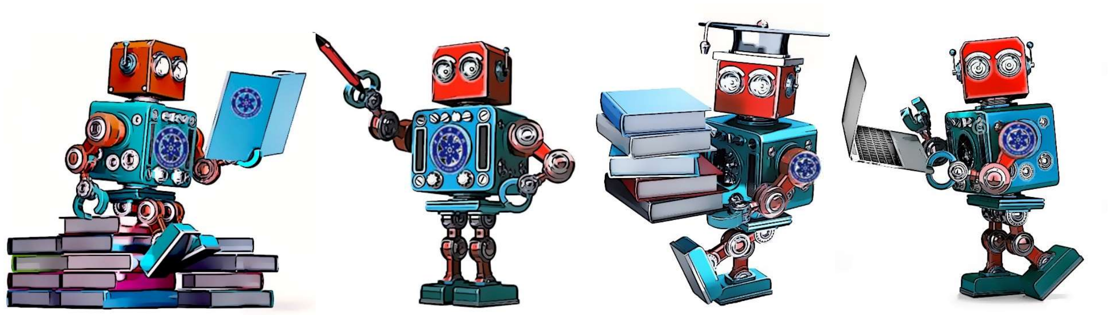
模式识别与机器学习

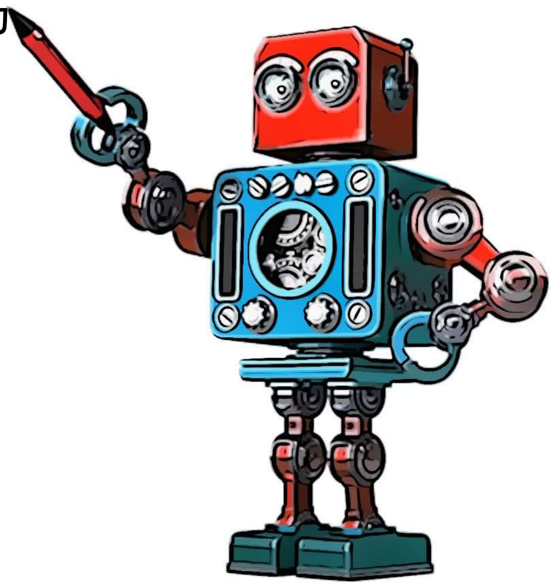
人工神经网络（四）



中国科学院大学 苏荔

suli@ucas.ac.cn

- 卷积神经网络CNN
- 深度神经网络训练技巧
  - 学习率
  - 梯度下降
  - 激活函数
  - Early Stop
  - Regularization
  - Dropout
- 循环神经网络RNN



DNN训练技巧

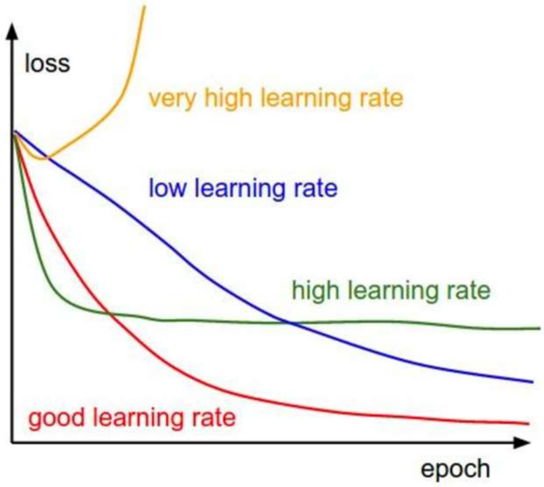
- 学习率
  - 过大
  - 过小
  - 动态调整

  SGD+Momentum、

  AdaGrad、

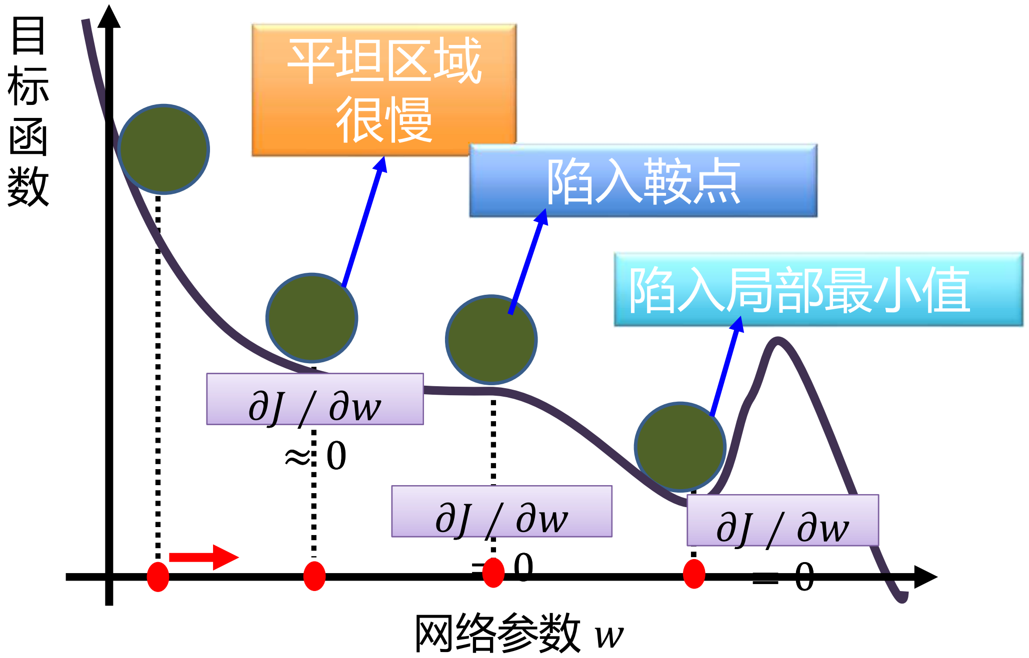
  RMSprop、

  Adam：Adaptive + Momentum



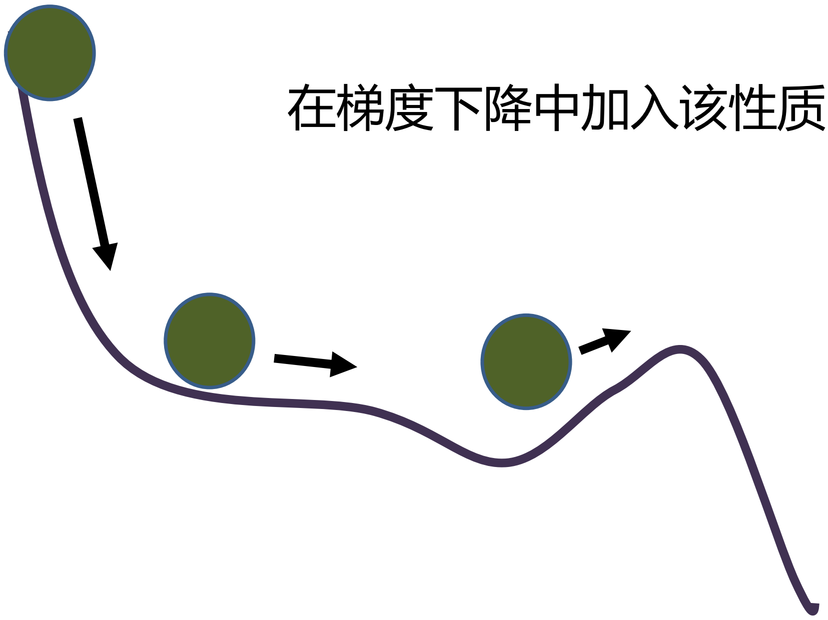
DNN训练技巧

■梯度下降法在有些地方进展缓慢



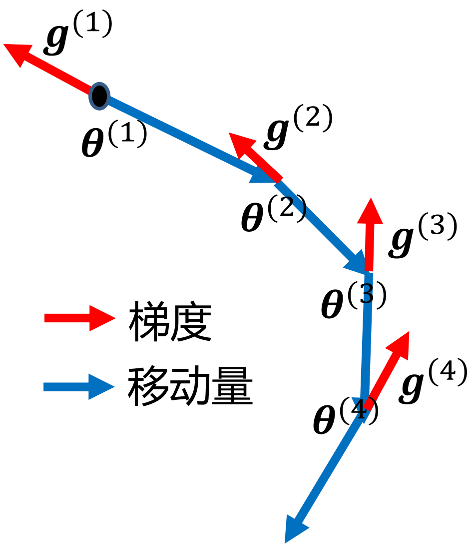
物理世界中……

■动量/惯性



在梯度下降中加入该性质?

回忆：朴素梯度下降



初始位置

计算处的梯度

移到 = -

计算处的梯度

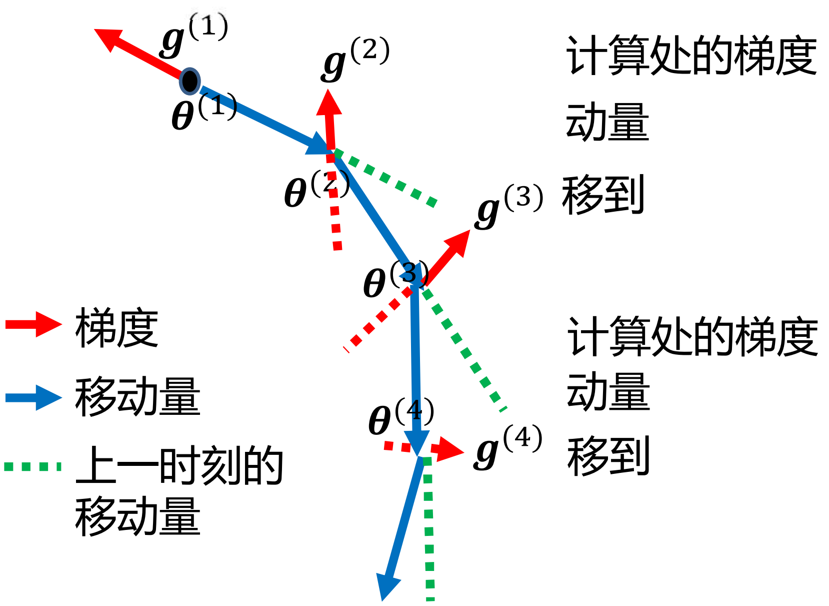
移到 = −

⋮ ⋮

迭代，直到

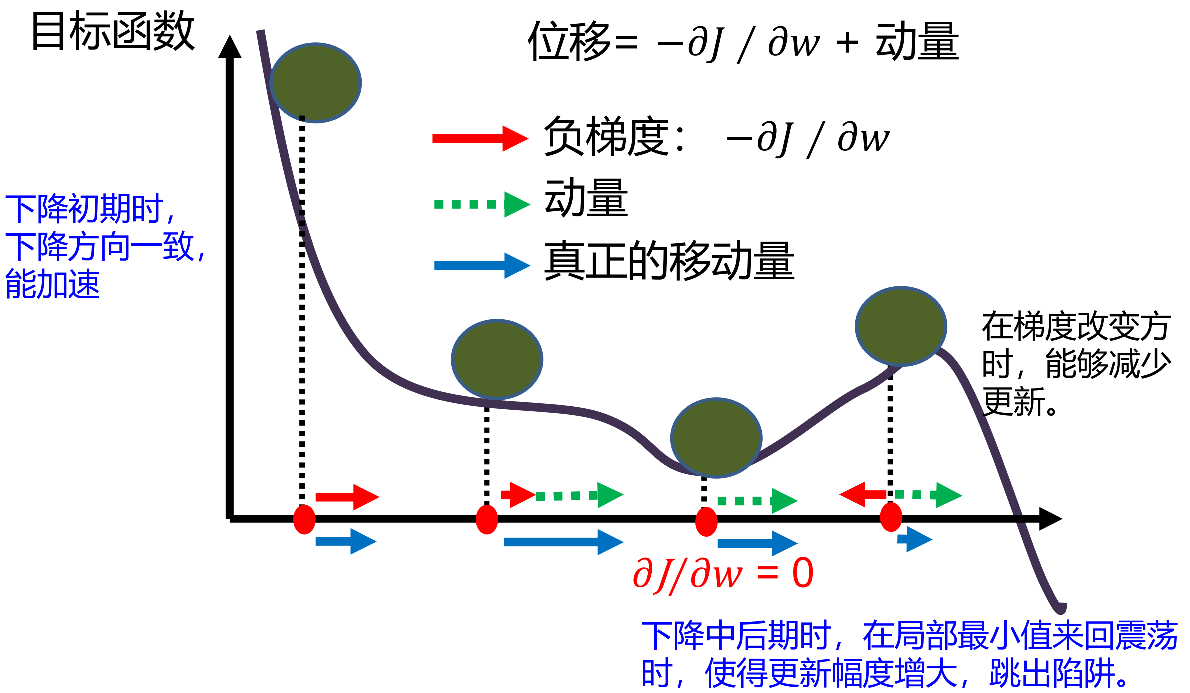
动量法(Momentum)

移动量不仅与梯度有关，还与前一时刻移动量有关



仍然不能保证到达全局最小值，but give some hope ......

