循环神经网络

循环神经网络（Recurrent Neural Network， RNN）是一类具有短期记忆能力的神经网络。不但可以接受其它神经元的信息，也可以接受自身的信息，形成具有环路的网络结构。

随时间反向传播算法即按照时间的逆序将错误信息一步步地往前传递。当输入序列比较长时，会存在梯度爆炸和消失问题。

两种更广义的记忆网络模型：递归神经网络和图网络。

给网络增加记忆能力

为了处理这些时序数据并利用其历史信息。

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

在前馈网络中的非输出层都添加一个延时器，记录最近几次神经元的输出。在第 t 个时刻，第 l+1 层神经元和第 l 层神经元的最近 p 次输出相关，即

$$h_t^{(l+1)} = f(h_t^{(l)}, h_{t-1}^{(l)}, \cdots, h_{t-p+1}^{(l)})$$

延时神经网络在时间维度上共享权值，以降低参数数量。因此对于序列输入来讲，延时神经网络就相当于卷积神经网络。

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

通过一个延时器记录最近几次的外部输入和输出，第 t 个时刻的输出$\mathbf{y}_t$为

$$y_t = f(x_t, x_{t-1}, \cdots, x_{t-p}, y_{t-1}, y_{t-2}, \cdots, y_{t-q})$$

循环神经网络

使用带自反馈的神经元，能够处理任意长度的时序数据。

给定一个输入序列$\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$，循环神经网络通过下面公式更新带反馈边的隐藏层的活性值$\mathbf{h}_t$：

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t), \tag{6.4}$$

简单循环网络（Simple Recurrent Network， SRN）

假设在时刻t时，网络的输入为$\mathbf{x}_t$，隐藏层状态（即隐藏层神经元活性值）为$\mathbf{h}_t$不仅和当前时刻的输入$\mathbf{x}_t$相关，也和上一个时刻的隐藏层状态$\mathbf{h}_{t-1}$相关。

$$\mathbf{z}_t = U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b}, \tag{6.5}$$

$$\mathbf{h}_t = f(\mathbf{z}_t), \tag{6.6}$$

循环神经网络的计算能力

前馈神经网络可以模拟任何连续函数，而循环神经网络可以模拟任何程序。

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，并且对状态空间的紧致性没有限制。

定理 6.1 – 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

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

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

图灵完备（Turing Completeness）

一种数据操作规则，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。

定理 6.2 – 图灵完备 [Siegelmann and Sontag, 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。

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。

序列到类别模式

用于序列数据的分类问题：输入为序列，输出为类别。

假设一个样本$\mathbf{x}_{1:T} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$为一个长度为$T$的序列，输出为一个类别$y \in \{1, \cdots, C\}$。我们可以将样本$\mathbf{x}$按不同时刻输入到循环神经网络中，并得到不同时刻的隐藏状态$\mathbf{h}_1, \cdots, \mathbf{h}_T$。我们可以将$\mathbf{h}_T$看作整个序列的最终表示（或特征），并输入给分类器$g(\cdot)$进行分类（如图6.3a所示）。

$$\hat{y} = g(\mathbf{h}_T), \tag{6.22}$$

除了将最后时刻的状态作为序列表示之外，我们还可以对整个序列的所有状态进行平均，并用这个平均状态来作为整个序列的表示（如图6.3b所示）。

$$\hat{y} = g(\frac{1}{T}\sum_{t=1}^{T}\mathbf{h}_t). \tag{6.23}$$

同步的序列到序列模式

主要用于序列标注（Sequence Labeling），即每一时刻都有输入和输出，输入序列和输出序列的长度相同。

异步的序列到序列模式（编码器-解码器，Encoder-Decoder）

输入序列和输出序列不需要有严格的对应关系，也不需要保持相同的长度。

在异步的序列到序列模式中（如图6.5所示），输入为一个长度为T的序列$\mathbf{x}_{1:T}=(\mathbf{x}_1,\cdots,\mathbf{x}_T)$，输出为长度为$M$的序列$y_{1:M}=(y_1,\cdots,y_M)$。经常通过先编码后解码的方式来实现。先将样本$\mathbf{x}$按不同时刻输入到一个循环神经网络（编码器）中，并得到其编码$\mathbf{h}_T$。然后在使用另一个循环神经网络（解码器）中，得到输出序列$\hat{y}_{1:M}$。为了建立输出序列之间的依赖关系，在解码器中通常使用非线性的自回归模型。

