0基础Python科研实践

第五课：神经网络概述

刘天禹

2020年3月28日

机器学习分类

机器学习分类

按照训练

- 监督学习（supervised learning）
- 非监督学习 （unsupervised learning）

机器学习分类

按照训练

- 监督学习（supervised learning）
- 非监督学习 （unsupervised learning）

按照任务

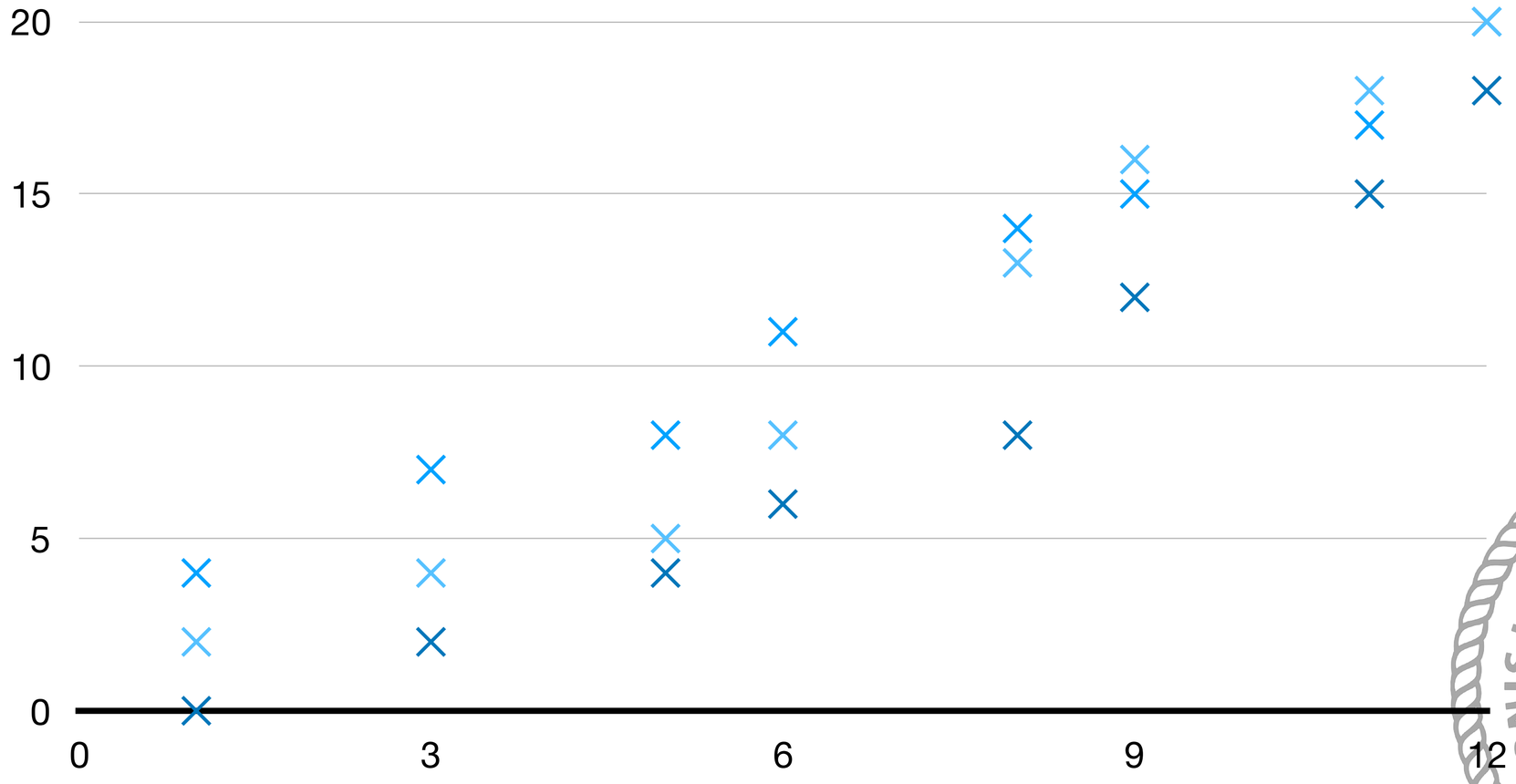
- 分类（Classification）
- 回归（Regression）

神经网络概述

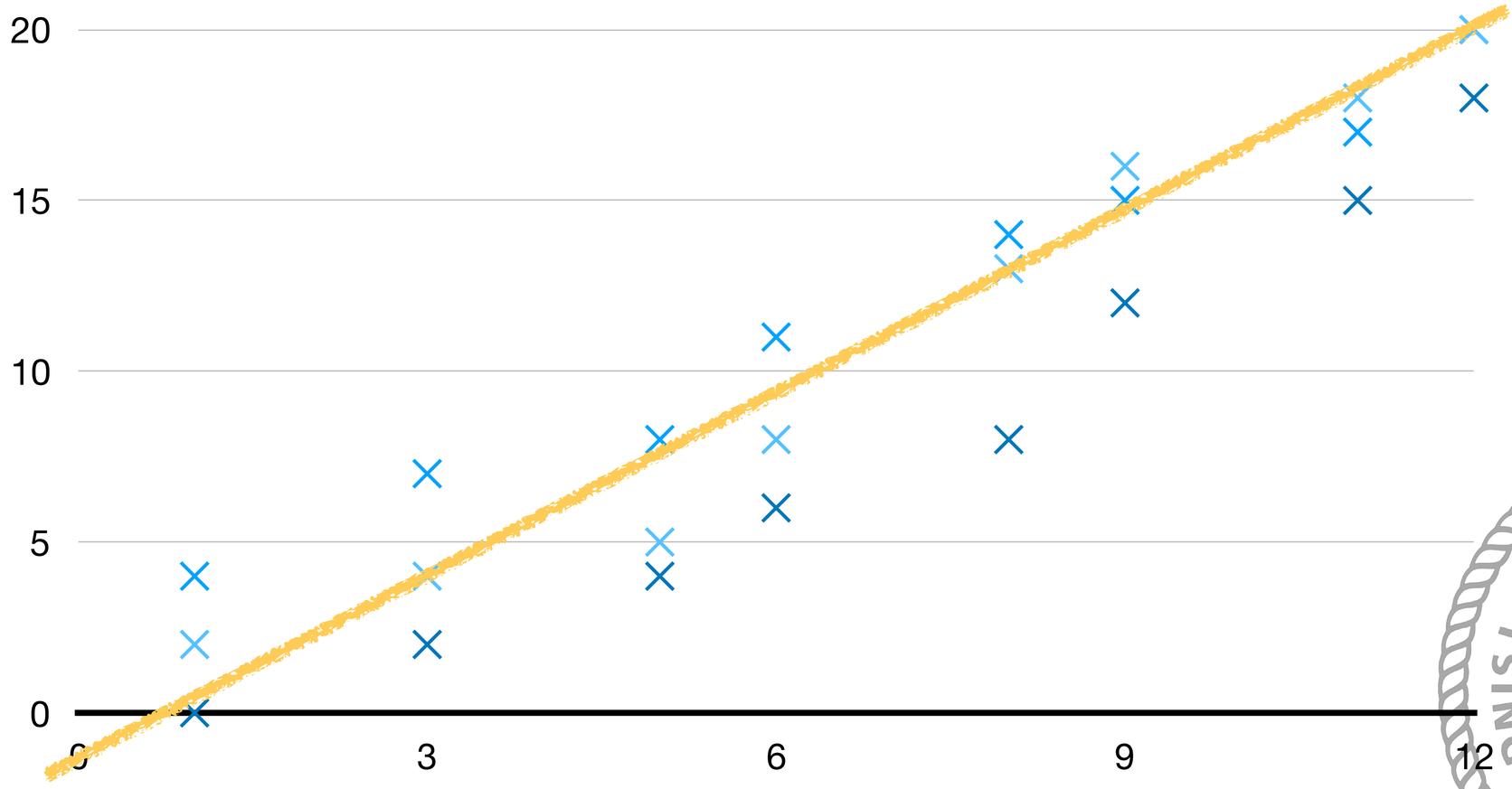
线性回归回顾

- 面积-房价
- 学习时间-分数
- 风力-雾霾
- 汽车价格-百公里加速度

线性回归

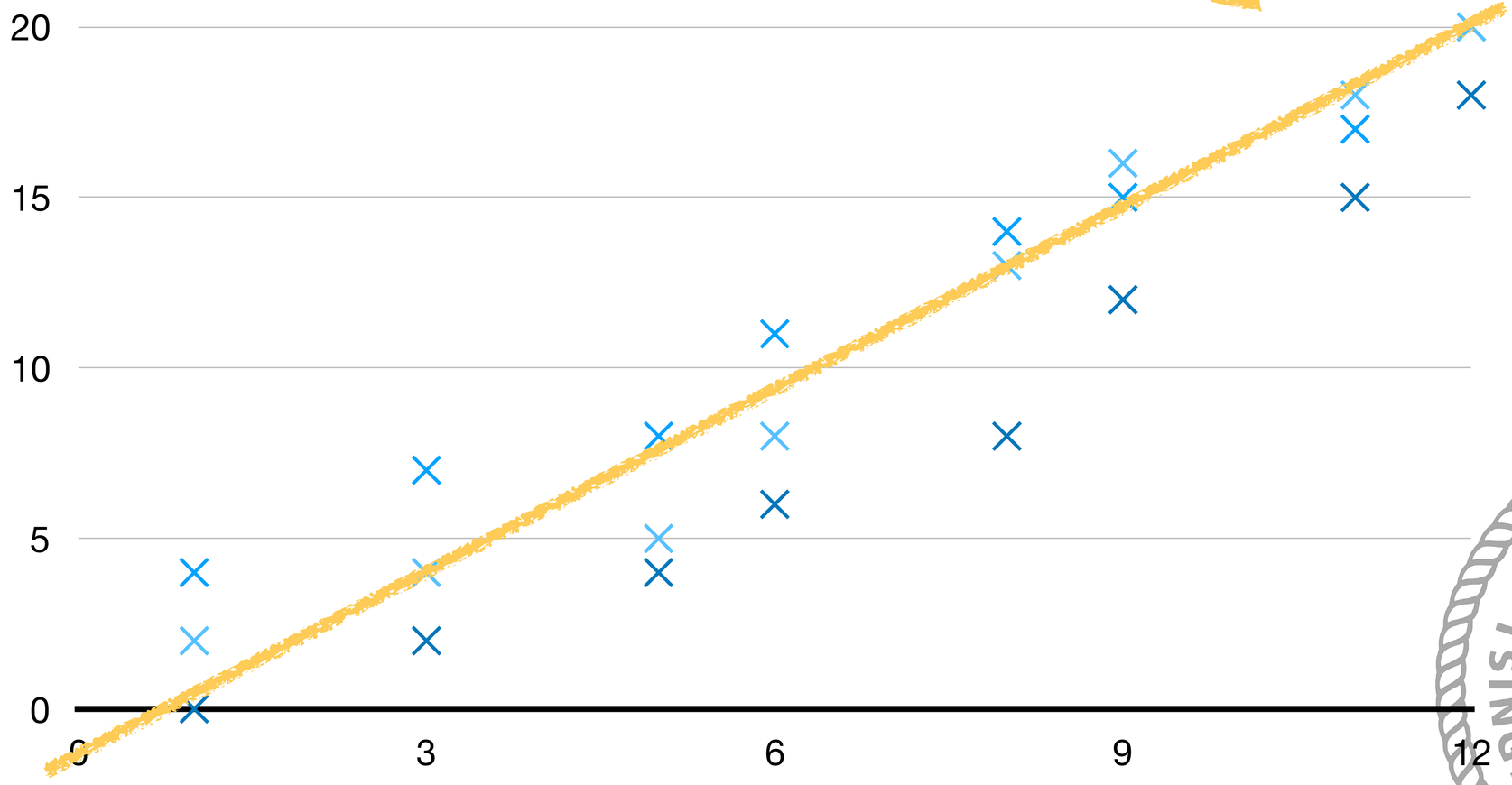


线性回归



线性回归

$$f(x) = \theta_0 * x + \theta_1$$



线性回归

Hypothesis

$$f(x) = \theta_0 * x + \theta_1$$

线性回归

Hypothesis

$$f(x) = \theta_0 * x + \theta_1$$

Cost Function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{n} (y_i - f(\theta, x_i))^2$$