位移= $-\partial J\,/\,\partial w$ + 动量



Nesterov动量法

SGD with Nesterov Momentum（涅斯捷罗夫动量法）

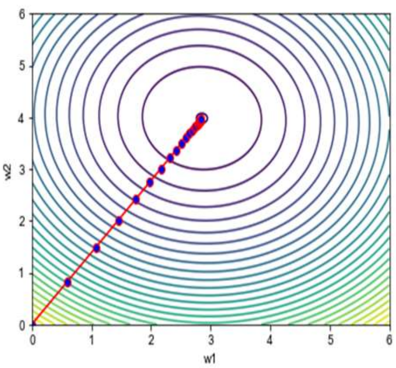
不依据当前位置的梯度，而是计算如果按照速度方向走了一步，那个时候的梯度，再与速度一起计算更新方向

梯度下降：
$$\boldsymbol{v}^{(t)} = -\eta \nabla J(\boldsymbol{\theta}) \mid_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

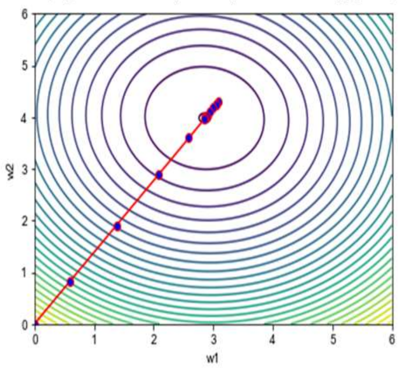
动量法：
$$\boldsymbol{v}^{(t)} = \rho \boldsymbol{v}^{(t-1)} - \eta \nabla J(\boldsymbol{\theta}) \mid_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

NAG：
$$\boldsymbol{v}^{(t)} = \rho \boldsymbol{v}^{(t-1)} - \eta \nabla J \big|_{\boldsymbol{\theta}^{(t)}+\rho \boldsymbol{v}^{(t-1)}}$$

动量法比较



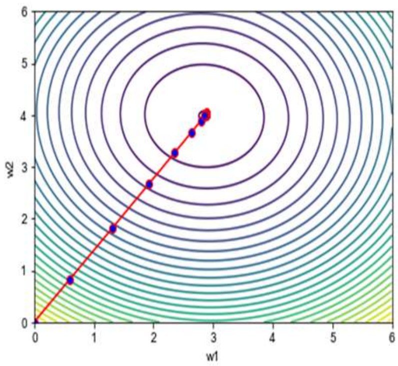
梯度下降



动量法

收敛快

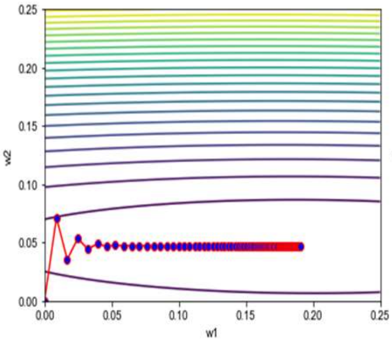
迭代次数比梯度下降法少一半。尤其前几次迭代参数更新量大。最后阶段搜索范围越过了最佳位置（学习率较大），这时两次更新方向相反，动量法会使得更新幅度减小，再慢慢回到最佳位置

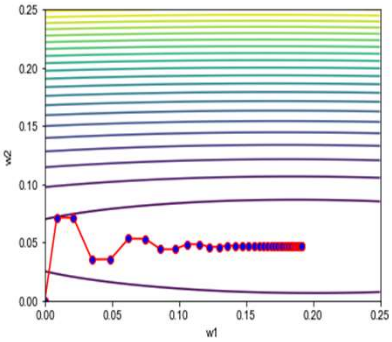


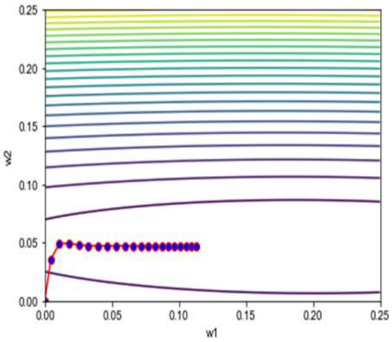
NAG

提前预知目标函数的信息，相当于多考虑了目标函数的二阶导数信息，类似牛顿法的思想，因此搜索路径更合理，收敛速度更快

动量法比较







梯度下降　　　　动量法　　　　NAG

梯度下降法中参数在竖直方向比在水平方向移动幅度更大，在长轴“之”字形反复跳跃，缓慢向最小值逼近 。（不同参数的梯度范围差异大）

竖直方向上的移动更加平滑，且在水平方向上更快逼近最优解，因为此时竖直方向的当前梯度与之前的梯度方向相反相互抵消，移动的幅度小

提前预知目标函数的信息，相当于多考虑了目标函数的二阶导数信息，类似牛顿法的思想，因此搜索路径更合理，收敛速度更快

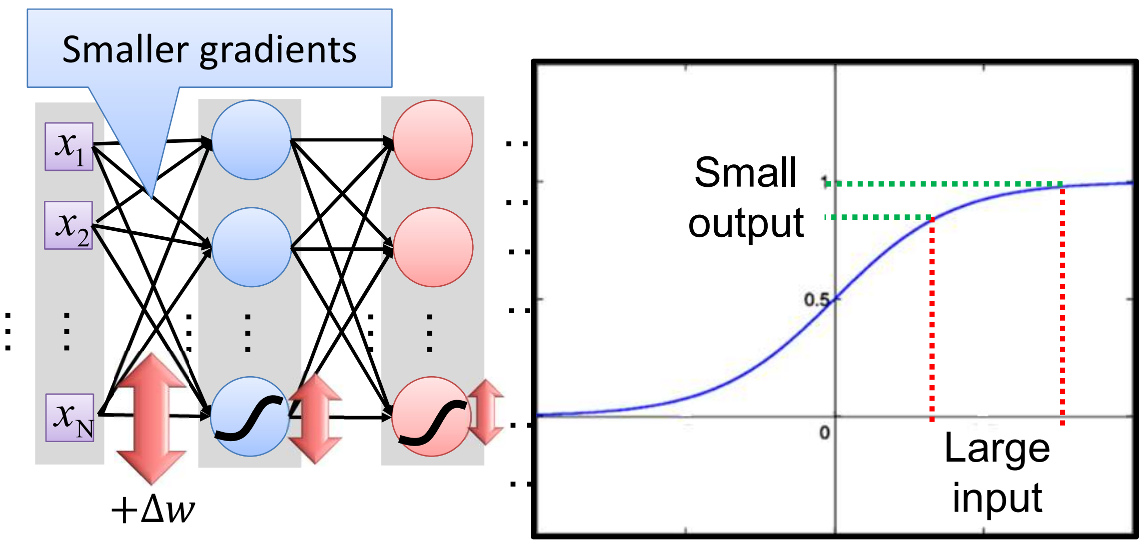
补充：BGD，SGD，miniBGD

- Batch gradient descent：遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。即每更新一次参数要把数据集里所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习
- Stochastic gradient descent：每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降。速度比较快，但是收敛性能不太好，可能造成目标函数震荡
- mini-batch gradient decent：折中，小批的梯度下降，把数据分为若干个批，按批来更新参数。这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大

DNN训练技巧

- 激活函数
  - Sigmoid
  - Rectified Linear Unit (ReLU)
  - Maxout: Learnable activation function
    [Ian J. Goodfellow, ICML' 13]
  - Scaled exponential linear units (SELU)
    缩放指数线性单元：具有自归一化功能，满足某些条件时，使用该激活函数后使得样本分布满足零均值和单位方差[NIPS' 17]

梯度消失

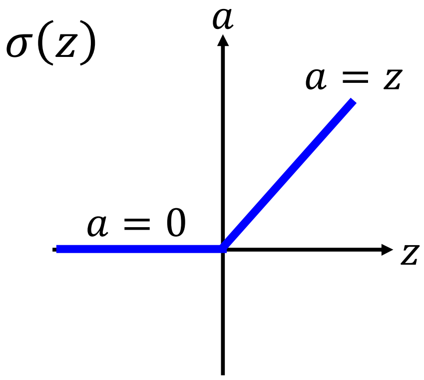


Intuitive way to compute the derivatives …

$$\frac{\partial l}{\partial w} =? \frac{\Delta l}{\Delta w}$$

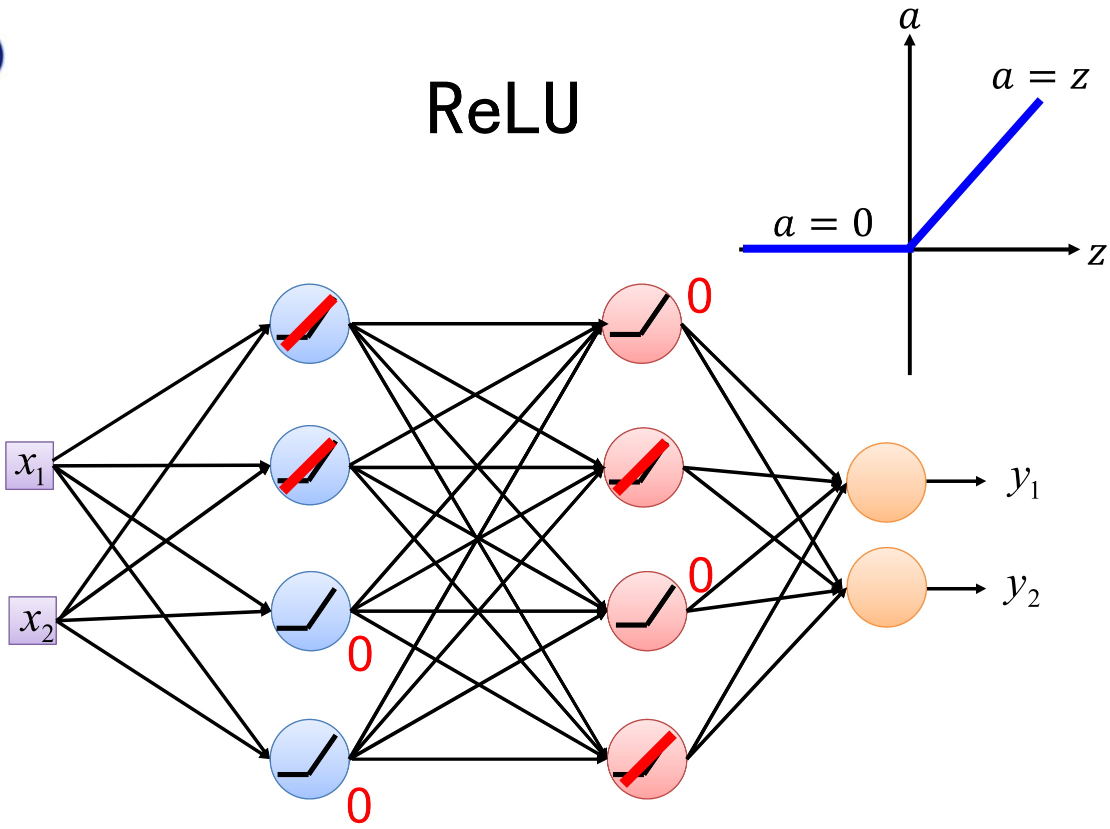
ReLU

- Rectified Linear Unit (ReLU)

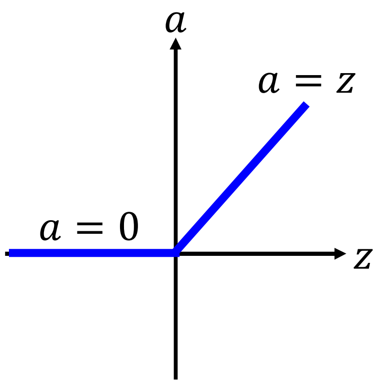


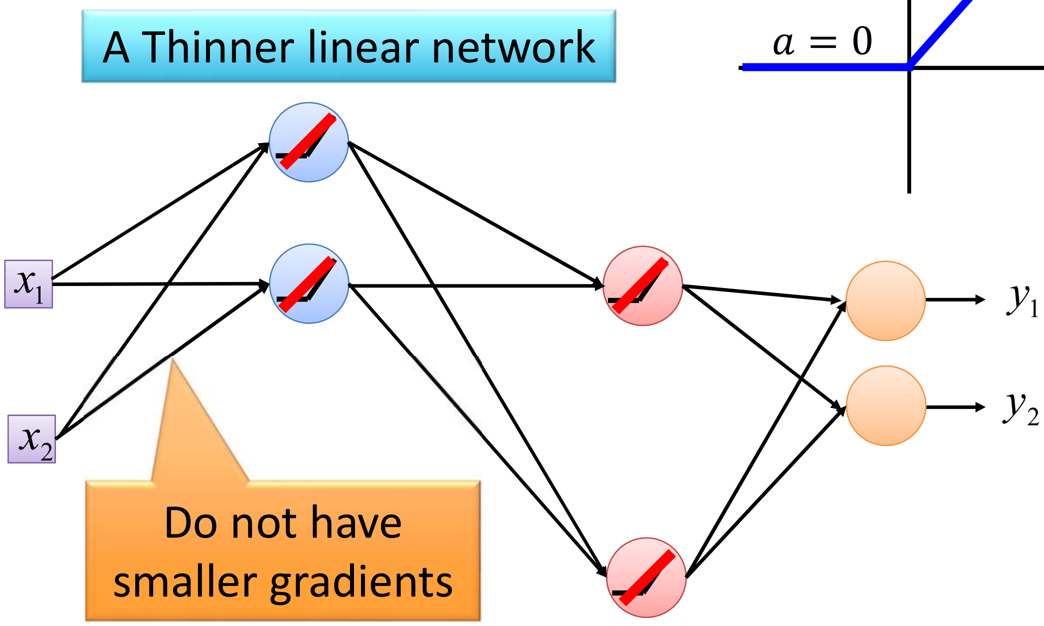
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

