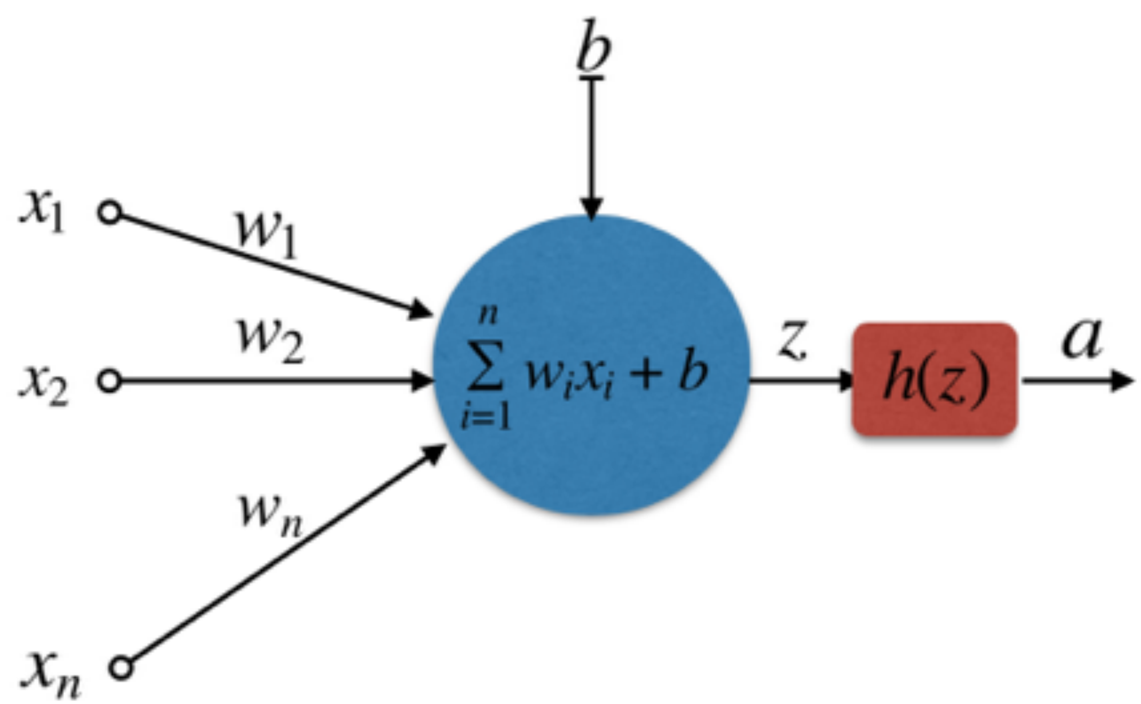
递归网络简介

A simple introduction about Recurrent Neural Network

Outline

- 复习：神经网络，全连接网络，卷积网络
- RNN网络介绍
- RNN网络的训练，BPTT算法
- LSTM介绍
- 基于Torch的RNN实验

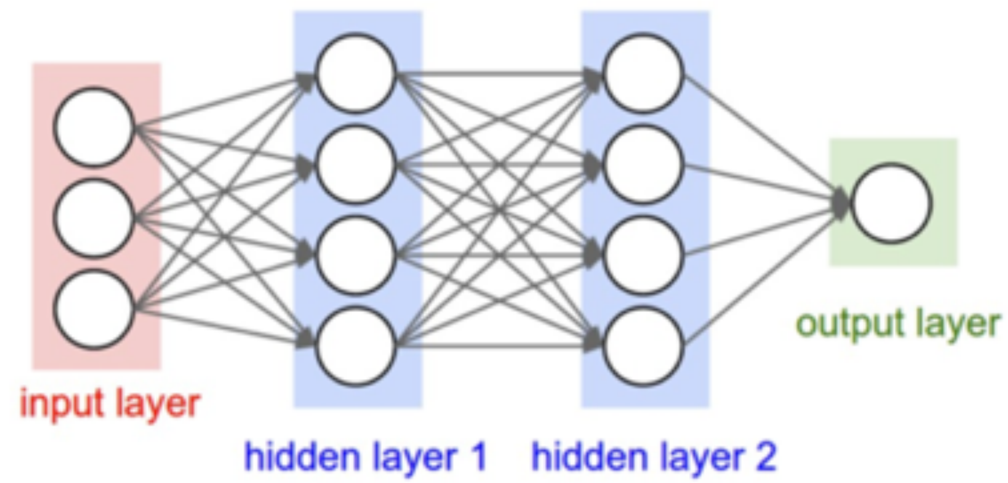
Recall：神经单元

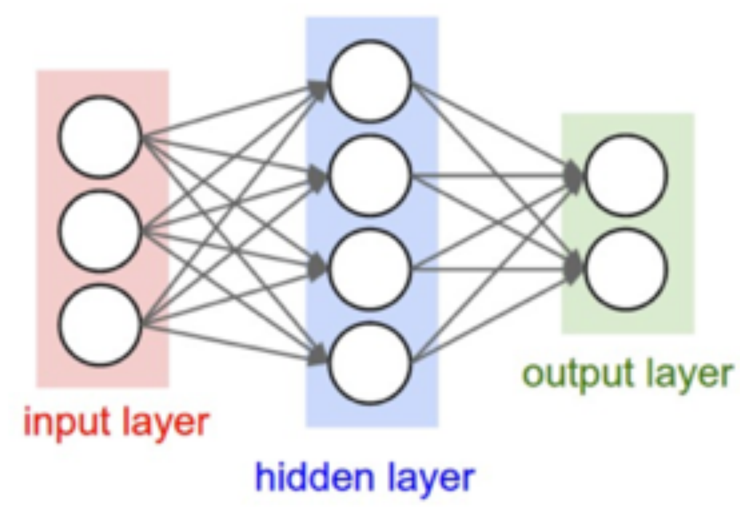


一个独立神经元包含: $\{n \ w_i \ b \ h\}$

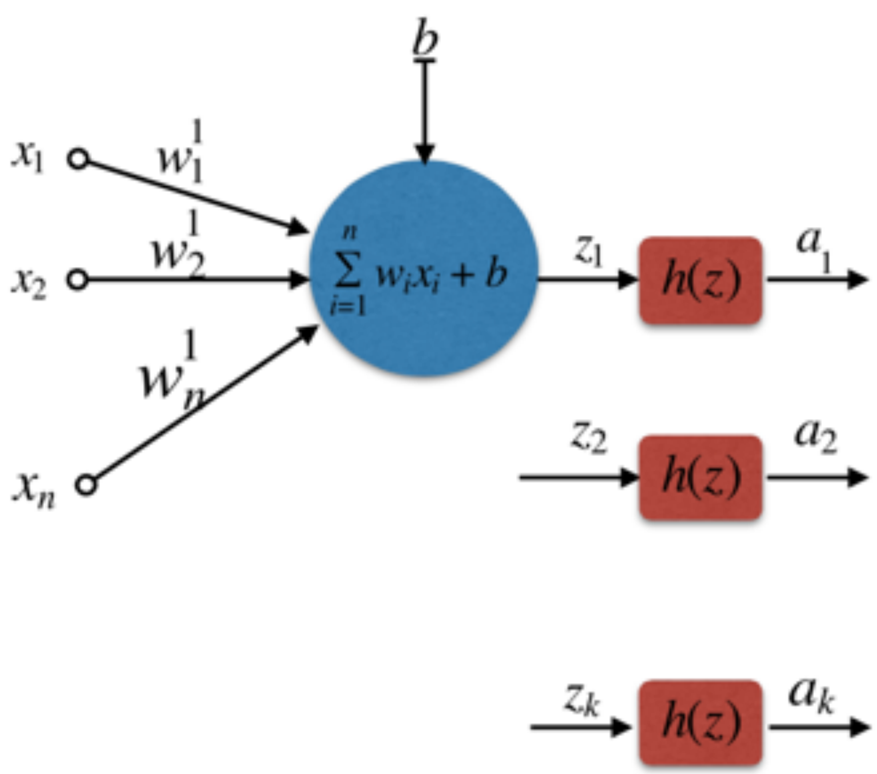
Recall：全连接网络

- 相同结构的神经元构成神经网络中的层
- 中间层又叫做隐含层、输出层也可以包含多个神经元
- 全连接结构：每一层的单元输出构成一个向量，这个向量输入到下一层的每个神经元当中
- 拥有足够单元的单隐层网络可以逼近任意的函数





