第八讲 循环神经网络I

主讲教师：丛润民

北京交通大学 《深度学习》课程组

目录

目录

8.1 网络记忆能力

- 全连接神经网络和卷积神经网络
  - 只能单独的处理一个个的输入，前一个输入和后一个输入是完全没有关系的。
- 某些任务需要能够更好的处理**序列**的信息
  - 前面的输入和后面的输入是有关系的。比如，当理解一句话意思时，孤立的理解这句话的每个词是不够的，需要处理这些词连接起来的整个**序列**；
  - 当处理视频的时候，不能只单独的去分析每一帧，而要分析这些帧连接起来的整个**序列**。

实例1：语言模型

给定一句话前面的部分，预测接下来最有可能的一个词是什么

我昨天上学迟到了，老师批评了____。

分词结果:

我 昨天 上学 迟到 了 ， 老师 批评 了 ____。

8.1 网络记忆能力

我 昨天 上学 迟到 了 ， 老师 批评 了 ____。

- 2-Gram模型

  在语料库中，搜索『了』后面最可能的一个词

- 3-Gram模型

  搜索『批评了』后面最可能的词

- 4-Gram, 5-Gram, ..., N-Gram

  模型的大小和N的关系是指数级的，会占用海量的存储空间

而循环神经网络RNN理论上可以往前(后)看任意多个词

实例2: 槽填充(Slot Filling)问题

例如在一个自动订票系统中，用户输入“我想八月二号**到达**台北”，需要填充槽(slot)，

目 的 地：台北

到达时间：2020.08.02

如何使用前馈神经网络(Feedforward network)解决此问题?

李宏毅-机器学习-RNN https://speech.ee.ntu.edu.tw/~tlkagk/courses_ML19.html

首先使用1-of-N encoding方法将每一个单词表示成一个向量。

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

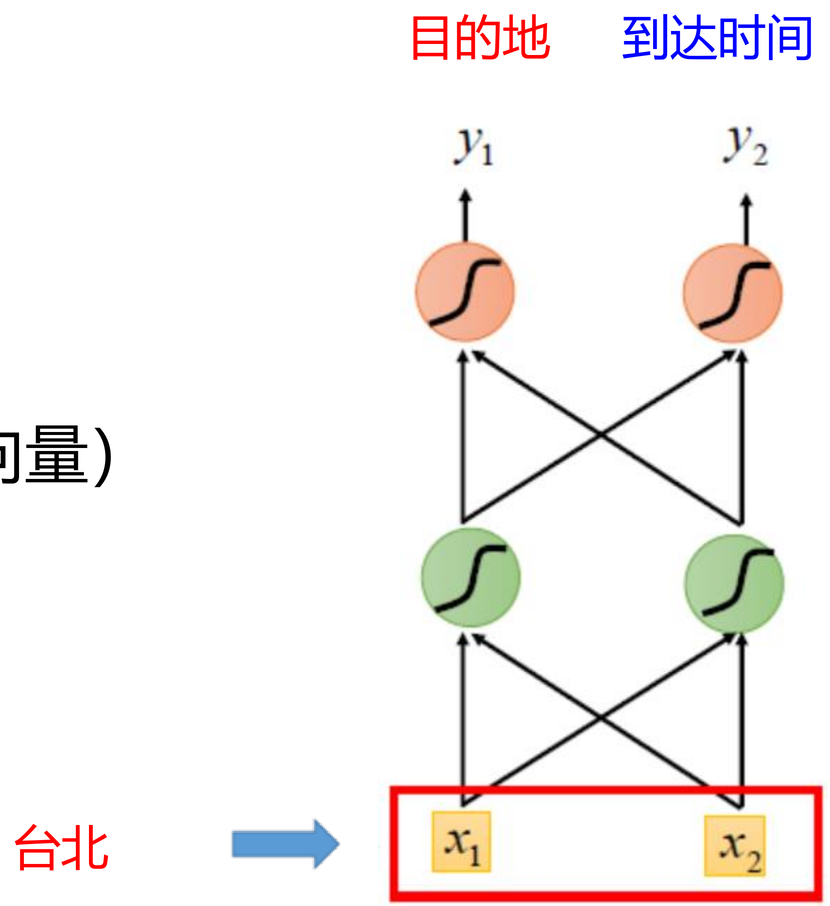
训练前馈神经网络，使得输入“台北”时，目的地的概率最大。

示例应用

通过前馈网络解决槽填充问题?

输入：一个单词 （每个单词都表示为一个向量）

输出：表示输入词属于槽的概率分布



8.1 网络记忆能力

若另一个用户输入“我想在八月二号**离开**台北”，需要填充槽(slot)，

出 发 地：台北
出发时间：2020.08.02

对于前馈神经网络，使得输入“台北”时，出发地的概率最大。

Problem：对于同一个输入，输出的概率分布应该也是一样的，不可能出现既是目的地的概率最高又是出发地的概率最高。

因此，我们希望神经网络拥有“记忆”（memory）的能力，能够根据之前的信息（如**到达**或**离开**）得到不同的输出。

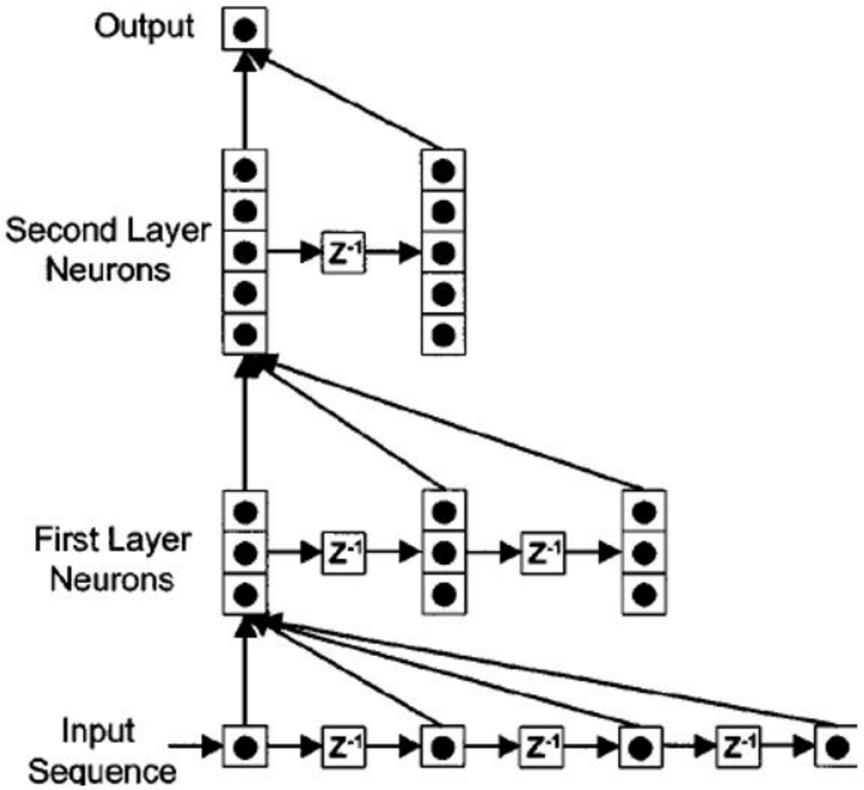
如何给网络增加记忆能力(除RNN外)?

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

建立一个额外的延时单元，用来存储网络的历史信息（可以包括输入、输出、隐状态等）

$$\boldsymbol{h}_t^{(l)} = f\left(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l-1)}, \cdots, \boldsymbol{h}_{t-K}^{(l-1)}\right)$$

这样，前馈网络就具有了短期记忆的能力



Jin C, Schenkel M, Carlile S. Neural system identification model of human sound localization[J]. The Journal of the Acoustical Society of America, 2000, 108(3): 1215-1235.

邱锡鹏.《神经网络和深度学习》

8.1 网络记忆能力

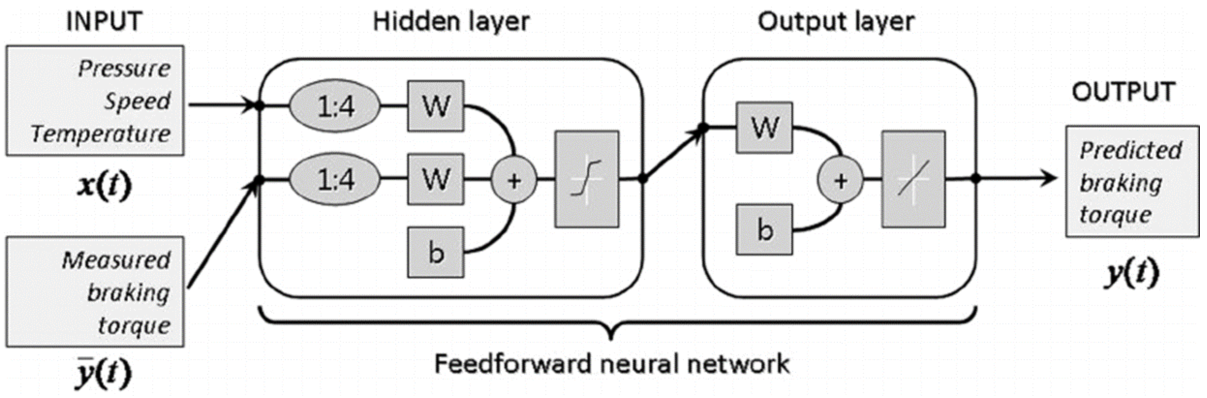
- **自回归模型（Autoregressive Model，AR）**：一类时间序列模型，用变量y_t的历史信息来预测自己

$$\boldsymbol{y}_t = w_0 + \sum_{k=1}^{K} w_k \boldsymbol{y}_{t-k} + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma^2)\text{为第t个时刻的噪声}$$

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

$$\boldsymbol{y}_t = f(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \cdots, \boldsymbol{x}_{t-K_x}, \boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2}, \cdots, \boldsymbol{y}_{t-K_y})$$

其中 $f(\cdot)$ 表示非线性函数，可以是一个前馈网络， K_x 和 K_y 为超参数.