ReLU

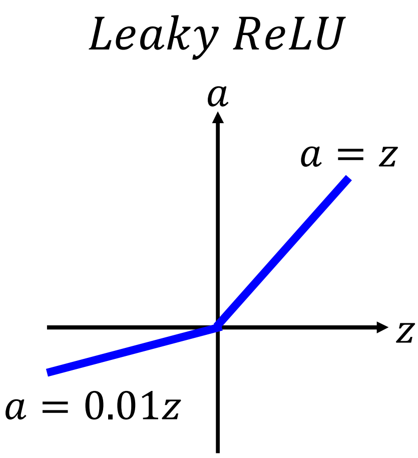


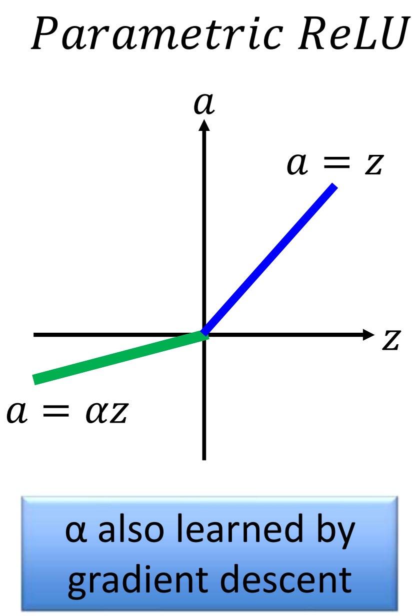
ReLU


a
a = z
a = 0
z


A Thinner linear network
x_1
x_2
y_1
y_2
Do not have smaller gradients

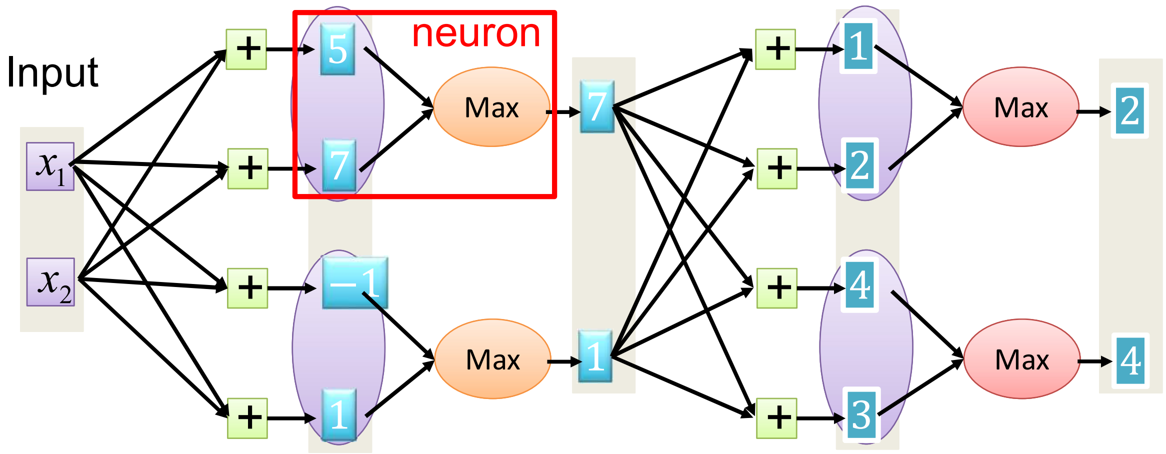
改进ReLU


Leaky ReLU
a
a = z
z
a = 0.01z


Parametric ReLU
a
a = z
z
a = αz
α also learned by gradient descent

Maxout

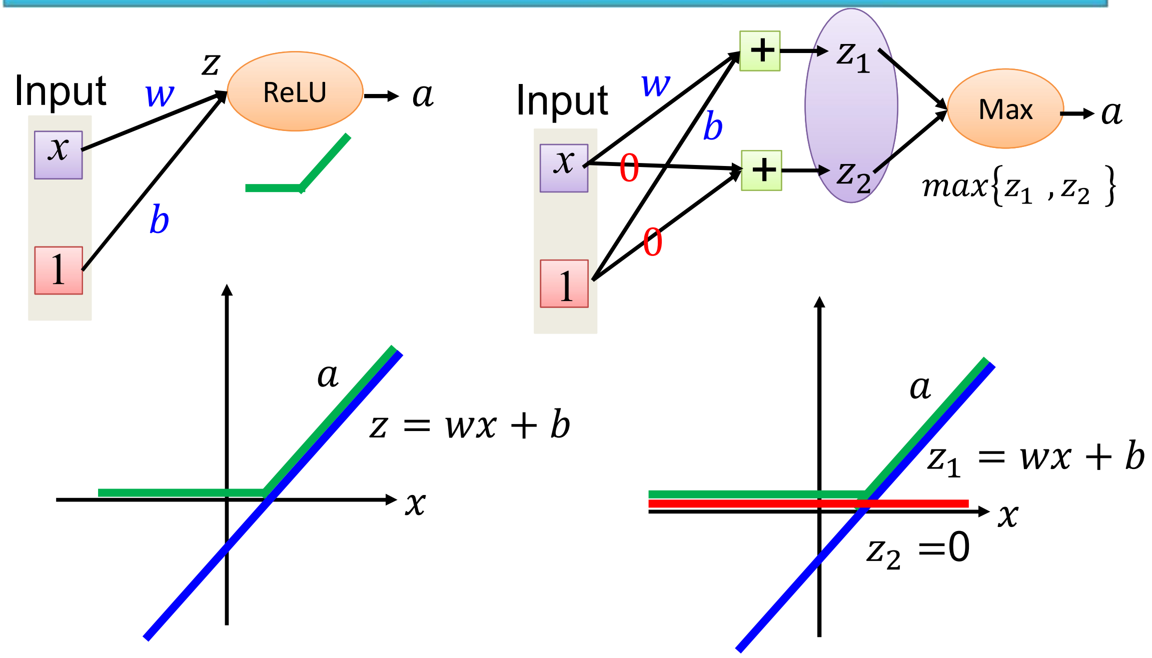
- Learnable activation function [Ian J. Goodfellow, ICML' 13]



You can have more than 2 elements in a group.

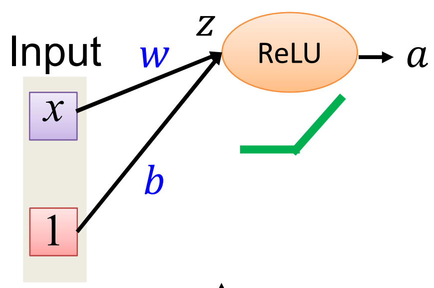
Maxout

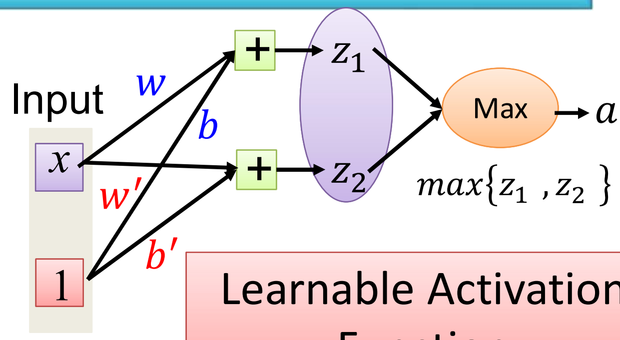
ReLU is a special cases of Maxout



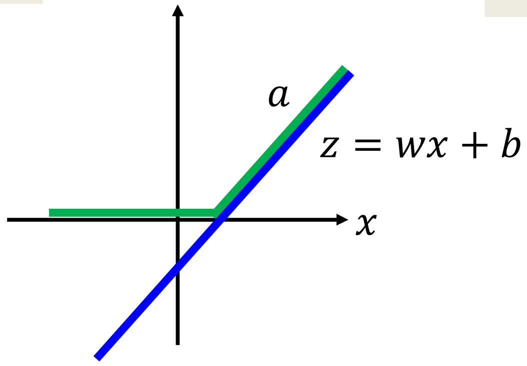
Maxout

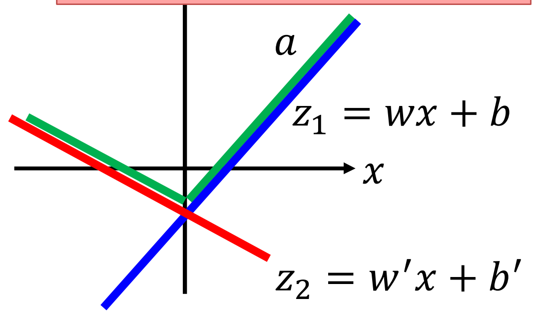
More than ReLU





Learnable Activation Function

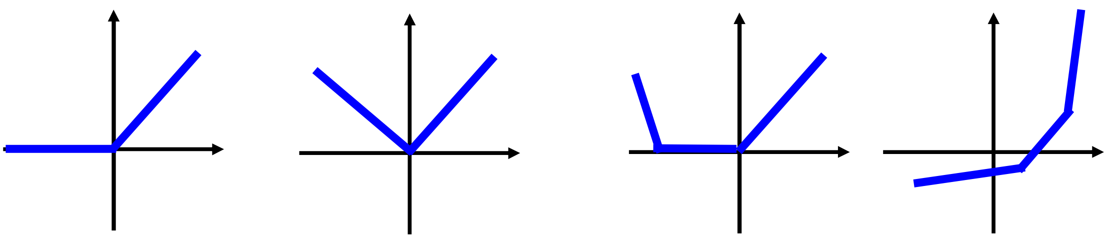




Maxout

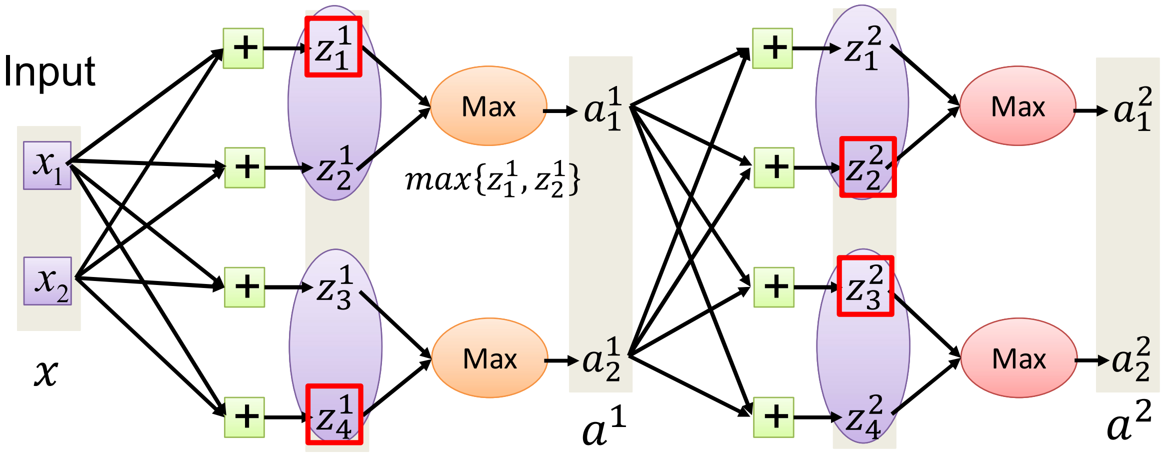
- Learnable activation function [Ian J. Goodfellow, ICML' 13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group **3 elements in a group**



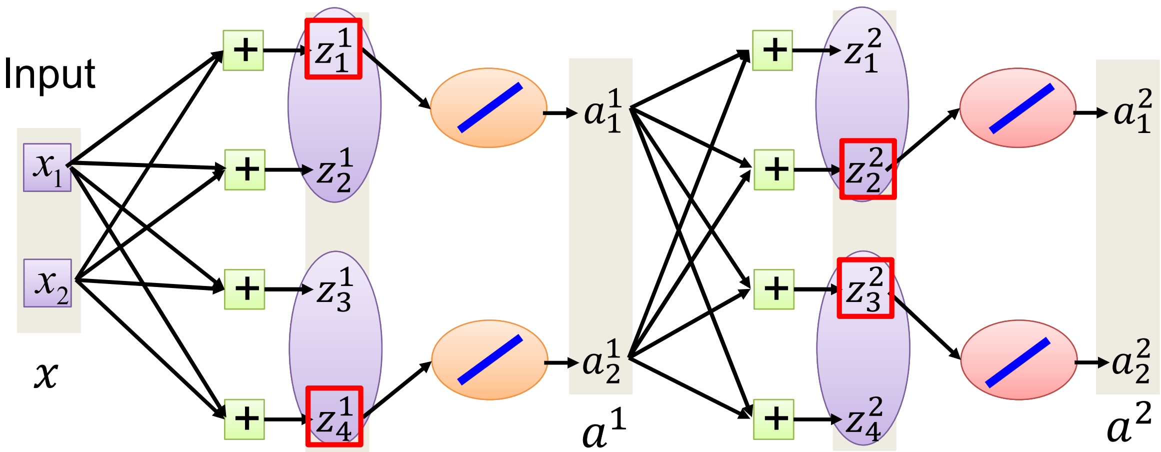
Maxout 训练

- Given a training data x, we know which z would be the max



Maxout 训练

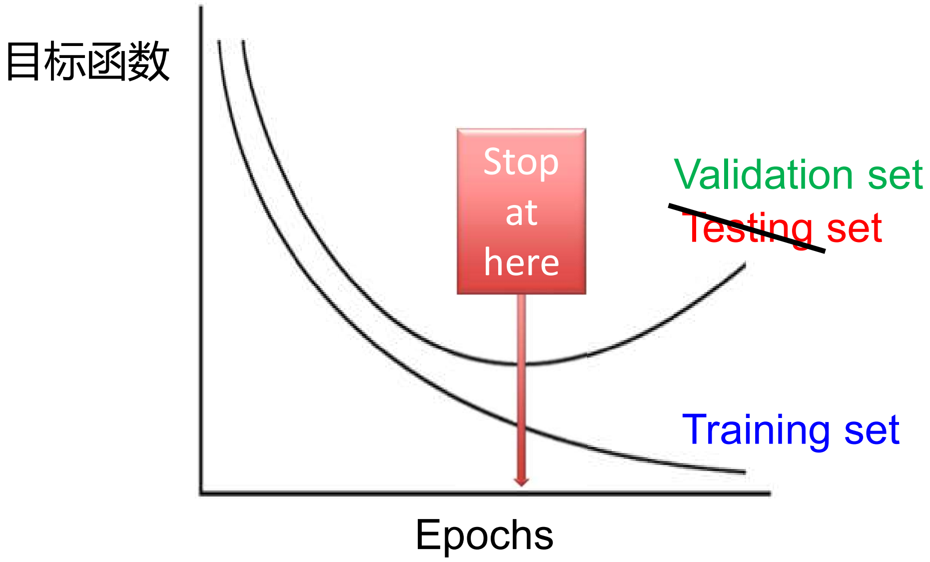
- Given a training data x, we know which z would be the max



- Train this thin and linear network

DNN训练技巧

- Early Stopping



DNN训练技巧

■正则（Regularization）

•机器学习模型的目标函数通常包含两项：模型不仅在训练集上的损失和要小，而且参数要尽可能接近0

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} L_i(y_i, \hat{y}_i) + \lambda R(\boldsymbol{\theta})$$

损失函数
(如交叉熵损失，L2损失 ...)