Recall：网络训练(1)



$\frac{dl}{da_i}$ 已知，由上一层的dl/dx计算得到

$\frac{dl}{dz_i} = \frac{dl}{da_i}\frac{da_i}{dz_i}$ 链式规则，h(z)函数求导得到

$\frac{dl}{dw_j^i} = \sum_{m=1}^{k} \frac{dl}{dz_m}\frac{dz_m}{dw_j^i} = \frac{dl}{dz_i} x_j$ 应用链式规则

$\frac{dl}{dx_j} = \sum_{i=1}^{k} \frac{dl}{dz_i}\frac{dz_i}{dx_j} = \sum_{i=1}^{k} \frac{dl}{dz_i} w_j^i$ 求解上一层准备

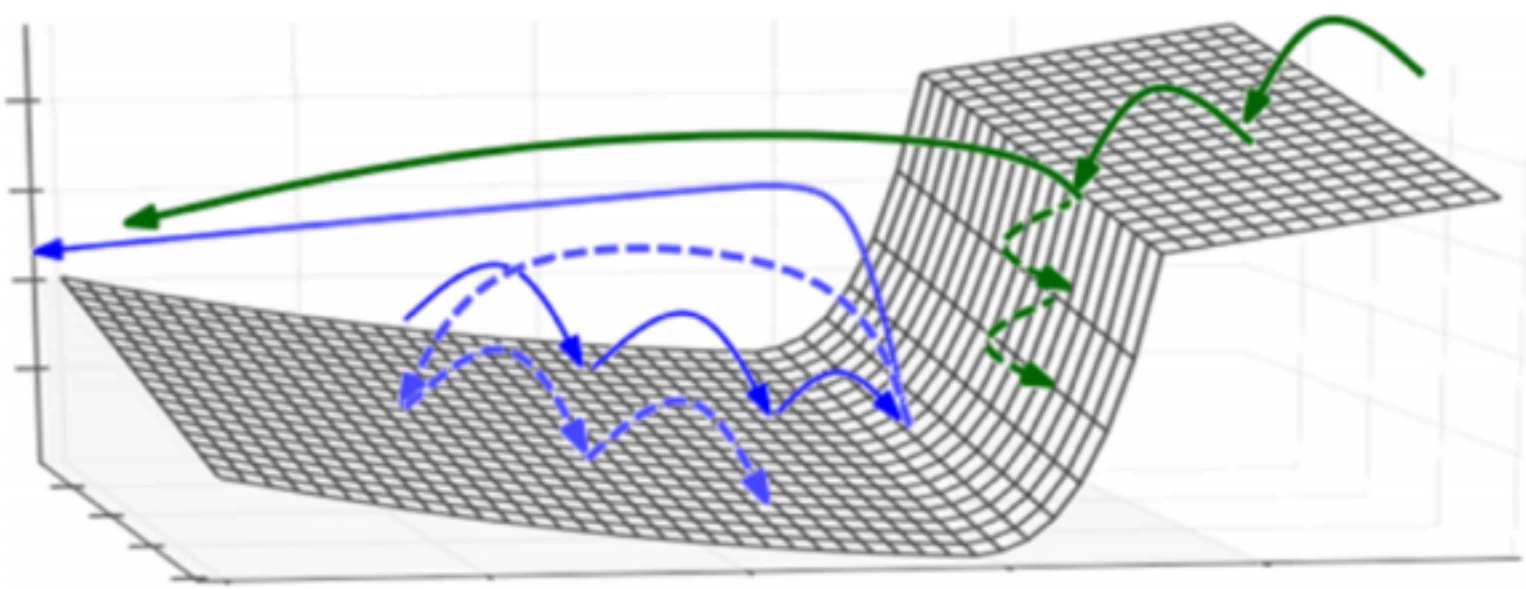
Chain Ruler

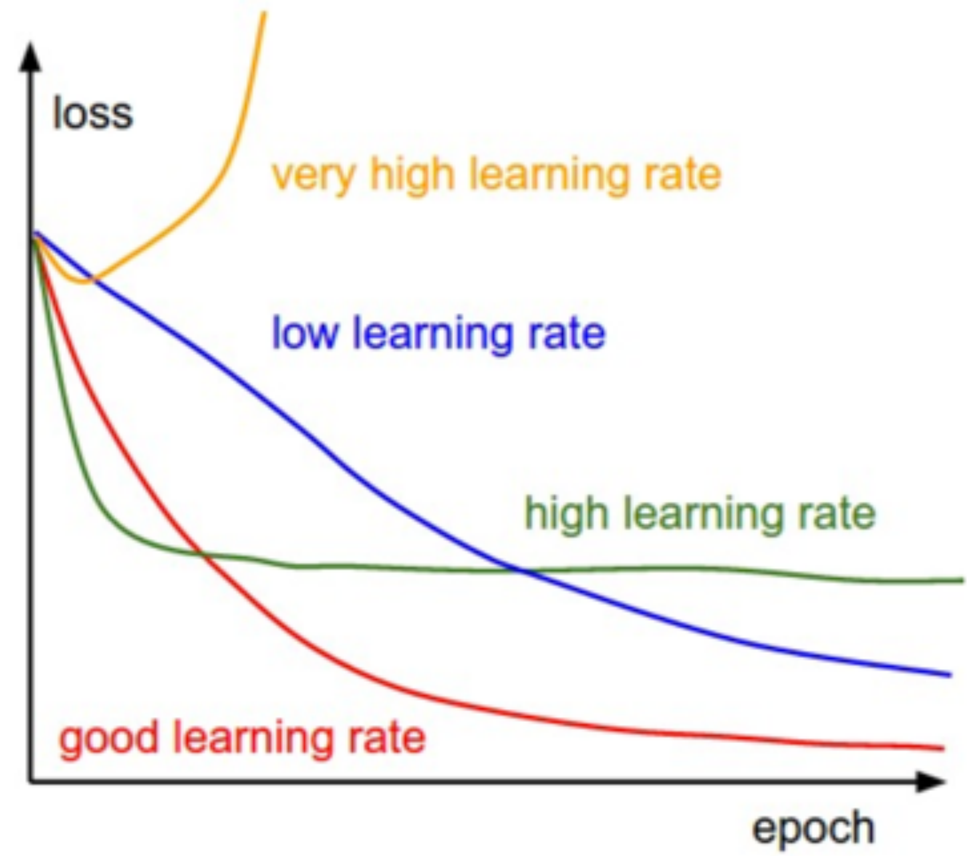
$$\frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt}$$

$$\frac{\partial y}{\partial x_i} = \sum_{\ell=1}^{m} \frac{\partial y}{\partial u_\ell}\frac{\partial u_\ell}{\partial x_i}.$$

