厦门大学信息学院 本科选修课
2021-2022 第二学期

模 式 识 别
Pattern Recognition

主 讲：王 程、陈龙彪

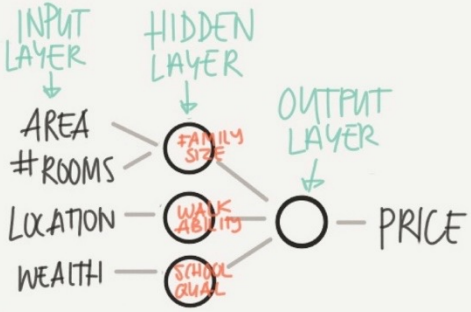
第七章

深度神经网络（DNN）

陈龙彪
longbiaochen@xmu.edu.cn

厦门大学信息学院

INTRO TO DEEP LEARNING



SUPERVISED LEARNING

INPUT: X	OUTPUT: y	NN TYPE
HOME FEATURES	PRICE	STANDARD NN
AD + USER INFO	WILL CLICK ON AD (0/1)	
IMAGE	OBJECT (1...1000)	CONV. NN (CNN)
AUDIO	TEXT TRANSCRIPT	RECURRENT NN (RNN)
ENGLISH	CHINESE	
IMAGE/RADAR	POS OF OTHER CARS	CUSTOM/HYBRID

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA

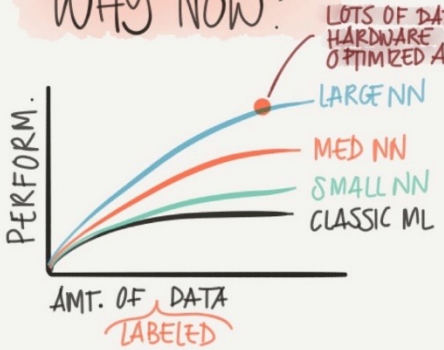
STRUCTURED

"THE QUICK BROWN FOX"

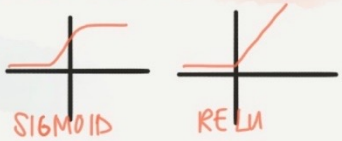
UNSTRUCTURED

HUMANS ARE GOOD AT THIS

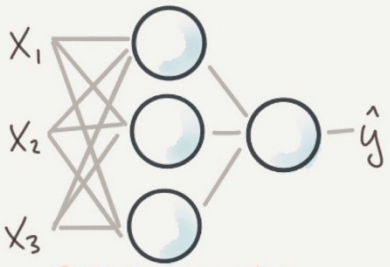
WHY NOW?



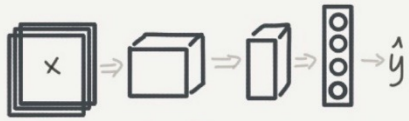
ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT



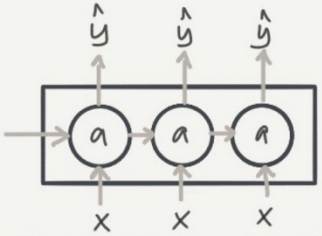
NETWORK ARCHITECTURES



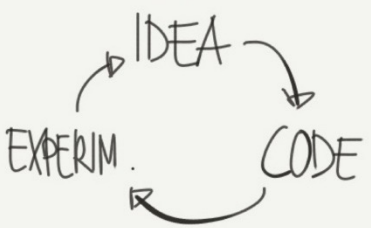
STANDARD NN



CONVOLUTIONAL NN



RECURRENT NN

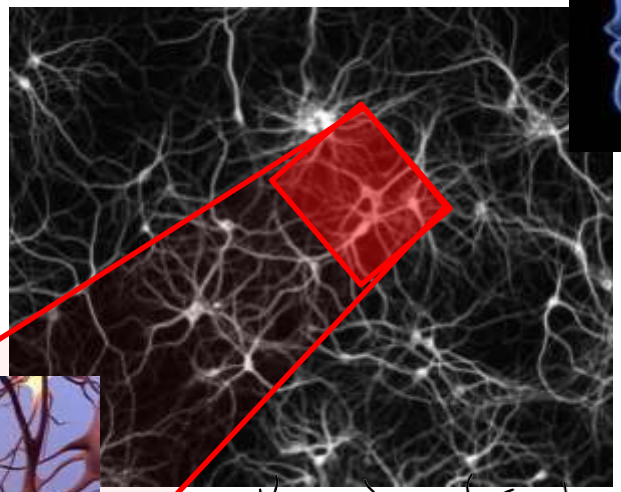


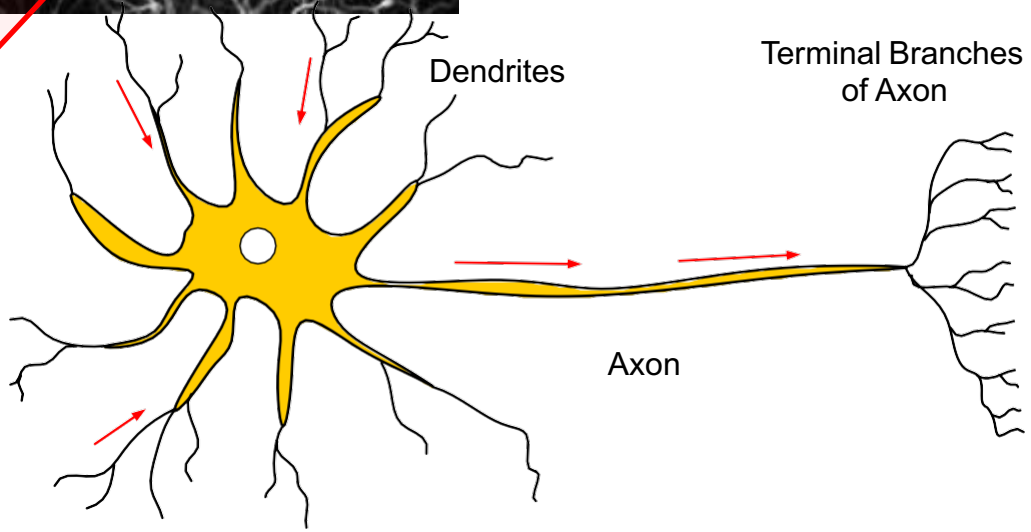
FASTER COMPUTATION IS IMPORTANT TO SPEED UP THE ITERATIVE PROCESS

大纲：深度神经网络（DNN）

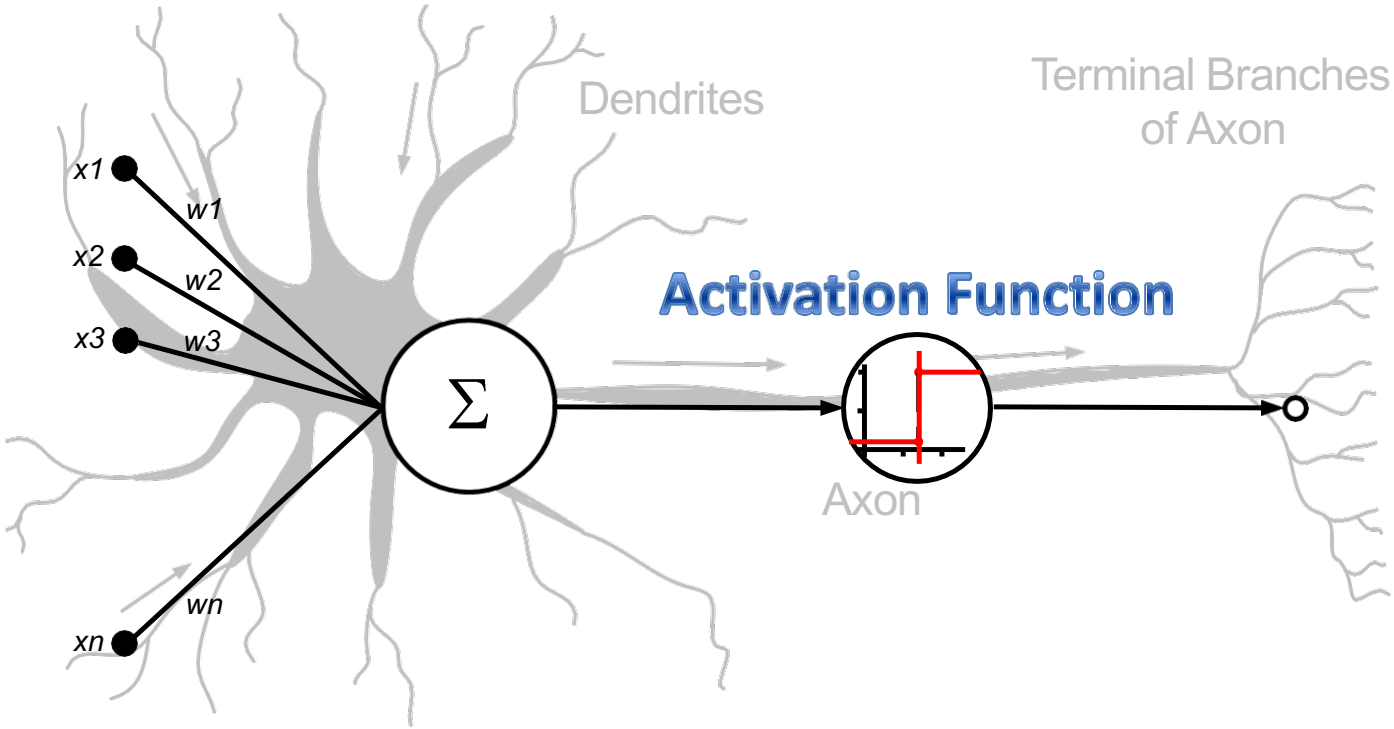
- 1 原理篇
- 2 实例篇
- 3 技巧篇

神经元（生物）



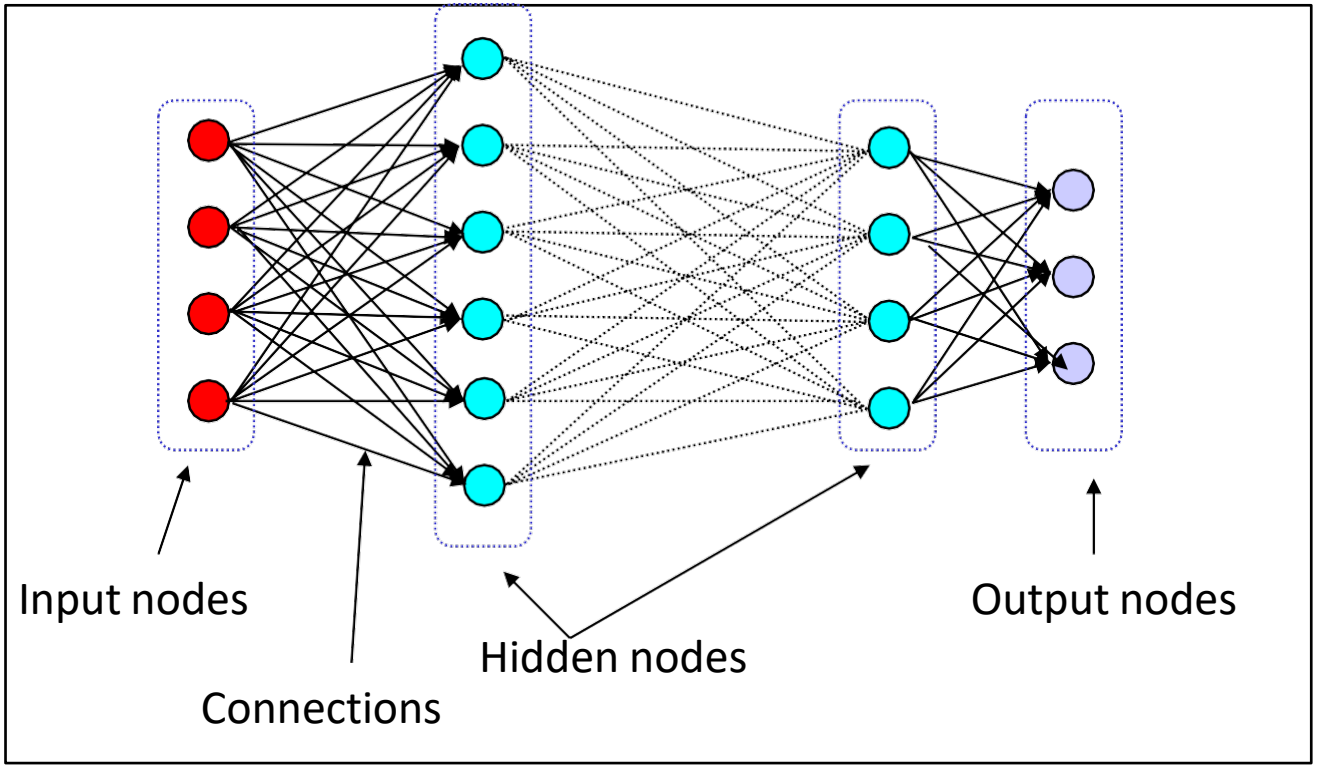


人工神经元



Slide credit : Andrew L. Nelson

人工神经网络



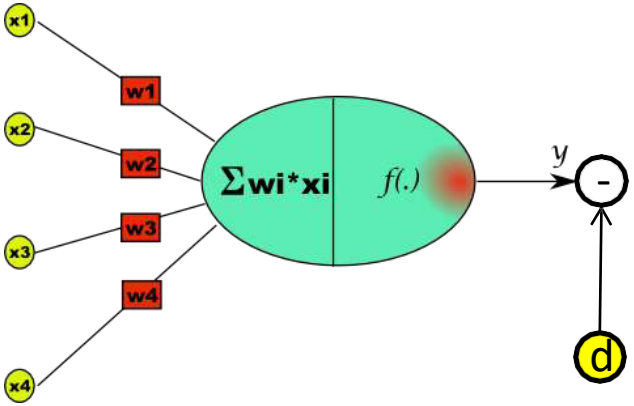
$$Output: \quad y_i = f(w_i^1 x_1 + w_i^2 x_2 + w_i^3 x_3 + \square + w_i^m x_m) = f(\sum_j w_i^j x_j)$$

Slide credit : Geoffrey Hinton

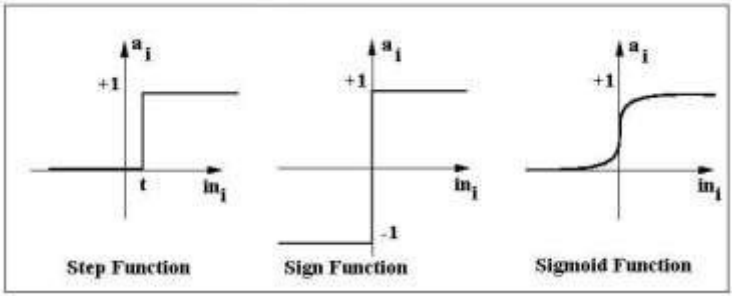
人工神经网络

- The Perceptron was introduced in 1957 by Frank Rosenblatt.

Perceptron:



Activation functions:



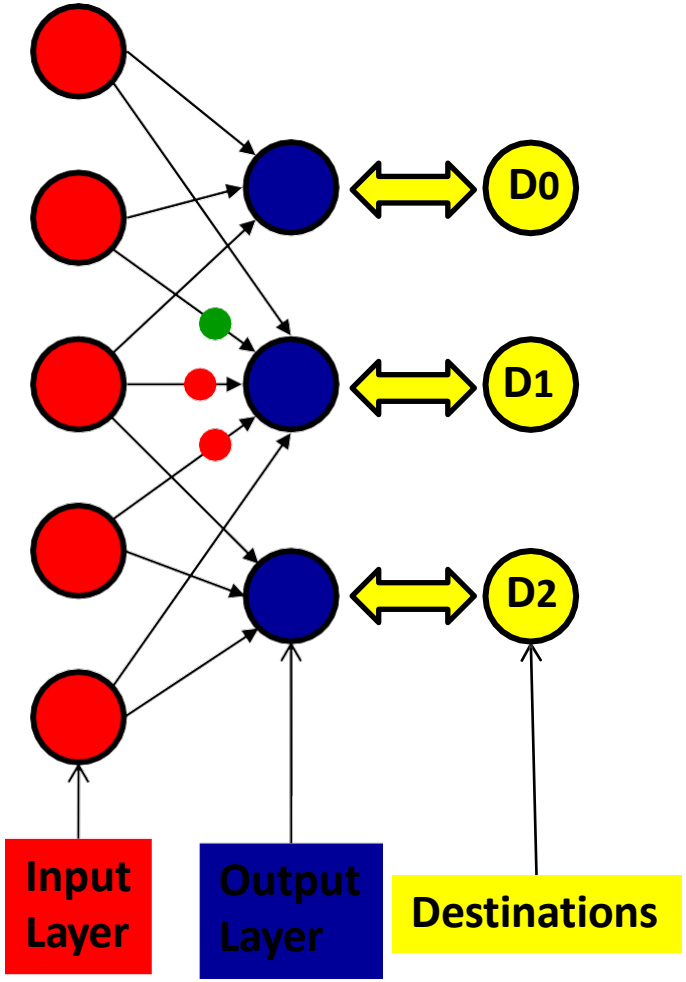
Learning:

$$y^{(t)} = f\left\{\sum_i w_i^{(t)} x_i^{(t)}\right\}$$

Update

$$\Delta w_i^{(t)} = \varepsilon(d^{(t)} - y^{(t)})x_i^{(t)}$$

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)}$$



Slide credit : Geoffrey Hinton

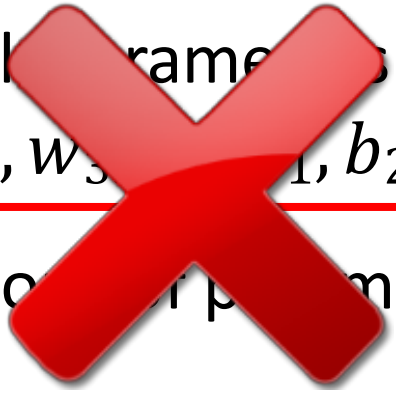
怎样找到最优的网络权重？

找到使总损失最小的参数 θ

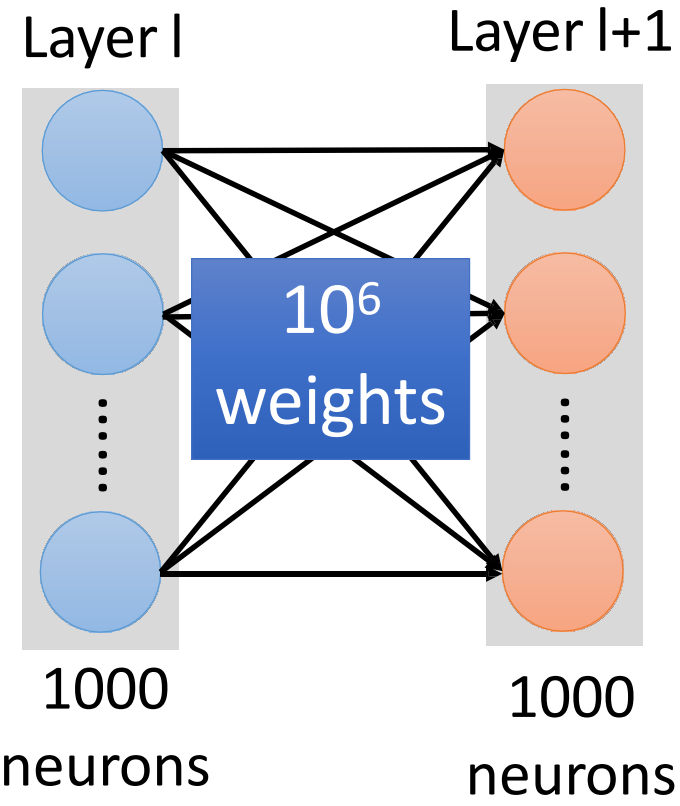
列举所有可能的值

Network parameters $\theta = \{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}$

Millions of parameters



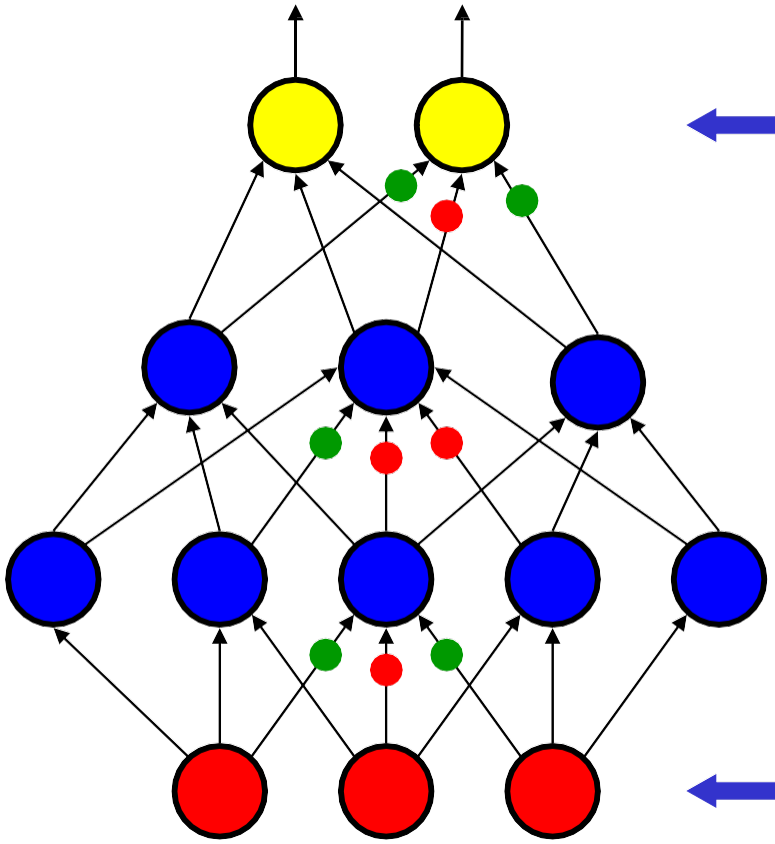
E.g. 语音识别: 8 layers and 1000 neurons each layer



BP神经网络（反向传播）

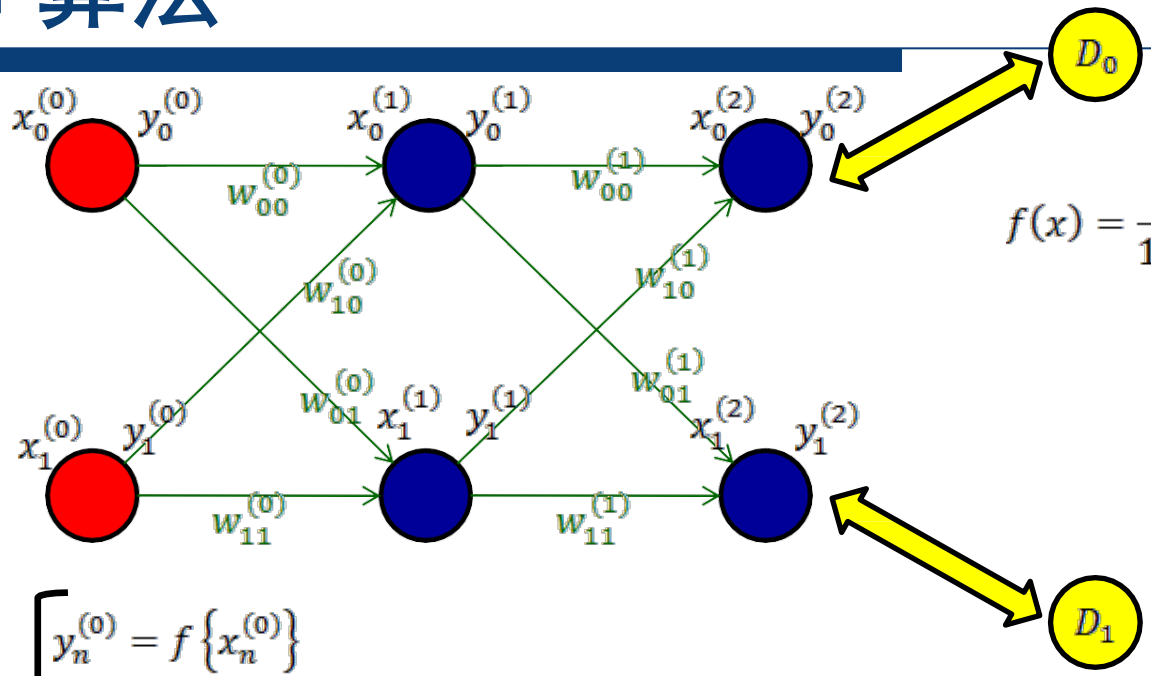
Back-propagate error signal to get derivatives for learning

Compare outputs with correct answer to get error signal

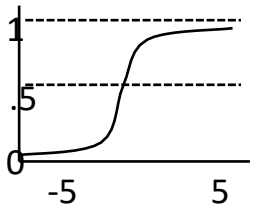


Slide credit : Geoffrey Hinton

BP算法



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



Activations

$$y_n^{(0)} = f\left\{x_n^{(0)}\right\}$$

$$y_n^{(1)} = f\left\{\sum_{i=0}^{1} y_i^{(0)} w_{in}^{(0)}\right\}$$

$$y_n^{(2)} = f\left\{\sum_{i=0}^{1} y_i^{(1)} w_{in}^{(1)}\right\}$$

errors

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = -2\left(D_j - y_j^{(2)}\right) f\left\{x_j^{(2)}\right\}\left(1 - f\left\{x_j^{(2)}\right\}\right) y_i^{(1)} \stackrel{\text{def}}{\equiv} \delta_j^{(1)} y_i^{(1)}$$

$$\frac{\partial E}{\partial w_{ij}^{(0)}} = f\left\{x_j^{(1)}\right\}\left(1 - f\left\{x_j^{(1)}\right\}\right) \sum_{n=0}^{1} \delta_n^{(1)} w_{jn}^{(1)} x_i^{(0)} \stackrel{\text{def}}{\equiv} \delta_j^{(0)} x_i^{(0)}$$

The error:

$$E = \sum_{n=0}^{1} (D_n - y_n^{(2)})^2$$

Update Weights:

$$w_{ij}^{(k)} = w_{ij}^{(k)} + \varepsilon \frac{\partial E}{\partial w_{ij}^{(k)}}$$

Updates

$$w_{ij}^{(1)} = w_{ij}^{(1)} + \varepsilon \delta_j^{(1)} y_i^{(1)}$$

$$w_{ij}^{(0)} = w_{ij}^{(0)} + \varepsilon \delta_j^{(0)} x_i^{(0)}$$

Slide credit : Geoffrey Hinton

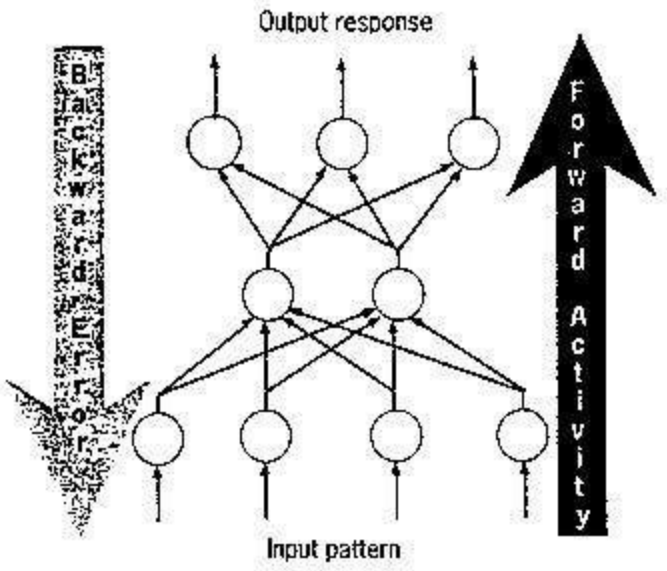
BP算法

Advantages

- Multi layer Perceptron network can be trained by the back propagation algorithm to perform any mapping between the input and the output.

What is wrong with back-propagation?

- It requires labeled training data.
  Almost all data is unlabeled.
- The learning time does not scale well
  It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.



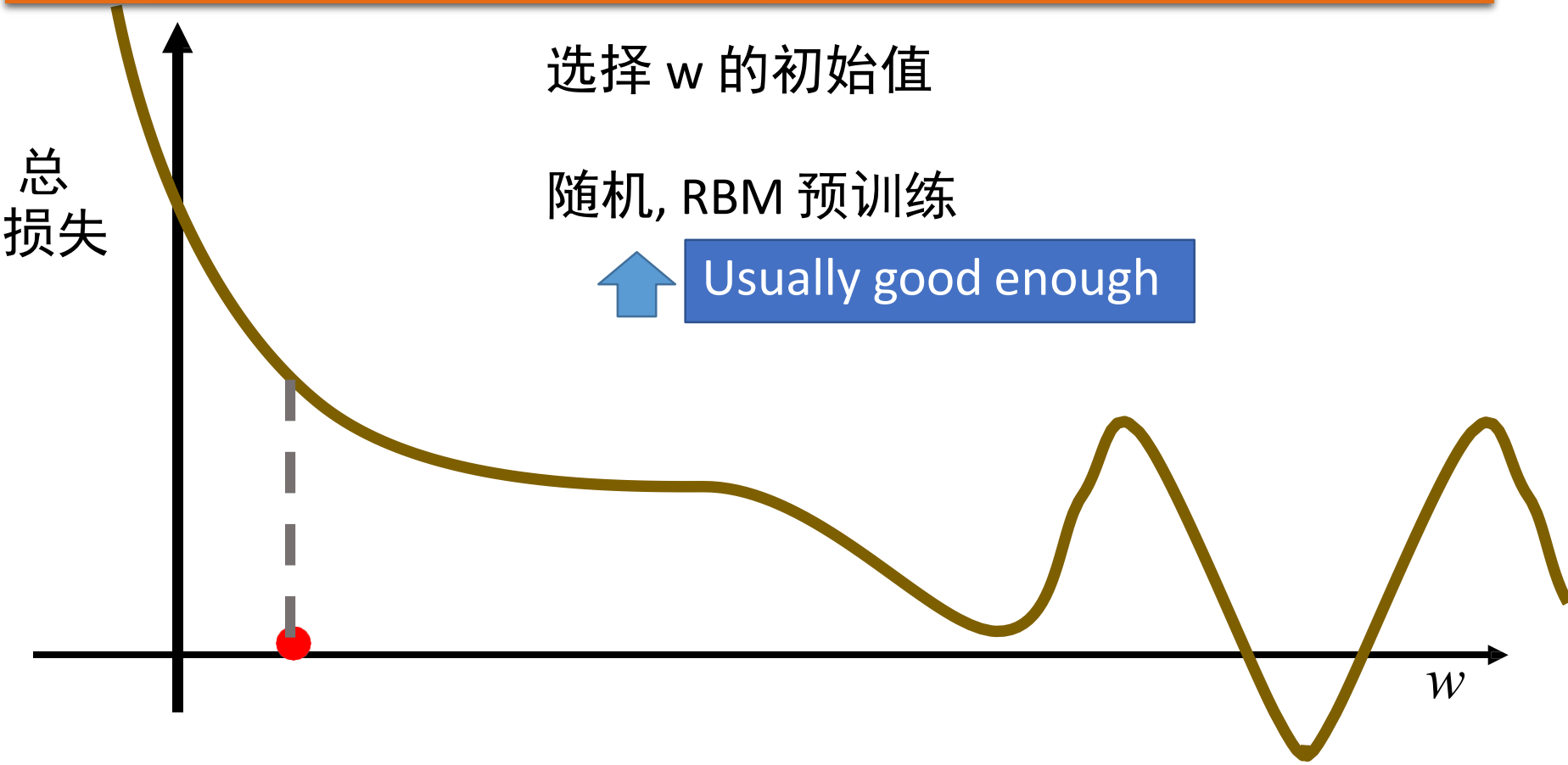
A backpropagation network trains with a two-step procedure. The activity from the input pattern flows forward through the network, and the error signal flows backward to adjust the weights.