正则项：$\boldsymbol{\theta} = [w_1, w_2, \ldots,]$

L2 正则：

L1 正则：

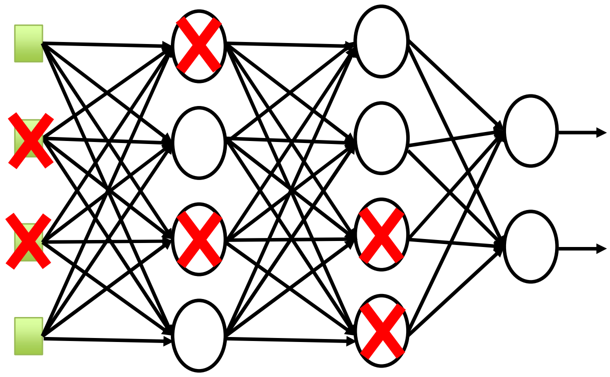
(正则项通常不考虑偏置)

Weight Decay

DNN训练技巧

■ **Dropout**

训练:

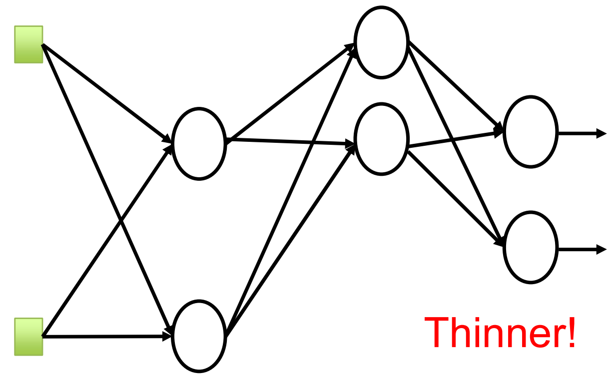


➢ 每次更新参数之前

  ➢ 每个神经元丢弃p%的连接

■ **Dropout**

训练:



➢ 每次更新参数之前

  ➢ 每个神经元丢弃p%连接

  ➡ 神经网络的结构变了

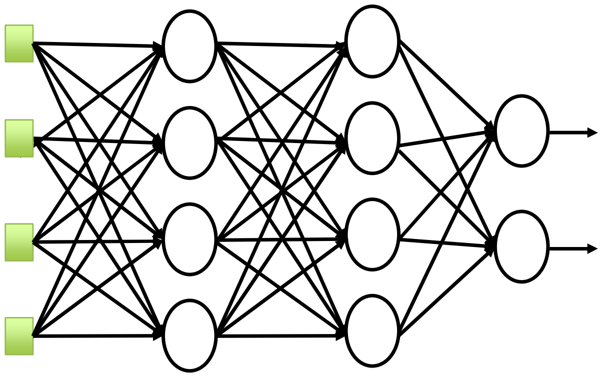
- 用新的网络参与训练

对每个mini-batch，重采样要丢弃的神经元

■Dropout

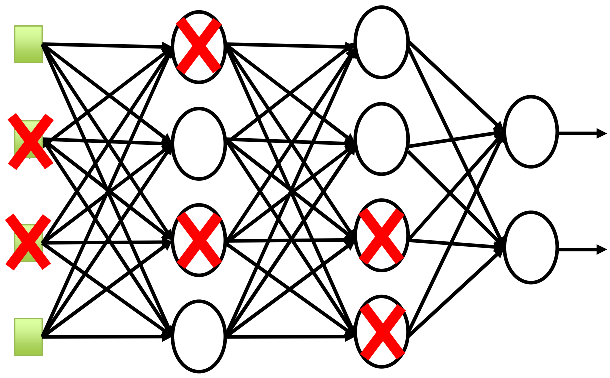
测试

No dropout ！



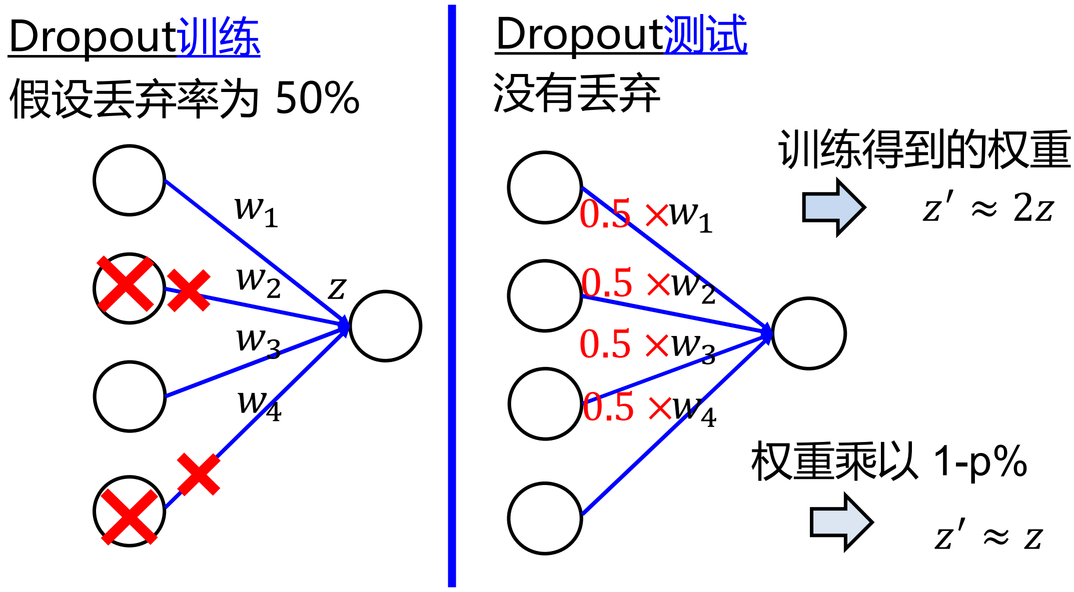
- 如果训练时的丢弃率 p%，所有的权重乘以 1-p%。
- 假设丢弃率为50%，如果训练得到的权重 ，则测试时 。

Dropout – 直觉解释



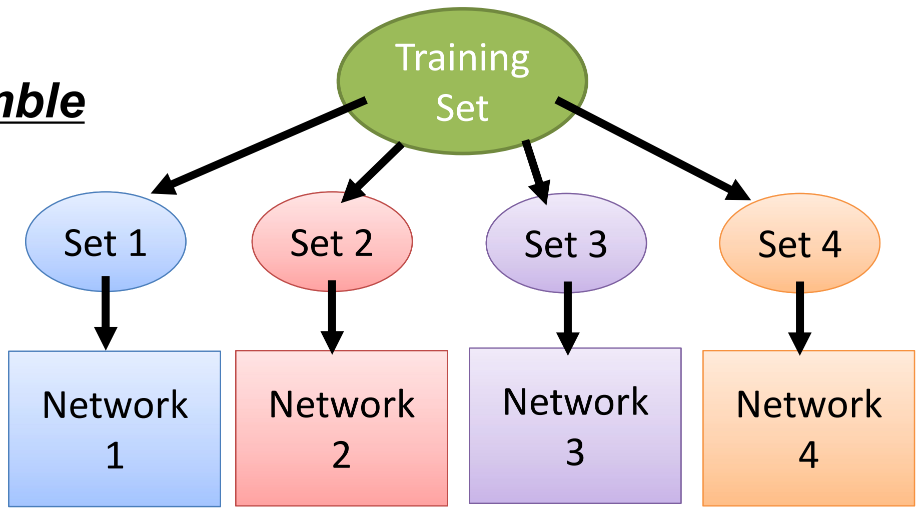
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

■为什么测试时权重要乘以 (1-p%)？



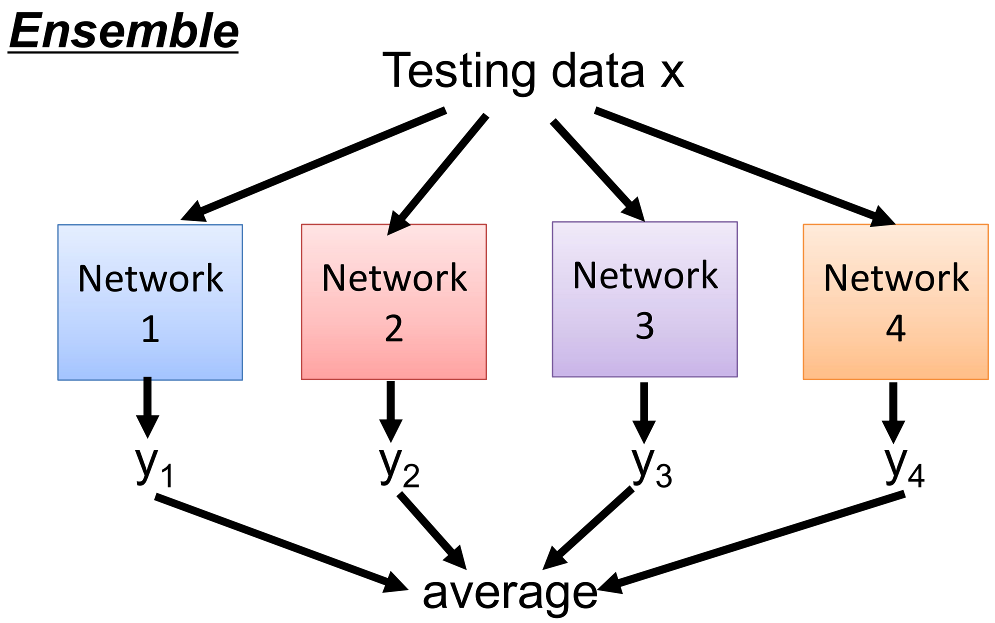
Dropout：可视为一种集成学习

Ensemble

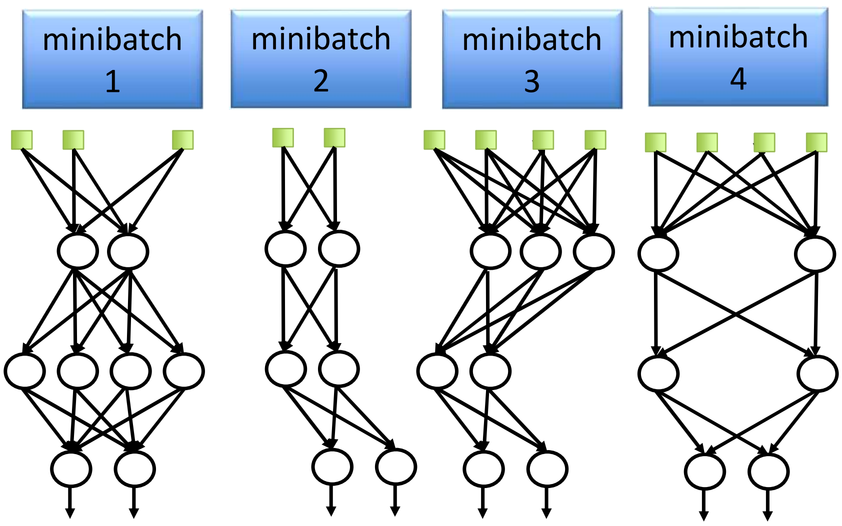


以不同的结构训练多个模型

Dropout：可视为一种集成学习



Dropout：可视为一种集成学习



Training of Dropout

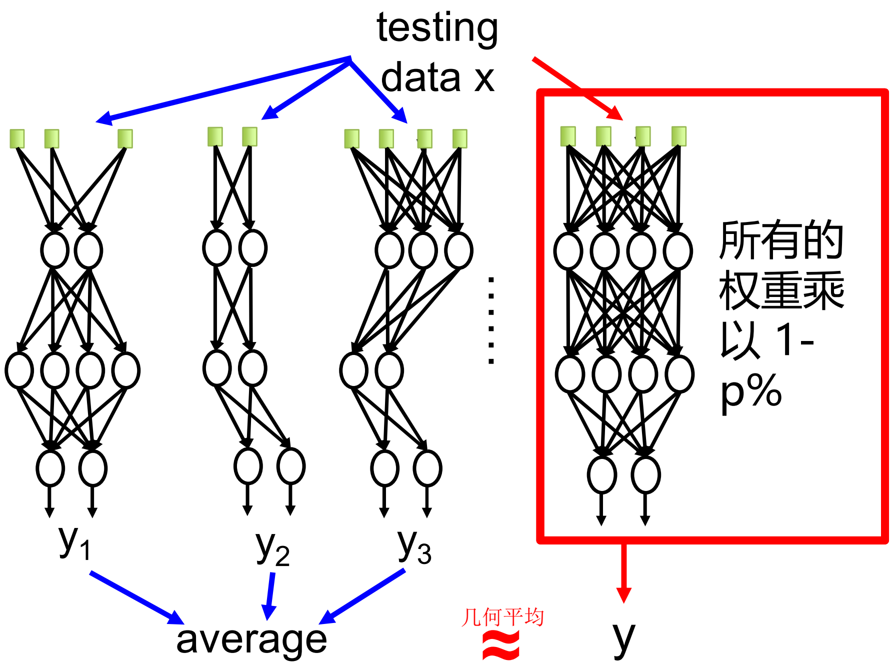
M neurons

2^M possible networks

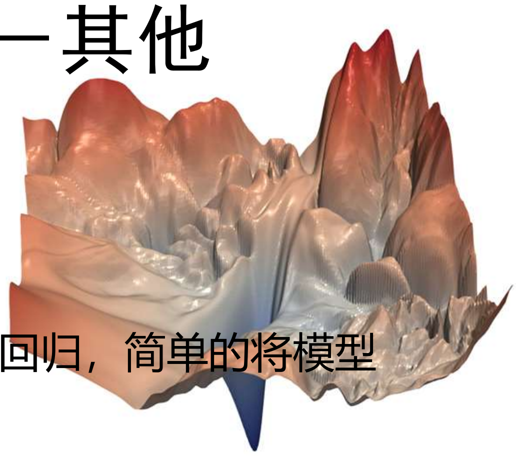
- ➢每次用一个mini-batch训练一个网络
- ➢这些网络中有些参数是共享的。