线性回归

Hypothesis

$$f(x) = \theta_0 * x + \theta_1$$

Cost Function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{n} (y_i - f(\theta, x_i))^2$$

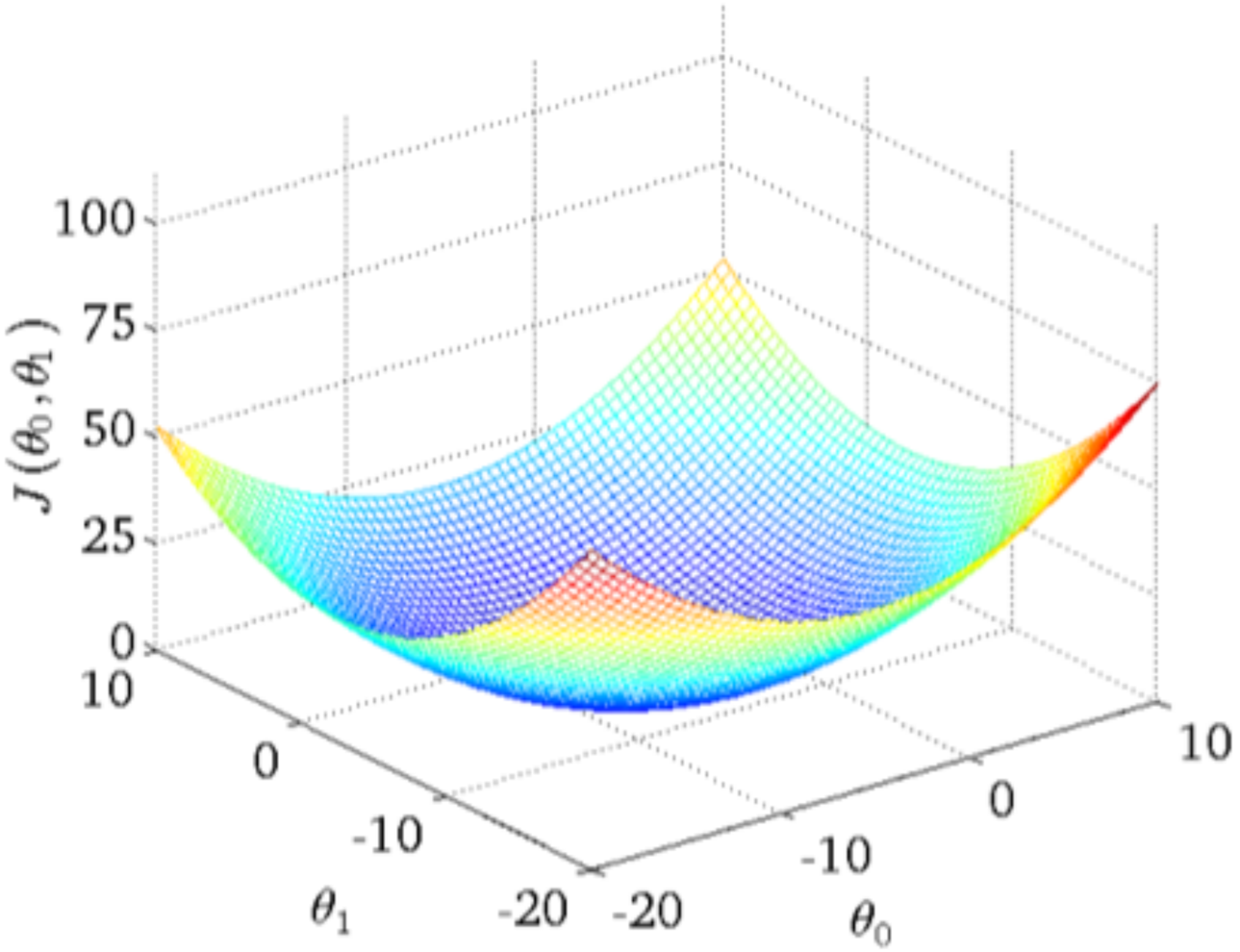
Target

$$arg\, min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

线性回归

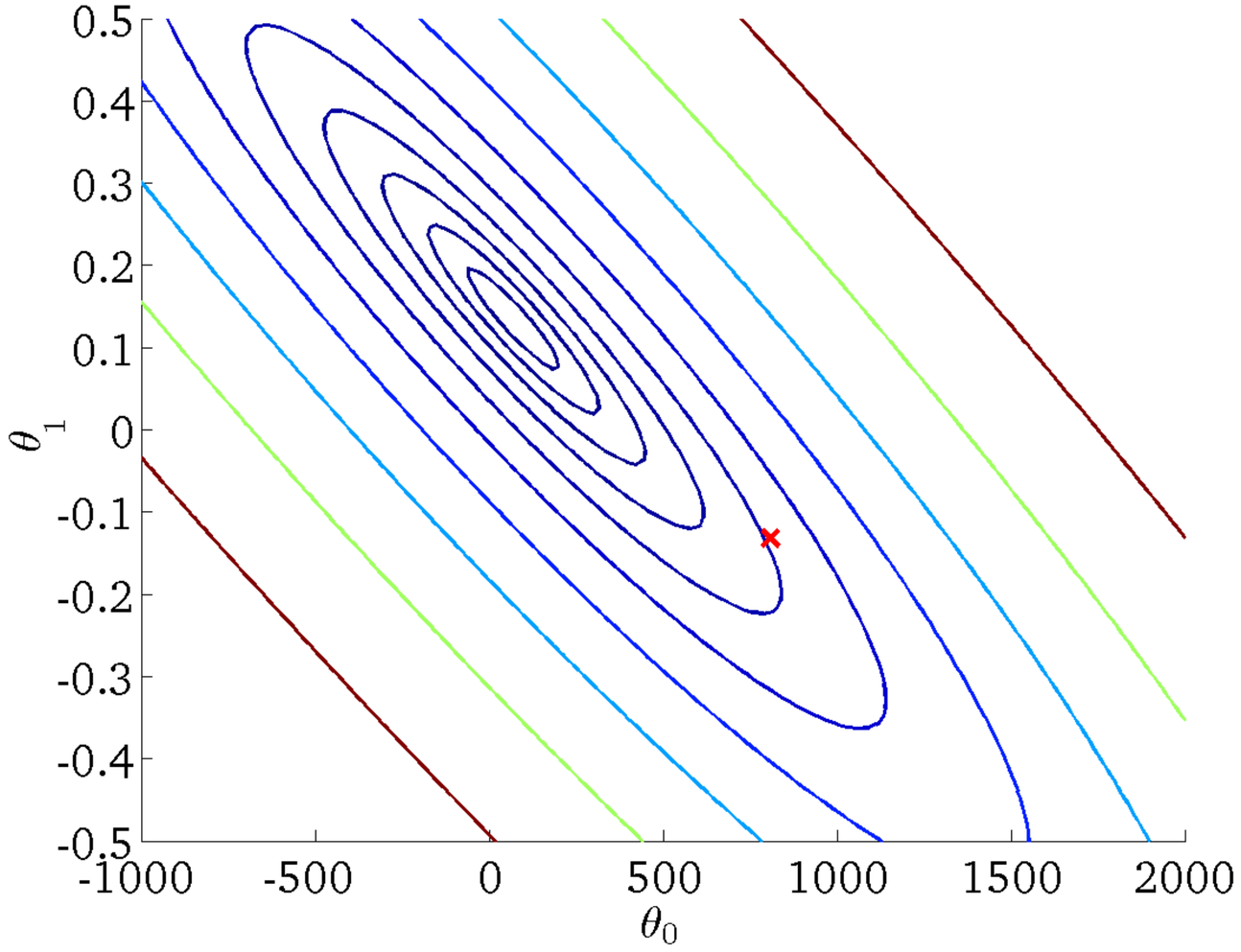
$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

