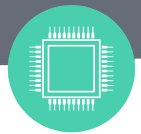
深度学习之 PyTorch 实战

循环神经网络 part2


主讲老师: 土豆老师

目录

01 通过时间反向传播

02 门控循环单元（GRU）

03 长短期记忆网络（LSTM）

04 深度循环神经网络

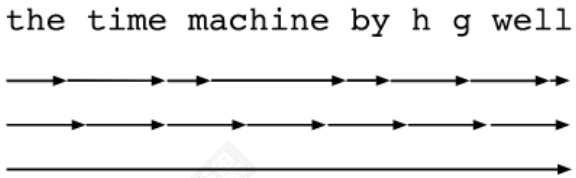
05 双向循环神经网络

通过时间反向传播

- 在上一讲中，如果不裁剪梯度，模型将无法正常训练。为了深刻理解这一现象，本讲将介绍循环神经网络中梯度的计算和存储方法，即通过时间反向传播（back-propagation through time）。
- 我们在「正向传播、反向传播和计算图」中介绍了神经网络中梯度计算与存储的一般思路，并强调正向传播和反向传播相互依赖。正向传播在循环神经网络中比较直观，而通过时间反向传播其实是反向传播在循环神经网络中的具体应用。它要求我们将循环神经网络的计算图一次展开一个时间步，以获得模型变量和参数之间的依赖关系，并依据链式法则应用反向传播计算并存储梯度。由于序列可能相当长，因此依赖关系可能相当冗长。
    - 例如，对于 1000 个字符的序列，第一个标记可能会对最后位置的标记产生重大影响。这在计算上并不是真正可行的（它需要太长时间，需要太多的内存），并且它需要超过 1000 个矩阵乘积才能得到非常难以捉摸的梯度。这是一个充满计算与统计不确定性的过程。

通过时间反向传播

- 下图说明了使用循环神经网络的通过时间反向传播的三种策略，以解决链就会变得很长的问题：
  - 第一行是将文本划分为不同长度的段的随机截断。
  - 第二行是将文本分解为相同长度的子序列的常规截断。
    这就是我们在 RNN 实验中一直在做的。
  - 第三行是通过时间的完全反向传播，导致完整计算上不可行的表达式。

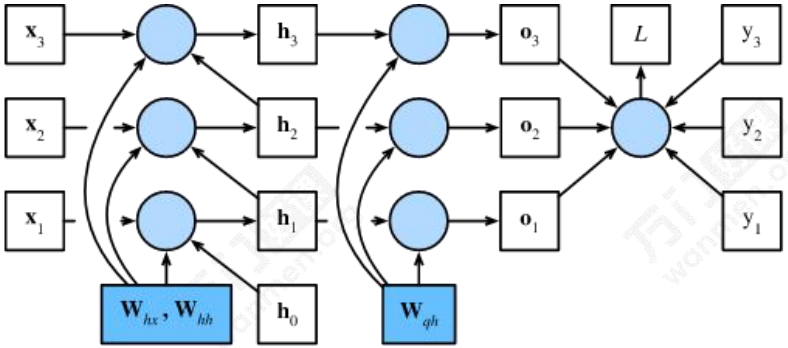


- 遗憾的是，虽然理论上具有吸引力，但随机截断并不比常规截断更好，很可能是由于多种因素。首先，经过一系列反向传播步后的观测结果足以捕获实际依赖关系。其次，增加的方差抵消了步长越多梯度越精确的事实。第三，我们实际上想要只有短范围交互的模型。因此，通过时间的规则截断的反向传播具有轻微的正则化效果。