Dropout：可视为一种集成学习

Testing of Dropout



DNN训练技巧—其他



■参数（权重）初始化

对简单的机器学习模型，如Logistic回归，简单的将模型参数初始化0或较小的随机数即可。

对深度模型，由于目标函数非凸，层次深，如何选择参数初始值便成为一个值得探讨的问题。

偏置参数通常设置为

模型权重的初始化对于网络的训练很重要：不好的初始化参数会导致梯度传播问题，降低训练速度；好的初始化参数能够加速收敛，并且更可能找到较优解。

DNN训练技巧—其他

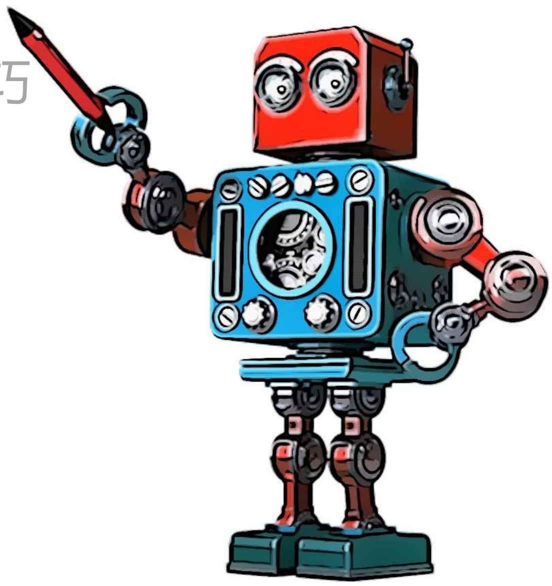
■ 输入数据标准化
- 例如：每个特征的均值为0、标准差为1
- 各个特征分布相近→更容易训练出有效的模型
- 在深层神经网络中，即使输入数据已做标准化，训练中模型参数的更新依然很容易造成靠近输出层输出的剧烈变化 → 难以训练出有效的模型
- Batch normalization：在小批量上进行归一化，从而使整个神经网络在各层的中间输出的数值更稳定

■ ……

□ 卷积神经网络CNN

□ 深度神经网络训练技巧

□ 循环神经网络RNN
- 基本原理
- LSTM
- GRU



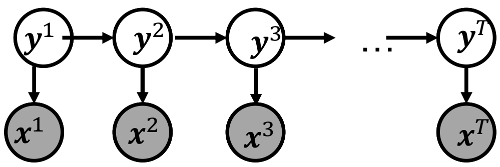
序列建模

■序列建模中上下文很重要，如语音识别中的同音字/词

- 例：工夫 - 功夫

你只要有工夫，就可以和我到操场打球。
语言的功夫，要从写作的实践上修养。

■HMM等模型可建模时序上下文信息（这里用上标表示时间索引）



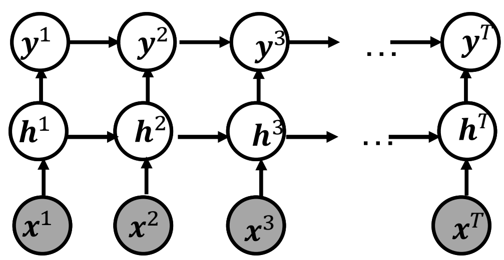
$$p(\mathbf{x},\mathbf{y}) = p(\mathbf{y}^1)\prod_{t=1}^{T-1} p(\mathbf{y}^{t+1}|\mathbf{y}^t)\prod_{t=1}^{T} p(\mathbf{x}^t|\mathbf{y}^t)$$

网络怎样记忆上下文信息?

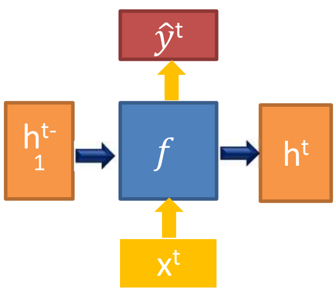
循环神经网络 (Recurrent Neural Network，RNN)

■隐含状态：存储与过去的相关信息

■按时间展开



■单个神经元

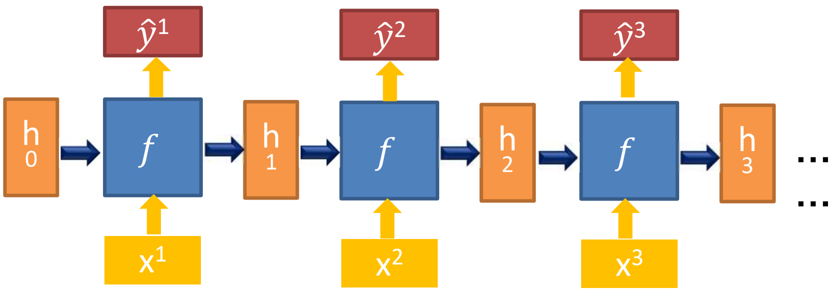


$$(\boldsymbol{y}^t, \boldsymbol{h}^t) = f(\boldsymbol{h}^{t-1}, \boldsymbol{x}^t)$$

模型与时间无关（时间上参数共享、时不变系统）
不管输入/输出序列有多长，我们只需要一个函数 f

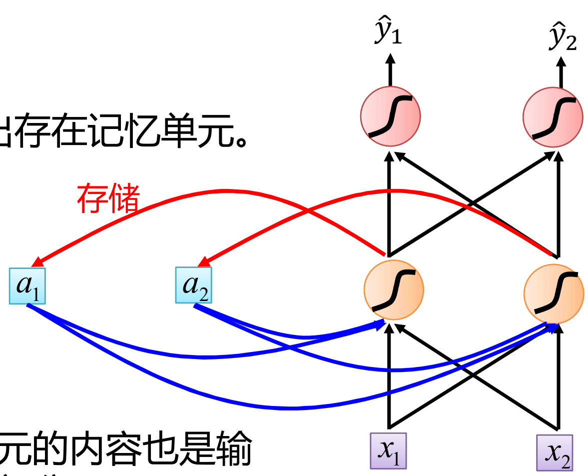
- $\boldsymbol{y}^t, \boldsymbol{h}^t = f(\boldsymbol{h}^{t-1}, \boldsymbol{x}^t)$

