第5章　循环神经网络

循环神经网络（Recurrent Neural Network, RNN）是专为处理序列数据（如文本或时间序列）而设计的神经网络。循环神经网络在处理时间序列的每个输入时都会考虑之前时间步的信息，从而能够综合整个序列的上下文进行输出。此外，循环神经网络还可以适应不同长度的输入序列。本章将介绍经典循环神经网络及其变体，如长短时记忆网络、门控循环单元等。

为了实现对序列数据的处理，循环神经网络使用了参数共享的策略，即模型在不同时间节点共享相同的参数，这一特性使模型能够更好地适应各种不同形式的输入数据、不同长度的序列，并更好地进行泛化。而上一章节中介绍的卷积神经网络在空间维度共享卷积核，同样使用了参数共享的思想。假如为循环神经网络每个时间点或位置都引入独立的参数，不仅会增加模型的复杂性，还会限制模型的泛化和适应能力。举例来说，比较以下两句话：“我在大学三年级学习过深度学习课程”与“大学三年级期间，我修过深度学习课程”，虽然表达方式不同，但包含了重复的信息片段，这正是共享机制可以提升模型表现的情形。一个理想的模型应该能够阅读这两个句子并提取出所描述的时间。不管“大学三年级”在句子中出现在第3至第7个字，还是第1～第5个字的位置，我们都期望模型能够将其识别为表示时间的关键信息。如果我们使用传统的全连接前馈神经网络来处理定长文本，那么该模型通常会为每个输入位置单独设定权重参数。这样的设计将使模型无法有效共享位置信息，进而限制了其在处理语序变化或变长输入时的能力。这意味着网络需要学习句子中的所有语言规则，并且对于不同位置的语法和语义差异需要分别建模。而循环神经网络通过在多个时间步共享相同的权重参数，可以有效地捕捉序列中的信息，不需要为每个位置都单独学习语言规则，从而提高了模型的学习效率和泛化能力。

本章节中我们使用以下符号来表示序列数据：本章中我们使用以下符号来表示序列数据 $\boldsymbol{x}_{1:\tau} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)})$，其中每个时刻 t（$t \in [1, \tau]$）都包含一个向量 $\boldsymbol{x}^{(t)}$。

5.1　经典循环神经网络

5.1.1　循环神经网络的两种设计模式

如同前馈神经网络可以用多种不同的方式定义一样，循环神经网络也有多种建立方式。给定输入序列 $\boldsymbol{x}_{1:\tau} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)})$，循环神经网络通过以下公式更新隐藏单元值，则

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{5.1}$$

式中，$\boldsymbol{\theta}$ 是网络参数，$\boldsymbol{h}^{(t)}$ 是 t 时刻的隐藏单元值。在获得当前时刻的隐藏单元值 $\boldsymbol{h}^{(t)}$ 后，循环神经网络输出层利用状态信息 $\boldsymbol{h}$ 来生成结果。循环神经网络的结构如图5-1所示。

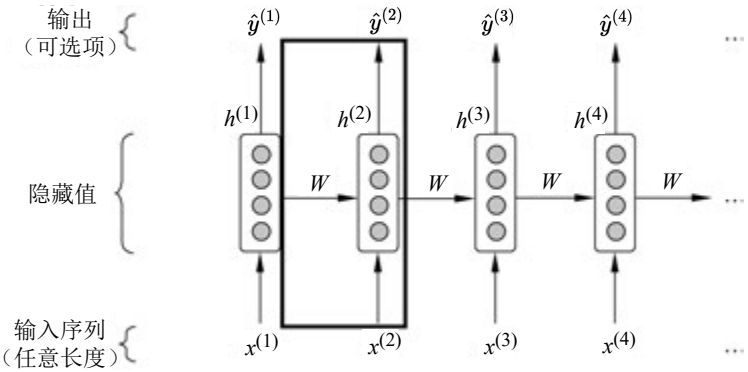

图 5-1　典型的循环神经网络架构

循环神经网络在预测未来输出时，会依赖先前时间步的信息。因此，它必须学会将当前的隐藏状态 $\boldsymbol{h}^{(t)}$ 表示为一个关于历史输入序列（从时刻 1 到 t）中与当前任务相关内容的压缩表达。由于这一表达需要将可能非常长的输入序列 $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)})$ 压缩成一个维度固定的向量 $\boldsymbol{h}^{(t)}$，因此不可避免地会丢失部分细节信息，即它是一个有损的表示。不同的训练目标会选择性地保留过去序列的某些方面。以影评情感分类任务为例，循环神经网络需要根据输入的词序列推断整段文本的情绪倾向。在这种场景下，模型并不需要保留从第一个词到当前词之间的全部内容，而是要学习提取与最终分类决策相关的关键信息。这意味着，隐藏状态 $\boldsymbol{h}^{(t)}$ 并非是对过去所有输入的完整记录，而是一种经过筛选的表示，仅保留对判定情感极性有用的部分特征。在某些任务中，例如语音识别，模型往往需要更全面地保留输入中的信息，以便能够较为完整地重构原始内容。因此，RNN 对隐藏状态的表示能力有不同的要求，具体取决于所处理的任务——有的任务只需保留某些方面的关键信息，有的则要求尽可能多地捕捉输入细节。

循环神经网络主要有两种设计模式，即多对多和多对一。我们将分别讨论这两种设计模式以及它们的应用场景。

1. 多对多循环神经网络

多对多循环神经网络的典型应用场景包括自然语言处理中的序列标注任务，如命名实体识别和词性标注。在这些任务中，输入是一个单词序列，而输出也是一个单词序列，每个输入单词都对应一个输出标签。例如，在命名实体识别中，输入是一句话，而输出是句子中每个单词的命名实体类别（如人名、地名等）。多对多循环神经网络之所以能够处理这种情况，是因为它可以在每个时间步产生一个输出，这些输出与输入序列的每个元素相关联。

