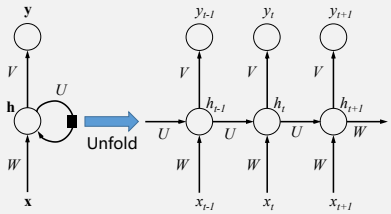
人工智能原理

第五章　循环神经网络

1．循环神经网络的模型结构

循环神经网络（Recurrent Neural Network，RNN）是一种对序列数据建模的神经网络，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不在无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。RNN模型的连接如图所示。



RNN模型结构图

2．循环神经网络的基本工作原理

循环神经网络的工程原理或是工作过程其实就是循环神经网络的训练算法，一种基于时间的反向传播算法BPTT(Bach Propagation Through Time)。BPTT算法是针对循环层设计的训练算法，它的基本原理和反向传播BP（Back Propagation）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值，它是误差函数E对神经元j的加权输入的偏导数。
3. 计算每个权重的梯度。

2．循环神经网络的基本工作原理

最后再用随机梯度下降算法更新权重。

假设时刻为t时，输入为x_t，隐层状态（隐层神经元活性值）为h_t，h_t不仅和当前时刻的输入x_t相关，也和上一个时刻的隐层状态h_{t-1}相关。

$$\mathbf{z}_t = \mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b} \tag{8-1}$$

$$\mathbf{h}_t = f(\mathbf{z}_t) \tag{8-2}$$

其中$\mathbf{z}_t$为隐藏层的净输入；$f(\cdot)$是非线性激活函数，通常为logistic函数或tanh函数；u为状态-状态权重矩阵；w为状态-输入权重矩阵；b为偏置。式（8-1）和式（8-2）也经常直接写为:

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}) \tag{8-3}$$

2．循环神经网络的基本工作原理

循环神经网络要求每一个时刻都有一个输入，但是不一定每一个时刻都需要有输出。其次循环神经网络可以往前看获得任意多个输入值，其递归推导方法如式（8-4）所示，即RNN的输出层$\mathbf{y}$和隐藏层h的计算方法：

$$\mathbf{y}_t = g(\mathbf{V}\mathbf{h}_t) \tag{8-4}$$

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}) \tag{8-5}$$

如果反复把式（8-5）带入到式（8-4），将得到递归如下，推导可以看出，RNN的输出层$\mathbf{y}$和输入系列x_t的前t个时刻都有关。

$$\begin{aligned}\mathbf{y}_t &= g(\mathbf{V}\mathbf{h}_t)\\ &= g(\mathbf{V}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_t))\\ &= g(\mathbf{V}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-1} + \mathbf{U}\mathbf{h}_{t-2} + \mathbf{b}_{t-1}) + \mathbf{b}_t))\\ &= g(\mathbf{V}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-1} + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-2} + \mathbf{U}\mathbf{h}_{t-3} + \mathbf{b}_{t-2}) + \mathbf{b}_{t-1}) + \mathbf{b}_t))\\ &= g(\mathbf{V}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-1} + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-2} + \mathbf{U}f(\mathbf{W}\mathbf{x}_{t-3} + \cdots) + \mathbf{b}_{t-2}) + \mathbf{b}_{t-1}) + \mathbf{b}_t))\end{aligned}$$

3. 循环神经网络的前向计算

循环神经网络中循环的意思就是同一网络结构不停的重复。相比普通的神经网络，循环神经网络的不同之处在于，隐层的神经元之间还有相互的连接，在隐层上增加了一个反馈连接，也就是说，RNN隐层当前时刻的输入有一部分是前一时刻隐层的输出，这使得RNN可以通过循环反馈连接保留前面所有时刻的信息，这赋予了RNN的记忆功能。这些特点使得RNN非常适合用于对时序信号的建模。

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}) \tag{8-6}$$

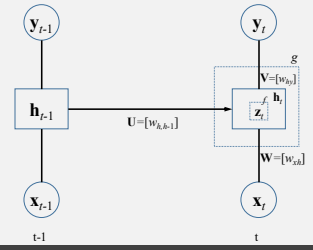
$$\mathbf{y}_t = g(\mathbf{V}\mathbf{h}_t) \tag{8-7}$$

整理一下可以写为：

$$\mathbf{y}_t = g(\mathbf{V}f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1})) \tag{8-8}$$