$\boldsymbol{h}^t$和$\boldsymbol{h}^{t-1}$为维度相同的向量



RNN例子

隐含层的输出存在记忆单元。

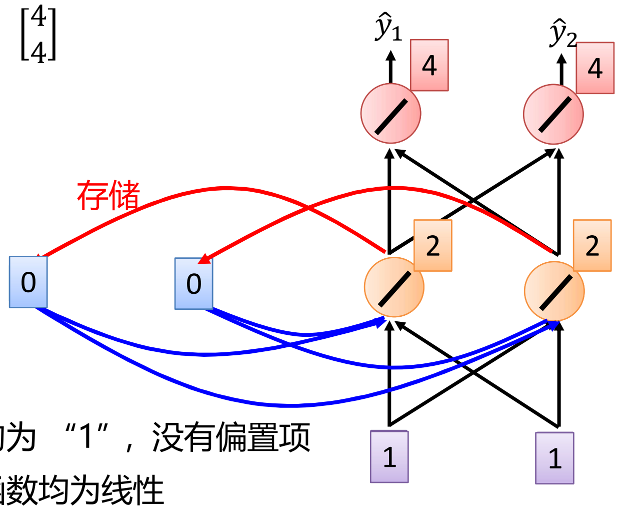


记忆单元的内容也是输入的一部分。

输入: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

输出: $\begin{bmatrix}4\\4\end{bmatrix}$

给定初始状态

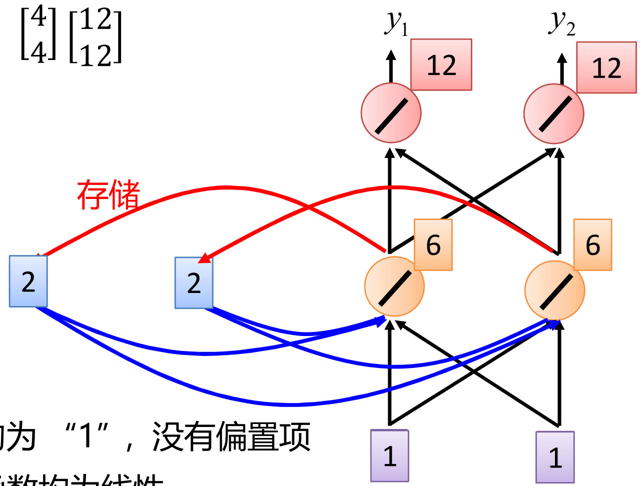


所有权重均为 “1”，没有偏置项

所有激活函数均为线性

输入: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

输出: $\begin{bmatrix}4\\4\end{bmatrix}$ $\begin{bmatrix}12\\12\end{bmatrix}$



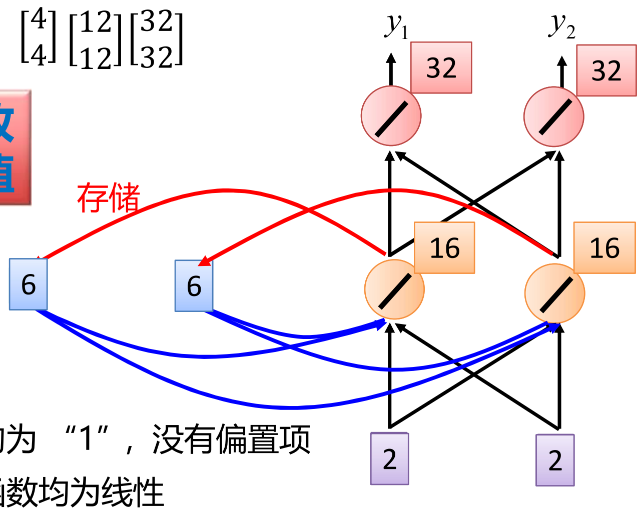
所有权重均为 “1”，没有偏置项

所有激活函数均为线性

输入: $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}$ ……

输出: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$

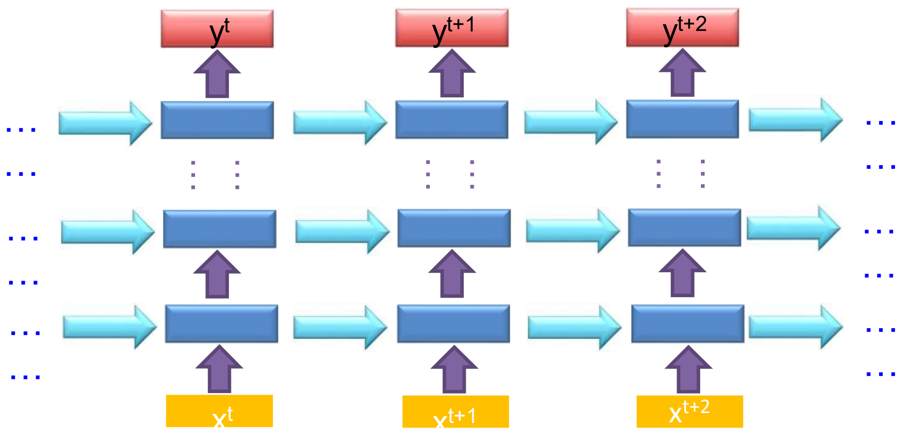
输入序列顺序改变会改变输出值



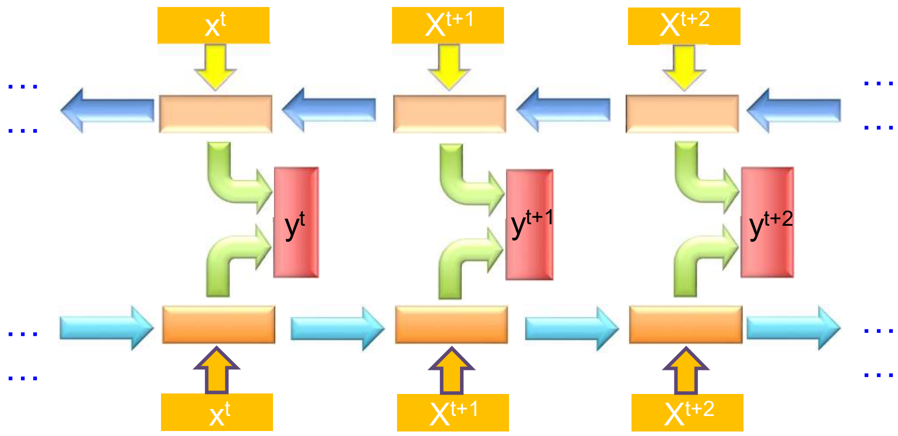
所有权重均为 “1”，没有偏置项

所有激活函数均为线性

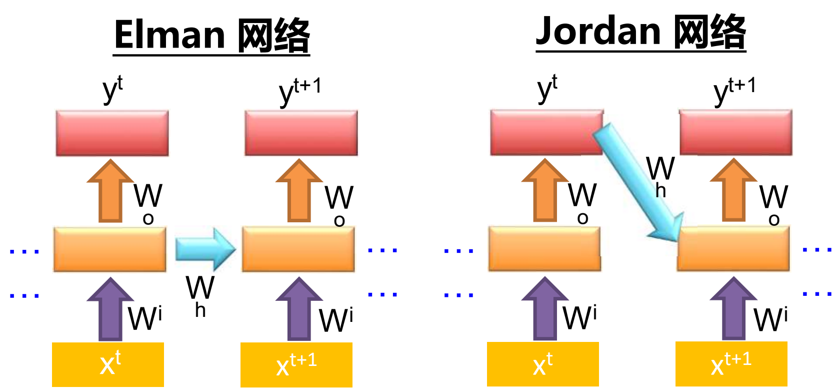
深度RNN



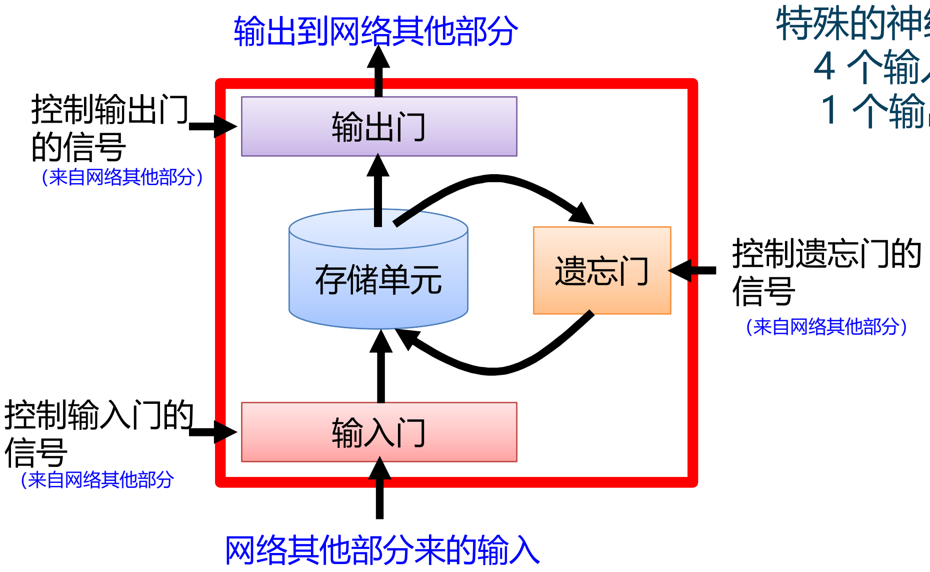
双向RNN



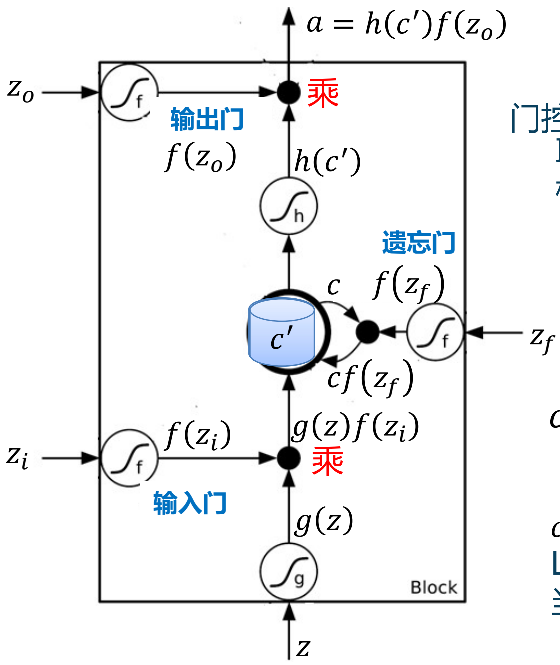
Elman 网络& Jordan 网络



Long Short-term Memory (LSTM)



特殊的神经元:
4 个输入
1 个输出



门控函数通常用sigmoid函数
取值为0 ~ 1之间，
模拟门的开关

$$c' = g(z)f(z_i) + cf(z_f)$$

c' 等于之前的信息 c（记忆Long short-term信息）加上当前的信息

LSTM例子

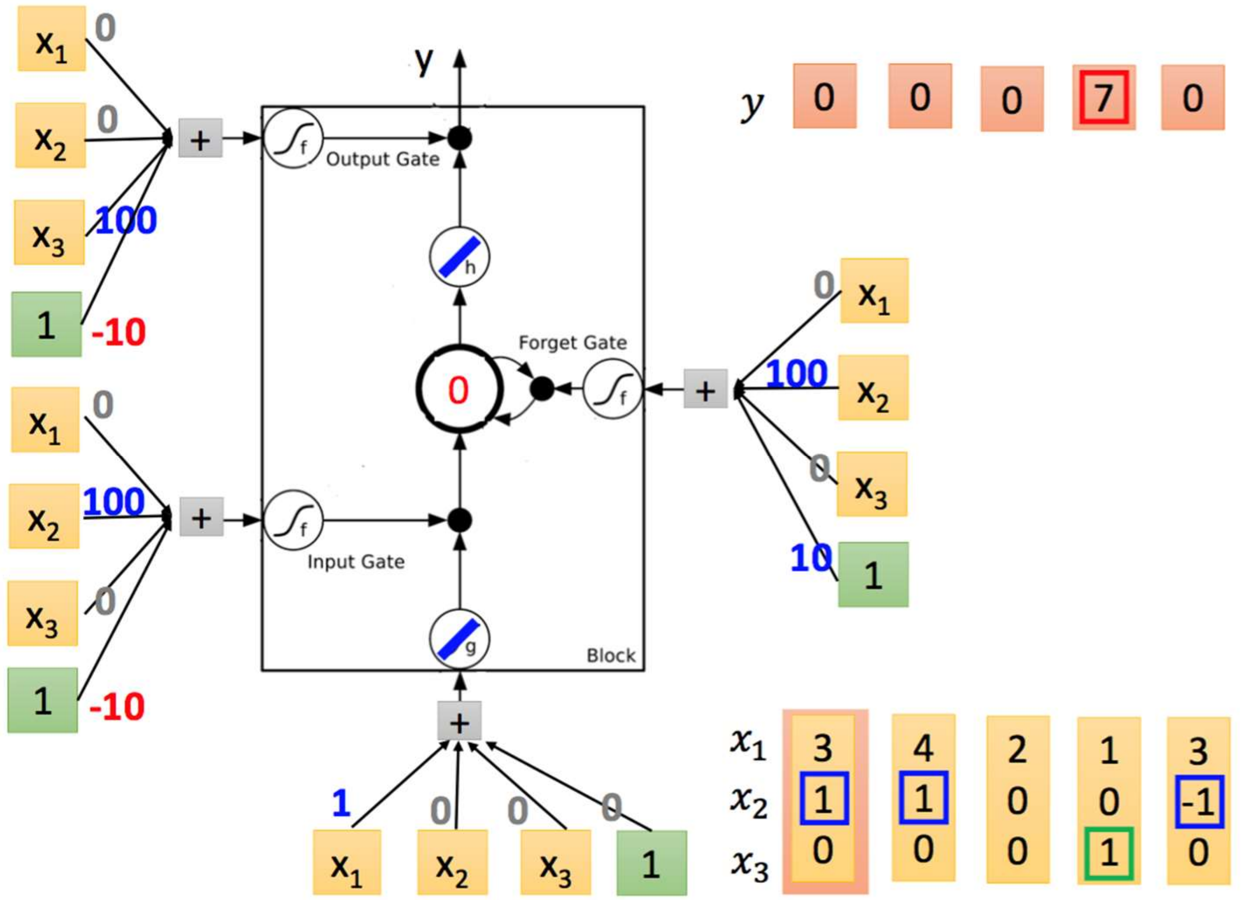
摘自李宏毅老师讲义

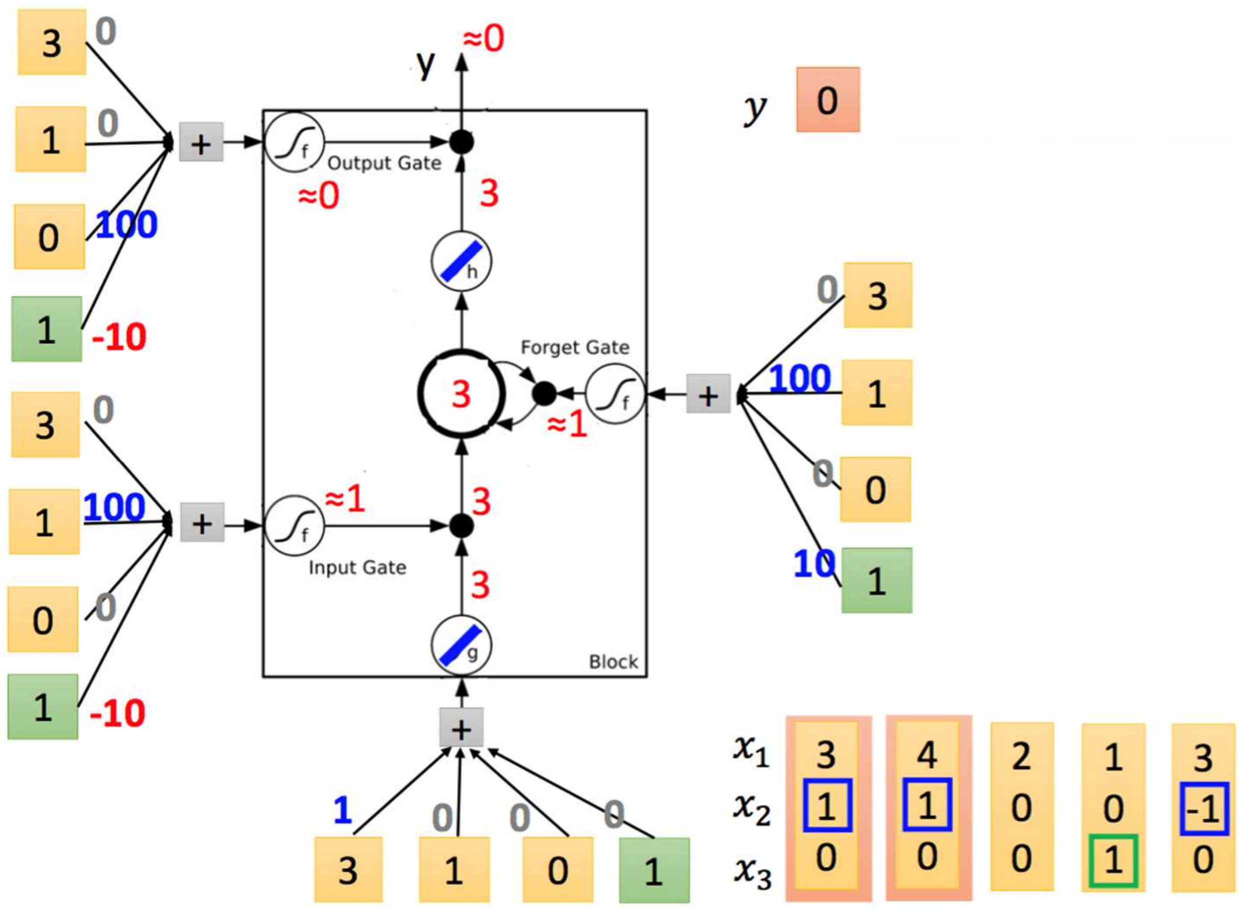
	0	0	3	3	7	7	7	0	6
x_1	1	3	2	4	2	1	3	6	1
x_2	0	1	0	1	0	0	-1	1	0
x_3	0	0	0	0	0	1	0	0	1
y	0	0	0	0	0	7	0	0	6