线性回归



$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

线性回归



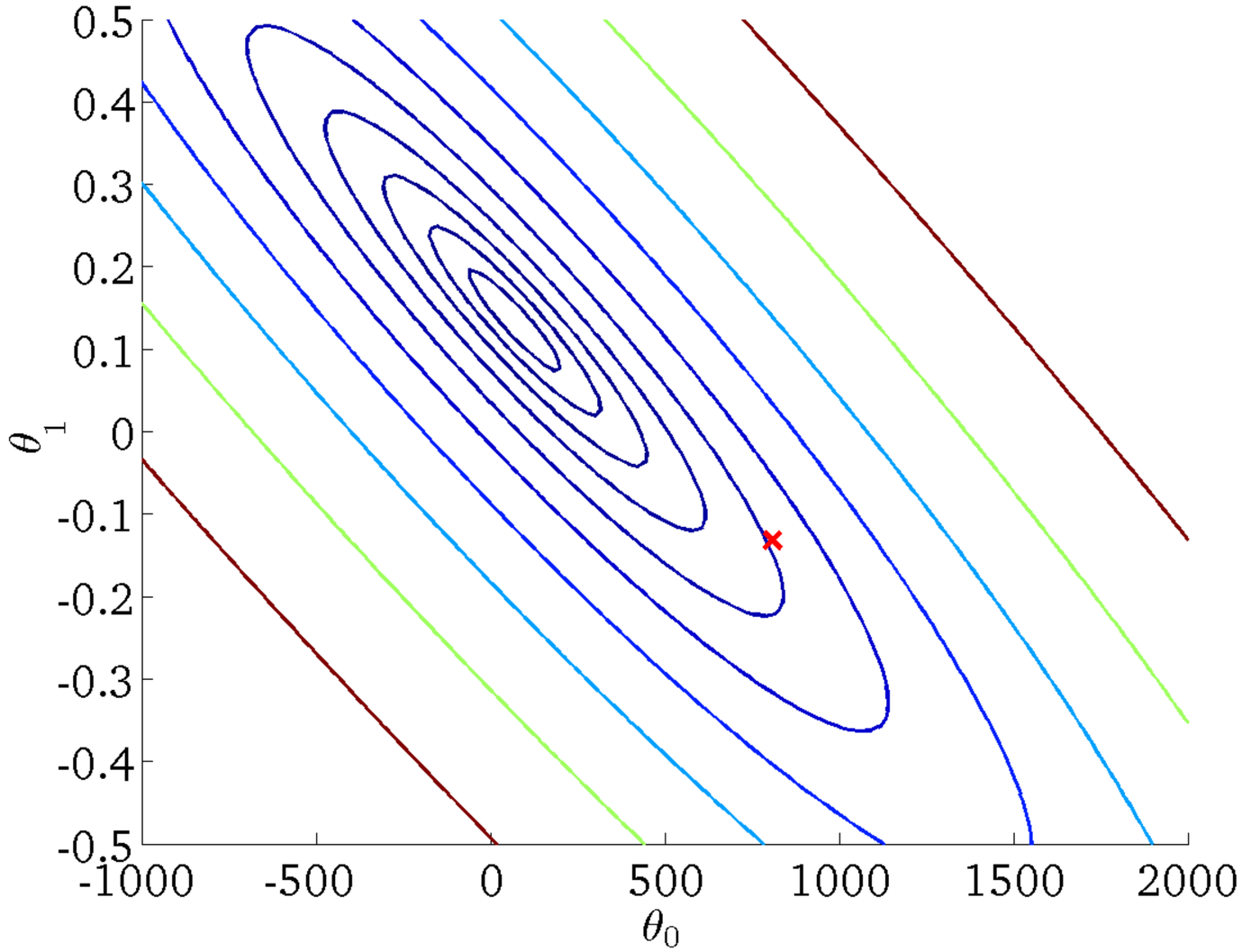
$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

$$J(\theta_0, \theta_1)$$

梯度下降

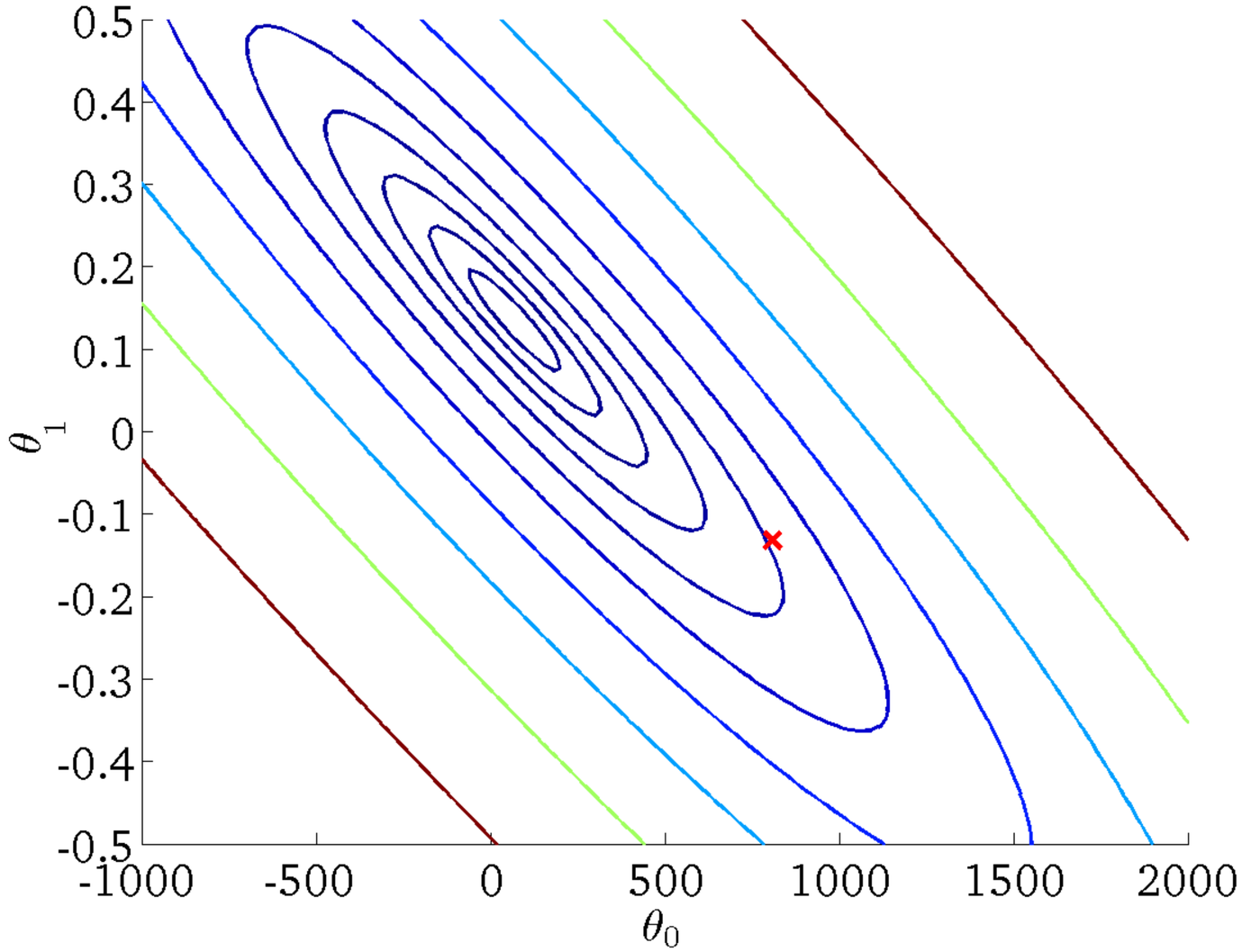
- 梯度的本意是一个向量（矢量），表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）。