通过时间反向传播

- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$

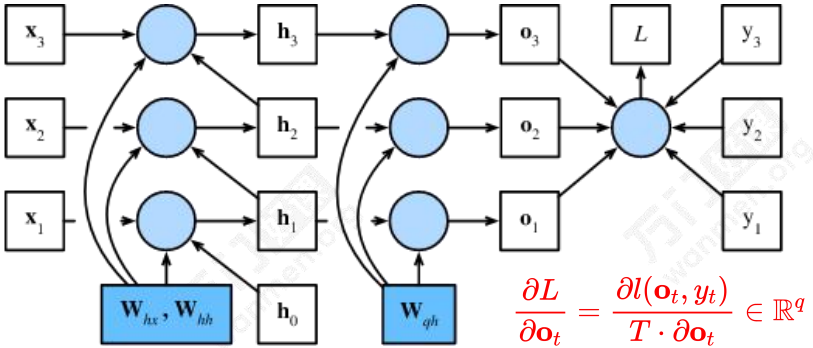


$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

通过时间反向传播

- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$



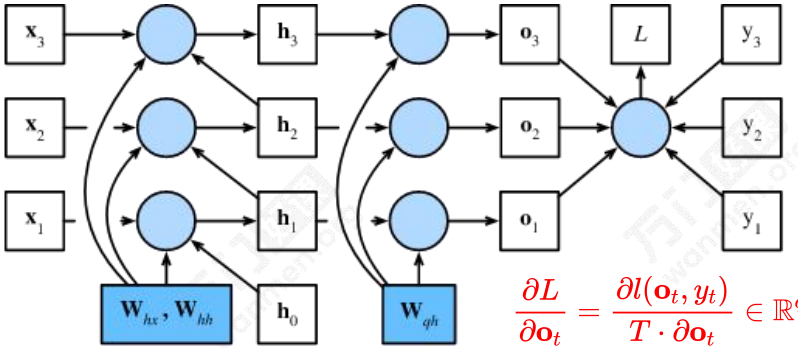
$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

通过时间反向传播

- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$



$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

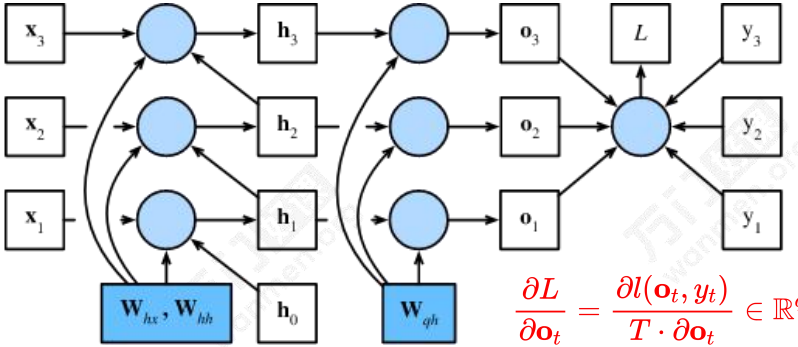
$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t}\mathbf{h}_t^\top$$

通过时间反向传播

- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}$$

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$



$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t}\mathbf{h}_t^\top$$

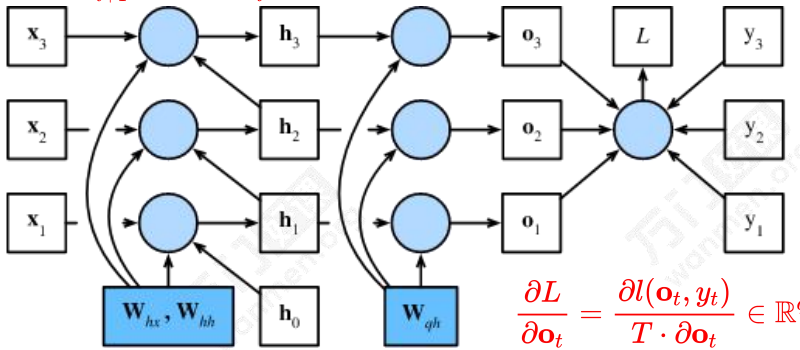
通过时间反向传播

- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_{t+1}}, \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right) + \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t}\right) = \mathbf{W}_{hh}^\top \frac{\partial L}{\partial \mathbf{h}_{t+1}} + \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_t}.$$