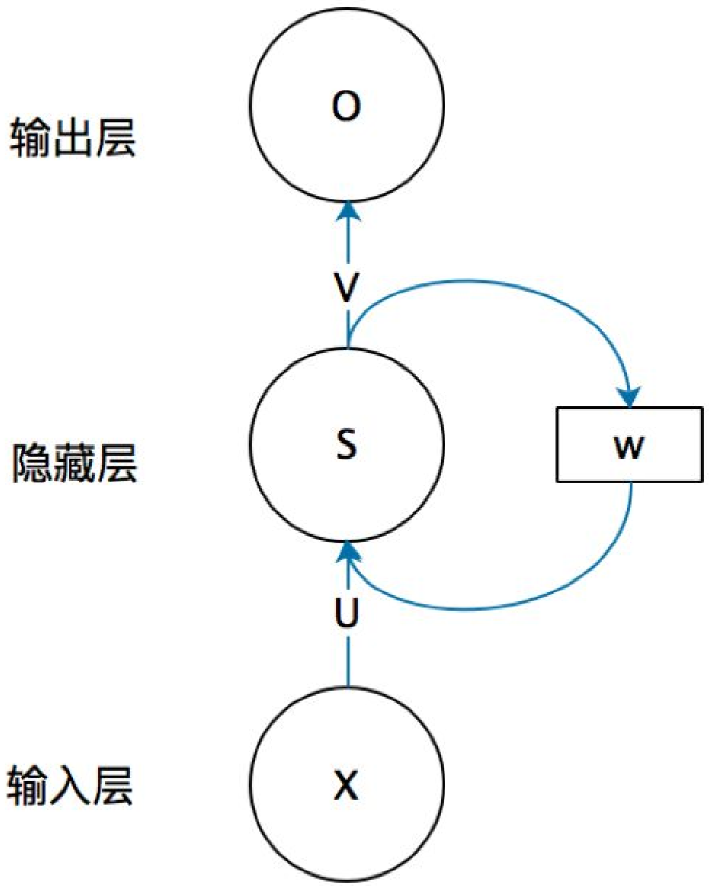


Leontaritis I, Billings S A, 1985. Input-output parametric models for non-linear systems part i: deterministic non-linear systems[J]. International journal of control, 41(2):303-328.

目录

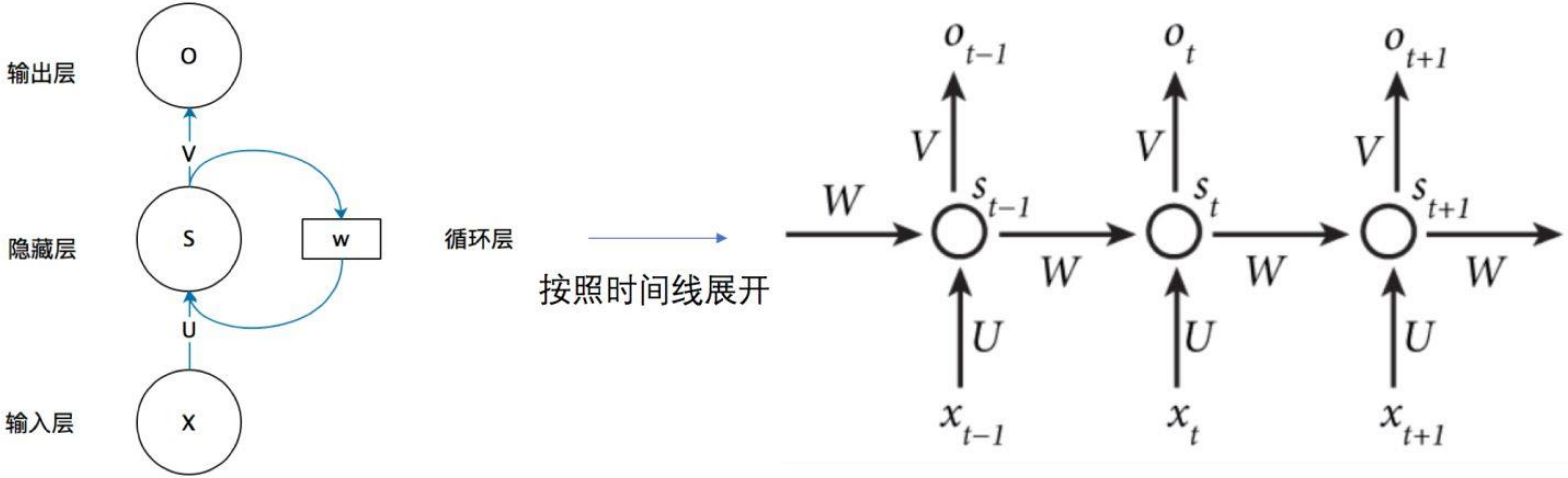
8.2 循环神经网络（RNN）

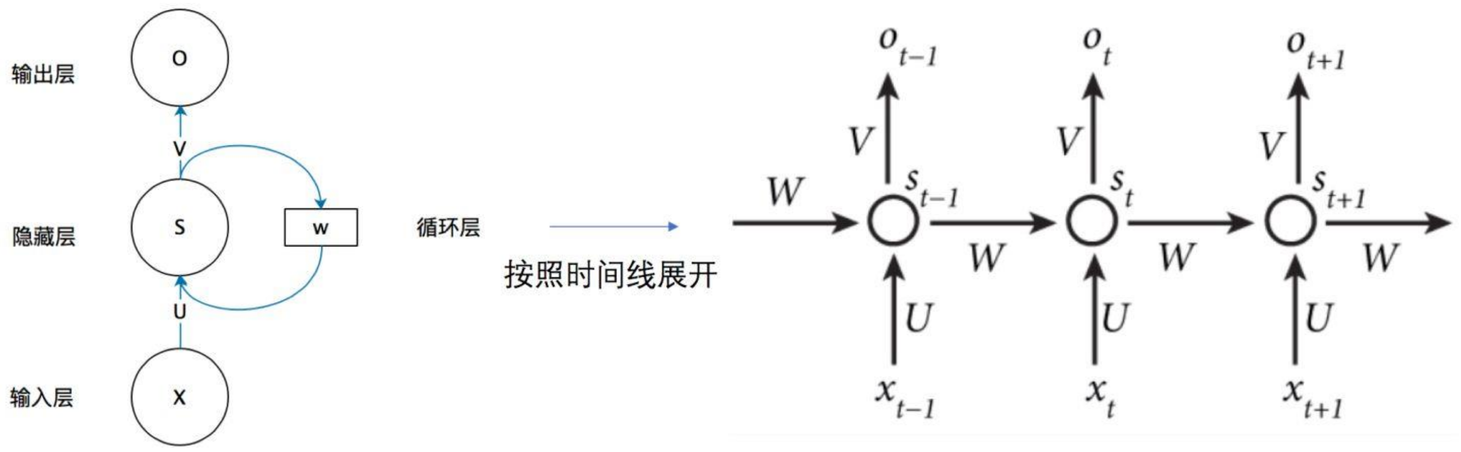
一个简单的循环神经网络由输入层、一个隐藏层和一个输出层组成。



- 去掉有W的带箭头的圈，它就变成普通的**全连接神经网络**
- x是一个向量，它表示**输入层**的值
- s是一个向量，它表示**隐藏层**的值， s不仅仅取决于当前这次的输入x，还取决于上一次**隐藏层**的值
- o也是一个向量，它表示**输出层**的值
- U是输入层到隐藏层的**权重矩阵**
- V是隐藏层到输出层的**权重矩阵**
- W是**隐藏层**上一次的值作为这一次的输入的**权重矩阵**

8.2 循环神经网络（RNN）





网络在t时刻接收到输入x_t之后，隐藏层的值是s_t，输出值是o_t。 s_t的值不仅仅取决于x_t，还取决于s_{t-1}。

循环神经网络的计算：

$$o_t = g(Vs_t) \qquad (式1)$$

$$s_t = f(Ux_t + Ws_{t-1}) \qquad (式2)$$

循环神经网络可以往前看任意多个**输入值**的原因

$$o_t = g(Vs_t) \quad \text{(式1)}$$
$$s_t = f(Ux_t + Ws_{t-1}) \quad \text{(式2)}$$

如果反复将（式2）带入（式1），我们将得到

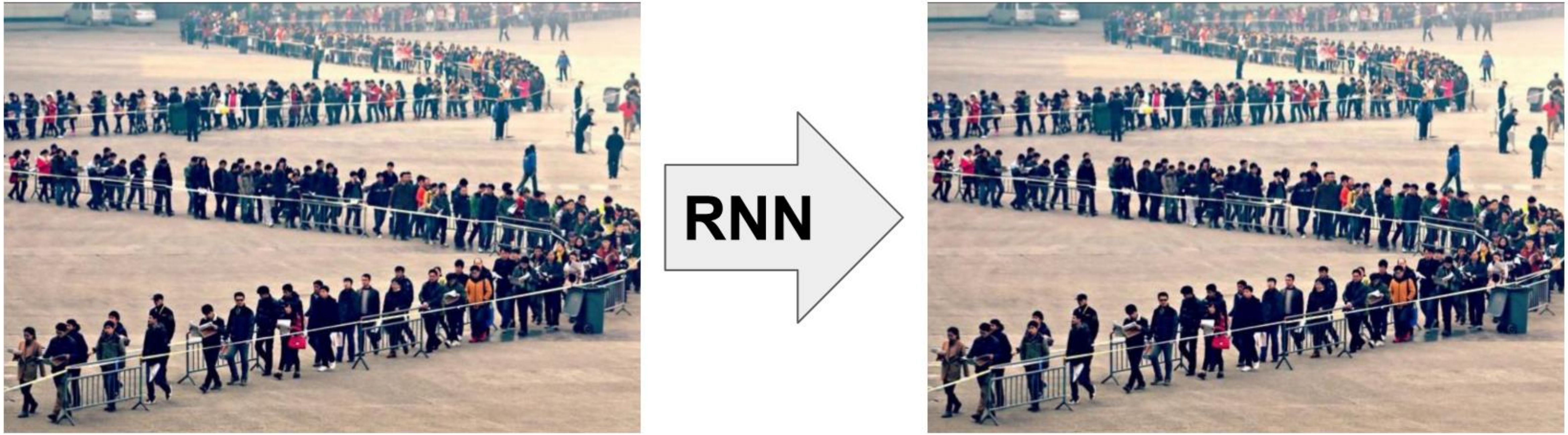
$$\begin{aligned} o_t &= g(Vs_t) \\ &= g(Vf(Ux_t + Ws_{t-1})) \\ &= g(Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2}))) \\ &= g(Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3})))) \\ &= g(Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))) \end{aligned}$$