梯度下降



$$J(\theta_0, \theta_1)$$

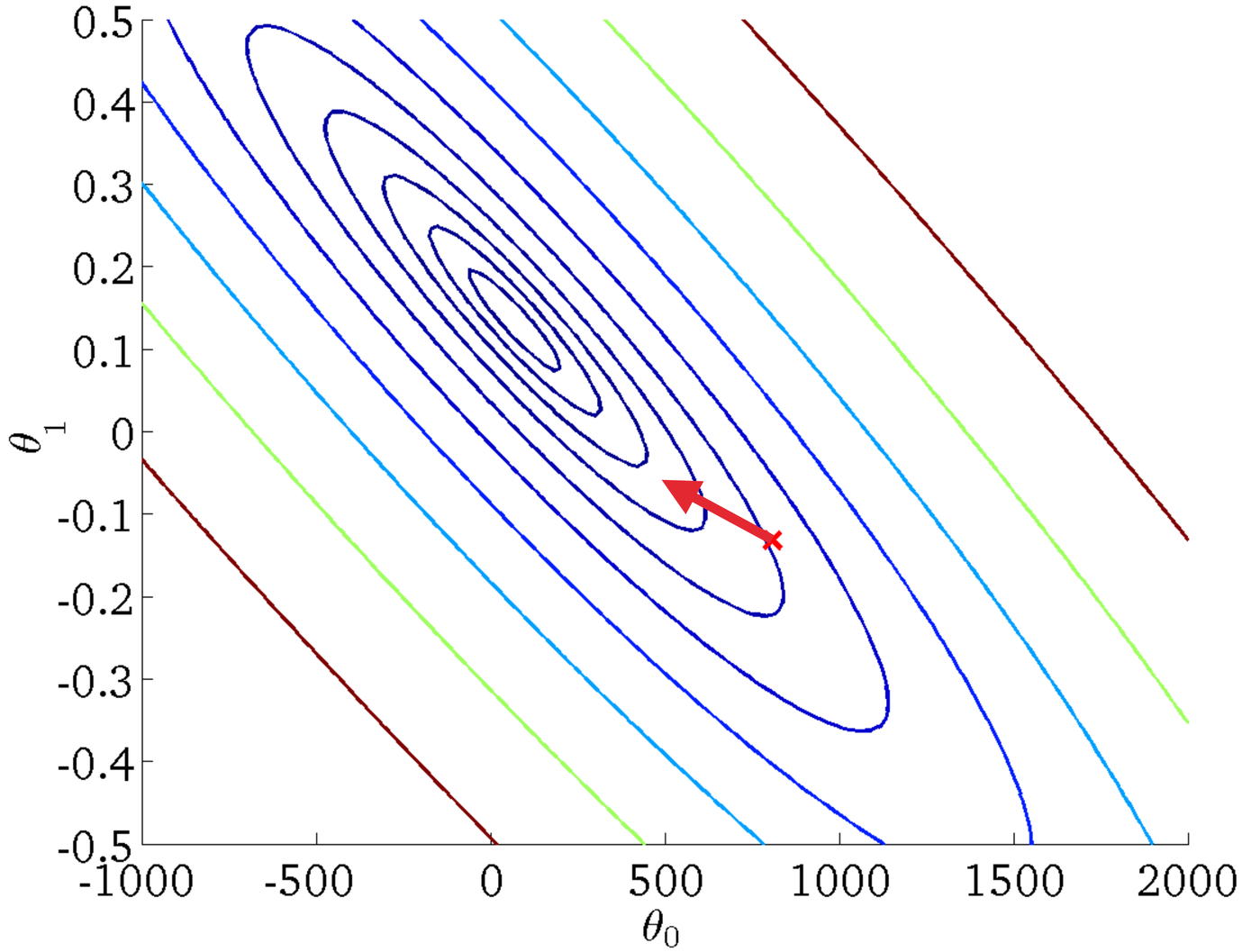
梯度下降



$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

$$J(\theta_0, \theta_1)$$

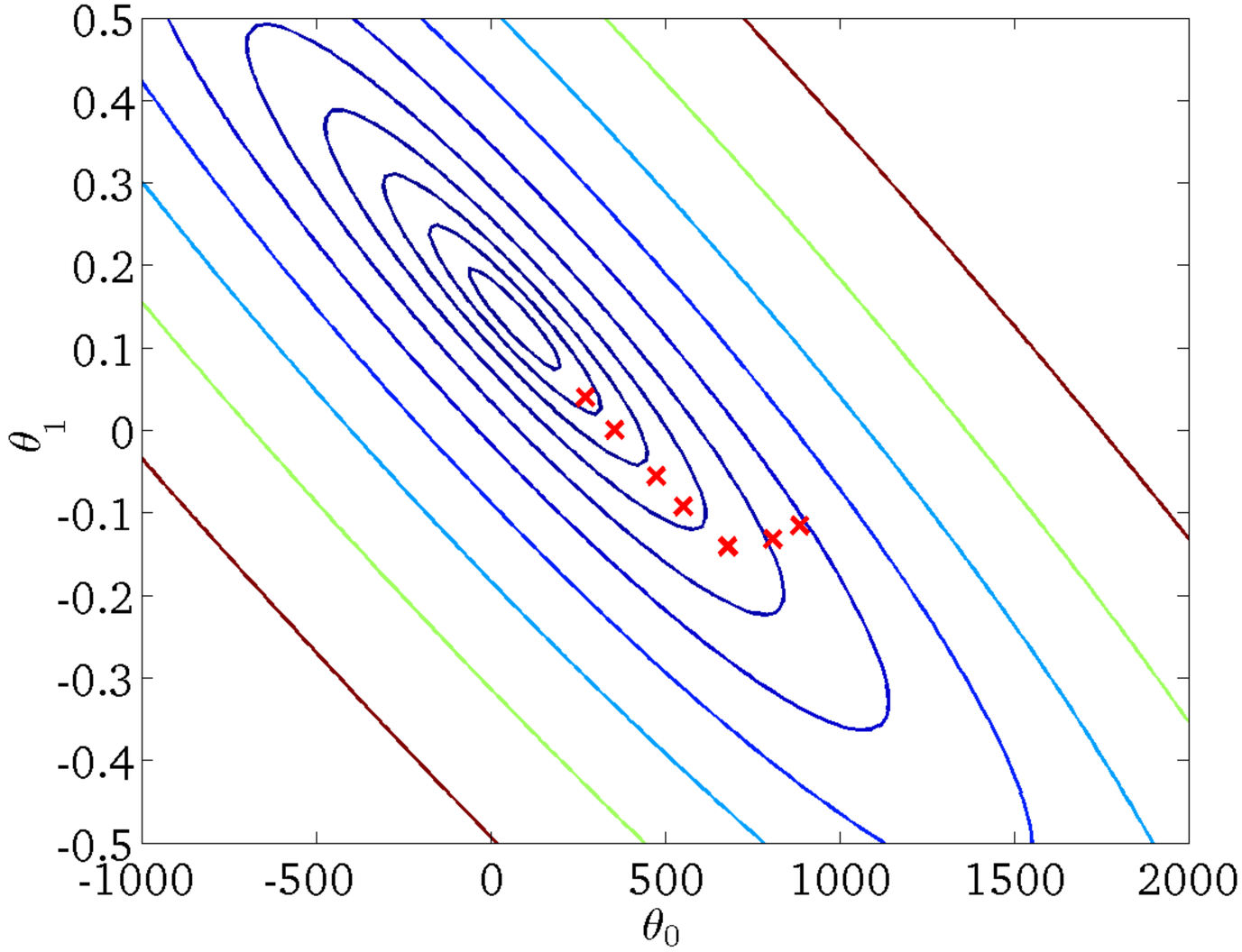
梯度下降



$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

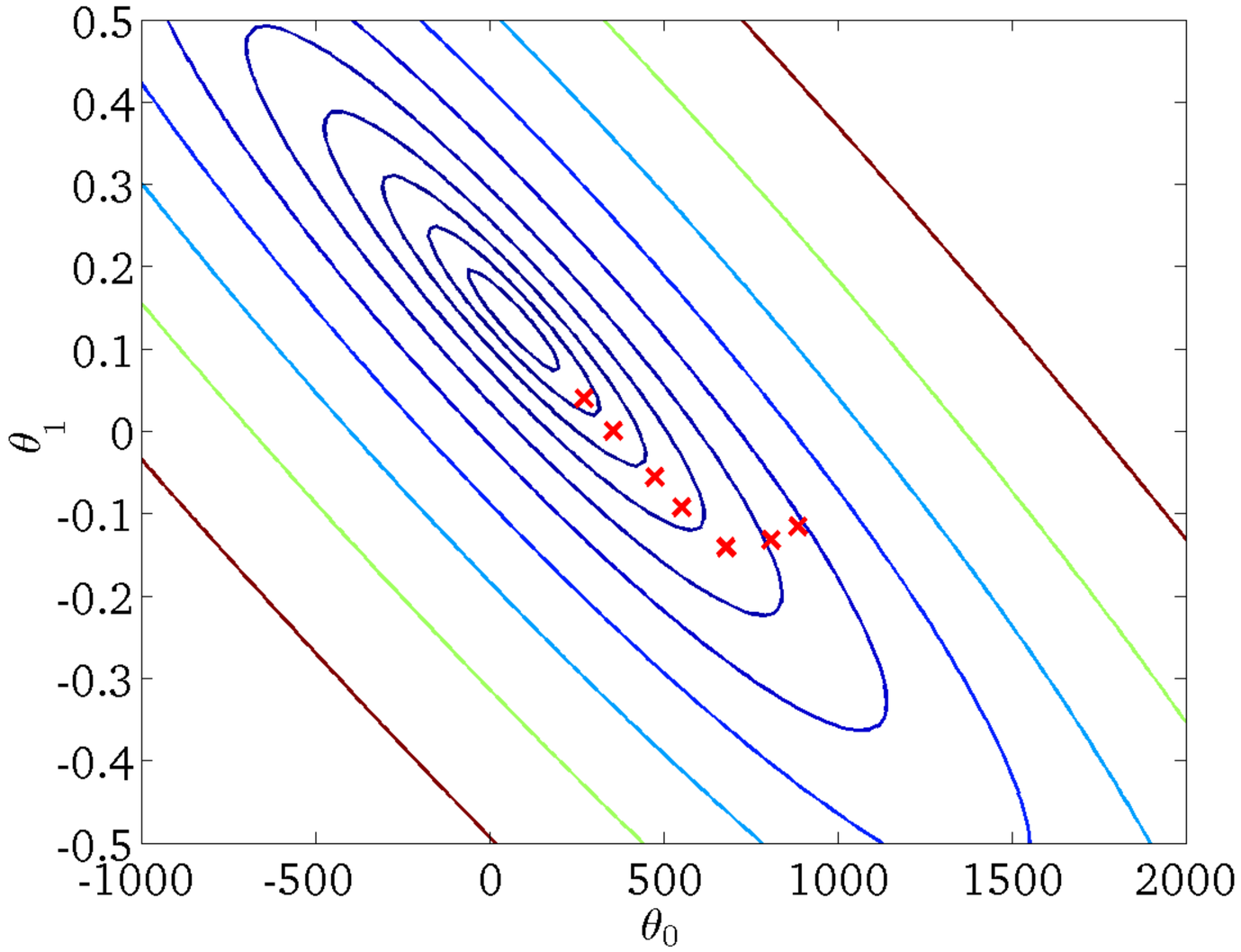
$$J(\theta_0, \theta_1)$$

梯度下降



$$J(\theta_0, \theta_1)$$

梯度下降