Slide credit : Geoffrey Hinton

梯度下降

参数 $\theta = \{w1, w2, \cdots, b1, b2, \cdots\}$

找到使总损失最小的参数θ

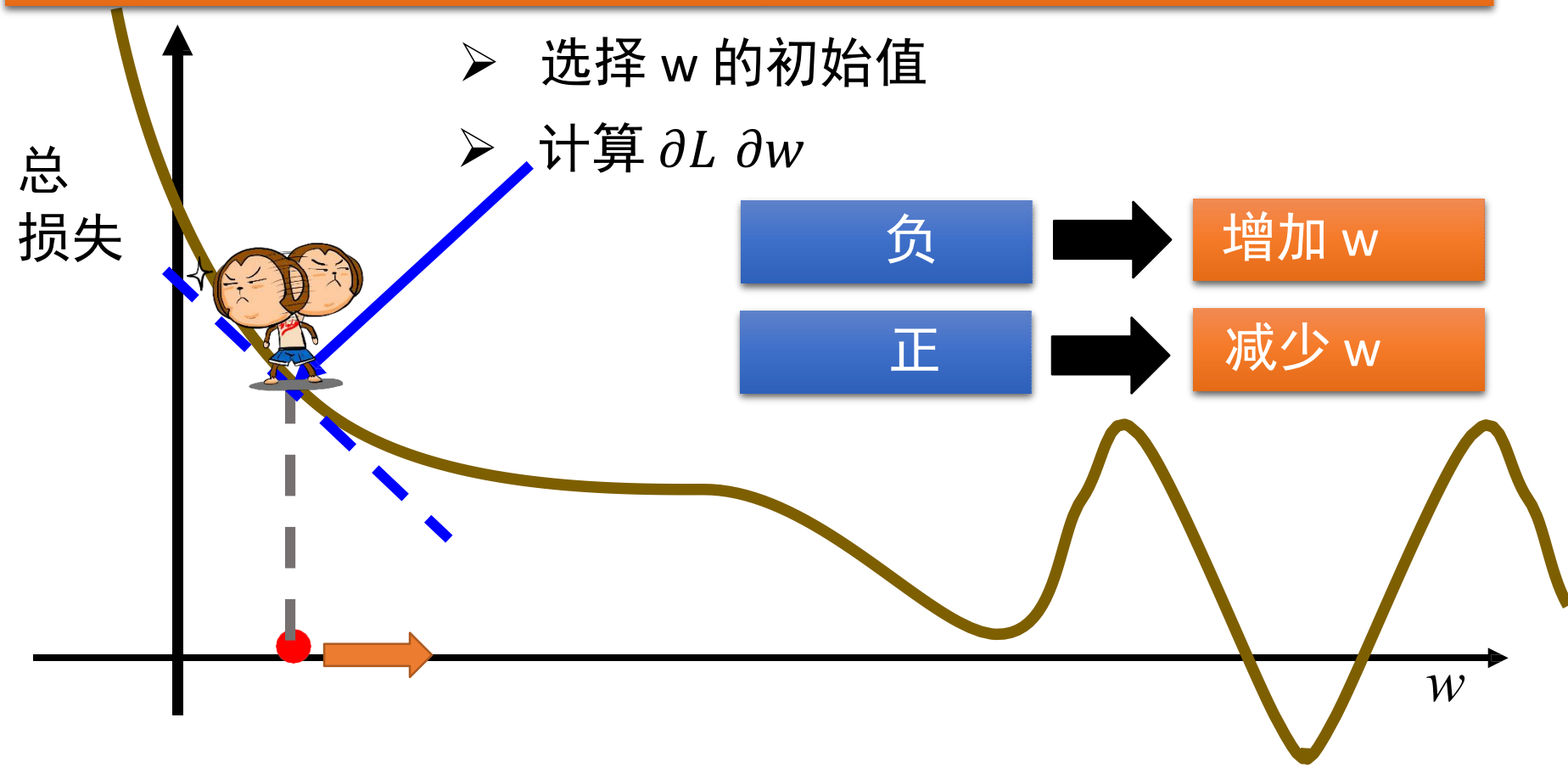


梯度下降

参数 $\theta = \{w1, w2, \cdots, b1, b2, \cdots\}$

找到使总损失最小的参数θ^*

- 选择 w 的初始值
- 计算 $\partial L / \partial w$



梯度下降

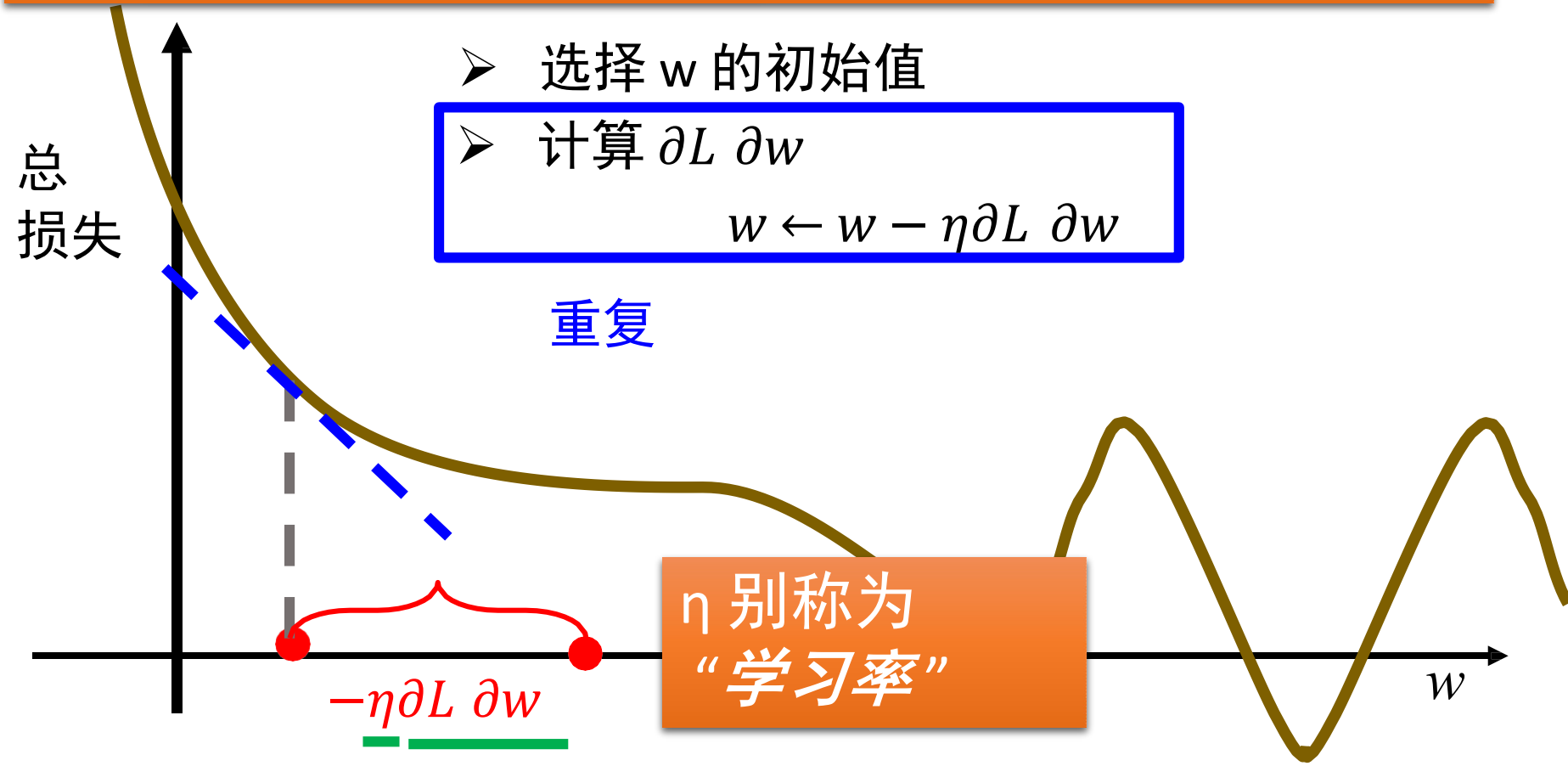
参数 $\theta = \{w1, w2, \cdots, b1, b2, \cdots\}$

找到使总损失最小的参数θ^*

- 选择 w 的初始值
- 计算 $\partial L / \partial w$

$$w \leftarrow w - \eta \partial L / \partial w$$

重复



梯度下降

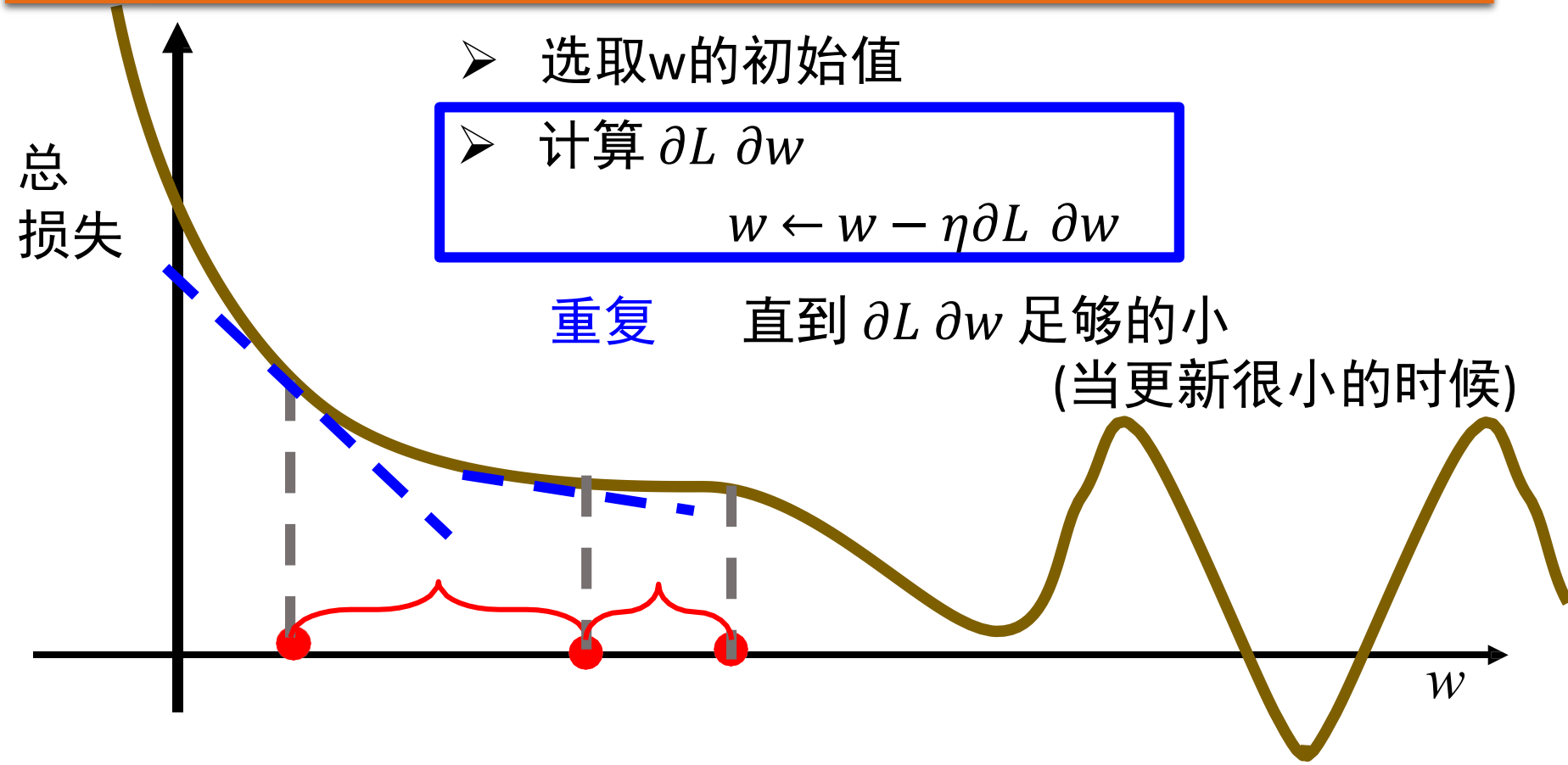
参数 $\theta = \{w1, w2, \cdots, b1, b2, \cdots\}$

找到使总损失最小的参数θ^*

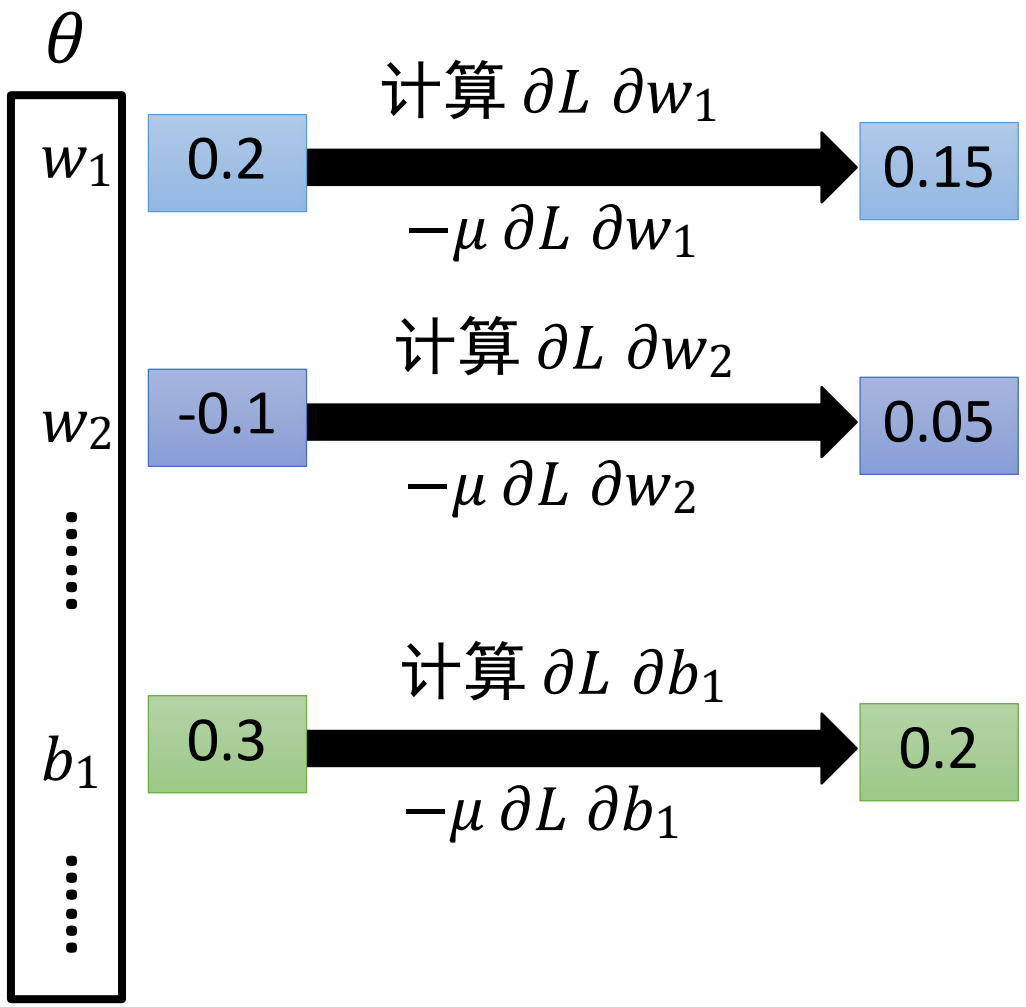
- 选取w的初始值
- 计算 $\partial L / \partial w$

$$w \leftarrow w - \eta \partial L / \partial w$$

重复　　直到 $\partial L / \partial w$ 足够的小
(当更新很小的时候)

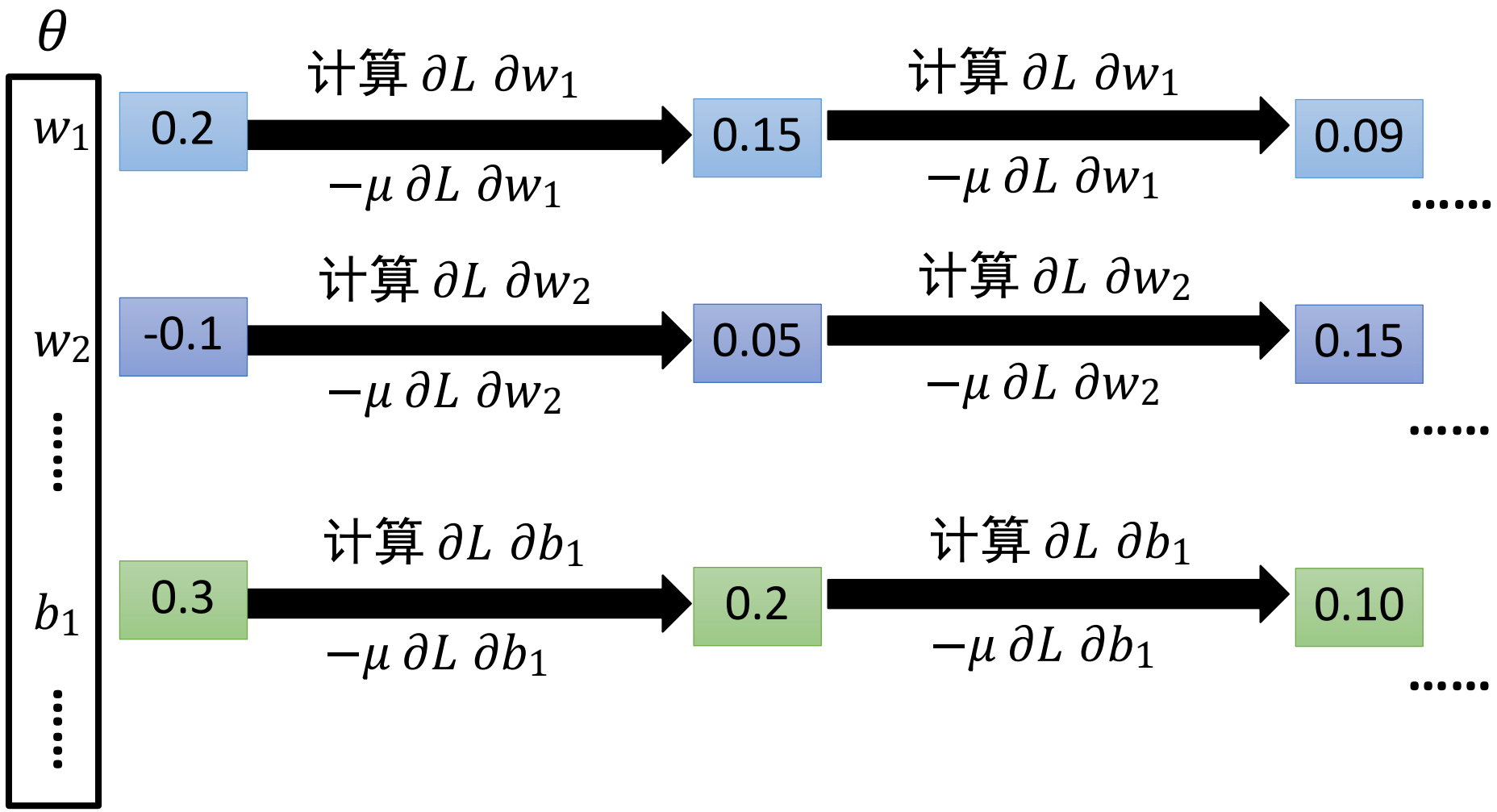


梯度下降

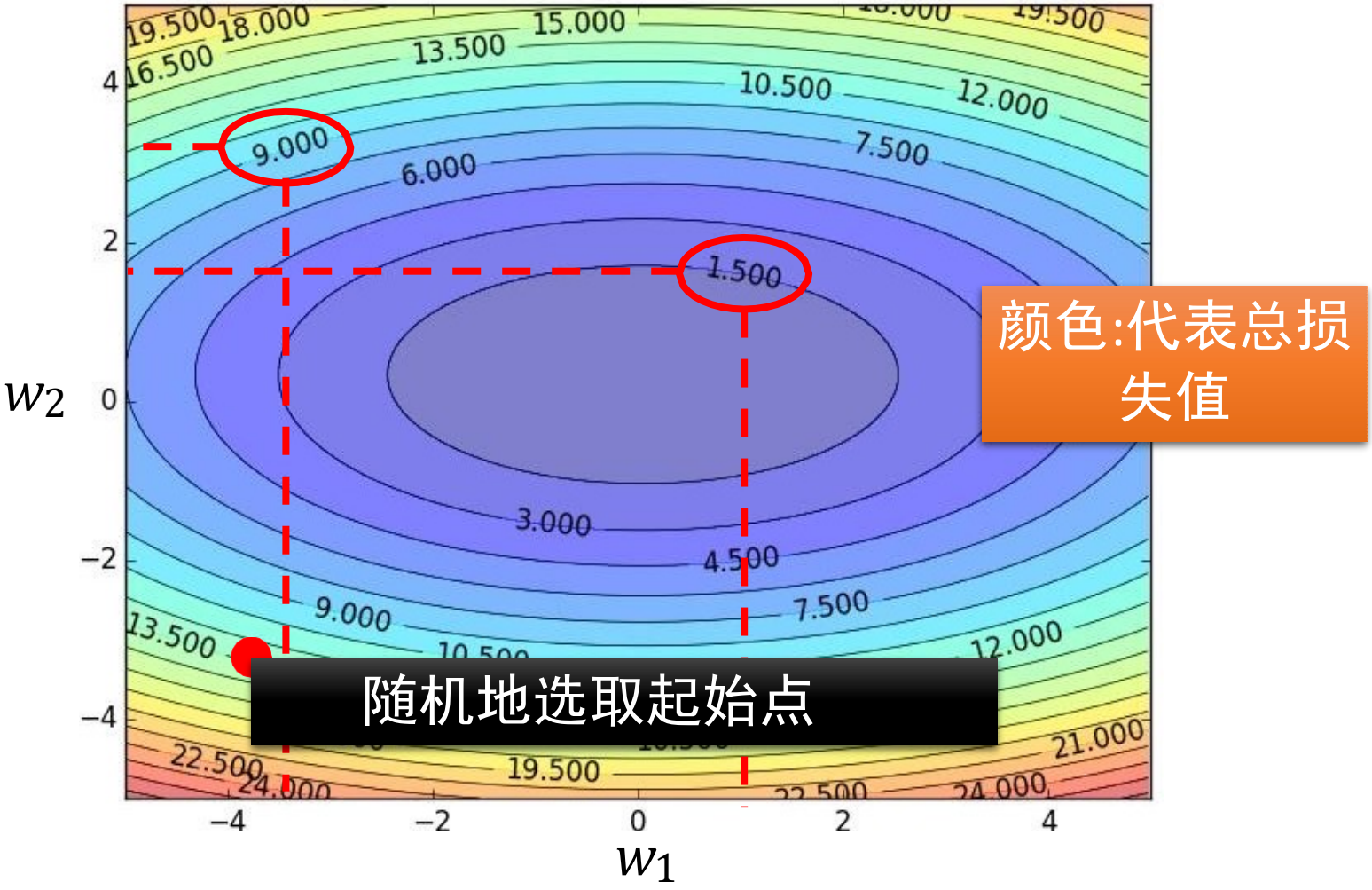


$$\nabla L = \text{梯度} \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

梯度下降

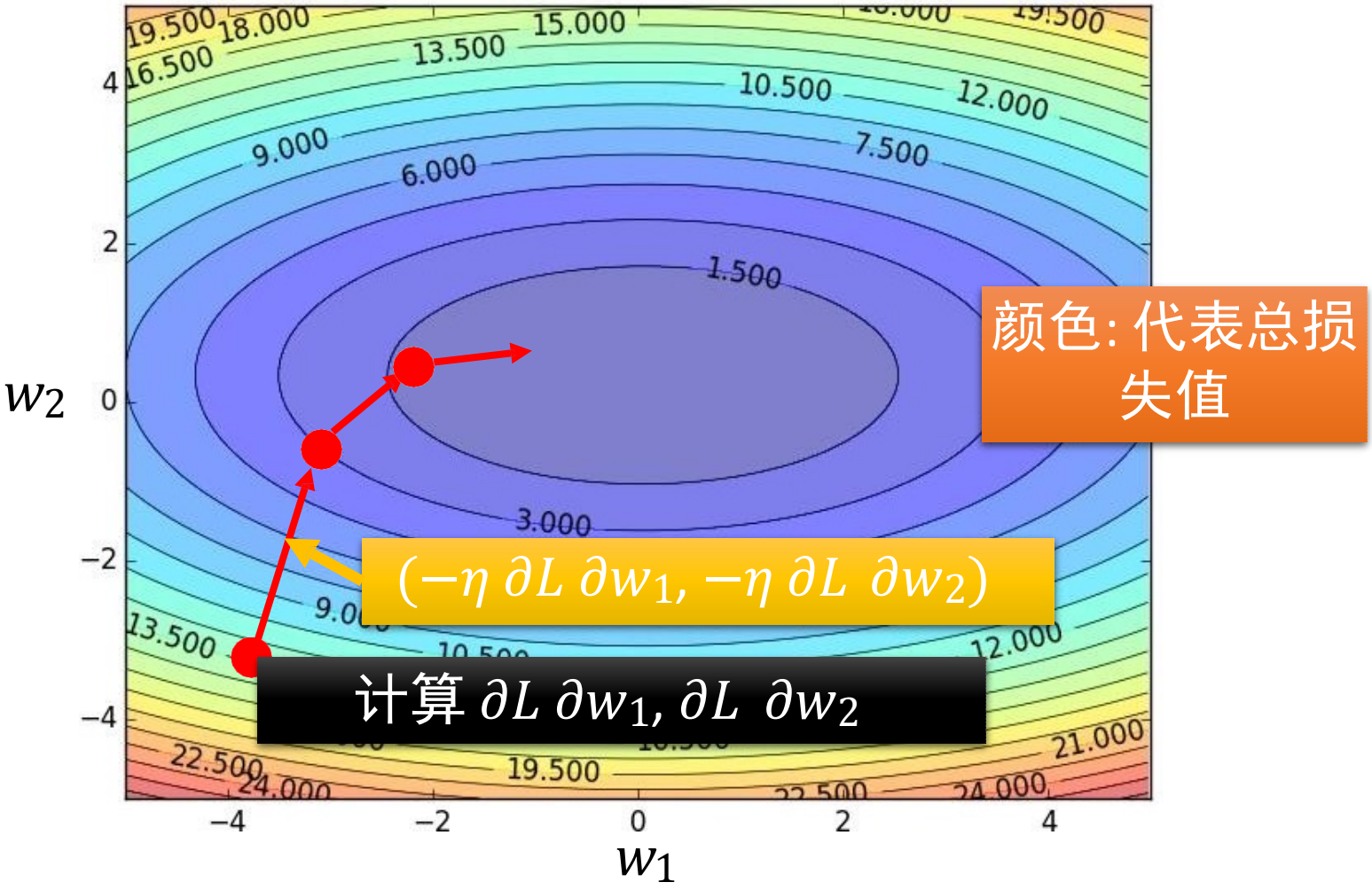

θ
w_1
w_2
b_1
0.2
计算 ∂L ∂w_1
$-\mu\,\partial L$ ∂w_1
0.15
计算 ∂L ∂w_1
$-\mu\,\partial L$ ∂w_1
0.09
-0.1
计算 ∂L ∂w_2
$-\mu\,\partial L$ ∂w_2
0.05
计算 ∂L ∂w_2
$-\mu\,\partial L$ ∂w_2
0.15
0.3
计算 ∂L ∂b_1
$-\mu\,\partial L$ ∂b_1
0.2
计算 ∂L ∂b_1
$-\mu\,\partial L$ ∂b_1
0.10

梯度下降


19.500
18.000
16.500
15.000
13.500
10.500
12.000
7.500
9.000
6.000
1.500
3.000
4.500
9.000
7.500
13.500
12.000
22.500
19.500
21.000
24.000
4
2
0
−2
−4
−4
−2
0
2
4
w_2
w_1
颜色:代表总损失值
随机地选取起始点

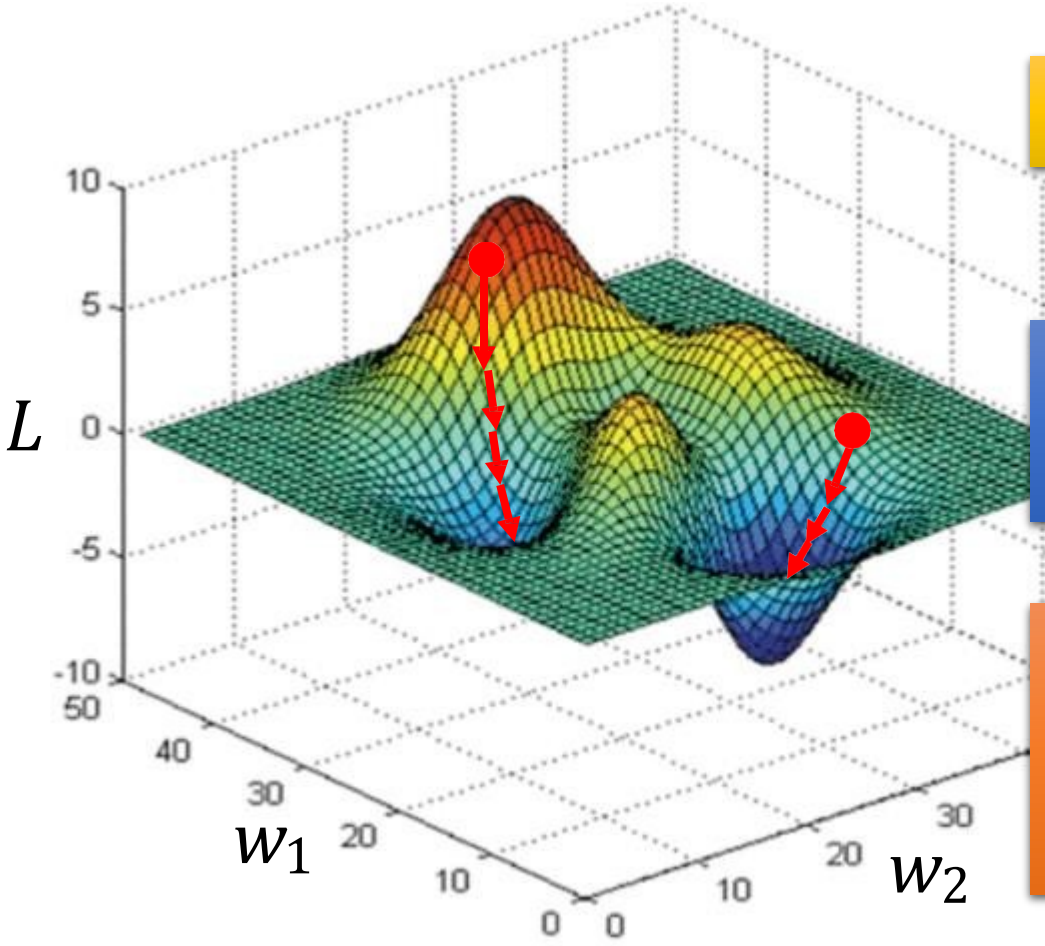
梯度下降

可以收敛到最小值.....