3. 循环神经网络的前向计算

循环神经网络正向计算如图所示。



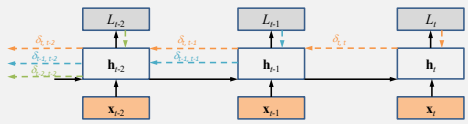
前向计算示意图

3. 循环神经网络的前向计算

给定计算t时刻的输入x_t求网络的输出y_t。输入x_t与权w_{x_k} 相乘（加上偏差b）与前一时刻的隐层输出与权重$h_{t-1}U_{h_k h_{t-1}}$的和为Z_t,即 $Z_t = U_{h_k h_{t-1}} h_{t-1} + W_{x_k} x_t + b$，且$Z_t$为 N × 1隐含层潜向量， w_{x_k}是N × K权重矩阵连接K个输入单元到N个隐含层单元；Z_t经过激活函数$f(\cdot)$之后即为隐藏层的输出h_t，$V_t = h_t V_{h_y}$，V_t是L × 1输出层潜向量， V_t经过激活函数$g(\cdot)$以后即得到输出y_t， $f(Z_t)$是隐含层激活函数， $g(V_t)$是输出层激活函数。典型的隐含层激活函数有sigmoid ,tanh 与 rectified linear units， （以去掉黄色阴影）典型的输出层所用的激活函数有linear和SoftMax函数。激活函数的主要作用是提供网络的非线性建模能力。如果没有激活函数，那么该网络仅能够表达线性映射，此时即便有再多的隐藏层，其整个网络跟单层神经网络也是等价的。因此也可以认为，只有加入了激活函数之后，深度神经网络才具备了分层的非线性映射学习能力。当然激活函数应具有的基本特性有：（1）可微性：当优化方法是基于梯度的时候，这个性质是必须的。（2）单调性：当激活函数是单调的时候，单层网络能够保证是凸函数。（3）输出值的范围：当激活函数输出值是有限的时候，基于梯度的优化方法会更加稳定，因为特征的表示受有限权值的影响更显著；当激活函数的输出是无限的时候，模型的训练会更加高效，不过在这种情况下，一般需要更小的学习率。

4. 循环神经网络的梯度计算

BPTT算法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”。这样，循环神经网络就可以按照前馈网络中的反向传播算法进行参数梯度计算。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所有“展开层”的参数梯度之和，其误差反向传播示意图如图所示。



误差反向传播示意图

4. 循环神经网络的梯度计算

给定一个训练样本(x, y)，其中$x = (x_1, x_2, \cdots, x_T)$为长度是T的输入序列，y=$(y_1, y_2, \cdots, y_T)$是长度为T的标签序列。即在每个时刻t，都有一个监督信息y_t，定义时刻t的损失函数为

$$\mathbf{L}_t = L(\mathbf{y}_t, g(\mathbf{h}_t)) \tag{8-9}$$

式（8-9）中， $g(h_t)$为第t时刻的输出；L为可微分的损失函数，比如交叉熵。那么整个序列上损失函数为

$$\mathbf{L} = \sum_{t=1}^{T} L_t \tag{8-10}$$

整个序列的损失函数 关于隐层向参数U的梯度为

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial U} \tag{8-11}$$

即每个时刻损失L_t对参数U的偏导数之和。

4. 循环神经网络的梯度计算

计算偏导数$\frac{\partial L_t}{\partial U}$先来计算公式(8-11)中第$t$时刻损失对参数$U$的偏导数$\frac{\partial L_t}{\partial U}$。

因为参数U和隐藏层在每个时刻$k(1 \le k \le t)$的净输入$Z_k = Uh_{k-1} + Wx_k + b$有关，因此第$t$时刻损失的损失函数$L_t$关于参数$U_{ij}$的梯度为：

$$\frac{\partial L_t}{\partial U_{ij}} = \sum_{k=1}^{t} tr\left(\left(\frac{\partial L_t}{\partial \mathbf{z}_k}\right)^T \frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}}\right) = \sum_{k=1}^{t} \left(\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}}\right)^T \frac{\partial L_t}{\partial \mathbf{z}_k} \tag{8-12}$$