即输出值o_t，受前面历次输入值x_t，x_{t-1}、x_{t-2}、x_{t-3}...的影响。

8.2 循环神经网络（RNN）

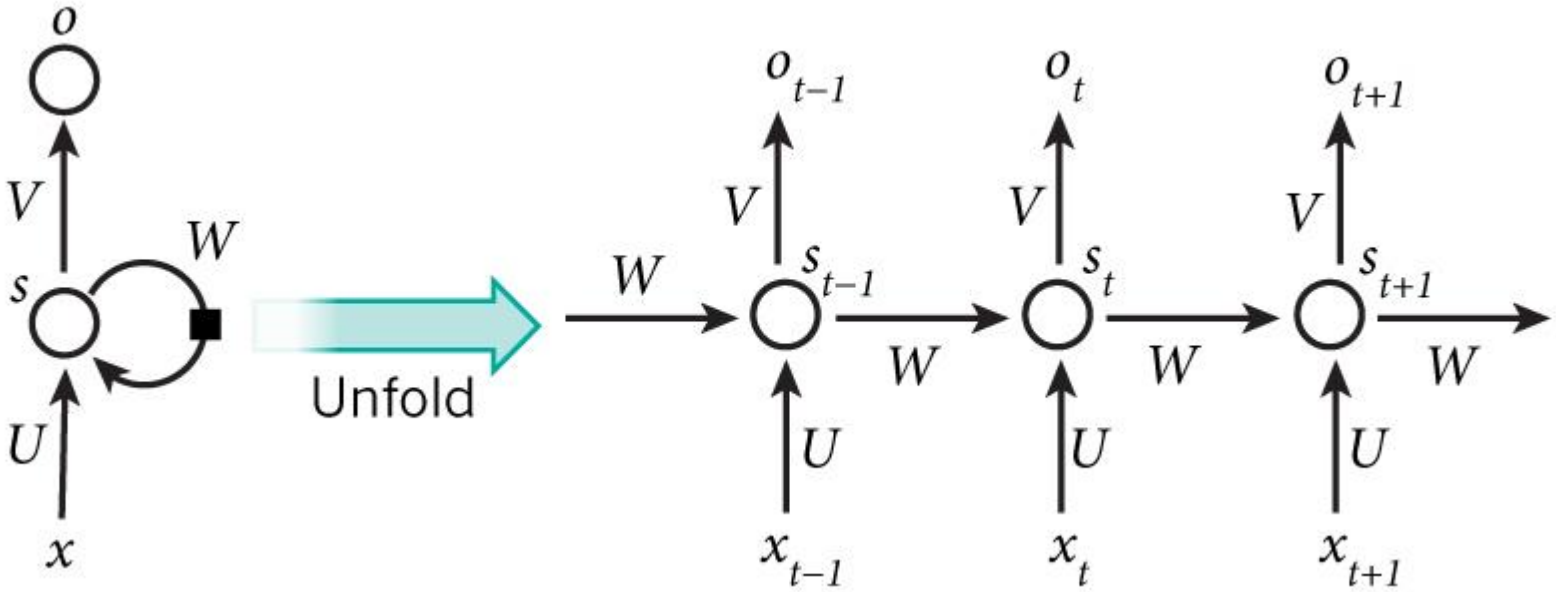
RNN是一个序列模型

- 如何将可变长度序列作为**输入**？
- 如何预测可变长度序列作为**输出**？



8.2 循环神经网络（RNN）

神经网络模型通过训练“学”到的东西蕴含在“权值”中。基础的神经网络只在层与层之间建立权连接，而RNN最大的不同之处在于隐藏层内的神经元也建立了权连接。

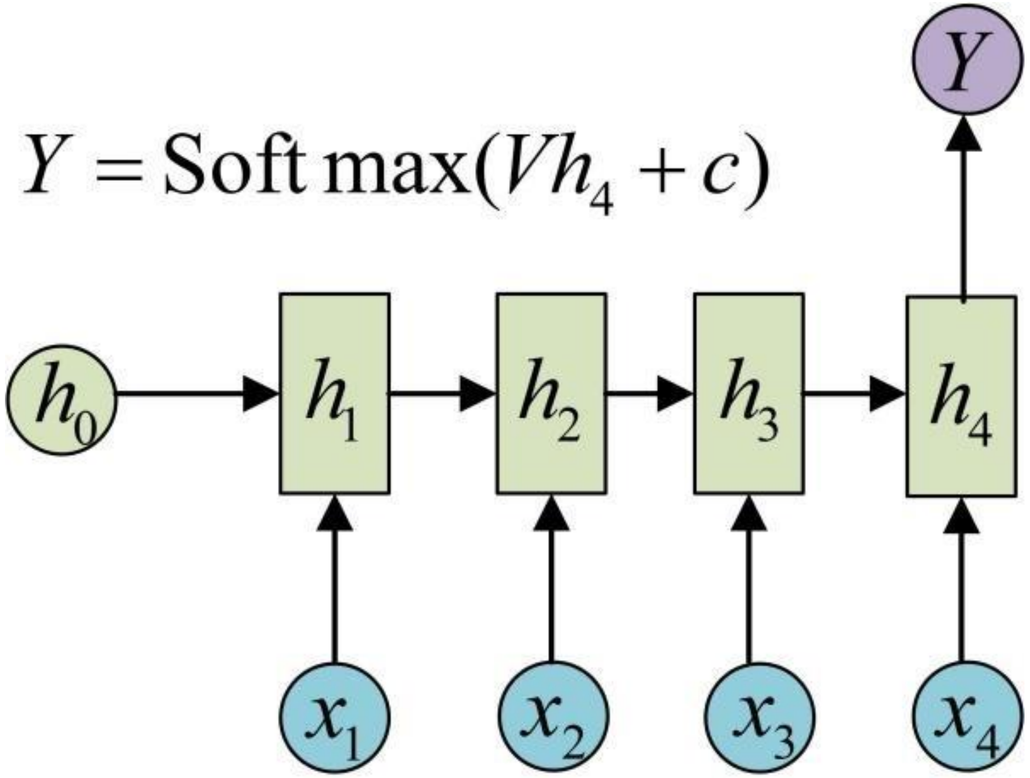


RNN的标准结构

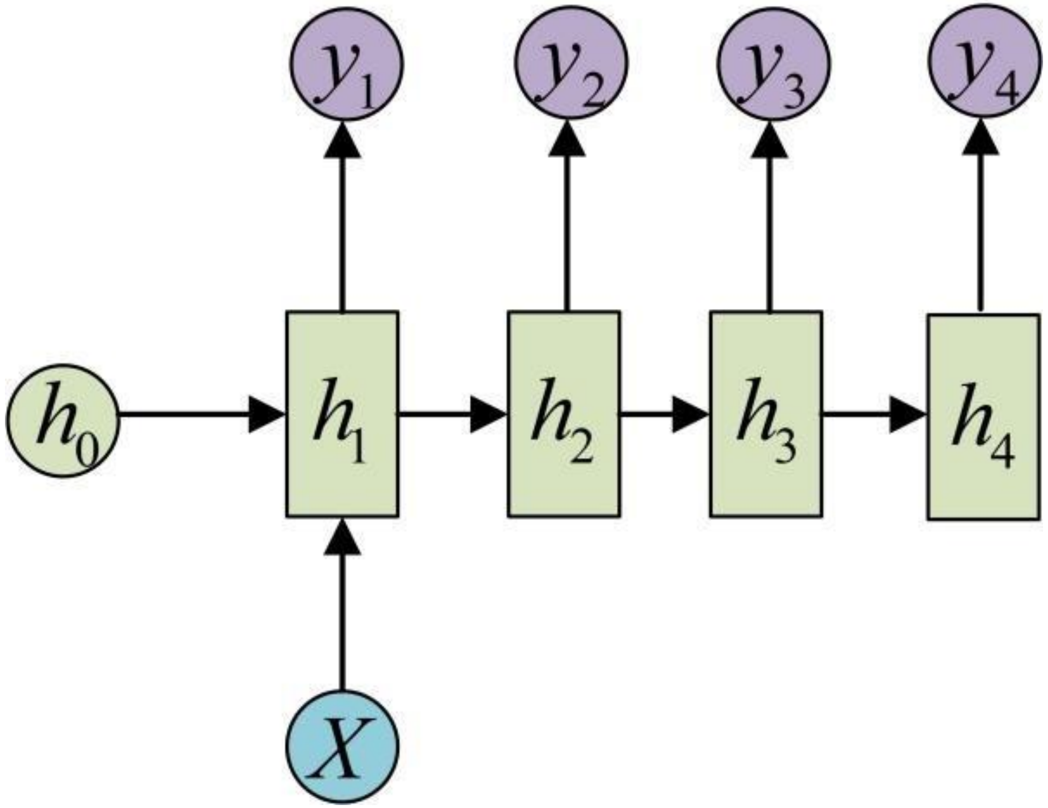
Many-to-One模型

在文本分类中，输入数据为单词的序列，输出为该文本的类别。

$$Y = \mathrm{Softmax}(Vh_4 + c)$$

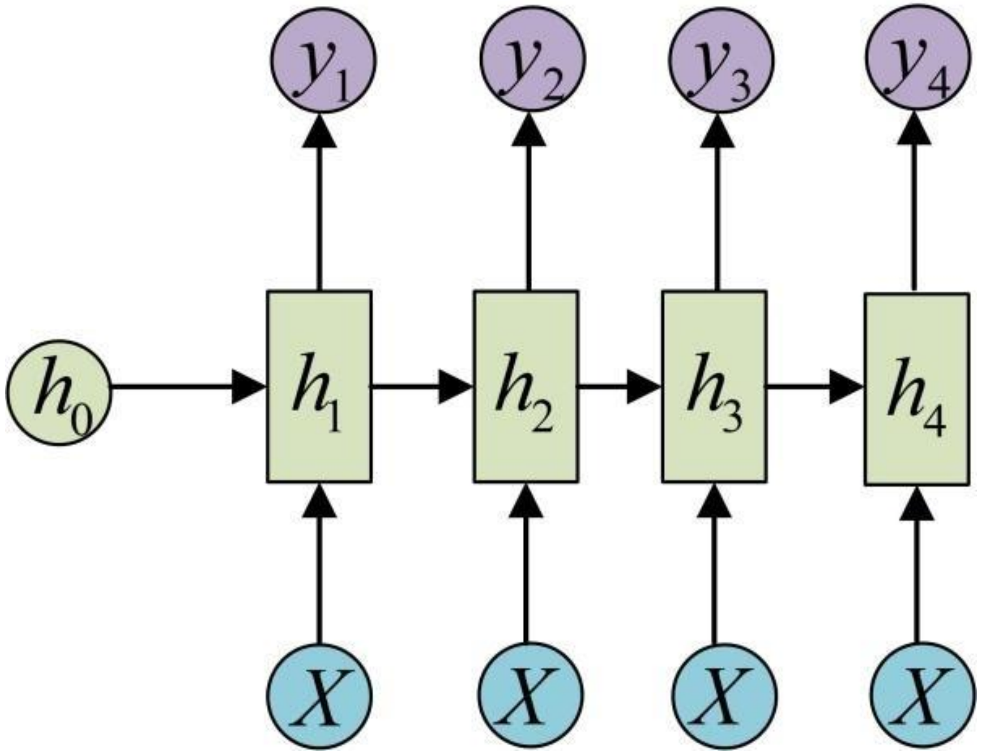


输入为一张图片，输出为图片的文字描述