$$\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \mathbf{x}_t), \qquad \forall t \in [1, T] \tag{6.25}$$

$$\mathbf{h}_{T+t} = f_2(\mathbf{h}_{T+t-1}, \hat{\mathbf{y}}_{t-1}), \qquad \forall t \in [1, M] \tag{6.26}$$

$$\hat{y}_t = g(\mathbf{h}_{T+t}), \qquad \forall t \in [1, M]. \tag{6.27}$$

参数学习

随时间反向传播（BPTT）和实时循环学习（RTRL）算法。

随时间反向传播算法

通过类似前馈神经网络的错误反向传播

将循环神经网络看作是一个展开的多层前馈网络，其中“每一层”对应循环网络中的“每个时刻”。循环神经网络就按照前馈网络中的反向传播算法进行计算参数梯度。在“展开”的前馈网络中，所有层的参数是共享的，因此参数的真实梯度是将所有“展开层”的参数梯度之和。

计算偏导数 $\frac{\partial \mathcal{L}_t}{\partial U}$ 先来计算公式(6.30)中第t时刻损失对参数U的偏导数$\frac{\partial \mathcal{L}_t}{\partial U}$。

因为参数U和隐藏层在每个时刻$k(1 \leq k \leq t)$的净输入$\mathbf{z}_k = U\mathbf{h}_{k-1} + W\mathbf{x}_k + \mathbf{b}$有关，因此第$t$时刻损失的损失函数$\mathcal{L}_t$关于参数$U_{ij}$的梯度为：

$$\frac{\partial \mathcal{L}_t}{\partial U_{ij}} = \sum_{k=1}^{t} tr\left((\frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k})^{\mathrm{T}} \frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} \right) \tag{6.31}$$

$$= \sum_{k=1}^{t} \left(\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} \right)^{\mathrm{T}} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k}, \tag{6.32}$$

其中$\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}}$表示“直接”偏导数，即公式$\mathbf{z}_k = U\mathbf{h}_{k-1} + W\mathbf{x}_k + \mathbf{b}$中保持$\mathbf{h}_{k-1}$不变，对$U_{ij}$进行求偏导数，得到

$$\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} = \begin{bmatrix} 0 \\ \vdots \\ [\mathbf{h}_{k-1}]_j \\ \vdots \\ 0 \end{bmatrix} \triangleq \mathbb{I}_i([\mathbf{h}_{k-1}]_j), \tag{6.33}$$

其中$[\mathbf{h}_{k-1}]_j$为第$k-1$时刻隐状态的第j维；$\mathbb{I}_i(x)$除了第i行值为x外，其余都

定义$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k}$为第$t$时刻的损失对第$k$时刻隐藏神经层的净输入$\mathbf{z}_k$的导数，则

$$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \tag{6.34}$$

$$= \frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_{k+1}} \tag{6.35}$$

$$= \mathrm{diag}(f'(\mathbf{z}_k)) U^{\mathrm{T}} \delta_{t,k+1}. \tag{6.36}$$

将公式 (6.36) 和 (6.33) 代入公式 (6.32) 得到

$$\frac{\partial \mathcal{L}_t}{\partial U_{ij}} = \sum_{k=1}^{t} [\delta_{t,k}]_i [\mathbf{h}_{k-1}]_j. \tag{6.37}$$

将上式写成矩阵形式为

$$\frac{\partial \mathcal{L}_t}{\partial U} = \sum_{k=1}^{t} \delta_{t,k} \mathbf{h}_{k-1}^{\mathrm{T}}. \tag{6.38}$$

参数梯度 将公式 (6.38) 代入到将公式 (6.30) 得到整个序列的损失函数 $\mathcal{L}$ 关于参数 U 的梯度

$$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} \mathbf{h}_{k-1}^{\mathrm{T}}. \tag{6.39}$$

同理可得，$\mathcal{L}$ 关于权重 W 和偏置 $\mathbf{b}$ 的梯度为

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k} \mathbf{x}_k^{\mathrm{T}}, \tag{6.40}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^{T} \sum_{k=1}^{t} \delta_{t,k}. \tag{6.41}$$

计算复杂度 在 BPTT 算法中，参数的梯度需要在一个完整的“前向”计算和“反向”计算后才能得到并进行参数更新。

实时循环学习算法

通过前向传播的方式来计算梯度

假设循环网络网络中第 $t+1$ 时刻的状态 $\mathbf{h}_{t+1}$ 为

$$\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(U\mathbf{h}_t + W\mathbf{x}_{t+1} + \mathbf{b}), \tag{6.42}$$