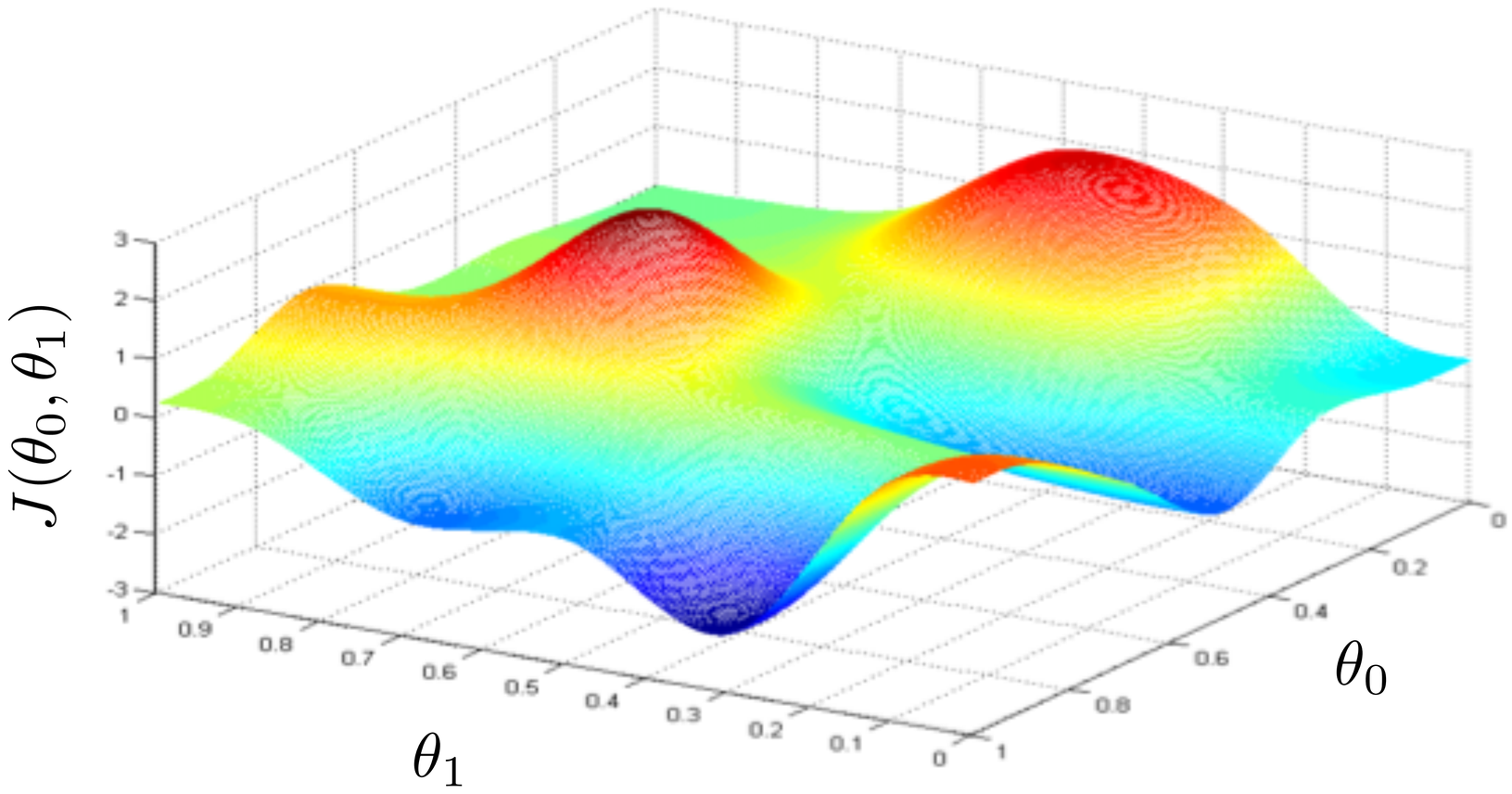


$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

$$J(\theta_0, \theta_1)$$

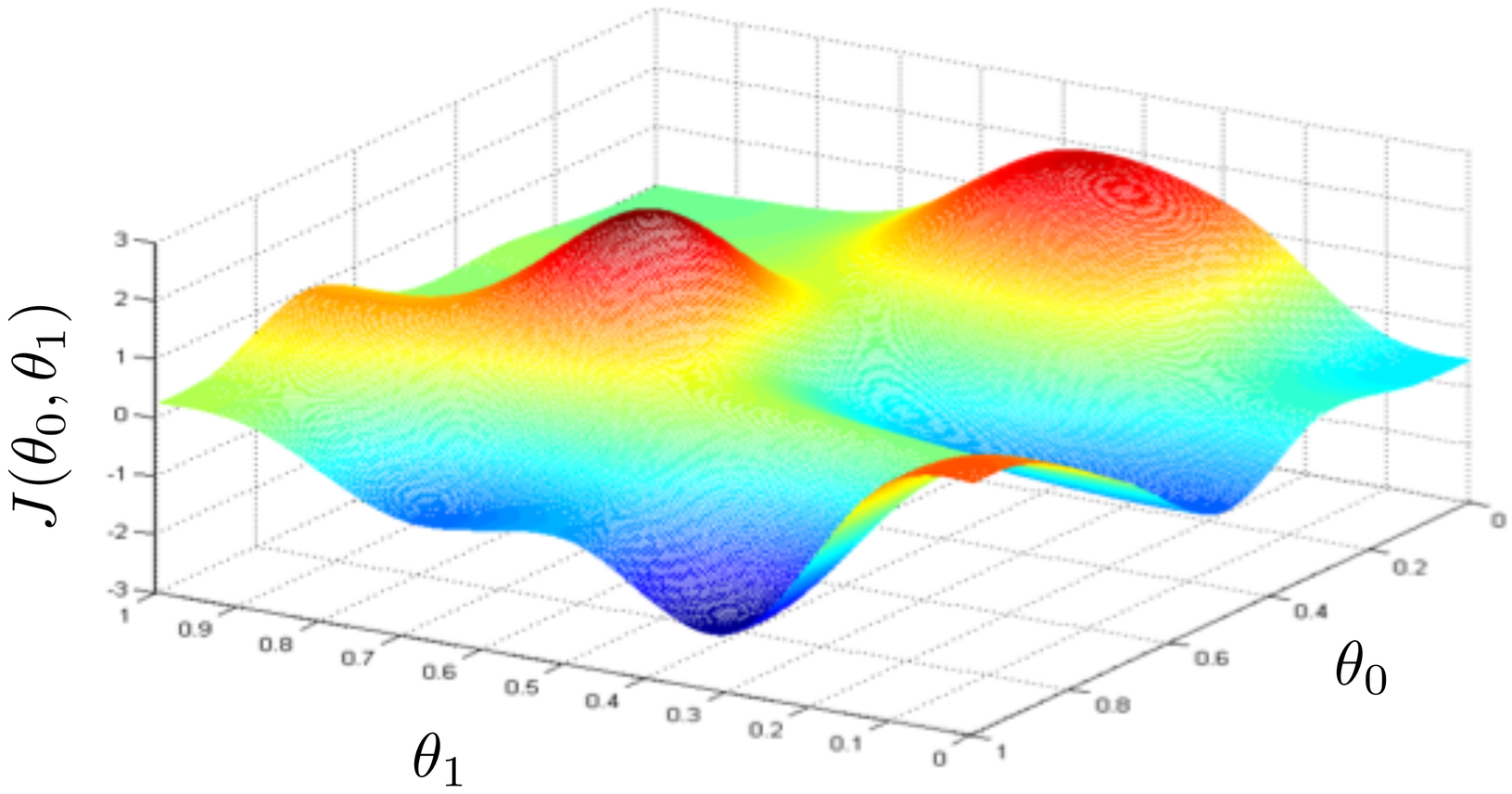
梯度下降

$$arg\,min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$



梯度下降

$$minimize_{\theta} J(\theta_1, \theta_2)$$



梯度下降

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\quad \text{(for } j = 1 \text{ and } j = 0) \\
&\}
\end{aligned}
$$