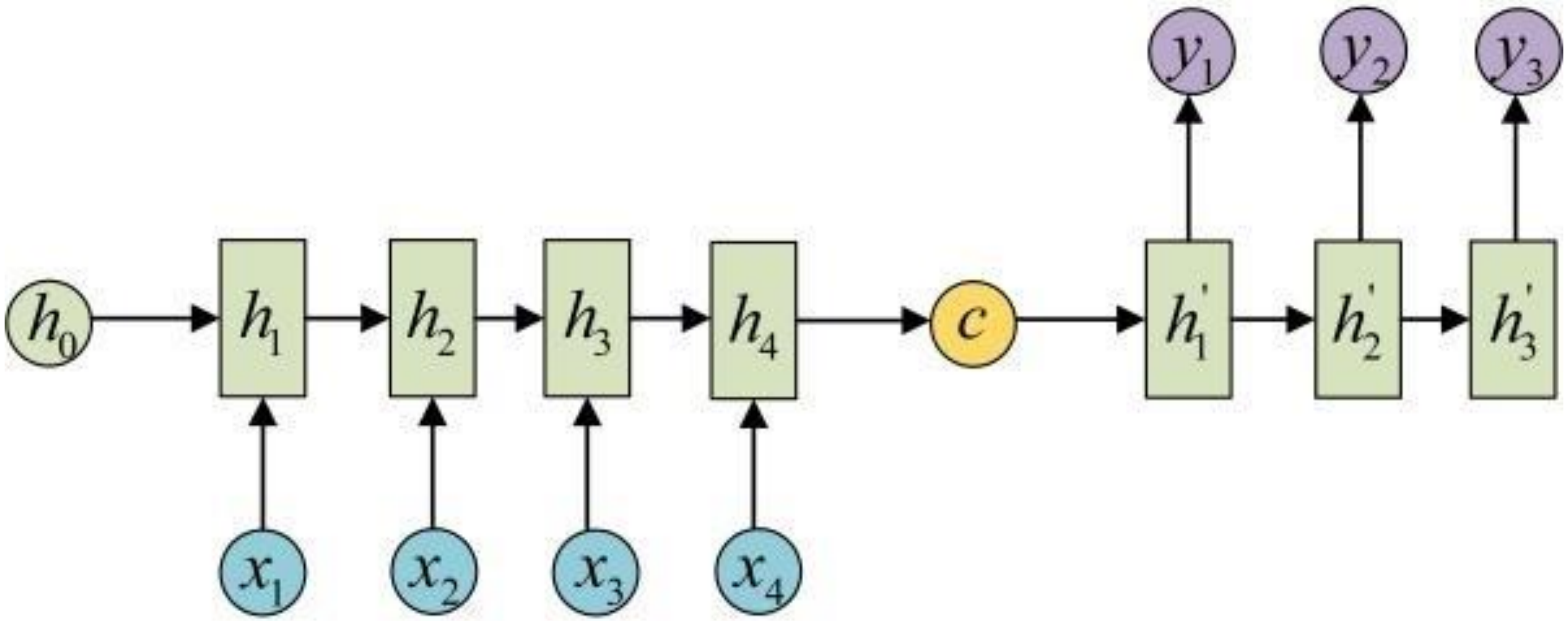
Many-to-Many模型

输入和输出的序列个数相同，如输入为视频序列，输出为每一帧对应的标签



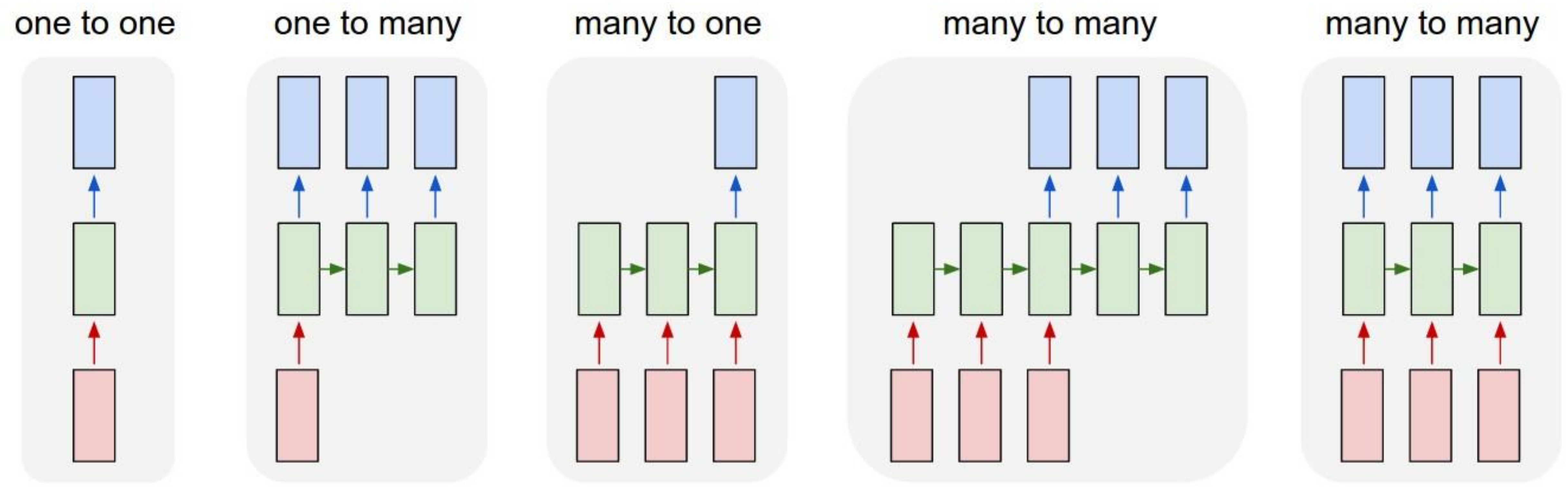
Many-to-Many模型

输入和输出的序列个数不同，如机器翻译中，源语言和目标语言的句子往往并没有相同的长度。



这种结构又叫Encoder-Decoder模型，也可以称之为Seq2Seq模型

8.2 循环神经网络（RNN）



编写诗歌

情感分析

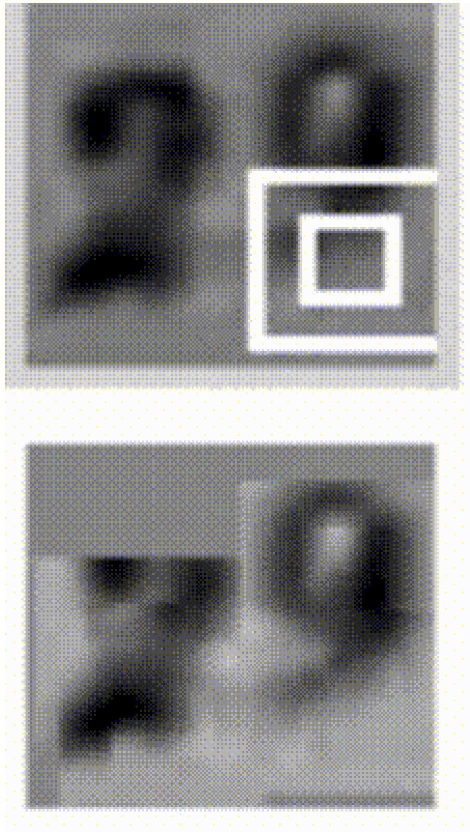
文本分类

自动问答

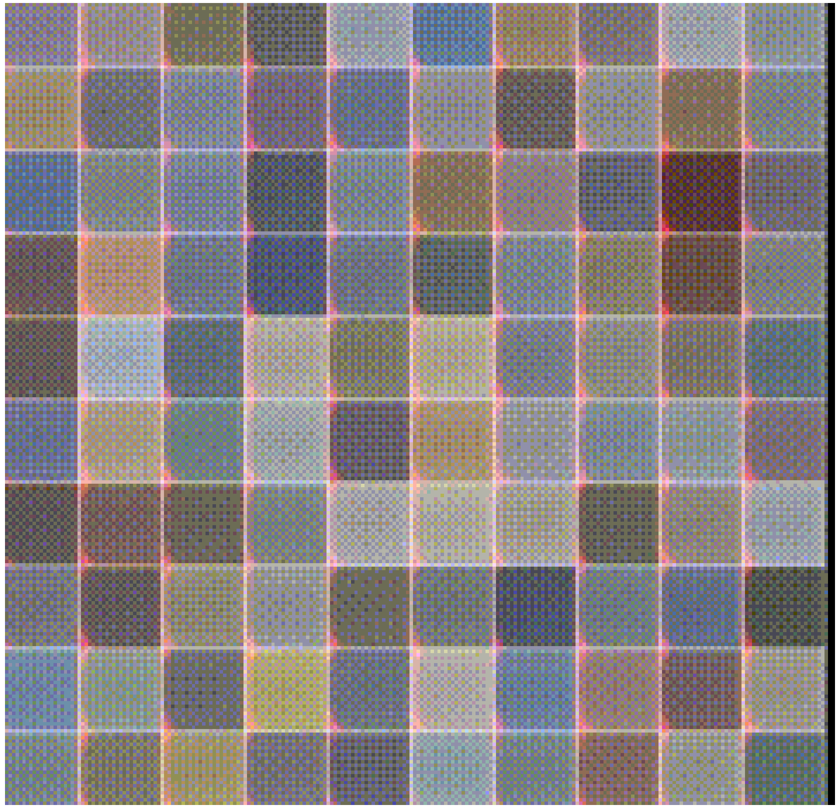
机器翻译

词性标记

RNN模型应用举例



RNN读门牌号



RNN绘制门牌号

8.2 循环神经网络（RNN）

RNN模型应用举例

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3128-3138.

