

LSTM RNN 简介

潘睿

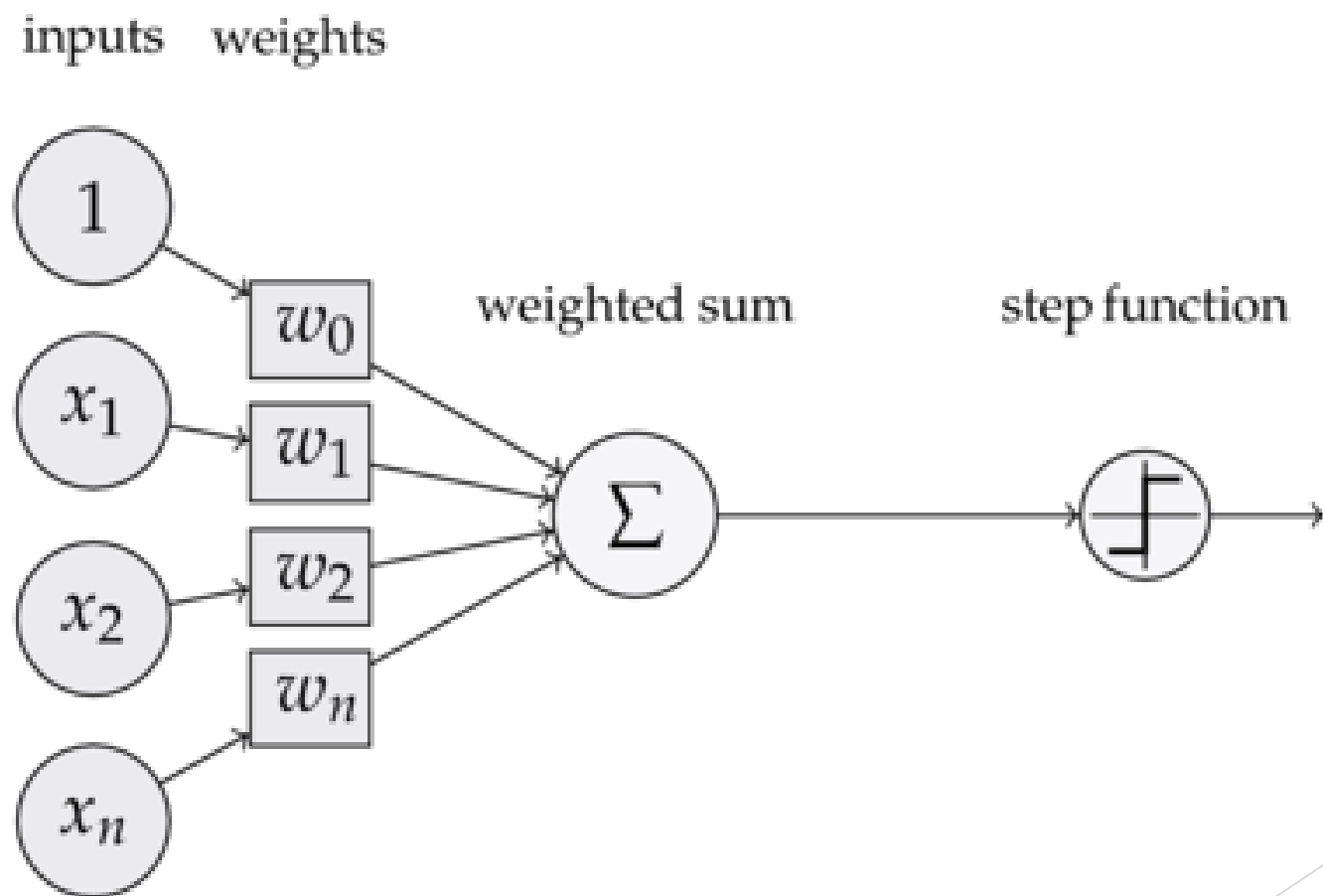
- ▶ **历史 & 基础知识**
- ▶ 简介
- ▶ 优化
- ▶ 应用
- ▶ 实现细节
- ▶ **工具库**
- ▶ 展望

历史& 基础知识

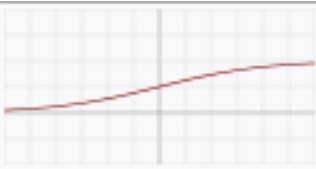
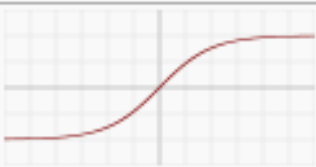
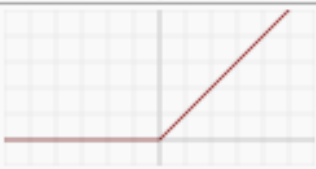
- ▶ 1943年MP模型提出
- ▶ 标志着神经计算时代的开始

- ▶ 1957年提出Perceptron模型（感知器）
- ▶ 可以解决分类问题

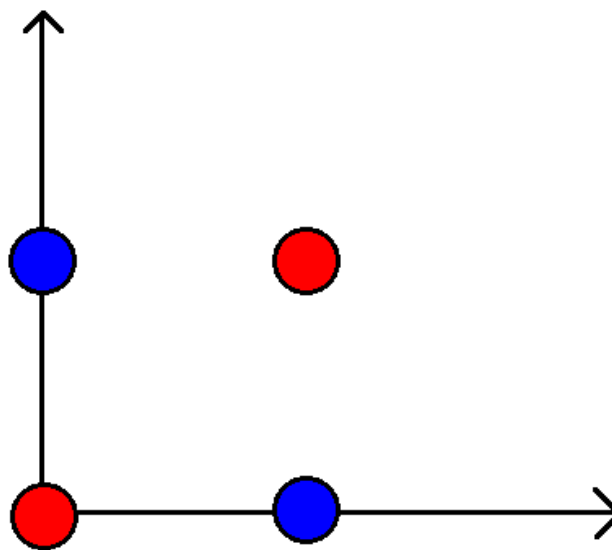
► 单个神经元



- Step function 也叫Activation function
- 比较常用的有

| | | | |
|------------------------------|--|--|---|
| Sigmoid |  | $f(x) = \frac{1}{1 + e^{-x}}$ | $f'(x) = f(x)(1 - f(x))$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ |
| Rectified Linear Unit (ReLU) |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |

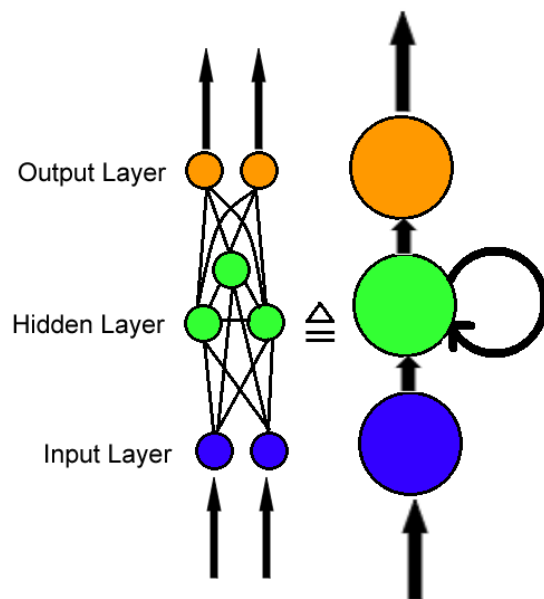
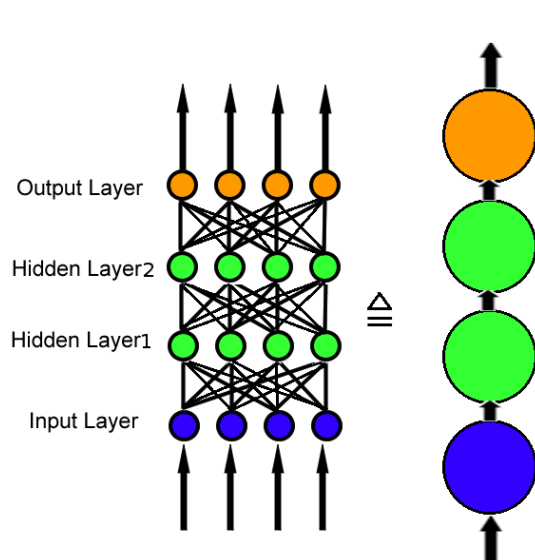
- ▶ 1969年Perceptron被指出只能做线性划分
- ▶ 神经网络进入第一次低潮



- ▶ 1982年John J. Hopfield提出一个具有完整理论基础的神经网络模型
- ▶ 神经网络复兴时期开始

- ▶ 1986年提出Back Propagation (BP) 反向传播学习算法
- ▶ 流行至今

- ▶ 同一时期
- ▶ Multilayer perceptron (MLP) 多层感知器
- ▶ Recurrent Neural Network (RNN) 循环神经网络



► Multilayer perceptron 多层感知器

- 定义

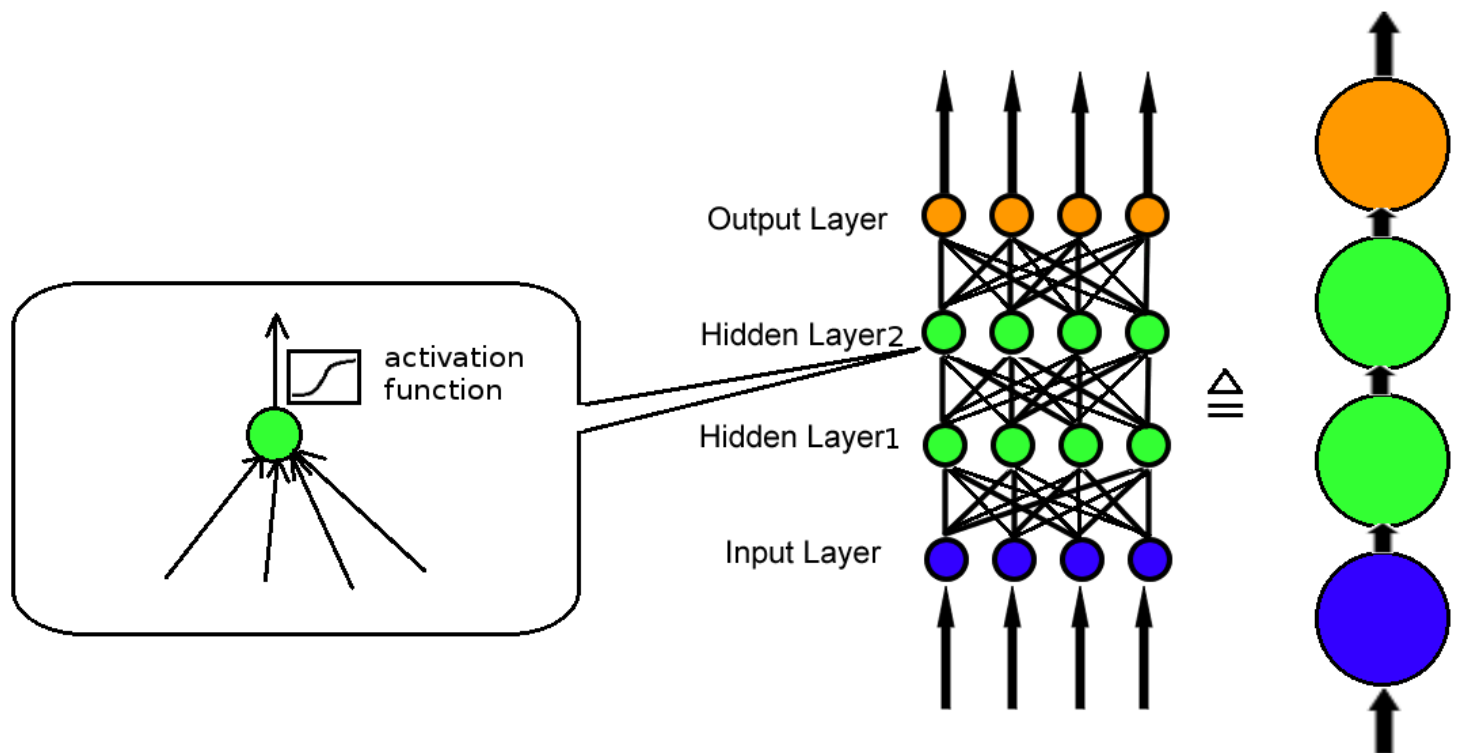
- 训练

- 理论基础

► Multilayer perceptron 多层感知器

► 定义

- An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one.
- 多层感知器就是多层神经元，每层有一堆神经元，这一层和下一层的每个神经元全连上



▶ Multilayer perceptron 多层感知器

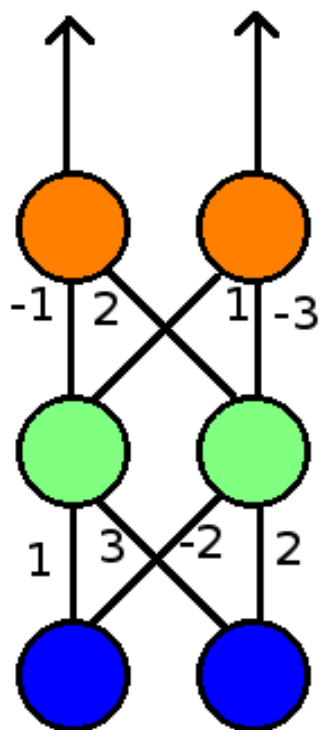
▶ 训练

- ▶ Forward Propagation 前向传播
- ▶ Back Propagation 反向传播

► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播



Rectified Linear
Unit (ReLU)

