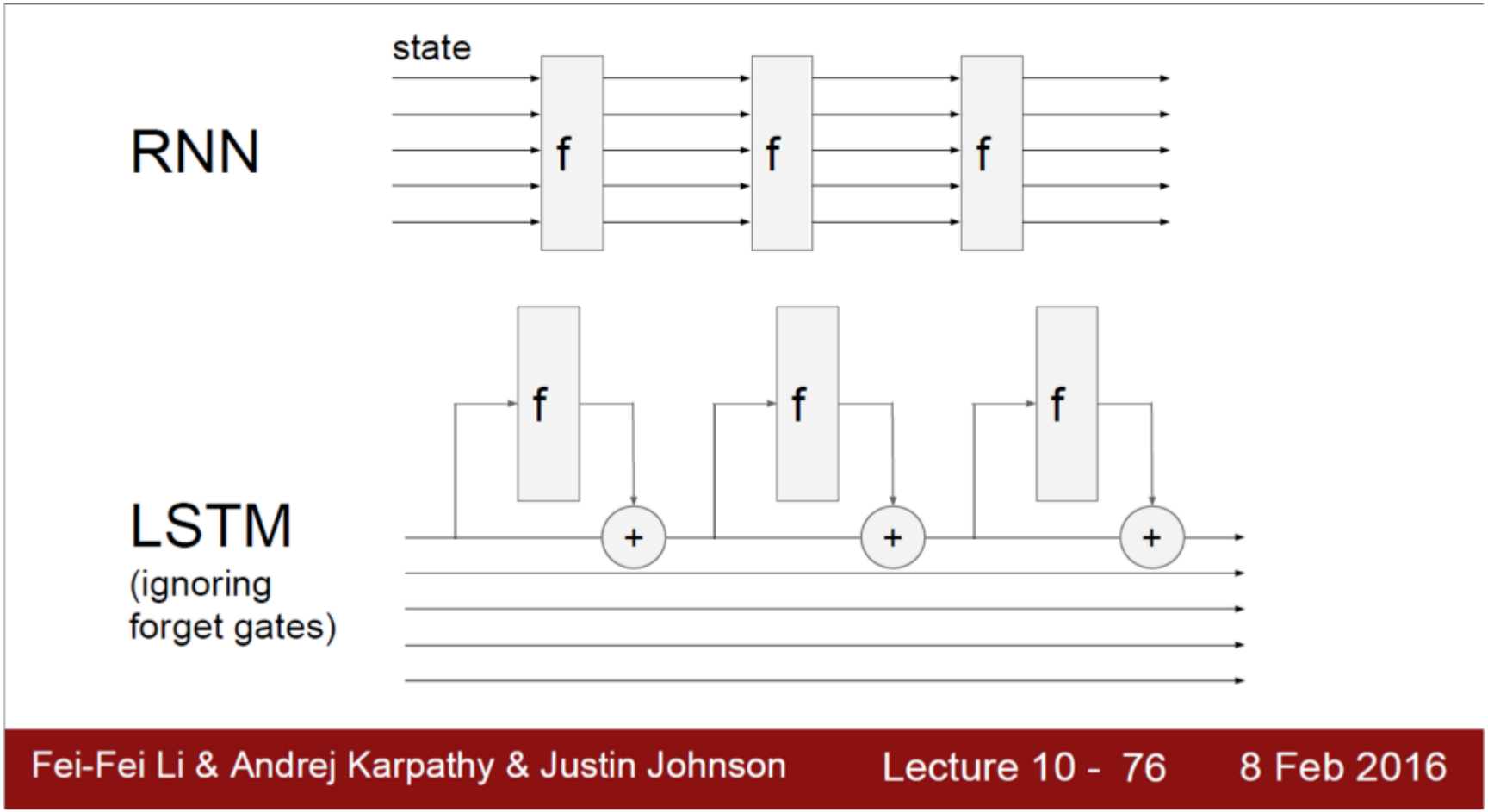
$$h_t^l = o \odot \tanh(c_t^l)$$

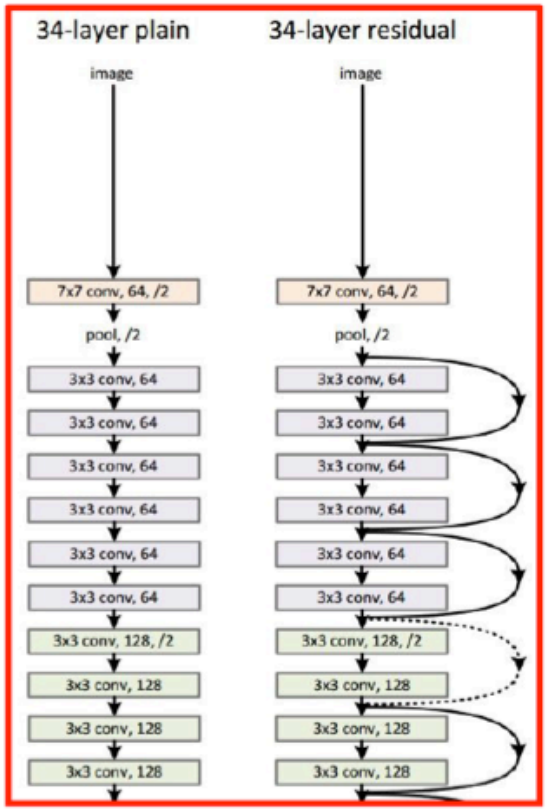
Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 10 - 74 8 Feb 2016

长短期记忆 LSTM



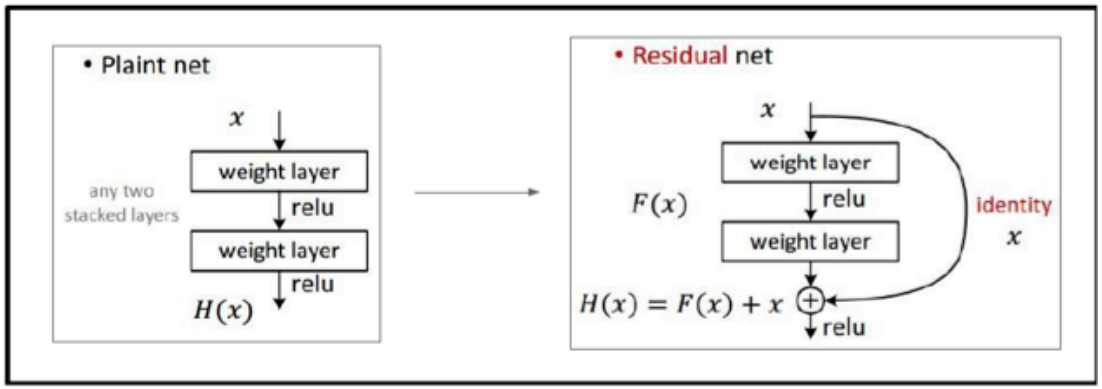
Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 10 - 75 8 Feb 2016


RNN
state
f
f
f
LSTM
(ignoring
forget gates)
f
f
f
+
+
+
Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 10 - 76
8 Feb 2016



Recall:
"PlainNets" vs. ResNets

ResNet is to PlainNet what LSTM is to RNN, kind of.



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，更新公式如下：

$$h_i^{(t)} = u_i^{(t-1)} h_i(t-1) + (1 - u_i^{(t-1)}) \sigma(b_i + \sum_j U_{i,j} x_j^{(t)} \tag{30}$$

$$+ \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)}) \tag{31}$$

其中u代表“更新”门，r表示“复位”门，它们的值就如通常所定义的：

$$u_i^{(t)} = \sigma(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t-1)}) \tag{32}$$

$$r_i^{(t)} = \sigma(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t-1)}) \tag{33}$$

复位和更新门能独立地“忽略”状态向量的一部分。

外显记忆

智能需要知识并且可以通过学习获取知识，这已促使大型深度架构的发展，知识是不同的并且种类繁多

- 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 - 如“与销售团队会议在141室于下午3:00开始”