8.2 循环神经网络（RNN）

RNN模型应用举例

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Shakespeare

RNN模型应用举例

[玉辇临遐远，林台向晚开。寒霜万里雨，古木白云台。杞梓长晴属，藩旌旋北来。朝山犹似盖，天仆日萧纷。]

[银阁为君贵，昭原气极功。不知能似面，合合不甘穷。]

[从容峰岸啸初开，曾道神功月会频。只为钱陵庄叟旧，不曾知我有何人。]

[梧桐叶暗如丝拍，一片寒波鬓似丝。与子近来弹铗嫁，羞他画领学蛾眉。]

[渴戍平千重帝城，郊园初见杜陵僧。壮仙若望煮舟马，石上萧遮两袖僧。]

RNN写诗

https://github.com/aipiano/PoemRNN

目录

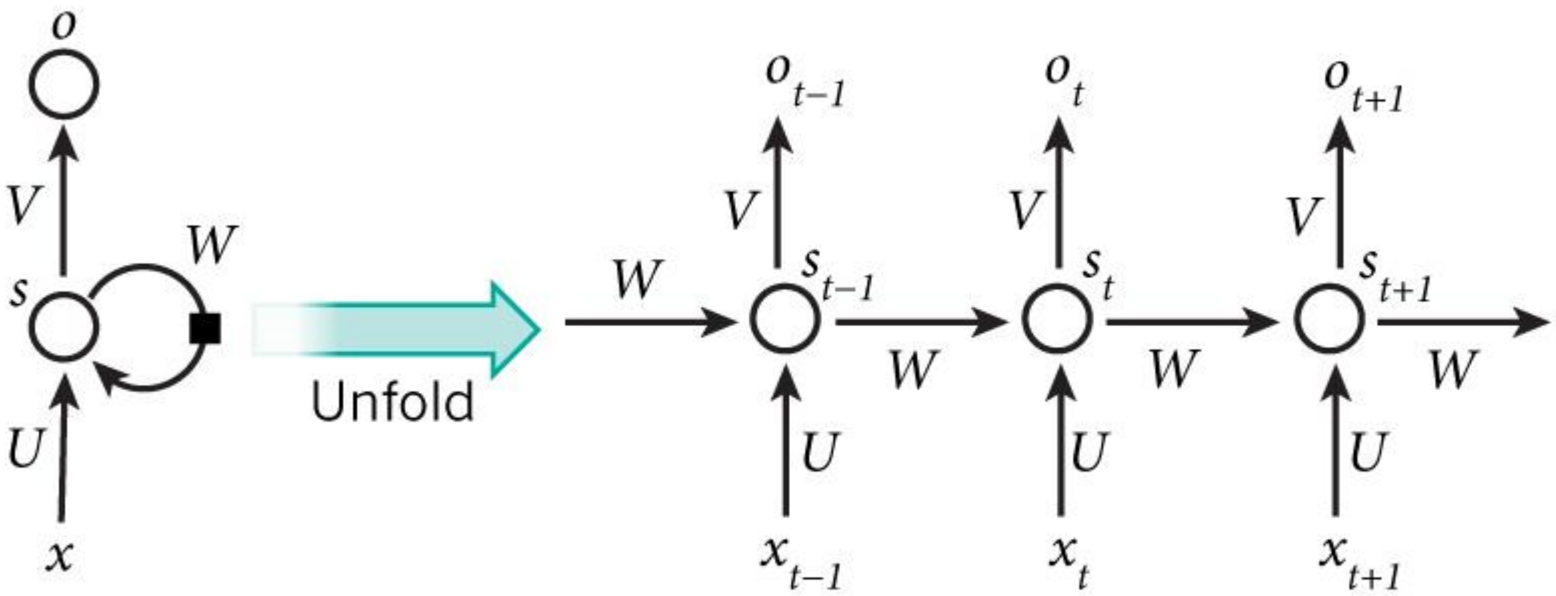
8.3 随时间反向传播（BPTT）

- BPTT（Back-Propagation Through Time）算法是常用的训练RNN的方法，因为RNN处理时间序列数据，所以要基于时间反向传播，故称作随时间反向传播。

- BPTT的中心思想是沿着需要优化的参数的负梯度方向不断寻找更优的点直至收敛（和BP算法相同）。

- BPTT算法本质上还是BP算法，BP算法本质是梯度下降法，那么求各个参数的梯度便成了此算法的核心。

8.3 随时间反向传播（BPTT）



需要寻优的参数有三个，分别是U、V、W。与BP算法不同的是，**权重矩阵W和U的寻优过程需要追溯之前的历史数据**（BPTT算法的重点）。

8.3 随时间反向传播（BPTT）

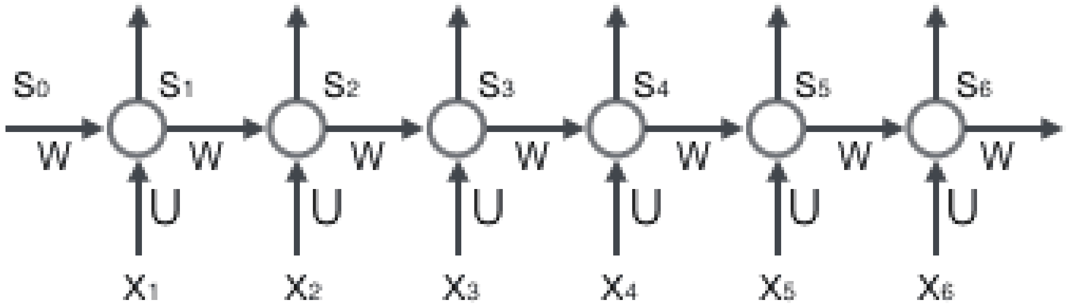
BPTT算法是针对**循环层**的训练算法，它的基本原理和BP算法是一样的，也包含同样的三个步骤：

1. 前向计算每个神经元的输出值；

2. 反向计算每个神经元的**误差项**δ_j值，它是误差函数E对神经元j的**加权输入**net_j的偏导数；

3. 计算每个权重的梯度。

最后再用**随机梯度下降**算法更新权重。



RNN的循环层

A. 前向计算

使用前面的**式2**对循环层进行前向计算：

$$\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1})$$

注意，上面的s_t、x_t、s_{t-1}都是向量，用**黑体字母**表示；而U、V是**矩阵**，用大写字母表示。**向量的下标**表示**时刻**，例如， s_t表示在t时刻向量s的值。

假设输入向量**x**的维度是m，输出向量**s**的维度是n，则矩阵U的维度是n×m，矩阵W的维度是n×n，下面是上式展开成矩阵的样子： $\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1})$

$$\begin{bmatrix} s_1^t \\ s_2^t \\ . \\ s_n^t \end{bmatrix} = f\left(\begin{bmatrix} u_{11} u_{12} \ldots u_{1m} \\ u_{21} u_{22} \ldots u_{2m} \\ . \\ . \\ u_{n1} u_{n2} \ldots u_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_m \end{bmatrix} + \begin{bmatrix} w_{11} w_{12} \ldots w_{1n} \\ w_{21} w_{22} \ldots w_{2n} \\ . \\ . \\ w_{n1} w_{n2} \ldots w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ . \\ s_n^{t-1} \end{bmatrix} \right)$$

B. 误差项的计算

BPTT算法将第l层t时刻的**误差项**δ_t^l值沿两个方向传播：

一个方向是其传递到上一层网络，得到δ_t^{l-1}，这部分只和权重矩阵U有关；

另一个方向是将其沿时间线传递到初始t_1时刻，得到δ_1^l，这部分只和权重矩阵W有关。

用向量net_t表示神经元在t时刻的**加权输入（净输入）**，因为：

$$\text{net}_t = U\mathrm{x}_t + W\mathrm{s}_{t-1}$$
$$\mathrm{s}_{t-1} = f(\text{net}_{t-1})$$

因此：

$$\frac{\partial \text{net}_t}{\partial \text{net}_{t-1}} = \frac{\partial \text{net}_t}{\partial \mathrm{s}_{t-1}} \frac{\partial \mathrm{s}_{t-1}}{\partial \text{net}_{t-1}}$$

用a表示列向量，用a^T表示行向量，上式的第一项是向量函数对向量求导，其结果为Jacobian矩阵：

$$\frac{\partial \text{net}_t}{\partial \text{s}_{t-1}} = \begin{bmatrix} \frac{\partial net_1^t}{\partial s_1^{t-1}} & \frac{\partial net_1^t}{\partial s_2^{t-1}} & \cdots & \frac{\partial net_1^t}{\partial s_n^{t-1}} \\ \frac{\partial net_2^t}{\partial s_1^{t-1}} & \frac{\partial net_2^t}{\partial s_2^{t-1}} & \cdots & \frac{\partial net_2^t}{\partial s_n^{t-1}} \\ & \cdot & & \\ & \cdot & & \\ \frac{\partial net_n^t}{\partial s_1^{t-1}} & \frac{\partial net_n^t}{\partial s_2^{t-1}} & \cdots & \frac{\partial net_n^t}{\partial s_n^{t-1}} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ & \cdot & & \\ & \cdot & & \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} = W$$

$$\begin{bmatrix} u_{11} u_{12} \ldots u_{1m} \\ u_{21} u_{22} \ldots u_{2m} \\ \cdot \\ \cdot \\ u_{n1} u_{n2} \ldots u_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ x_m \end{bmatrix} + \begin{bmatrix} w_{11} w_{12} \ldots w_{1n} \\ w_{21} w_{22} \ldots w_{2n} \\ \cdot \\ \cdot \\ w_{n1} w_{n2} \ldots w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ \cdot \\ s_n^{t-1} \end{bmatrix}$$