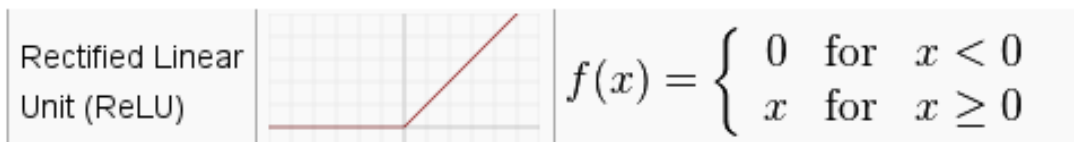
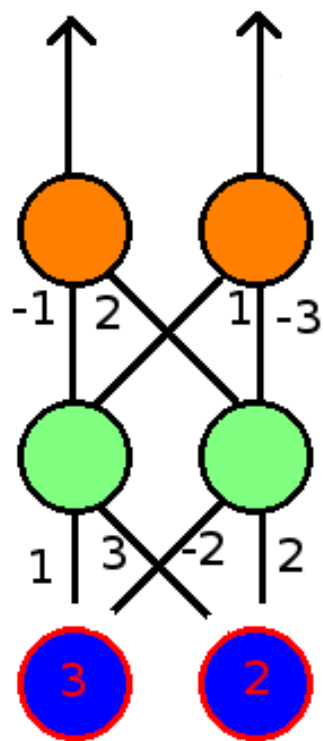


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

► Multilayer perceptron 多层感知器

► 训练

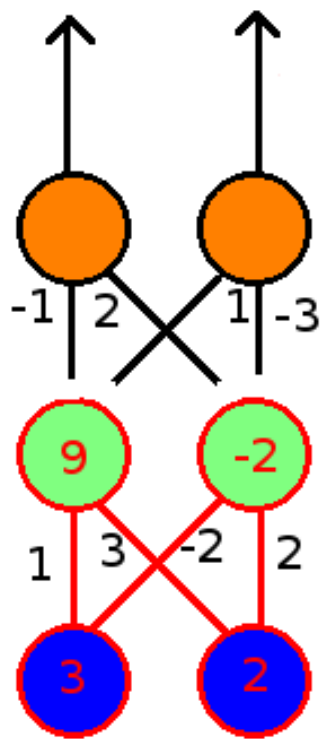
► Forward Propagation 前向传播



► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播



Rectified Linear
Unit (ReLU)

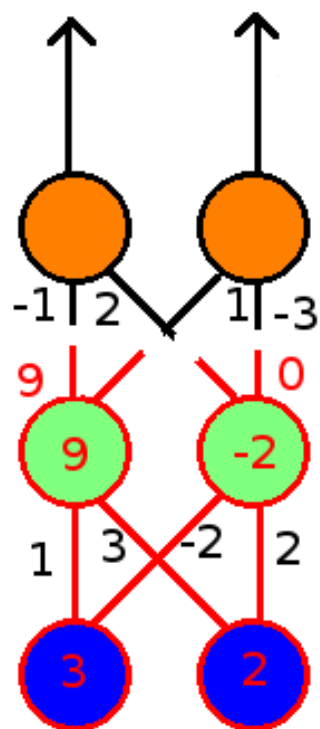


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

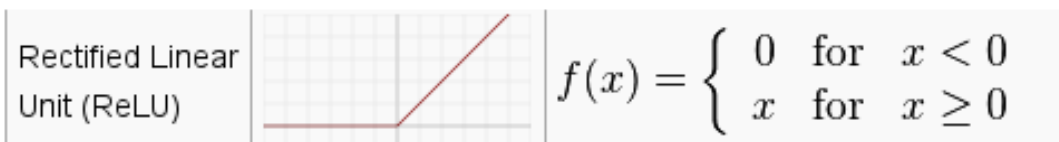
► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播



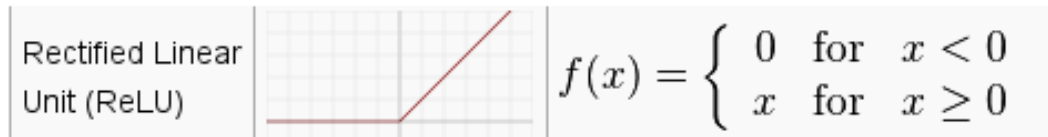
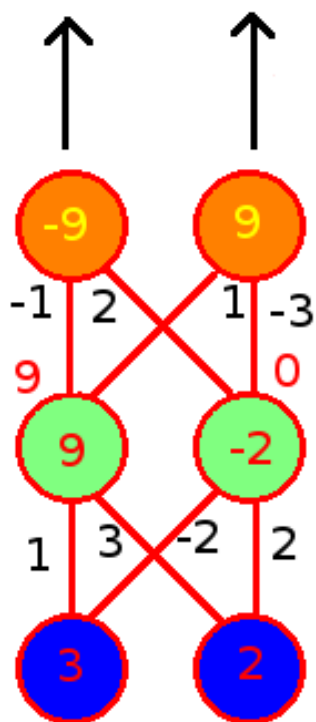
Activation



► Multilayer perceptron 多层感知器

► 训练

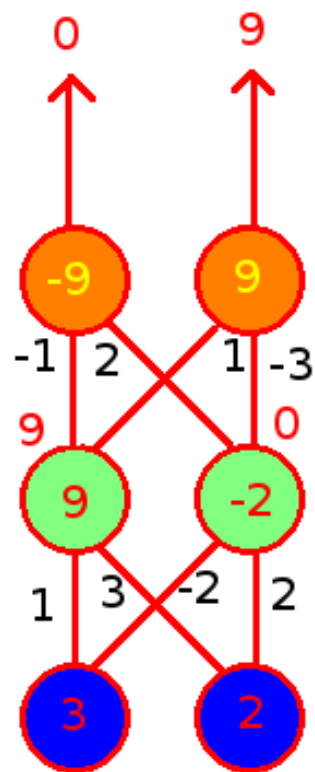
► Forward Propagation 前向传播



► Multilayer perceptron 多层感知器

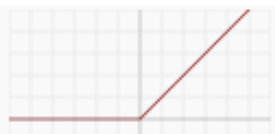
► 训练

► Forward Propagation 前向传播



Activation

Rectified Linear
Unit (ReLU)



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

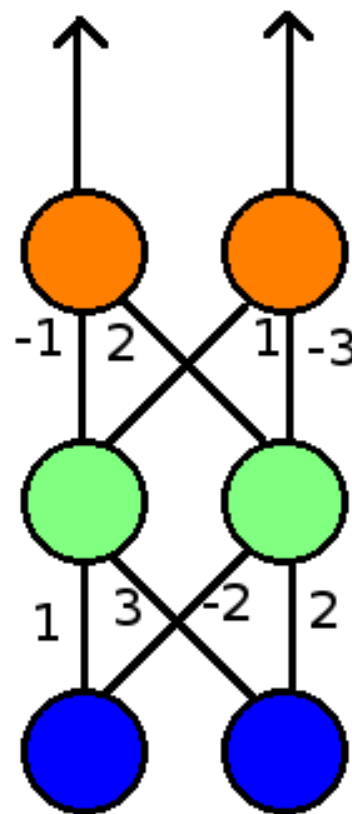
► Multilayer perceptron 多层感知器

► 训练

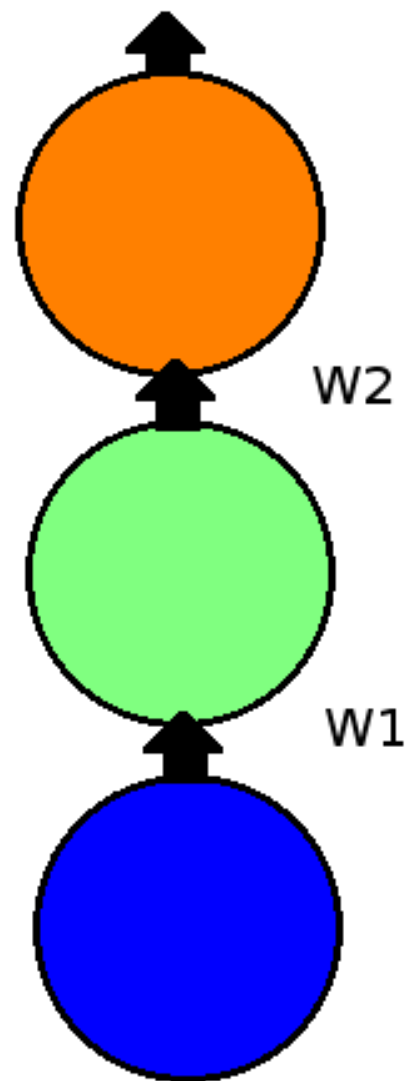
► Forward Propagation 前向传播

► 矩阵的观点

► $W1 =$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$
► $W2 =$
► $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



\cong



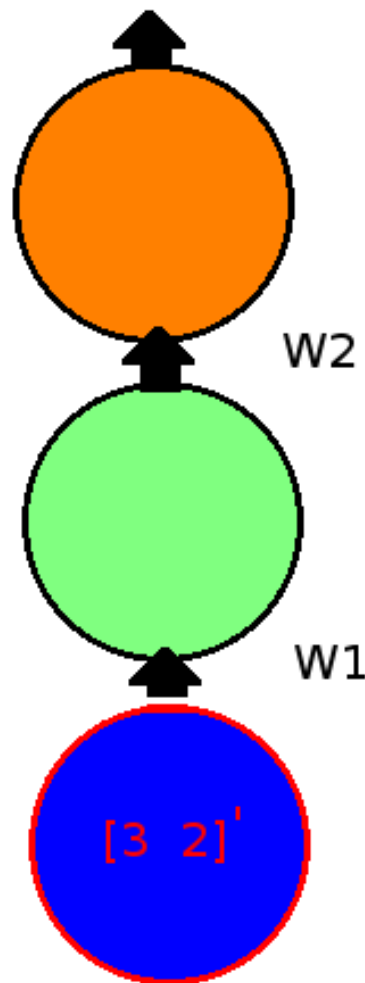
► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播

► 矩阵的观点

► $W1 =$ $W2 =$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



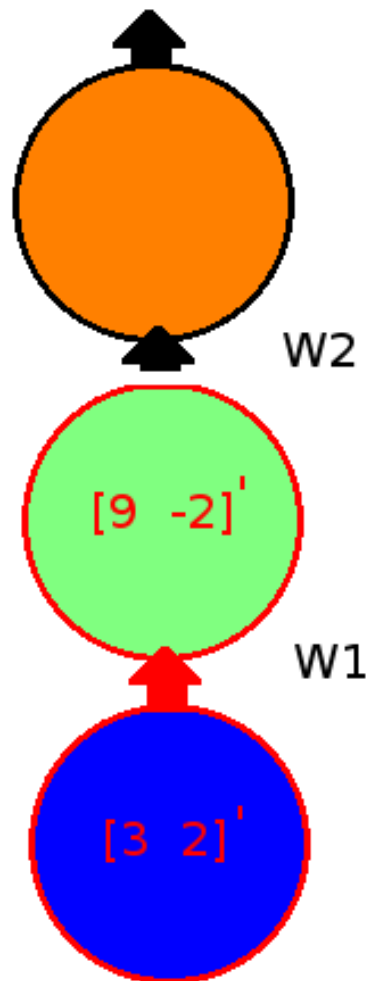
► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播

► 矩阵的观点

► $W1 =$ $W2 =$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



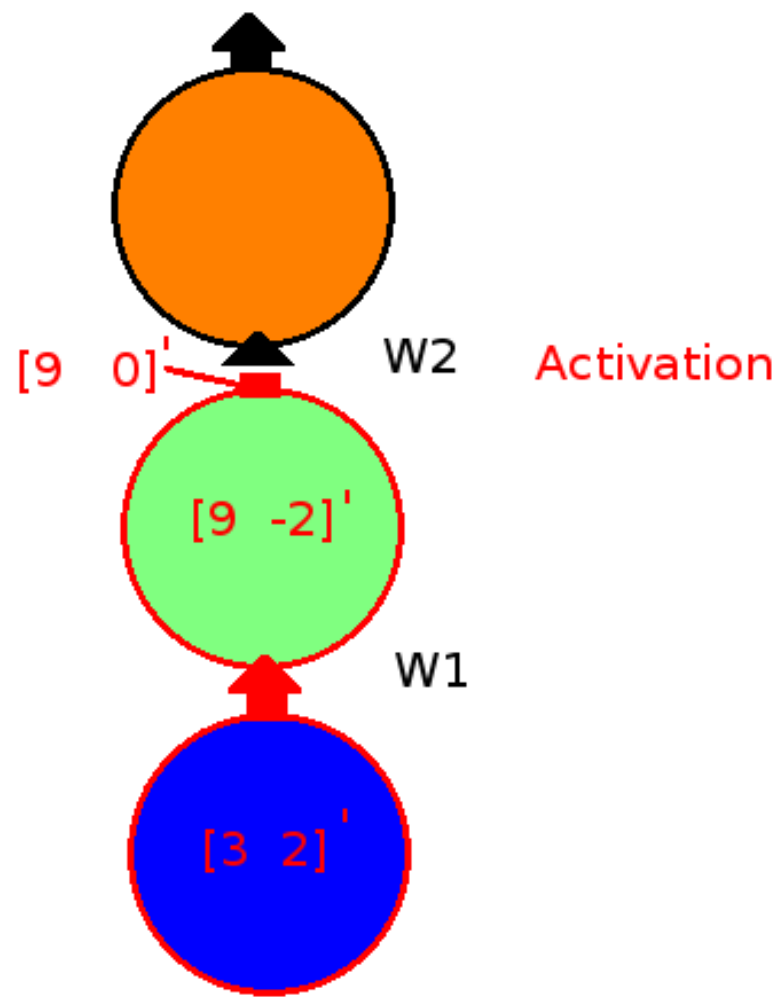
► Multilayer perceptron 多层感知器

► 训练

- Forward Propagation 前向传播

► 矩阵的观点

- $W1 =$
- $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$
- $W2 =$
- $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