梯度下降

- 梯度下降法不能保证全局最小值



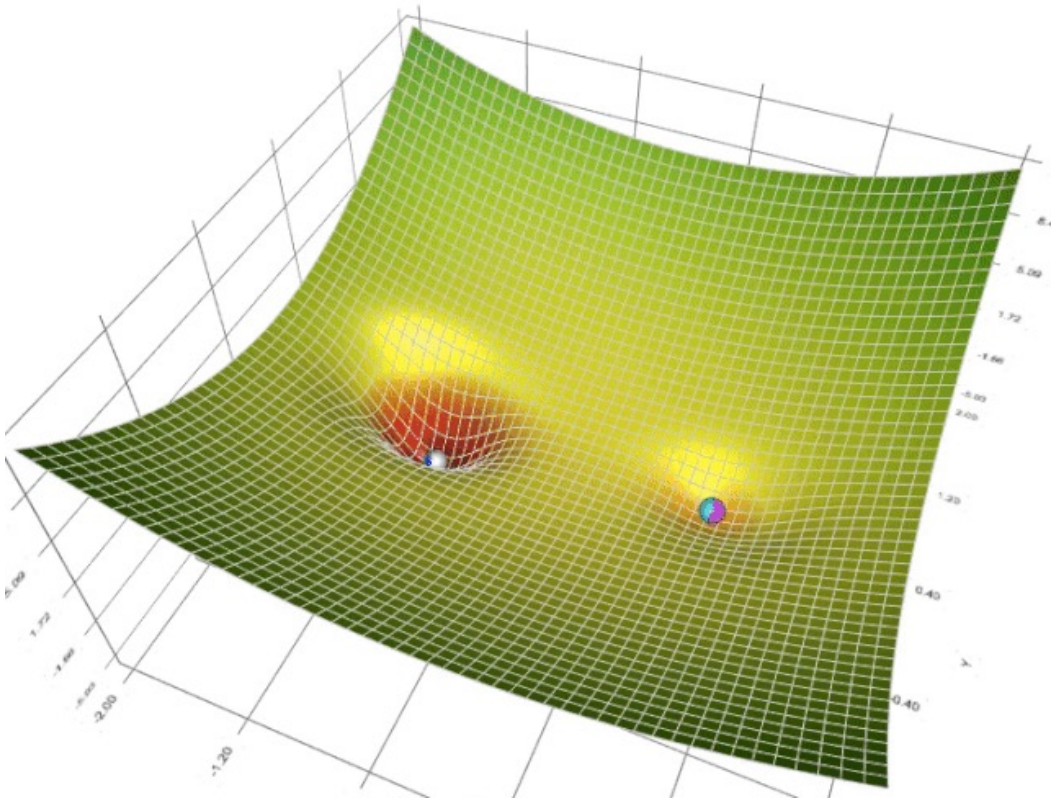
不同的初始值

到达不同的最小值，所以有不同的结果

有一些技巧可以帮助避免局部最小值，但不能保证

梯度下降

- 动画演示
  - https://www.bilibili.com/video/BV1Db4y1n7VW/



https://github.com/lilipads/gradient_descent_viz

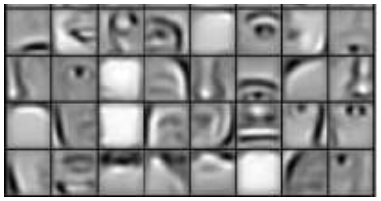
梯度下降

- 成熟的梯度下降工具
  - 高效计算 $\partial L\ \partial w$

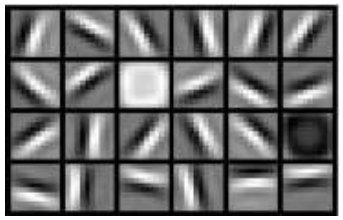
不用担心 $\partial L\ \partial w$, 有专门的工具去解决

深度神经网络

object
models



object parts
(combination of edges)



edges

pixels

深度神经网络

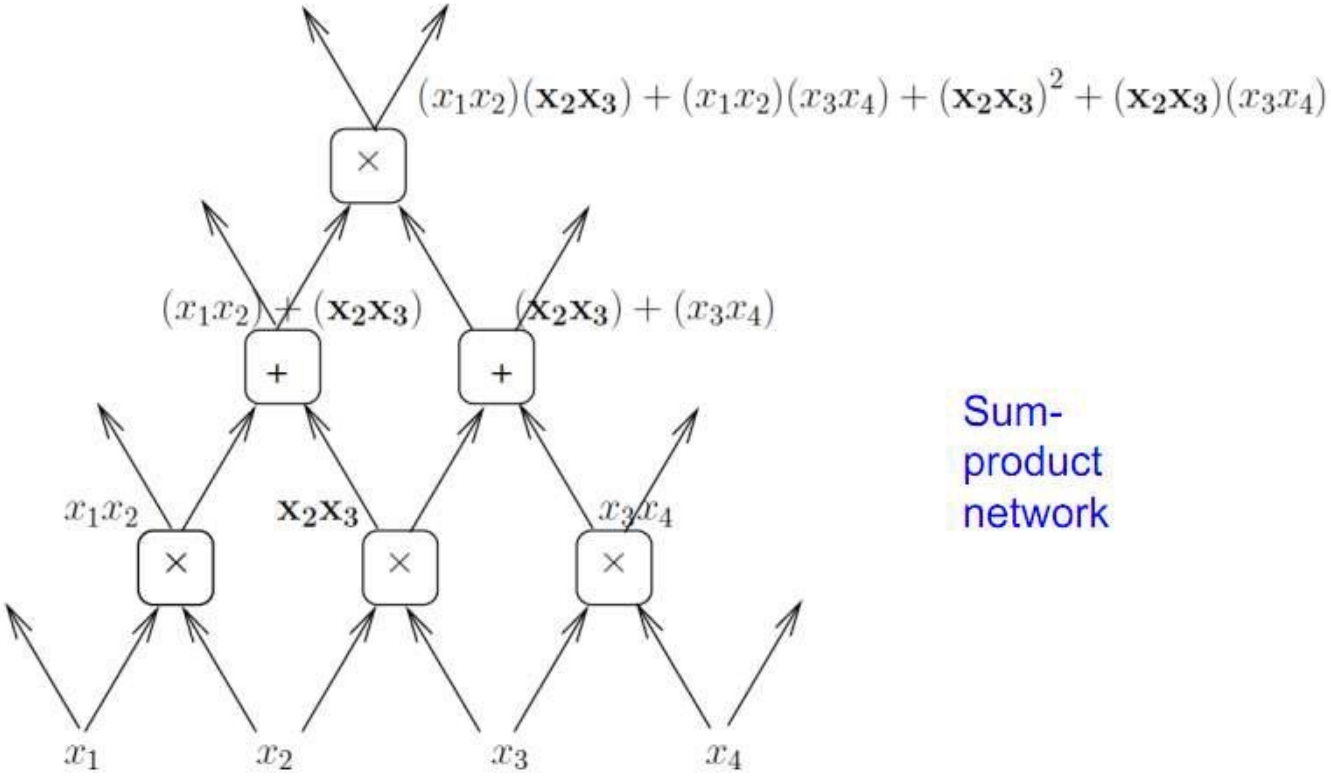
- **Standard learning strategy**
  - Randomly initializing the weights of the network
  - Applying gradient descent using backpropagation
- **But, backpropagation does not work well** (if randomly initialized)
  - Deep networks trained with back-propagation (without unsupervised pre-train) perform worse than shallow networks
  - ANN have limited to one or two layers

深度神经网络

Sharing Components in a Deep Architecture

Polynomial expressed with shared components:

advantage of depth may grow exponentially



Sum-product network

However…

Before 2006

Failing to train deep architectures

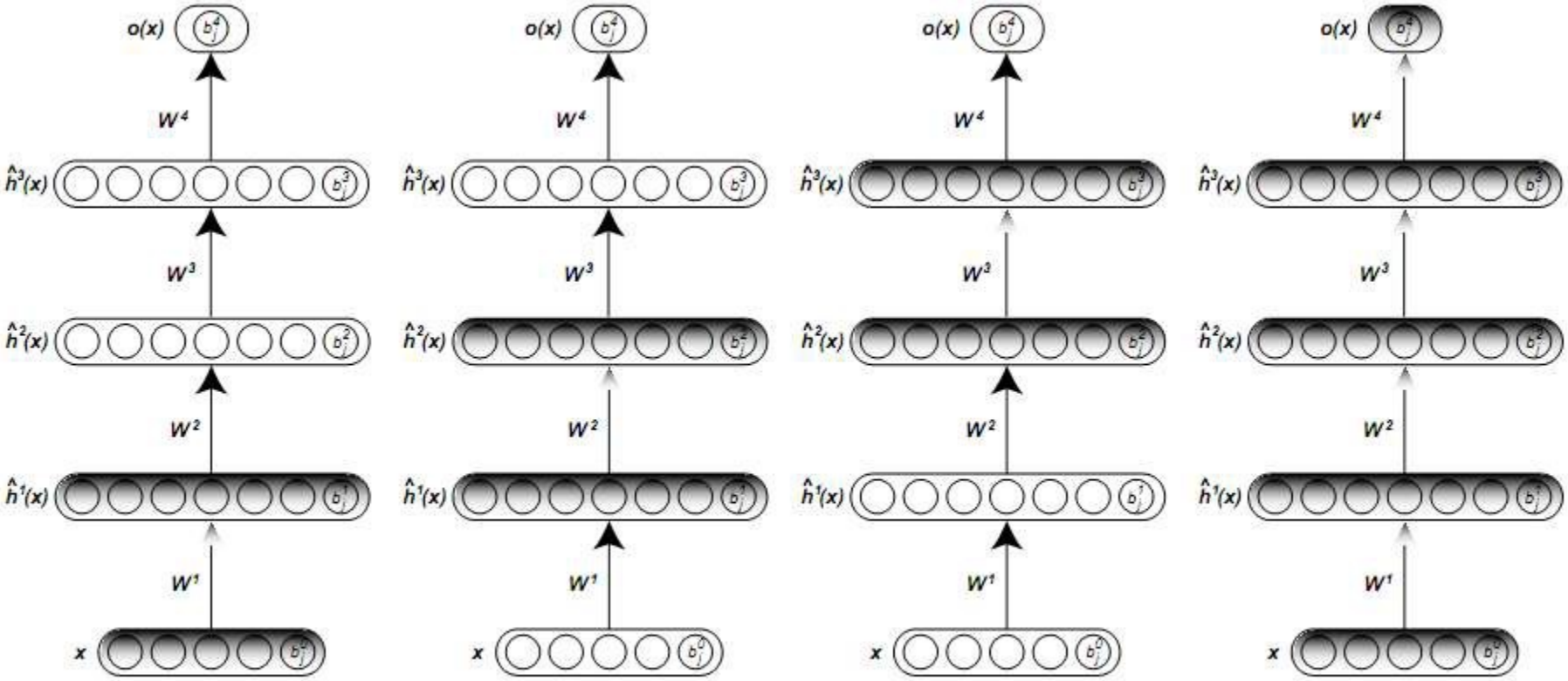
2006

Breakthrough!

2006: The Deep Breakthrough

- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

无监督式分层训练



(a) First hidden layer pre-training

(b) Second hidden layer pre-training

(c) Third hidden layer pre-training

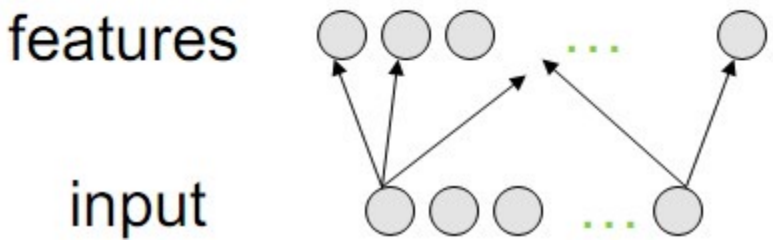
(d) Fine-tuning of whole network

无监督式分层训练

Deep training

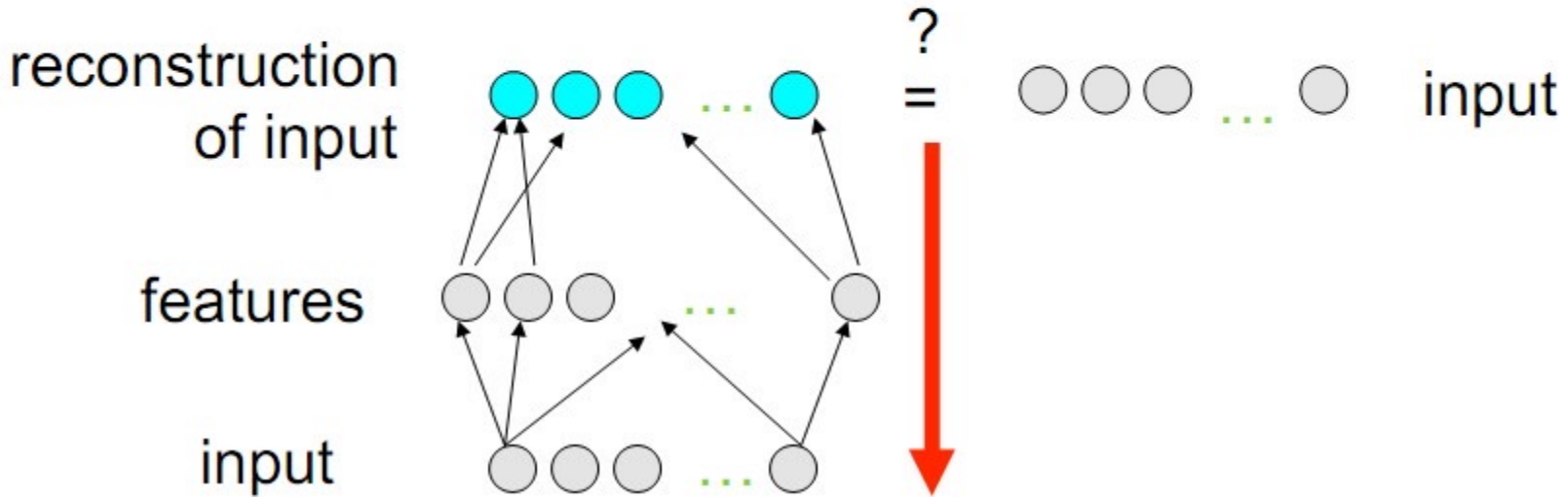
无监督式分层训练

Layer-Wise Unsupervised Pre-training



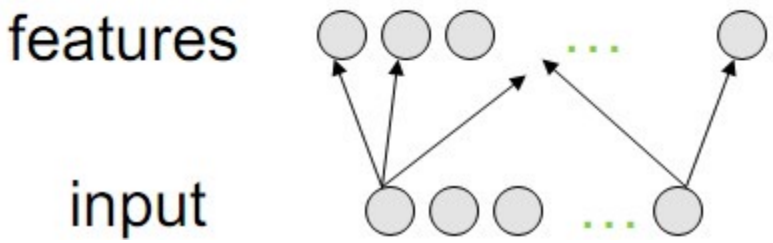
无监督式分层训练

Layer-Wise Unsupervised Pre-training



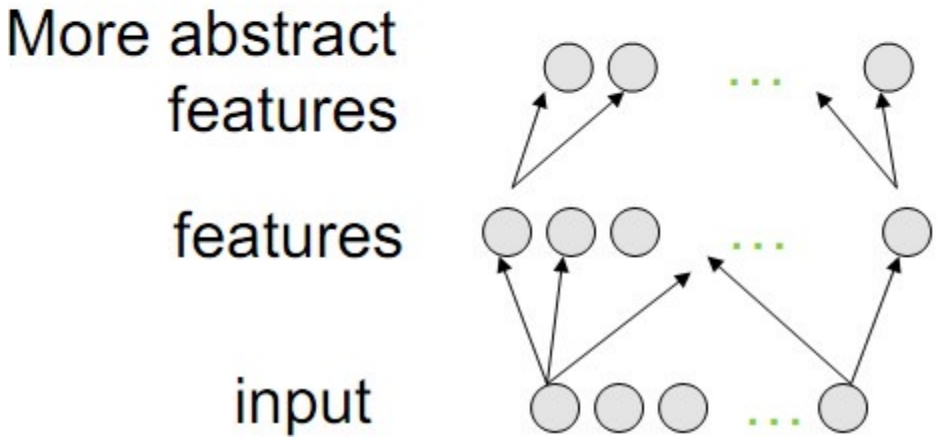
无监督式分层训练

Layer-Wise Unsupervised Pre-training



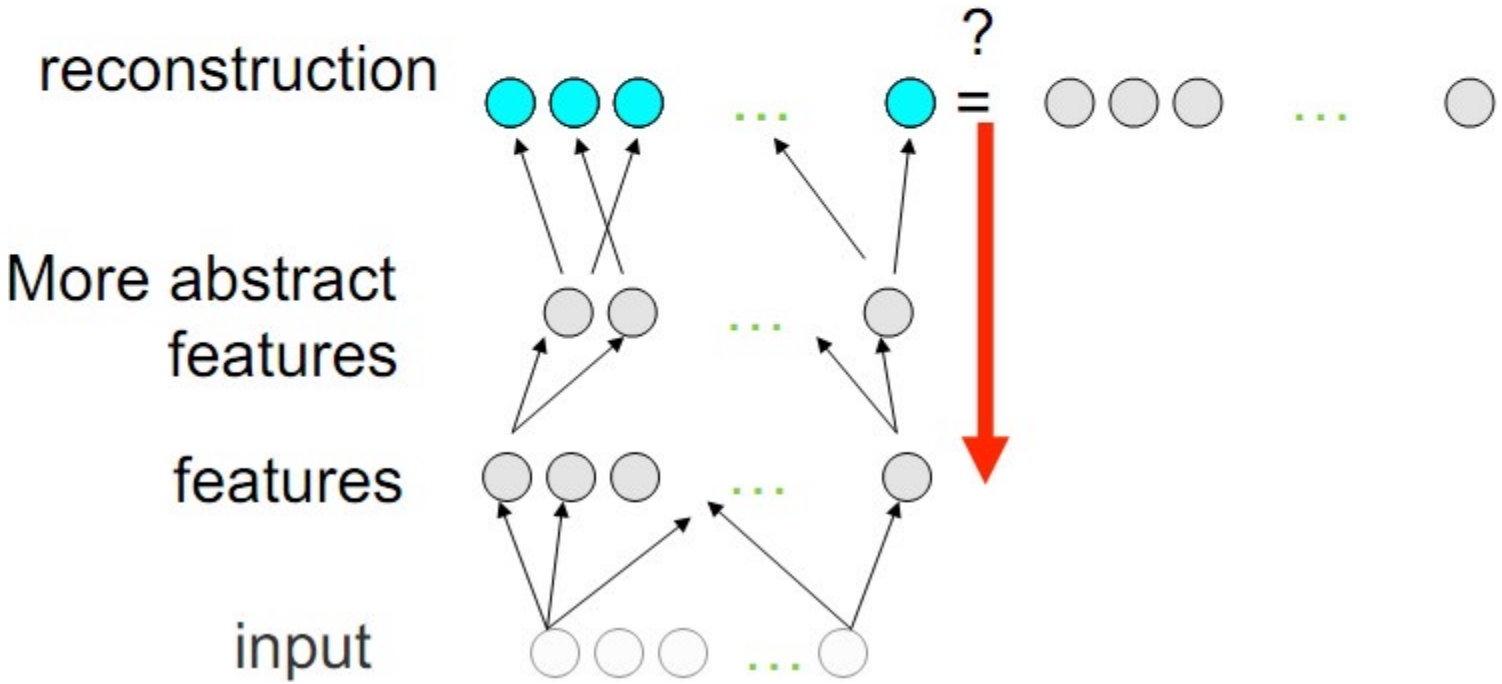
无监督式分层训练

Layer-Wise Unsupervised Pre-training



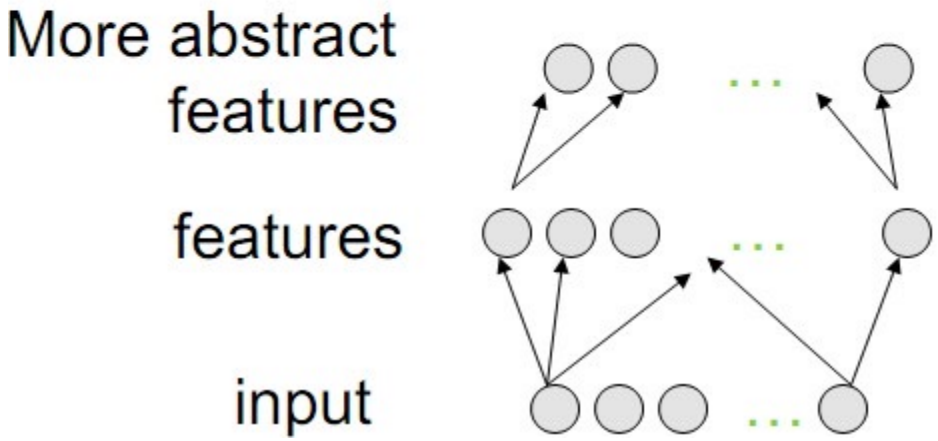
无监督式分层训练

Layer-Wise Unsupervised Pre-training



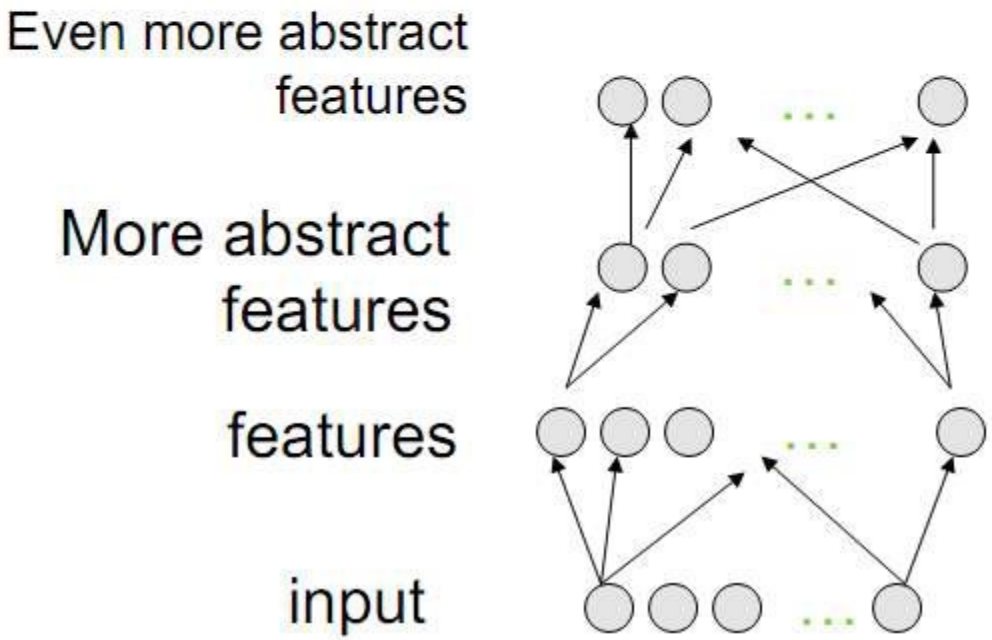
无监督式分层训练

Layer-Wise Unsupervised Pre-training

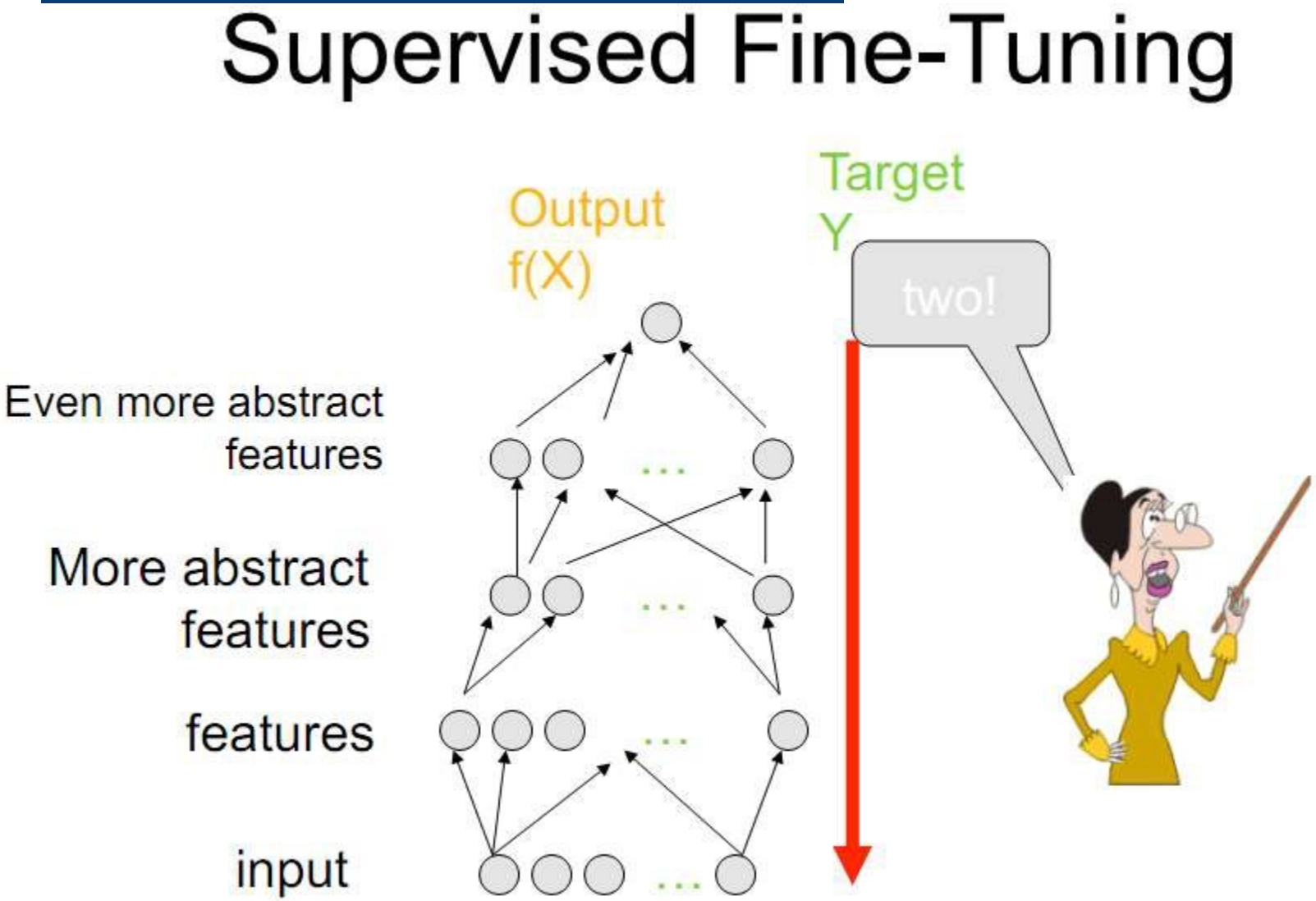


无监督式分层训练

Layer-Wise Unsupervised Pre-training



无监督式分层训练


Supervised Fine-Tuning
Output
f(X)
Target
Y
two!
Even more abstract features
More abstract features
features
input

深度神经网络模型

- Restricted Boltzmann Machine (SRBM) *(Hinton et al, NC'2006)*
- Auto-encoders *(Bengio et al, NIPS'2006)*
- Sparse auto-encoders *(Ranzato et al, NIPS'2006)*
- Kernel PCA *(Erhan 2008)*
- Denoising auto-encoders *(Vincent et al, ICML'2008)*
- Unsupervised embedding *(Weston et al, ICML'2008)*
- Slow features *(Mohabi et al, ICML'2009, Bergstra & Bengio NIPS'2009)*

实验验证

MNIST dataset

- A benchmark for handwritten digit recognition
- 10 classes (corresponding to the digits from 0 to 9)
- The inputs were scaled between 0 and 1

Exploring Strategies for Training Deep Neural Networks. Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, Pascal Lamblin; 10(Jan):1--40, 2009.

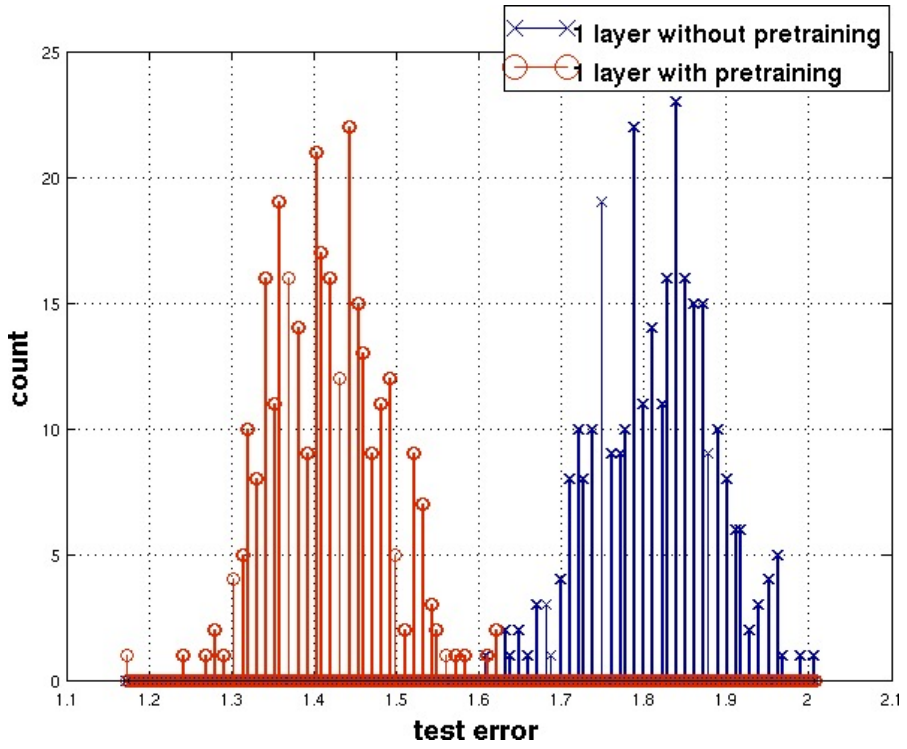
MNIST分类效果

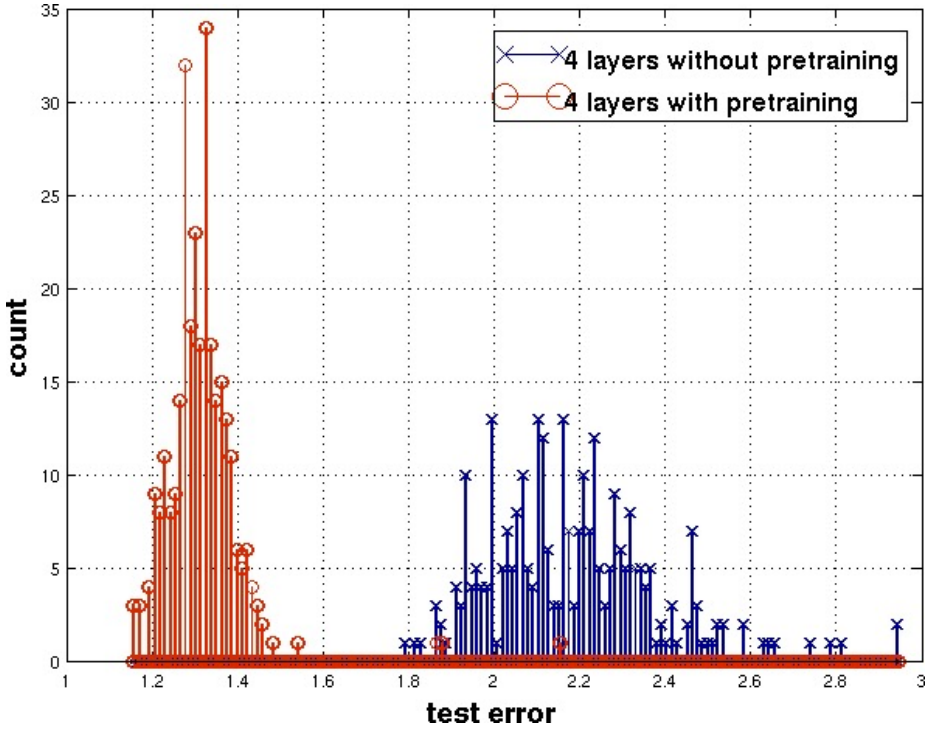
Models	Train.	Valid.	Test
SRBM (stacked restricted Boltzmann machines) network	0%	1.20%	1.20%
SAA (stacked autoassociators) network	0%	1.31%	1.41%
Stacked logistic autoregressions network	0%	1.65%	1.85%
Deep network with supervised pre-training	0%	1.74%	2.04%
Deep network, no pre-training	0.004%	2.07%	2.40%
Shallow network, no pre-training	0%	1.91%	1.93%

Table 1: Classification error on MNIST training, validation, and test sets, with the best hyperparameters according to validation error.

实验验证

- **Histograms presenting the test errors obtained on MNIST using models trained with or without pre-training.**



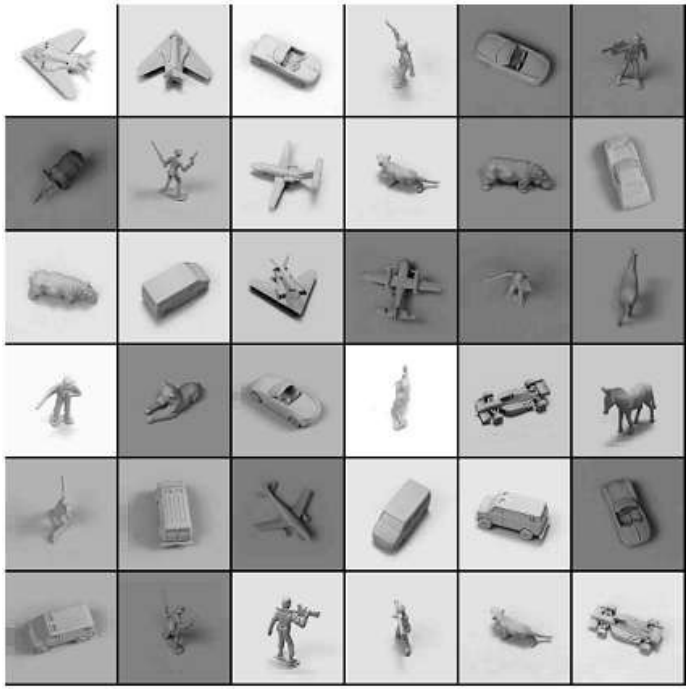


The difficulty of training deep architectures and the effect of unsupervised pre-training. Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent; pages 153-160, 2009.

实验验证

NORB dataset

- 5 object categories, 5 difference objects within each category
- Classification error rate



– DBM : 10.8 %

– SVM : 11.6 %

– Logistic Regression : 22.5 %

– KNN : 18.4 %

Efficient Learning of Deep Boltzmann Machines.
Ruslan Salakhutdinov, Hugo Larochelle ;
JMLR W&CP 9:693-700, 2010.