式（8-12）中，$\frac{\partial^+ z_k}{\partial u_{ij}}$表示“直接”偏导数，即公式$Z_k = Uh_{k-1} + Wx_k + b$中保持$h_{k-1}$不变,对$U_{ij}$进行求偏导数，得到

$$\frac{\partial^+ \mathbf{z}_k}{\partial U_{ij}} = \begin{bmatrix} 0 \\ \vdots \\ [h_{k-1}]_j \\ \vdots \\ 0 \end{bmatrix} = \prod_i ([\mathbf{h}_{k-1}])_j \tag{8-13}$$

5.1 循环神经网络的工作原理

4. 循环神经网络的梯度计算

式（8-13）中，$[h_{k-1}]_j$为第$k-1$时刻隐层状态的第j维；$\prod_i(x)$除了第i行值为x外，其余都为0的向量。

定义$\delta_{t,k}=\frac{\partial L_t}{\partial z_k}$为第$t$时刻的损失对第$k$时刻隐藏神经层的净输入$z_k$的导数，则

$$\begin{aligned}\delta_{t,k}&=\frac{\partial L_t}{\partial \mathbf{z}_k}\\&=\frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k}\frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k}\frac{\partial L_t}{\partial \mathbf{z}_{k+1}}\\&=diag(f'(\mathbf{z}_k))U^T\delta_{t,k+1}\end{aligned} \tag{8-14}$$

将式(8-14)和(8-13)代入公式(8-12)得到

$$\frac{\partial L_t}{\partial U_{i,j}}=\sum_{k=1}^{t}[\delta_{t,k}]_i[\mathbf{h}_{k-1}]_j \tag{8-15}$$

将式(8-15)写成矩阵形式为

$$\frac{\partial L_t}{\partial U}=\sum_{k=1}^{t}\delta_{t,k}\mathbf{h}_{k-1}^T \tag{8-16}$$

5.1 循环神经网络的工作原理

4. 循环神经网络的梯度计算

将式(8-16)代入到将式(8-11)得到整个序列的损失函数L关于参数U的梯度：

$$\frac{\partial L}{\partial U}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\mathbf{h}_{k-1}^T \tag{8-17}$$

同理可得，L关于权重W、偏置b以及参数V的梯度为：

$$\frac{\partial L}{\partial W}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\mathbf{x}_k^T \tag{8-18}$$

$$\frac{\partial L}{\partial b}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k} \tag{8-19}$$

$$\frac{\partial L}{\partial V}=\sum_{t=1}^{T}\sum_{k=1}^{t}\delta_{t,k}\mathbf{h}_{k-1}^T \tag{8-20}$$

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。

第五章　循环神经网络

5.2 改进的循环神经网络

1. 梯度爆炸与梯度消失

循环神经网络在学习过程中的主要问题是长期依赖问题。

在BPTT算法中，将公式(8-14)展开得到

$$\delta_{t,k}=\prod_{i=k}^{t-1}(diag(f'(z_i))U^T)\delta_{t,t} \tag{8-21}$$

如果定义$\gamma\cong\|diag(f'(z_i))U^T\|$，则

$$\delta_{t,k}=\gamma^{t-k}\delta_{t,t} \tag{8-22}$$

若$\gamma>1$，当$t-k\to\infty$时，$\gamma^{t-k}\to\infty$，会造成系统不稳定，此时称为梯度爆炸问题（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)。

值得注意的是，在循环神经网络中的梯度消失不是说$\frac{\partial L_t}{\partial U}$的梯度消失了，而是$\frac{\partial L_t}{\partial h_k}$的梯度消失了（当$t-k$比较大时）。也就是说，参数$U$的更新主要靠当前时刻$k$的几个相邻状态$h_k$来更新，长距离的状态对$U$没有影响。

5.2 改进的循环神经网络

1. 梯度爆炸与梯度消失

为了避免梯度爆炸或消失问题，一种最直接的方式就是选取合适的参数，同时使用非饱和的激活函数，尽量使得$diag(f'(z_i))U^T\approx 1$，这种方式需要足够的人工调参经验，限制了模型的广泛应用。采用比较有效的方式改进模型或优化方法来缓解循环神经网络的梯度爆炸和梯度消失问题。

梯度爆炸	梯度消失
一般而言，循环网络的梯度爆炸问题比较容易解决，主要通过权重衰减或梯度截断来避免。	梯度消失是循环神经网络的主要问题。除了使用一些优化技巧外，更有效的方式就是改变模型。