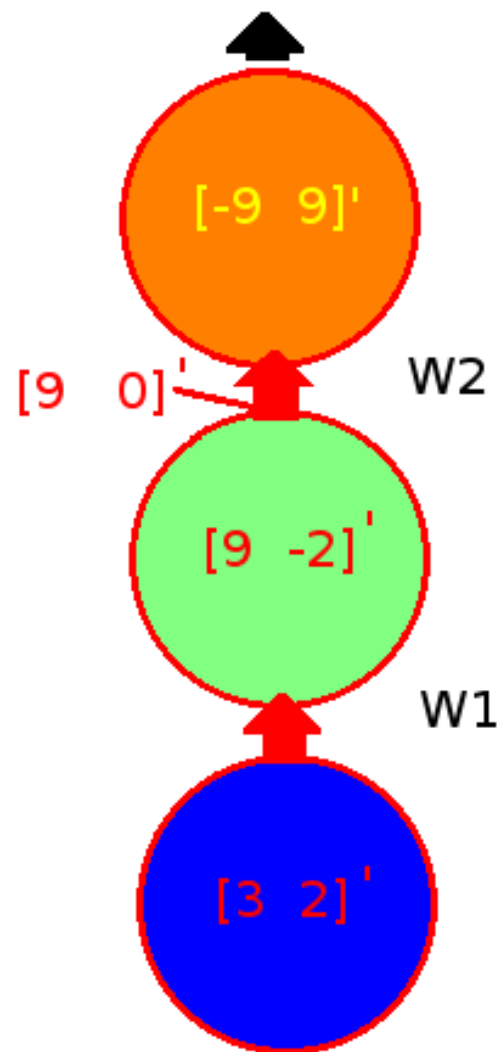
► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播

► 矩阵的观点

► $W1 =$ $W2 =$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



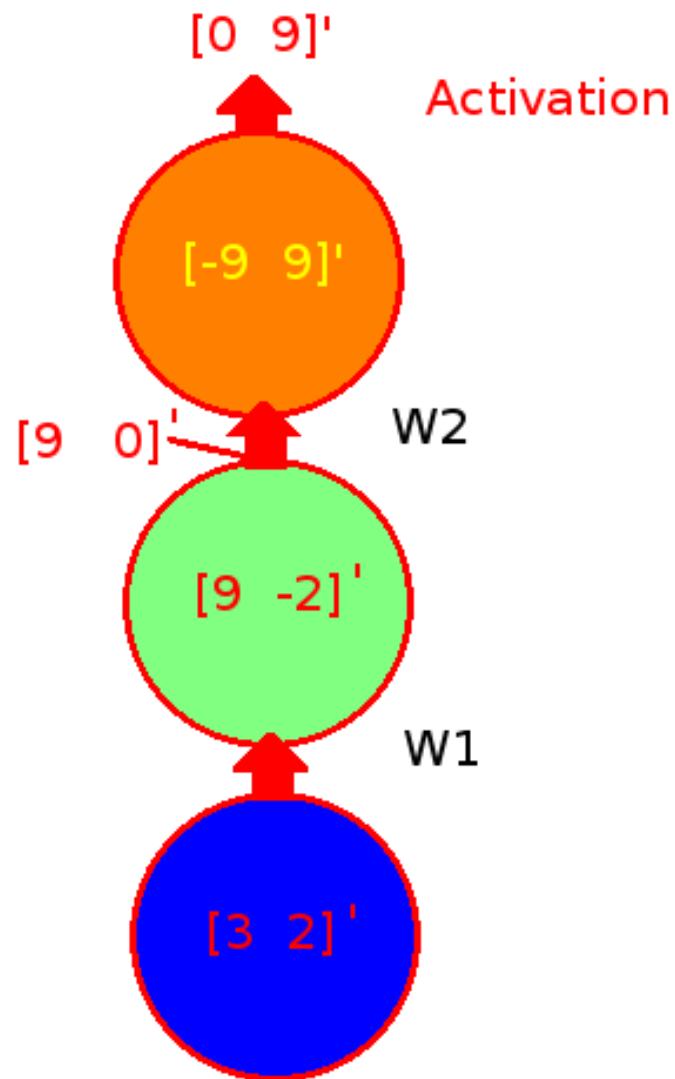
► Multilayer perceptron 多层感知器

► 训练

► Forward Propagation 前向传播

► 矩阵的观点

► $W1 =$ $W2 =$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$



- ▶ Multilayer perceptron 多层感知器
- ▶ 训练
 - ▶ Back Propagation 反向传播
- ▶ 高中题目：如何计算 $f(g(x))$ 的导函数

- ▶ Multilayer perceptron 多层感知器
- ▶ 训练
 - ▶ Back Propagation 反向传播
- ▶ 高中题目：如何计算 $f(g(x))$ 的导函数
- ▶ 答： $f'(g(x)) * g'(x)$

- ▶ Multilayer perceptron 多层感知器
- ▶ 训练
 - ▶ Back Propagation 反向传播
- ▶ 大学题目：如何计算 $f(g(\mathbf{x}))$ 的导函数

► Multilayer perceptron 多层感知器

► 训练

► Back Propagation 反向传播

► 大学题目：如何计算 $f(g(\mathbf{x}))$ 的导函数

► 答： $D_{\mathbf{a}}(f \circ g) = D_{g(\mathbf{a})} f \circ D_{\mathbf{a}} g,$

$$Df(\mathbf{a}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{a}) & \frac{\partial f_1}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{a}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{a}) & \frac{\partial f_2}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_2}{\partial x_n}(\mathbf{a}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{a}) & \frac{\partial f_m}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_m}{\partial x_n}(\mathbf{a}) \end{bmatrix}.$$

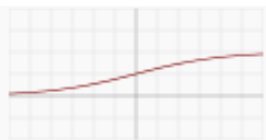
► Multilayer perceptron 多层感知器

► 训练

► Back Propagation 反向传播

► 一个特例：如何计算 $\sigma(\mathbf{W}\mathbf{x})$ 的导函数

Sigmoid



$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = f(x)(1 - f(x))$$

$$Df(\mathbf{a}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{a}) & \frac{\partial f_1}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{a}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{a}) & \frac{\partial f_2}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_2}{\partial x_n}(\mathbf{a}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{a}) & \frac{\partial f_m}{\partial x_2}(\mathbf{a}) & \dots & \frac{\partial f_m}{\partial x_n}(\mathbf{a}) \end{bmatrix}.$$

- ▶ Multilayer perceptron 多层感知器
- ▶ 训练

- ▶ Back Propagation 反向传播

- ▶ 一个特例：如何计算 $\sigma(\mathbf{W}\mathbf{x})$ 的导函数

- ▶ 答： $\sigma'(\mathbf{W}\mathbf{x}) * \mathbf{W} = [\sigma(\mathbf{W}\mathbf{x})(1-\sigma(\mathbf{W}\mathbf{x}))] * \mathbf{W}$

Hadamard product (逐元素相乘)

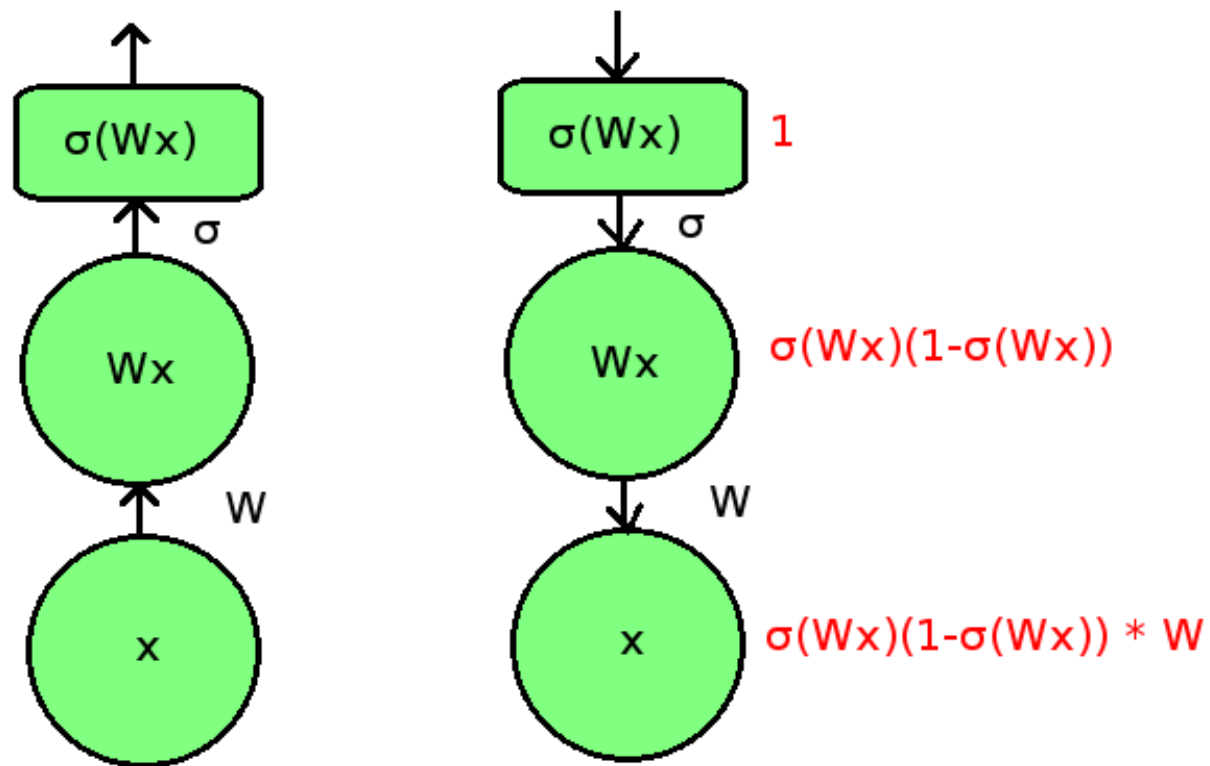
全1向量

矩阵乘法

► Multilayer perceptron 多层感知器

► 训练

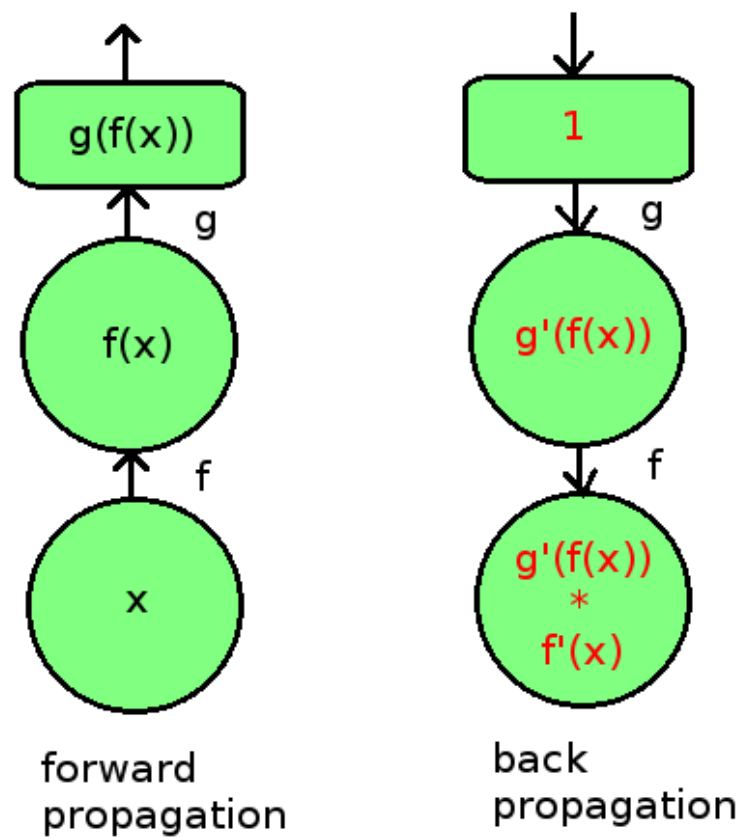
► Back Propagation 反向传播



► Multilayer perceptron 多层感知器

► 训练

► Back Propagation 反向传播



► Multilayer perceptron 多层感知器

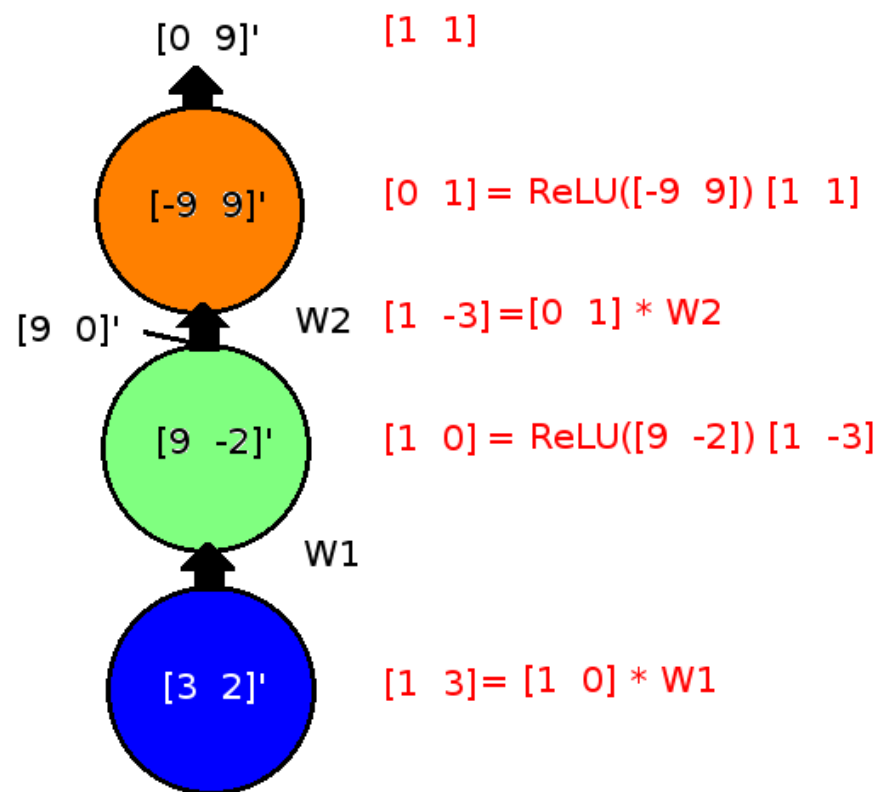
► 训练

► Back Propagation 反向传播

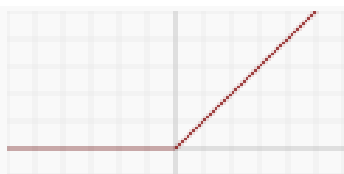
► $W1 =$ $W2 =$

► $\begin{bmatrix} 1 & 3 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \end{bmatrix}$