深度神经网络总结

- Powerful arguments and generalization principles
- Unsupervised Feature Learning is crucial
- Deep Learning is suited for multi-task learning, domain adaptation and semi-learning with few labels

One more thing…

- 第一个阶段就是2006年开始的逐层预训练，虽然也解决了一些问题，但并没有特别火
- 第二个阶段开始的标志就是2012年imagenet比赛中，cnn以压倒性优势取得胜利，自此开始深度学习才真正引人关注起来
- 虽然都叫深度学习，但其侧重点完全不同，通过一些手段，比如relu, dropout等小技巧，第二波深度学习算法已经完全抛弃了预训练的做法

https://www.zhihu.com/question/37280092/answer/78759772

第七章　深度神经网络

3.1 原理篇

3.2 实例篇

3.3 技巧篇

Keras

or theano

非常灵活

需要一些努力去学习

TensorFlow 或 Theano 的界面

keras

易于学习和使用

(仍然有一些灵活性)

Keras

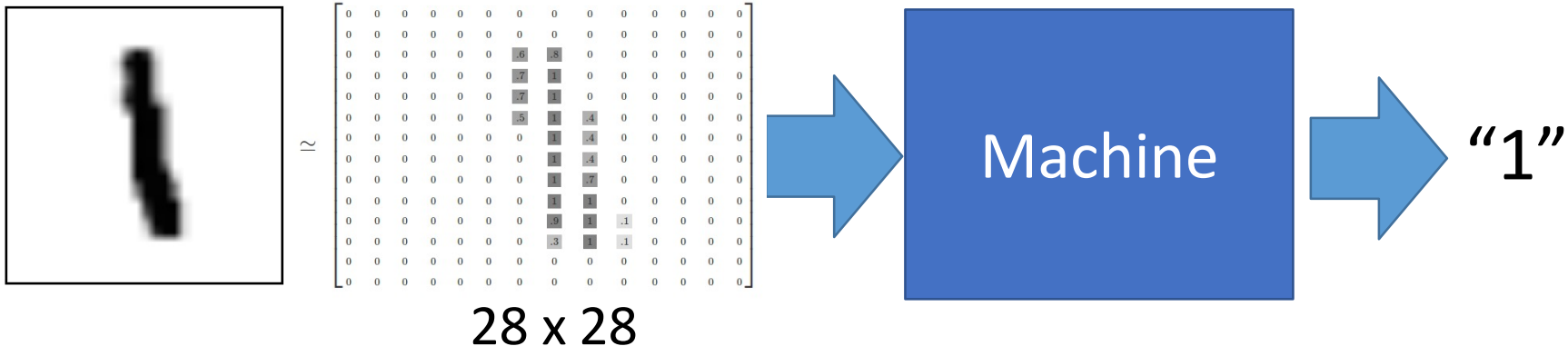
- **François Chollet 是 Keras 的创建者**
  - 他目前在谷歌担任深度学习工程师和研究员
- **Keras means *horn* in Greek**
- **文档: http://keras.io/**
- **示例:**
  - https://github.com/fchollet/keras/tree/master/examples

使用 Keras 心得


Deep Learning研究生
朋友覺得我在
我媽覺得我在
大眾覺得我在
指導教授覺得我在
我以為我在
事實上我在

示例应用程序

- 手写体数字识别



MNIST Data: http://yann.lecun.com/exdb/mnist/

“Hello world” for deep learning

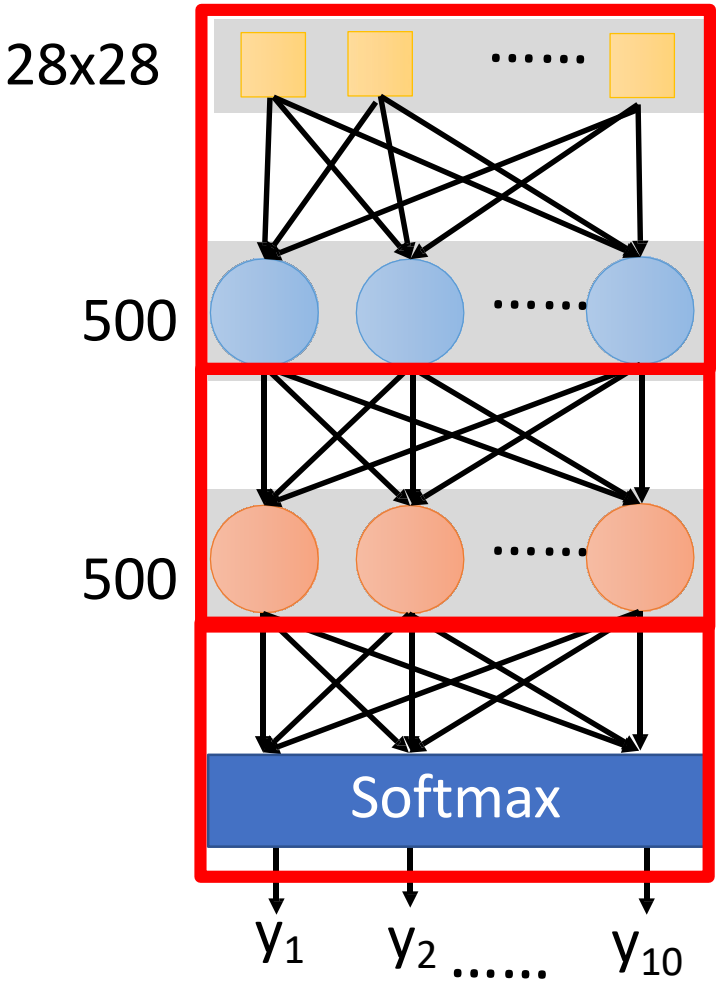
Keras provides data sets loading function: http://keras.io/datasets/

Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

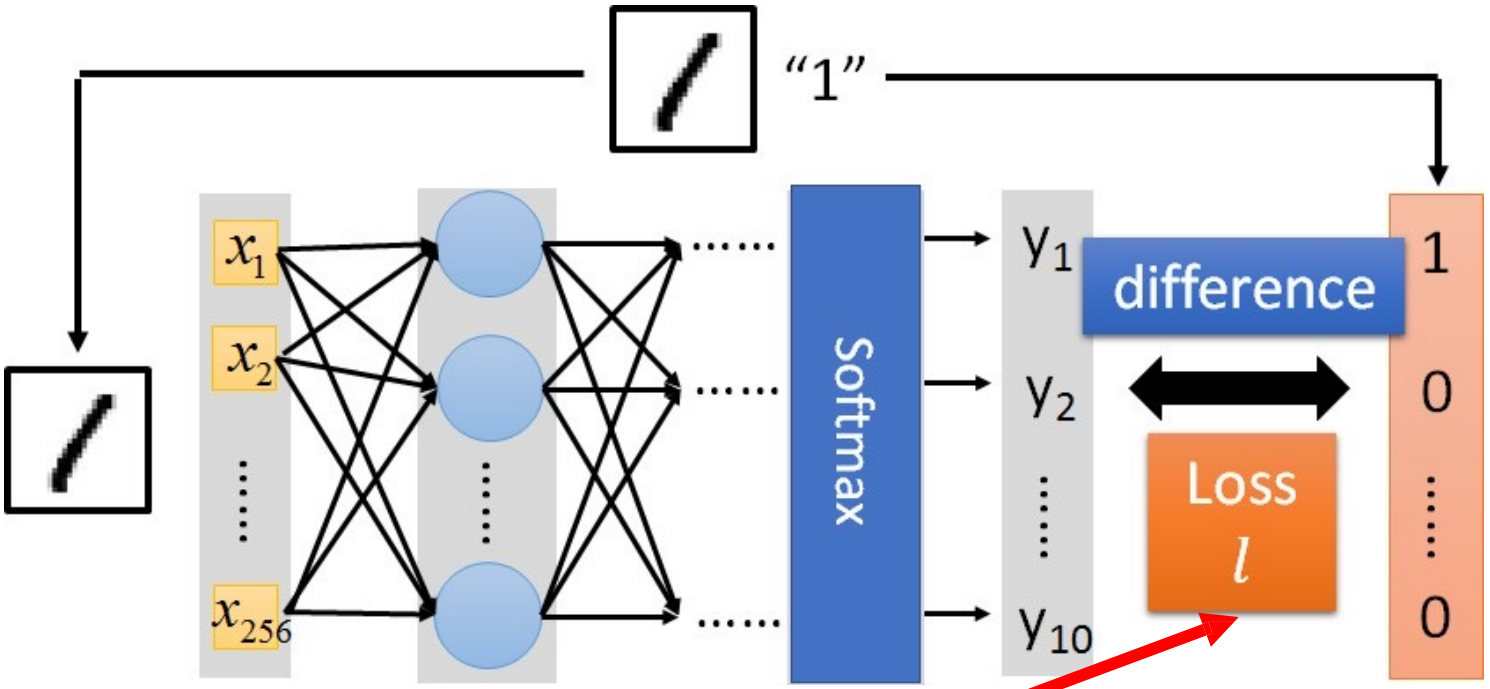
```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Keras

Step 3.1: 配置

```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

$$w \leftarrow w - \eta \partial L \ \partial w$$

0.1

Step 3.2: 找到最优的参数

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Training data (Images)

Labels (digits)

Next lecture

Keras

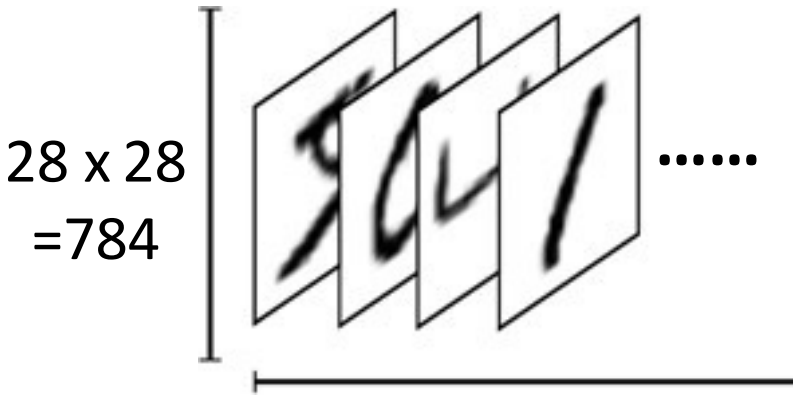
Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

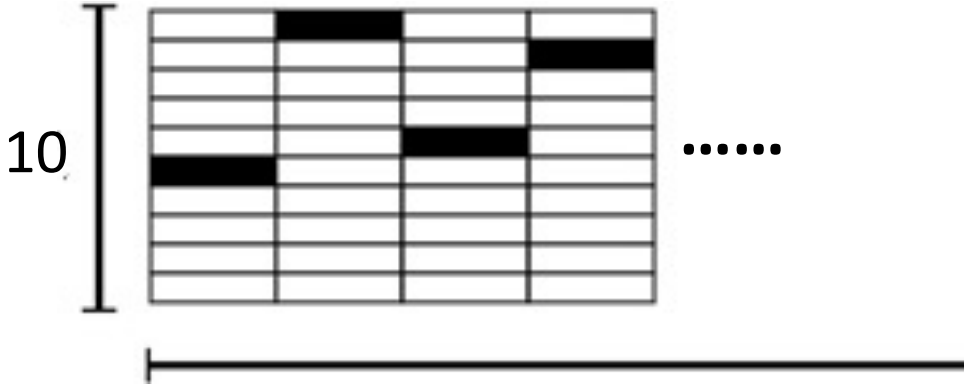
Step 3.2: 找到最优的参数

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

numpy array

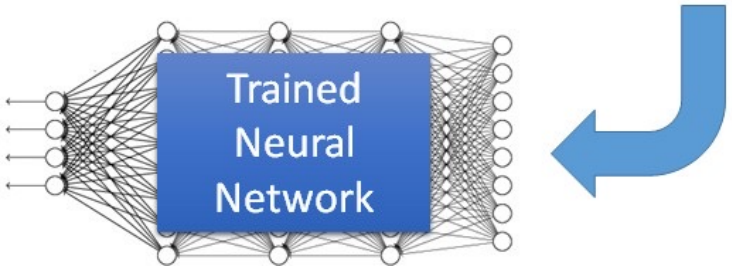
numpy array





https://www.tensorflow.org/versions/r0.8/tutorials/mnist/beginners/index.html

Keras



保存和加载模型

http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

如何使用神经网络 (测试):

case 1:

```
score = model.evaluate(x_test,y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])
```

case 2:

```
result = model.predict(x_test)
```

Keras

- 使用 GPU 加速
  - import os
  - os.environ["THEANO_FLAGS"] = "device=gpu0"

第七章　深度神经网络

3.1 原理篇

3.2 实例篇

3.3 技巧篇

深度学习的过拟合


淡奶饮品
生产日期、保质期至
生产许可证编号：
SC10544011600068
淡奶饮品
通用
龟
大乌龟(Chinemys reevesii)别称金龟、草龟、
泥龟和山龟等，在动物分类学上隶属于爬行纲、...
搜图
购物


植物
朴树
朴树，本名濮树，1973年11月8日出生于江
苏南京，中国内地男歌手。朴树成长于北...
搜图
购物
智能识物

深度学习的过拟合

YES

在训练数据效果好?

NO

过拟合!

YES

在训练数据效果好?

NO

Step 1: 定义一组函数

Step 2: 模型和数据的拟合

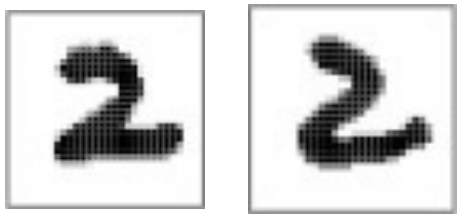
Step 3: 选择最优函数

神经网络

为什么会过拟合?

- 训练数据和测试数据是不同的

训练数据:



测试数据:

学习目标是由训练数据定义的

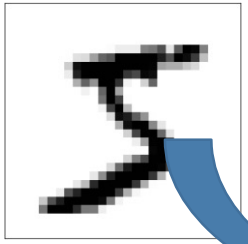
达到学习目标的参数不一定对测试数据有好的结果

过拟合的灵丹妙药

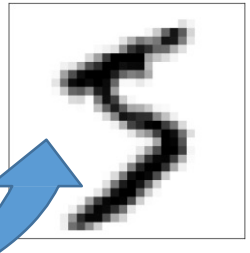
- **增加**更多的训练数据
- **创造**更多的训练数据

手写体识别:

原始的训练数据:

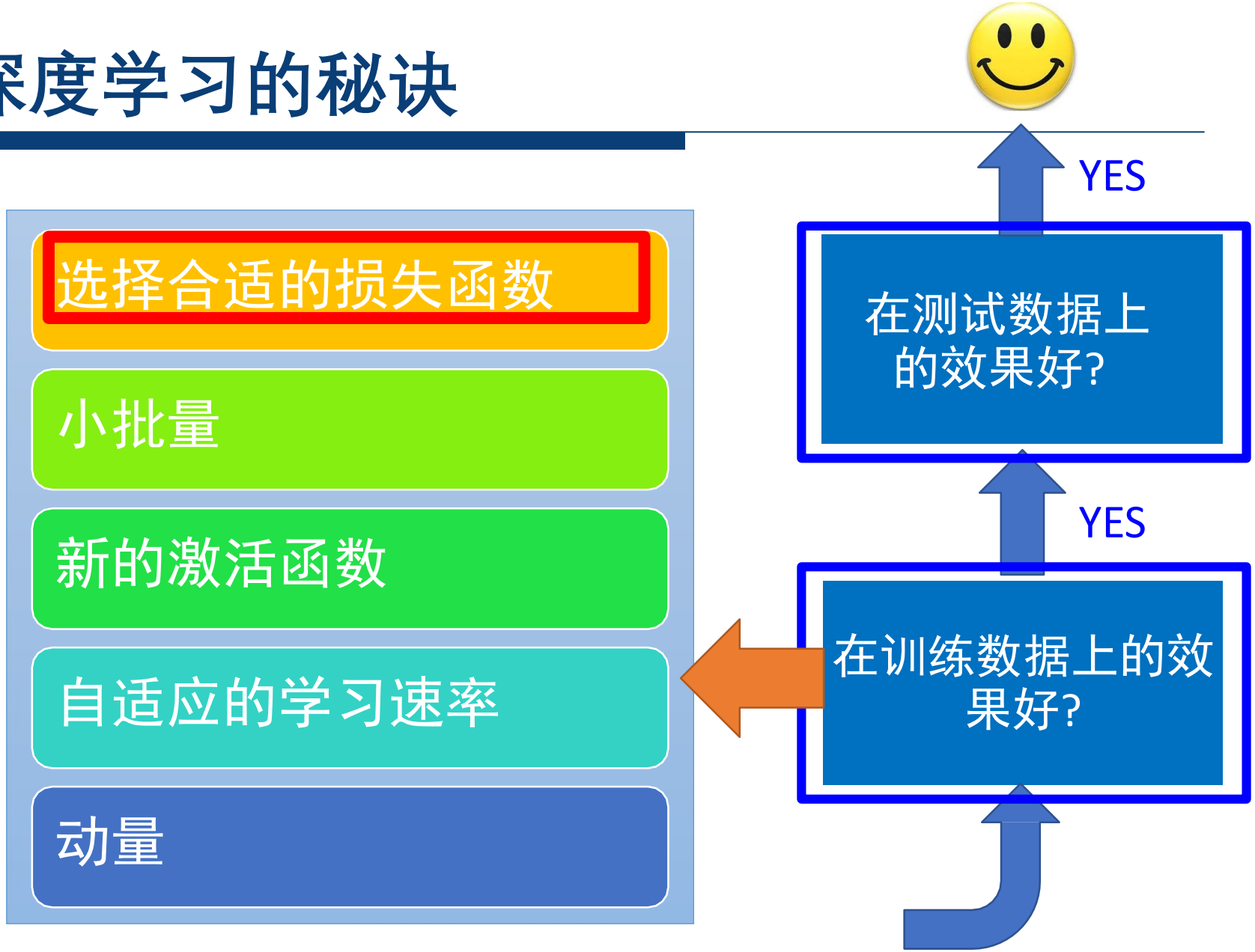


创造的训练数据:

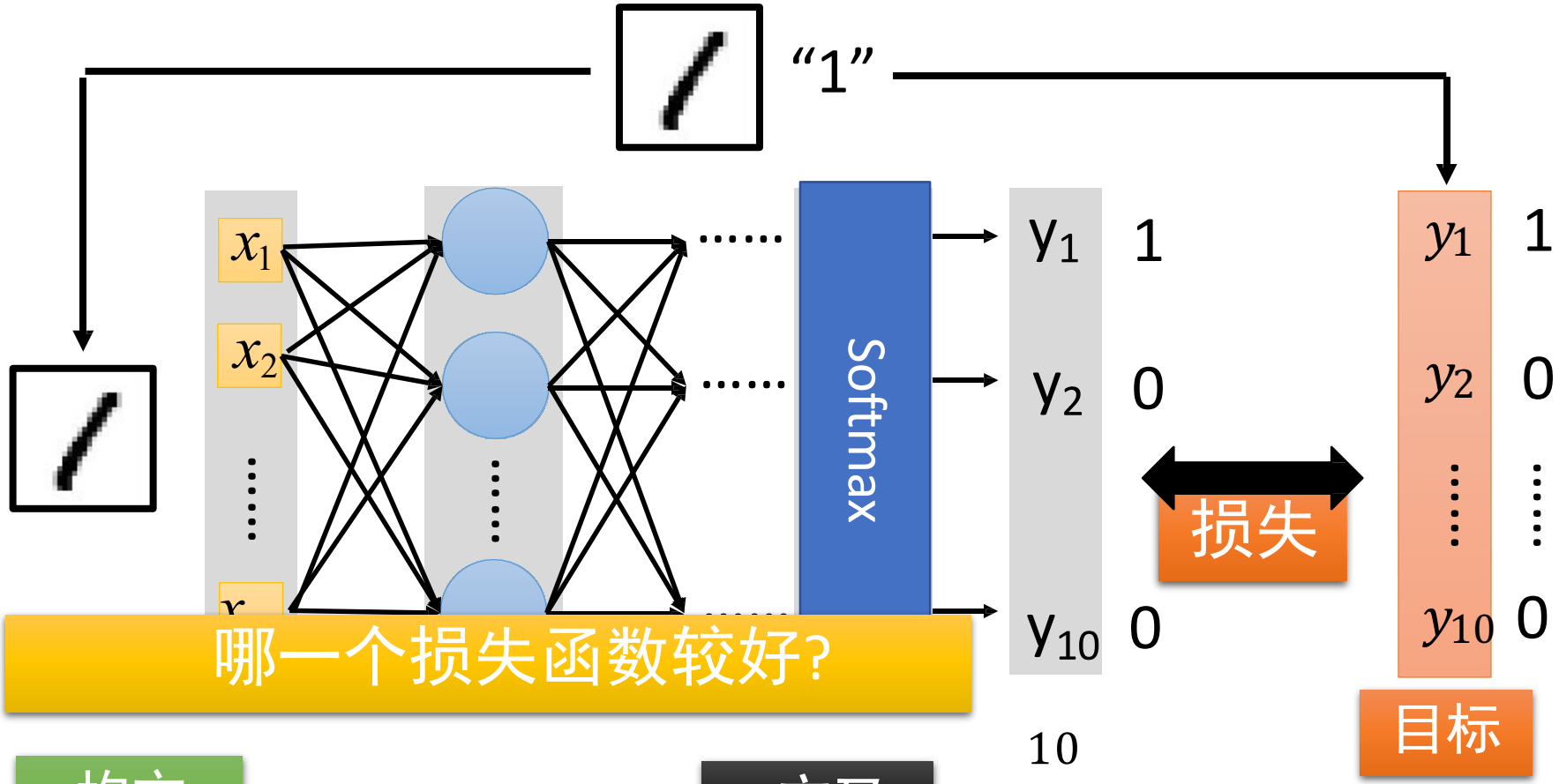


旋转 15°

深度学习的秘诀


选择合适的损失函数
小批量
新的激活函数
自适应的学习速率
动量
在训练数据上的效果好?
YES
在测试数据上的效果好?
YES

选择合适的损失函数


“1”
x_1
x_2
Softmax
y_1 1
y_2 0
y_{10} 0
损失
y_1 1
y_2 0
y_{10} 0
目标
哪一个损失函数较好?
均方误差
$\sum_0$
交叉熵
$-\sum_{i=1}^{10} y_i \ln y_i$
=0

选择合适的损失函数

平方误差

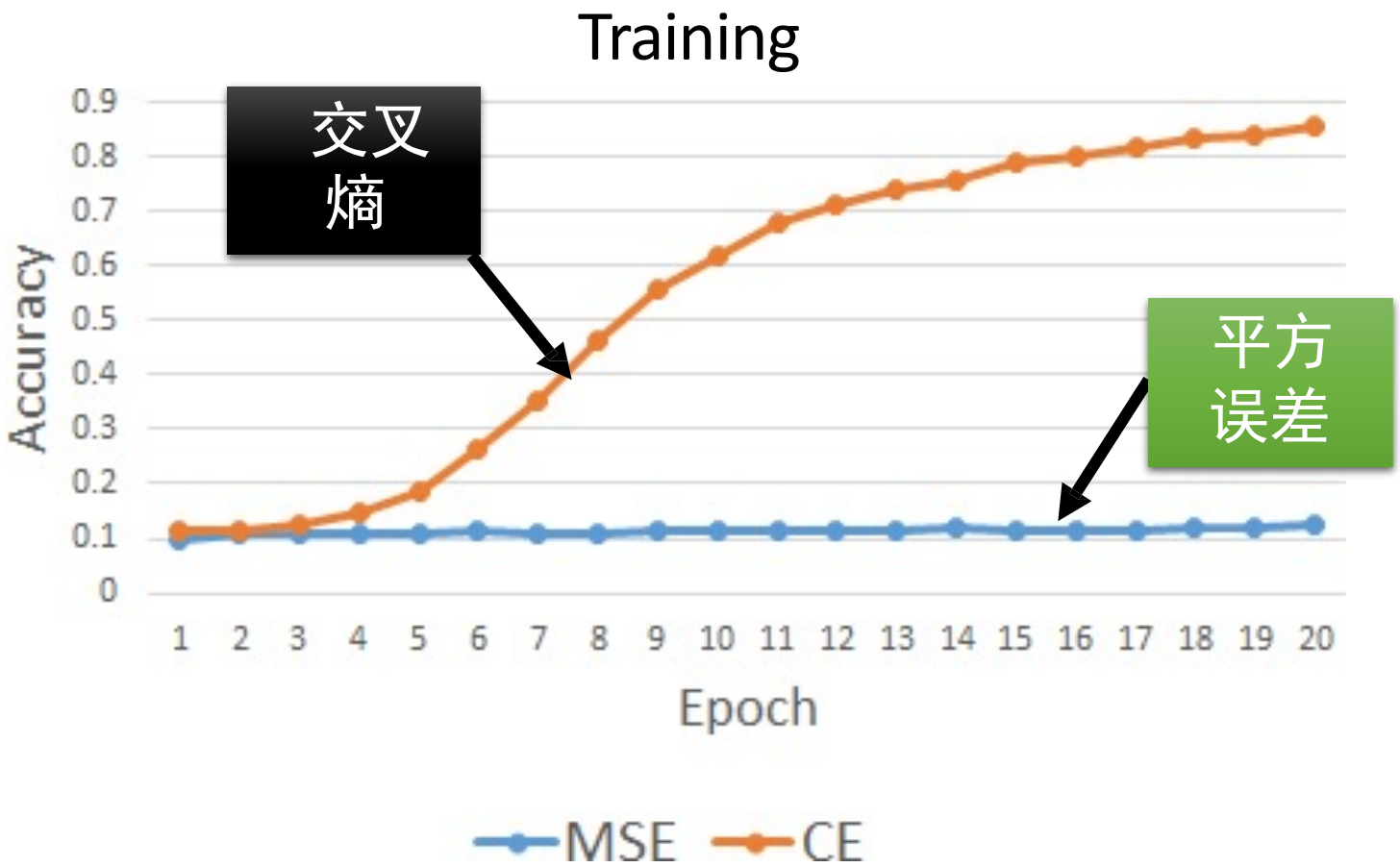
```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

交叉熵