5.2 改进的循环神经网络

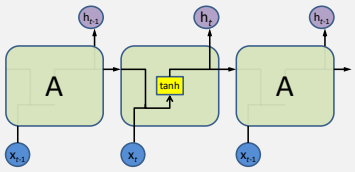
2. 长短时记忆神经网络

Long Short-Term Memory Neural Network一般就叫做LSTM，是一种RNN特殊的类型，可以学习长期依赖信息。LSTM 由Hochreiter & Schmidhuber (1997)提出，并在近期被Alex Graves进行了改良和推广。在很多问题，LSTM 都取得相当巨大的成功，并得到了广泛的使用。LSTM 通过刻意的设计来避免长期依赖问题。记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力，所有RNN都具有一种重复神经网络模块的链式的形式。LSTM能避免RNN的梯度消失问题，其使用“累加”的形式计算状态，这种累加形式导致导数也是累加形式，因此避免了梯度消失。

2. 长短时记忆神经网络

(1) LSTM的结构

所有循环神经网络都有一个重复结构的模型形式，在标准的RNN中，重复的结构是一个简单的循环体，如图所示的A循环体。

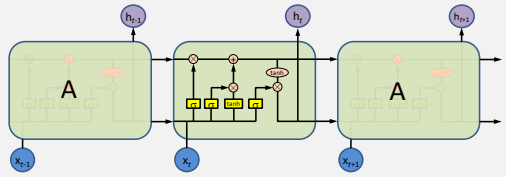


循环神经网络重复结构图

2. 长短时记忆神经网络

(1) LSTM的结构

LSTM的循环体是一个拥有四个相互关联的全连接前馈神经网络的复制结构，如图所示。

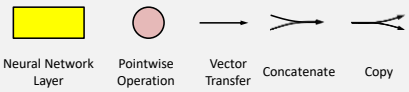


LSTM结构图

2. 长短时记忆神经网络

(1) LSTM的结构

LSTM结构图中具体的符号语义如图所示。其中英文对应的意思是：Neural Network Layer: 该图表示一个神经网络层；Pointwise Operation: 该图表示一种操作；Vector Transfer: 每一条线表示一个向量，从一个节点输出到另一个节点；Concatenate: 该图表示两个向量的合并，即由两个向量合并为一个向量；Copy: 该图表示一个向量复制了两个向量，其中两个向量值相同。



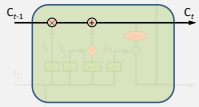
LSTM符号语义图

2. 长短时记忆神经网络

(2) LSTM结构分析

1）核心设计

LSTM设计的关键是神经元的状态，即为图所示顶部的水平线。神经元的状态类似传送带一样，按照传送方向从左端被传送到右端，在传送过程中基本不会改变，只是进行一些简单的线性运算：加或减操作。神经元间通过线性操作能够小心地管理神经元的状态信息，将这种管理方式称为门操作(gate)。门操作能够随意地控制神经元状态信息的流动，如图所示，它由一个sigmoid激活函数的神经网络层和一个点乘运算组成。LSTM有三个门来管理和控制神经元的状态信息。



LSTM的C线

LSTM的基本控制门

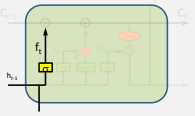
2. 长短时记忆神经网络

(2) LSTM结构分析

2）遗忘门

LSTM的第一步是决定要从上一个时刻的状态中丢弃什么信息，其是由一个sigmoid全连接的前馈神经网络的输出来管理，将这种操作称为遗忘门（forget gate layer），如图8-9所示。这个全连接的前馈神经网络的输入是h_{t-1}和x_t组成的向量，输出是f_t向量。f_t向量是由1和0组成，1表示能够通过，0表示不能通过。其函数式为：