► $\begin{bmatrix} -2 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & -3 \end{bmatrix}$



Rectified Linear
Unit (ReLU)



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

- ▶ Multilayer perceptron 多层感知器

- ▶ 训练

 - ▶ Back Propagation 反向传播

- ▶ $W1 =$ $W2 =$

- ▶ $\begin{bmatrix} 1 & 3 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \end{bmatrix}$

- ▶ $\begin{bmatrix} -2 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & -3 \end{bmatrix}$

- ▶ $W1, W2$ 的导函数？

► Multilayer perceptron 多层感知器

► 训练

► Back Propagation 反向传播

► $W1 =$ $W2 =$

► $\begin{bmatrix} 1 & 3 \end{bmatrix}$ $\begin{bmatrix} -1 & 2 \end{bmatrix}$

► $\begin{bmatrix} -2 & 2 \end{bmatrix}$ $\begin{bmatrix} 1 & -3 \end{bmatrix}$

$$\frac{\partial f(Wx)}{\partial W} = \frac{\partial f(Wx)}{Wx} * \frac{\partial (Wx)}{W} = \frac{\partial f(Wx)}{\partial Wx} * x$$

► Multilayer perceptron 多层感知器

► 训练

► Back Propagation 反向传播

► $W1 =$

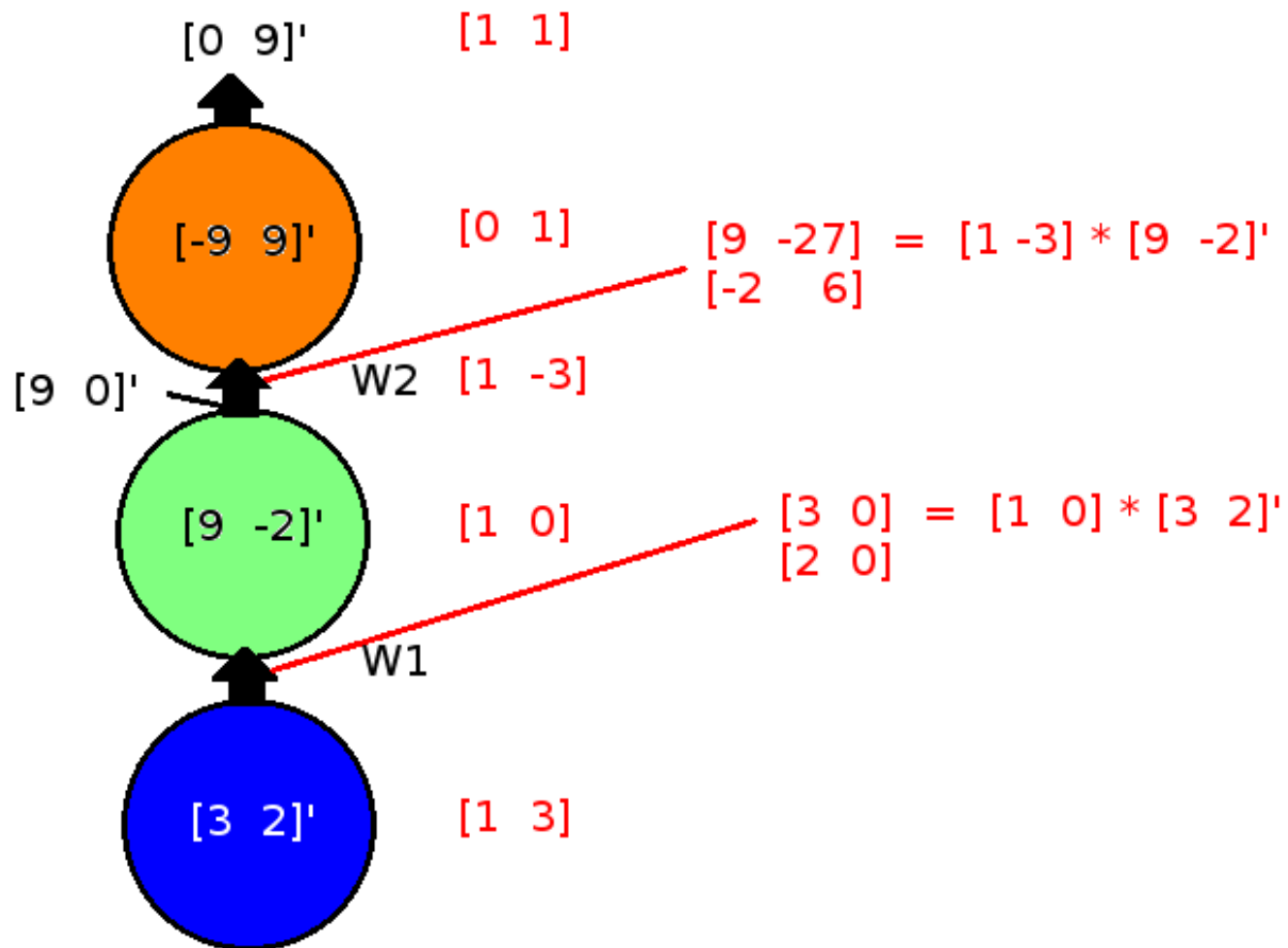
► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$

► $\begin{bmatrix} 1 & 3 \\ -2 & 2 \end{bmatrix}$

$W2 =$

$\begin{bmatrix} -1 & 2 \\ 1 & -3 \end{bmatrix}$

$\begin{bmatrix} 1 & -3 \\ 1 & -3 \end{bmatrix}$



► Multilayer perceptron 多层感知器

► 理论基础

► Universal approximation theorem 通用近似定理

► 大意就是三层神经网络只要隐层点够多，就可以近似任何连续函数

► 视觉直观解释见

<http://neuralnetworksanddeeplearning.com/chap4.html>

Let $\varphi(\cdot)$ be a nonconstant, **bounded**, and **monotonically-increasing continuous** function. Let I_m denote the m -dimensional **unit hypercube** $[0, 1]^m$. The space of continuous functions on I_m is denoted by $C(I_m)$. Then, given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exists an integer N and real constants $v_i, b_i \in \mathbb{R}$, where $i = 1, \dots, N$ such that we may define:

$$F(x) = \sum_{i=1}^N v_i \varphi(w_i^T x + b_i)$$

as an approximate realization of the function f where f is independent of φ ; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are **dense** in $C(I_m)$.

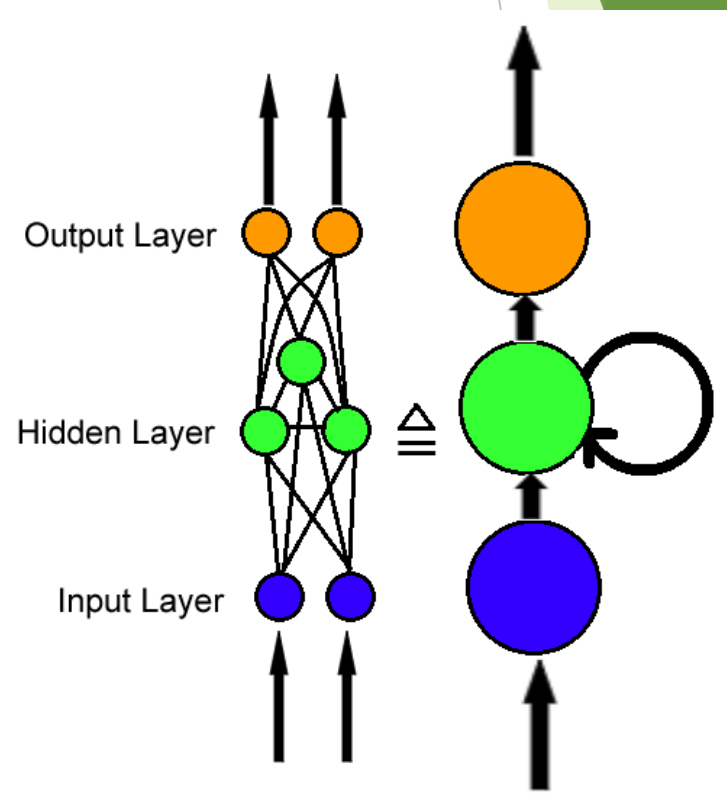
▶ Recurrent Neural Network 循环神经网络

- ▶ 定义
- ▶ 用法
- ▶ 训练
- ▶ 优缺点

► Recurrent Neural Network 循环神经网络

► 定义

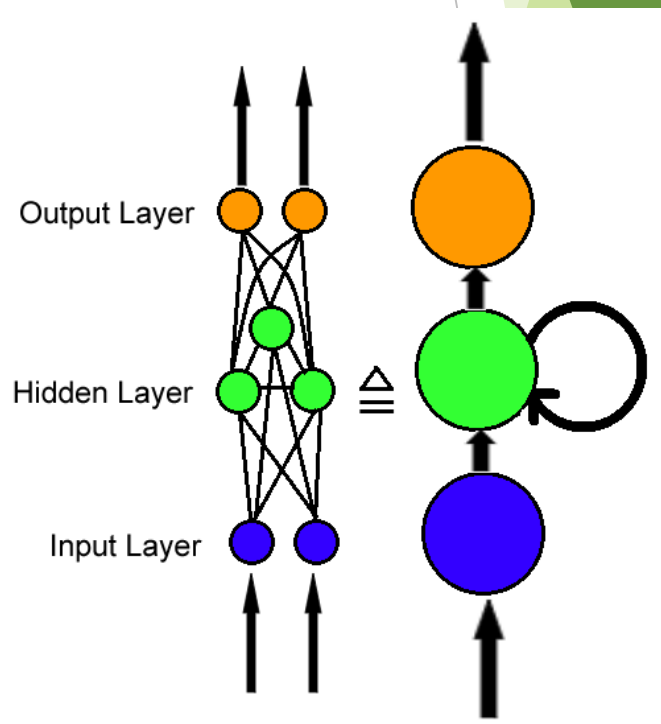
- A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle.
- 这个定义更广泛些，下图所示的是当前比较常见的RNN



► Recurrent Neural Network 循环神经网络

► 定义

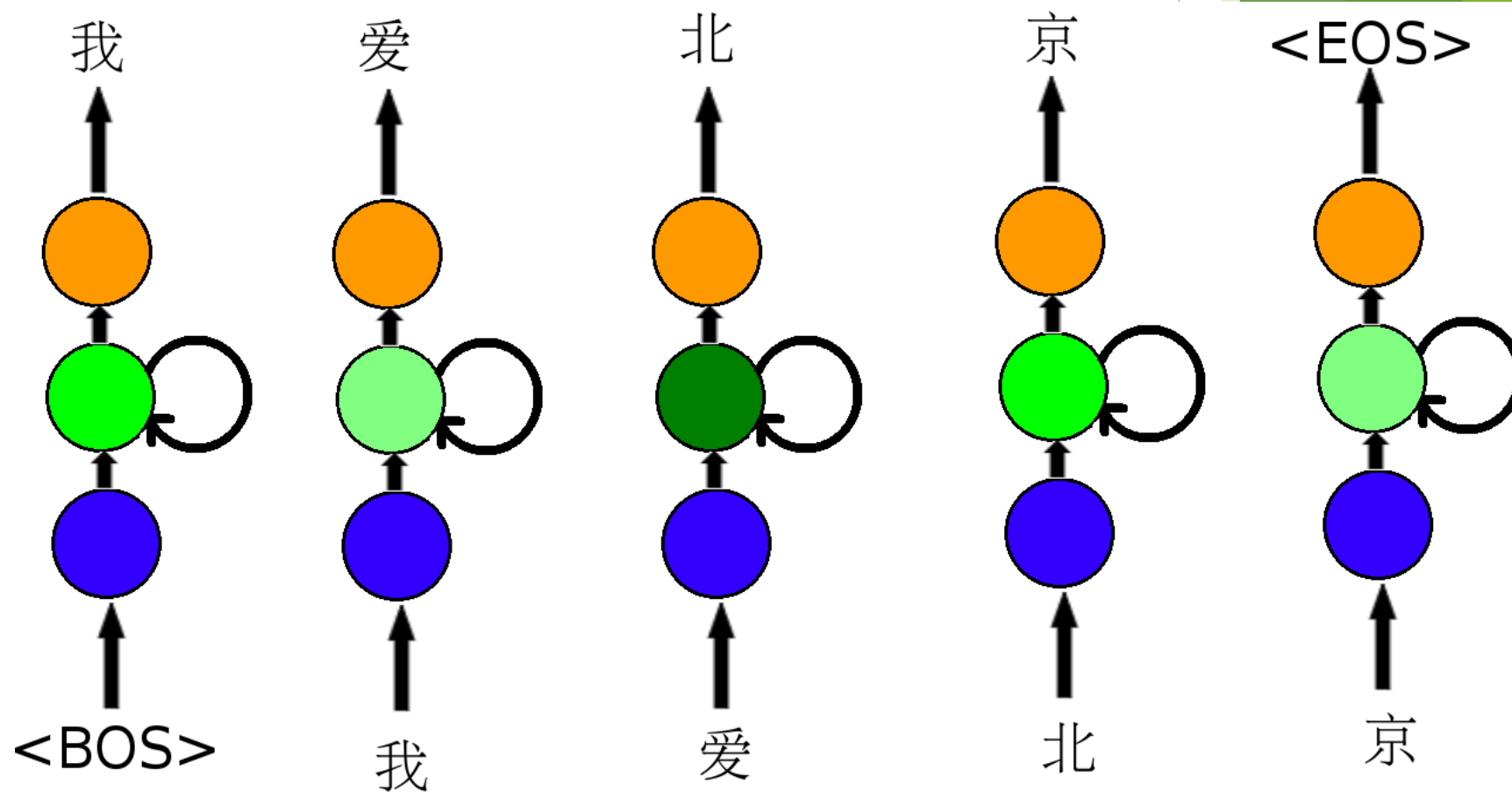
- 注意不要与Recursive Neural Network (递归神经网络) 弄混
- Recurrent Neural Network是Recursive Neural Network的一个特例
- Recursive Neural Network代表了更复杂的结构 (树状, 图状)
- Recurrent Neural Network只代表**链状(?)** 时序结构