Backpropagation算法

Recall：网络训练(2)

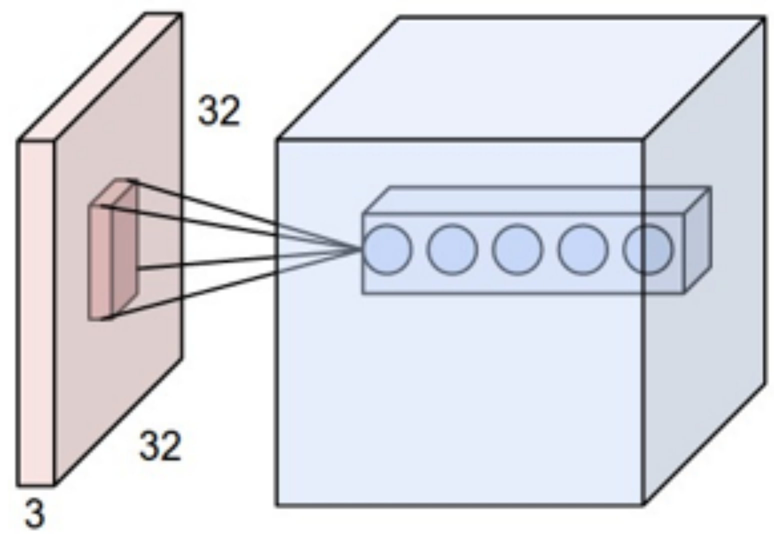


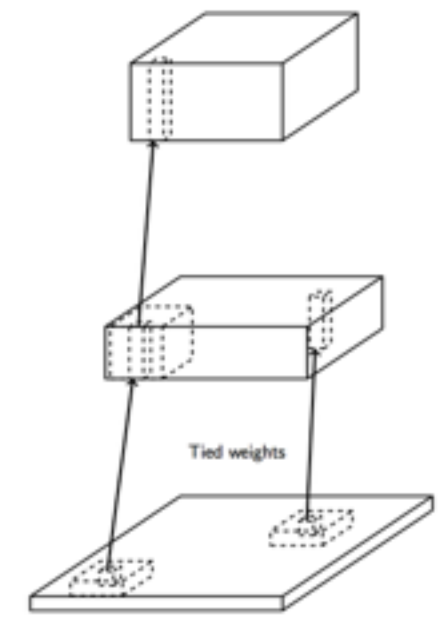


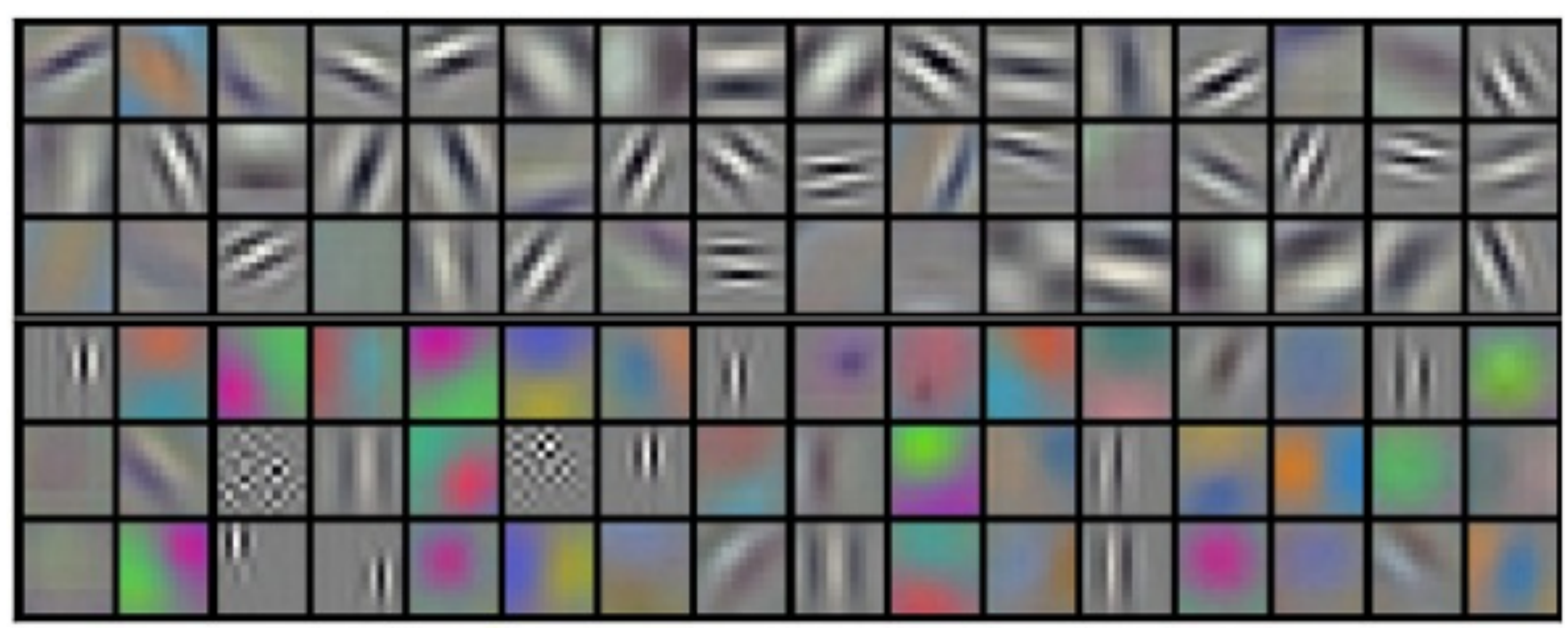
Stochastic (minibatch) gradient descent 算法