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{8-25}$$



LSTM的遗忘门图

2. 长短时记忆神经网络

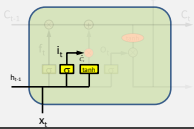
(2) LSTM结构分析

3）输入门

第二步决定哪些输入信息要保存到神经元的状态中。这由两队前馈神经网络决定，如图8-10所示。首先是一个有sigmoid层的全连接前馈神经网络，称为输入门（input gate layer），其决定了哪些值将被更新；然后是一个tanh层的全连接前馈神经网络，其输出是一个向量 C_t，C_t 向量可以被添加到当前时刻的神经元状态中；最后根据两个神经网络的结果创建一个新的神经元状态。其函数关系为：

$$C_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{8-26}$$

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{8-27}$$



LSTM的输入门

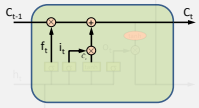
2. 长短时记忆神经网络

(2) LSTM结构分析

4）状态控制

第三步就可以更新上一时刻的状态C_{t-1}为当前时刻的状态C_t了。上述的第一步的遗忘门计算了一个控制向量，此时可通过这个向量过滤了一部分C_{t-1}信息，如图8-11所示的乘法操作；上述第二步的输入门根据输入向量计算了新状态，此时可以通过这个新状态和C_{t-1}状态构建一个新的状态 C_t，如图8-11所示的加法操作。其函数关系为：

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (8\text{-}28)$$



LSTM状态控制图

2. 长短时记忆神经网络

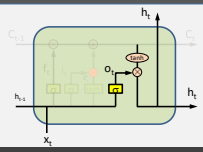
(2) LSTM结构分析

5）输出门

最后一步就是决定神经元的输出向量h_t是什么，此时的输出是根据上述第三步的C_t状态进行计算的，即根据一个sigmoid层的全连接前馈神经网络过滤掉一部分C_t状态作为当前时刻神经元的输出，如图8-12所示。其函数关系为：

$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0) \quad (8\text{-}29)$$

$$h_t = o_t * \tanh(C_t) \quad (8\text{-}30)$$



LSTM的输出门

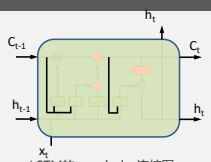
2. 长短时记忆神经网络

(3) LSTM的延伸网络

上述介绍的LSTM结构是一个正常的网络结构，然而并不是所有的LSTM网络都是这种结构，实际上，LSTM有很多种变体，即为多种变化形态。如下介绍几种常用形态结构：

1）Peephole Connections

一种流行的LSTM变体是由Gers&Schmidhuber（2000）提出的网络结构，如图所示。



LSTM的peepholes连接图

2. 长短时记忆神经网络

(3) LSTM的延伸网络

1）Peephole Connections

通过将上一时刻的状态Ct-1合并到各个门上，从而更详细控制各个门的管理。其具体的各层函数关系式为：

$$f_t = \sigma(W_f[C_{t-1}, h_{t-1}, x_t] + b_f) \quad (8\text{-}31)$$

$$i_t = \sigma(W_i[C_{t-1}, h_{t-1}, x_t] + b_i) \quad (8\text{-}32)$$

$$o_t = \sigma(W_o[C_t, h_{t-1}, x_t] + b_o) \quad (8\text{-}33)$$

2. 长短时记忆神经网络

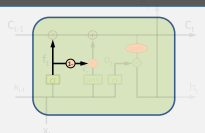
(3) LSTM的延伸网络

2) Coupled Forget and Input Gates

另一种变体是使用耦合的遗忘门和输入门，如图所示。

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，将这两个门合并为一个门。其具体的函数关系为：

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t \quad (8\text{-}34)$$



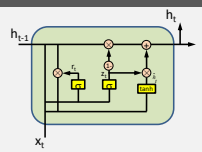
LSTM变体形式图

2. 长短时记忆神经网络

(3) LSTM的延伸网络

3) Gated Recurrent Unit

门限循环单元（Gated Recurrent Unit, GRU）是一种比LSTM更加简化的版本，是LSTM的一种变体，如图8-15所示。在LSTM中，输入门和遗忘门是互补关系，因为同时用两个门比较冗余。GRU将输入门与遗忘门合并成一个门：更新门（Update Gate），同时还合并了记忆单元和神经元的活性值。



GRU模型结构图

2. 长短时记忆神经网络

(3) LSTM的延伸网络

3) Gated Recurrent Unit

GRU模型中有两个门：更新门z和重置门r，更新门z用来控制当前的状态需要遗忘多少历史信息和接受多少新信息。重置门r用来控制候选状态中有多少信息是从历史信息中得到。