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$



$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t}\mathbf{h}_t^\top$$

通过时间反向传播

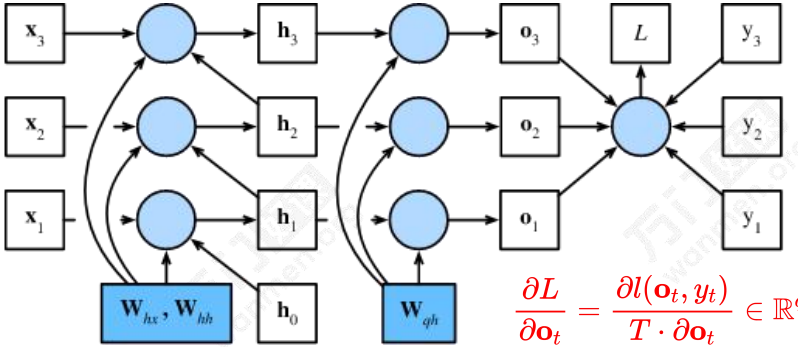
- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_{t+1}}, \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right) + \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t}\right) = \mathbf{W}_{hh}^\top \frac{\partial L}{\partial \mathbf{h}_{t+1}} + \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_t}.$$

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{i=t}^{T} \left(\mathbf{W}_{hh}^\top\right)^{T-i} \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_{T+t-i}}$$



$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t}\mathbf{h}_t^\top$$

通过时间反向传播

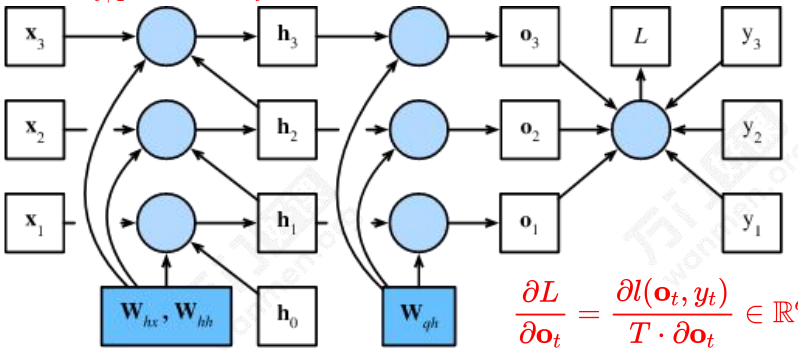
- 下面我们将展示如何计算目标函数相对于所有分解模型参数的梯度。为了保持简单，我们考虑一个没有偏置参数的循环神经网络，其在隐藏层中的激活函数使用恒等映射 $\phi(x)=x$ 。

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_{t+1}}, \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right) + \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t}\right) = \mathbf{W}_{hh}^\top \frac{\partial L}{\partial \mathbf{h}_{t+1}} + \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_t}.$$

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{i=t}^{T} \left(\mathbf{W}_{hh}^\top\right)^{T-i} \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_{T+t-i}}$$



$$\frac{\partial L}{\partial \mathbf{W}_{hx}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hx}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{x}_t^\top,$$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{h}_{t-1}^\top,$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t} \mathbf{h}_t^\top$$

通过时间反向传播

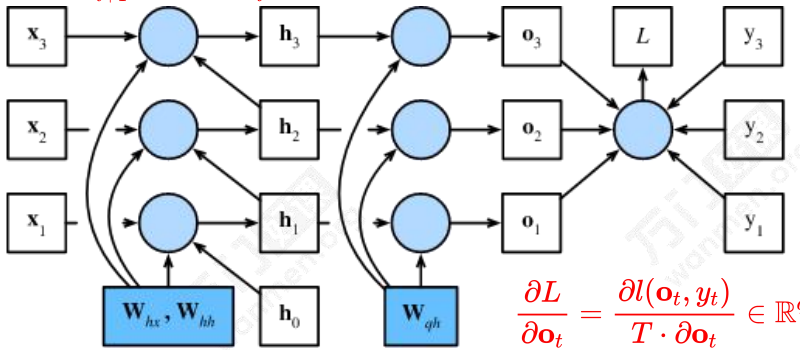
- 这个简单的线性例子已经展现了长序列模型的一些关键问题：它涉及到 $\mathbf{W}_{hh}^\top$ 的潜在非常大的指数。其中，小于1的特征值消失，大于1的特征值发散。这在数值上是不稳定的，表现为梯度消失或梯度爆炸。解决此问题的一种方法是按照计算方便的大小截断时间步长。