```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

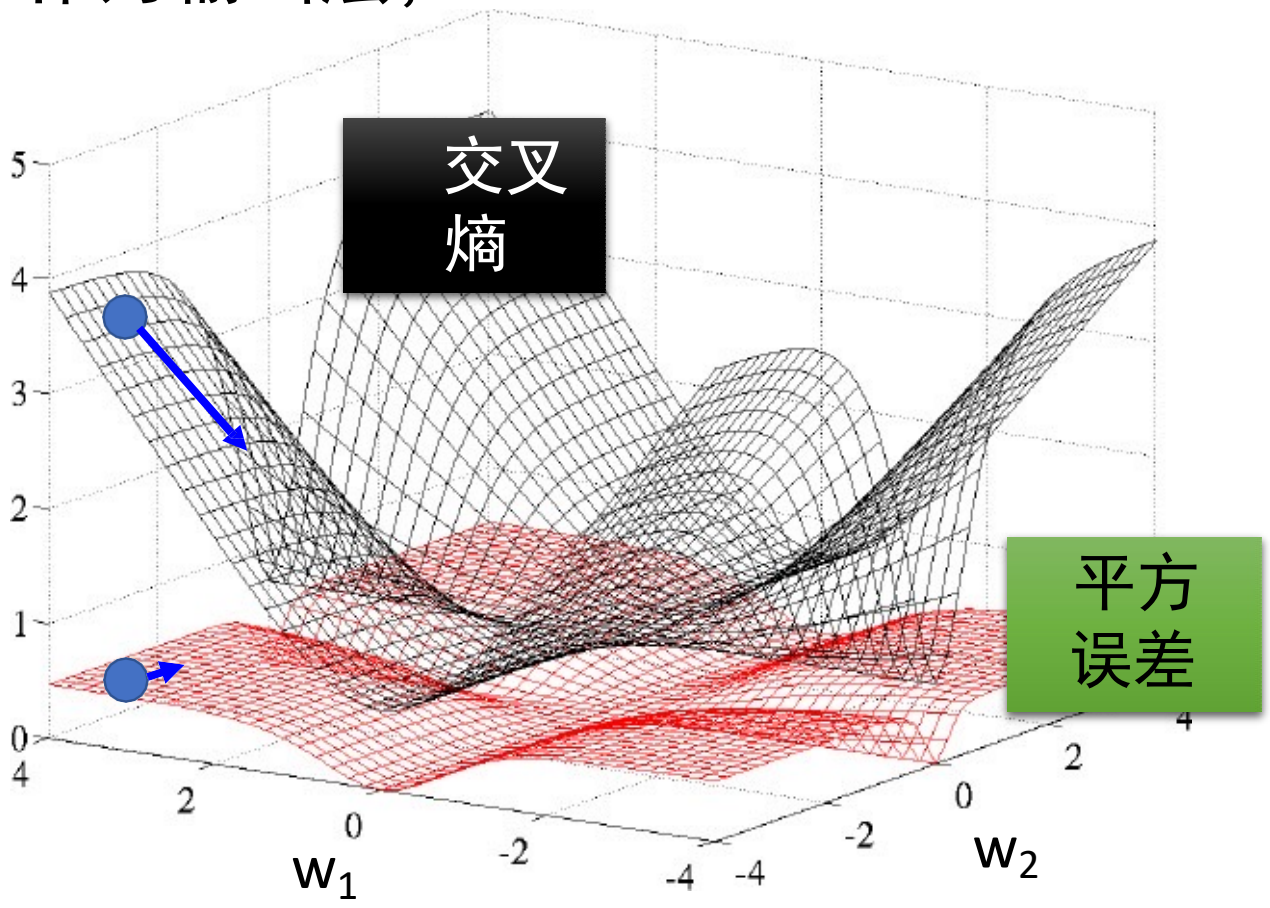
选择合适的损失函数

	准确率
平方误差	0.11
交叉熵	0.84



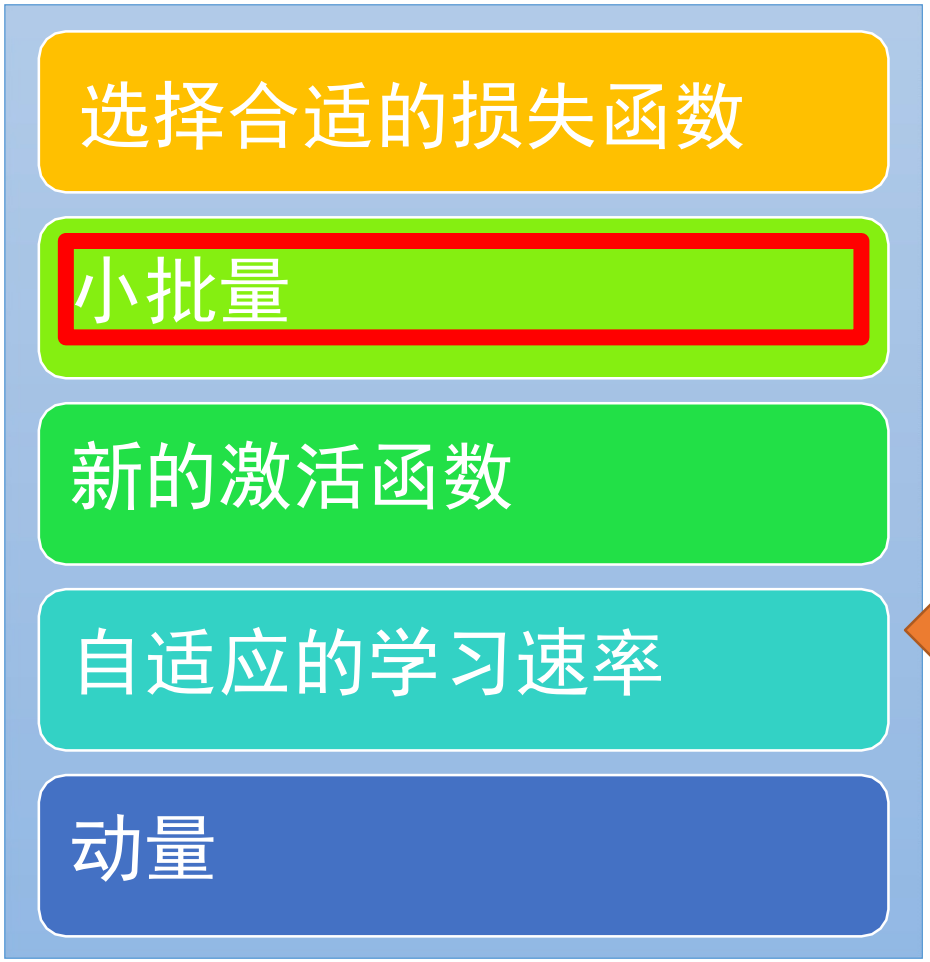
选择合适的损失函数

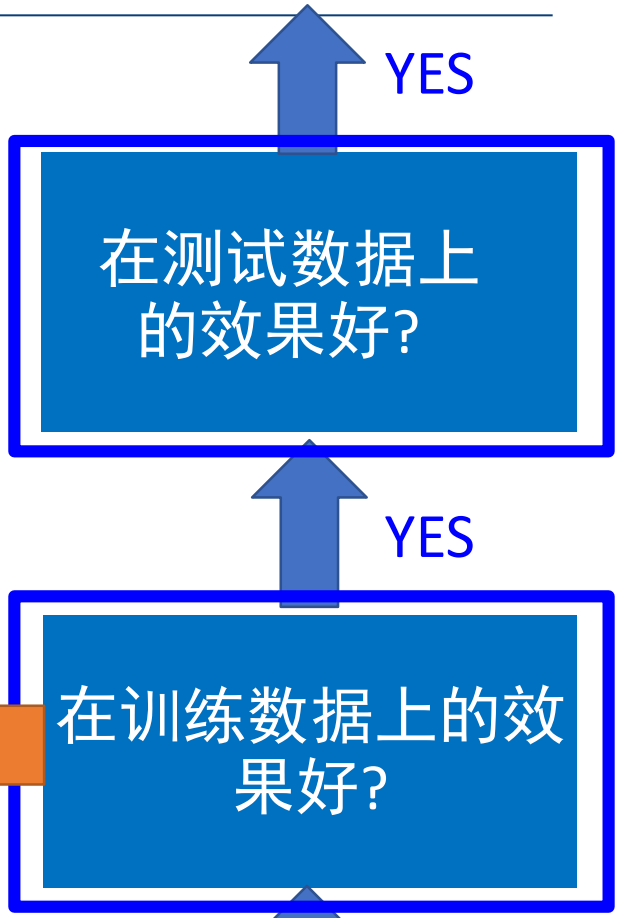
当使用softmax 作为输出层，
选择交叉熵



http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

深度学习的秘诀

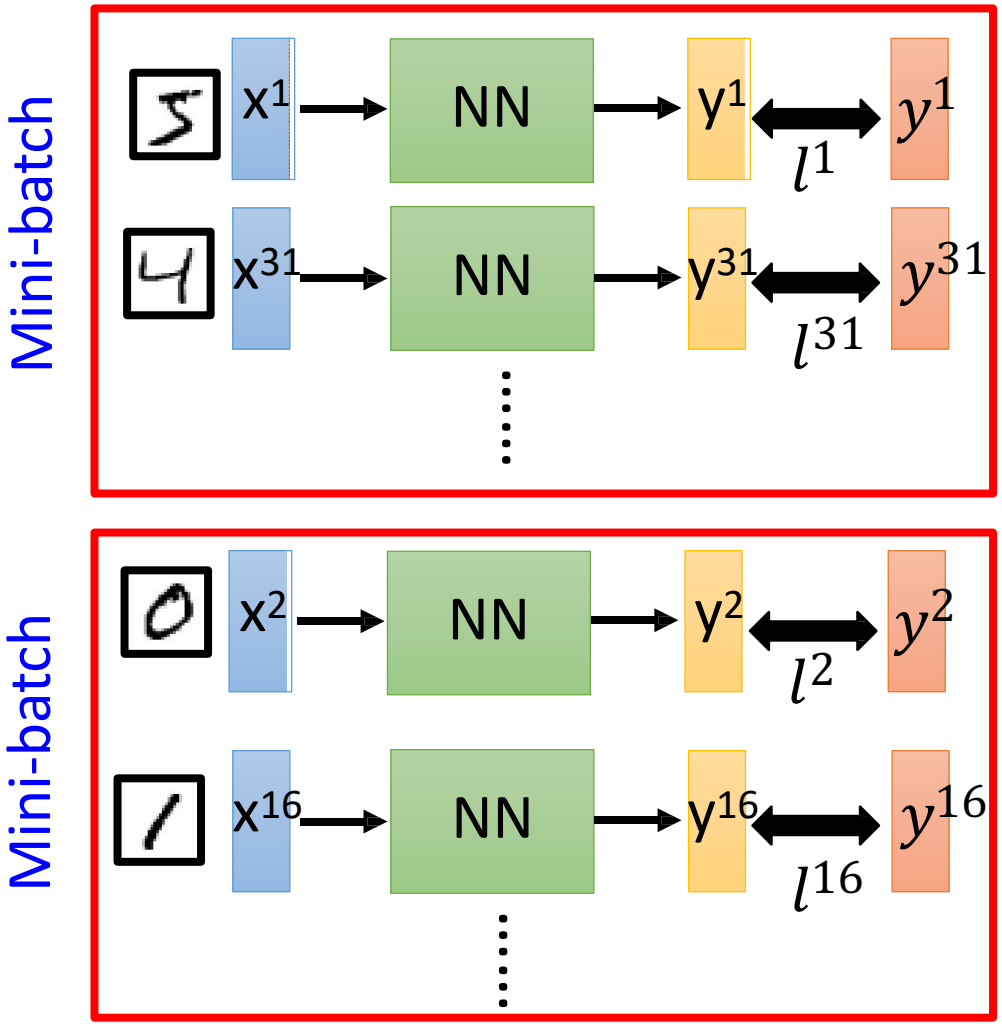





```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

小批量

随机初始化参数



- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  更新一次参数

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  更新一次参数

  ⋮

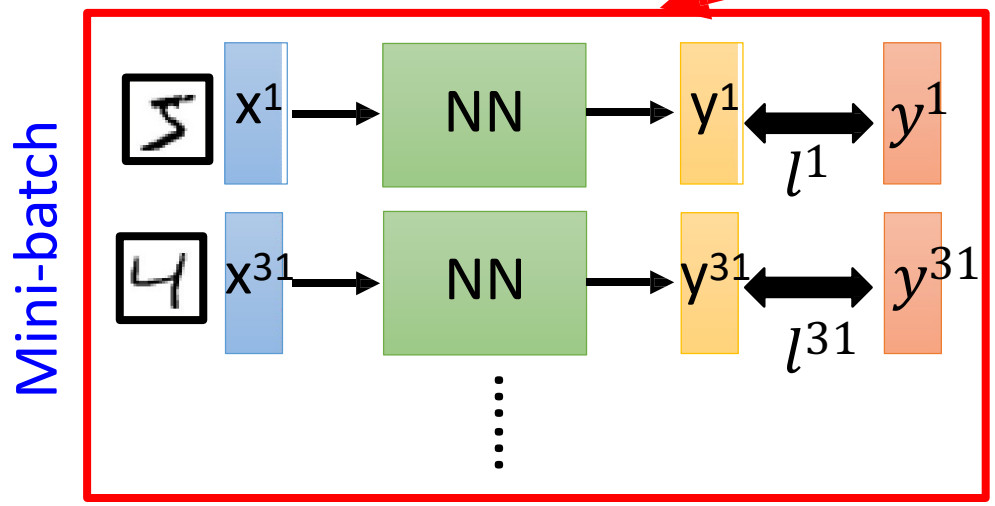
- 直到所有的小批量都被选中

one epoch

重复上述过程

小批量

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```



100个数据在 mini-batch 中

重复20次

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  更新一次参数
- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

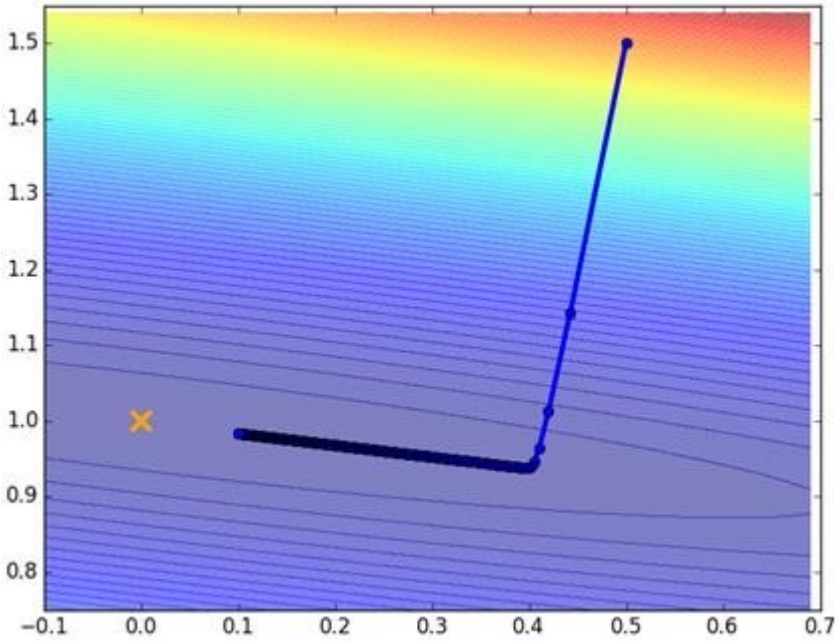
  更新一次参数

  ⋮
- 直到所有的小批量都被选中

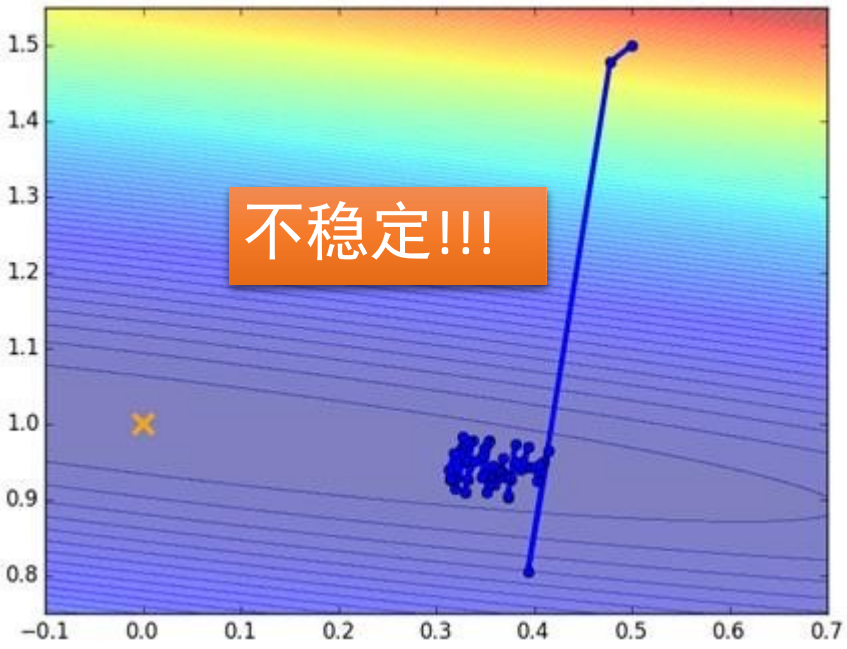
one epoch

小批量

Original Gradient Descent



With Mini-batch



颜色代表全部损失

小批量

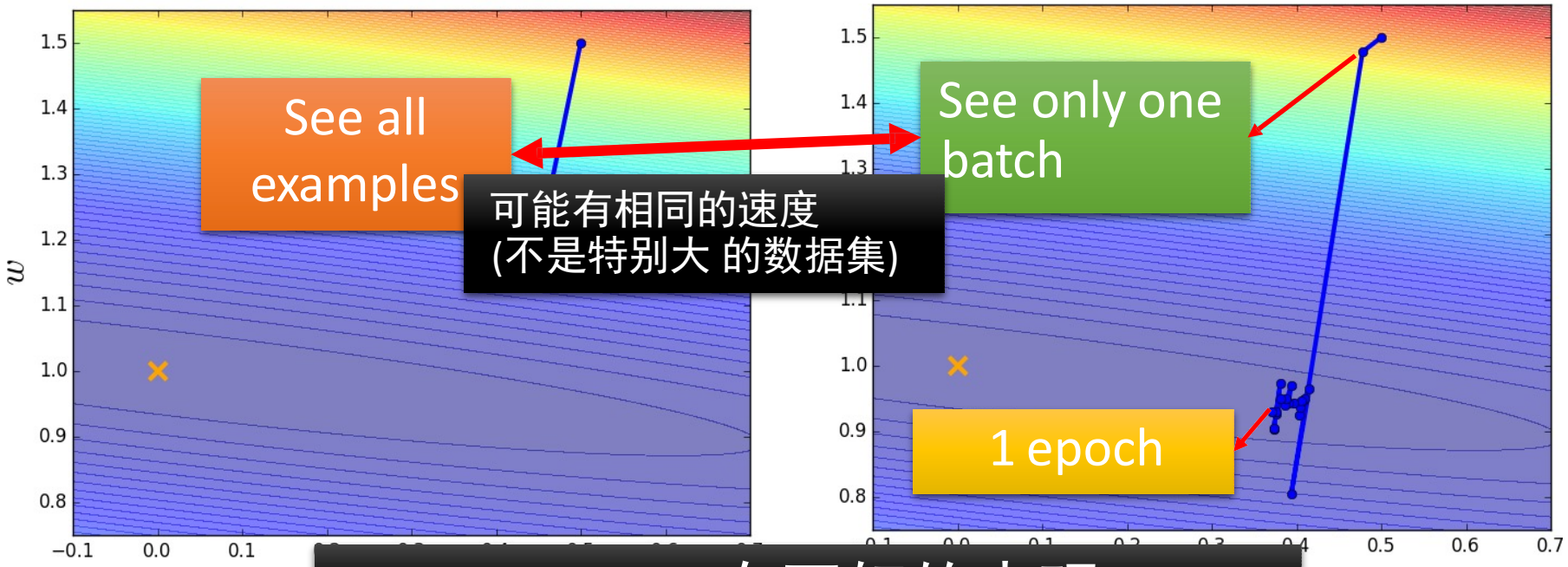
收敛速度更快

Original Gradient Descent

Update after seeing all examples

With Mini-batch

If there are 20 batches, update 20 times in one epoch.

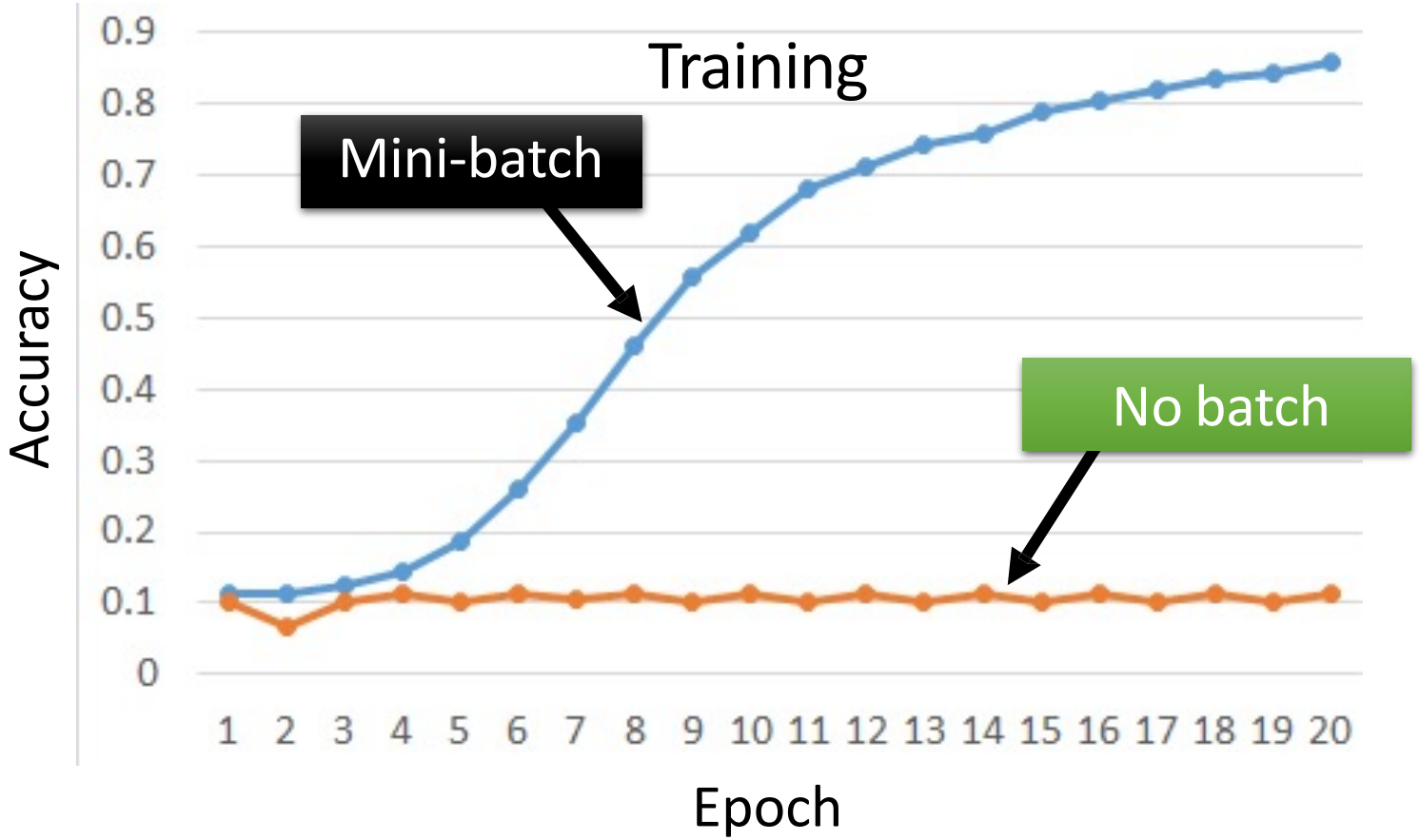