外显记忆

神经网络擅长存储隐性知识,但是他们很难记住事实。被存储在神经网络参数中之前,随机梯度下降需要多次提供相同的输入,即使如此,该输入也不会被特别精确地存储。推测这是因为神经网络缺乏工作存储系统,人类为实现一些目标而明确保存和操作相关信息片段的系统。这种外显记忆组件将使我们的系统不仅能够快速“故意”地存储和检索具体的事实，也能利用他们循序推论

- 记忆网络(memory network)，其中包括一组可以通过寻址机制来访问的记忆单元，记忆网络需要监督信号指示他们如何使用自己的记忆单元
- 神经网络图灵机(Bahdanau et al. (2015),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,并通过使用基于内容的软注意机制允许端到端的训练

外显记忆

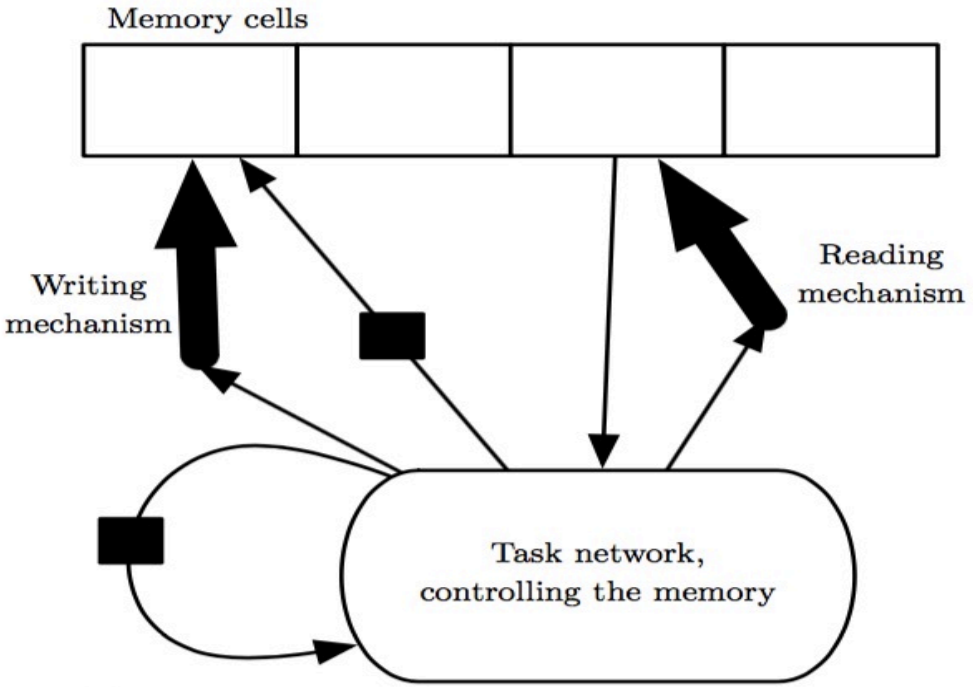


图 10.18: 具有外显记忆网络的示意图，具备神经网络图灵机的一些关键设计元素。在此图中，我们将模型的“表示”部分（“任务网络”，这里是底部的循环网络）与存储事实的模型（记忆单元的集合）的“存储器”部分区分开。任务网络学习“控制”存储器，决定从哪读取以及在哪写入（通过读取和写入机制，由指向读取和写入地址的粗箭头指示）。

主要内容

- 循环神经网络的基本概念
- 展开计算图
- 循环神经网络
- 导师驱动过程和输出循环网络
- 常用的循环神经网络结构
- 其他门控RNN和LSTM
- 优化长期依赖
- 外显记忆

引用

http://cs231n.stanford.edu/syllabus.html
https://exacity.github.io/deeplearningbook-chinese

Wechat Official Accounts: machinelearning
Wechat Admin ID: ml-admin