其关于参数 U_{ij} 的偏导数为

$$\frac{\partial \mathbf{h}_{t+1}}{\partial U_{ij}} = \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{z}_{t+1}} \left(\frac{\partial^{+} \mathbf{z}_{t+1}}{\partial U_{ij}} + U \frac{\partial \mathbf{h}_t}{\partial U_{ij}} \right) \tag{6.43}$$

$$= \mathrm{diag}(f'(\mathbf{z}_{t+1})) \left(\mathbb{I}_i([\mathbf{h}_t]_j) + U \frac{\partial \mathbf{h}_t}{\partial U_{ij}} \right) \tag{6.44}$$

$$= f'(\mathbf{z}_{t+1}) \odot \left(\mathbb{I}_i([\mathbf{h}_t]_j) + U \frac{\partial \mathbf{h}_t}{\partial U_{ij}} \right), \tag{6.45}$$

其中 $\mathbb{I}_i(x)$ 除了第 i 行值为 x 外，其余都为 0 的向量。

RTRL 算法从第 1 个时刻开始，除了计算循环神经网络的隐状态之外，还利用公式 (6.45) 依次前向计算偏导数 $\frac{\partial \mathbf{h}_1}{\partial U_{ij}}, \frac{\partial \mathbf{h}_2}{\partial U_{ij}}, \frac{\partial \mathbf{h}_3}{\partial U_{ij}}, \cdots$。

这样，假设第t个时刻存在一个监督信息，其损失函数为$\mathcal{L}_t$，就可以同时计算损失函数对U_{ij}的偏导数

$$\frac{\partial \mathcal{L}_t}{\partial U_{ij}} = \left(\frac{\partial \mathbf{h}_t}{\partial U_{ij}}\right)^{\mathrm{T}} \frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_t}. \tag{6.46}$$

这样在第t时刻，可以实时地计算损失$\mathcal{L}_t$关于参数U的梯度，并更新参数。参数W和$\mathbf{b}$的梯度也可以同样按上述方法实时计算。

两种算法比较 RTRL 算法和 BPTT 算法都是基于梯度下降的算法，分别通过前向模式和反向模式应用链式法则来计算梯度。在循环神经网络中，一般网络输出维度远低于输入维度，因此 BPTT 算法的计算量会更小，但是 BPTT 算法需要保存所有时刻的中间梯度，空间复杂度较高。RTRL 算法不需要梯度回传，因此非常适合用于需要在线学习或无限序列的任务中。

长期依赖问题

在BPTT算法中，将公式(6.36)展开得到

$$\delta_{t,k} = \prod_{i=k}^{t-1} \left(\mathrm{diag}(f'(\mathbf{z}_i))U^{\mathrm{T}}\right)\delta_{t,t}. \tag{6.47}$$

如果定义$\gamma \cong \|\mathrm{diag}(f'(\mathbf{z}_i))U^{\mathrm{T}}\|$，则

$$\delta_{t,k} = \gamma^{t-k}\delta_{t,t}. \tag{6.48}$$

若$\gamma > 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to \infty$，会造成系统不稳定，称为梯度爆炸问题（Gradient Exploding Problem）；相反，若$\gamma < 1$，当$t-k \to \infty$时，$\gamma^{t-k} \to 0$，会出现和深度前馈神经网络类似的梯度消失问题（gradient vanishing problem）

由于循环神经网络经常使用非线性激活函数为logistic函数或tanh函数作为非线性激活函数，其导数值都小于1；并且权重矩阵$\|U\|$也不会太大，因此如果时间间隔$t-k$过大，$\delta_{t,k}$会趋向于0，因此经常会出现梯度消失问题。

虽然简单循环网络理论上可以建立长时间间隔的状态之间的依赖关系，但是由于梯度爆炸或消失问题，实际上只能学习到短期的依赖关系。这样，如果t时刻的输出y_t依赖于$t-k$时刻的输入$\mathbf{x}_{t-k}$，当间隔k比较大时，简单神经网络很难建模这种长距离的依赖关系，称为长期依赖问题（Long-Term Dependencies Problem）。

改进模型或优化方法来缓解循环网络的梯度爆炸和梯度消失问题：

梯度爆炸 一般而言，循环网络的梯度爆炸问题比较容易解决，一般通过权重衰减或梯度截断来避免。

权重衰减是通过给参数增加ℓ_1或ℓ_2范数的正则化项来限制参数的取值范

围，从而使得$\gamma \leq 1$。梯度截断是另一种有效的启发式方法，当梯度的模大于一定阈值时，就将它截断成为一个较小的数。

梯度消失 梯度消失是循环网络的主要问题。除了使用一些优化技巧外，更有效的方式就是改变模型，比如让$U = I$，同时使用$f'(\mathbf{z}_i) = 1$，即