梯度下降

$minimize_\theta J(\theta_1, \theta_2)$

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

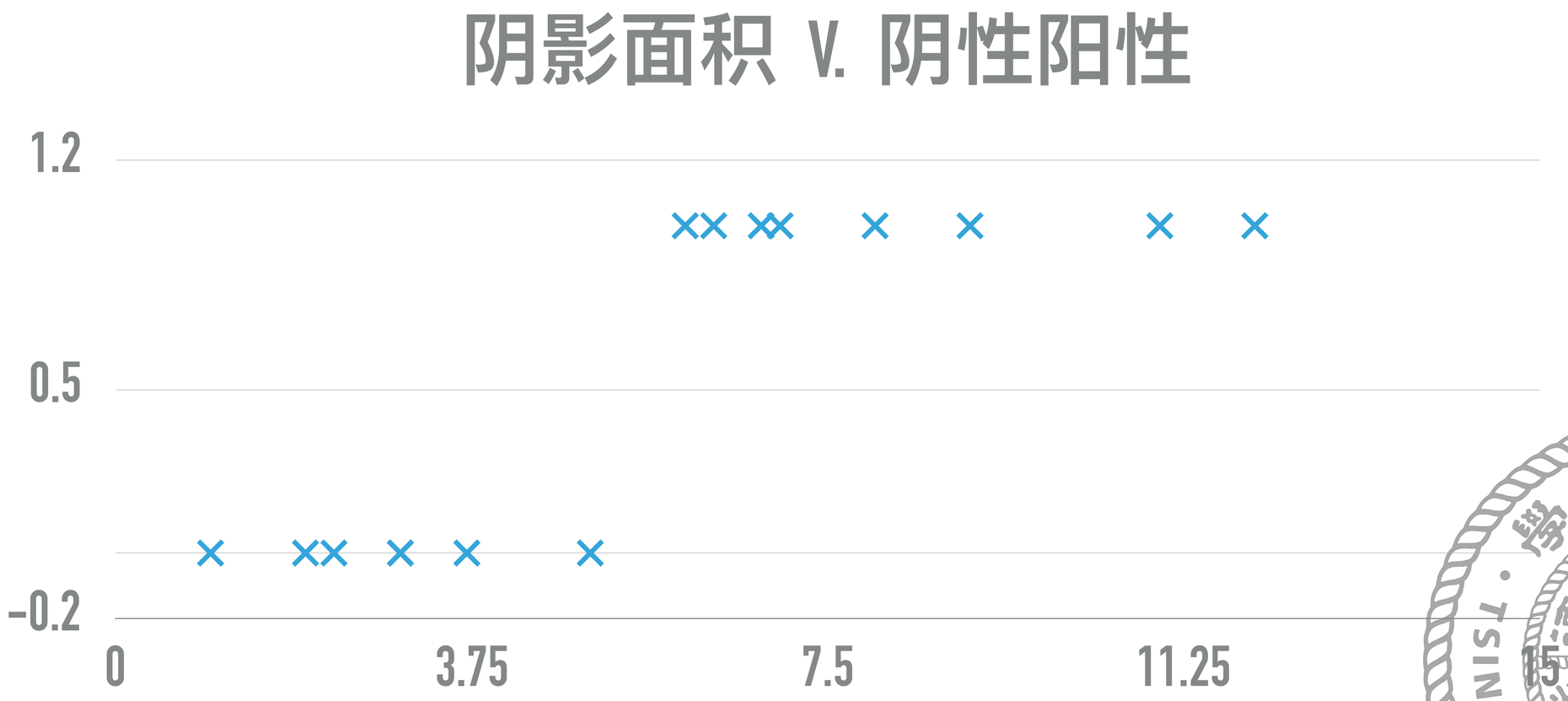
}

CLASSIFICATION

分类问题

TOY-PROBLEM

TOY-PROBLEM

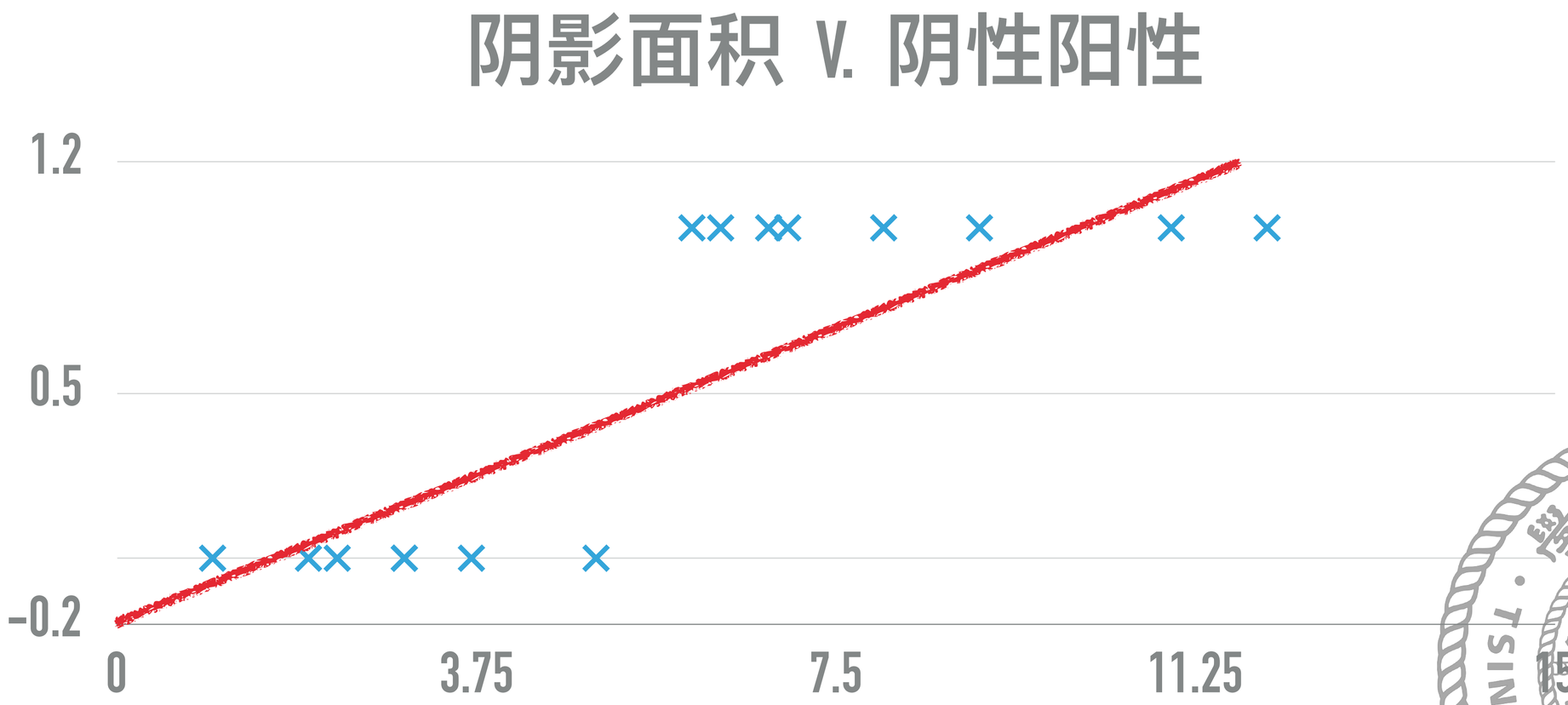


TOY-PROBLEM

$$f(x) = \theta_0 * x + \theta_1$$

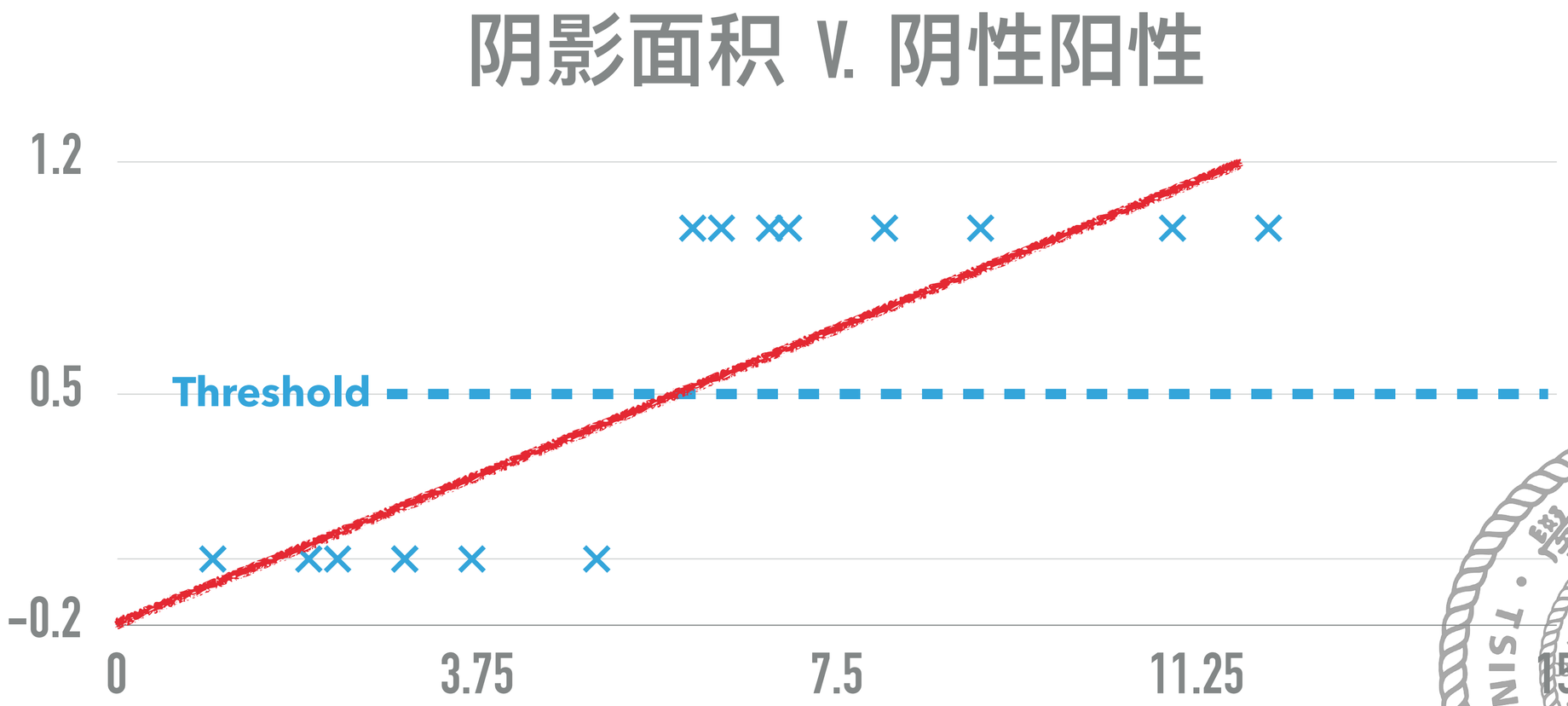
TOY-PROBLEM

$$f(x) = \theta_0 * x + \theta_1$$



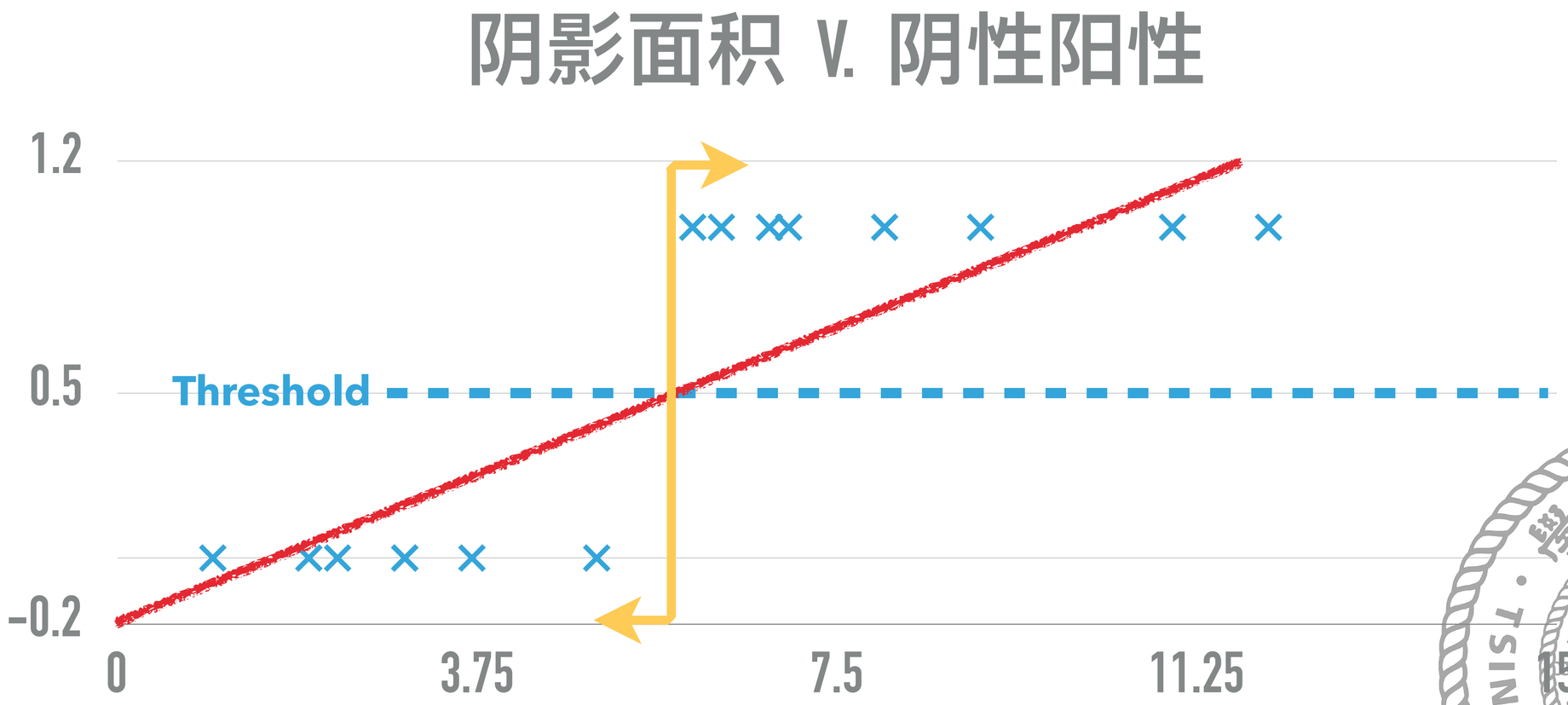
TOY-PROBLEM

$$f(x) = \theta_0 * x + \theta_1$$



TOY-PROBLEM

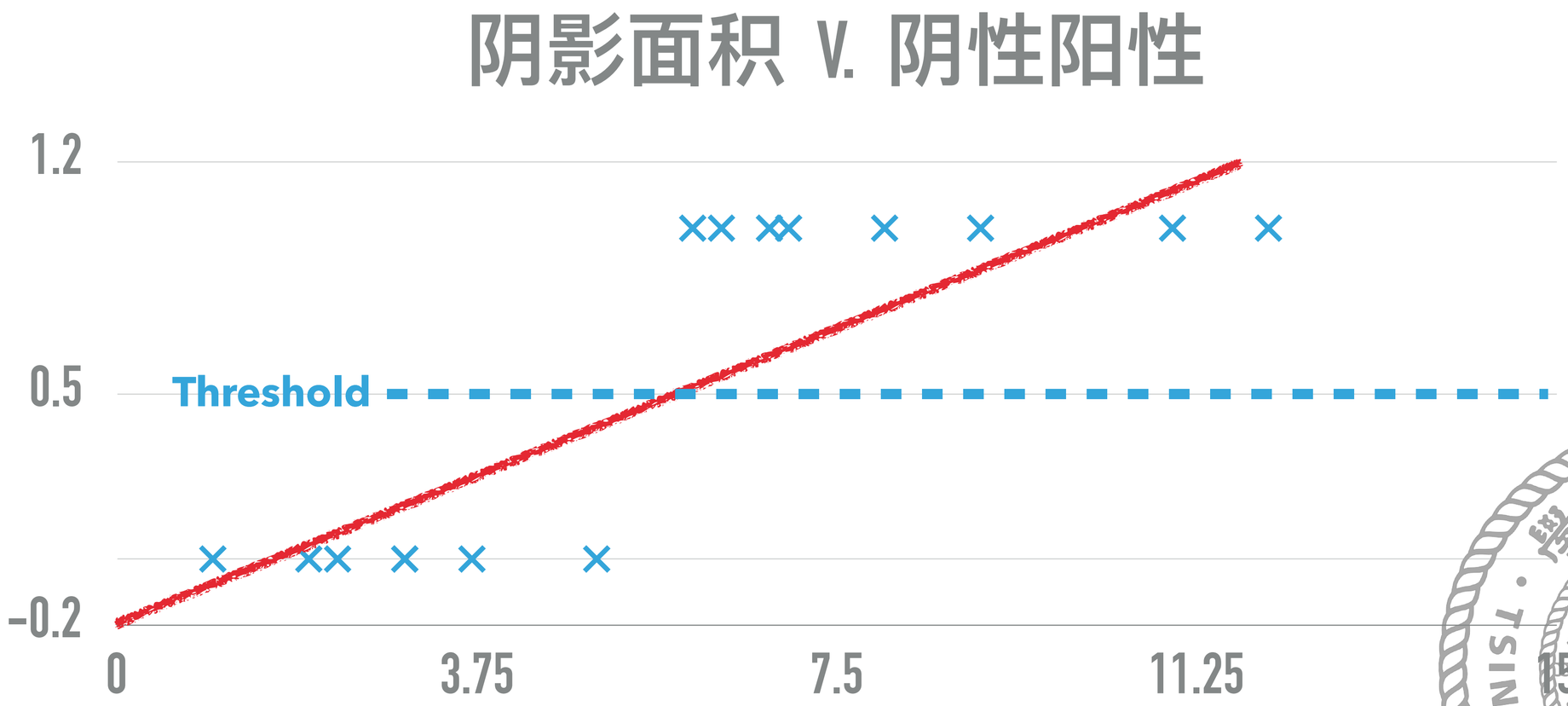
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