GRU模型的更新关系式为：

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{8-35}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{8-36}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{8-37}$$

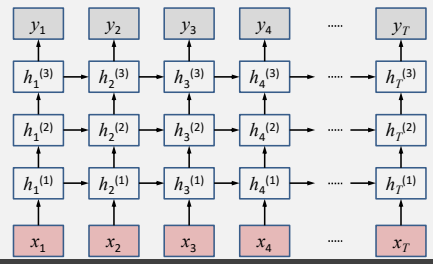
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{8-38}$$

如果将深度定义为网络中信息传递路径长度的话，循环神经网络可以看作是既“深”又“浅”的网络。一方面来说，如果把循环网络按时间展开，长时间间隔的状态之间的路径很长，循环网络可以看作是一个非常深的网络了。从另一方面来说，如果同一时刻网络输入到输出之间的路径为$x_t \to y_t$，那么这个网络是非常浅的。

既然增加深度可以极大地增强前馈神经网络的处理能力，那么如何增加循环神经网络的深度呢？

增加循环神经网络的深度主要是增加同一时刻网络输入到输出之间的路径$x_t \to y_t$，比如增加隐藏状态到输出$h_t \to y_t$，以及输入到隐藏状态$x_t \to h_t$之间的路径的深度。

一种常见的做法是将多个循环网络堆叠起来，称为堆叠循环神经网络（Stacked Recurrent Neural Network，SRNN）。一个堆叠的简单循环神经网络也称为循环网络多层感知器（Recurrent Multi-layer Perception，RMLP）。下图给出了按时间展开的堆叠循环神经网络。



按时间展开的堆叠循环神经网络

$$\mathbf{h}_t^{(l)} = f(U^{(l)}\mathbf{h}_{t-1}^{(l)} + W^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{b}^{(l)}) \tag{8-39}$$

式（8-39）中，$U^{(l)}$，$W^{(l)}$和$b^{(l)}$为权重矩阵和偏置向量，当$l = 1$时，$h_t^{(0)} = x_t$。

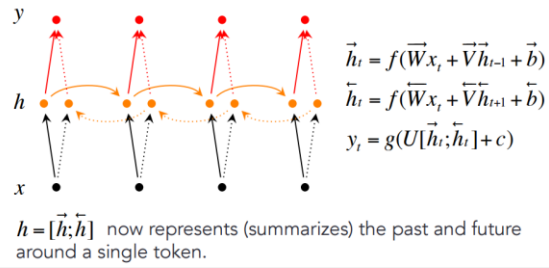
5.4 双向循环神经网络

从单向的循环神经网络结构中可以知道它的下一刻预测输出是根据前面多个时刻的输入来共同影响的，而有些时候预测可能需要由前面若干输入和后面若干输入共同决定，这样会更加准确。

鉴于单向循环神经网络在某些情况下的不足，提出了双向循环神经网络，因为在许多应用中是需要能关联未来的数据，而单向循环神经网络属于关联历史数据，所以对于未来数据的关联就提出了反向循环神经网络，两个方向的网络结合到一起就能关联历史与未来了。

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。

5.4 双向循环神经网络

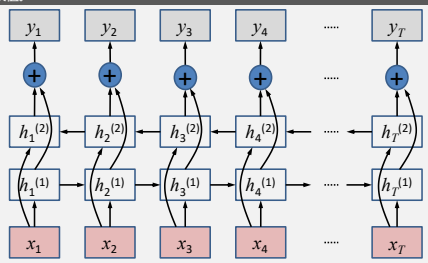


$h = [\vec{h}; \overleftarrow{h}]$ now represents (summarizes) the past and future around a single token.

按时间展开的双向循环神经网络结构图

5.4 双向循环神经网络

双向循环神经网络按时刻展开的结构如图8-17所示，可以看到向前和向后层共同连接着输出层，其中包含了6个共享权值，分别为输入到向前层和向后层两个权值、向前层和向后层各自隐含层到隐含层的权值、向前层和向后层各自隐含层到输出层的权值。