图 5-2 展示了一个典型的多对多循环神经网络用于序列标注任务的示例。在该网络中，损失函数 L 衡量了模型生成的标签序列与真实标签序列之间的差异，从而引导模型学习更精确地标注输入数据。具体地说，L 评估模型每一步输出 $\boldsymbol{o}$ 与其对应目标标签 y 之间的误差。当模型使用 softmax 层进行分类时，输出 $\boldsymbol{o}$ 被视为一组未归一化的对数概率。此时，模型会先通过 softmax 函数计算出预测概率 $\hat{y} = \text{softmax}(\boldsymbol{o})$，再将其与真实标签 y 进行比较以计算损失。损失函数通常采用多分类任务常用的交叉熵函数。模型包含三组参数矩阵：从输入 $\boldsymbol{x}^{(t)}$ 到隐藏状态 $\boldsymbol{h}^{(t)}$ 的映射参数矩阵 $\boldsymbol{U}$，隐含层内部状态 $\boldsymbol{h}^{(t-1)}$ 到 $\boldsymbol{h}^{(t)}$ 的递归连接参数矩阵 W，隐藏状态 $\boldsymbol{h}^{(t)}$ 到输出 $\boldsymbol{o}^{(t)}$ 的映射参数矩阵 $\boldsymbol{V}$。

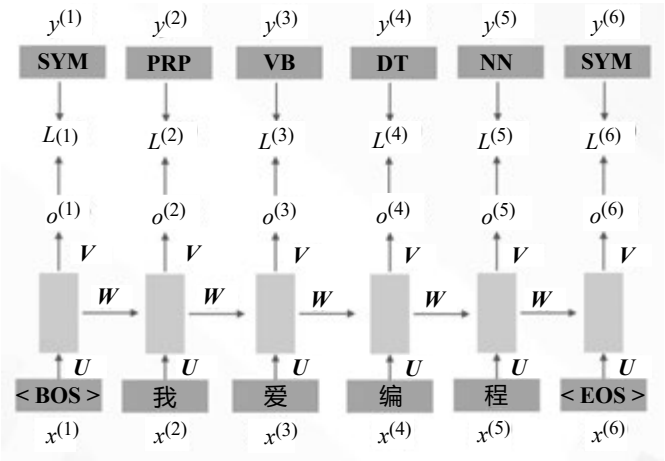

图 5-2　多对多循环神经网络

2. 多对一循环神经网络

如图 5-3 所示，这类模型的特点是在处理完整个输入序列之后，仅在最后一个时间步产生一次输出。这种设计适用于从整个序列中提取关键信息，并将其压缩为一个定长向量，以便用于后续的判别任务。这种结构广泛用于情绪识别、文本分类等任务。例如，在电影评论情感分析中，输入是一个词语序列，输出则是对整段文本情绪取向的判断（如积极、消极或中立）。模型通过将整个输入映射为一个语义表达向量，再通过输出层给出最终预测

$\boldsymbol{o}^{(\tau)}$。损失函数 L 用于评估模型输出 $\boldsymbol{o}^{(\tau)}$ 与真实标签 $\boldsymbol{y}^{(\tau)}$ 之间的误差，并反向传播该误差以优化模型参数，使其更准确地执行情感判断任务。

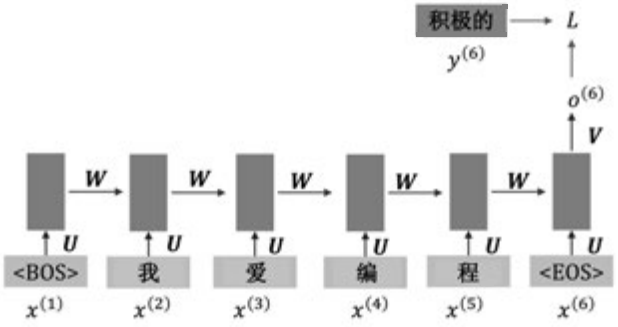

图 5-3　多对一循环神经网络

5.1.2　前向传播和反向传播

接下来我们以多对多循环神经网络为例，介绍循环神经网络的前向传播机制。此处我们假设模型使用的是双曲正切激活函数（tanh），并聚焦于离散型输出的生成任务，例如单词或字符序列的预测。在这类场景中，模型的输出 $\boldsymbol{o}$ 通常对应于一个未归一化的得分向量，每个分量代表某一候选词（或字符）的评分。为了将这些得分转换为概率分布，我们可以对 $\boldsymbol{o}$ 应用 softmax 函数，得到归一化后的输出概率向量 $\hat{\boldsymbol{y}}$。RNN 的前向传播从一个预定义的初始隐藏状态 $\boldsymbol{h}^{(0)}$ 开始，并在 $t=1$ 到 $t=\tau$ 每个时间步依次执行隐藏状态更新、输出计算以及预测生成。具体步骤通过如下递归公式进行定义：

$$\boldsymbol{h}^{(t)} = \tanh(W\boldsymbol{h}^{(t-1)} + U\boldsymbol{x}^{(t)} + \boldsymbol{b}_i) \tag{5.2}$$

$$\boldsymbol{o}^{(t)} = V\boldsymbol{h}^{(t)} + \boldsymbol{b}_o \tag{5.3}$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}) \tag{5.4}$$