► Recurrent Neural Network 循环神经网络

► 用法

► 序列映射



▶ Recurrent Neural Network循环神经网络

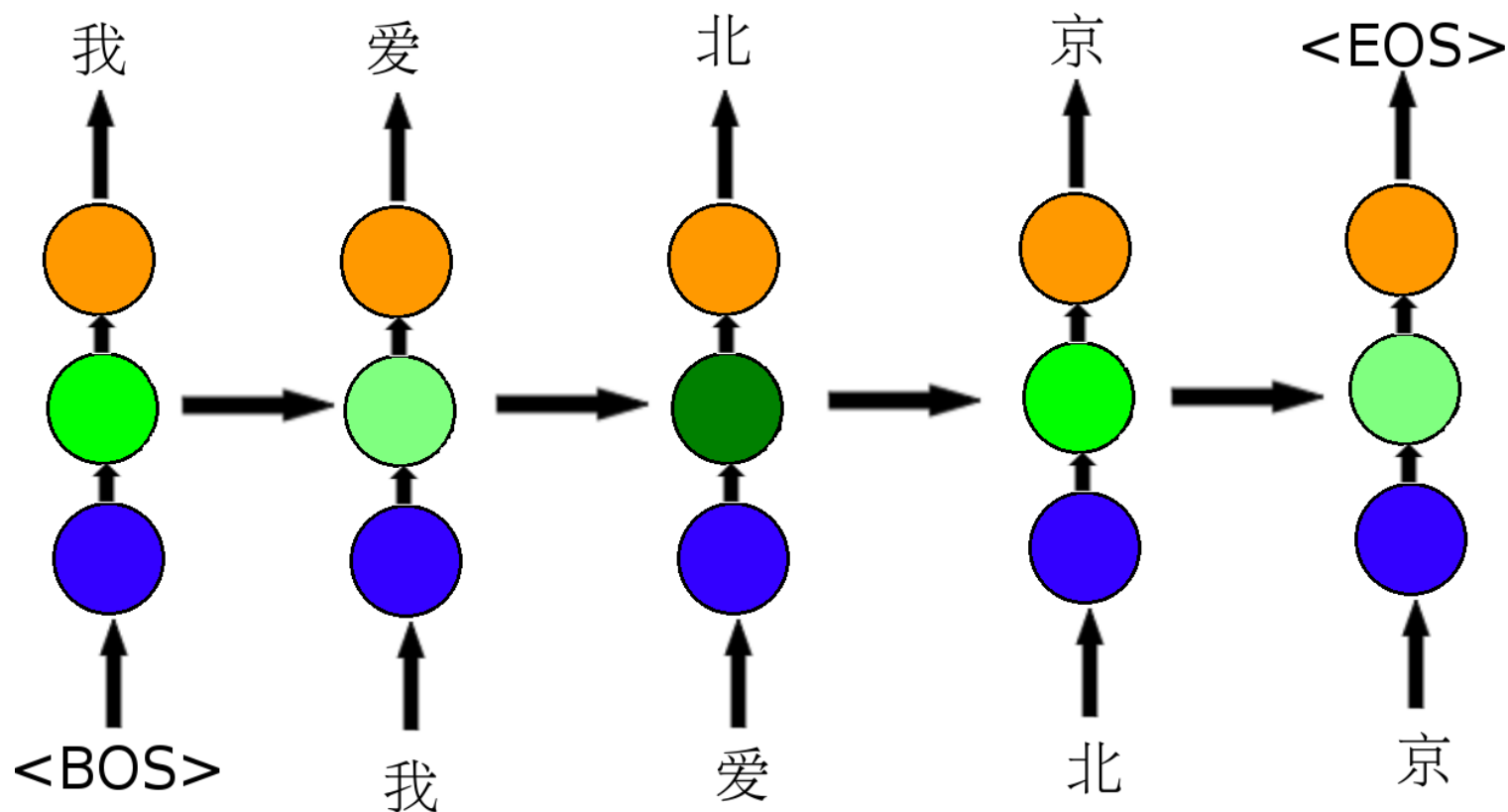
▶ 训练

- ▶ Back Propagation Through Time (BPTT)沿时间反向传播
- ▶ 1987、1988、1995年由多名研究者分别独立提出
- ▶ BPTT有多种变种，这里只介绍目前最常用的一种

► Recurrent Neural Network 循环神经网络

► 训练

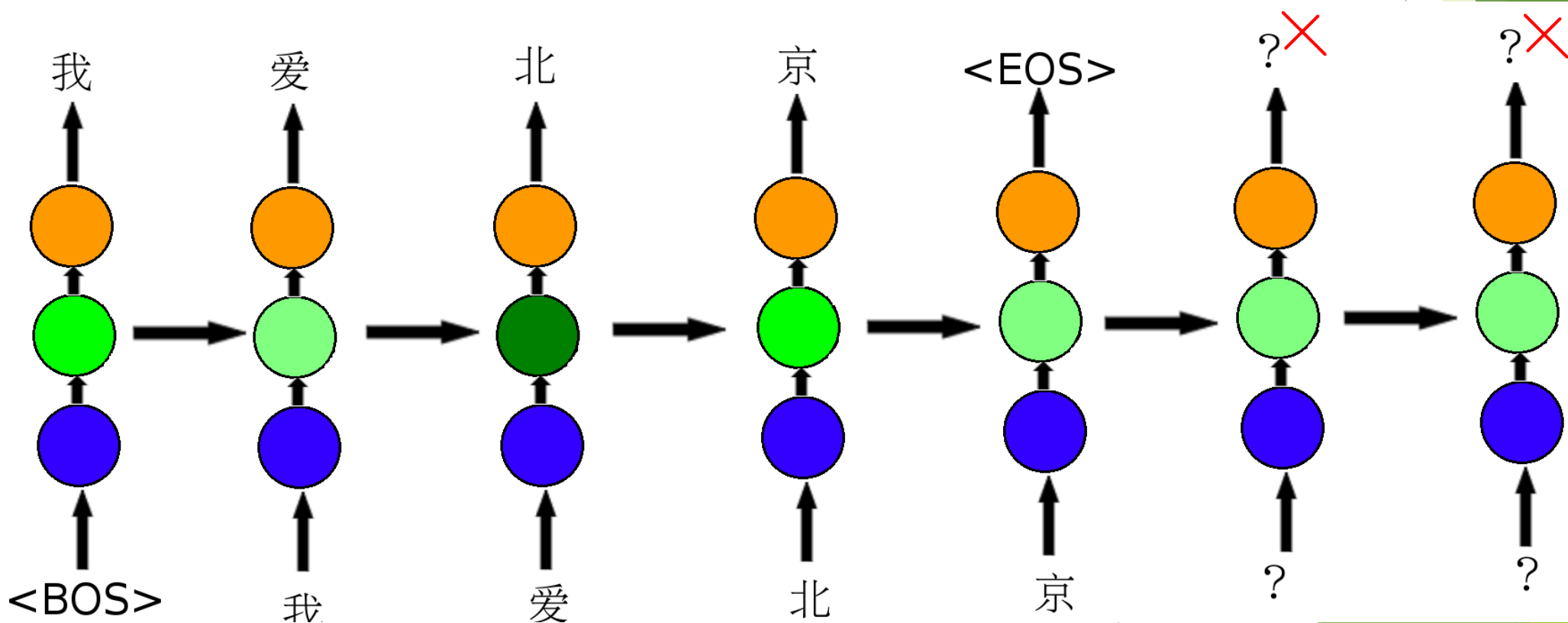
- Back Propagation Through Time (BPTT) 沿时间反向传播



► Recurrent Neural Network循环神经网络

► 训练

- 实际情况中往往展开一个最大的固定长度（下图中为7）



▶ Recurrent Neural Network循环神经网络

▶ 训练

- ▶ 对于过长的数据，可以作截断处理
- ▶ 使用上一截的隐层状态初始化（当然梯度仍无法回传至上一截数据）

▶ Recurrent Neural Network循环神经网络

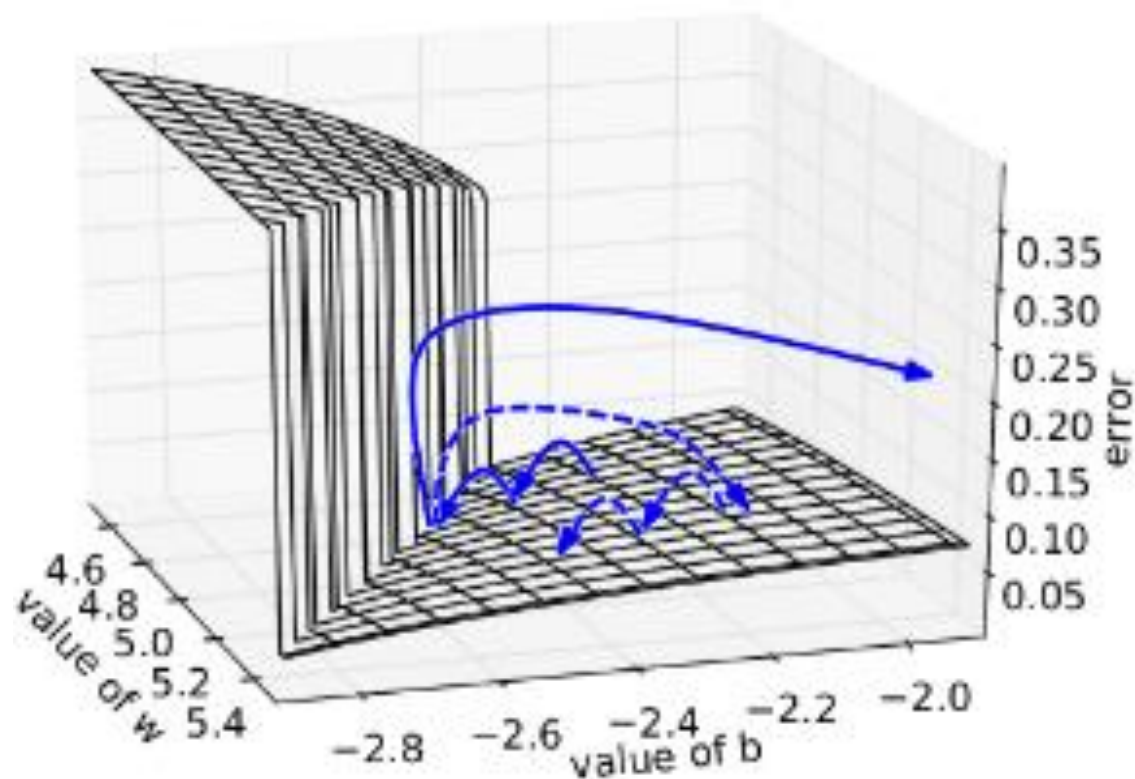
▶ 优点

- ▶ 相比多层感知器，可以处理变长数据

▶ 缺点

- ▶ Gradient Explosion （梯度爆炸）
- ▶ Gradient Vanishing （梯度消失）

- ▶ Recurrent Neural Network循环神经网络
- ▶ 尝试
 - ▶ [2012] On the difficulty of training recurrent neural networks



- ▶ 1997年Long Short-Term Memory Recurrent Neural Network (LSTM RNN)提出
- ▶ 当时神经网络正因为Support Vector Machine (SVM)支撑向量机的火爆而陷入低潮

► LSTM最初的提出有几个原因

- 为了解决RNN的Gradient Explosion/Vanishing的问题
- 为了得到能够学习如何学习的神经网络

- LSTM, 5000 weights, 5 months training:
metalearns fast online learning algorithm for
quadratic functions $f(x,y)=a_1x^2+a_2y^2+a_3xy+a_4x+a_5y+a_6$
Huge time lags.
- After metalearning, **freeze weights**.
- **Now use net**: Select new f , feed training exemplars
...data/target/data/target/data... **into input units**, one
at a time. After 30 exemplars the net predicts target
inputs before it sees them.

No weight changes!

How?

- ▶ Metalearner也用RNN应用过，但没有LSTM如此好的效果
- ▶ LSTM的Memory存储了神奇的知识

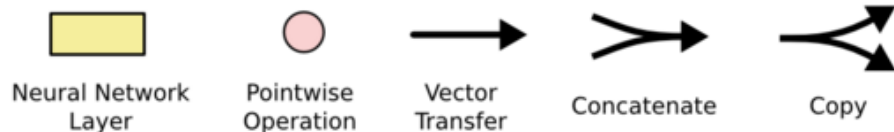
- ▶ 2006年，Deeplearning在ImageNet上取得巨大突破
- ▶ 深度学习开始火爆

- ▶ 2013年，LSTM在Speech Recognition上取得很好的效果
 - ▶ SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS
- ▶ 2014年，LSTM在Machine Translation上取得不错的效果，进入NLP领域
 - ▶ Sequence-to-sequence-learning-with-neural-networks
- ▶ 只要是和序列映射有关的，LSTM都可以做！
- ▶ LSTM开始火热

简介

- ▶ LSTM的结构
- ▶ LSTM的思想
- ▶ 用LSTM实现Seq2Seq

通用LSTM的结构



$$\bar{z}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1} + \mathbf{b}_z$$

$$\mathbf{z}^t = g(\bar{z}^t)$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block input