按时间展开的双向循环神经网络结构图

5.4 双向循环神经网络

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

$$\mathbf{h}_t^{(1)} = f(U^{(1)}\mathbf{h}_{t-1}^{(1)} + W^{(1)}\mathbf{x}_t + \mathbf{b}^{(1)}) \quad (8\text{-}40)$$

$$\mathbf{h}_t^{(2)} = f(U^{(2)}\mathbf{h}_{t+1}^{(2)} + W^{(2)}\mathbf{x}_t + \mathbf{b}^{(2)}) \quad (8\text{-}41)$$

$$\mathbf{h}_t = \mathbf{h}_t^{(1)} \oplus \mathbf{h}_t^{(2)} \quad (8\text{-}42)$$

式(8-42)中，$\oplus$为向量拼接操作。

从图8-17以及式（8-40）、式（8-41）以及式（8-42）中可以看出一般的规律：正向计算时，隐藏层的值$h_t^{(1)}$与$h_{t-1}^{(1)}$有关；反向计算时，隐藏层的值$h_t^{(2)}$与$h_{t+1}^{(2)}$有关；最终的输出取决于正向和反向计算的求和。

从式（8-40）、式（8-41）以及式（8-42）中还可以看到，正向计算和反向计算不共享权值，也就是说$U^{(1)}$和$U^{(2)}$、$W^{(1)}$和$W^{(2)}$、$b^{(1)}$和$b^{(2)}$、$V^{(1)}$和$V^{(2)}$都是不同的权重矩阵。

双向RNN需要的内存是单向RNN的两倍，因为在同一时间点，双向RNN需要保存两个方向上的权重参数，在分类的时候，需要同时输入两个隐藏层输出的信息。

第五章　循环神经网络

5.5 循环神经网络的应用

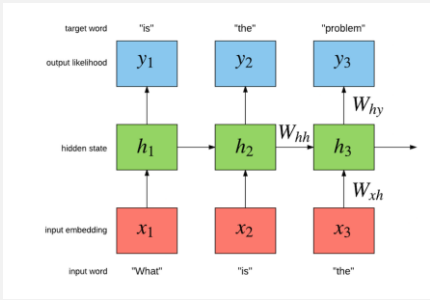
1. 语言模型

- 自然语言理解 → 一个句子的可能性/合理性
  - ！在报那猫告做只
  - 那只猫在作报告！
  - 那个人在作报告！

- 一切都是概率！
  - $P(x_1, x_2, \cdots, x_T)$
  - $= \prod_i P(x_i | x_{i-1}, \cdots, x_1)$
  - $\approx \prod_i P(x_i | x_{i-1}, \cdots, x_{i-n+1})$