其中，$\boldsymbol{b_i}$ 和 $\boldsymbol{b_o}$ 是可学习的偏置向量，U、V 和 W 是上面介绍过的参数矩阵。

对于反向传播过程，我们需要使用“通过时间反向传播”（Back-Propagation Through Time，BPTT）的技术。以多对多循环神经网络为例，给定输入序列 $(\boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \cdots, \boldsymbol{x^{(\tau)}})$ 和输出序列 $(\boldsymbol{y^{(1)}}, \boldsymbol{y^{(2)}}, \cdots, \boldsymbol{y^{(\tau)}})$，损失函数定义为

$$L\left((\boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \cdots, \boldsymbol{x^{(\tau)}}), (\boldsymbol{y^{(1)}}, \boldsymbol{y^{(2)}}, \cdots, \boldsymbol{y^{(\tau)}})\right) = \sum_{t=1}^{\tau} L^{(t)} \tag{5.5}$$

$$= -\sum_{t=1}^{\tau} \log p_{\text{model}}\left(\boldsymbol{y^{(t)}} \mid \boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \cdots, \boldsymbol{x^{(t)}}\right) \tag{5.6}$$

其中，$p_{\text{model}}\left(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(t)}\right)$ 表示模型对于 $\boldsymbol{y}^{(t)}$ 的预测概率，这个概率由模型输出的预测向量 $\hat{\boldsymbol{y}}^{(t)}$ 中选取对应于类别 $\boldsymbol{y}^{(t)}$ 的那一项得到。模型的训练目标是使这个预测概率尽可能接近 1，即尽量增强对正确标签的预测置信度。

接下来从最后一个时间步 $t=\tau$ 开始，逐步计算每个时间步的梯度，直至 $t=1$ 。首先，计算输出层梯度：

$$\boldsymbol{\delta}_o^{(t)} = \frac{\partial L^{(t)}}{\partial \mathbf{o}^{(t)}} = \hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)} \tag{5.7}$$

其中，$\hat{\boldsymbol{y}}^{(t)}$ 是模型预测的概率分布，$\boldsymbol{y}^{(t)}$ 是真实标签的独热编码。

然后计算隐含层梯度：

$$\boldsymbol{\delta}_a^{(t)} = \frac{\partial L}{\partial \mathbf{a}^{(t)}} = \left(\mathbf{V}^{\mathrm{T}}\, \boldsymbol{\delta}_o^{(t)} + \mathbf{W}^{\mathrm{T}}\, \boldsymbol{\delta}_a^{(t+1)}\right) \odot \left(1 - [h^{(t)}]^2\right) \tag{5.8}$$

其中，$\odot$ 表示按元素相乘，边界条件为 $\boldsymbol{\delta}_a^{(\tau+1)} = \mathbf{0}$。

最后计算权重矩阵的梯度：

$$\frac{\partial L}{\partial \mathbf{U}} = \sum_{t=1}^{\tau} \boldsymbol{\delta}_a^{(t)}\, \boldsymbol{x}^{(t)\mathrm{T}} \tag{5.9}$$

$$\frac{\partial L}{\partial \mathbf{V}} = \sum_{t=1}^{\tau} \boldsymbol{\delta}_o^{(t)}\, \boldsymbol{h}^{(t)\mathrm{T}} \tag{5.10}$$

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{t=1}^{\tau} \boldsymbol{\delta}_a^{(t)}\, \boldsymbol{h}^{(t-1)\mathrm{T}} \tag{5.11}$$

最终利用梯度法更新权重矩阵。

反向传播的计算成本比较高昂。这一过程包括两部分：首先进行一次完整的前向传播（可以类比为从图 5-2 中左侧向右侧的展开计算），然后进行反向传播阶段，从序列的末尾依次回传误差。由于模型在每一个时间步的计算依赖于前一个时间步的隐藏状态，因此这些计算必须严格按照时间顺序进行，无法像其他网络结构那样进行时间维度上的并行处理。这导致整个训练过程的时间复杂度为 $O(\tau)$，其中 τ 表示序列的长度。此外，为了在反向传播时正确地计算梯度，前向传播过程中每个时间步的隐藏状态都需要被缓存，直到它们在反向过程中再次被访问。这也带来了线性增长的内存需求，其空间复杂度同样为 $O(\tau)$。因此，虽然循环连接赋予了 RNN 很强的序列建模能力，但这种结构也导致其训练过程在计算和存储方面的代价都相对较高。

5.1.3 深度循环网络

正如我们在卷积神经网络中通过增加多层卷积来提高性能，循环神经网络也可以通过引入更多层数来增强其表征能力。如图5-4所示，我们可以将多个循环神经网络层堆叠起来，

得到深度循环神经网络。在该架构中最浅层直接处理原始输入数据，随着层级的提升，数据被转化为更高层次的抽象表示。每一层都可以视为对输入数据不同级别的抽象表示，通过这种层次化的结构，深度循环神经网络能够更有效地捕捉输入数据中的复杂模式，从而更精准地建模复杂的序列数据。

设输入序列为 $X = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)})$，深度循环神经网络中第 l 层的隐藏状态表示为 $\boldsymbol{h}^{(t,l)}$。则第 l 层的前向传播公式如下。

对于第 1 层（底层）：

$$\boldsymbol{a}^{(t,1)} = \boldsymbol{b}^{(1)} + \boldsymbol{W}^{(1)}\boldsymbol{h}^{(t-1,1)} + \boldsymbol{U}^{(1)}\boldsymbol{x}^{(t)} \tag{5.12}$$

$$\boldsymbol{h}^{(t,1)} = \tanh(\boldsymbol{a}^{(t,1)}) \tag{5.13}$$

对于第 l 层（$l > 1$）：

$$\boldsymbol{a}^{(t,l)} = \boldsymbol{b}^{(l)} + \boldsymbol{W}^{(l)}\boldsymbol{h}^{(t-1,l)} + \boldsymbol{U}^{(l)}\boldsymbol{h}^{(t,l-1)} \tag{5.14}$$

$$\boldsymbol{h}^{(t,l)} = \tanh(\boldsymbol{a}^{(t,l)}) \tag{5.15}$$

最后一层的输出为

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t,L)} \tag{5.16}$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}) \tag{5.17}$$