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_{t+1}}, \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right) + \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t}\right) = \mathbf{W}_{hh}^\top \frac{\partial L}{\partial \mathbf{h}_{t+1}} + \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_t}.$$

$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t, \qquad L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t)$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{i=t}^{T} \left(\mathbf{W}_{hh}^\top\right)^{T-i} \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_{T+t-i}}$$



$$\frac{\partial L}{\partial \mathbf{W}_{hx}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hx}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{x}_t^\top,$$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{h}_{t-1}^\top,$$

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1}$$

$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q$$

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t} \mathbf{h}_t^\top$$

门控循环单元（GRU）

- 虽然裁剪梯度可以应对梯度爆炸，但无法解决梯度衰减的问题。通常由于这个原因，循环神经网络在实际中较难捕捉时间序列中时间步距离较大的依赖关系。
- 我们还可能会遇到这样的情况：
    1. 一些标记没有相关的观测值。例如，在解析网页时，可能有一些辅助 HTML 代码与评估网页上传达的情绪无关。我们希望有一些机制来**跳过**隐状态表示中的此类标记。
    2. 序列的各个部分之间存在逻辑中断。例如，书的章节之间可能会有一个过渡，或者证券的熊市和牛市之间可能会有一个过渡。在这种情况下，最好有一种方法来**重置**我们的内部状态表示。
- 门控循环神经网络（gated recurrent neural network）的提出，正是为了解决上述的问题，它通过可以学习的门来控制信息的流动。其中，门控循环单元（gated recurrent unit，GRU）是一种常用的门控循环神经网络。另一种常用的门控循环神经网络则将在下一节中介绍。

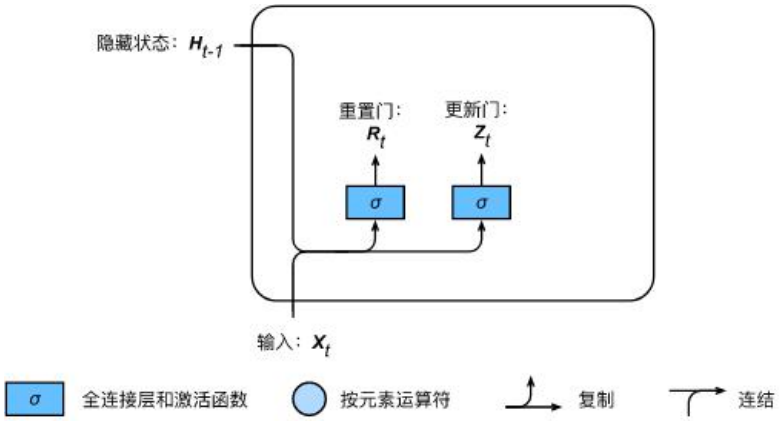
门控循环单元（GRU）

- 普通的循环神经网络和门控循环单元之间的关键区别在于：
  - 后者支持隐藏状态的门控（或者说选通）。这意味着有专门的机制来确定何时应该更新隐藏状态，以及何时应该重置隐藏状态。这些机制是可学习的，它们解决了上面列出的问题。
    - 例如，如果第一个标记非常重要，我们将学会在第一次观测之后不更新隐藏状态。同样，我们也可以学会跳过不相关的临时观测。最后，我们将学会在需要的时候重置隐藏状态。我们将在下面详细讨论这一点。

门控循环单元（GRU）

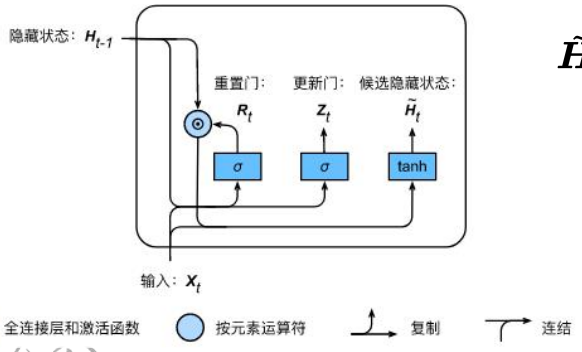
- 重置门和更新门
  - 门控循环单元中的重置门和更新门的输入均为当前时间步输入 $\boldsymbol{X}_t$ 与上一时间步隐藏状态 $\boldsymbol{H}_{t-1}$，输出由激活函数为 **sigmoid** 函数的全连接层计算得到。