5.5 循环神经网络的应用

1. 语言模型



5.5 循环神经网络的应用

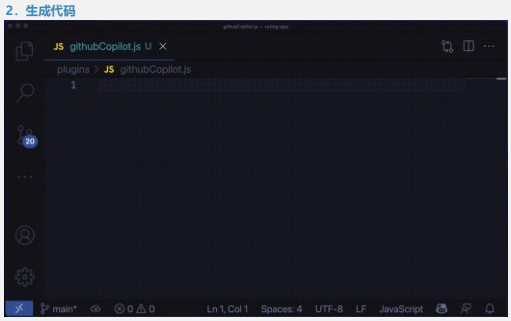
2. 生成LINUX内核代码

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
```

5.5 循环神经网络的应用

2. 生成代码



5.5 循环神经网络的应用

3. 作词机

- RNN在“学习”过汪峰全部作品后自动生成的歌词
- https://github.com/phunterlau/wangfeng-rnn

我在这里中的夜里
就像一场是一种生命的意叶
就像我的生活变得在我一样
可我们这是一个知道
我只是一天你会怎吗
可我们这是我们的是不要为你
我们想这有一种生活的时候

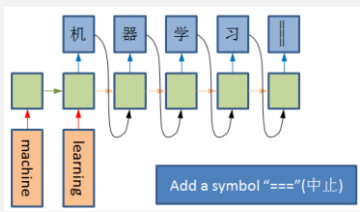
5.5 循环神经网络的应用

4. 作诗 （九歌）

5.5 循环神经网络的应用

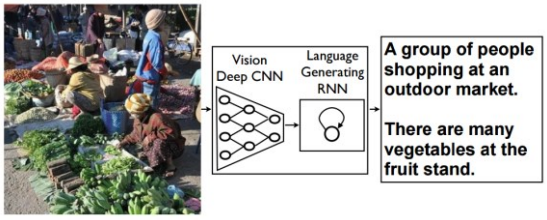
5. 基于序列到序列的机器翻译

机器翻译如图所示。将整个句子输入循环神经网络后，这个时候最后一刻的输出就已经处理完了整个句子。



机器翻译示意图

6. 看图说话



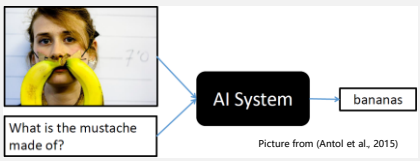
6. 看图说话

Figure 5. A selection of evaluation results, grouped by human rating.

7. 视觉问答 (Visual Question Answer)

Demo Website

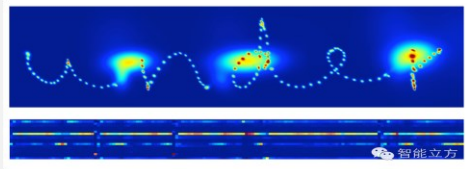
VQA: Given an image and a natural language question about the image, the task is to provide an accurate natural language answer



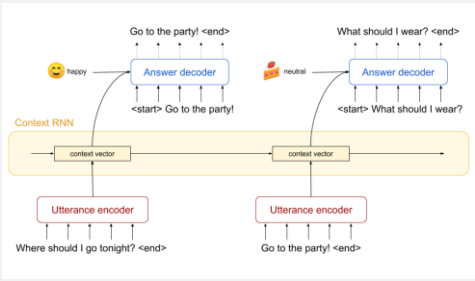
Picture from (Antol et al., 2015)

8. 写字

把一个字母的书写轨迹看作是一连串的点。一个字母的“写法”其实是每一个点相对于前一个点的偏移量，记为(offset x, offset y)。再增加一维取值为0或1来记录是否应该“提笔”。



9. 对话系统



CakeChat: Emotional Generative Dialog System

10. 情感分析

情感分析（Sentiment Analysis），又称倾向性分析，意见抽取（Opinion Extraction），意见挖掘（Opinion Mining），情感挖掘（Sentiment Mining），主观分析（Subjectivity Analysis），它是对带有情感色彩的主观性文本进行分析、处理、归纳和推理的过程。

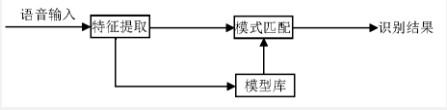
情感分析最常用的做法就是在文中找到具有各种感情色彩属性的词，统计每个属性词的个数，哪个类多，这段话就属于哪个属性。但是这存在一个问题，例如 don't like，一个属于否定，一个属于肯定，统计之后变成 0 了，而实际上应该是否定的态度。再有一种情况是，前面几句是否定，后面又是肯定，那整段到底是中立还是肯定呢，为了解决这样的问题，就需要考虑上下文的环境。

5.5 循环神经网络的应用

11. 语音识别

语音识别技术是一门交叉技术，近二十年来，语音识别技术取得显著进步，开始从实验室走向市场。人们预计，未来10年内，语音识别技术将进入工业、家电、通信、汽车电子、医疗、家庭服务、消费电子产品等各个领域。语音识别技术，也被称为自动语音识别，其目标是将人类的语音中的词汇内容转换为计算机可读的输入，要实现语音识别，其实现过程如图所示。

语音识别方法主要是模式匹配法，其包括两个阶段，其一是训练阶段，用户将词汇表中的所有词依次说一遍，并且将其特征矢量作为模板存入模型库；其二是识别阶段，将输入语音的特征矢量依次与模型库中的每个模板进行相似度比较，将相似度最高者作为识别结果的输出。



语音识别过程

第五章 循环神经网络

习题：

1. 简述循环神经网络工作过程。
2. 循环神经网络同卷积神经网络有什么区别?
3. 计算式(8-18)、式（8-19）和式(8-20) 中的梯度。
4. 计算LSTM网络中参数的梯度，并分析其避免梯度消失的效果。
5. 计算GRU网络中参数的梯度，并分析其避免梯度消失的效果。
6. 查找资料，为何能将GRU的输入门和遗忘门合并成一个门。
7. 简述双向循环神经网络的工作过程。

5.循环神经网络除了本章介绍的基本应用外，还有什么其他应用?

感谢聆听