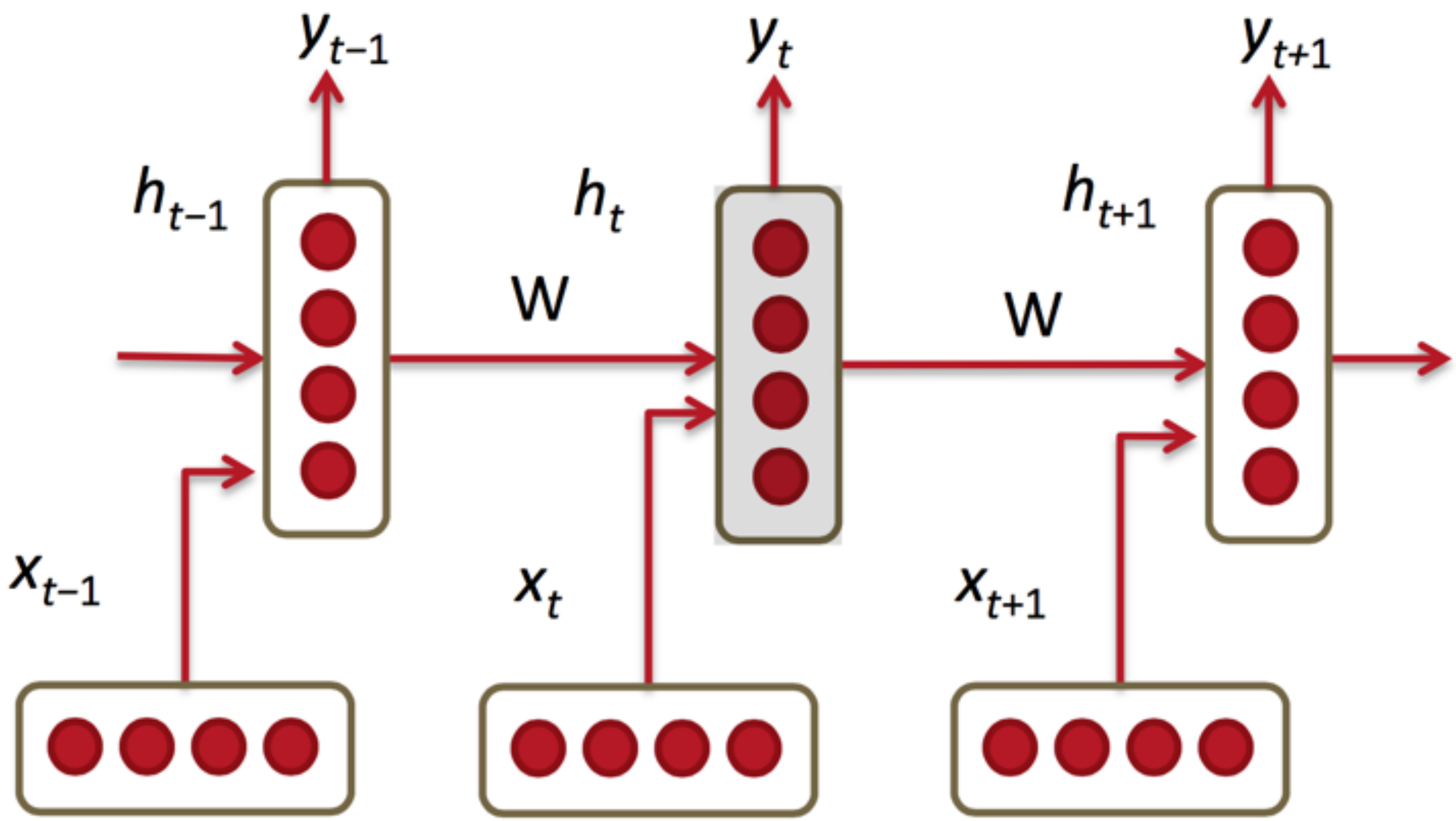
Recall：卷积网络

- 局部连接结构:
  - ReLU
  - Pooling
  - DropOut
- CNN主要作用是特征表达









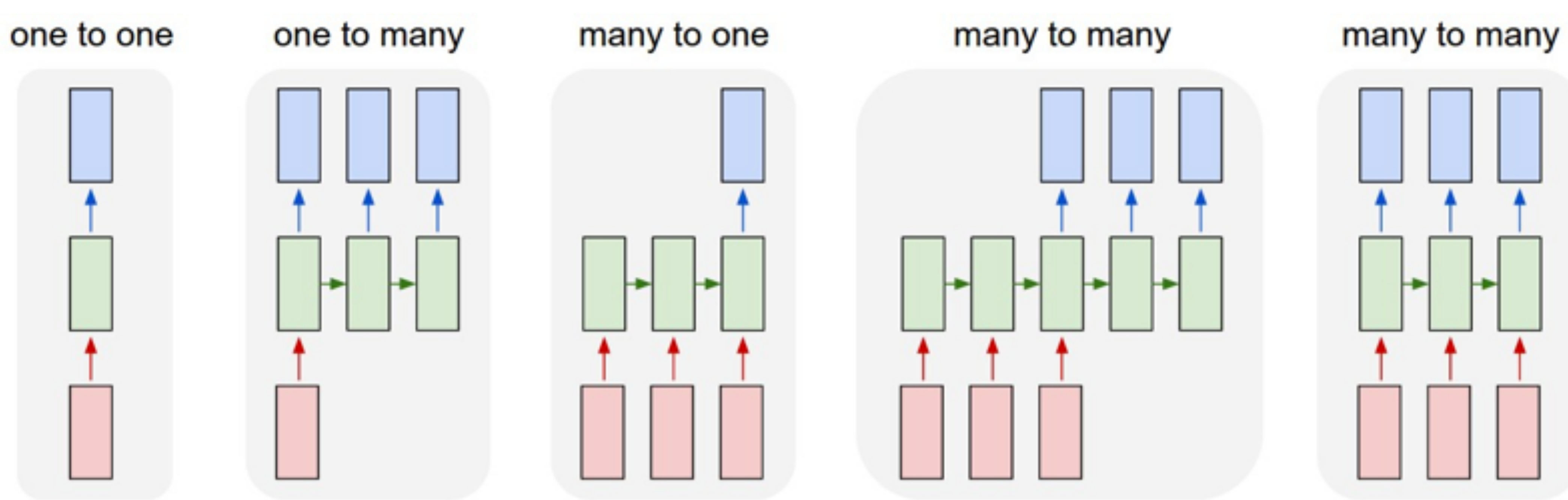
递归神经网络

Recurrent Neural Network

状态和模型

- IID 数据
  - 分类问题
  - 回归问题
  - 特征表达
- 大部分数据都不满足IID
  - 序列分析（Tagging, Annotation)
  - 序列生成，如语言翻译，自动文本生成
  - 内容提取（Content Extraction），如图像描述

序列样本



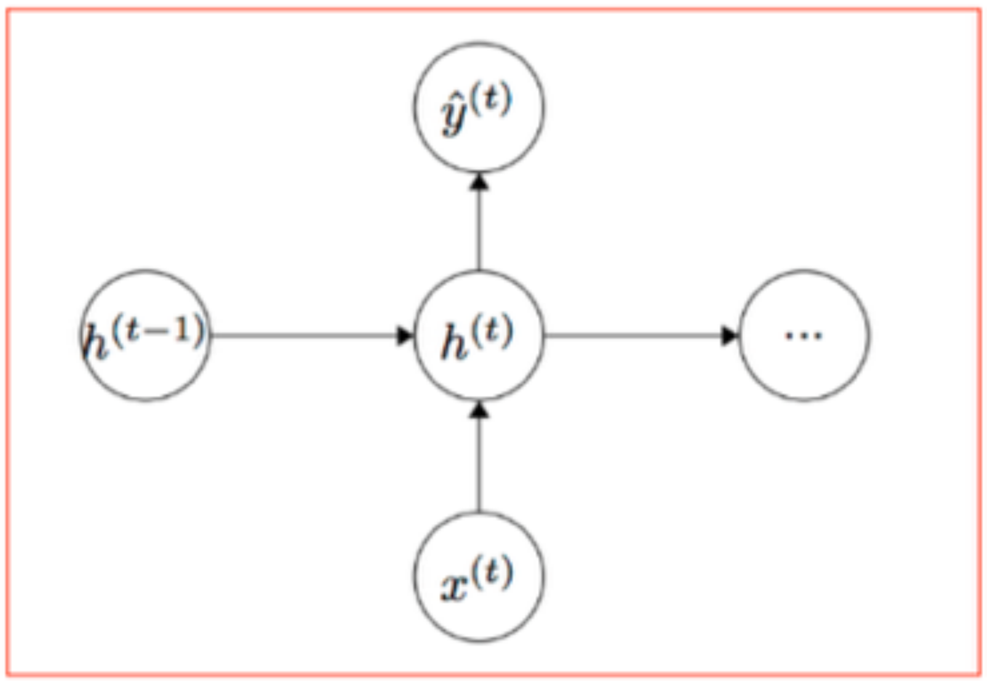
- RNN不仅仅能够处理序列输出，也能得到序列输出，这里序列指的是向量的序列。
- RNN学习出来的是程序，不是函数