式中，L 是网络的总层数，$\boldsymbol{b}^{(l)}$ 是第 l 层的偏置向量，$\boldsymbol{W}^{(l)}$ 是第 l 层隐藏状态的权重矩阵，$\boldsymbol{U}^{(l)}$ 是第 l 层从前一层接收输入的权重矩阵，$\boldsymbol{h}^{(t,l)}$ 是第 l 层在时间步 t 的隐藏状态。

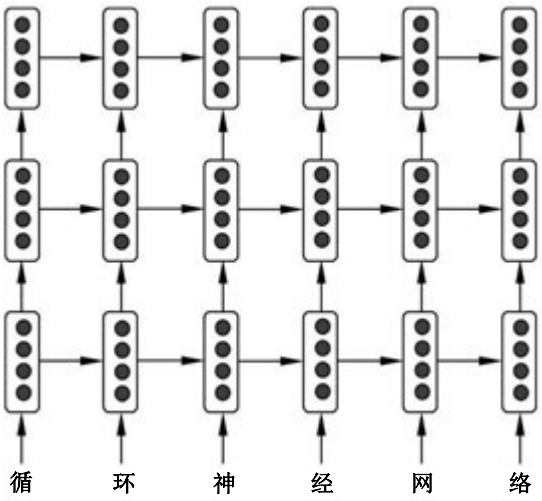

图 5-4　深度循环神经网络

5.1.4 双向循环网络

前面讨论的循环神经网络（RNN）结构，均具有“因果性”的设计特征：在时间步 t，模型的状态只依赖于当前输入 $\boldsymbol{x}^{(t)}$ 以及此前的输入序列 $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t-1)}$。然而，在很多自然语言处理任务中，当前的预测不仅与过去相关，也受到未来输入内容的影响。以命名实体识别（Named Entity Recognition, NER）为例，模型需要判断一个词或短语是否属于某种实体类型（如人名、地名、组织名等），而某个词的实体类别往往依赖于它后续出现的词语。例如，在短语“乔治·华盛顿大学”中，只有当模型看到“大学”之后，才能确认“乔治·华盛顿”在这里是学校名称的一部分，而不是人名。因此，仅依靠过去的上下文不足以完成精确判断。为了充分利用整个上下文，双向循环神经网络（Bidirectional RNN）应运而生。该结构结合了两个方向的RNN：一个从左到右处理序列，另一个从右到左处理相同输入，从而在每一个时间步都能同时访问前文和后文信息，使得模型在每个时间步都可以利用完整的输入序列信息，实现更精确的预测。

设双向循环神经网络的输入序列为 $\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)}\right)$，前向隐藏状态为 $\boldsymbol{h}_{\mathrm{f}}^{(t)}$，后向隐藏状态为 $\boldsymbol{h}_{\mathrm{b}}^{(t)}$，输出为 $\boldsymbol{o}^{(t)}$，则双向循环神经网络的前向传播公式如下：

前向循环神经网络：

$$\boldsymbol{a}_{\mathrm{f}}^{(t)} = \boldsymbol{b}_{\mathrm{f}} + \boldsymbol{W}_{\mathrm{f}} \boldsymbol{h}_{\mathrm{f}}^{(t-1)} + \boldsymbol{U}_{\mathrm{f}} \boldsymbol{x}^{(t)} \tag{5.18}$$

$$\boldsymbol{h}_{\mathrm{f}}^{(t)} = \tanh(\boldsymbol{a}_{\mathrm{f}}^{(t)}) \tag{5.19}$$

后向循环神经网络：

$$\boldsymbol{a}_{\mathrm{b}}^{(t)} = \boldsymbol{b}_{\mathrm{b}} + \boldsymbol{W}_{\mathrm{b}} \boldsymbol{h}_{\mathrm{b}}^{(t+1)} + \boldsymbol{U}_{\mathrm{b}} \boldsymbol{x}^{(t)} \tag{5.20}$$

$$\boldsymbol{h}_{\mathrm{b}}^{(t)} = \tanh(\boldsymbol{a}_{\mathrm{b}}^{(t)}) \tag{5.21}$$

结合前向和后向隐藏状态：

$$\boldsymbol{h}^{(t)} = [\boldsymbol{h}_{\mathrm{f}}^{(t)}, \boldsymbol{h}_{\mathrm{b}}^{(t)}] \tag{5.22}$$

输出层计算：

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V} \boldsymbol{h}^{(t)} \tag{5.23}$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)}) \tag{5.24}$$

图5-5展示了一个典型的双向循环神经网络，分为前向循环神经网络以及后向循环神经网络两部分，两部分分别计算后，把向前和向后循环神经网络的隐藏状态拼接在一起作为输出层的输入。这种结构允许输出单元 $\boldsymbol{o}^{(t)}$ 同时依赖于过去和未来的输入值。$\boldsymbol{b}_{\mathrm{f}}$ 和 $\boldsymbol{b}_{\mathrm{b}}$ 分别是前向和后向的偏置向量，$\boldsymbol{W}_{\mathrm{f}}$ 和 $\boldsymbol{W}_{\mathrm{b}}$ 是前向和后向的隐藏状态权重矩阵，$\boldsymbol{U}_{\mathrm{f}}$ 和 $\boldsymbol{U}_{\mathrm{b}}$ 是输入到隐藏状态的权重矩阵。通过这种方式，双向循环神经网络能够更好地捕捉输入序列中的上下文信息，从而提高了在复杂依赖关系下的建模能力，在需要双向信息依赖的场景中被广泛采用。