同理，上式第二项也是一个Jacobian矩阵：

$$\frac{\partial \mathbf{s}_{t-1}}{\partial \mathbf{net}_{t-1}} = \begin{bmatrix} \frac{\partial s_1^{t-1}}{\partial net_1^{t-1}} & \frac{\partial s_1^{t-1}}{\partial net_2^{t-1}} & \dots & \frac{\partial s_1^{t-1}}{\partial net_n^{t-1}} \\ \frac{\partial s_2^{t-1}}{\partial net_1^{t-1}} & \frac{\partial s_2^{t-1}}{\partial net_2^{t-1}} & \dots & \frac{\partial s_2^{t-1}}{\partial net_n^{t-1}} \\ & \cdot & & \\ & \cdot & & \\ \frac{\partial s_n^{t-1}}{\partial net_1^{t-1}} & \frac{\partial s_n^{t-1}}{\partial net_2^{t-1}} & \dots & \frac{\partial s_n^{t-1}}{\partial net_n^{t-1}} \end{bmatrix}$$

$$= \begin{bmatrix} f'(net_1^{t-1}) & 0 & \dots & 0 \\ 0 & f'(net_2^{t-1}) & \dots & 0 \\ & \cdot & & \\ & \cdot & & \\ 0 & 0 & \dots & f'(net_n^{t-1}) \end{bmatrix}$$

$$= \mathrm{diag}[f'(\mathbf{net}_{t-1})]$$

其中，diag[a]表示根据向量a创建一个对角矩阵，即

$$\mathrm{diag}(\mathrm{a}) = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ & \cdot & & \\ 0 & 0 & \dots & a_n \end{bmatrix}$$

8.3 随时间反向传播（BPTT）

最后，将两项合在一起，可得：

$$
\begin{aligned}
\frac{\partial \mathrm{net}_t}{\partial \mathrm{net}_{t-1}} &= \frac{\partial \mathrm{net}_t}{\partial \mathrm{s}_{t-1}} \frac{\partial \mathrm{s}_{t-1}}{\partial \mathrm{net}_{t-1}} \\
&= W \,\mathrm{diag}\left[f'(\mathrm{net}_{t-1})\right]
\end{aligned}
$$

$$
= \begin{bmatrix}
w_{11} f'\left(net_1^{t-1}\right) & w_{12} f'\left(net_2^{t-1}\right) & \ldots & w_{1n} f'\left(net_n^{t-1}\right) \\
w_{21} f'\left(net_1^{t-1}\right) & w_{22} f'\left(net_2^{t-1}\right) & \ldots & w_{2n} f'\left(net_n^{t-1}\right) \\
 & . & & \\
 & . & & \\
w_{n1} f'\left(net_1^{t-1}\right) & w_{n2} f'\left(net_2^{t-1}\right) & \ldots & w_{nn} f'\left(net_n^{t-1}\right)
\end{bmatrix}
$$

8.3 随时间反向传播（BPTT）

上式描述了将δ沿时间往前传递一个时刻的规律，有了这个规律，就可以求得任意时刻k的误差项δ_k

$$\begin{aligned}
\delta_k^T &= \frac{\partial E}{\partial \mathrm{net}_k} \\
&= \frac{\partial E}{\partial \mathrm{net}_t}\frac{\partial \mathrm{net}_t}{\partial \mathrm{net}_k} \\
&= \frac{\partial E}{\partial \mathrm{net}_t}\frac{\partial \mathrm{net}_t}{\partial \mathrm{net}_{t-1}}\frac{\partial \mathrm{net}_{t-1}}{\partial \mathrm{net}_{t-2}}\cdots\frac{\partial \mathrm{net}_{k+1}}{\partial \mathrm{net}_k} \\
&= W\,\mathrm{diag}[f'(\mathrm{net}_{t-1})]W\,\mathrm{diag}[f'(\mathrm{net}_{t-2})]\ldots W\,\mathrm{diag}[f'(\mathrm{net}_k)]\,\delta_t^T \\
&= \delta_t^T\prod_{i=k}^{t-1} W\,\mathrm{diag}[f'(\mathrm{net}_i)]
\end{aligned} \qquad \text{(式3)}$$

（式3）就是将误差项**沿时间**反向传播的算法。

循环层将**误差项**反向传递到上一层网络，与普通的**全连接层完全一样**

循环层的**加权输入**net^l与上一层的**加权输入**net^{l-1}关系如下：

$$\mathrm{net}_t^l = U\mathrm{a}_t^{l-1} + W\mathrm{s}_{t-1}$$
$$\mathrm{a}_t^{l-1} = f^{l-1}(\mathrm{net}_t^{l-1})$$

上式中的net_t^l式第l层神经元的**加权输入**（假设第l层是**循环层**）； net_t^{l-1}是第$l-1$层神经元的**加权输入**；a_t^{l-1}是第$l-1$层神经元的输出；f^{l-1}是第$l-1$层的**激活函数**。

$$
\begin{aligned}
\frac{\partial \mathrm{net}_t^l}{\partial \mathrm{net}_t^{l-1}} &= \frac{\partial \mathrm{net}^l}{\partial \mathrm{a}_t^{l-1}} \frac{\partial \mathrm{a}_t^{l-1}}{\partial \mathrm{net}_t^{l-1}} \\
&= U \operatorname{diag}\left[f'^{l-1}\left(\mathrm{net}_t^{l-1}\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{net}_t^l &= U \mathrm{a}_t^{l-1} + W \mathrm{s}_{t-1} \\
\mathrm{a}_t^{l-1} &= f^{l-1}\left(\mathrm{net}_t^{l-1}\right)
\end{aligned}
$$

所以

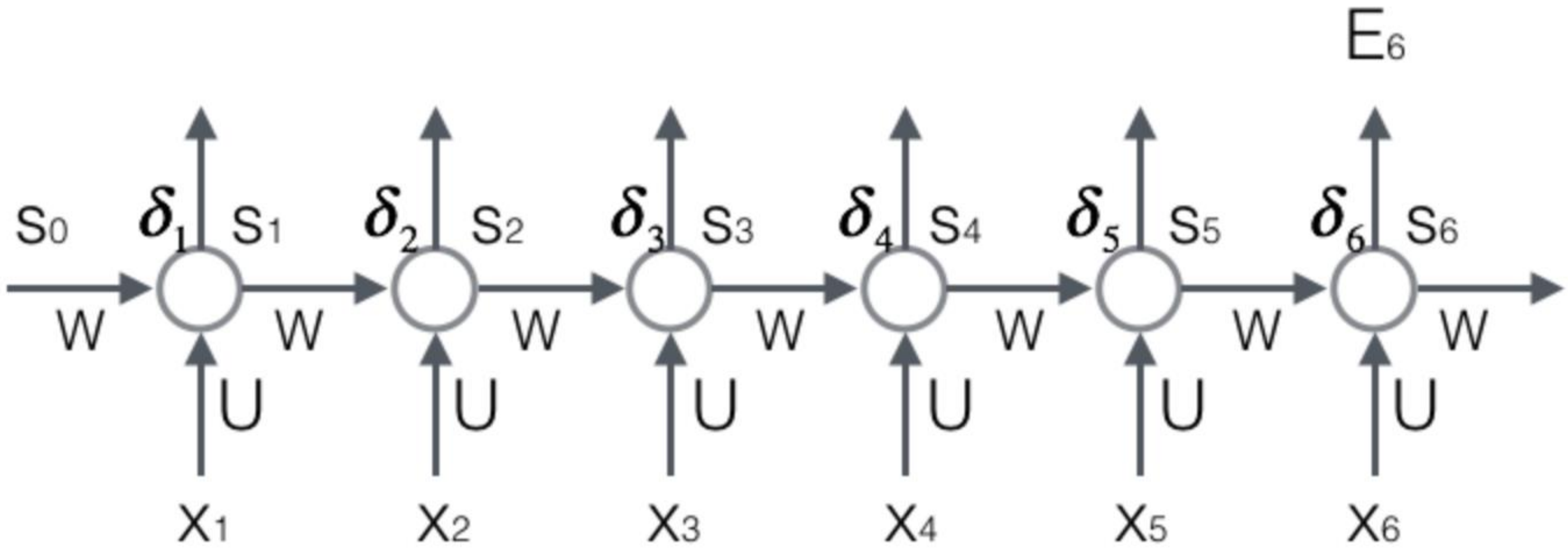
$$
\begin{aligned}
\left(\delta_t^{l-1}\right)^T &= \frac{\partial E}{\partial \mathrm{net}_t^{l-1}} \\
&= \frac{\partial E}{\partial \mathrm{net}_t^l} \frac{\partial \mathrm{net}_t^l}{\partial \mathrm{net}_t^{l-1}} \\
&= \left(\delta_t^l\right)^T U \operatorname{diag}\left[f'^{l-1}\left(\mathrm{net}_t^{l-1}\right)\right] \qquad \text{（式4）}
\end{aligned}
$$

（式4）就是将误差项传递到**上一层**算法。

C. 权重梯度的计算

BPTT算法的最后一步：计算每个权重的梯度。

首先，计算误差函数E对权重矩阵W的梯度 $\frac{\partial E}{\partial W}$



8.3 随时间反向传播（BPTT）

前两步中已经计算得到的量包括每个时刻t **循环层**的输出值s_t，以及误差项δ_t。

求得任意一个时刻的**误差项**δ_t，以及上一个时刻循环层的输出值s_{t-1}，就可以按照下面的公式求出权重矩阵在t时刻的梯度$\nabla_{W_t}E$：

$$\nabla_{W_t}E = \begin{bmatrix} \delta_1^t s_1^{t-1} & \delta_1^t s_2^{t-1} & \ldots & \delta_1^t s_n^{t-1} \\ \delta_2^t s_1^{t-1} & \delta_2^t s_2^{t-1} & \ldots & \delta_2^t s_n^{t-1} \\ . & & & \\ . & & & \\ \delta_n^t s_1^{t-1} & \delta_n^t s_2^{t-1} & \ldots & \delta_n^t s_n^{t-1} \end{bmatrix} \quad (式5)$$