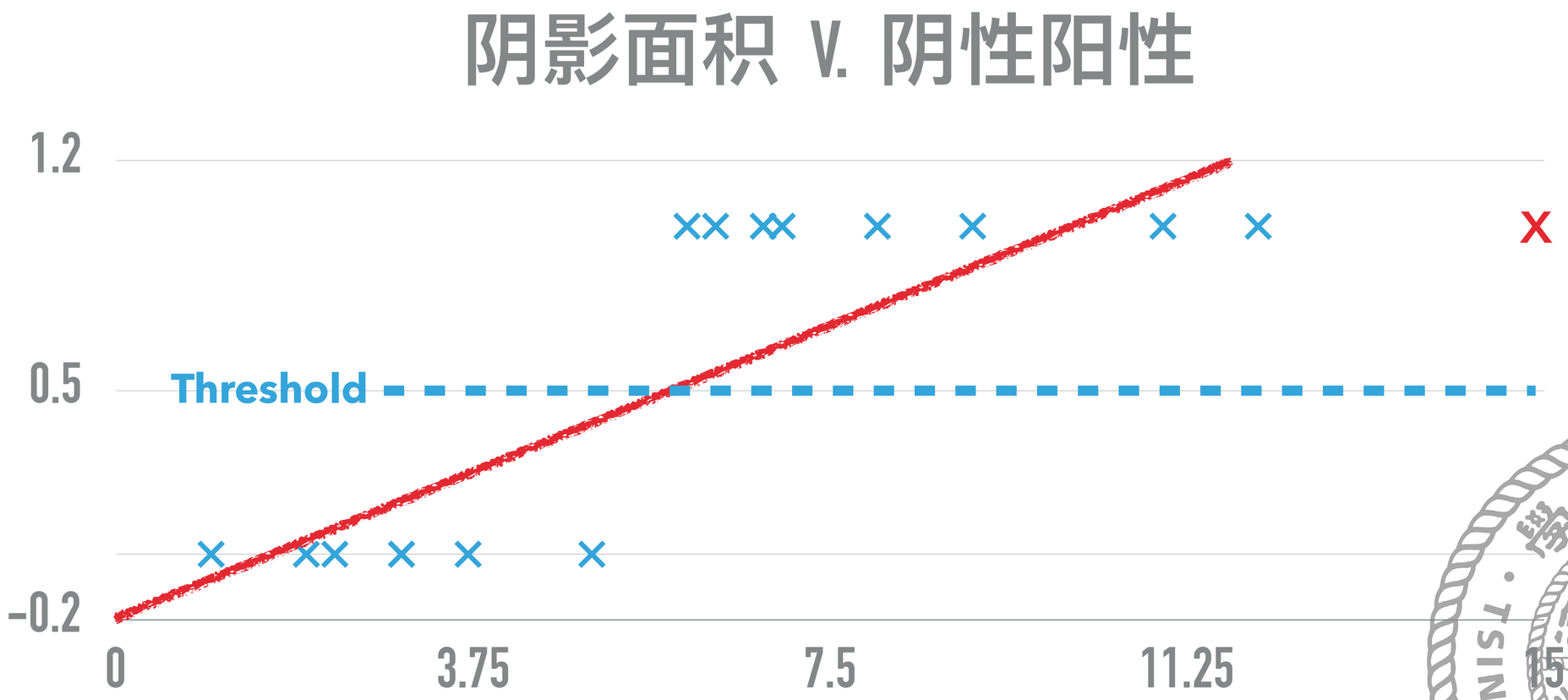
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

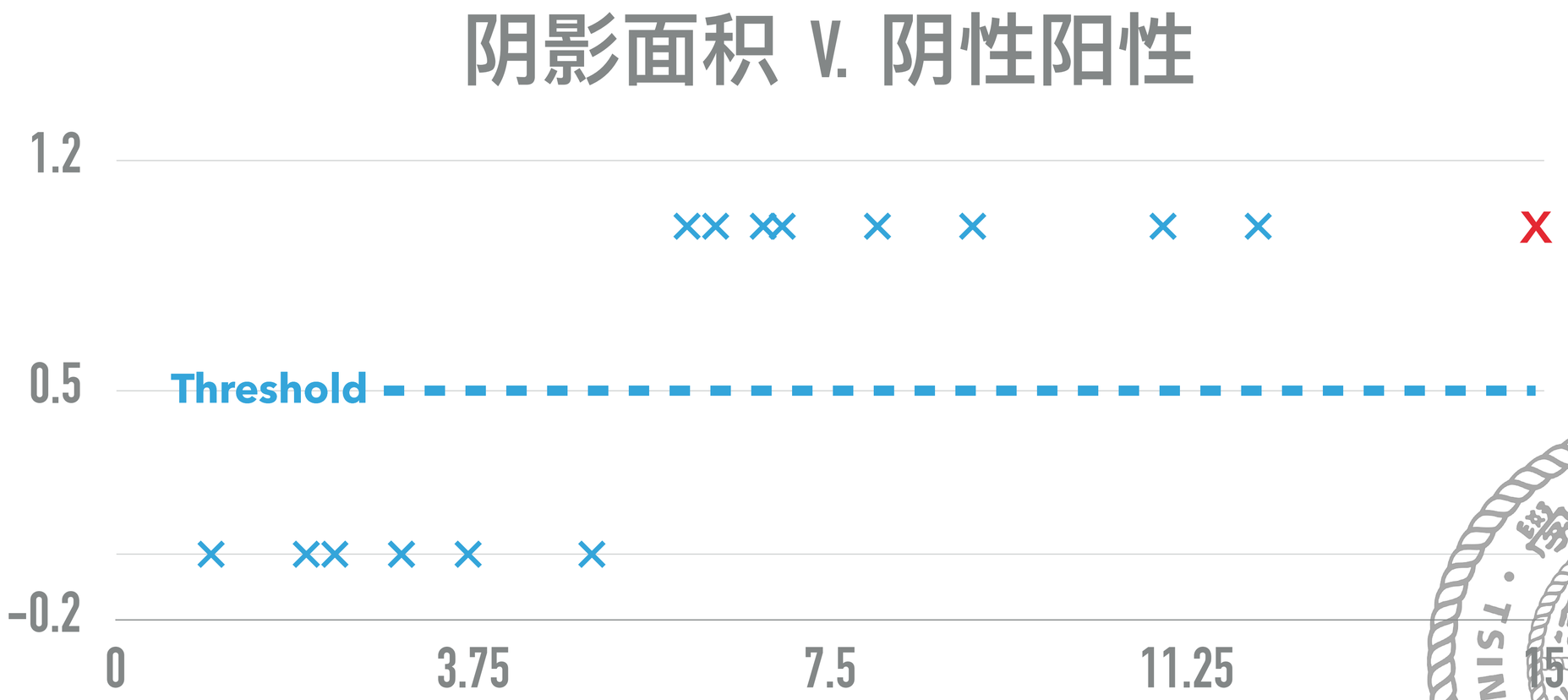
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

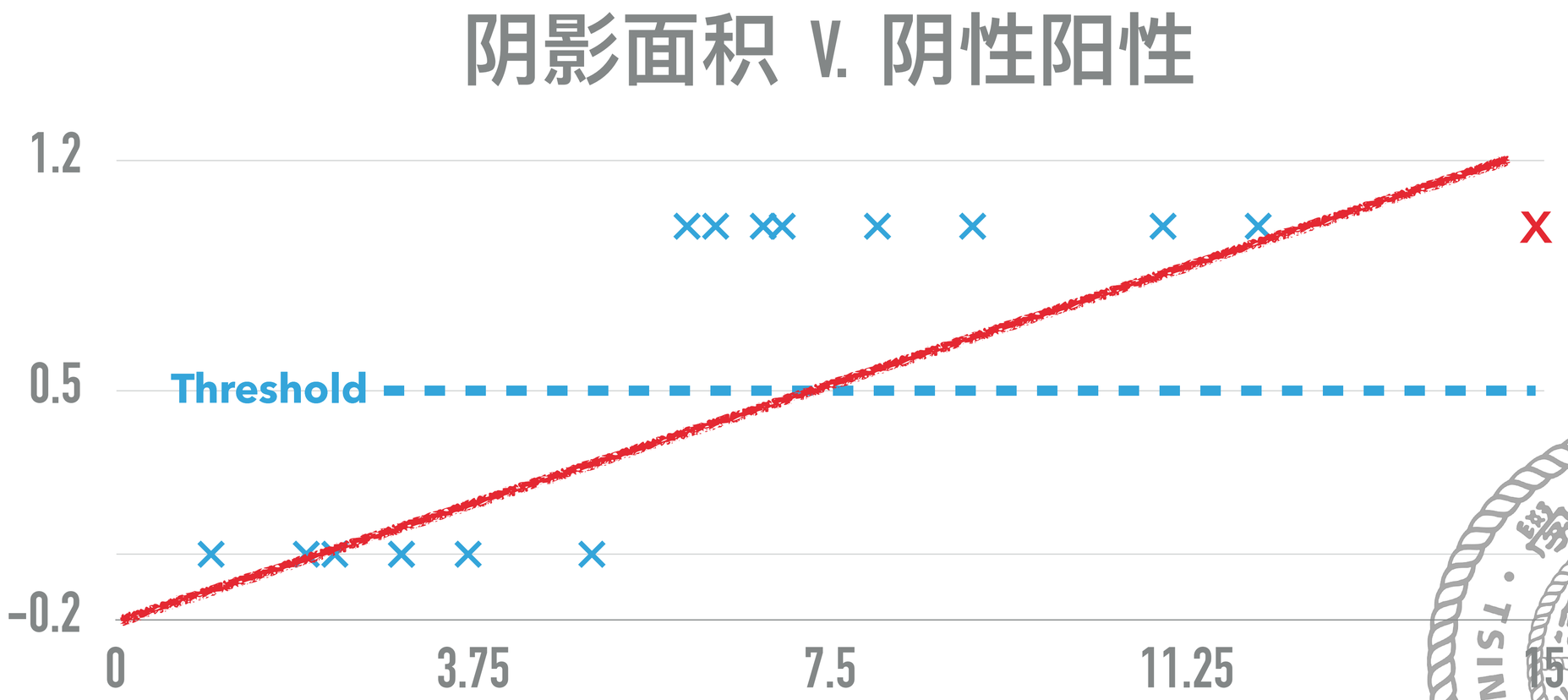
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