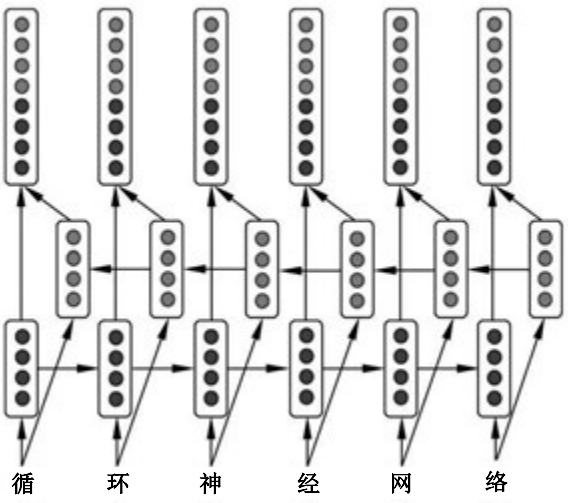

图 5-5　双向循环神经网络

5.1.5　基于编码-解码的序列到序列架构

此前我们已经介绍了两种常见的循环神经网络结构：多对多和多对一。前者适用于输入和输出序列长度一致的任务，如词性标注；后者则适合将整个序列压缩为单个输出的场景，如情感分类。然而，在诸如语音识别、机器翻译、问答等复杂任务中，模型所需处理的输入序列和输出序列通常长度不一致，尽管它们之间可能存在关联。

为了处理这种变长输入与变长输出的情况，我们引入一种更通用的结构——编码器-解码器架构（Encoder-Decoder），也被称为序列到序列（Sequence-to-Sequence）模型，如图 5-6 所示。这个架构由两个主要部分组成：编码器负责读取整个输入序列，并将其压缩为一个包含全局上下文信息的向量 $\boldsymbol{C}$；这个上下文向量通常是编码器最后一个隐藏状态的函数。随后，解码器接收该上下文向量，并逐步生成目标输出序列。该方法为解决输入输出长度不一致的问题提供了灵活有效的解决方案。

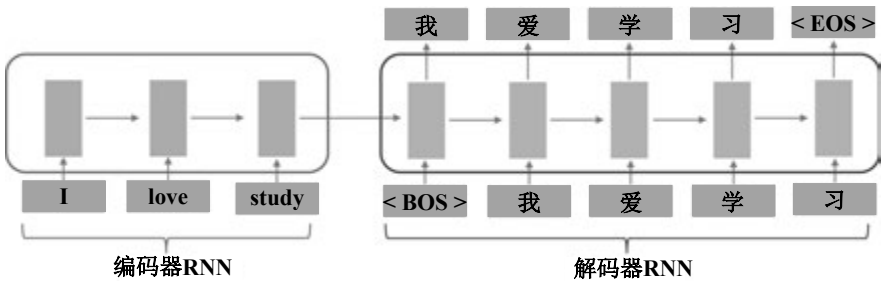

图 5-6　机器翻译任务为例，编码-解码的序列到序列架构的循环神经网络

编码器负责将输入序列 X 转换为上下文向量 $\boldsymbol{C}$，对于时间步 t（$t = 1, 2, \cdots, n_x$）：

$$\boldsymbol{a}_{\text{enc}}^{(t)} = \boldsymbol{b}_{\text{enc}} + \boldsymbol{W}_{\text{enc}}\boldsymbol{h}_{\text{enc}}^{(t-1)} + \boldsymbol{U}_{\text{enc}}\boldsymbol{x}^{(t)} \tag{5.25}$$

$$\boldsymbol{h}_{\text{enc}}^{(t)} = \tanh(\boldsymbol{a}_{\text{enc}}^{(t)}) \tag{5.26}$$

最终，编码器的隐藏状态 $\boldsymbol{h}_{\text{enc}}^{(n_x)}$ 被用作上下文向量 $\boldsymbol{C}$：

$$\boldsymbol{C} = \boldsymbol{h}_{\text{enc}}^{(n_x)} \tag{5.27}$$

解码器负责根据上下文向量 $\boldsymbol{C}$ 生成输出序列 Y，对于时间步 t（$t = 1, 2, \cdots, n_y$）：

$$\boldsymbol{a}_{\text{dec}}^{(t)} = \boldsymbol{b}_{\text{dec}} + \boldsymbol{W}_{\text{dec}}\boldsymbol{h}_{\text{dec}}^{(t-1)} + \boldsymbol{U}_{\text{dec}}\boldsymbol{y}^{(t-1)} + \boldsymbol{V}_{\text{dec}}\boldsymbol{C} \tag{5.28}$$

$$\boldsymbol{h}_{\text{dec}}^{(t)} = \tanh(\boldsymbol{a}_{\text{dec}}^{(t)}) \tag{5.29}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}_{\text{dec}}^{(t)} \tag{5.30}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)}) \tag{5.31}$$

其中，$\boldsymbol{y}^{(t)}$ 是目标输出序列在时间步 t 的真实值。例如，在机器翻译任务中，$\boldsymbol{y}^{(t)}$ 可能是目标语言中第 t 个词语。$\hat{\boldsymbol{y}}^{(t)}$ 是模型在时间步 t 的预测输出。$\hat{\boldsymbol{y}}^{(t)}$ 是通过 softmax 函数计算得到的概率分布，表示模型认为每个可能输出类别的概率。最终，模型会选择概率最大的类别作为当前时间步的预测结果。

训练时，编码器和解码器一起优化，我们通常使用交叉熵损失函数来最小化 $\hat{\boldsymbol{y}}^{(t)}$ 和 $\boldsymbol{y}^{(t)}$ 之间的差异。

序列到序列架构的编码器最终生成的上下文向量 $\boldsymbol{C}$ 是一个固定维度的压缩表示，这种压缩在处理较长的输入序列时会显得信息承载能力不足，导致模型性能下降。为了解决这一瓶颈，研究者提出了一种改进方法——将上下文表示从单一向量扩展为一个由多个向量组成的序列，从而保留更丰富的输入信息。在此基础上，他们引入了注意力机制（Attention Mechanism），使得解码器在生成每一个输出时，不再依赖于统一的上下文表示，而是能够根据当前解码状态，动态选择并聚焦于输入序列中最相关的部分。通过这一机制，模型能够更有效地捕捉长距离依赖，提升长句翻译等任务中的表现。