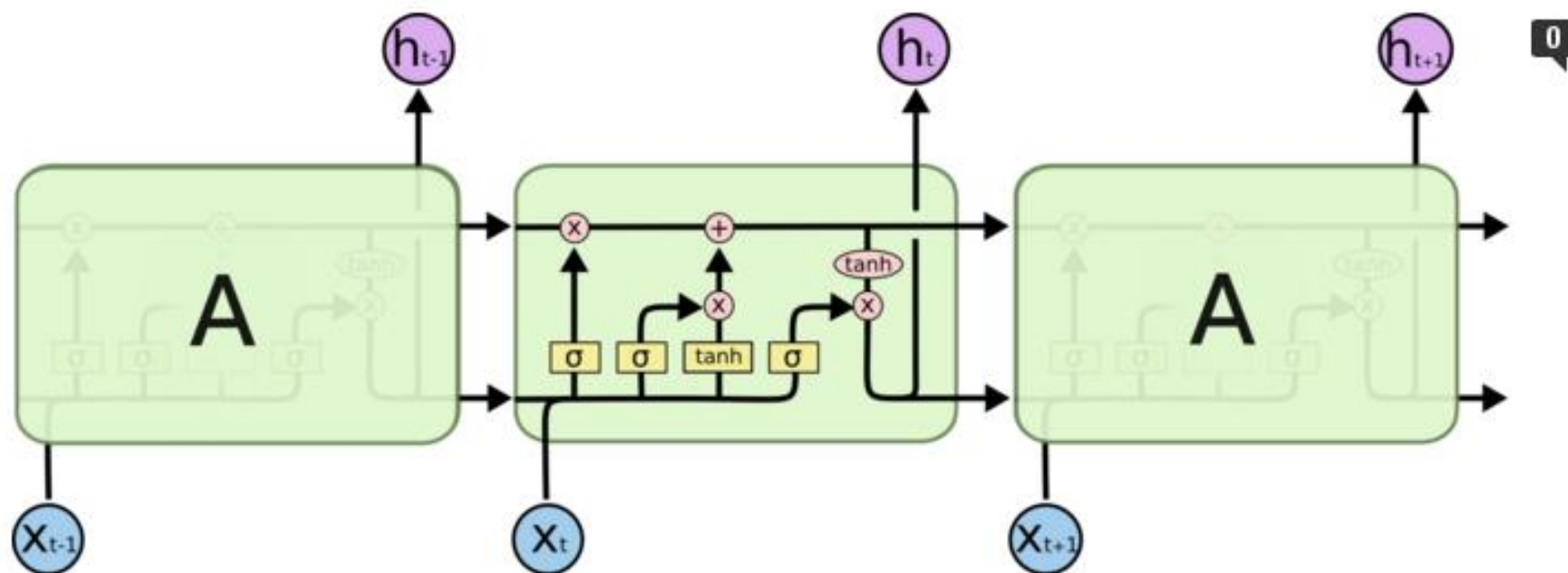
input gate

forget gate

cell

output gate

block output



常见LSTM结构

$$\bar{z}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{z}^t)$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block input

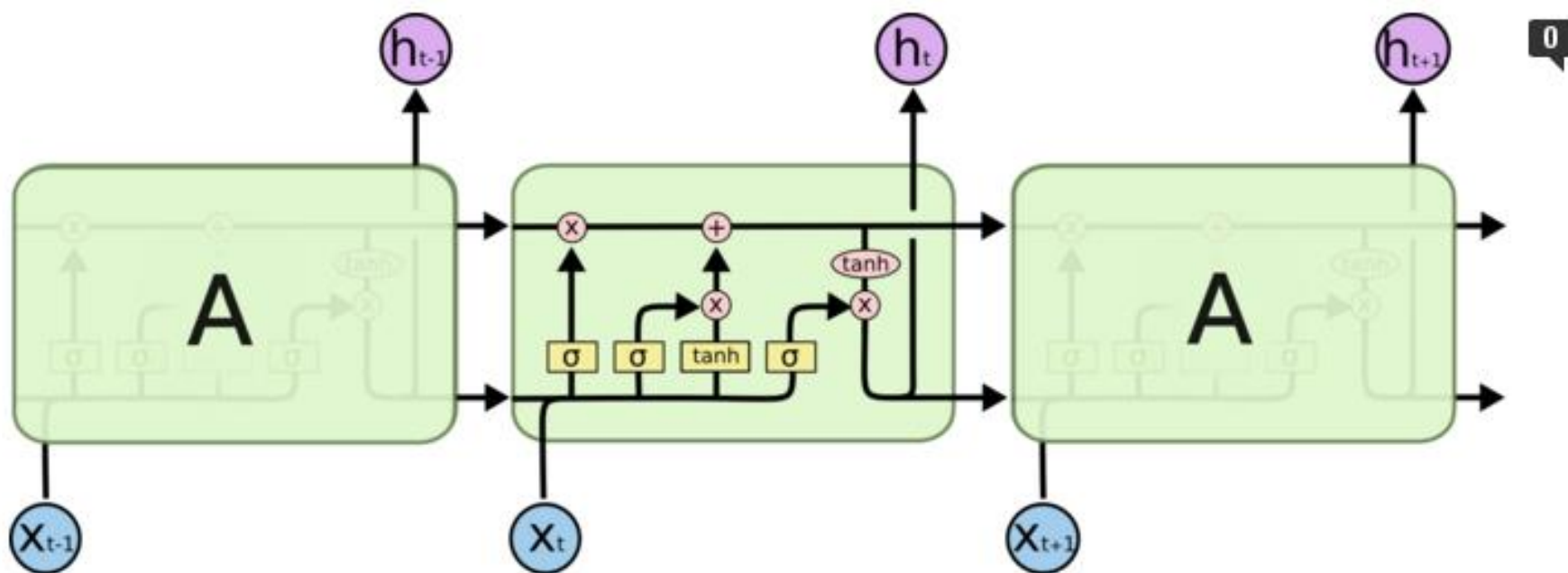
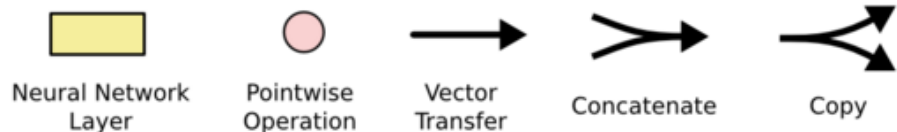
input gate

forget gate

cell

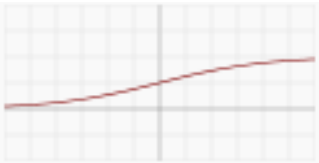
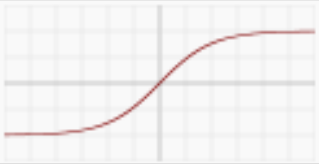
output gate

block output



► LSTM结构

- Gate是一种限制，代表一种选择（0或1）
- 不过为了保证可导，所以设置为Sigmoid函数 $\sigma(x)$

| | | | |
|---------|--|---|--------------------------|
| Sigmoid |  | $f(x) = \frac{1}{1 + e^{-x}}$ | $f'(x) = f(x)(1 - f(x))$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ |

► LSTM结构

- Block input
- Input gate
- Forget gate
- Output gate
- Block output

- 输入为3gate , 1input
- 存储为1memory
- 输出为1output

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block input

input gate

forget gate

cell

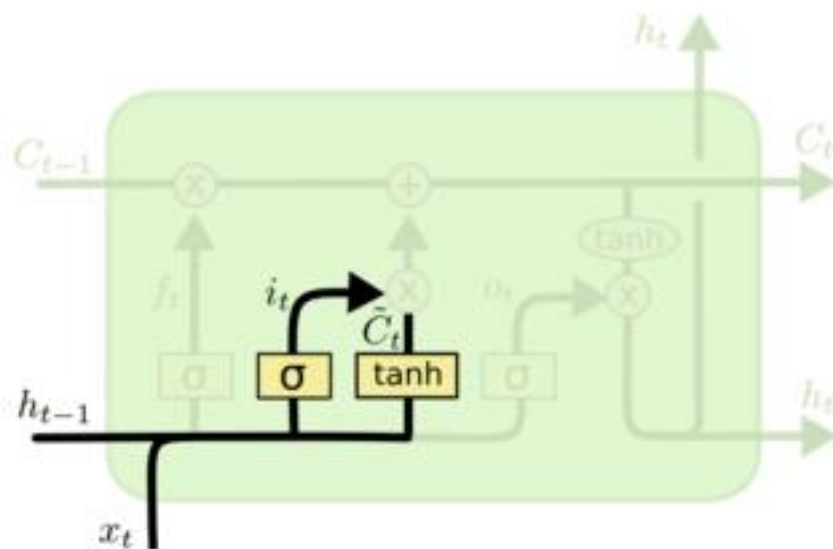
output gate

block output

► LSTM结构

► Block input

- 代表输入信息
- 结合上一时间的输出和当前输入
- RNN中的原始结构



$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block output

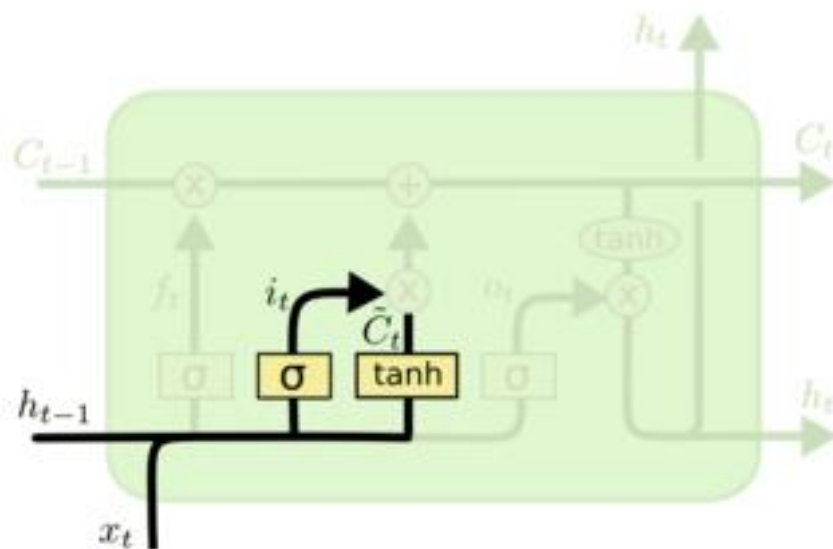
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t])$$

► LSTM结构

► Input gate

- 是否需要加入新信息？
- （比如情感分析中的'the'就可以直接无视了）



$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block output

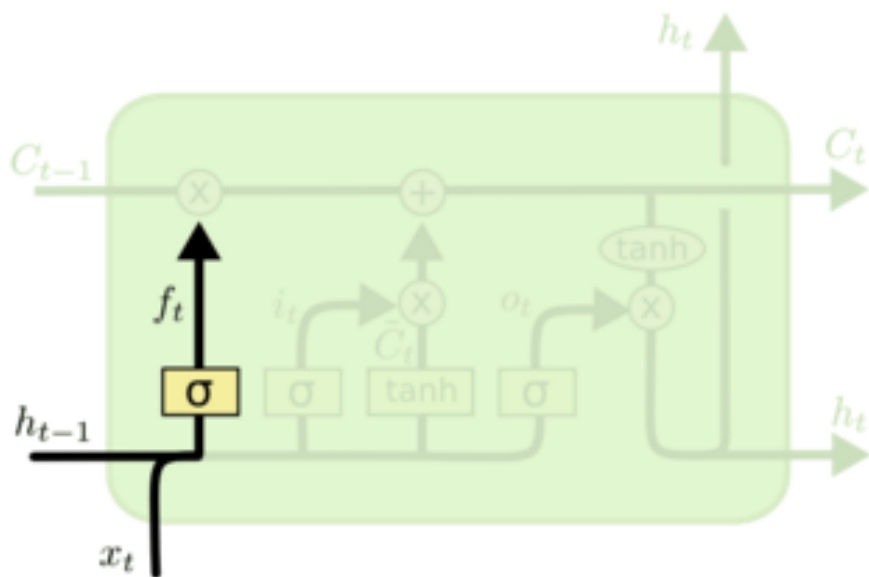
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t])$$

► LSTM结构

► Forget gate

- 是否需要记住/忘记之前的信息？
- （比如在情感分析中，看到'but'这种转折词就忘记之前的内容，看到'the'这类无关的词就记住）



$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

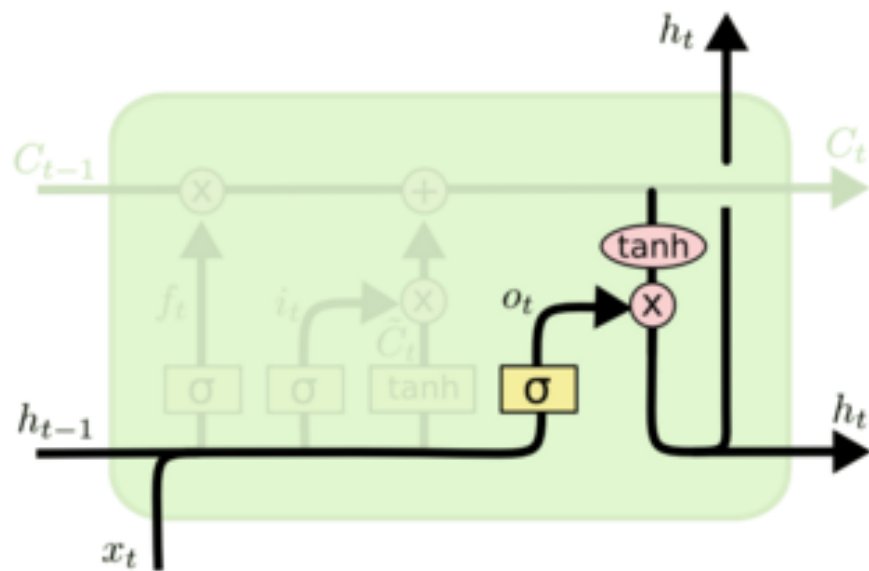
block output

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$$

► LSTM结构

► Output gate

- 选取特定记忆影响这次的结果和控制下次的gate



$$\bar{z}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{z}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block output

$$o_t = \sigma(W_o [h_{t-1}, x_t])$$

$$h_t = o_t * \tanh(C_t)$$

► LSTM结构

► Cell

- 记忆即存储
- 抹去一些记忆后加入新信息

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

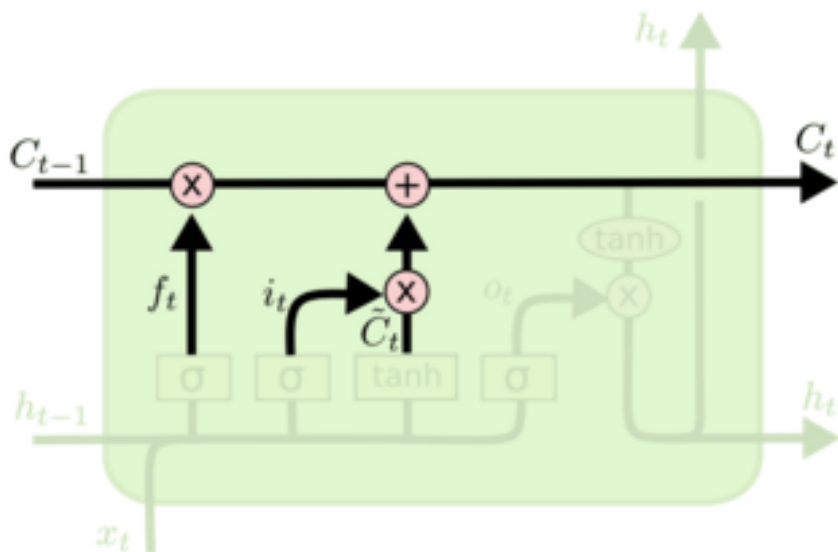
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block output