When x_2 = 1, add the numbers of x_1 into the memory

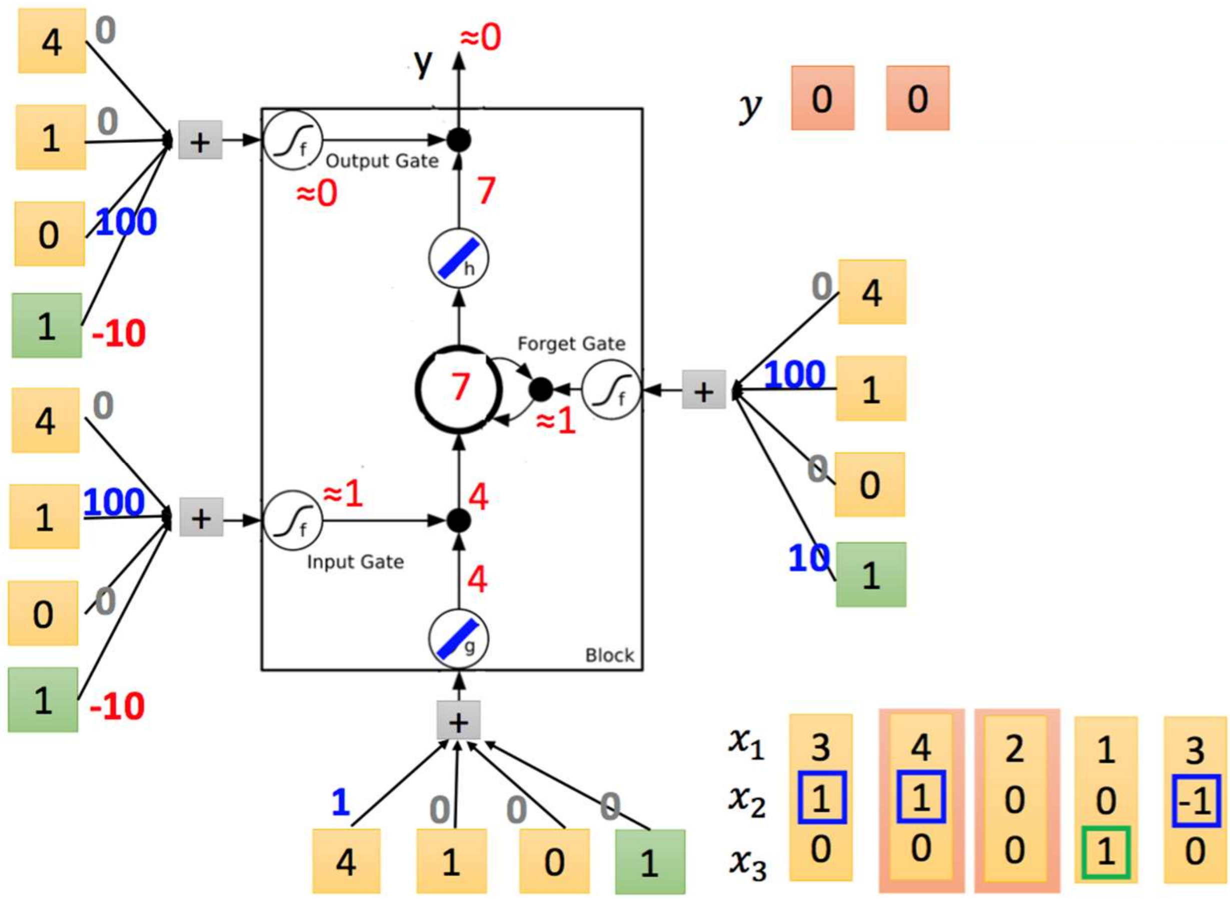
When x_2 = -1, reset the memory

When x_3 = 1, output the number in the memory.

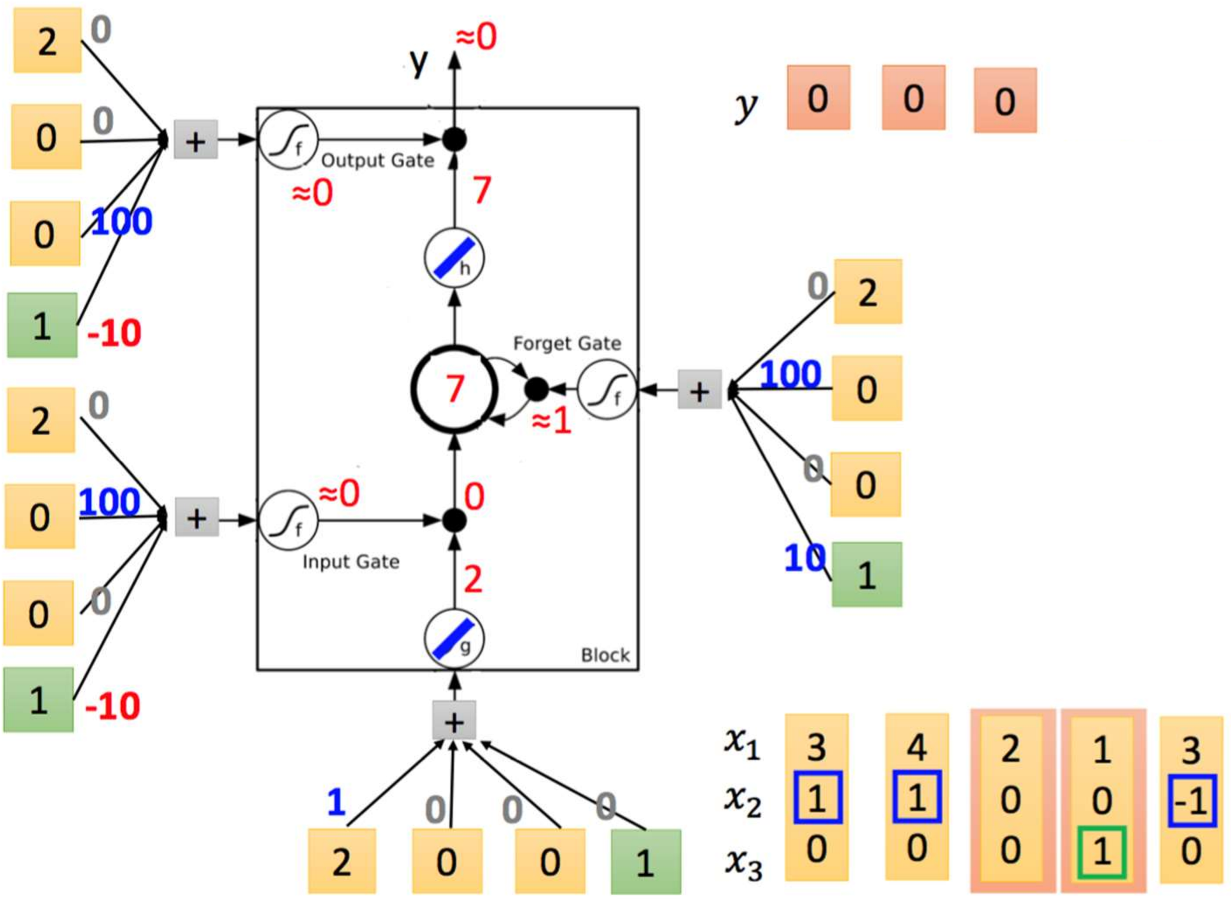




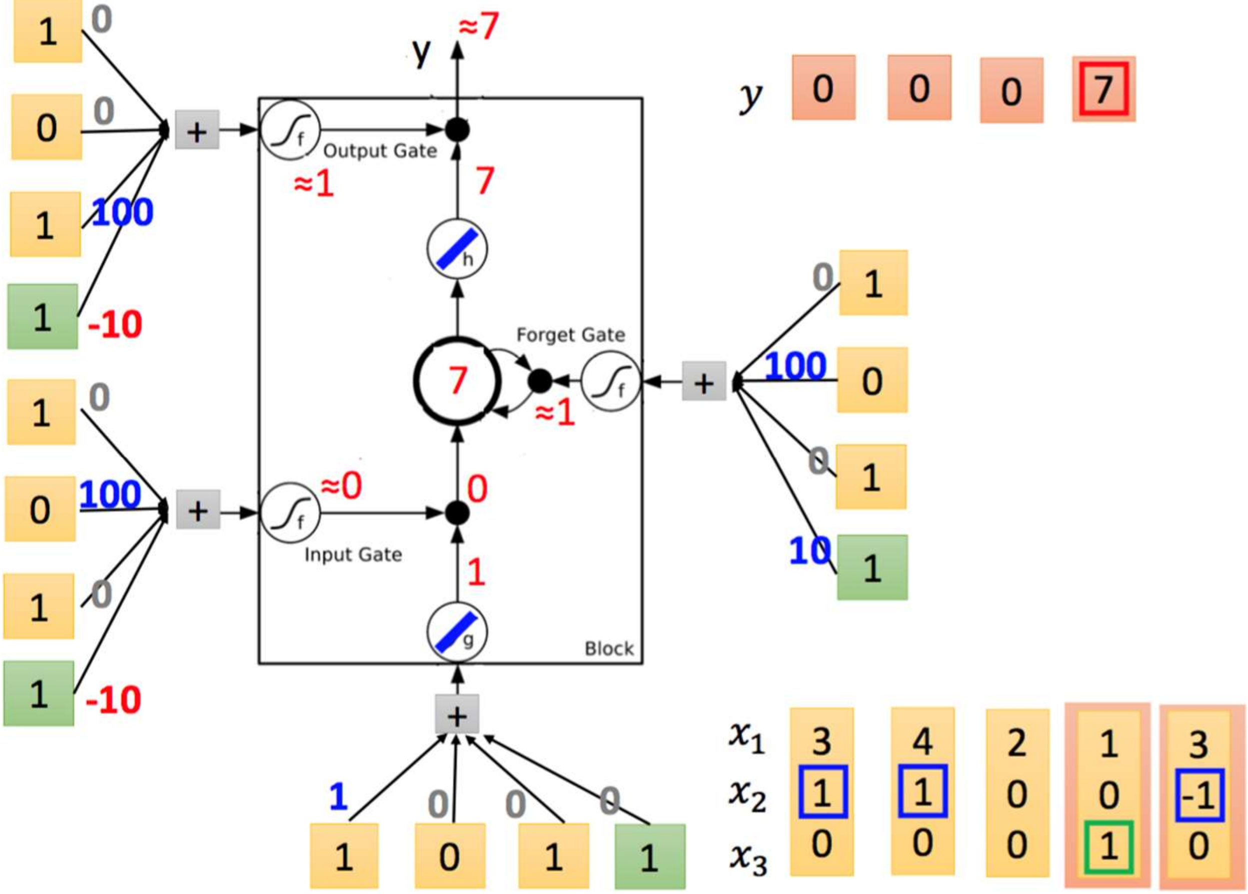
y
≈0
Output Gate
≈0
3
Forget Gate
3
≈1
≈1
3
Input Gate
3
Block
100
-10
10
y 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0



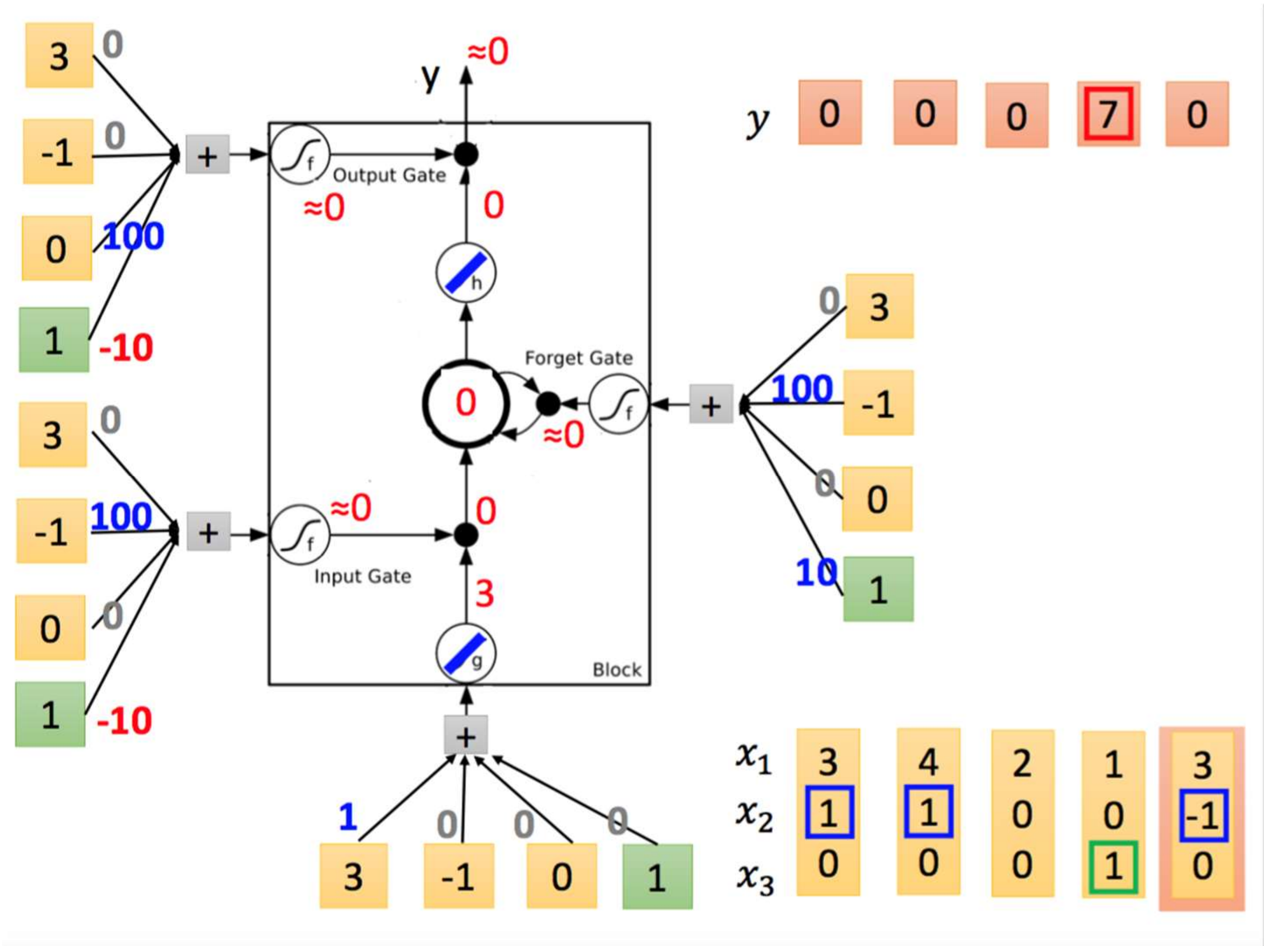
y
≈0
Output Gate
≈0
7
Forget Gate
7
≈1
≈1
4
Input Gate
4
Block
100
-10
10
y 0 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0