5.2 长短时记忆网络

5.2.1 长序列的挑战

深度网络所面临的梯度消失和梯度爆炸问题在前面已经介绍过。在训练循环神经网络时也面临类似的问题，这被称为长序列挑战，即信息在经过多个时间步传播后，梯度可能

会迅速减小到接近零，或者迅速增大到不稳定的程度，从而导致训练困难；以及随着时间步的增加，早期时间步的信息逐渐被后期信息覆盖，导致模型无法有效建模长序列早期关键信息。

上述问题的本质是循环神经网络中信息的传递方式涉及多次将相同的权重矩阵应用于先前时间步的隐藏状态，即 $\boldsymbol{h}^{(t)} = W^{\mathrm{T}}\boldsymbol{h}^{(t-1)}$。这种反复矩阵乘法导致梯度在传播过程中可能会指数级地减小或增大，并且导致早期时间步的信息被后来时间步信息覆盖。

特别地，如果 W 的特征值大多数都小于 1，那么梯度在传播过程中会指数级地减小，导致梯度消失。相反，如果 W 的特征值中有一些大于 1，那么梯度可能会指数级地增大，导致梯度爆炸。这两种情况都会导致训练问题。

除了难以训练，循环神经网络还存在长期遗忘问题。一般而言，来自远处的梯度信号远远小于来自近处的梯度信号，因此在训练过程中，模型的权重基本上只会针对近距离的影响而不是长期的影响进行更新，导致了长期依赖关系的衰退。在实际应用中，一个简单的循环神经网络通常只能依赖前面很少（约 7 个）词语的信息。

以预测下一个词的任务为例，考虑以下句子：“她试图打印她的机票，但发现打印机没有碳粉了。她去文具店买了更多的碳粉。之后，她终于打印出了她的_____”，这个任务要求模型能够理解“她的机票”和后续词语之间的依赖关系。然而，由于梯度消失问题，循环神经网络模型很难捕捉到这种长期的依赖，因此可能无法正确地预测空白处的单词。为了克服这个问题，研究人员提出了许多改进的循环神经网络架构，如长短时记忆（LSTM）网络和门控网络，以及各种训练技巧，以更好地捕获和利用长期依赖关系，提高模型在各种长序列任务中的性能。

5.2.2　长短时记忆网络

长短时记忆网络（LSTM）是一种带有独立内存单元的循环神经网络，用于应对传统模型面临的长序列挑战。LSTM 通过在每个时间步维护一个细胞状态（Cell State）和多个门控单元，实现对长期依赖信息的选择性保留与更新，从而能够在大约上百个时间步的序列上仍然保持稳定的梯度流动。

1．细胞状态与更新公式

细胞状态 $\boldsymbol{c}^{(t)}$ 是 LSTM 网络中的核心记忆载体（如图5-7所示），用于存储长期信息。它在时间步之间通过下列公式进行更新：

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \widetilde{\boldsymbol{c}}^{(t)} \tag{5.32}$$

其中，$\odot$ 表示逐元素相乘；$\boldsymbol{f}^{(t)}$ 为遗忘门的输出，决定遗忘多少上一时刻的细胞状态；$\boldsymbol{i}^{(t)}$ 为输入门的输出，决定写入多少新信息；$\widetilde{\boldsymbol{c}}^{(t)}$ 为候选细胞内容，通过当前输入和上一隐藏状

态生成：

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_{\mathrm{c}}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{c}}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_{\mathrm{c}}\right) \tag{5.33}$$

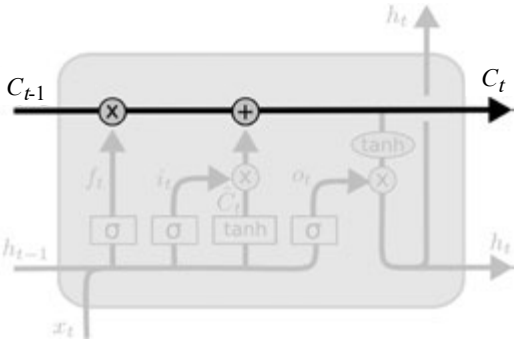

图 5-7　LSTM 的长期记忆通路

2. 门控机制

LSTM 引入了三种主要门控单元：遗忘门（Forget Gate）、输入门（Input Gate）和输出门（Output Gate）。它们均使用 sigmoid 激活函数，输出 0~1 之间的门控值。

遗忘门：

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_{\mathrm{f}}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{f}}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_{\mathrm{f}}\right) \tag{5.34}$$

控制细胞状态中哪些信息需要被遗忘。

输入门：

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_{\mathrm{i}}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{i}}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_{\mathrm{i}}\right) \tag{5.35}$$

决定新候选内容 $\tilde{\boldsymbol{c}}^{(t)}$ 中哪些部分写入细胞状态。遗忘门、输入门作用方式如图5-8所示。

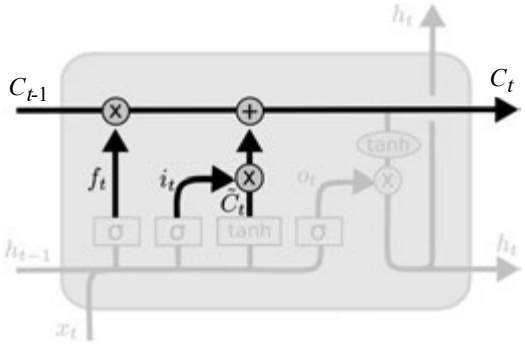

图 5-8　遗忘门、输入门、细胞状态的更新

输出门：

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_{\mathrm{o}}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{o}}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_{\mathrm{o}}\right) \tag{5.36}$$

决定从细胞状态输出到隐藏状态的信息比重，最终隐藏状态计算如下：

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tan h\left(\boldsymbol{c}^{(t)}\right) \tag{5.37}$$