Mini-batch 有更好的表现!

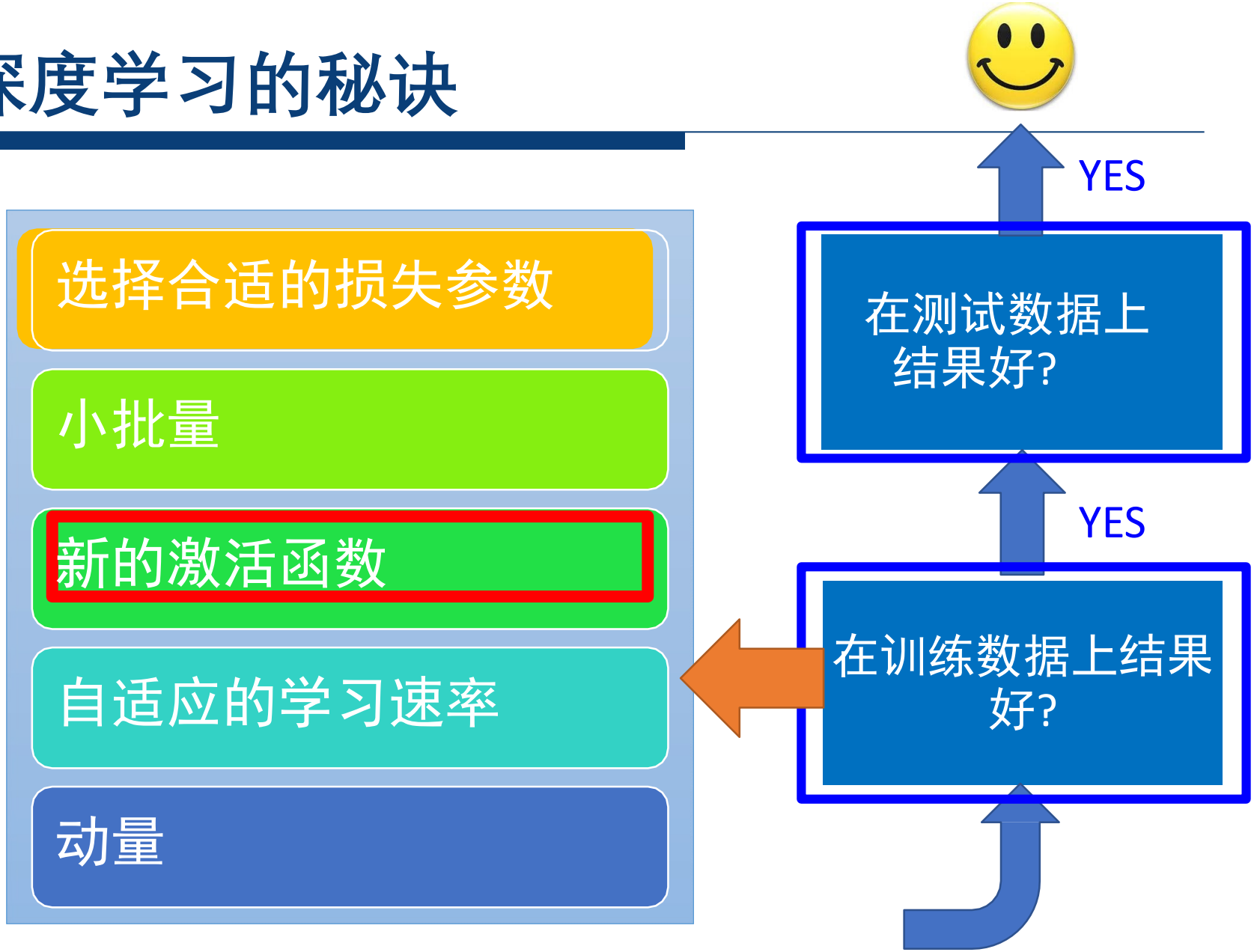
小批量

	Accuracy
Mini-batch	0.84
No batch	0.12

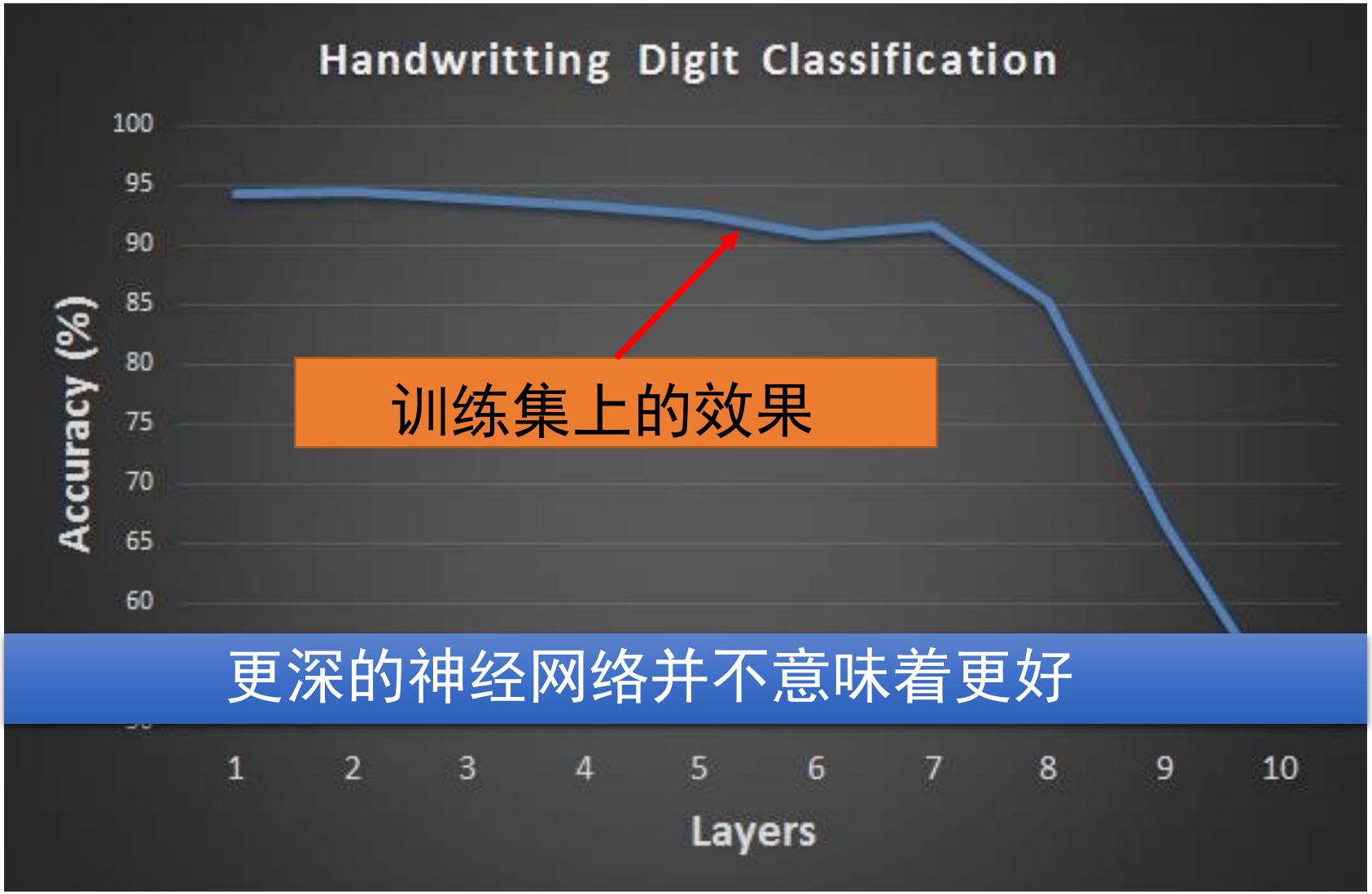
- 训练效果更好



深度学习的秘诀

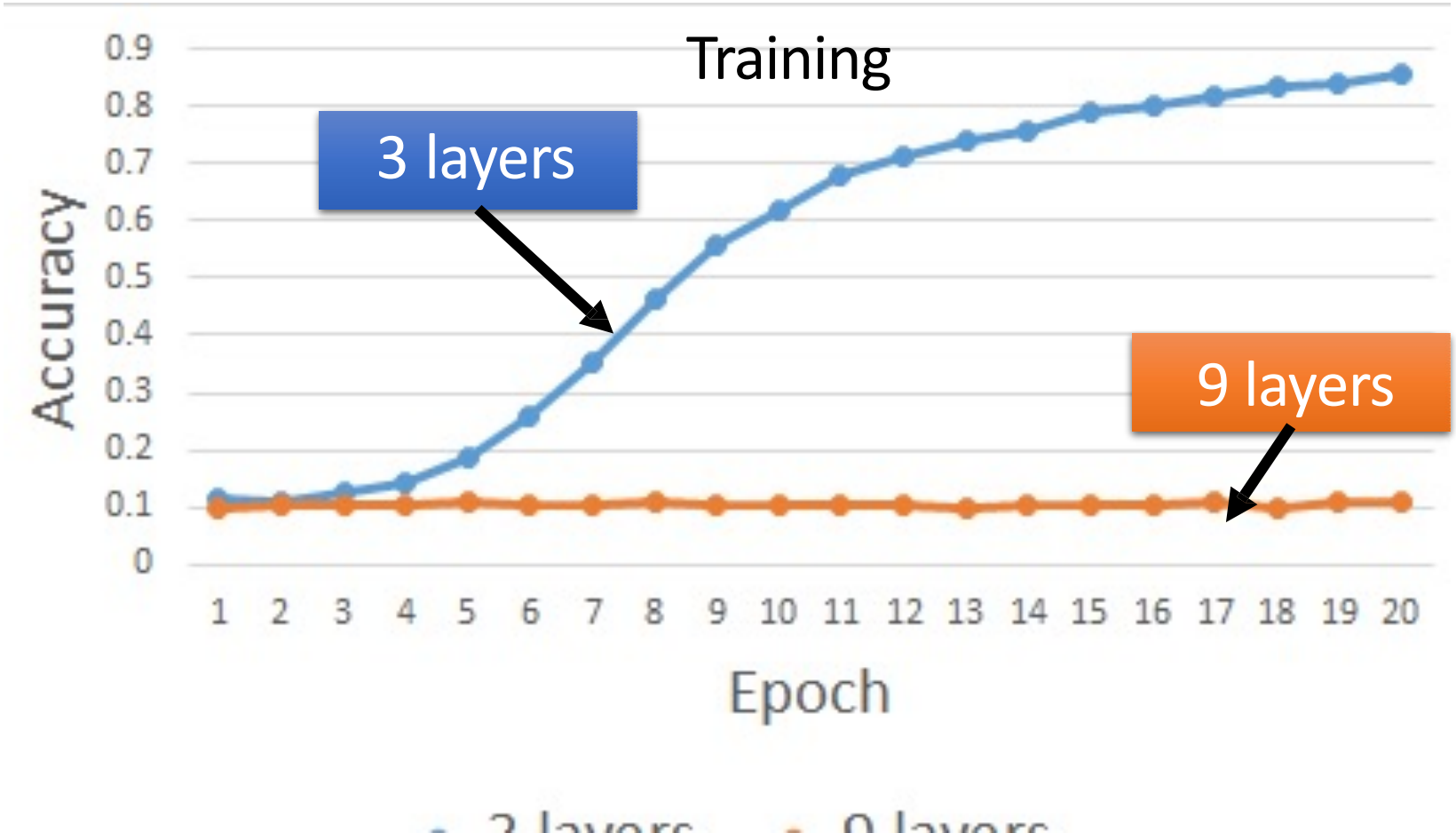

选择合适的损失参数
小批量
新的激活函数
自适应的学习速率
动量
在训练数据上结果好?
YES
在测试数据上结果好?
YES

新的激活函数

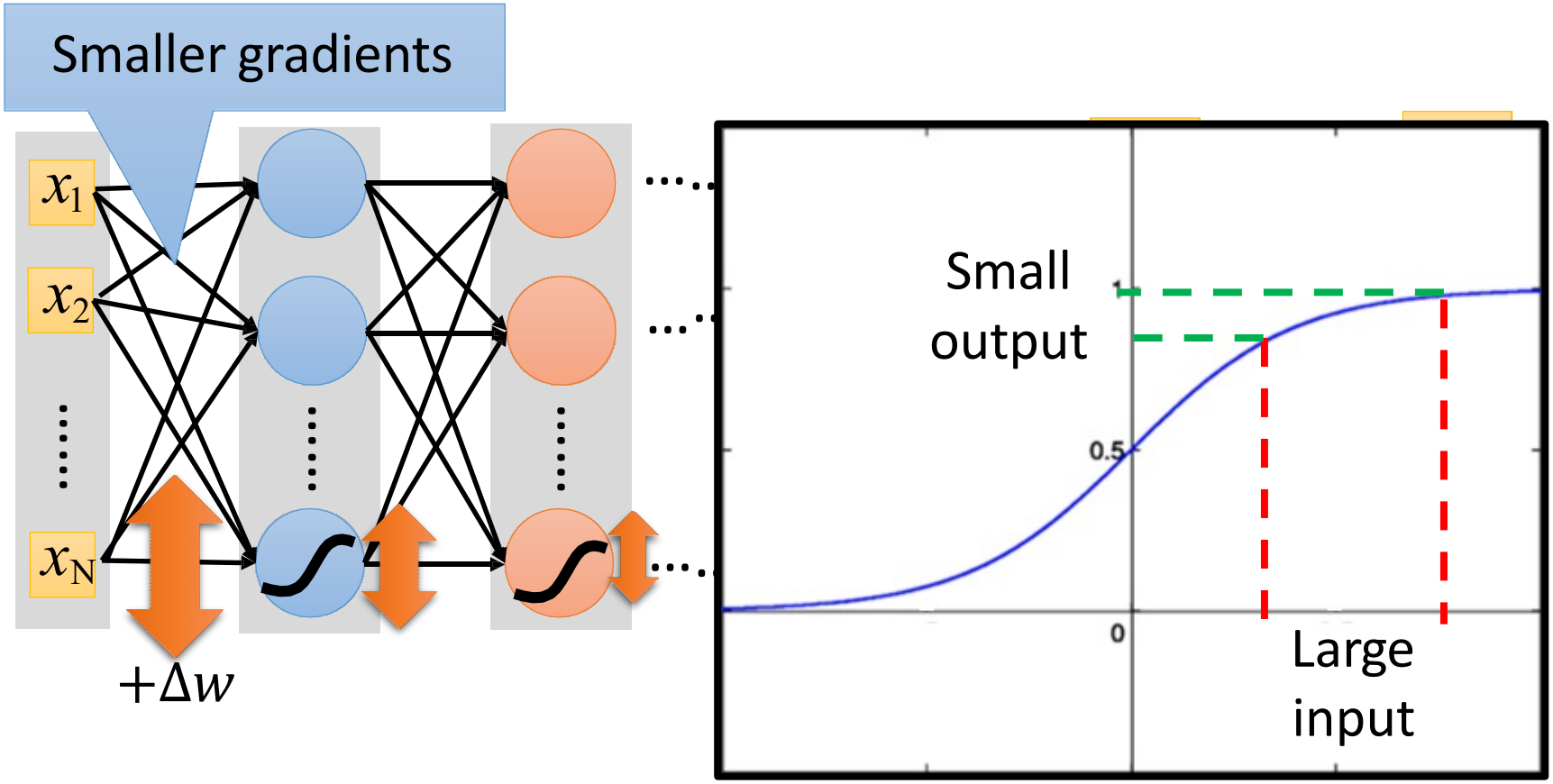

Handwritting Digit Classification
Accuracy (%)
100
95
90
85
80
75
70
65
60
训练集上的效果
更深的神经网络并不意味着更好
1
2
3
4
5
6
7
8
9
10
Layers

新的激活函数

	Accuracy
3 layers	0.84
9 layers	0.11



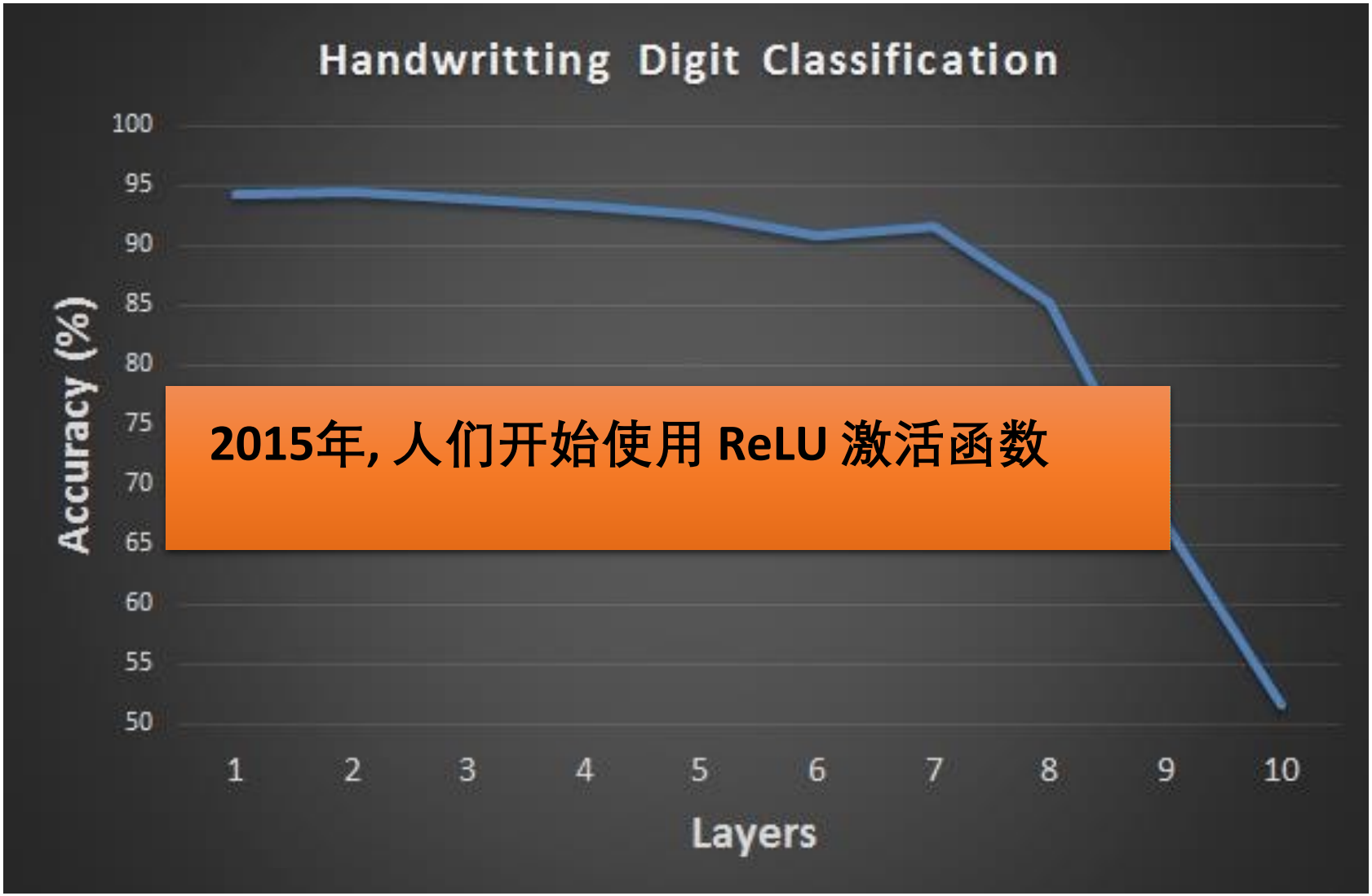
新的激活函数



直观的计算导数的方法 …

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

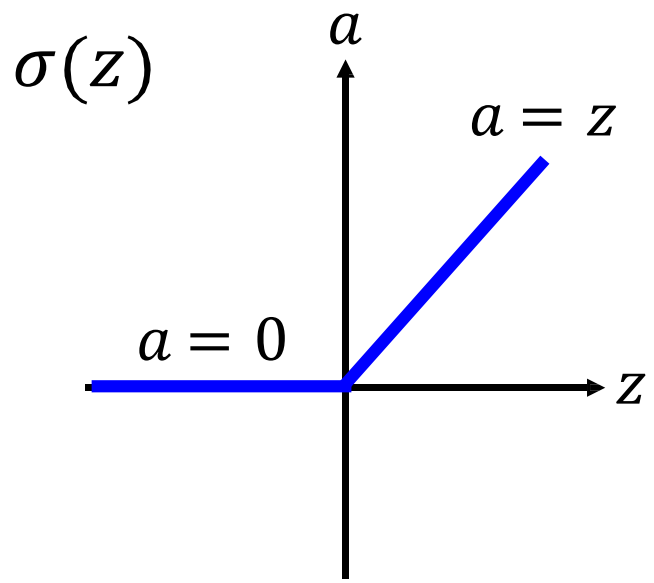
新的激活函数


Handwritting Digit Classification
Accuracy (%)
100
95
90
85
80
75
70
65
60
55
50
1
2
3
4
5
6
7
8
9
10
Layers
2015年, 人们开始使用 ReLU 激活函数

新的激活函数

ReLU

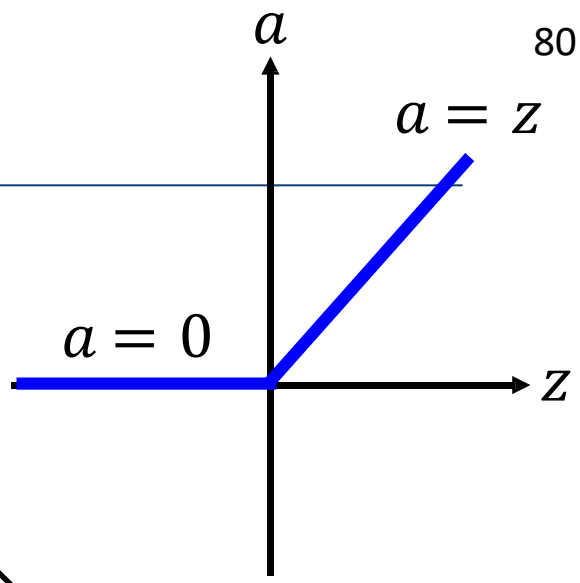
- Rectified Linear Unit (ReLU)
- Reason:

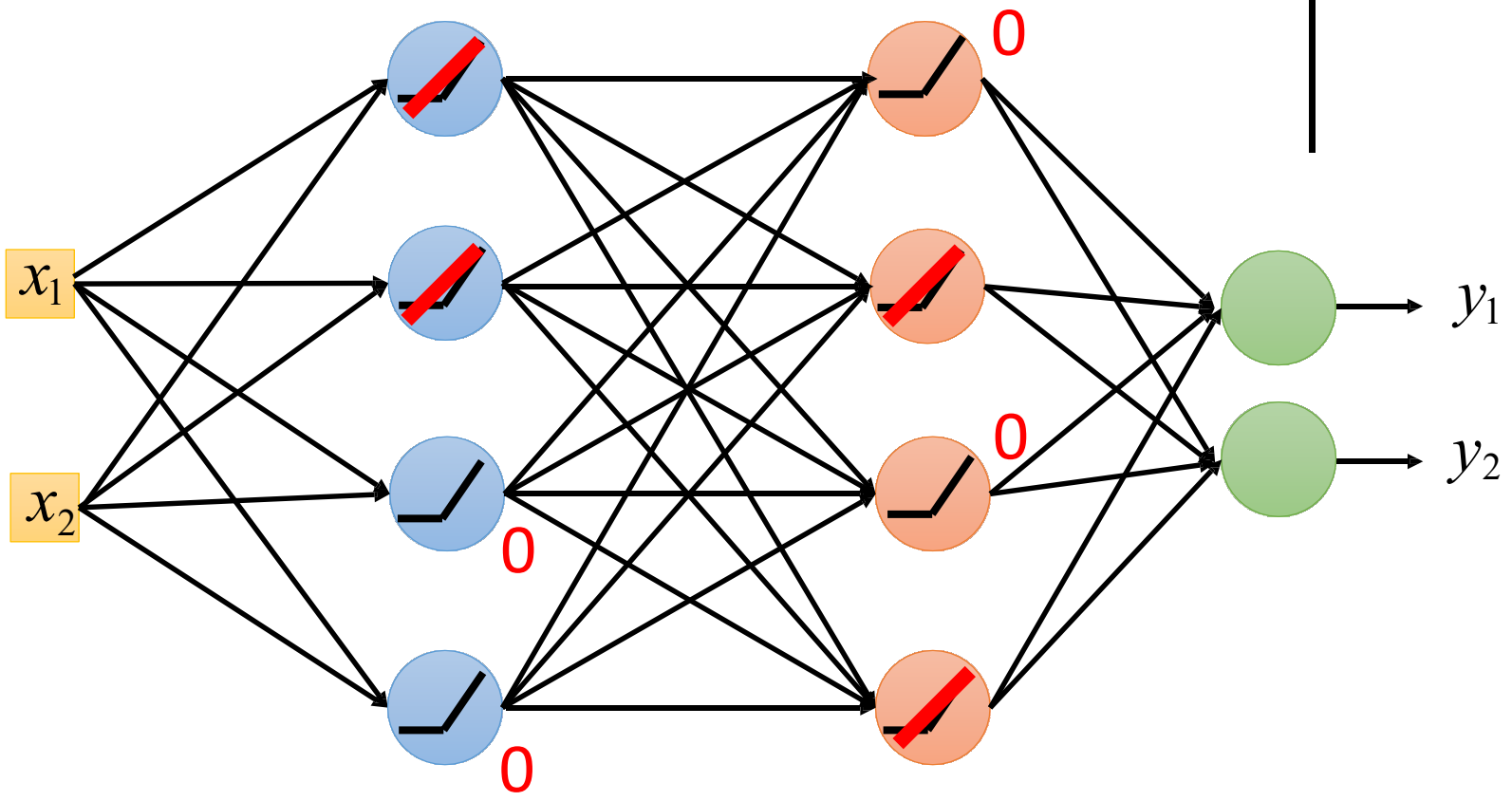


1. 计算速度快
2. 生物上的原因
3. 解决梯度下降问题

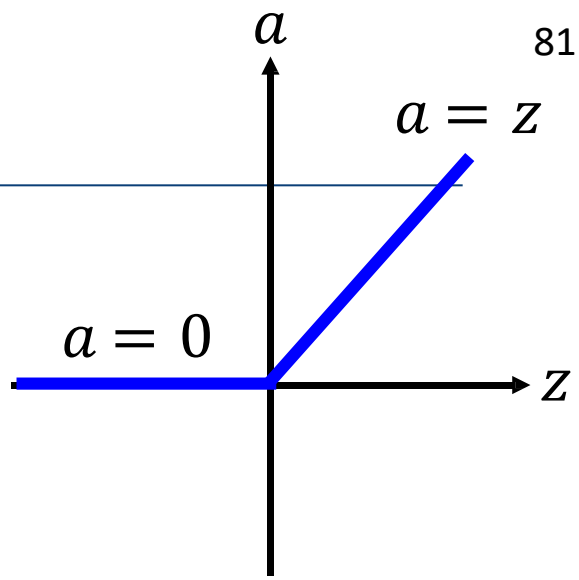
新的激活函数

- ReLU

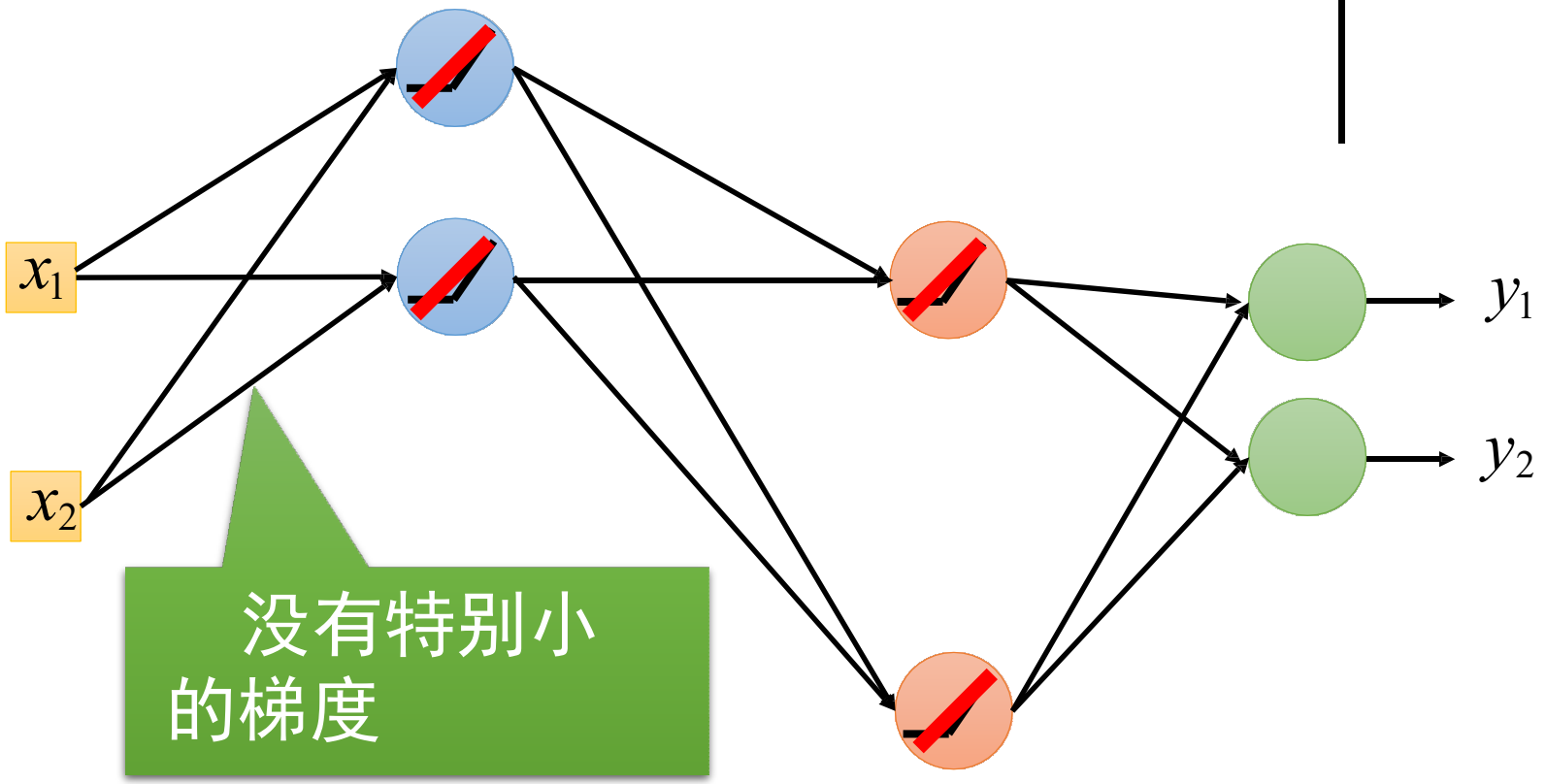




新的激活函数



一个更瘦长的线性网络

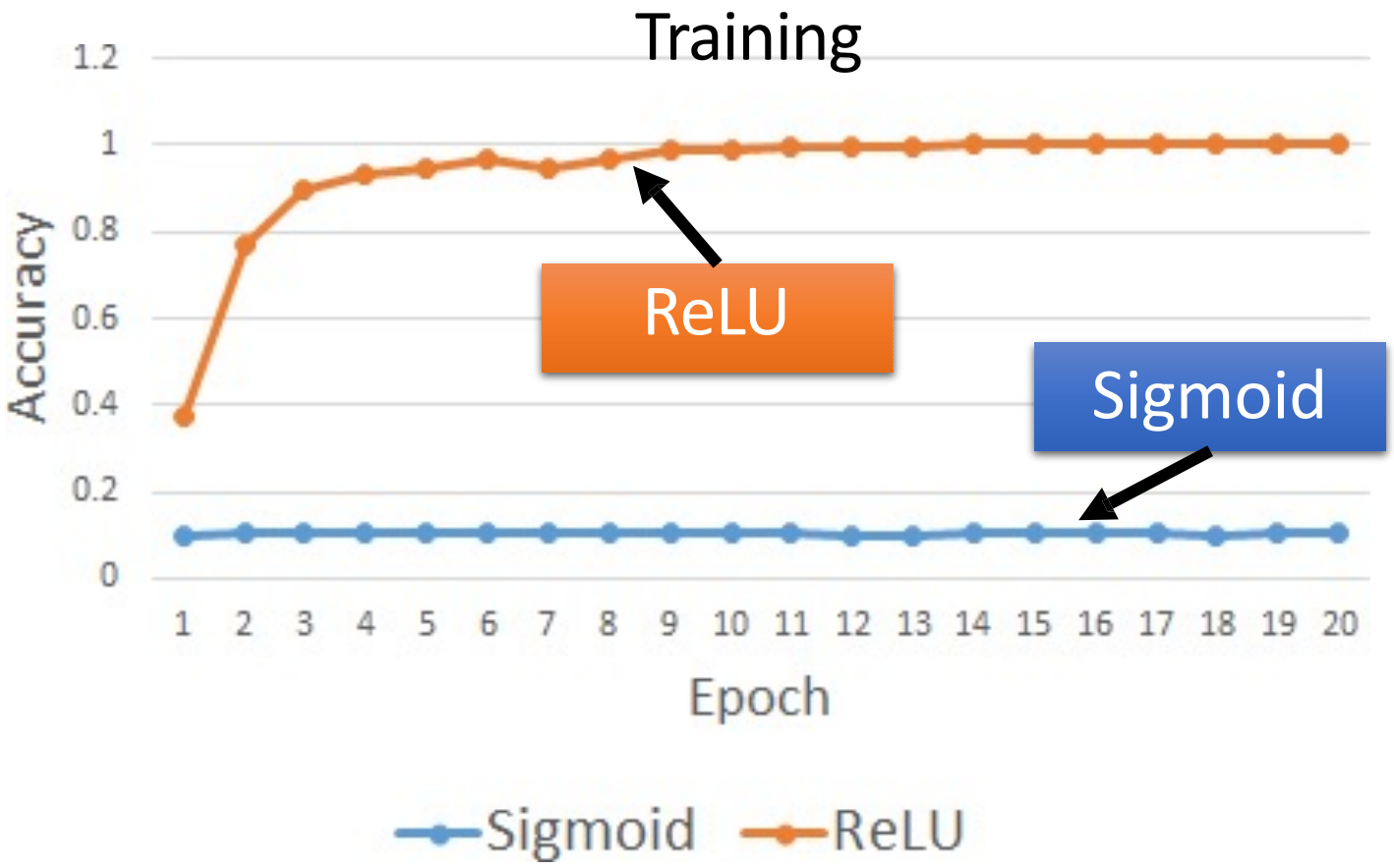


没有特别小的梯度

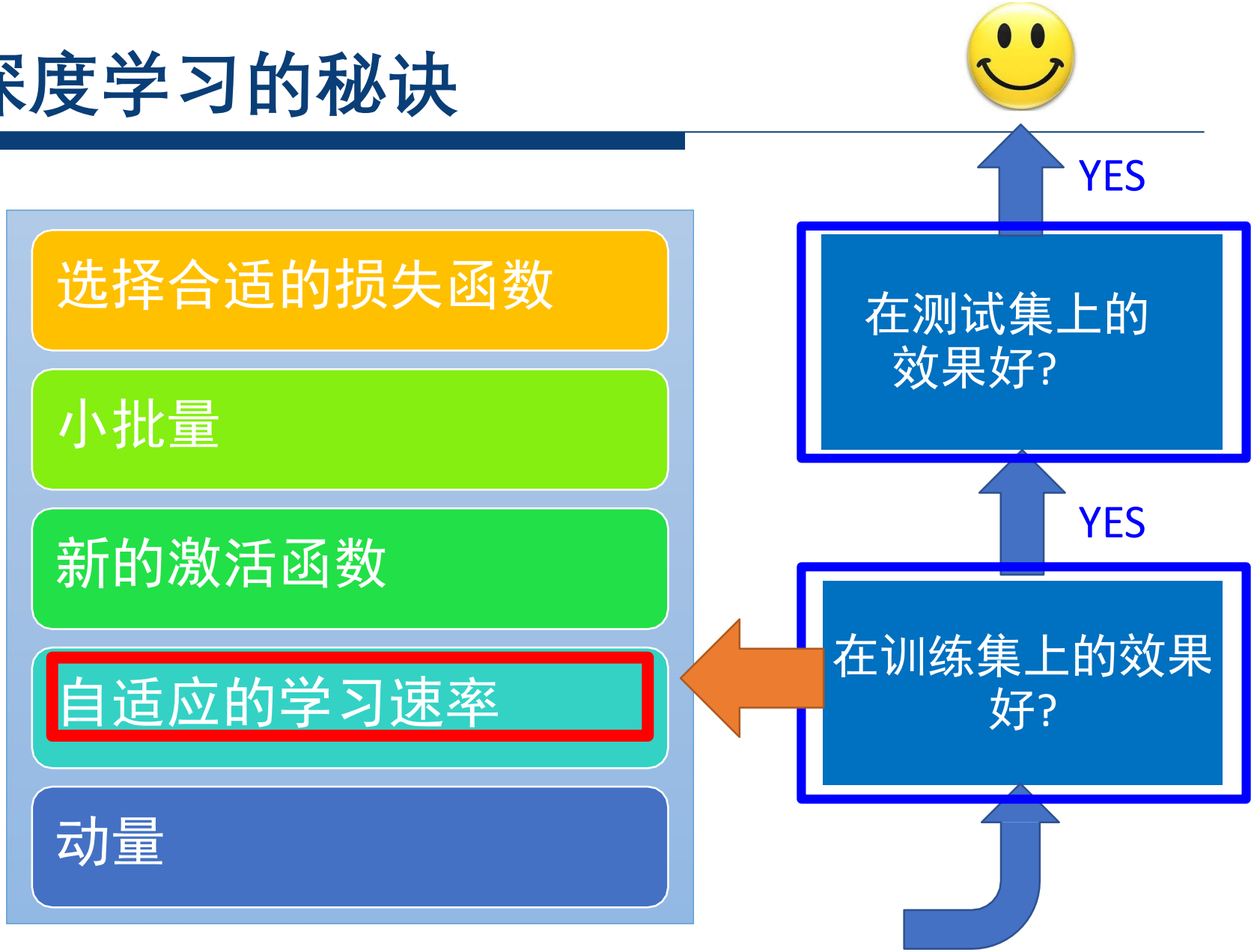
新的激活函数

9 layers	Accuracy
Sigmoid	0.11
ReLU	0.96

- 实验验证