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



门控循环单元（GRU）

- 候选隐藏状态
  - 门控循环单元将计算候选隐藏状态来辅助稍后的隐藏状态计算。
  - 我们将当前时间步重置门的输出与上一时间步隐藏状态做按元素乘法（符号为⊙）。如果重置门中元素值接近 0，那么意味着重置对应隐藏状态元素为 0，即丢弃上一时间步的隐藏状态；若接近 1，那么表示保留上一时间步的隐藏状态。
  - 从下面的公式可以看出，重置门控制了上一时间步的隐藏状态如何流入当前时间步的候选隐藏状态。而上一时间步的隐藏状态可能包含了时间序列截至上一时间步的全部历史信息。因此，重置门可以用来丢弃与预测无关的历史信息。



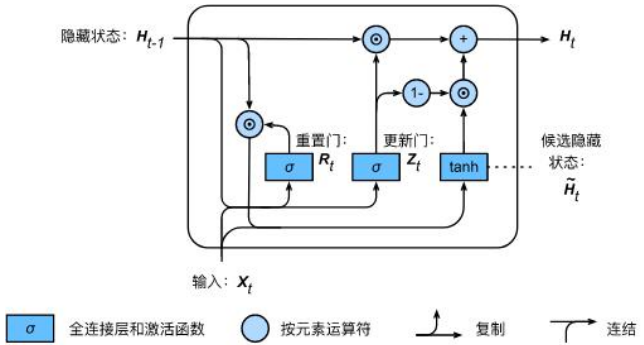
$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + (\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$

门控循环单元（GRU）

“Talk is cheap. Show me the code.”

- 隐藏状态
  - 值得注意的是，更新门可以控制隐藏状态应该如何被包含当前时间步信息的候选隐藏状态所更新，如下图所示。假设更新门在时间步 t' 到 $t(t' < t)$ 之间一直近似 1。那么，在时间步 t' 到 t 之间的输入信息几乎没有流入时间步 t 的隐藏状态 $\boldsymbol{H}_t$。实际上，这可以看作是较早时刻的隐藏状态 $\boldsymbol{H}_{t'-1}$ 一直通过时间保存并传递至当前时间步 t。这个设计可以应对循环神经网络中的梯度衰减问题，并更好地捕捉时间序列中时间步距离较大的依赖关系。

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



  - 重置门有助于捕捉时间序列里短期的依赖关系；
  - 更新门有助于捕捉时间序列里长期的依赖关系。

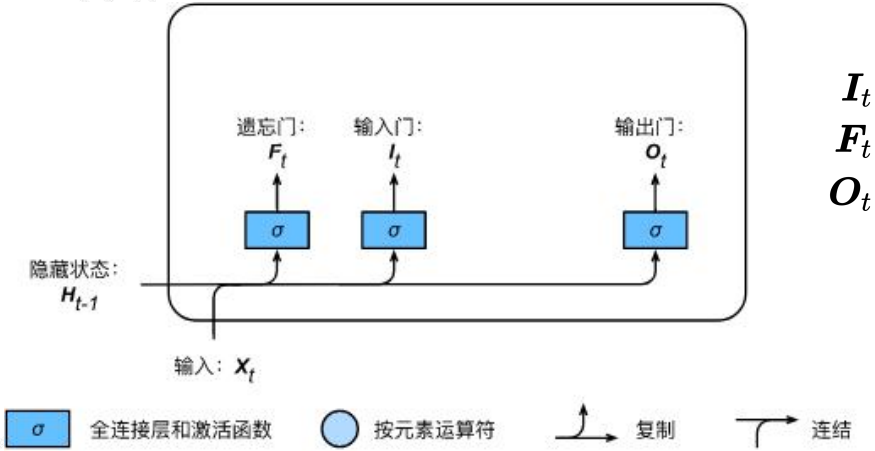
长短期记忆网络（LSTM）

- 长期以来，隐变量模型存在着长期信息保存和短期输入跳跃的问题。解决这一问题的最早方法之一是长短期记忆网络 (LSTM) 。它有许多与门控循环单元一样的属性。有趣的是，长短期记忆网络的设计比门控循环单元稍微复杂一些，但比门控循环单元早诞生了近20年。