8.3 随时间反向传播（BPTT）

下面可以简单推导一下**式5**。

已知：

$$\text{net}_t = Ux_t + Ws_{t-1}$$

$$\begin{bmatrix} net_1^t \\ net_2^t \\ . \\ . \\ net_n^t \end{bmatrix} = U\mathrm{x}_t + \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ . & & & \\ . & & & \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ . \\ . \\ s_n^{t-1} \end{bmatrix} = Ux_t + \begin{bmatrix} w_{11}s_1^{t-1} + w_{12}s_2^{t-1} \dots w_{1n}s_n^{t-1} \\ w_{21}s_1^{t-1} + w_{22}s_2^{t-1} \dots w_{2n}s_n^{t-1} \\ . \\ . \\ w_{n1}s_1^{t-1} + w_{n2}s_2^{t-1} \dots w_{nn}s_n^{t-1} \end{bmatrix}$$

8.3 随时间反向传播（BPTT）

$$\begin{bmatrix} net_1^t \\ net_2^t \\ . \\ . \\ net_n^t \end{bmatrix} = U\mathrm{x}_t + \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ . & & & \\ . & & & \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ . \\ . \\ s_n^{t-1} \end{bmatrix} = Ux_t + \begin{bmatrix} w_{11}s_1^{t-1} + w_{12}s_2^{t-1} \dots w_{1n}s_n^{t-1} \\ w_{21}s_1^{t-1} + w_{22}s_2^{t-1} \dots w_{2n}s_n^{t-1} \\ . \\ . \\ w_{n1}s_1^{t-1} + w_{n2}s_2^{t-1} \dots w_{nn}s_n^{t-1} \end{bmatrix}$$

因为对W求导与$U\mathrm{x}_t$无关，因此不再考虑。现在考虑对权重项w_{ji}求导。通过观察上式可以看到w_{ji}**只与**net_j^t**有关**，所以：

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial net_j^t} \frac{\partial net_j^t}{\partial w_{ji}} \\ &= \delta_j^t s_i^{t-1} \end{aligned}$$

按照上面的规律就可以生成**式5**里面的矩阵。

求得了权重矩阵W在t时刻的梯度$\nabla_{W_t}E$，**最终的梯度$\nabla_W E$是各个时刻的梯度之和：**

$$\nabla_W E = \sum_{i=1}^{t} \nabla_{W_i} E$$

$$= \begin{bmatrix} \delta_1^t s_1^{t-1} & \delta_1^t s_2^{t-1} & \dots & \delta_1^t s_n^{t-1} \\ \delta_2^t s_1^{t-1} & \delta_2^t s_2^{t-1} & \dots & \delta_2^t s_n^{t-1} \\ . \\ . \\ \delta_n^t s_1^{t-1} & \delta_n^t s_2^{t-1} & \dots & \delta_n^t s_n^{t-1} \end{bmatrix} + \dots + \begin{bmatrix} \delta_1^1 s_1^0 & \delta_1^1 s_2^0 & \dots & \delta_1^1 s_n^0 \\ \delta_2^1 s_1^0 & \delta_2^1 s_2^0 & \dots & \delta_2^1 s_n^0 \\ . \\ . \\ \delta_n^1 s_1^0 & \delta_n^1 s_2^0 & \dots & \delta_n^1 s_n^0 \end{bmatrix} \qquad (式6)$$

式6就是计算**循环层**权重矩阵W的梯度的公式。

8.3 随时间反向传播（BPTT）

$$\text{net}_t = Ux_t + Ws_{t-1}$$

$$\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1})$$

为什么最终的梯度是各个时刻的梯度**之和**呢?

还是从这个式子开始：

$$\text{net}_t = Ux_t + Wf(\text{ net }_{t-1})$$

因为 Ux_t 与W完全无关，可以把它看做常量。现在，考虑第一个式子加号右边的部分，因为W和$f(net_{t-1})$都是W的函数，因此要用到导数乘法运算：

$$(uv)' = u'v + uv'$$

因此，上面第一个式子写成：

$$\frac{\partial \text{net}_t}{\partial W} = \frac{\partial W}{\partial W} f(\text{net}_{t-1}) + W \frac{\partial f(\text{net}_{t-1})}{\partial W}$$

8.3 随时间反向传播（BPTT）

最终需要计算的是$\nabla_W E$：

$$
\begin{aligned}
\nabla_W E &= \frac{\partial E}{\partial W} \\
&= \frac{\partial E}{\partial \mathrm{net}_t} \frac{\partial \mathrm{net}_t}{\partial W} \\
&= \delta_t^T \frac{\partial W}{\partial W} f(\mathrm{net}_{t-1}) + \delta_t^T W \frac{\partial f(\mathrm{net}_{t-1})}{\partial W}
\end{aligned}
\qquad \text{(式7)}
$$

先计算**式7**加号左边的部分。$\frac{\partial W}{\partial W}$是**矩阵对矩阵求导**，其结果是一个四维**张量(tensor)**，如下所示：

$$\frac{\partial W}{\partial W}=\begin{bmatrix}\frac{\partial w_{11}}{\partial W} & \frac{\partial w_{12}}{\partial W} & \dots & \frac{\partial w_{1n}}{\partial W}\\ \frac{\partial w_{21}}{\partial W} & \frac{\partial w_{22}}{\partial W} & \dots & \frac{\partial w_{2n}}{\partial W}\\ . & & & \\ . & & & \\ \frac{\partial w_{n1}}{\partial W} & \frac{\partial w_{n2}}{\partial W} & \dots & \frac{\partial w_{nn}}{\partial W}\end{bmatrix}=\begin{bmatrix}\begin{bmatrix}\boxed{\frac{\partial w_{11}}{\partial w_{11}}} & \frac{\partial w_{11}}{\partial w_{12}} & \dots & \frac{\partial w_{11}}{\partial_{1n}}\\ \frac{\partial w_{11}}{\partial w_{21}} & \frac{\partial w_{11}}{\partial w_{22}} & \dots & \frac{\partial w_{11}}{\partial_{2n}}\\ . & & & \\ . & & & \\ \frac{\partial w_{11}}{\partial w_{n1}} & \frac{\partial w_{11}}{\partial w_{n2}} & \dots & \frac{\partial w_{11}}{\partial_{nn}}\end{bmatrix} & \begin{bmatrix}\frac{\partial w_{12}}{\partial w_{11}} & \boxed{\frac{\partial w_{12}}{\partial w_{12}}} & \dots & \frac{\partial w_{12}}{\partial_{1n}}\\ \frac{\partial w_{12}}{\partial w_{21}} & \frac{\partial w_{12}}{\partial w_{22}} & \dots & \frac{\partial w_{12}}{\partial_{2n}}\\ . & & & \\ . & & & \\ \frac{\partial w_{12}}{\partial w_{n1}} & \frac{\partial w_{12}}{\partial w_{n2}} & \dots & \frac{\partial w_{12}}{\partial_{nn}}\end{bmatrix} & \dots\\ \vdots & & \end{bmatrix}$$

$$=\begin{bmatrix}\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 0 & \dots & 0\\ \vdots & & & \\ 0 & 0 & \dots & 0\end{bmatrix} & \begin{bmatrix}0 & 1 & \dots & 0\\ 0 & 0 & \dots & 0\\ \vdots & & & \\ 0 & 0 & \dots & 0\end{bmatrix} & \dots\\ \vdots & & \end{bmatrix}$$

接下来，因为$s_{t-1}=f(net_{t-1})$，它是一个**列向量**。让上面的四维张量与这个向量相乘，得到了一个三维张量，再左乘行向量δ_t^T，最终得到一个矩阵：

$$\delta_t^T\frac{\partial W}{\partial W}f(\mathrm{net}_{t-1})=\delta_t^T\frac{\partial W}{\partial W}s_{t-1}=\delta_t^T\begin{bmatrix}\begin{bmatrix}1 & 0 & \dots & 0\\0 & 0 & \dots & 0\\\vdots & & & \\0 & 0 & \dots & 0\end{bmatrix} & \begin{bmatrix}0 & 1 & \dots & 0\\0 & 0 & \dots & 0\\\vdots & & & \\0 & 0 & \dots & 0\end{bmatrix} & \dots\\ \vdots & & \end{bmatrix}\begin{bmatrix}s_1^{t-1}\\s_2^{t-1}\\.\\.\\s_n^{t-1}\end{bmatrix}$$

$$=\delta_t^T\begin{bmatrix}\begin{bmatrix}s_1^{t-1}\\0\\.\\.\\0\end{bmatrix} & \begin{bmatrix}s_2^{t-1}\\0\\.\\.\\0\end{bmatrix} & \dots\\ \vdots & & \end{bmatrix}=\begin{bmatrix}\delta_1^t & \delta_2^t & \dots & \delta_n^t\end{bmatrix}\begin{bmatrix}\begin{bmatrix}s_1^{t-1}\\0\\.\\.\\0\end{bmatrix} & \begin{bmatrix}s_2^{t-1}\\0\\.\\.\\0\end{bmatrix} & \dots\\ \vdots & & \end{bmatrix}=\begin{bmatrix}\delta_1^ts_1^{t-1} & \delta_1^ts_2^{t-1} & \dots & \delta_1^ts_n^{t-1}\\\delta_2^ts_1^{t-1} & \delta_2^ts_2^{t-1} & \dots & \delta_2^ts_n^{t-1}\\.\\.\\\delta_n^ts_1^{t-1} & \delta_n^ts_2^{t-1} & \dots & \delta_n^ts_n^{t-1}\end{bmatrix}$$

$$=\nabla_{W_t}E$$

接下来计算**式7**加号右边的部分：

$$
\begin{aligned}
\nabla_W E &= \frac{\partial E}{\partial W} \\
&= \frac{\partial E}{\partial \mathrm{net}_t} \frac{\partial \mathrm{net}_t}{\partial W} \\
&= \delta_t^T \frac{\partial W}{\partial W} f(\mathrm{net}_{t-1}) + \delta_t^T W \frac{\partial f(\mathrm{net}_{t-1})}{\partial W}
\end{aligned}
$$

$$
\begin{aligned}
\delta_t^T W \frac{\partial f(\mathrm{net}_{t-1})}{\partial W} &= \delta_t^T W \frac{\partial f(\mathrm{net}_{t-1})}{\partial \mathrm{net}_{t-1}} \frac{\partial \mathrm{net}_{t-1}}{\partial W} \\
&= \boxed{\delta_t^T W f'(\mathrm{net}_{t-1})} \frac{\partial \mathrm{net}_{t-1}}{\partial W} \\
&= \delta_{t-1}^T \frac{\partial \mathrm{net}_{t-1}}{\partial W}
\end{aligned}
$$

$$
\boxed{\delta_k^T = \delta_t^T \prod_{i=k}^{t-1} W \, \mathrm{diag}\left[f'(\mathrm{net}_i)\right]}
$$