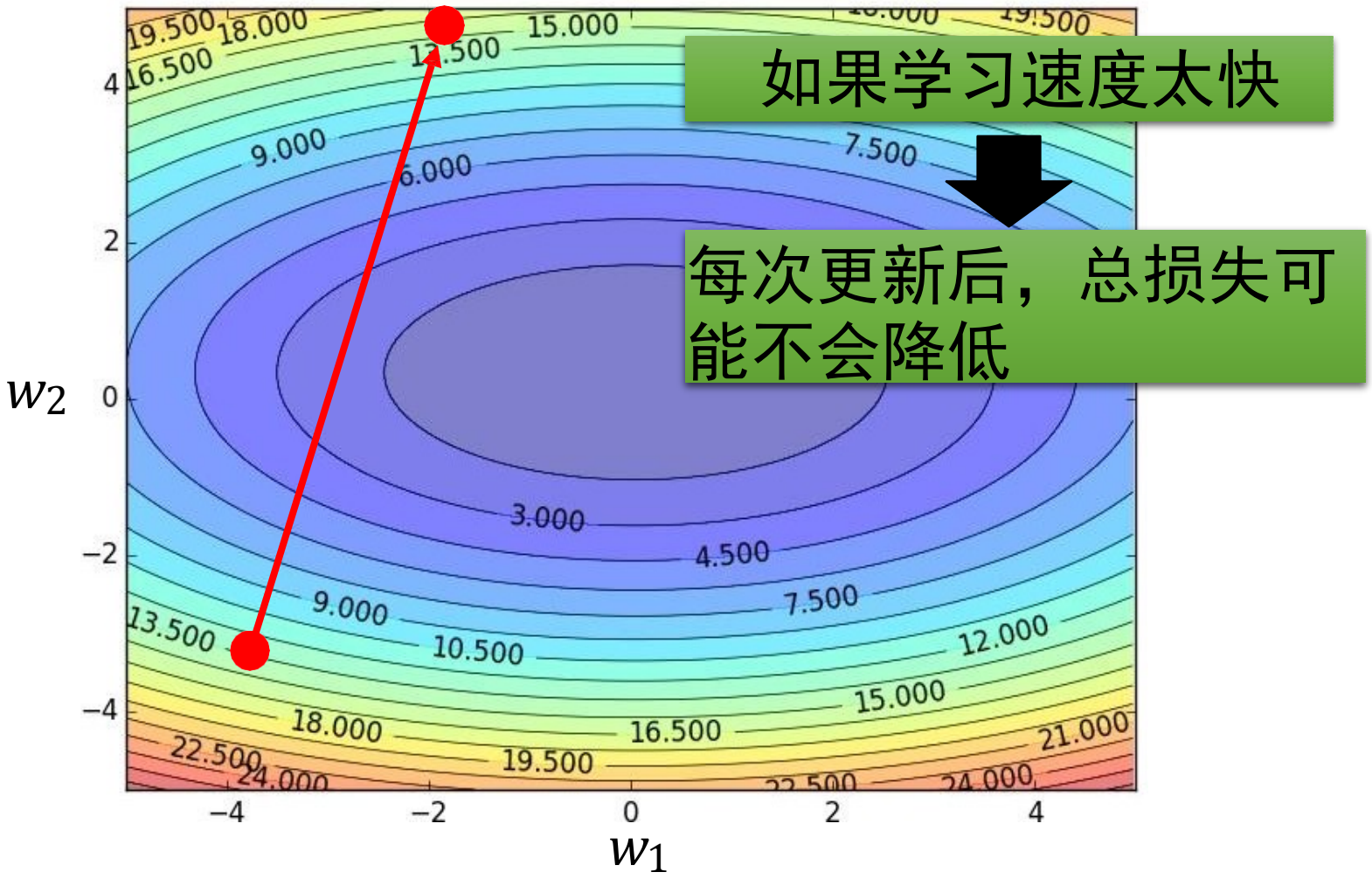


深度学习的秘诀


选择合适的损失函数
小批量
新的激活函数
自适应的学习速率
动量
在训练集上的效果好?
YES
在测试集上的效果好?
YES

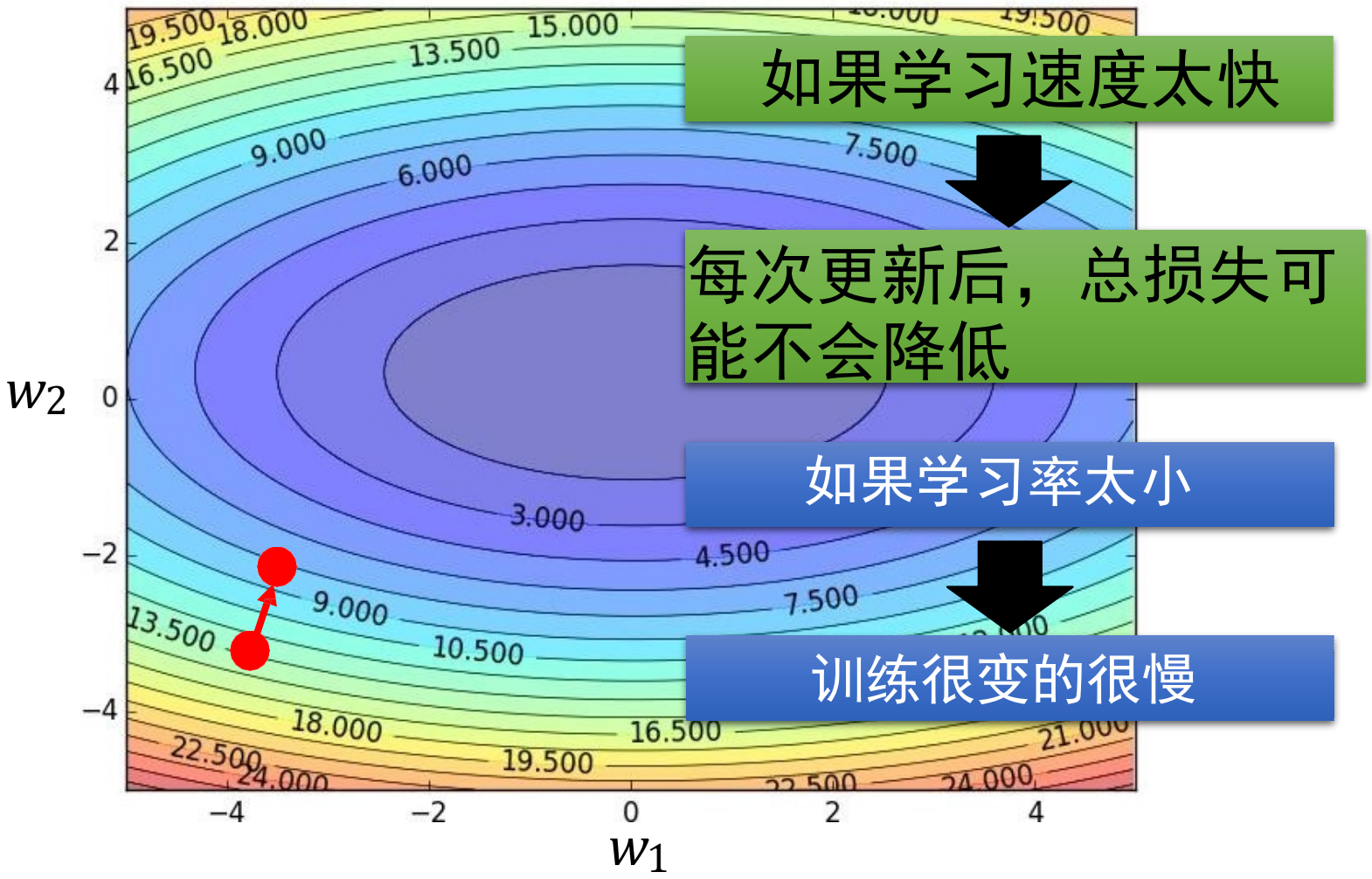
学习率

设置学习率 η


如果学习速度太快
每次更新后，总损失可能不会降低
w_2
w_1
19.500
18.000
16.500
15.000
13.500
9.000
7.500
6.000
3.000
4.500
9.000
7.500
10.500
12.000
13.500
15.000
18.000
16.500
21.000
22.500
19.500
24.000
4
2
0
−2
−4
−4
−2
0
2
4

学习率

设置学习率 η



如果学习速度太快

每次更新后，总损失可能不会降低

如果学习率太小

训练很变的很慢

自适应学习率：Adagrad

Original: $w \leftarrow w - \eta \partial L / \partial w$

Adagrad: $w \leftarrow w - \eta_w \partial L / \partial w$

参数依赖学习率

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^i)^2}}$$

常量

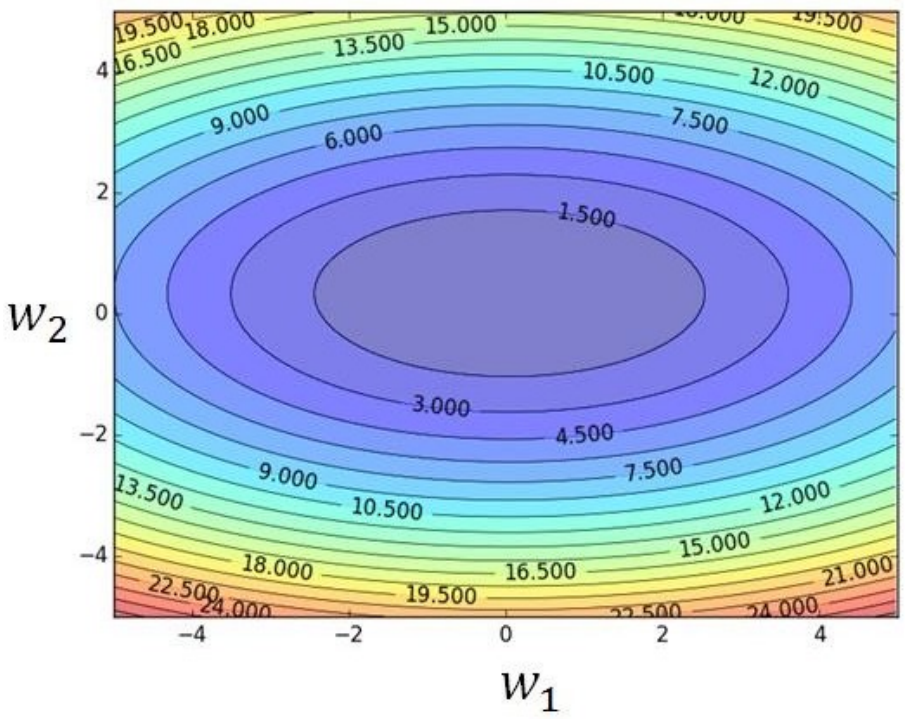
g^i is $\partial L / \partial w$ obtained at the i-th update

前一个导数的平方的总和

自适应学习率

较大的导数

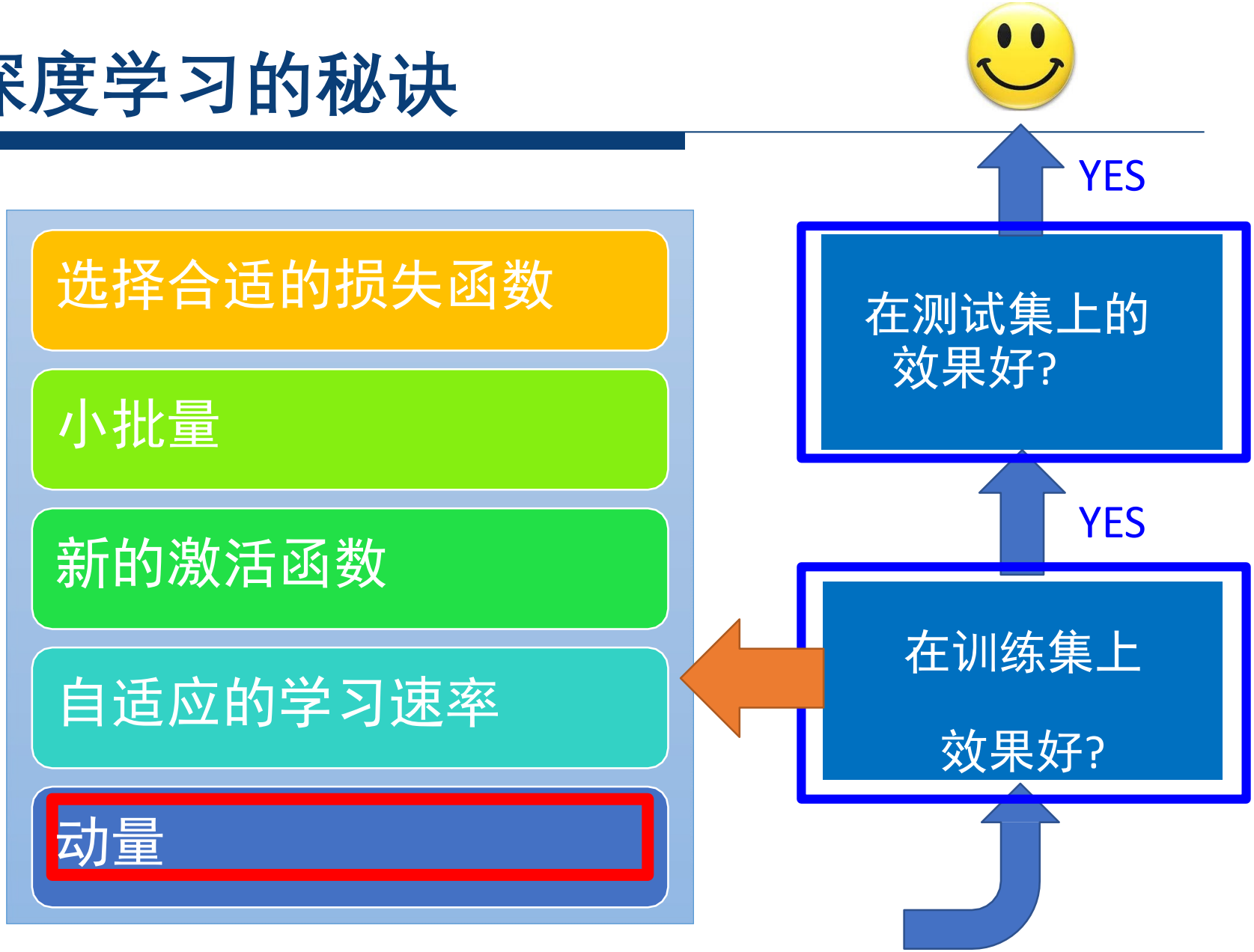
较小的学习速率



较小的导数

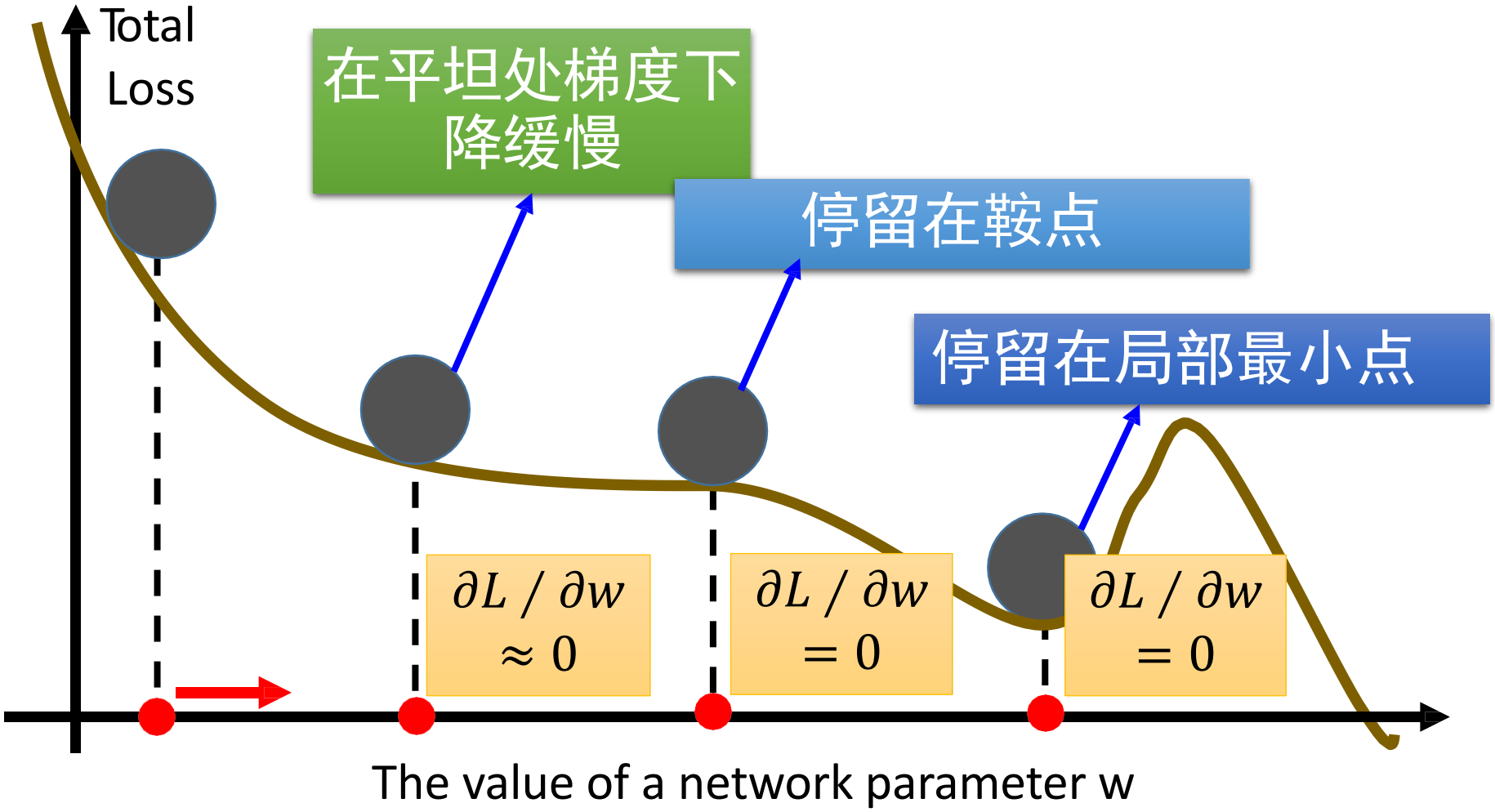
较大的学习速率

深度学习的秘诀


选择合适的损失函数
小批量
新的激活函数
自适应的学习速率
动量
在训练集上
效果好?
YES
在测试集上的
效果好?
YES

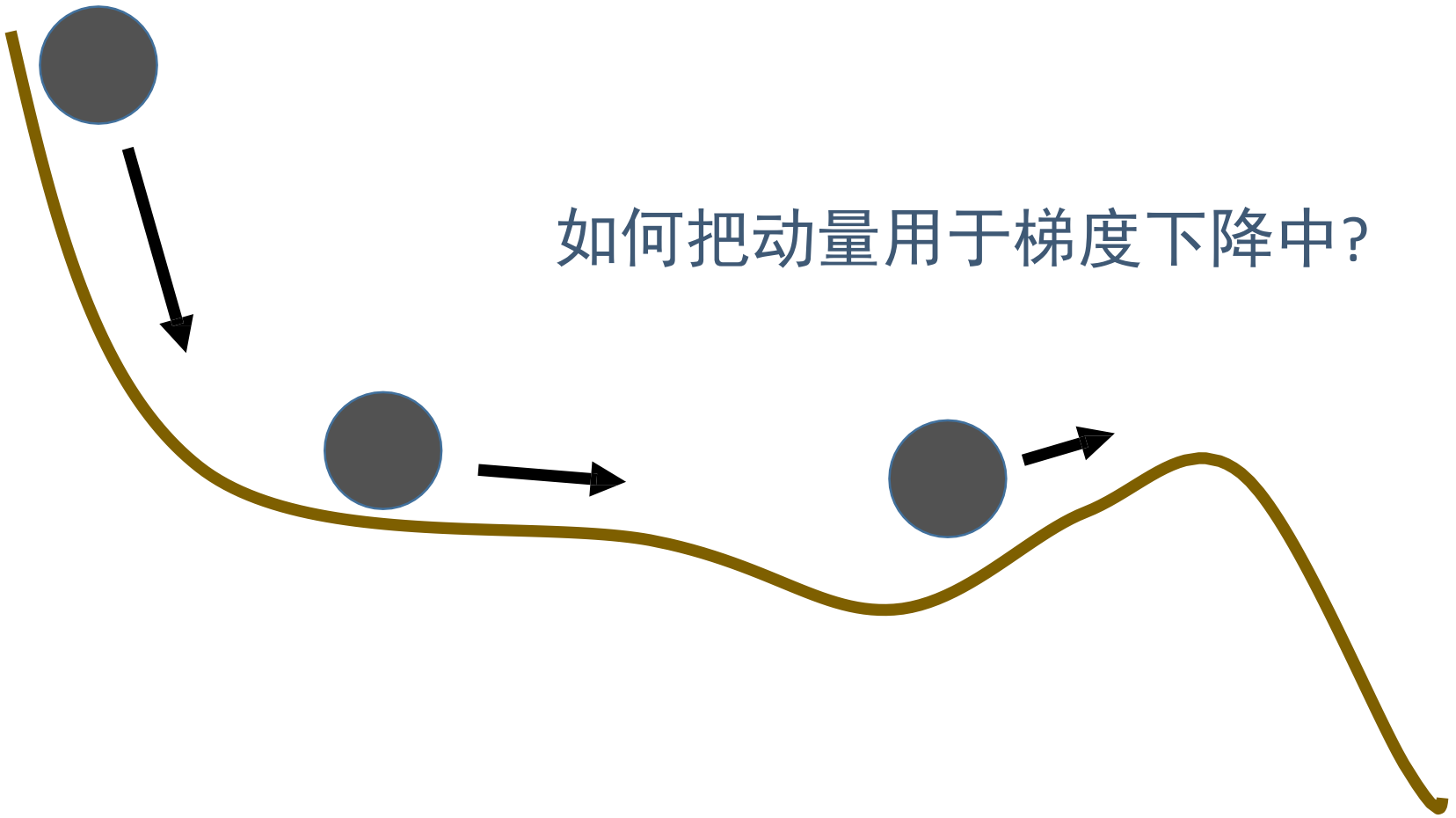
动量

- 很难找到最佳网络参数



动量

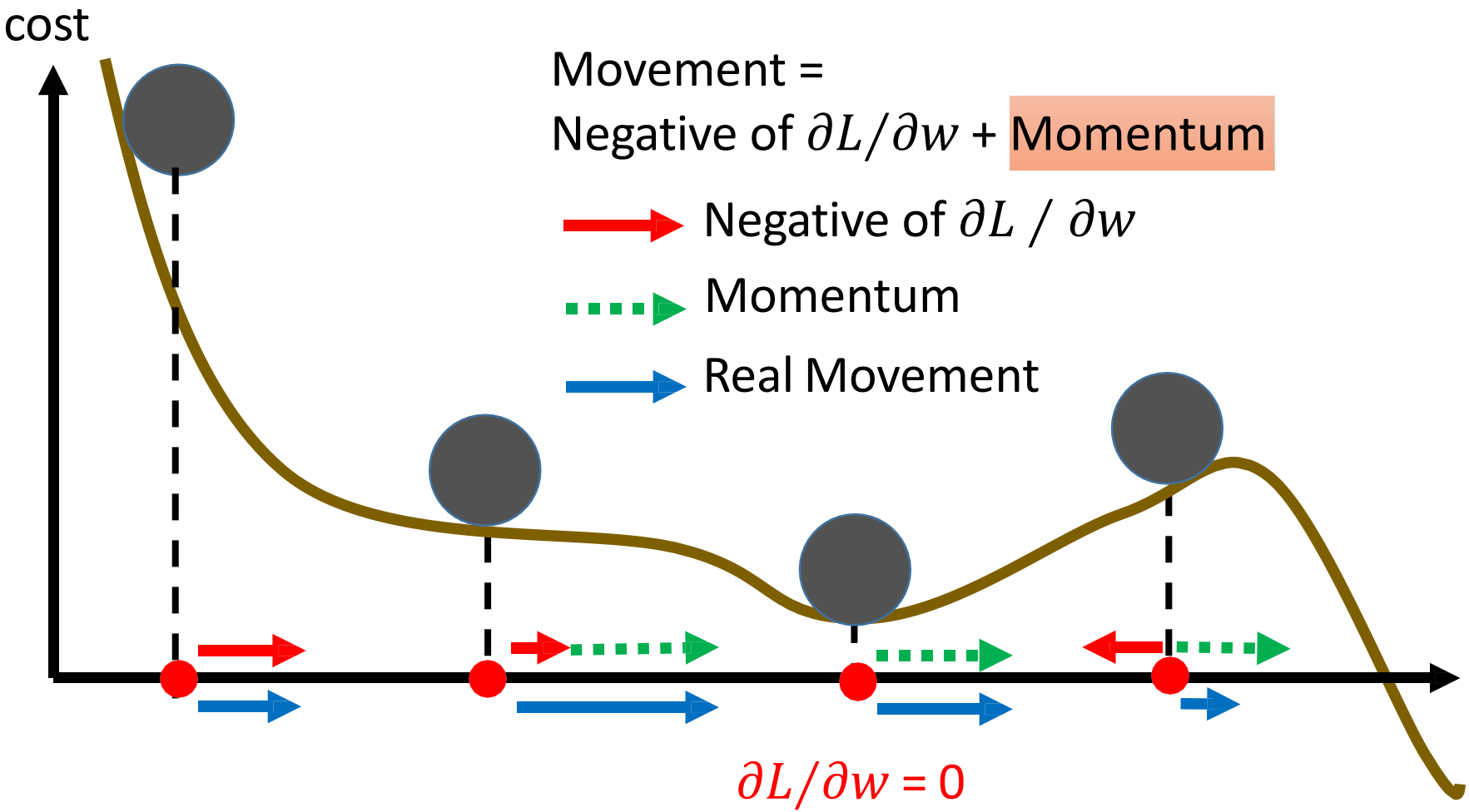
- 在真实的世界里…



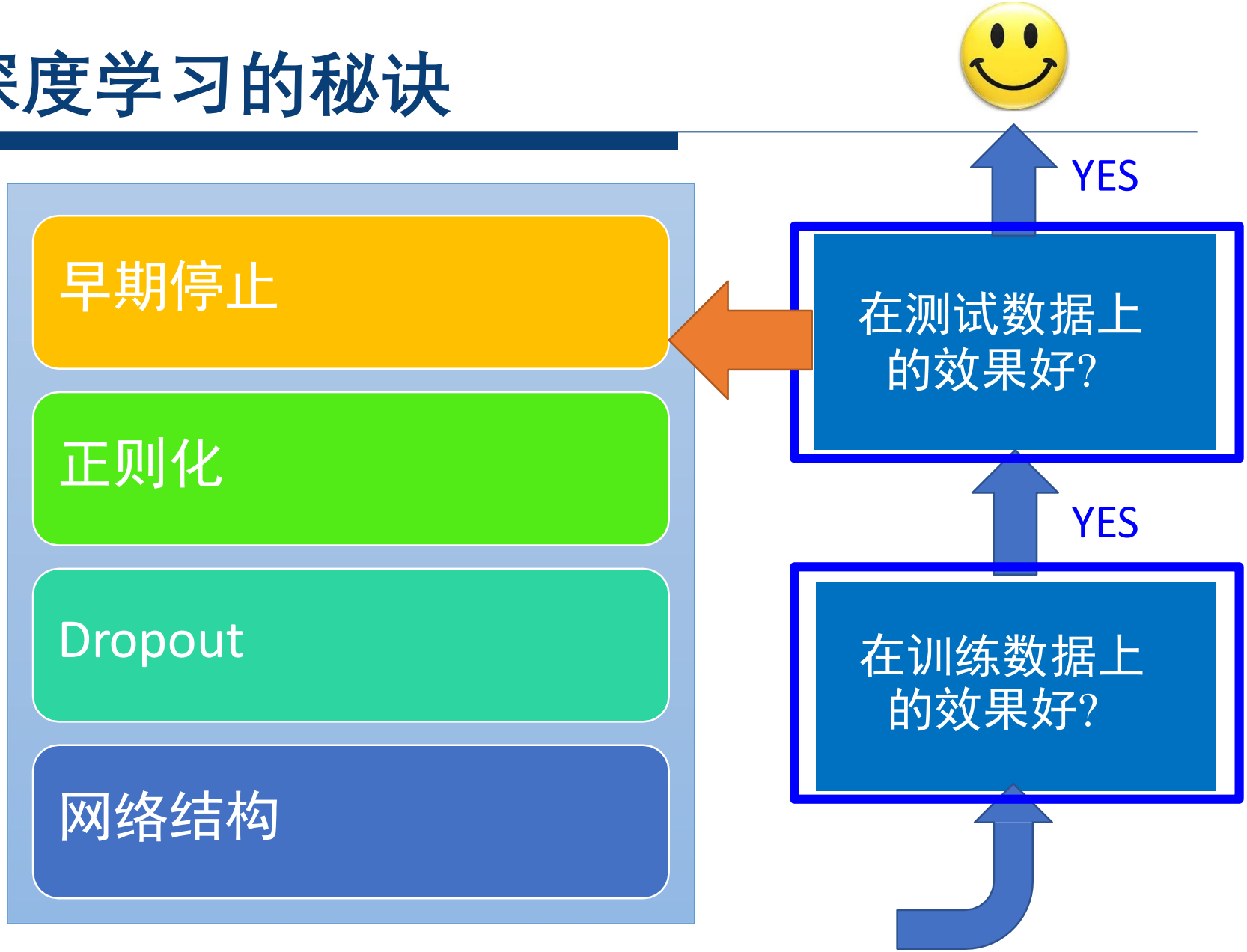
如何把动量用于梯度下降中?

动量

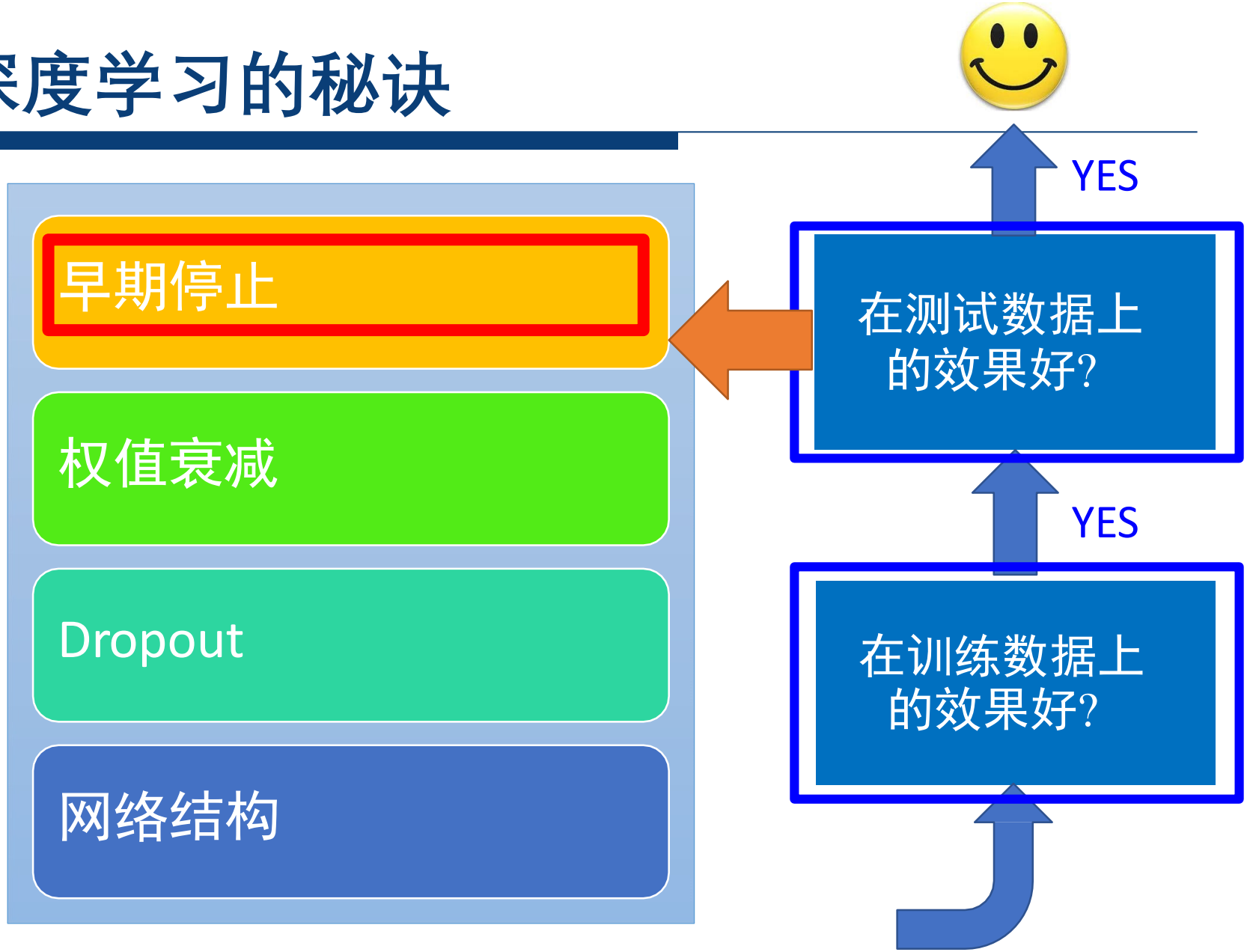
- 仍然不能保证到达全局最优，但给了一些希望 …



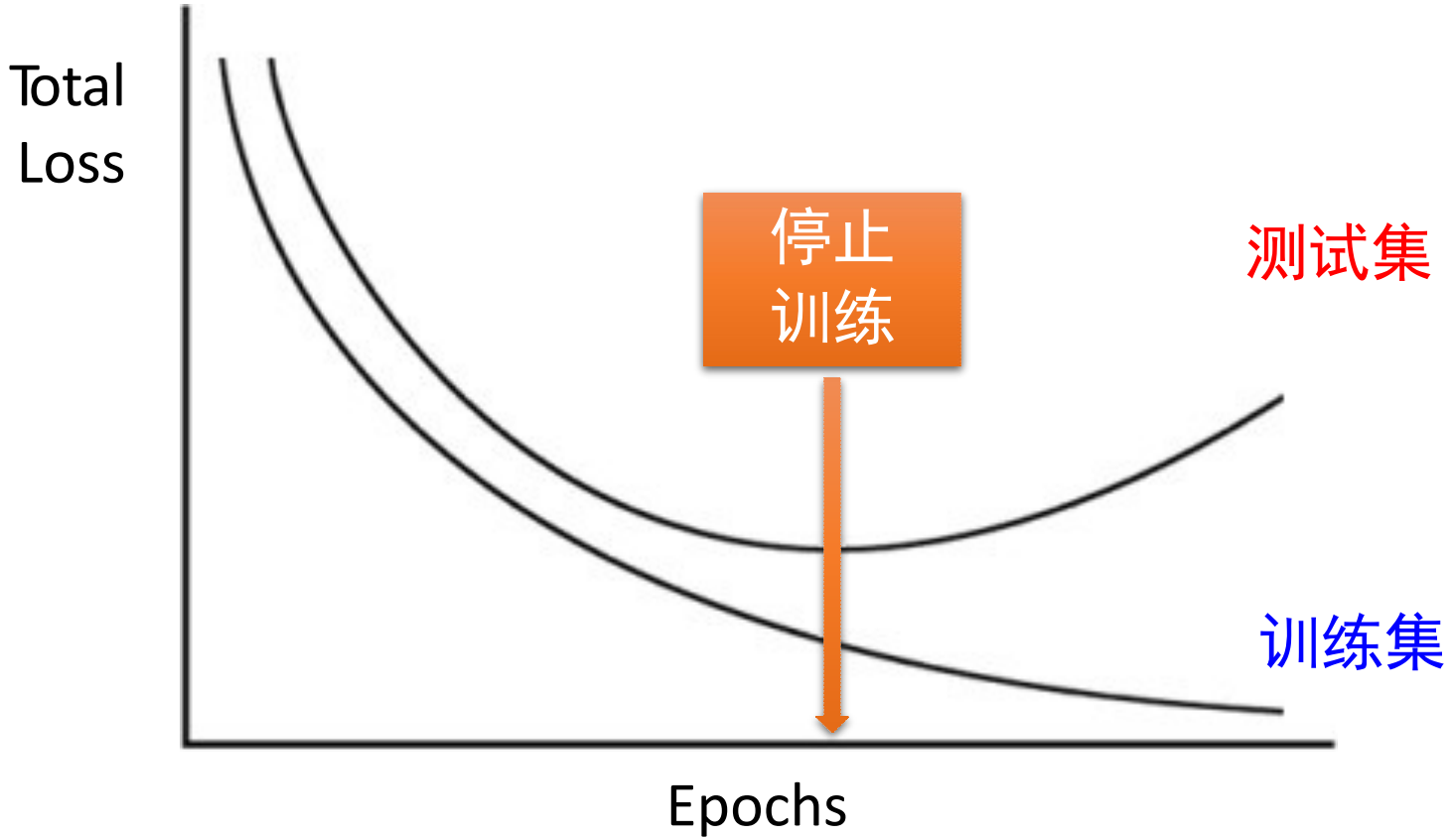
深度学习的秘诀