序列预测

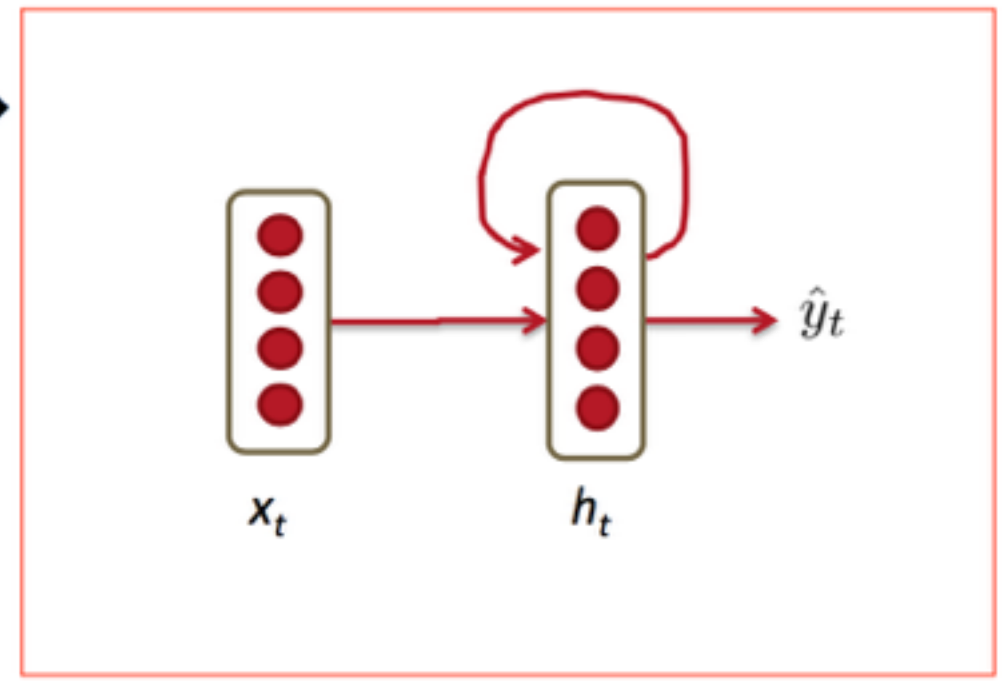
- 输入的是时间变化向量序列： x_{t-2}，x_{t-1}，x_t，x_{t+1}，x_{t+2}
- 在t时刻通过模型来估计 $x_{t+1} = f(x_t, \ldots, x_{t-\tau})$
- 问题：
  - 对内部状态难以建模和观察
  - 对长时间范围的场景(context)难以建模和观察
- 解决方案：引入内部隐含状态变量

$$x_{t+1} = f(x_t, \ldots, x_{t-\tau}, z_t, \ldots, z_{t-\tau})$$
$$z_{t+1} = g(x_{t+1}, \ldots, x_{t-\tau}, z_t, \ldots, z_{t-\tau})$$

序列预测模型



↔

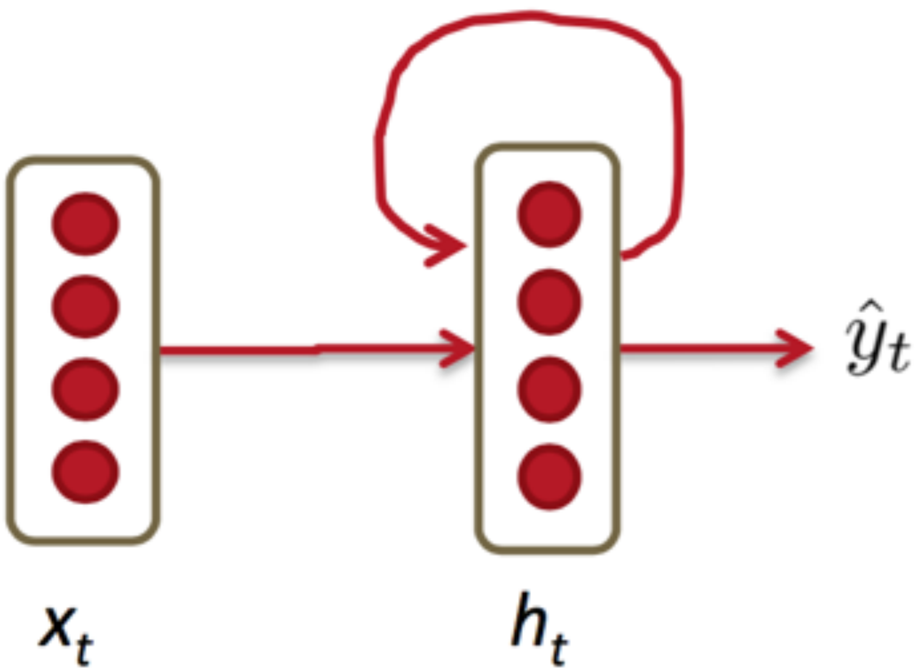


- 输入离散列序列 $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$
- 在时间t的更新计算:

$$
\begin{aligned}
h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\
\hat{y}_t &= \operatorname{softmax}\left(W^{(S)} h_t\right)
\end{aligned}
$$

- 预测计算 $\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$

序列预测模型



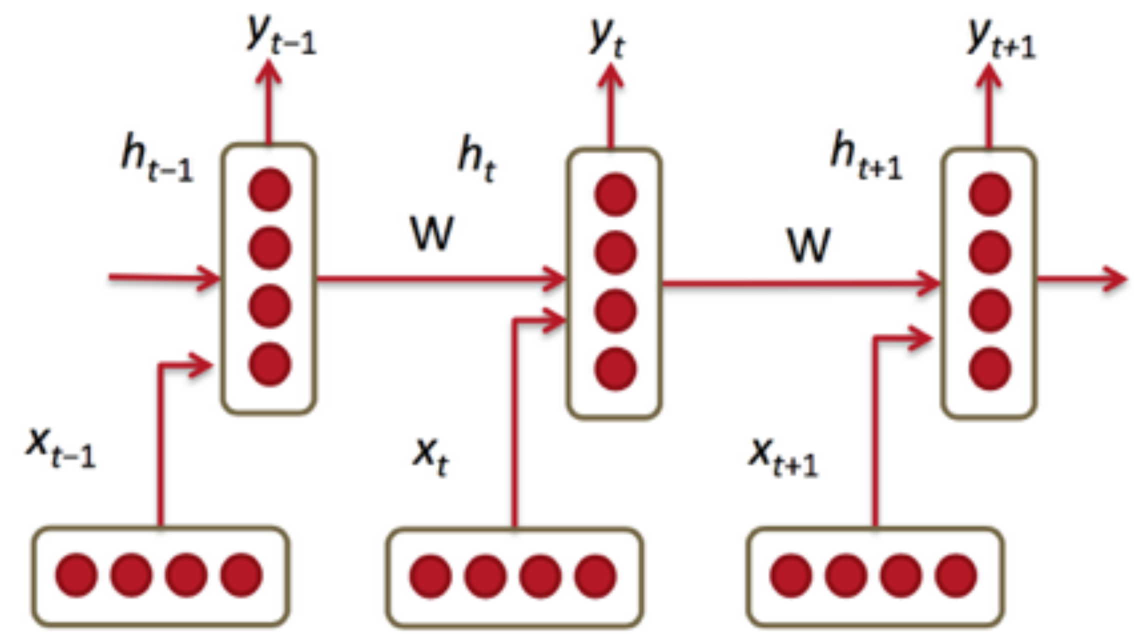
$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}\right)$$
$$\hat{y}_t = \mathrm{softmax}\left(W^{(S)}h_t\right)$$
$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$