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

► LSTM结构

► Block output

► 利用记忆产生行为
以及改变下一次的控制

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

block input

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

input gate

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

forget gate

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

cell

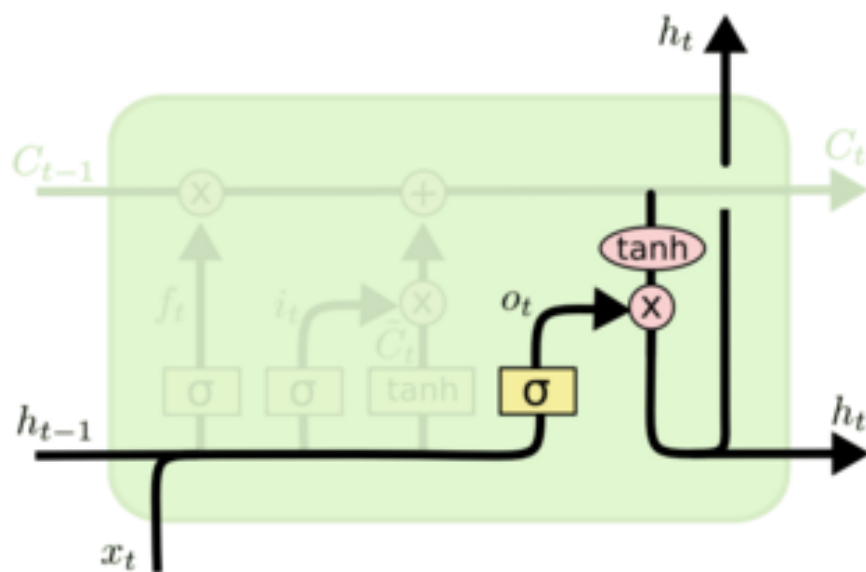
$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

output gate

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block output



$$o_t = \sigma(W_o [h_{t-1}, x_t])$$

$$h_t = o_t * \tanh(C_t)$$

► LSTM结构总结

- 选择性输入
- 选择性记忆
- 记忆和输入合并
- 记忆选择性输出

$$\bar{\mathbf{z}}^t = \mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1}$$

$$\mathbf{z}^t = g(\bar{\mathbf{z}}^t)$$

$$\bar{\mathbf{i}}^t = \mathbf{W}_i \mathbf{x}^t + \mathbf{R}_i \mathbf{y}^{t-1}$$

$$\mathbf{i}^t = \sigma(\bar{\mathbf{i}}^t)$$

$$\bar{\mathbf{f}}^t = \mathbf{W}_f \mathbf{x}^t + \mathbf{R}_f \mathbf{y}^{t-1}$$

$$\mathbf{f}^t = \sigma(\bar{\mathbf{f}}^t)$$

$$\mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t$$

$$\bar{\mathbf{o}}^t = \mathbf{W}_o \mathbf{x}^t + \mathbf{R}_o \mathbf{y}^{t-1}$$

$$\mathbf{o}^t = \sigma(\bar{\mathbf{o}}^t)$$

$$\mathbf{y}^t = h(\mathbf{c}^t) \odot \mathbf{o}^t$$

block input

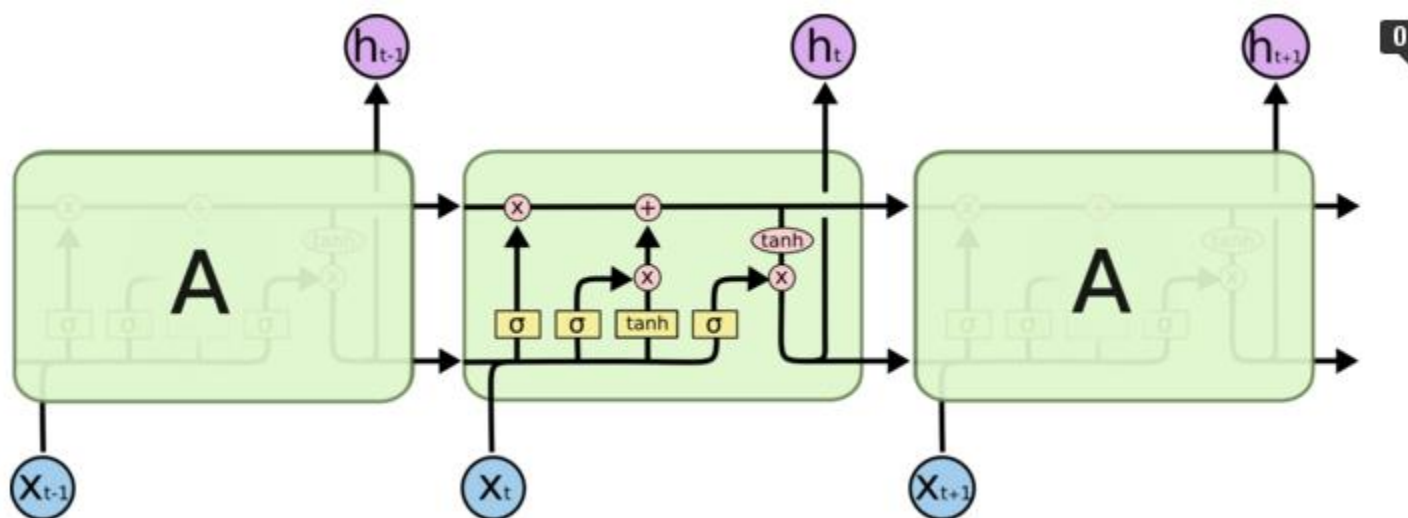
input gate

forget gate

cell

output gate

block output



0

▶ LSTM思想

- ▶ 记忆 Memory
- ▶ 选择 Gate

▶ LSTM思想

▶ 记忆 Memory

- ▶ RNN不具备很强的记忆性，所以LSTM弥补了这一点
- ▶ 利用强制的记忆结构
- ▶ 实际上RNN在经过适当的初始化(Recurrent矩阵类似I矩阵)后，也可以达到类似效果
 - ▶ [2015] A Simple Way to Initialize Recurrent Networks of Rectified Linear Units

► LSTM思想

► 选择 Gate

- LSTM让RNN有了选择信息的能力，更容易排除冗余信息，找到时序依赖关系
- 实验表明，情感分析中'the', 'an', 'he', 'she'的input gate基本都为0，而强情感词语如'good', 'bad', 'excellent'的input gate基本都为1

- ▶ LSTM实现Seq2Seq

- ▶ 为什么LSTM这么火？

- ▶ 处理Sequence序列之间映射关系的强大工具

- ▶ 句子

- ▶ 语音

- ▶ 代码

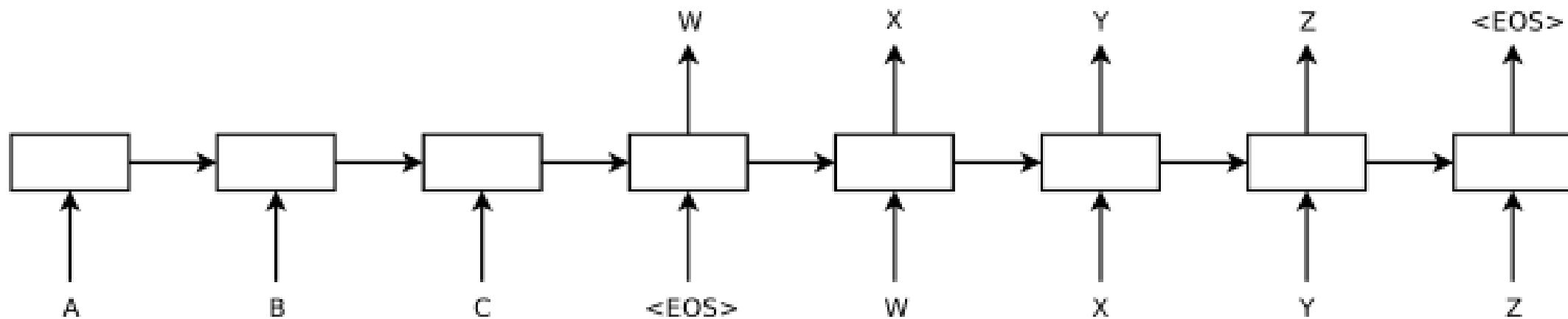
- ▶ 视频

- ▶ 轨迹

- ▶(只要是Sequence都能试一试)

► LSTM实现Seq2Seq

- [2014] Sequence-to-sequence-learning-with-neural-networks



优化

► Bidirectional LSTM(双向LSTM)

- 单向LSTM不能很好处理未来信息对当前预测的影响
- 双向LSTM可以缓解这一问题，实际中效果也确实好
- 就是暴力把正向和逆向的输出向量拼在一起

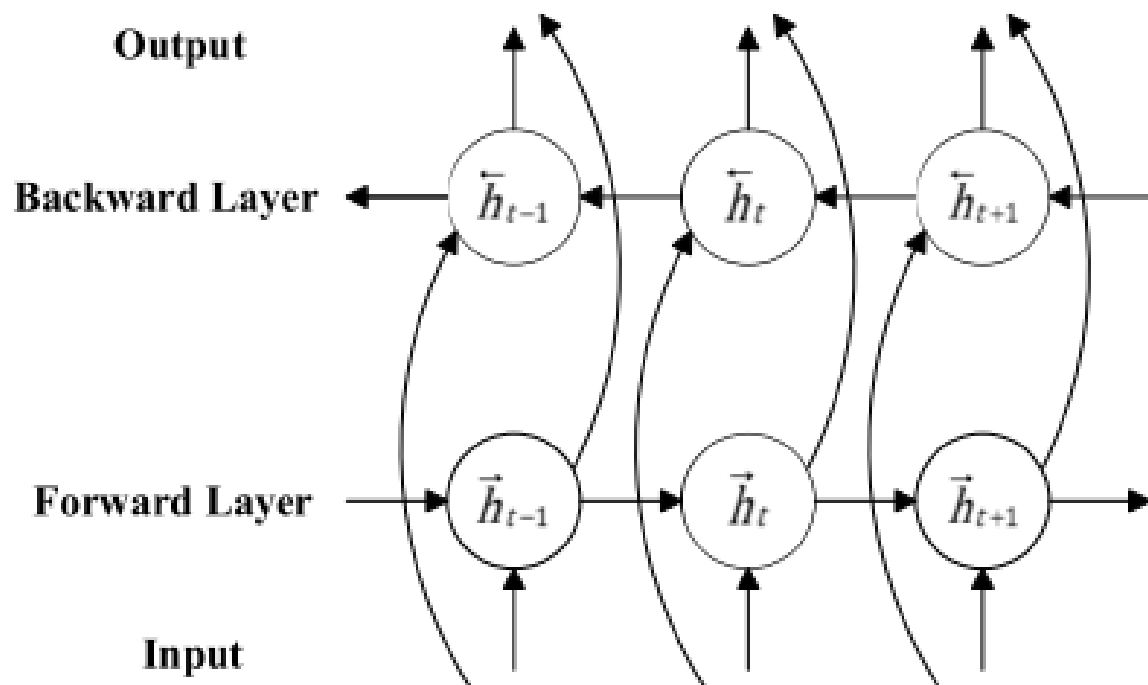


Fig.2. Bidirectional RNN

► Gated Recurrent Unit(GRU)

► LSTM训练非常慢，GRU对此做了优化

► 减少一个Gate，速度快25%，
效果没有明显下降

$$\tilde{h}_t^j = \tanh(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))^j,$$

$$z_t^j = \sigma(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})^j.$$

$$r_t^j = \sigma(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1})^j.$$

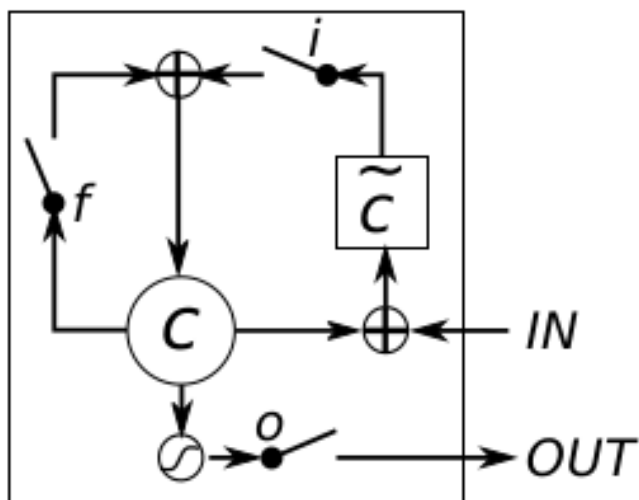
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j\tilde{h}_t^j,$$

block input

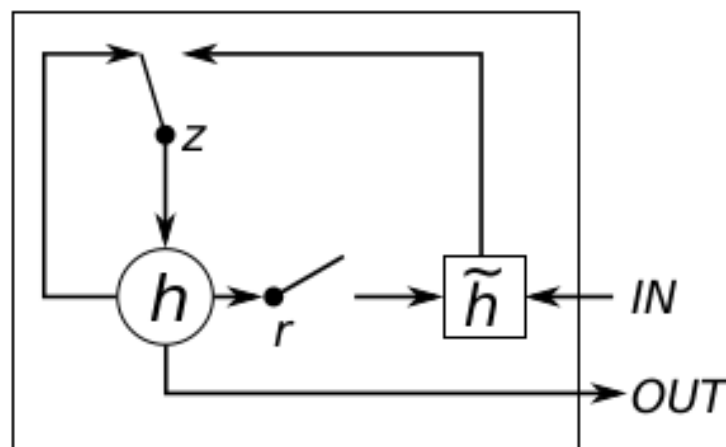
update gate

reset gate

output/memory



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

► Gated Recurrent Unit(GRU)