$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t; \theta), \tag{6.49}$$

此时便转换为线性依赖关系，且权重系数为1，这样就不存在梯度爆炸或消失问题。但是，这种改变也丢失了神经元在反馈边上的非线性激活的性质，因此也降低了模型的表示能力。

$$\mathbf{h}_t = \mathbf{h}_{t-1} + g(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta), \tag{6.50}$$

这样$\mathbf{h}_t$和$\mathbf{h}_{t-1}$之间为既有线性关系，也有非线性关系，在一定程度上可以缓解梯度消失问题。但这种改进依然有一个问题就是记忆容量（memory capacity）。随着$\mathbf{h}_t$不断累积存储新的输入信息，会发生饱和现象。假设$g(\cdot)$为logistic函数，则随着时间t的增长，$\mathbf{h}_t$会变得越来越大，从而导致$\mathbf{h}$变得饱和。也就是说，隐状态$\mathbf{h}_t$可以存储的信息是有限的，随着记忆单元存储的内容越来越多，其丢失的信息也越来越多。

一种是增加一些额外的存储单元：外部记忆；另一种是进行选择性的遗忘，同时也进行有选择的更新。

基于门控的循环神经网络

基于门控的循环神经网络（Gated RNN）：长短期记忆（LSTM）网络和门控循环单元（GRU）网络。

长短期记忆网络

新的内部状态 LSTM网络引入一个新的内部状态（internal state）$\mathbf{c}_t$专门进行线性的循环信息传递，同时（非线性）输出信息给隐藏层的外部状态h_t。

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{6.51}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{6.52}$$

其中f_t，i_t和o_t为三个门（gate）来控制信息传递的路径；$\odot$为向量元素乘积；c_{t-1}为上一时刻的记忆单元；$\tilde{c}_t$是通过非线性函数得到候选状态，

$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c). \tag{6.53}$$

在每个时刻t，LSTM网络的内部状态c_t记录了到当前时刻为止的历史信息。

门机制 LSTM网络引入门机制（gating mechanism）来控制信息传递的路径。分别为输入门i_t，遗忘门f_t和输出门o_t：

- 遗忘门f_t控制上一个时刻的内部状态c_{t-1}需要遗忘多少信息
- 输入门i_t控制当前时刻的候选状态$\tilde{c}_t$有多少信息需要保存
- 输出门o_t控制当前时刻的内部状态c_t有多少信息需要输出给外部状态h_t

三个门的计算方式为：

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{6.54}$$

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{6.55}$$

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{6.56}$$

LSTM 网络的循环单元结构计算过程：（1）首先利用上一时刻的外部状态h_{t-1}当前时刻的输入x_t，计算出三个门，以及候选状态$\tilde{c}_t$；（2）结合遗忘门f_t和输入门i_t来更新记忆单元c_t；（3）结合输出门o_t，将内部状态的信息传递给外部状态h_t。

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right), \tag{6.57}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{6.58}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{6.59}$$

其中$x_t \in R^e$为当前时刻的输入，$\mathrm{W} \in R^{4d\times(d+e)}$和$\mathrm{b} \in R^{4d}$为网络参数。

记忆　循环神经网络中的隐状态 h 存储了历史信息，可以看作是一种记忆（memory）。在简单循环网络中，隐状态每个时刻都会被重写，因此可以看作是一种短期记忆（short-term memory）。在神经网络中，长期记忆（long-term memory）可以看作是网络参数，隐含了从训练数据中学到的经验，并更新周期要远远慢于短期记忆。而在 LSTM 网络中，记忆单元 c 可以在某个时刻捕捉到某个关键信息，并有能力将此关键信息保存一定的时间间隔。记忆单元 c 中保存信息的生命周期要长于短期记忆 h，但又远远短于长期记忆，因此称为长的短期记忆（long short-term memory）。

LSTM 网络的各种变体

无遗忘门的 LSTM 网络：　$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t.$ (6.60)

记忆单元 c 会不断增大。当输入序列的长度非常大时，记忆单元的容量会饱和，从而大大降低 LSTM 模型的性能。

peephole 连接 另外一种变体是三个门不但依赖于输入 $\mathbf{x}_t$ 和上一时刻的隐状态 $\mathbf{h}_{t-1}$，也依赖于上一个时刻的记忆单元 $\mathbf{c}_{t-1}$。