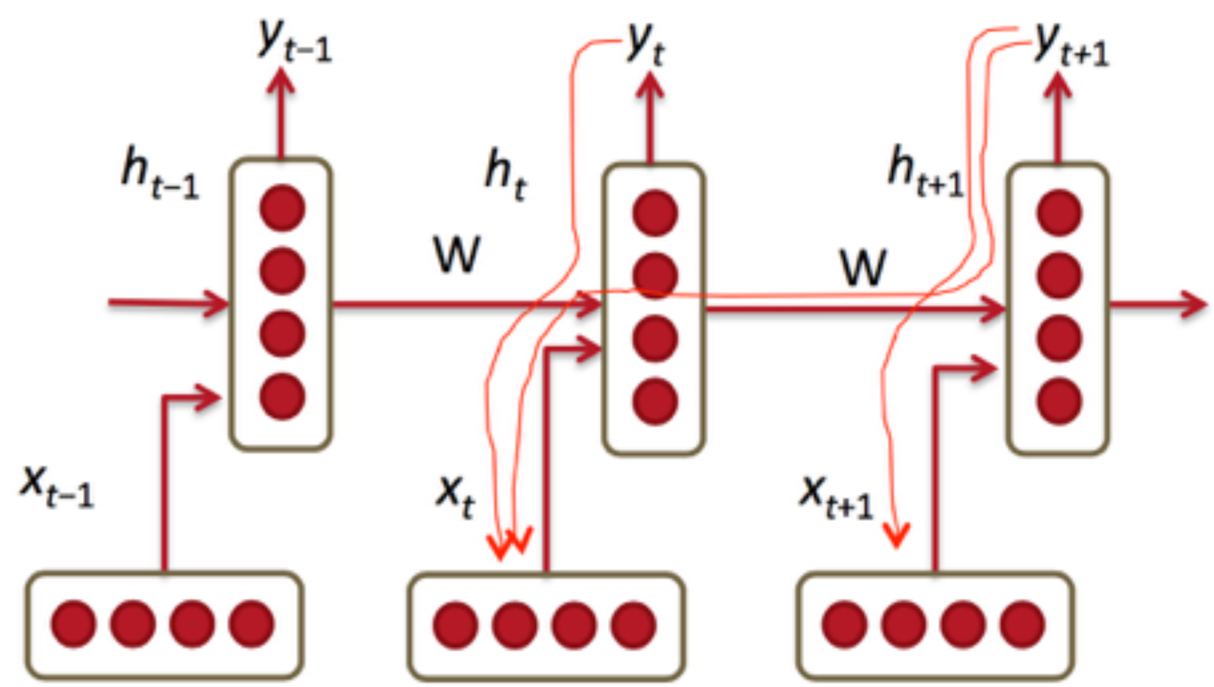
$$h_0 \in \mathbb{R}^{D_h}$$
$$W^{(hh)} \in \mathbb{R}^{D_h \times D_h}$$
$$W^{(hx)} \in \mathbb{R}^{D_h \times d}$$
$$W^{(S)} \in \mathbb{R}^{|V| \times D_h}$$
$$\hat{y} \in \mathbb{R}^{|V|}$$

- 在整个计算过程中，W保持不变
- h_0 在0时刻初始化

RNN训练（1）

- 前向计算，相同的W矩阵需要乘以多次
- 多步之前的输入x，会影响当前的输出
- 在后向计算的时候，同样相同的矩阵也会乘以多次





BPTT算法 – BackProp Through Time

- RNN前向计算

$$\begin{aligned} h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\ \hat{y}_t &= W^{(S)} f(h_t) \end{aligned}$$

- 计算W的偏导，需要把所有Time Step加起来

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

- 应用链式规则

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

BPTT算法：计算实现

- 计算目标 $\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$
- 已知: $h_t \quad = \quad W f(h_{t-1}) + W^{(hx)} x_{[t]}$
- 因此: $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \mathrm{diag}[f'(h_{j-1})]$$

$$\mathrm{diag}(z) = \begin{pmatrix} z_1 & & & 0 & \\ & z_2 & & & \\ & & \ddots & & \\ & 0 & & z_{n-1} & \\ & & & & z_n \end{pmatrix}$$

这里使用是向量微分

BPTT算法：梯度 vanishing/exploding 现象分析

- 已知 $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \text{diag}[f'(h_{j-1})]$
- 根据‖XY‖≤‖X‖‖Y‖，可以知道：

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\text{diag}[f'(h_{j-1})]\| \leq \beta_W \beta_h$$

- 其中Beta代表上限，因此：

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

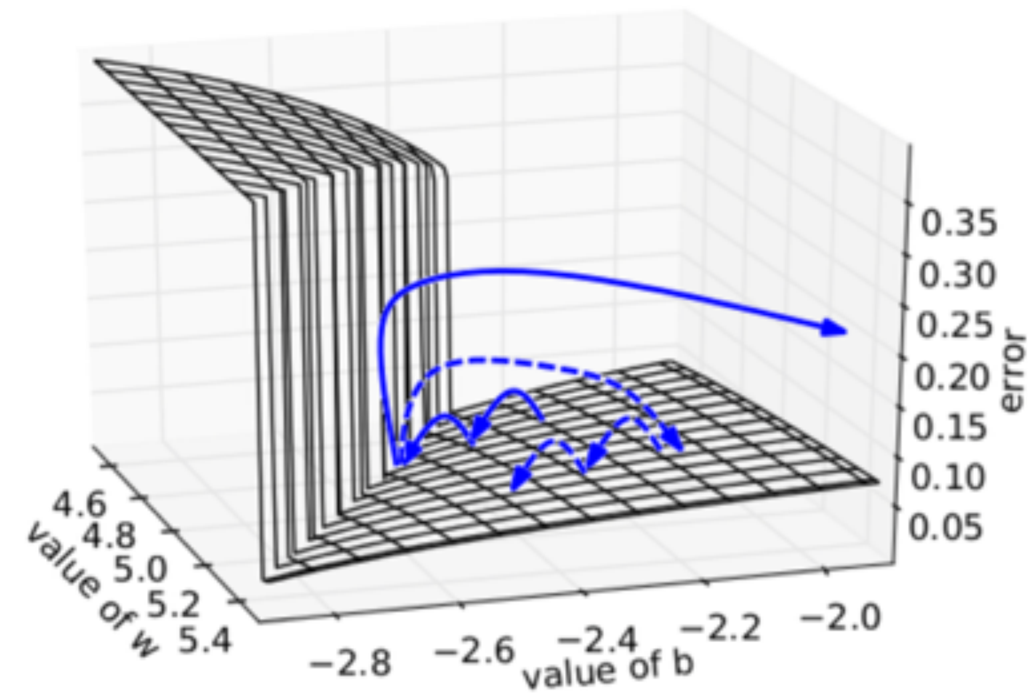
This can become very small or very large quickly [Bengio et al 1994], and the locality assumption of gradient descent breaks down. → **Vanishing or exploding gradient**