- 主要合并了input gate和forget gate
- 另外还合并了隐层和记忆

$$\tilde{h}_t^j = \tanh(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}))^j,$$

$$z_t^j = \sigma(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1})^j.$$

$$r_t^j = \sigma(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1})^j.$$

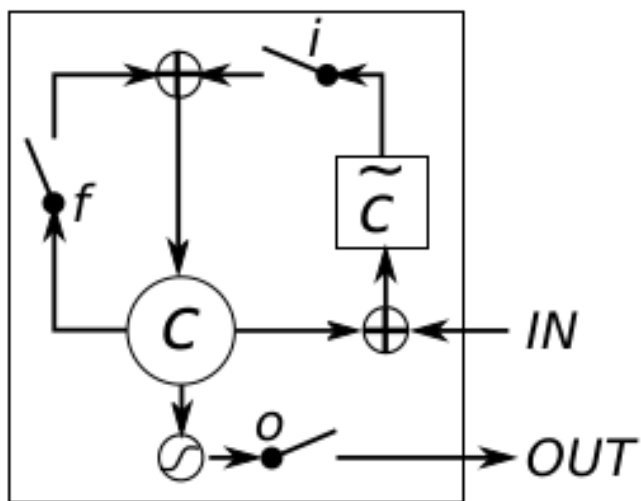
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j\tilde{h}_t^j,$$

block input

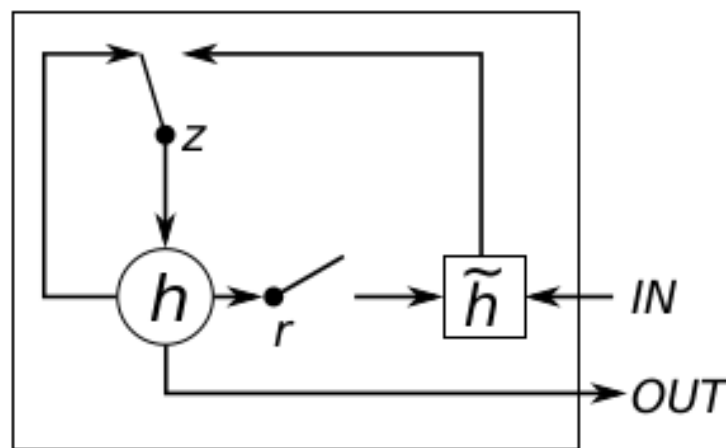
update gate

reset gate

output/memory



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

► Multilayer LSTM(多层次LSTM)

- 对深度的无限追求 (RNN本身横着就够深了, 现在竖着也要变深)
- 正常2~3层都会有提升, 4层以上就不好说了

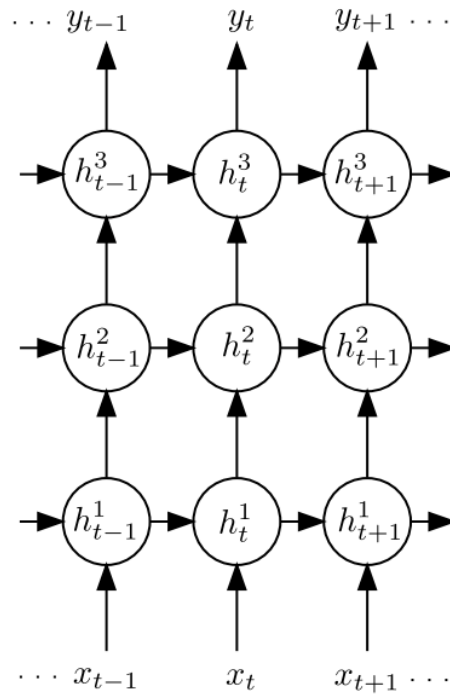
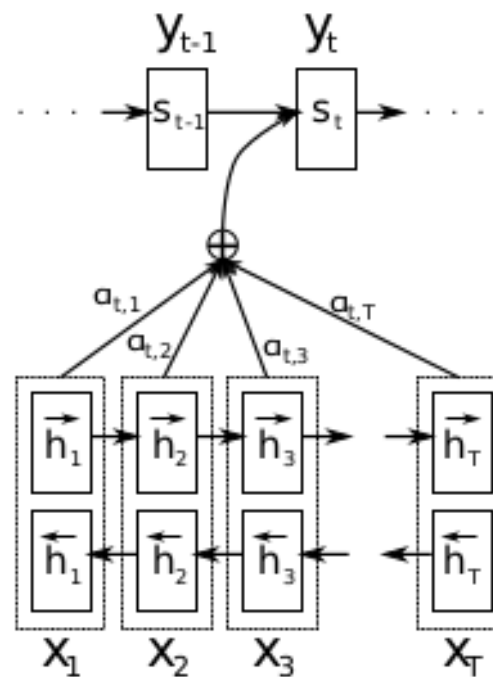
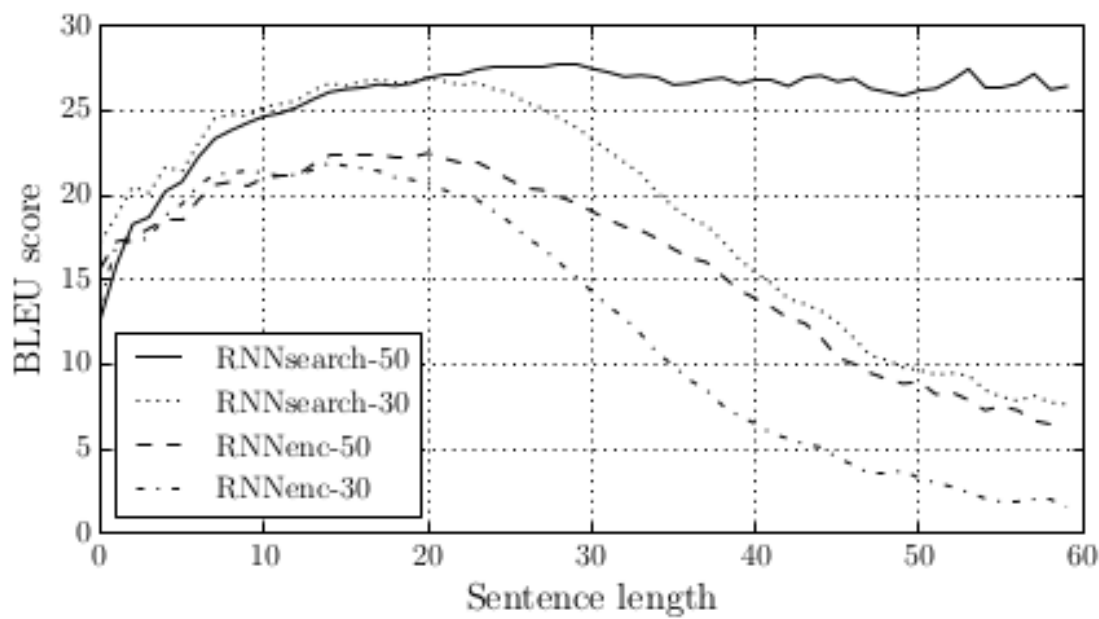


Fig. 3. Deep Recurrent Neural Network

► Attention-based LSTM (基于注意力的LSTM)

- LSTM记忆力不好
- 训练用30长度的句子，测试用50长度的，效果就不行
- Attention-based把句子全给LSTM连上，让他挑看着顺眼的最匹配的



The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern, layered effect. The shapes are concentrated on the left and right sides, leaving a large white central area.

应用

- ▶ Sentiment classification情感分析
 - ▶ [2015] Improved semantic representations from tree-structured long short-term memory networks
- ▶ Language Modeling语言模型
 - ▶ [2015] Character-Aware Neural Language Models
- ▶ Sequence Generation 序列生成
 - ▶ <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- ▶ Speech Recognition 语音识别
 - ▶ [2013] SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS
- ▶ Machine Translation 机器翻译
 - ▶ [2014] Neural machine translation by jointly learning
- ▶ Question Answering 问答系统
 - ▶ [2015] Neural Responding Machine for Short-Text Conversation

► 我说这是LSTM生成的你信吗？

"The surprised in investors weren't going to raise money. I'm not the company with the time there are all interesting quickly, don't have to get off the same programmers. There's a super-angel round fundraising, why do you can do. If you have a different physical investment are become in people who reduced in a startup with the way to argument the acquirer could see them just that you're also the founders will part of users' affords that and an alternation to the idea. [2] Don't work at first member to see the way kids will seem in advance of a bad successful startup. And if you have to act the big company too."

我说这是LSTM生成的你信吗？

For $\bigoplus_{n=1, \dots, m}$ where $\mathcal{L}_{m, \bullet} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ?? . Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X, x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X, x'} \rightarrow \mathcal{O}'_{X', x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S, s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{opp}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ?? . It may replace S by $X_{spaces, \acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ?? . Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1, \dots, n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X, \dots, 0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{I}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{I}_{n,0} \circ \bar{A}_2$ works.

Lemma 0.3. In Situation ?? . Hence we may assume $\mathfrak{q}' = 0$.

Proof. We will use the property we see that \mathfrak{p} is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

► 我说这是LSTM生成的你信吗？

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 1;
    return segtable;
}
```

- ▶ 但这还远远没有结束.....
- ▶ Sorting 排序!?
 - ▶ [2014] Neural Turing Machine
- ▶ Taxi Destination Prediction的士目的预测!?
 - ▶ [2015] Artificial Neural Networks Applied to Taxi Destination Prediction
- ▶ Python Interpreter Python解释器!?
 - ▶ [2014] Learn to execute

实现细节

► 一些小Trick

- 初始化矩阵都在 $[-0.01, 0.01]$ 比较好，均匀分布
- Peephole和bias不要比较好（常用版本LSTM）
- GRU很好用
- 用GPU训练，多GPU更好
- Learning rate学习率开始可以比较大，之后可以变小
- 用Gradient Clipping防止梯度爆炸
- 在适当的地方放Dropout防止Overfitting
- Mini-batch，不然GPU没法加速，batch大小很讲究
- 使用Momentum加速收敛

► 梯度更新方法

► 不同task不同，大概有以下几种

► 裸的SGD

► 前期learning rate很大，过一段时间后逐渐减少

► AdaGrad

► Rmsprop

► AdaDelta （现在貌似很流行）

工具库

► Theano

- Python前端，C++实现
- 自动求导
- 将图结构编译成GPU流，存储时间都有优化

► Torch7

- Lua前端，C/CUDA实现
- 嵌入式支持得很好

► TensorFlow

- Google第二代深度学习系统

► MXNet

- 结合命令式语言（如Caffe）和符号式语言（如Theano）
- 对多GPU有一定支持

.....

- ▶ 后端库（想自己实现的请看这里）
- ▶ CUDA
 - ▶ 非常容易上手的GPU编程语言
 - ▶ 然而Parallel programming is easy as long as you don't care about performance
- ▶ CUBLAS
 - ▶ 支持大部分矩阵/向量操作，已经做过大量底层优化
- ▶ CUDNN
 - ▶ 支持神经网络的常用操作，已经做过大量底层优化
- ▶ CUSPARSE
 - ▶ 支持一部分稀疏矩阵/向量操作，已经做过大量底层优化

- ▶ LSTM已经有大量现成实现
 - ▶ C++
 - ▶ Python
 - ▶ Matlab
 -
- ▶ 所以没事千万别自己实现
- ▶ (用Theano只要5行我会乱说?)

► 当前工具库的评价

- [2015] Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning
- 目前还没有通用的支持多机多GPU的框架出现

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect. The shapes are concentrated on the left and right sides of the frame, leaving a central white area for the text.

展望

- ▶ LSTM存在的问题
 - ▶ 记忆力不足
 - ▶ 只能处理序列映射
 - ▶ 理论基础仍缺乏，调参手段一大堆
 - ▶ 结合更多类型数据

- ▶ 继LSTM之后又一火爆话题
- ▶ Attention-based mechanism 注意力机制
 - ▶ [2014] Neural machine translation by jointly learning
 - ▶ 论如何用一个模型达到Google Translation的效果

参考文献

参考文献

- ▶ [1] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 1997.
- ▶ [2] Sutskever, I., Vinyals, O., and Le, Q. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS 2014), 2014.
- ▶ [3] Greff, Klaus, Srivastava, Rupesh Kumar, Koutník, Jan, Steunebrink, Bas R, and Schmidhuber, Jurgen. Lstm: A " search space odyssey. arXiv preprint arXiv:1503.04069, 2015.
- ▶ [4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
- ▶ [5] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

参考文献

- ▶ [6] W. Zaremba and I. Sutskever. Learning to execute. arXiv preprint arXiv:1410.4615, 2014.
- ▶ [7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS Deep Learning Workshop, 2014.
- ▶ [8] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. ACL, 2015.
- ▶ [9] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. arXiv preprint arXiv:1211.5063, 2012.
- ▶ [10] Le, Q. V., Jaitly, N., & Hinton, G. E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. arXiv:1504.00941 [cs], 2015.

参考文献

- ▶ [11] Shang, L., Lu, Z., and Li, H. Neural responding machine for short-text conversation. In Proceedings of ACL, 2015.
- ▶ [12] Kim, Yoon, Jernite, Yacine, Sontag, David, and Rush, Alexander M. Character-aware neural language models. CoRR, abs/1508.06615, 2015.
- ▶ [13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In Proc. ICASSP, 2013.
- ▶ [14] Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. ArXiv preprint arXiv:1508.00021, 2015.
- ▶ [15] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, Mohak Shah. Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning. ArXiv preprint arXiv:1511.06435v2, 2015.

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect. The shapes are layered, with some appearing more prominent than others, and they extend towards the corners of the frame.

Q&A