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + V_i\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{6.61}$$

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + V_f\mathbf{c}_{t-1} + \mathbf{b}_f), \tag{6.62}$$

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + V_o\mathbf{c}_t + \mathbf{b}_o), \tag{6.63}$$

耦合输入门和遗忘门 LSTM 网络中的输入门和遗忘门有些互补关系，因此同时用两个门比较冗余。为了减少 LSTM 网络的计算复杂度，将这两门合并为一个门。

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

门控循环单元网络

门控循环单元（Gated Recurrent Unit，GRU）将输入门与和遗忘门合并成一个门：更新门。同时，GRU 也不引入额外的记忆单元，直接在当前状态h_t和历史状态h_{t-1}之间引入线性依赖关系。

深层循环神经网络

增加循环神经网络的深度从而增强循环神经网络的能力， 主要是增加同一时刻网络输入到输出之间的路径。

堆叠循环神经网络（Stacked Recurrent Neural Network， SRNN）

第 1 层网络的输入是第 l-1 层网络的输出。定义$h_t{}^l$为在时刻 t 时第 l 层的隐状态

$$\mathbf{h}_t^{(l)} = f(U^{(l)}\mathbf{h}_{t-1}^{(l)} + W^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{b}^{(l)}), \tag{6.70}$$

双向循环神经网络（bidirectional recurrent neural network， Bi-RNN）

由两层循环神经网络组成，它们的输入相同，只是信息传递的方向不同。

假设第 1 层按时间顺序，第 2 层按时间逆序，在时刻 t 时的隐状态定义为$h_t{}^1$和$h_t{}^2$，则

$$\mathbf{h}_t^{(1)} = f(U^{(1)}\mathbf{h}_{t-1}^{(1)} + W^{(1)}\mathbf{x}_t + \mathbf{b}^{(1)}), \tag{6.71}$$

$$\mathbf{h}_t^{(2)} = f(U^{(2)}\mathbf{h}_{t+1}^{(2)} + W^{(2)}\mathbf{x}_t + \mathbf{b}^{(2)}), \tag{6.72}$$

$$\mathbf{h}_t = \mathbf{h}_t^{(1)} \oplus \mathbf{h}_t^{(2)}, \tag{6.73}$$

扩展到图结构

如果将循环神经网络按时间展开，每个时刻的隐状态h_t看做一个节点，那么这些节点构成一个链式结构，每个节点 t 都收到其父节点的消息（message），更新自己的状态，并传递给其子节点。

递归神经网络（Recursive Neural Network， RecNN）

主要用来建模自然语言句子的语义。给定一个句子的语法结构（一般为树状结构），可以使用递归神经网络来按照句法的组合关系来合成一个句子的语义。句子中每个短语成分可以在分成一些子成分，即每个短语的语义都可以由它的子成分语义组合而来，并进而合成整句的语义。

树结构的长短期记忆模型（Tree-Structured LSTM）：用门机制来改进递归神经网络中的长距离依赖问题。

图网络（Graph Network， GN）

图中每个节点 v 都用一组神经元来表示其状态 h(v)，初始状态可以为节点 v 的输入特征 x(v)。每个节点可以收到来自相邻节点的消息，并更新自己的状态。

$$\mathbf{m}_t^{(v)} = \sum_{u \in N(v)} f\left(\mathbf{h}_{t-1}^{(v)}, \mathbf{h}_{t-1}^{(u)}, \mathbf{e}^{(u,v)}\right), \tag{6.79}$$

$$\mathbf{h}_t^{(v)} = g\left(\mathbf{h}_{t-1}^{(v)}, \mathbf{m}_t^{(v)}\right), \tag{6.80}$$

其中 N(v)表示节点 v 的邻居，$m_t^{(v)}$表示在第 t 时刻节点 v 收到的信息，$\mathbf{e}^{(u,v)}$为边$\mathbf{e}^{(u,v)}$上的特征。

一种同步的更新方式，所有的结构同时接受信息并更新自己的状态。而对于有向图来说，使用异步的更新方式会更有效率。

在整个图更新 T 次后，可以通过一个读出函数（readout function）g(•)来得到整个网络的表示。

$$\mathbf{y}_t = g\left(\{\mathbf{h}_T^{(v)} | v \in \mathcal{V}\}\right). \tag{6.81}$$