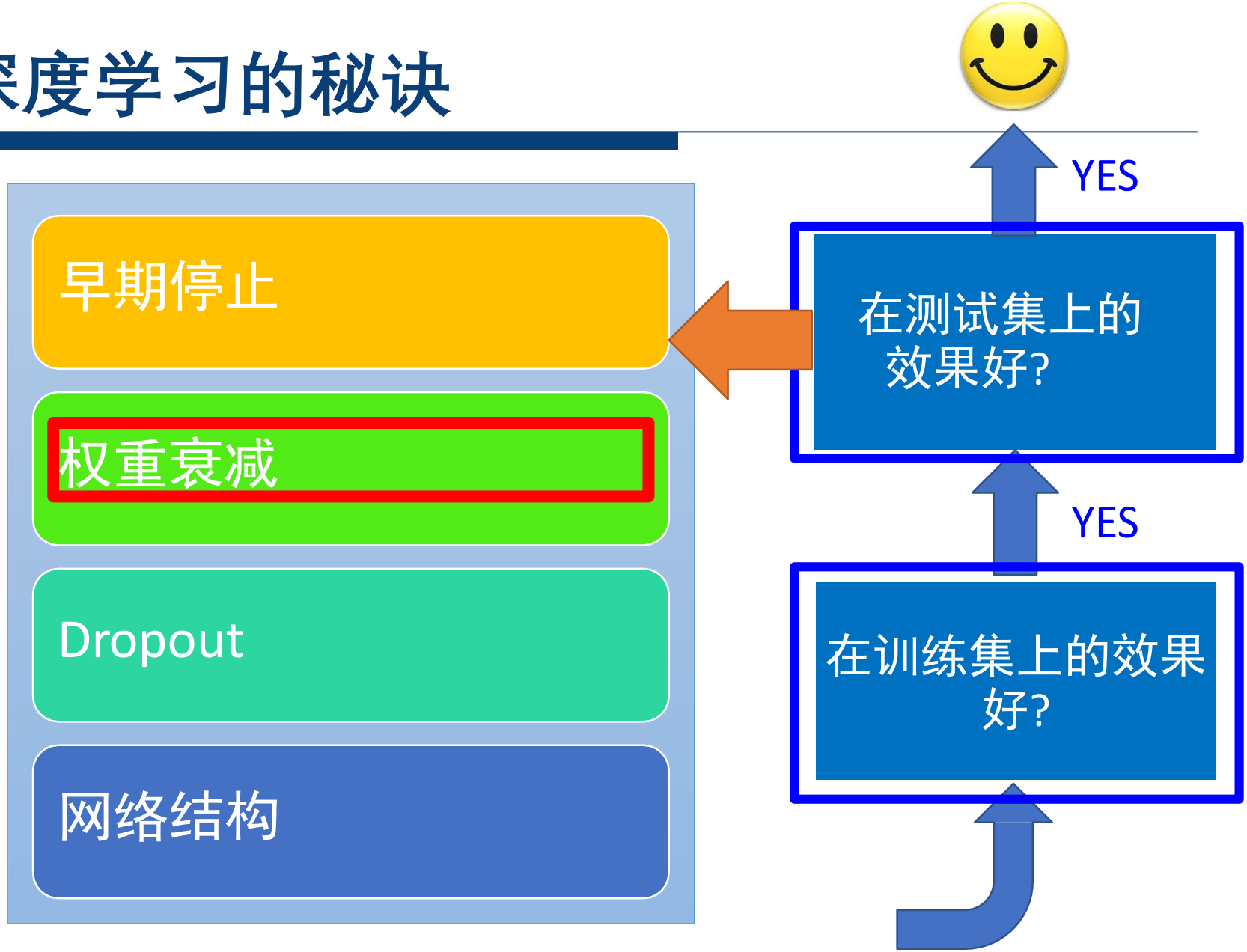
深度学习的秘诀



早期停止

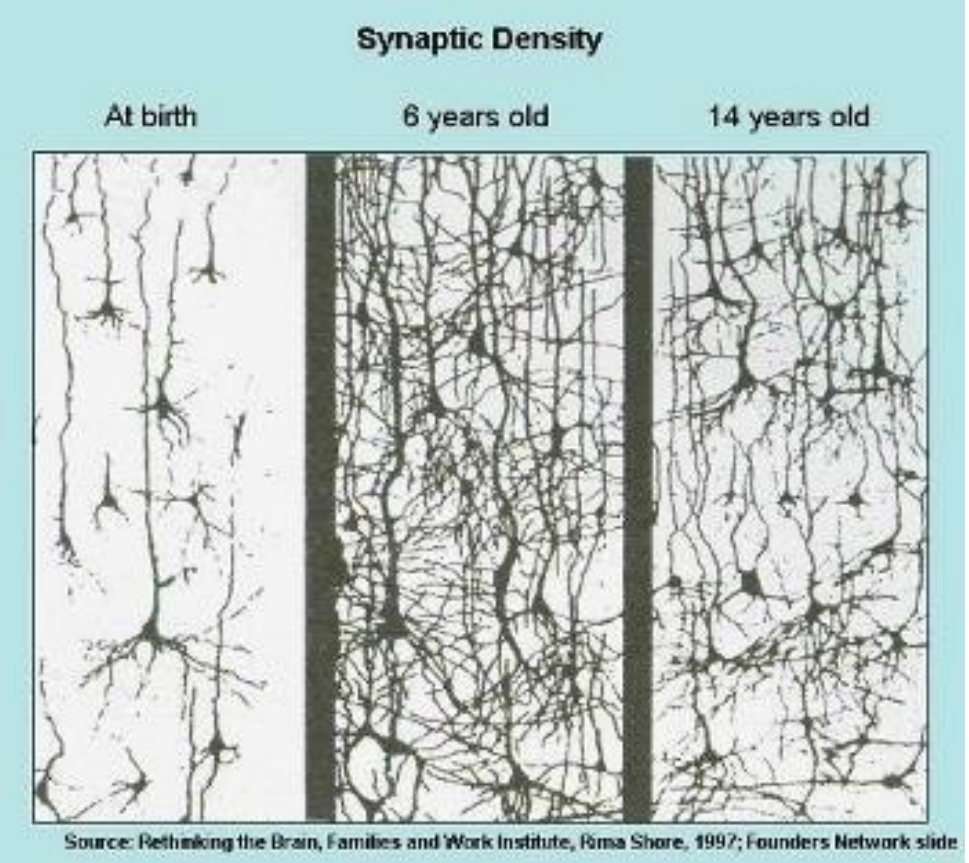


深度学习的秘诀

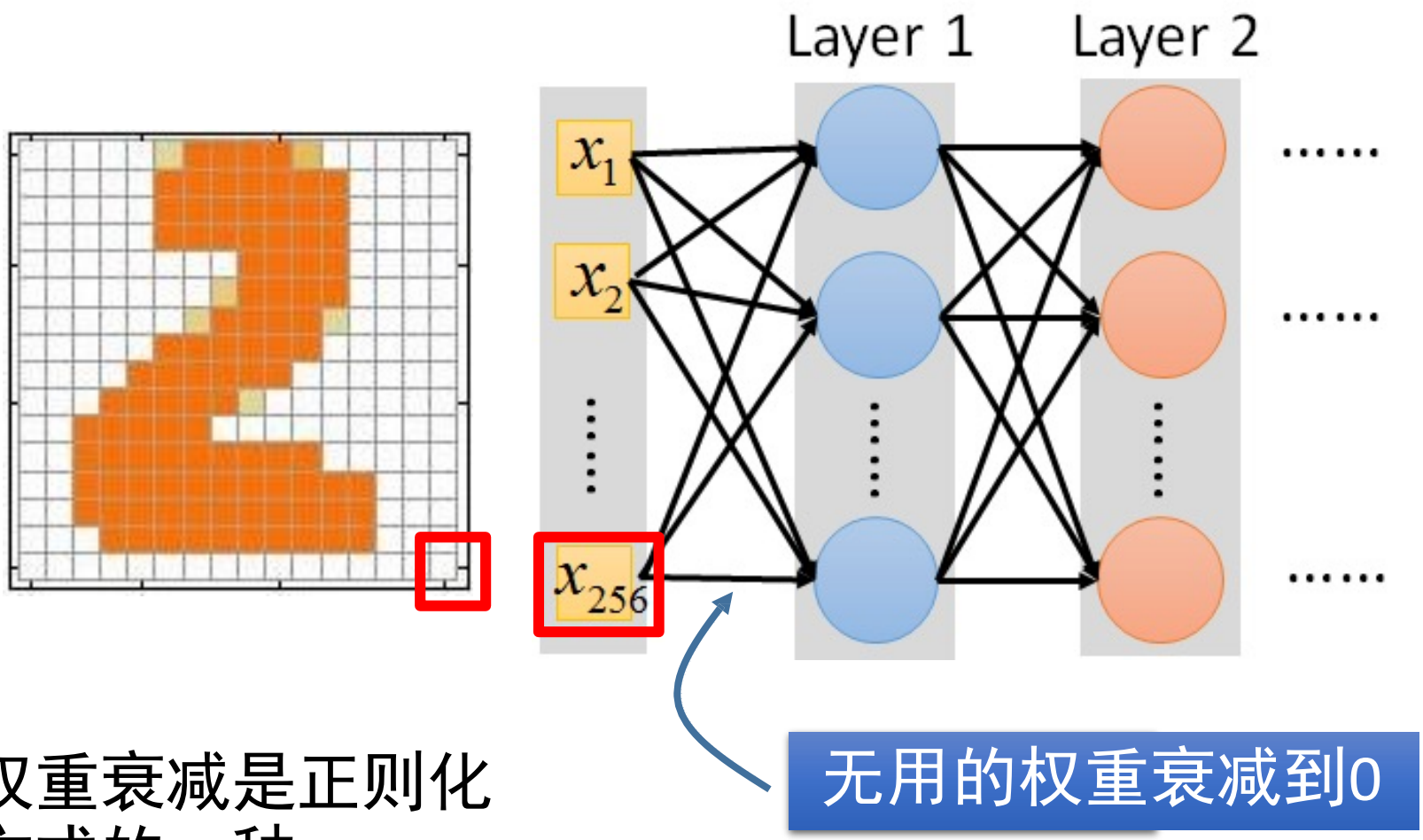


权重衰减

- 我们的大脑删除掉了神经元之间无用的联系
  - 对机器的大脑做同样的事情可以提高性能

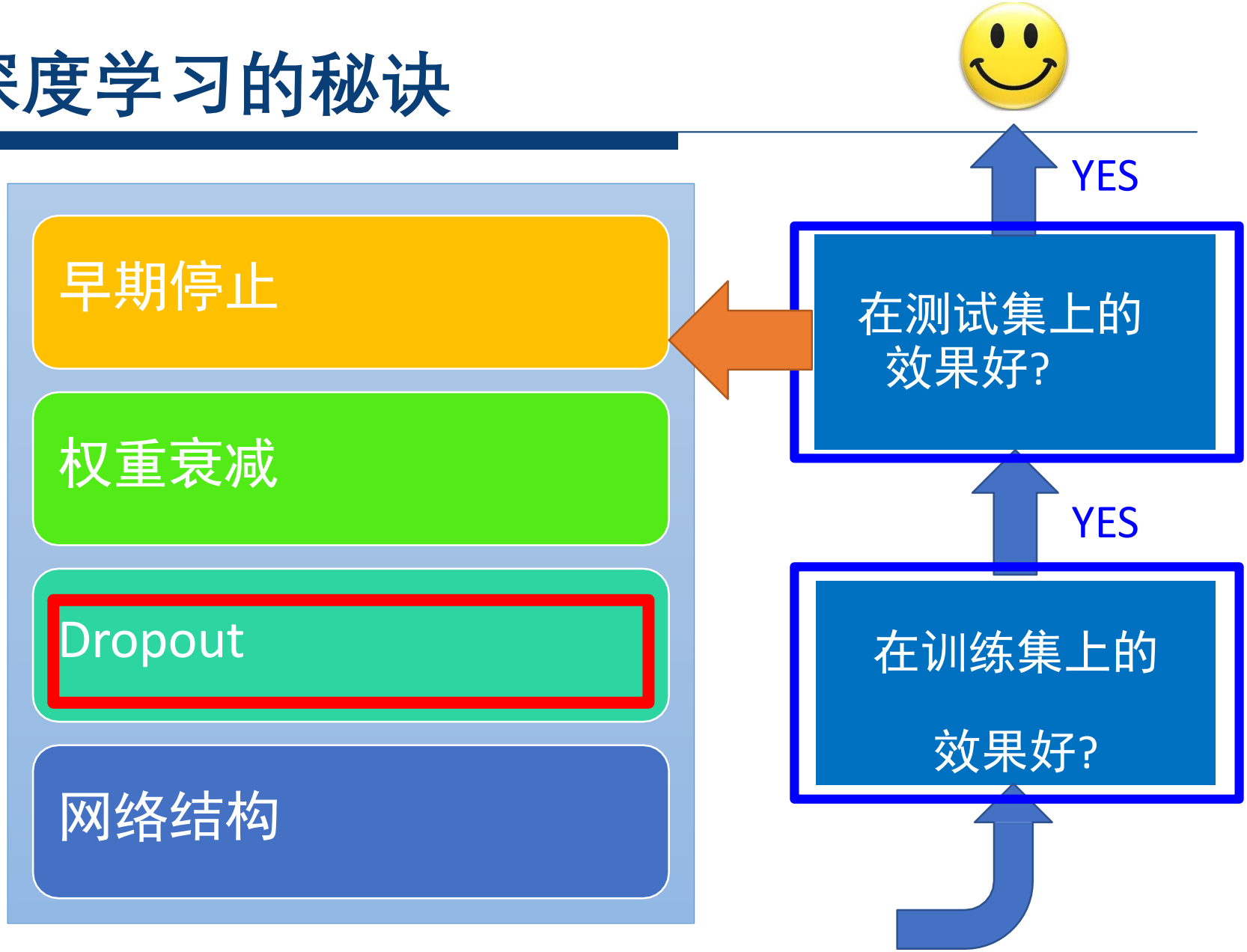


权重衰减



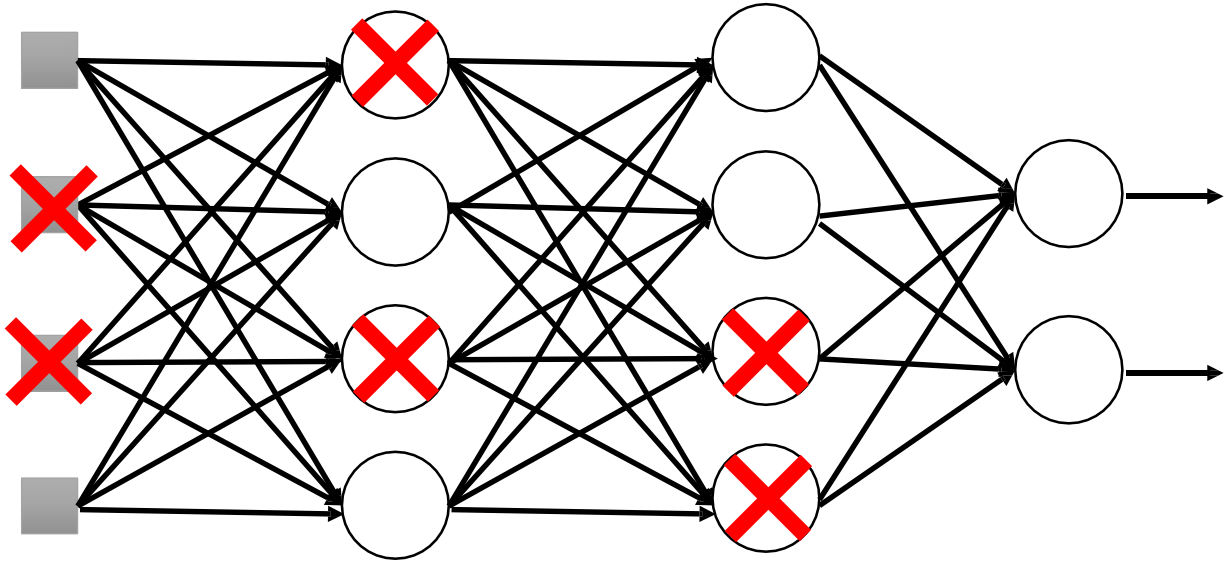
权重衰减是正则化方式的一种

深度学习的秘诀



Dropout

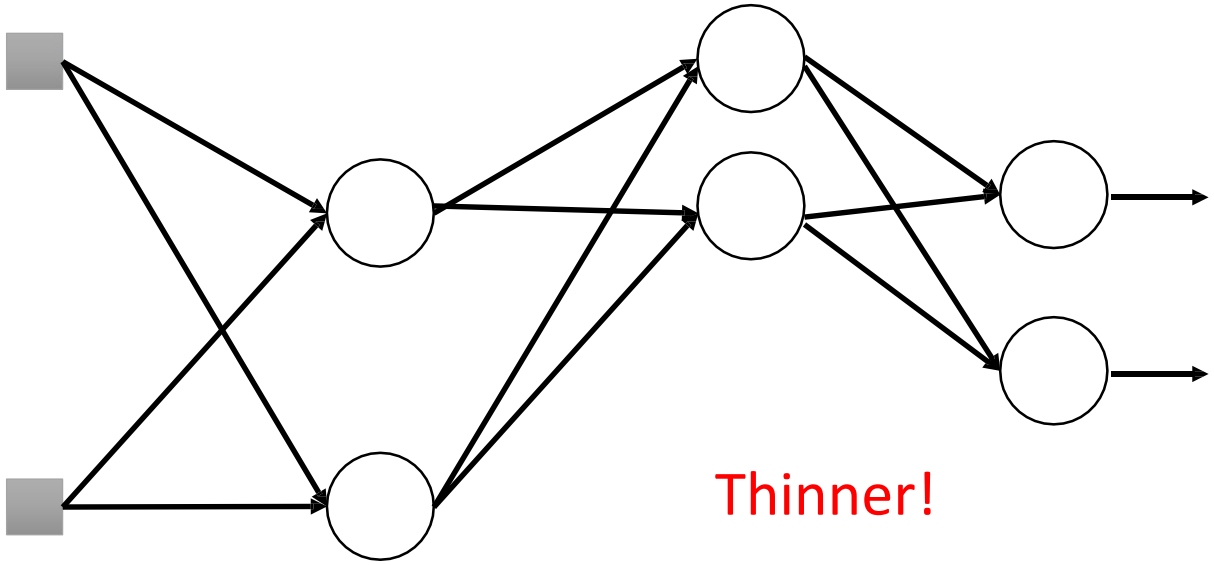
训练:



- **每次更新参数之前**
  - 每个神经元 都有p%的概率被丢弃

Dropout

训练:



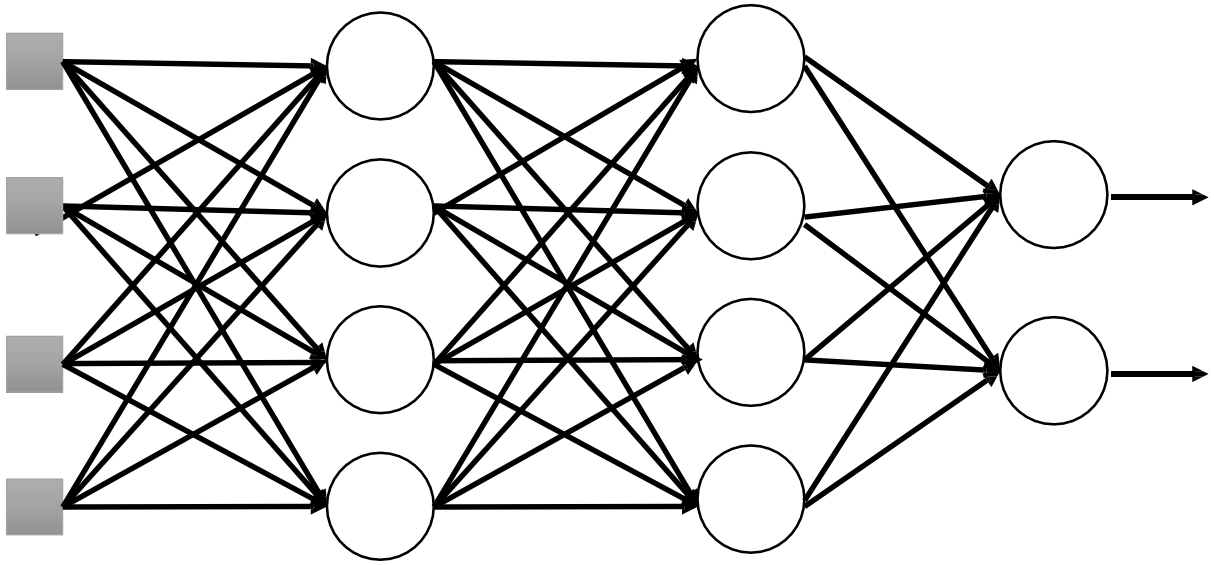
- **每个神经元 都有p%的概率被丢弃**
  - 每个神经元 都有p%的概率被丢弃

    网络结构发生了变化
  - 使用新的网络去训练

对于每个小批量数据，我们重新采样来dropout神经元

Dropout

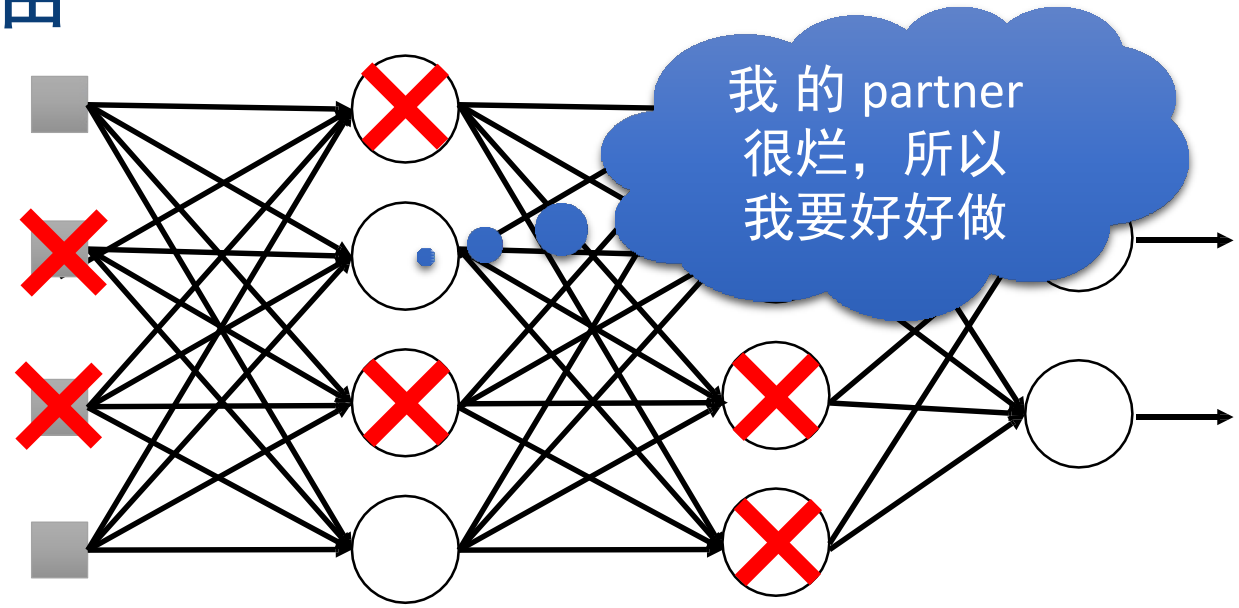
测试:



- **没有 dropout**
  - 如果dropout rate 在训练集上是 p%，所有的权重值乘以 (1-p)%
  - 假设 dropout rate 是 50%.
    If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout

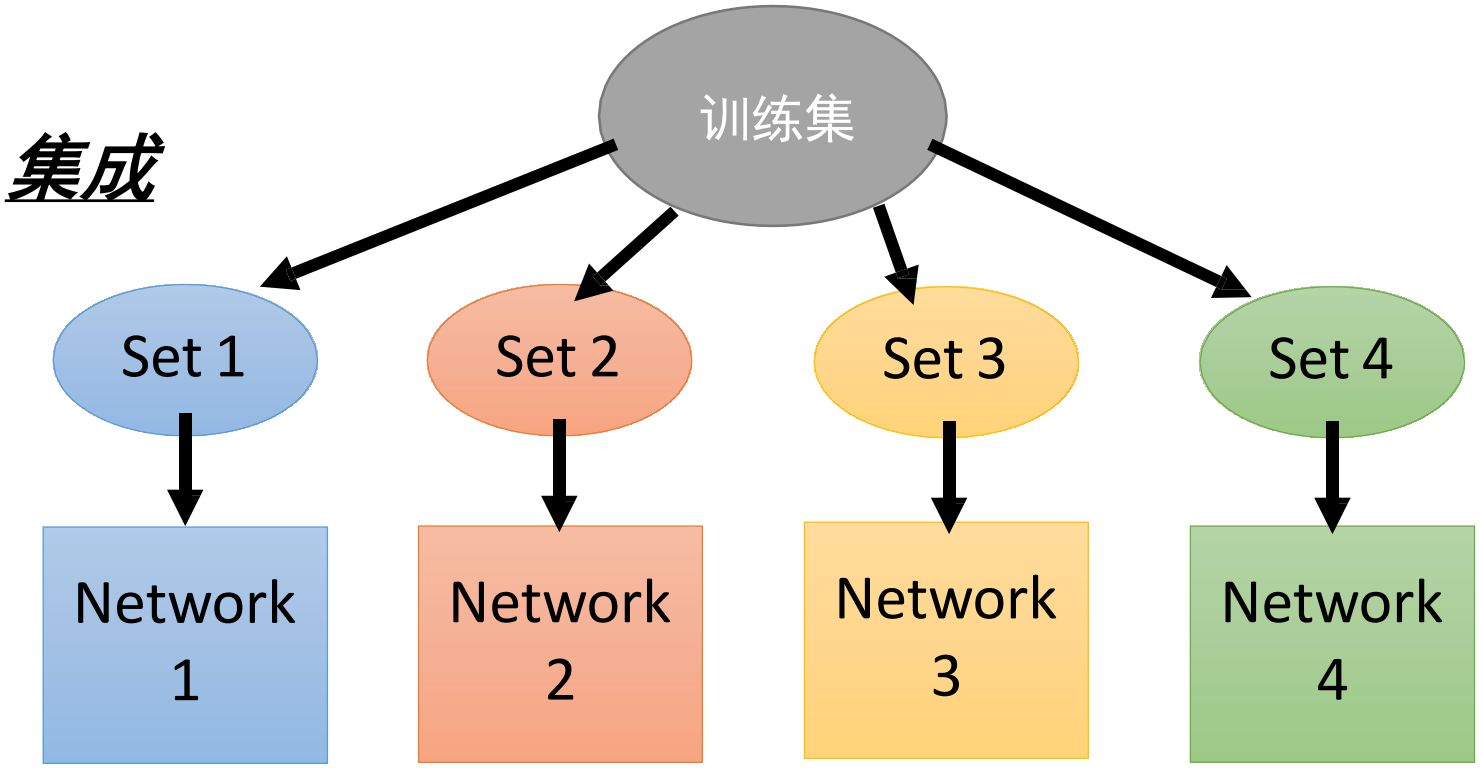
直观理由



- 团队合作时，如果大家都期待合作伙伴做好工作，最后什么都不会做
- 但是如果你知道你的搭档会被抛弃，你会做的更好
- 当测试时, 实际上没有人被抛弃, 所有最终能取得不错的结果

Dropout

- 一种集成方法

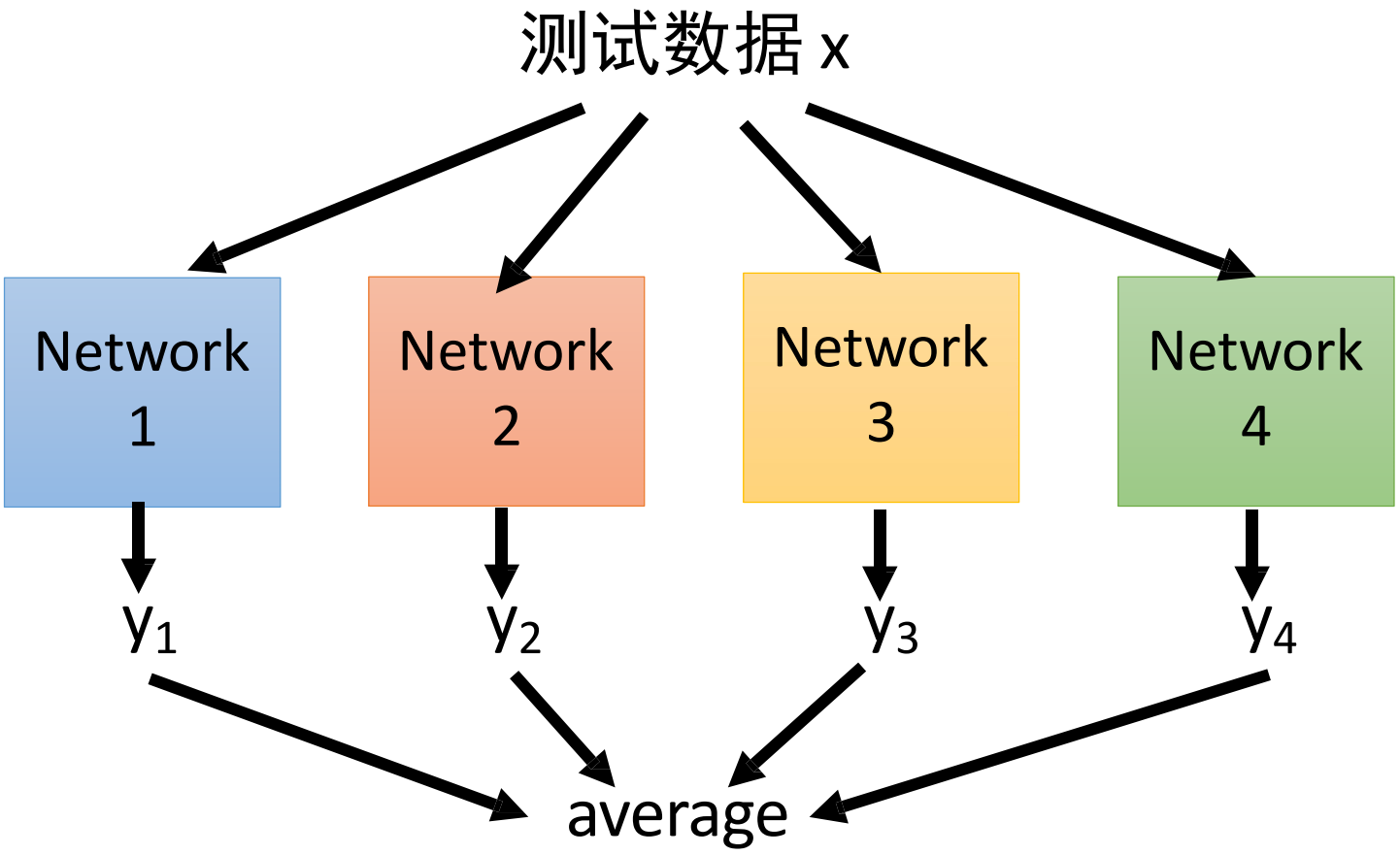


训练一些具有不同结构的网络

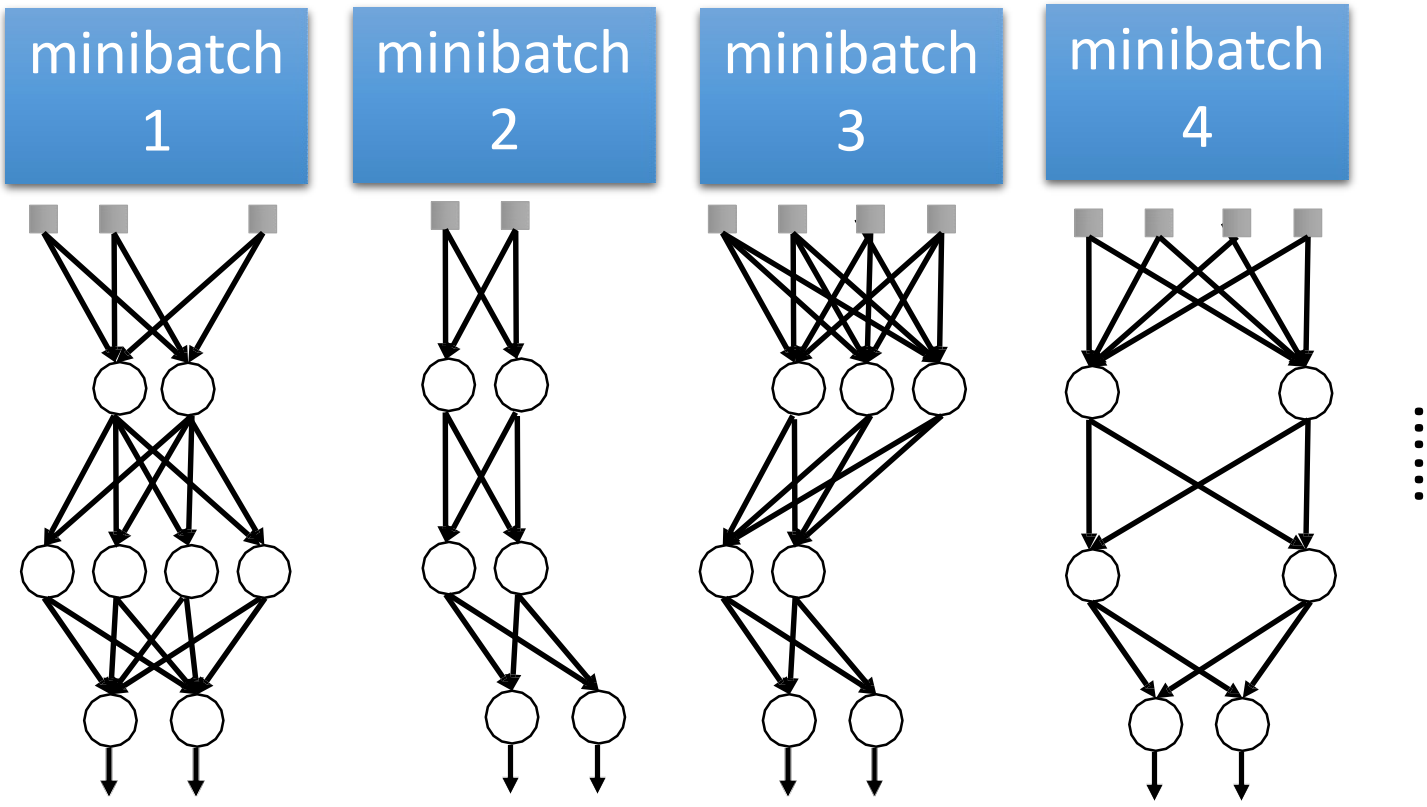
Dropout

口 一种集成方法

集成



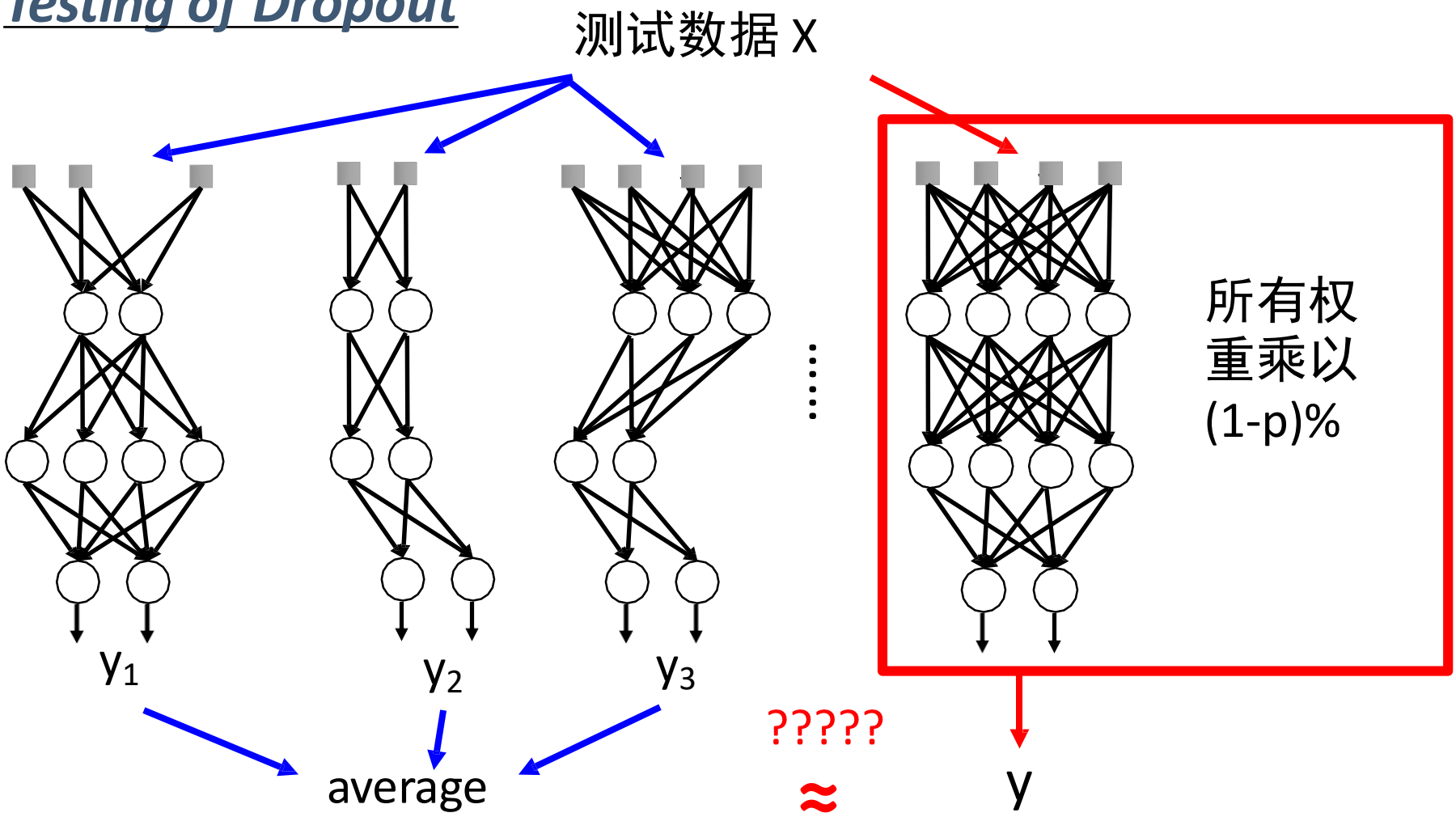
Dropout 是集成方法的一种



- 使用一个小批量数据来训练一个网络
- 网络中的一些参数是可以共享的

Dropout

Testing of Dropout



深度学习的秘诀

YES

在测试集上的效果好?

YES

在训练集上的效果好?

早期停止

正则化

Dropout

网络结构

CNN 就是一个很好的例子!

End