- 可以说，长短期记忆网络的设计灵感来自于计算机的逻辑门。长短期记忆网络引入了存储单元（memory cell），或简称为单元（cell）。（一些文献认为存储单元是隐藏状态的一种特殊类型）。它们与隐藏状态具有相同的形状，被设计为记录附加信息。为了控制存储单元，我们需要许多门。如，我们需要一个门来从单元中读出条目。我们将其称为输出门（output gate）。另外，需要一个门来决定何时将数据读入单元。我们将其称为输入门（input gate）。最后，我们需要一种机制来重置单元的内容，由遗忘门（forget gate）来管理。这种设计的动机与门控循环单元相同，即能够通过专用机制决定什么时候记忆或忽略隐藏状态中的输入。让我们看看这在实践中是如何运作的。

长短期记忆网络（LSTM）

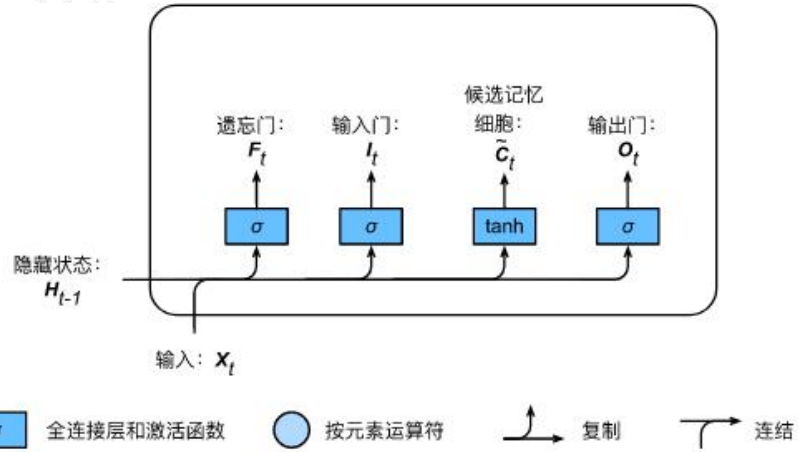
- 输入门、遗忘门和输出门
  - 与门控循环单元中的重置门和更新门一样，输出由激活函数为 sigmoid 函数的全连接层计算得到。



$$\boldsymbol{I}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i)$$
$$\boldsymbol{F}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$
$$\boldsymbol{O}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o)$$

长短期记忆网络（LSTM）

- 候选记忆细胞
  - 类似地，长短期记忆需要计算候选记忆细胞。

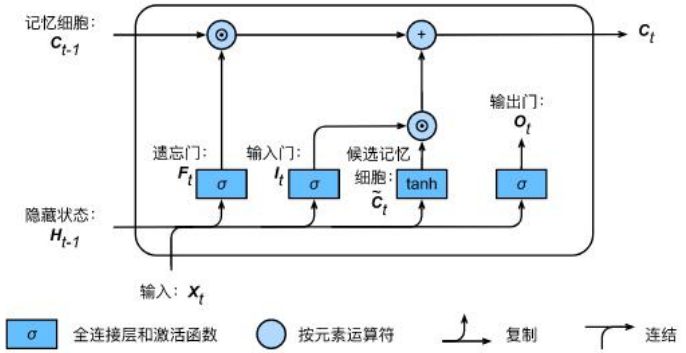


$$\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)$$

长短期记忆网络（LSTM）

- 记忆细胞
  - 我们可以通过元素值域在 [0,1] 的输入门、遗忘门和输出门来控制隐藏状态中信息的流动，这一般也是通过使用按元素乘法（符号为 ⊙）来实现的。当前时间步记忆细胞 $\boldsymbol{C}_t$ 的计算组合了上一时间步记忆细胞和当前时间步候选记忆细胞的信息，并通过遗忘门和输入门来控制信息的流动：