输出门作用方式如图5-9所示。

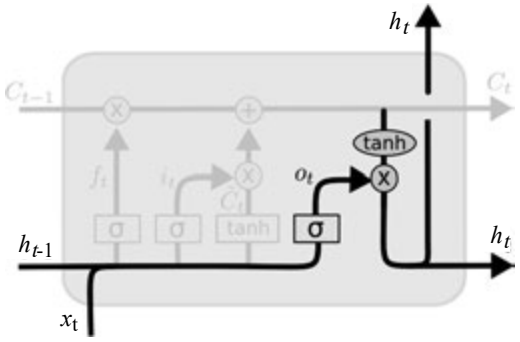

图 5-9　输出门、隐藏状态和输出

LSTM 的门控机制使得模型能够处理更长的时间步。相比于传统的循环神经网络，LSTM 能够轻松地处理具有约 100 个时间步的序列，而不会出现梯度消失的问题。这使得 LSTM 在各种序列建模任务中取得了显著的性能提升，尤其是自然语言翻译、语音识别和时间序列预测等任务。

5.3　门控网络

LSTM 在长序列建模任务上取得了显著的性能提升，但其结构较为复杂。为此，研究人员提出了一些替代架构，如门控循环单元（Gated Recurrent Unit，GRU）等，它们在一些任务上表现同样出色并且具有更简单的结构。GRU 可以看作一种精简版的门控循环神经网络。相对于 LSTM，GRU 合并了遗忘门和输入门，从而减少了参数量，其更新公式如下：

$$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}_{\mathrm{z}}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{z}}\boldsymbol{x}^{(t)}) \tag{5.38}$$

$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{\mathrm{r}}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_{\mathrm{r}}\boldsymbol{x}^{(t)}) \tag{5.39}$$

$$\widetilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}(\boldsymbol{r}^{(t)} \odot \boldsymbol{x}^{(t)})) \tag{5.40}$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \widetilde{\boldsymbol{h}}^{(t)} \tag{5.41}$$

可以看到，GRU 主要包括以下组成部分。

（1）重置门 $\boldsymbol{r}^{(t)}$：控制当前输入和前一个隐藏状态对当前隐藏状态的影响程度。

（2）更新门 $\boldsymbol{z}^{(t)}$：控制前一个隐藏状态和新的候选隐藏状态的权重。

（3）候选隐藏状态 $\widetilde{\boldsymbol{h}}^{(t)}$：由当前输入和重置门控制的前一个隐藏状态的信息组合而成。

（4）隐藏状态 $\boldsymbol{h}^{(t)}$：通过更新门控制的前一个隐藏状态和候选隐藏状态的组合来计算。

GRU 的结构相对简单，参数较少，在某些情况下表现得与 LSTM 相当，甚至更好。此外，由于其简洁性，GRU 更容易训练和部署，同时减少了过拟合的风险。

总之，LSTM 和 GRU 都是循环神经网络的经典架构。2017 年，谷歌提出了更简单有效的架构 Transformer，成为了更强有力的竞争者，我们将在下一章重点介绍。不过，选择哪种架构取决于任务的性质和数据的特点。

5.4 带外置记忆的循环网络

外置存储器网络（External Memory Network）是一种特殊的循环神经网络，旨在通过引入显式的外部记忆单元，增强模型的记忆能力和处理复杂序列任务的能力。这种模型在处理具有长期依赖性的序列数据，如机器翻译、问题回答、语言建模以及其他时间序列任务时表现出色。其中，神经图灵机（Neural Turing Machine, NTM）[98] 为这一类循环神经网络的代表。

NTM 从图灵机的设计哲学中汲取灵感，如图5-10所示，其结构主要包括以下几个部分。

（1）控制器：控制器是一个循环神经网络，通常是 LSTM 或 GRU，负责管理外部存储器的读取和写入，即根据输入序列和当前状态生成读操作和写操作，以交互并更新外部存储器。

（2）外部存储器（External Memory）：这是模型的主要记忆单元，由一个可变大小的矩阵构成，可以存储各种数据，例如键-值对，以便后续查询和检索。

（3）读操作（Read Operation）：读操作定义了如何从外部存储器中检索信息，包括查找方式以及如何将检索到的信息传递给控制器。

（4）写操作（Write Operation）：写操作定义了如何将新信息写入外部存储器，包括决定何时写入、写入的内容以及写入方式。

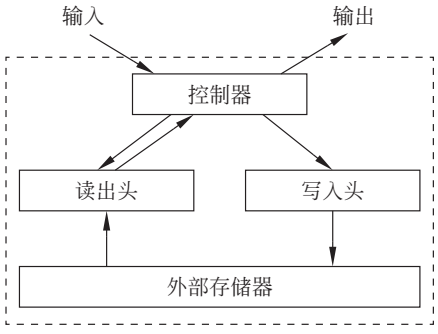

图 5-10 NTM 结构示意图[98]

NTM 在工作时，输入序列首先通过控制器，控制器生成读操作和写操作的指令；读操作通过外部存储器，使用注意力机制检索存储器中的相关信息，然后将这些信息返回给控

制器；控制器将读取的信息与输入数据进行组合，然后生成输出；写操作根据需要将新信息写入外部存储器。

在 NTM 的应用场景中，一个有趣的例子是识别括号匹配。解决这个问题涉及栈匹配算法的使用，随着左括号的输入将其入栈，然后尝试去匹配与之对应的右括号。传统的神经网络模型难以完成这类任务，而 NTM 则可以将入栈左括号存入外部存储器，在右括号出现时进行读取并匹配。