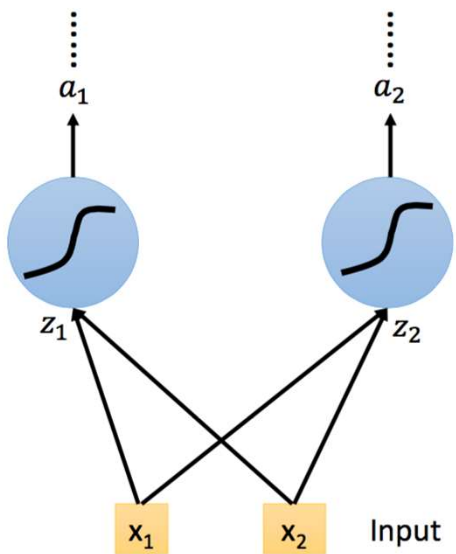
y
≈0
Output Gate
≈0
7
Forget Gate
7
≈1
100
10
≈0
0
Input Gate
2
Block
100
-10
-10
1
x1
x2
x3



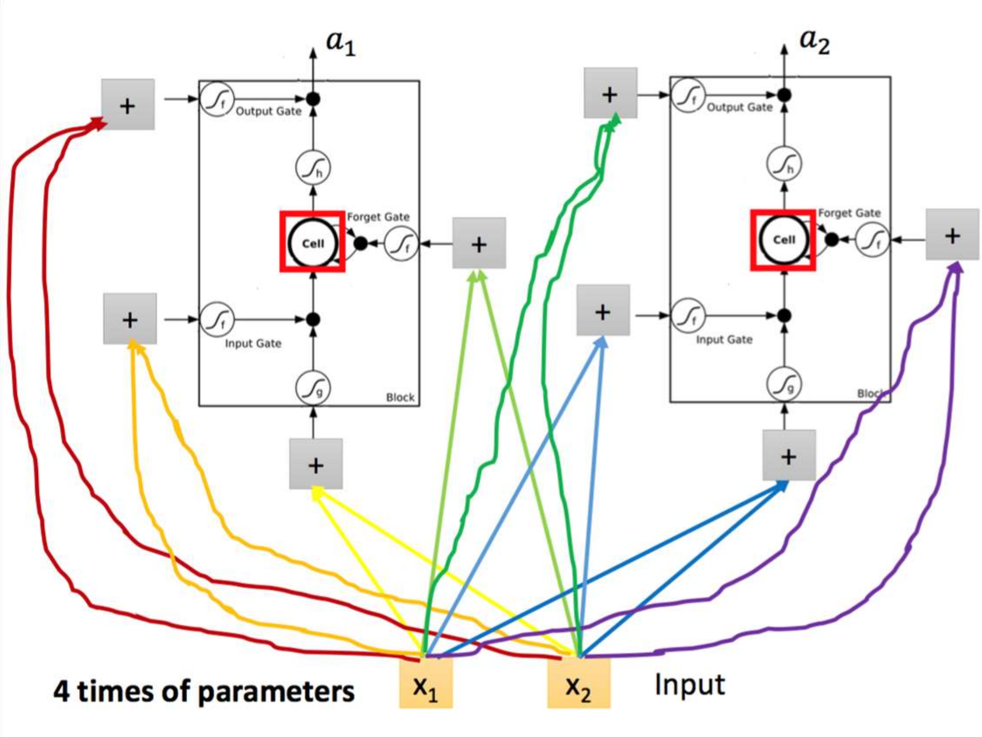
y
≈7
Output Gate
≈1
7
Forget Gate
7
≈1
100
10
≈0
0
Input Gate
1
Block
100
-10
-10
1
x1
x2
x3



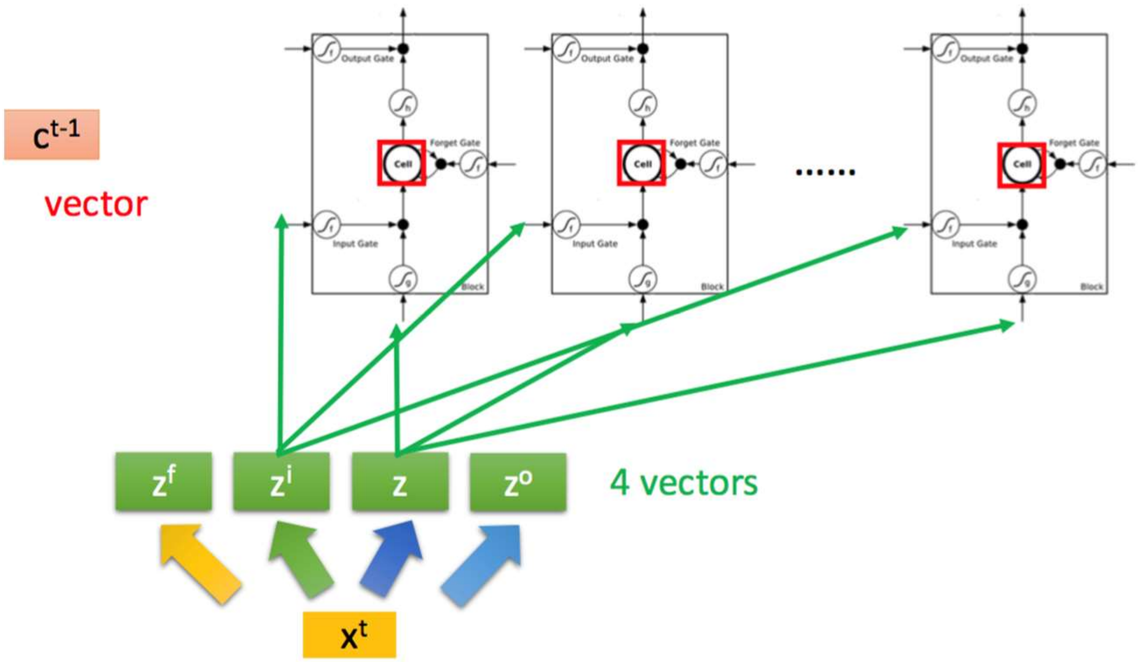
传统NN中的神经元用LSTM神经元代替



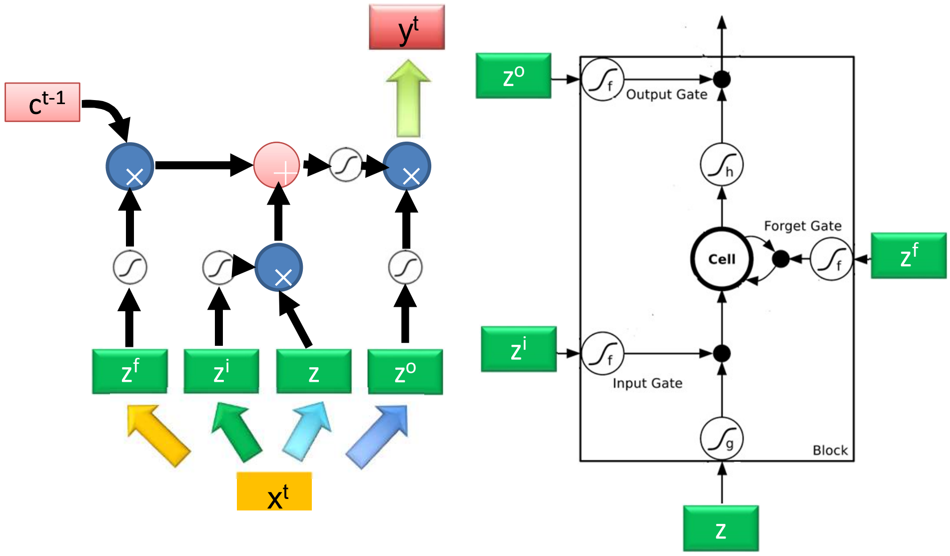
传统NN中的神经元用LSTM神经元代替



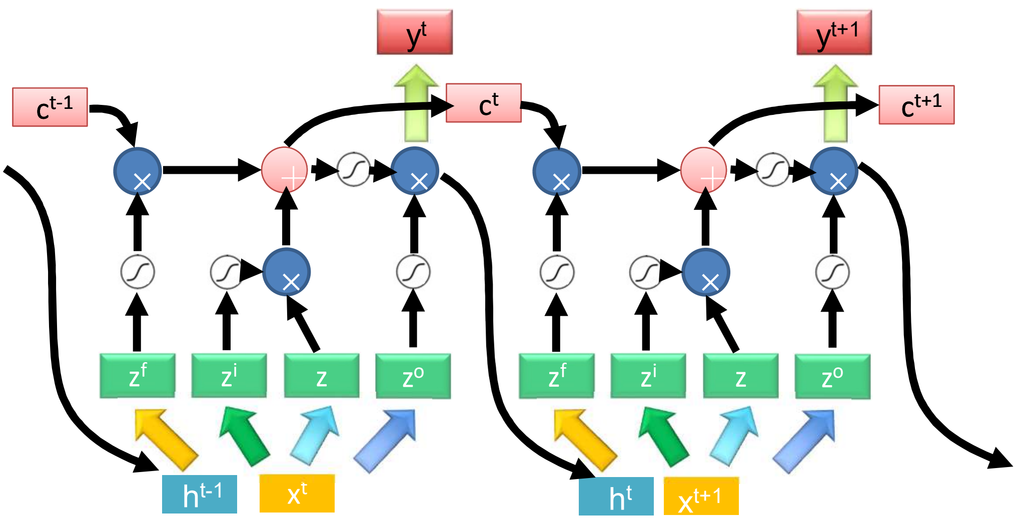
传统NN中的神经元用LSTM神经元代替



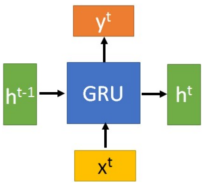
传统NN中的神经元用LSTM神经元代替

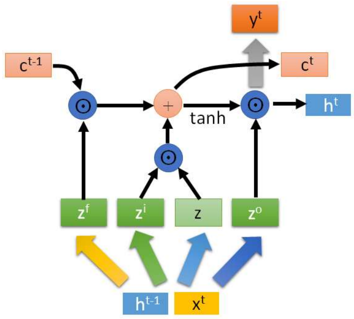


多个Recurrent神经元



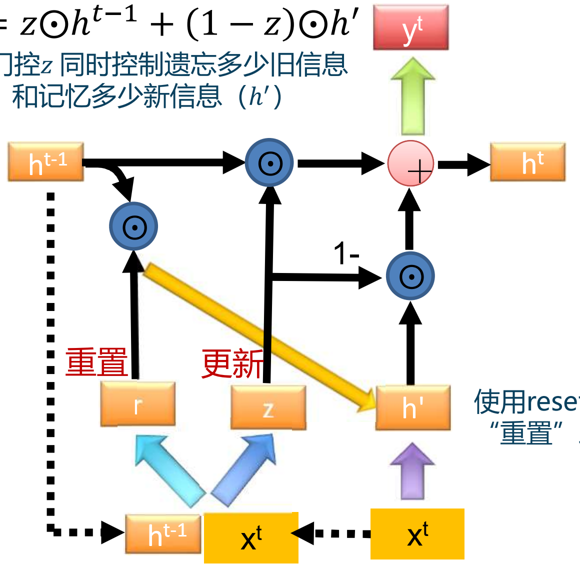
GRU（Gated Recurrent Unit)





$$h^t = z \odot h^{t-1} + (1 - z) \odot h'$$

同一个门控z 同时控制遗忘多少旧信息（h^{t-1}）和记忆多少新信息（h'）

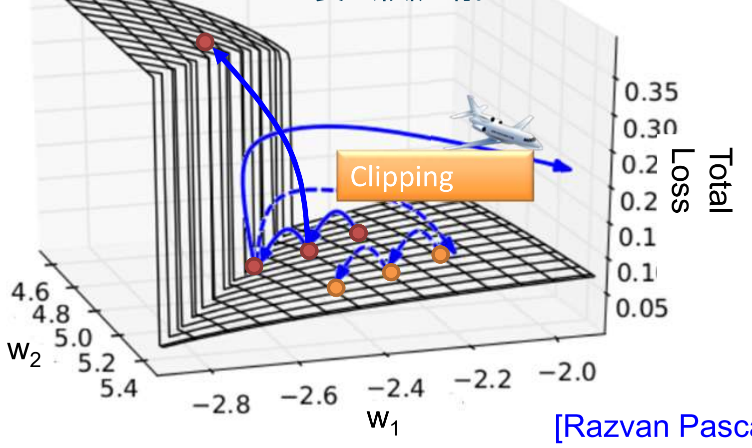


使用reset门控来得到“重置”之后的数据。

重置：历史信息保留多少

训练困难：误差表面分析

误差表面要么很平滑、要么很陡峭。



[Razvan Pascanu, ICML’13]

原因分析

$w = 1 \Rightarrow y^{1000} = 1$

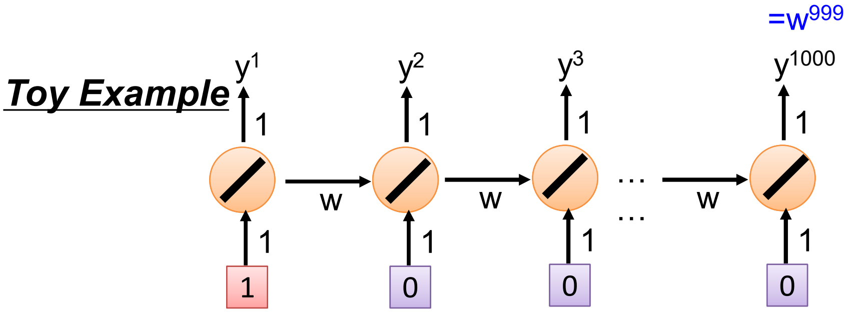
$w = 1.01 \Rightarrow y^{1000} \approx 20000$

$\partial L / \partial w$ 很大 → 小的学习率?

$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

$\partial L / \partial w$ 很小 → 大的学习率?



参考文献

- 卿来云老师，《模式识别和机器学习》讲义
- 李宏毅老师，《Machine Learning》 http://speech.ee.ntu.edu.tw/~tlkagk/courses.html
- LeCun，Hinton等个人主页