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



  - 如图，遗忘门控制上一时间步的记忆细胞 $\boldsymbol{C}_{t-1}$ 中的信息是否传递到当前时间步，而输入门则控制当前时间步的输入 $\boldsymbol{X}_t$ 通过候选记忆细胞 $\tilde{\boldsymbol{C}}_t$ 如何流入当前时间步的记忆细胞。如果遗忘门一直近似 1 且输入门一直近似 0，过去的记忆细胞将一直通过时间保存并传递至当前时间步。这个设计可以应对循环神经网络中的梯度衰减问题，并更好地捕捉时间序列中时间步距离较大的依赖关系。

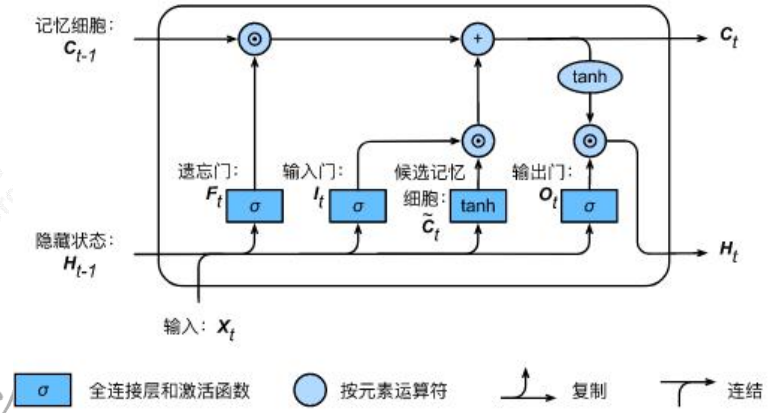
长短期记忆网络（LSTM）

“Talk is cheap. Show me the code.”

- 隐藏状态
  - 有了记忆细胞以后，接下来我们还可以通过输出门来控制从记忆细胞到隐藏状态 $\boldsymbol{H}_t$ 的信息的流动：

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

  - 这里的 tanh 函数确保隐藏状态元素值在 -1 到 1 之间。需要注意的是，当输出门近似 1 时，记忆细胞信息将传递到隐藏状态供输出层使用；当输出门近似 0 时，记忆细胞信息只自己保留。下图展示了长短期记忆中隐藏状态的计算。

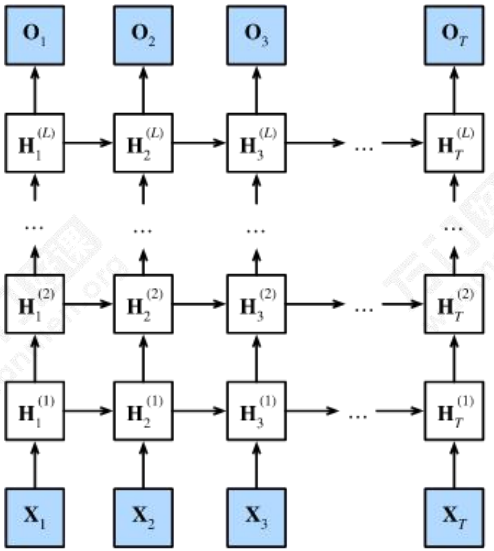


深度循环神经网络

- 到目前为止，我们只讨论了一个单向隐藏层的RNN。其中，隐变量和观测值如何相互作用的具体函数形式是相当任意的。只要我们有足够的灵活性来建模不同类型的交互，这就不是一个大问题。然而，对于一个单隐藏层来说，这可能是相当具有挑战性的。在线性模型的情况下，我们通过添加更多的层来解决这个问题。在循环神经网络中，这有点棘手。因为我们首先需要决定如何以及在哪里添加额外的非线性函数。