BPTT算法：解决方案

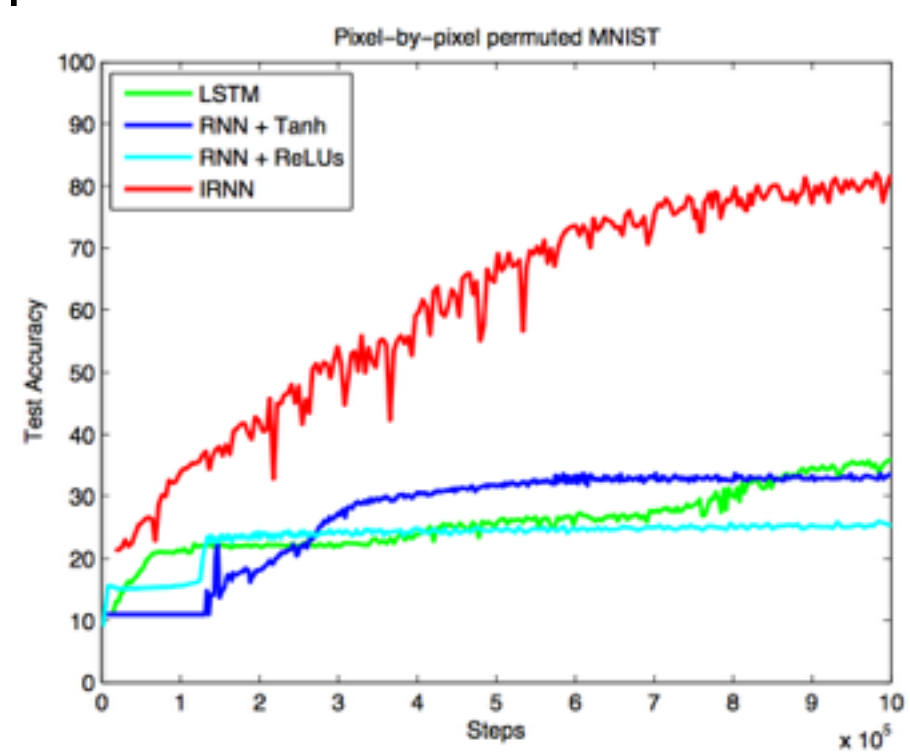
- Clipping

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if



- W初始化为I，使用Relu替换Tanh



LSTM（Long Short Term Memory) Cell

- 一种实际中使用最广泛的RNN基础单元变形
- 具有所谓的“记忆性”，保存隐变量的机制

$$i_t = \sigma(W_i(x_t, h_{t-1}) + b_i)$$

input

$$f_t = \sigma(W_f(x_t, h_{t-1}) + b_f)$$

forgetting

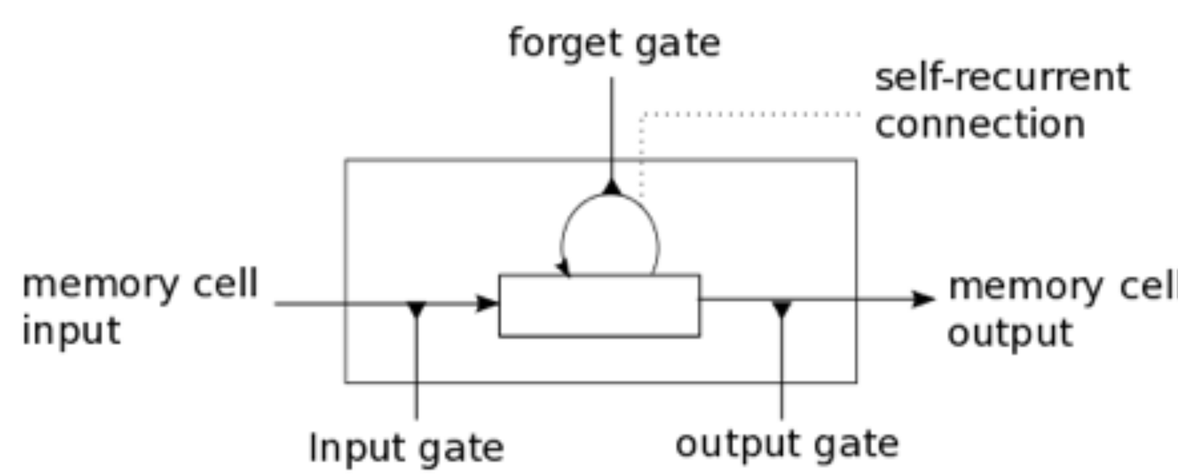
$$\boxed{z_t} = f_t * z_{t-1} + i_t * \tanh(W_z(x_t, h_{t-1}) + b_z)$$