8.3 随时间反向传播（BPTT）

于是，得到了如下递推公式：

$$
\begin{aligned}
\nabla_W E &= \frac{\partial E}{\partial W} \\
&= \frac{\partial E}{\partial \mathrm{net}_t}\frac{\partial \mathrm{net}_t}{\partial W} \\
&= \nabla_{Wt}E + \delta_{t-1}^{T}\frac{\partial \mathrm{net}_{t-1}}{\partial W} \\
&= \nabla_{Wt}E + \nabla_{Wt-1}E + \delta_{t-2}^{T}\frac{\partial \mathrm{net}_{t-2}}{\partial W} \\
&= \nabla_{Wt}E + \nabla_{Wt-1}E + \ldots + \nabla_{W1}E \\
&= \sum_{k=1}^{t}\nabla_{Wk}E
\end{aligned}
$$

这样就证明了：最终的梯度$\nabla_W E$是各个时刻的梯度之和。

8.3 随时间反向传播（BPTT）

同权重矩阵W类似，可以得到权重矩阵U的计算方法：

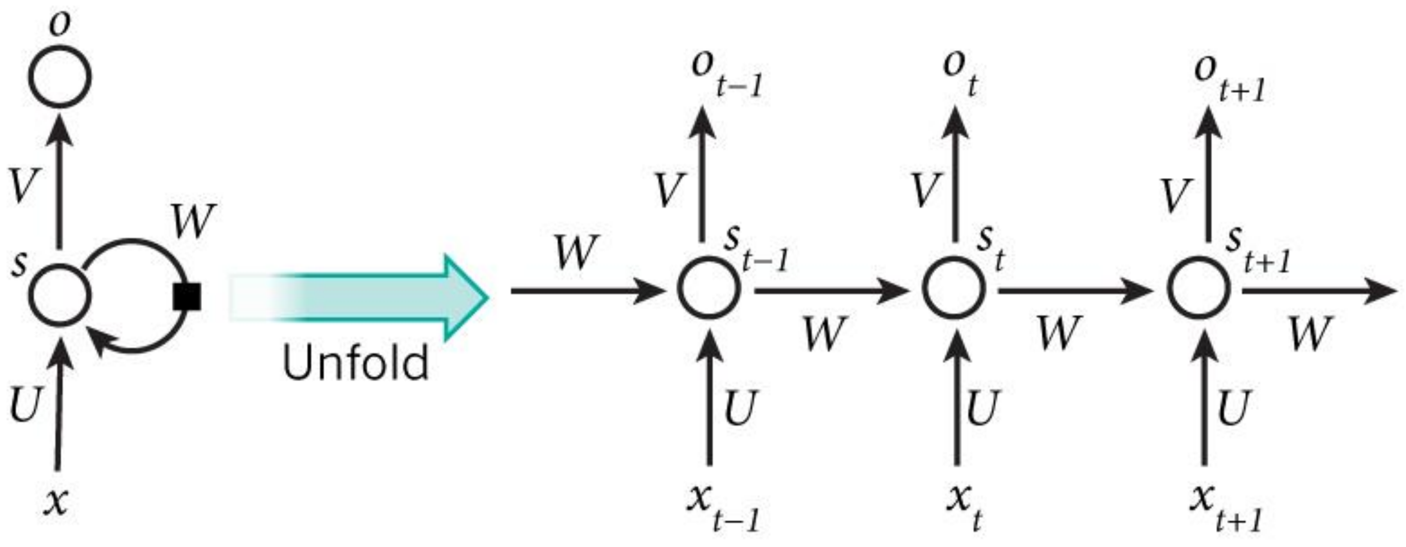
$$\nabla_{U_t}E=\begin{bmatrix}\delta_1^t x_1^t & \delta_1^t x_2^t & \ldots & \delta_1^t x_m^t \\ \delta_2^t x_1^t & \delta_2^t x_2^t & \ldots & \delta_2^t x_m^t \\ . & & & \\ . & & & \\ \delta_n^t x_1^t & \delta_n^t x_2^t & \ldots & \delta_n^t x_m^t\end{bmatrix} \qquad \text{（式8）}$$

式8是误差函数在t时刻对权重矩阵U的梯度。和权重矩阵W一样，最终的梯度也是各个时刻的梯度之和：

$$\nabla_U E=\sum_{i=1}^{t}\nabla_{U_i}E$$

权重矩阵V的计算

参数V相对简单，只需关注目前的状态。另外，RNN的损失也是会随着时间累加的，所以不能只求t时刻的偏导。



$$\frac{\partial E^{(t)}}{\partial V} = \frac{\partial E^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V}$$

$$E = \sum_{t=1}^{n} E^{(t)}$$

$$\frac{\partial E}{\partial V} = \sum_{t=1}^{n} \frac{\partial E^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V}$$

目录

8.4 双向循环神经网络

对于语言模型来说，很多时候光看前面的词是不够的，比如下面这句话：

我的手机坏了，我打算____一部新手机。

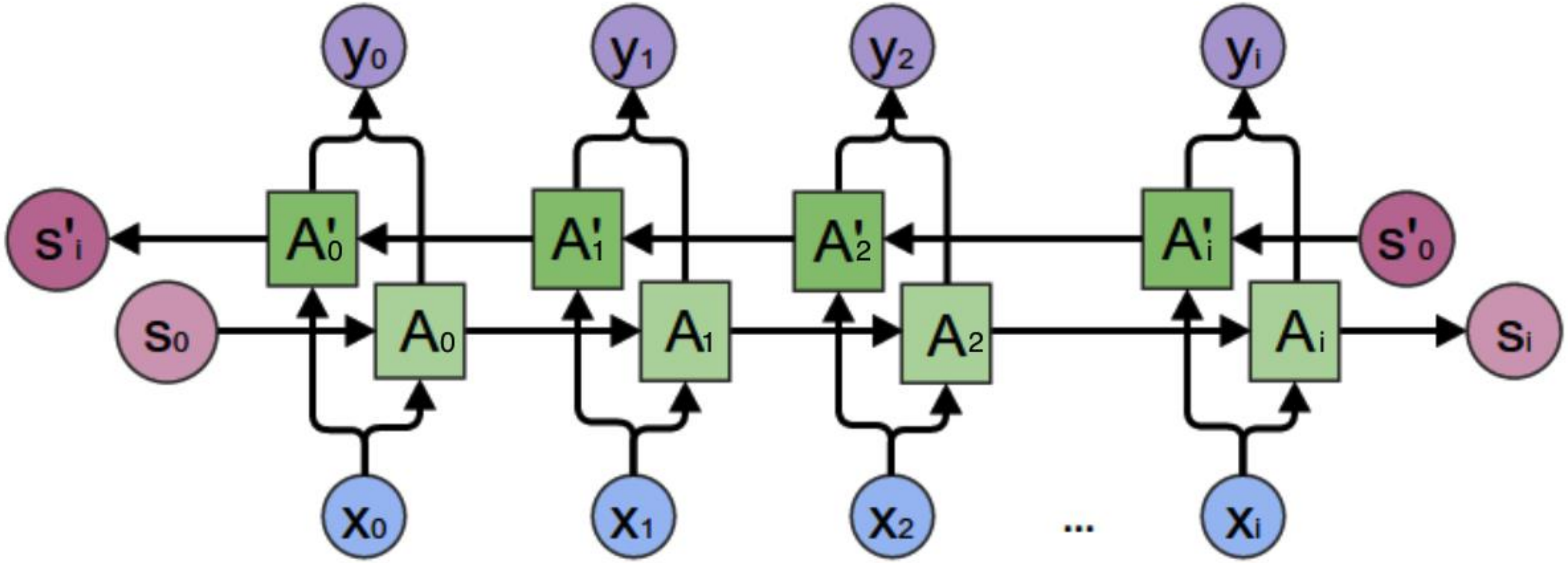
如果我们只看横线前面的词，手机坏了，那么我是打算修一修？换一部新的？还是大哭一场？这些都是无法确定的。

但如果我们也看到了横线后面的词是『一部新手机』，那么，横线上的词填『买』的概率就大得多了。

在此任务中，可以增加一个按照时间的逆序来传递信息的网络层，增强网络的能力。

8.4 双向循环神经网络

基本循环神经网络无法对此进行建模，因此，需要双向循环神经网络，如下图所示：



双向循环神经网络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。

8.4 双向循环神经网络

双向循环神经网络的隐藏层要保存两个值，一个值A参与正向计算，另一个值A'参与反向计算。最终的输出值取决于A和A'。其计算方法为（以y_2为例）：

$$\mathbf{y}_2 = g\left(VA_2 + V'A_2'\right)$$

A_2和A_2' 则分别计算：

$$A_2 = f(WA_1 + Ux_2)$$
$$A_2' = f(W'A_3' + U'x_2)$$

正向计算时，隐藏层的值s_t与s_{t-1}有关；
反向计算时，隐藏层的值s_t'与s_{t+1}'有关；
最终的输出取决于正向和反向计算的**加和。**

$$
\begin{aligned}
\mathrm{o}_t &= g(V\mathrm{s}_t + V'\mathrm{s}_t') \\
\mathrm{s}_t &= f(U\mathrm{x}_t + W\mathrm{s}_{t-1}) \\
\mathrm{s}_t' &= f(U'\mathrm{x}_t + W'\mathrm{s}_{t+1}')
\end{aligned}
$$

从上面三个公式得到，正向计算和反向计算**不共享权重**，也就是说U和U'、W和W'、V和V'都是不同的**权重矩阵。**

主要参考文献

1. https://zybuluo.com/hanbingtao/note/541458
2. Recurrent Neural Networks Tutorial http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
3. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks[C] //International conference on machine learning. 2013: 1310-1318.
4. Attention and Augmented Recurrent Neural Networks. https://distill.pub/2016/augmented-rnns/
5. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/
6. Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions [C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3128-3138.