  - 在深度学习应用里，我们通常会用到含有多个隐藏层的循环神经网络，也称作深度循环神经网络。右图演示了一个有 L 个隐藏层的深度循环神经网络，每个隐藏状态不断传递至当前层的下一时间步和当前时间步的下一层。

  - 幸运的是，实现多层循环神经网络所需的许多细节在高级 API 中都是现成的。



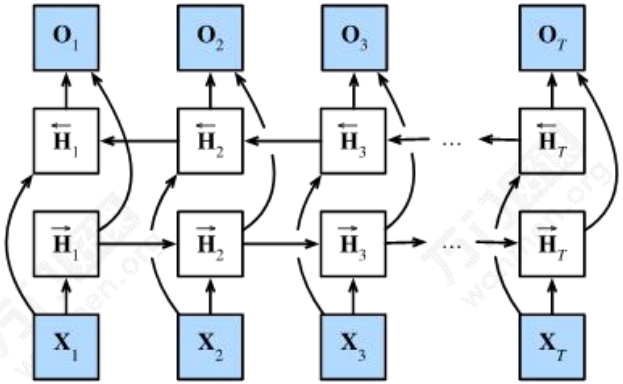
“Talk is cheap. Show me the code.”

双向循环神经网络

- 在序列学习中，我们以往假设的目标是：到目前为止，在给定所观测的情况下对下一个输出进行建模。例如，在时间序列的上下文中或在语言模型的上下文中。虽然这是一个典型的情况，但这并不是我们可能遇到的唯一情况。为了说明这个问题，考虑以下三个在文本序列中填空的任务：
  - 我 ___。
  - 我 ___ 饿了。
  - 我 ___ 饿了，我可以吃半头猪。
- 根据可获得的信息量，我们可以用不同的词填空，如“很高兴”（“happy”）、“不”（“not”）和“非常”（“very”）。很明显，短语的结尾（如果有的话）传达了重要信息。这些信息关乎到选择哪个词来填空。不能利用这一点的序列模型将在相关任务上表现不佳。例如，如果要做好命名实体识别（例如，识别“Green”指的是“格林先生”还是绿色），更长的范围上下文同样重要。

双向循环神经网络

- 双向循环神经网络通过增加从后往前传递信息的隐藏层来更灵活地处理这类信息。下图演示了一个含单隐藏层的双向循环神经网络的架构。
- 我们从最后一个标记开始从后向前运行循环神经网络，而不是只在前向模式下从第一个标记开始运行循环神经网络。双向循环神经网络添加了反向传递信息的隐藏层，以更灵活地处理此类信息。



$$\vec{\boldsymbol{H}}_t = \phi\left(\boldsymbol{X}_t \boldsymbol{W}_{xh}^{(f)} + \vec{\boldsymbol{H}}_{t-1} \boldsymbol{W}_{hh}^{(f)} + \boldsymbol{b}_h^{(f)}\right)$$

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

“ Talk is cheap. Show me the code. ”

小结

- 通过时间反向传播仅仅是反向传播对具有隐藏状态的序列模型的应用。
- 为了计算方便和数值稳定，需要截断，如规则截断和随机截断。
- 门控循环神经网络可以更好地捕获具有长时间步距离序列上的依赖关系。
- 重置门有助于捕获序列中的短期相互依赖关系。更新门有助于捕获序列中的长期相互依赖关系。
- 重置门打开时，门控循环单元包含基本循环神经网络；更新门打开时，门控循环单元可以跳过子序列。
- 长短期记忆网络有三种类型的门：输入门、遗忘门和控制信息流的输出门。
- 长短期记忆网络的隐藏层输出包括“隐藏状态”和“记忆单元”。只有隐藏状态会传递到输出层，记忆单元的信息完全储存在内部。
- 长短期记忆网络可以缓解梯度消失和梯度爆炸。
- 在深层循环神经网络中，隐藏状态信息被传递到当前层的下一时间步和下一层的当前时间步。
- 在双向循环神经网络中，每个时间步的隐藏状态由当前时间步前后信息同时决定。

谢谢观看

更多好课，请关注万门好课APP