state

$$o_t = \sigma(W_o(x_t, h_{t-1}, z_t) + b_f)$$

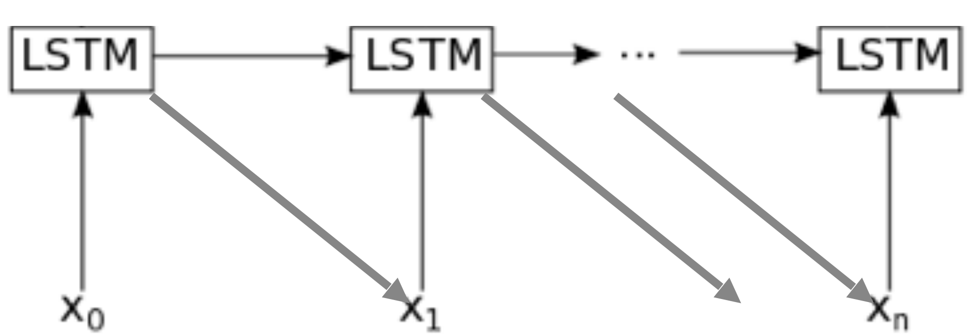
$$h_t = o_t * \tanh z_t$$

output

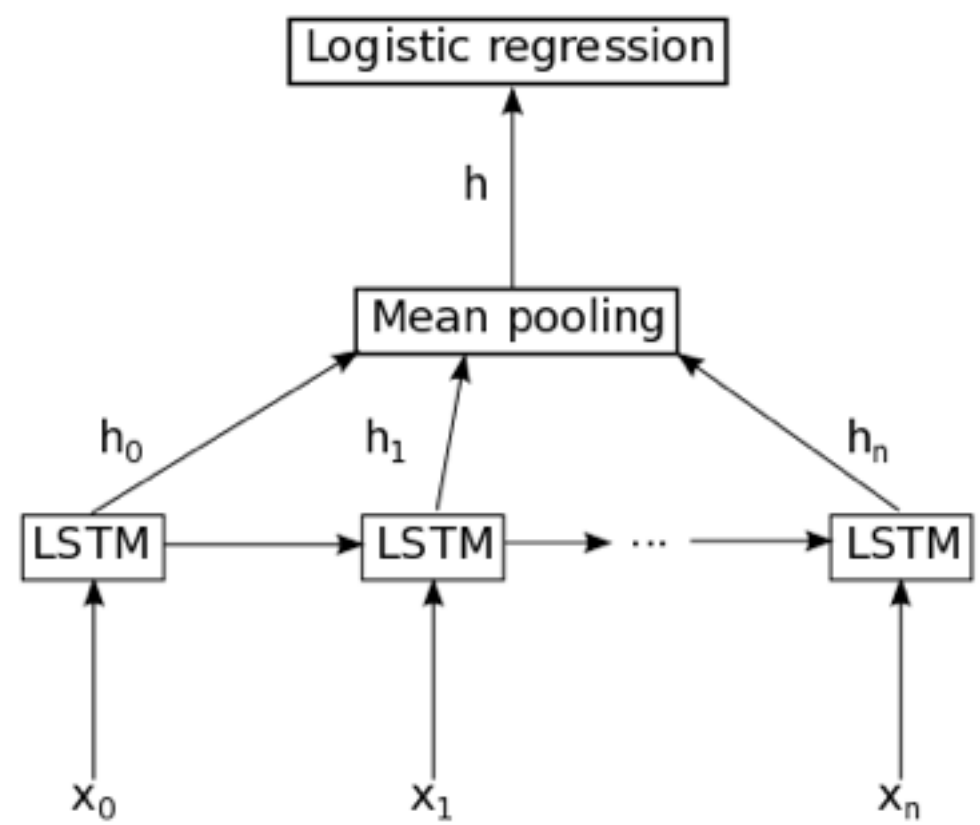


使用LSTM

- 将多个LSTM单元组合为层
- 网络中有多层
- 复杂的结构能够处理更大范围的动态性



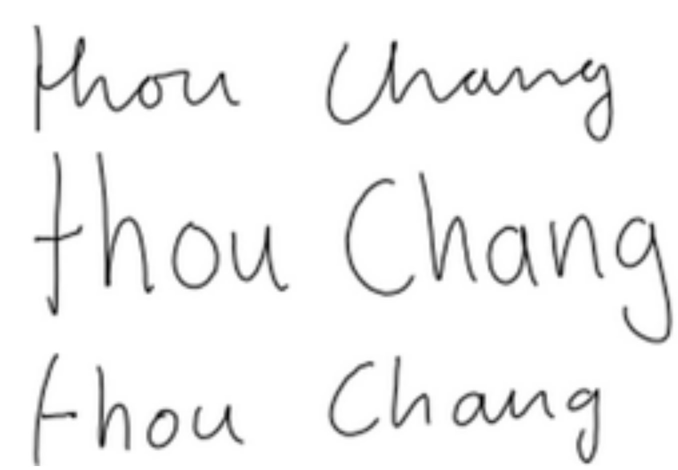
sequence generation



sequence classification

RNN算法应用

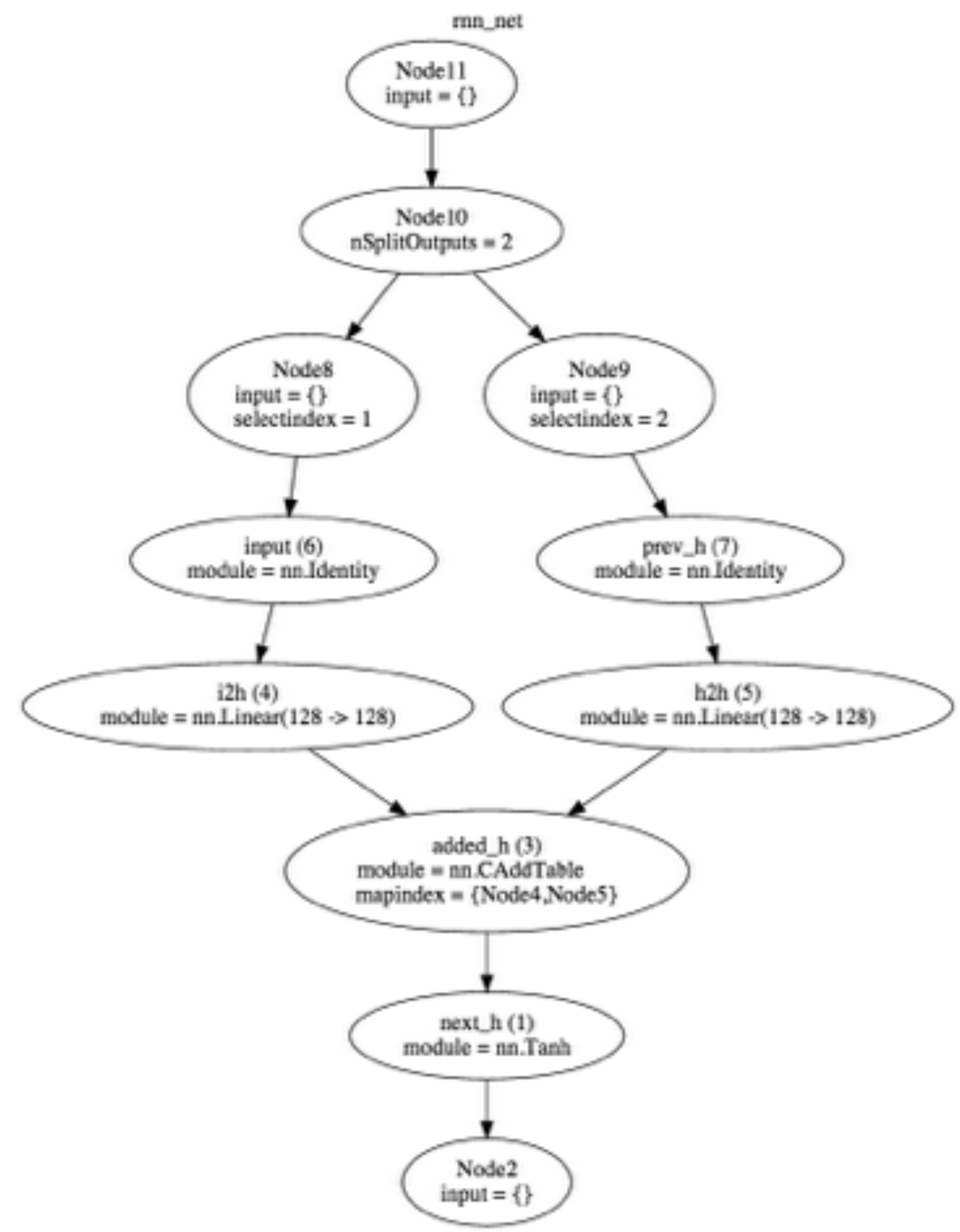
- 手写文字输出：
  - http://www.cs.toronto.edu/~graves/handwriting.html



- 文本生成
- 机器翻译
- 非常好的介绍文章：http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN实验1：基于Torch构建RNN网络

- 使用nngraph模块可以构造复杂的有向图
- BPTT算法依旧基于nn模块提供的BP算法
- 各种optim算法依旧适用RNN



RNN实验2：char_cnn分析

- 基于字符的文本生成工具
  - https://github.com/karpathy/char-rnn
  - 生成汪峰风格的歌词: https://github.com/phunterlau/wangfeng-rnn
  - 莫奈风格绘画生成：http://blog.manugarri.com/teaching-recurrent-neural-networks-about-monet/

课程预告：9月机器学习在线班

- 9月19日晚7点开始上课，每周六周日上课
- 总计20次课，涵盖机器学习主流算法
    - 周末直播，平时答疑
- 邹博我和一起主讲，理论 + 实践
- 三人组队或发微博8折
- 报名链接：
    - http://julyedu.com/course/index/category/machinelearning.html#m31