外置存储器网络不仅在各种自然语言处理任务中表现出色，还适用于时间序列预测、游戏策略学习和其他需要处理长序列和复杂依赖性的任务。

5.5　Mamba 架构

Mamba 模型[99] 是一种从状态空间模型衍生而来的新兴网络架构，它与循环神经网络和 Transformer 模型有着深刻而密切的联系。接下来简要介绍 Mamba 模型的构成及其基本性质。

Mamba 模型的基本构建单元是 Mamba 块（Mamba Block）（如图5-11所示），主要包括以下几点。

（1）线性扩展（Linear Expansion）：对输入进行线性变换，扩展其维度以适应后续处理。

（2）局部卷积（Local Convolution）：应用一维卷积捕捉局部上下文信息。

（3）选择性状态空间模型（Selective SSM）：核心模块，基于输入执行状态更新。

（4）线性投影（Linear Projection）：将 SSM 的输出映射回原始维度。

（5）残差连接与归一化（Residual Connection and Normalization）：增强训练稳定性和模型深度。

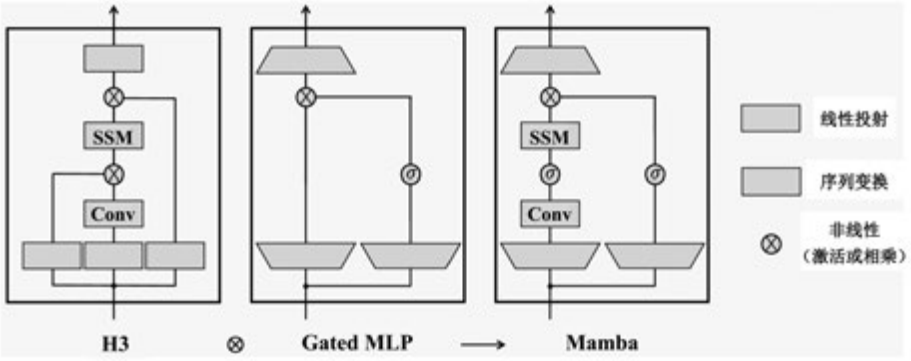

图 5-11　Mamba 块示意图[99]

Mamba模型最为核心的组件是选择性状态空间（Selective State Space），其计算过程可以表达为如下数学形式：

$$\begin{aligned} h_i &= \overline{A}_i h_{i-1} + \overline{B}_i x_i, \qquad x_i \in \mathbb{R},\ \ \overline{A}_i \in \mathbb{R}^{d\times d},\ \ \overline{B}_i, h_{i-1}, h_i \in \mathbb{R}^{d\times 1} \\ y_i &= C_i h_i + D x_i, \qquad\quad y_i \in \mathbb{R},\ \ C_i \in \mathbb{R}^{1\times d},\ \ D \in \mathbb{R} \end{aligned} \tag{5.42}$$

式中，x_i, y_i, h_i 分别为输入、输出变量和隐藏状态；$\overline{A}_i, \overline{B}_i, C_i, D$ 是离散状态方程的4个参数；d 是模型隐藏状态维度。参数 $\overline{A}_i, \overline{B}_i$ 是由连续状态方程的参数通过时间尺度 Δ_i 离散化得到的：

$$\overline{A}_i = \exp(\Delta_i A), \quad \overline{B}_i = (\Delta_i A)^{-1}(\exp(\Delta_i A) - I) \cdot \Delta_i B_i \approx \Delta_i B_i \tag{5.43}$$

尽管Mamba模型衍生于状态空间模型，其数学形式（式(5.42)）与传统的循环神经网络十分相似。具体来说，$\overline{B}_i$ 可以视为从输入变量映射到隐藏状态的矩阵；$\overline{A}_i$ 可以看作对隐藏状态进行变换的遗忘门；而 C_i 则等效于从隐藏状态映射到输出变量的矩阵。

Mamba架构具有良好的表征能力，部分研究者认为它是当前深度学习主流架构Transformer的有力挑战者。相较于传统的循环神经网络和Transformer模型，Mamba模型主要有以下两方面的优势。

（1）关于序列长度 τ 具有线性计算复杂度。从式(5.42)可以看到，Mamba模型采用类似于循环神经网络的方式逐一处理输入序列中的每一个特征，处理每一个特征所需要的空间和时间复杂度不会随序列长度增长而改变。因此，Mamba模型能够以 $\mathcal{O}(\tau)$ 的计算复杂度实现对整个序列的建模。

（2）模型表达能力较好。Mamba模型中的参数（B, C）是由输入 x 经线性映射得到，这使得模型能够动态地将输入特征编码到状态空间，进而动态解码当前输出。这种参数动态性使得Mamba模型展现出比传统循环神经网络更优的性能，能够更有效地处理序列输入。

然而，Mamba模型也存在一些不足。例如，Mamba模型与传统的循环神经网络一样，将所有输入信息映射到有限的隐藏空间中，这会不可避免地导致一些信息损失，在长文本、高分辨率图片等应用场景中仍可能面临挑战。

5.6 习题

（1）循环神经网络的两种基本模式分别是什么？各适用于什么任务？

（2）为什么循环神经网络训练的内存开销大？什么是“通过时间反向传播”？

（3）请写出第 l 层深度循环网络的计算公式。

（4）请描述双向循环神经网络的原理。

（5）基于编码-解码的序列到序列架构分为哪些部分？请给出具体计算过程？

（6）循环神经网络的梯度消失和梯度爆炸是如何产生的？通过怎样设计可以避免？

（7）LSTM 中细胞状态与隐藏状态的区别是什么？

（8）请具体比较 GRU（门控循环单元）和 LSTM（长短期记忆网络）的结构与工作机制，包括它们在参数数量、计算效率和记忆能力等方面的异同。

（9）NTM（神经图灵机）在结构上与传统神经网络（如 RNN、LSTM）有何不同？请结合一个实际任务（如复制、排序等）阐述 NTM 引入外部记忆机制的优势。

（10）Mamba 结构相比循环神经网络有哪些优势和不足？