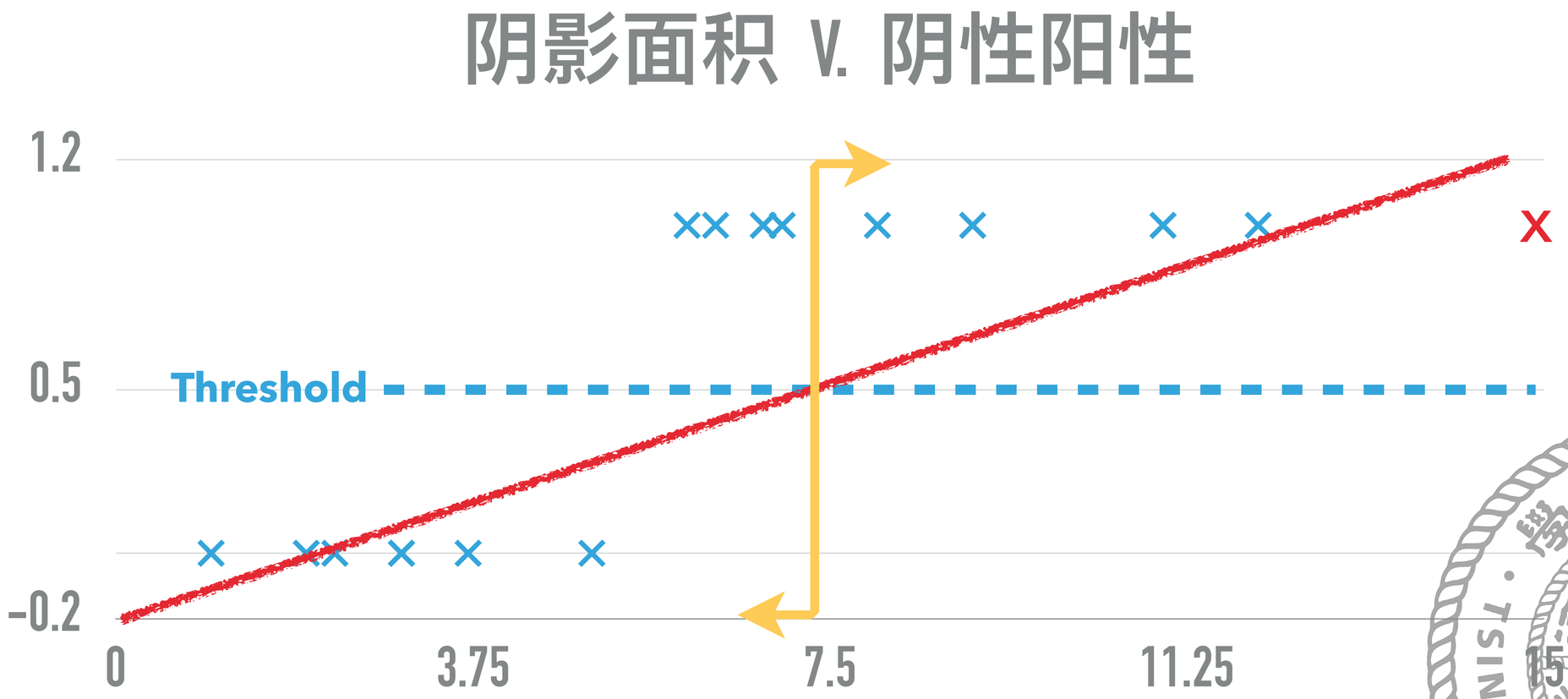
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

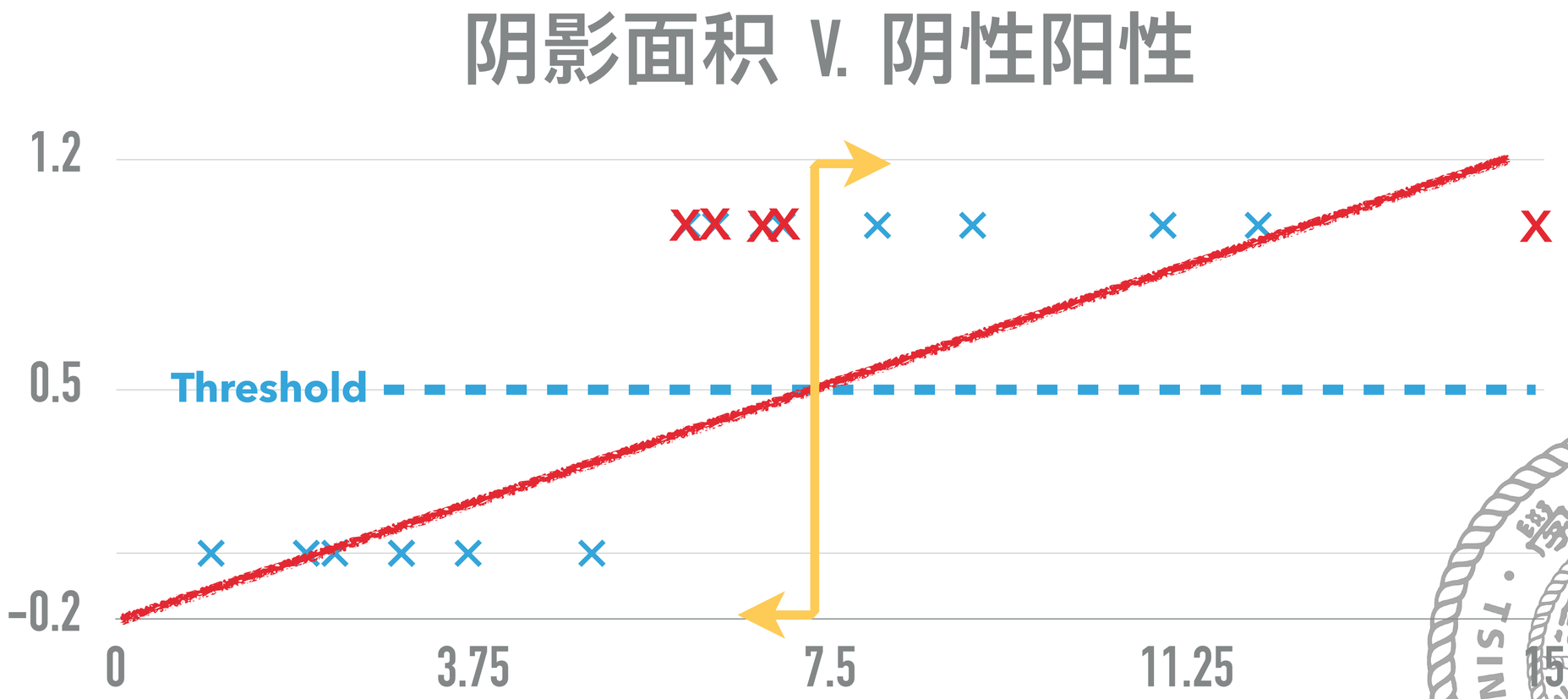
$$f(x) = \theta_0 * x + \theta_1$$



CLASSIFICATION

TOY PROBLEM

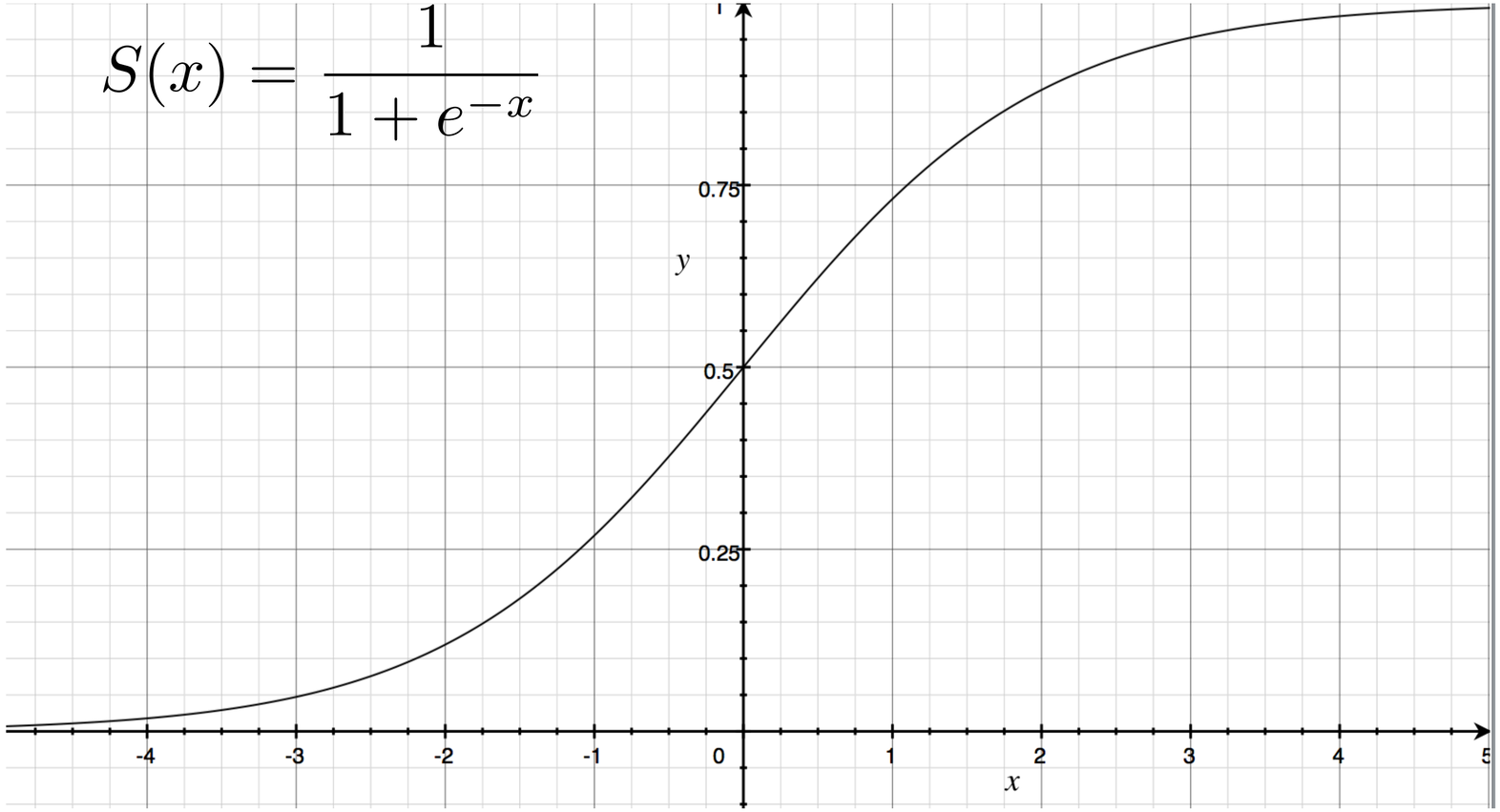
$$f(x) = \theta_0 * x + \theta_1$$



分类问题

SIGMOID 函数

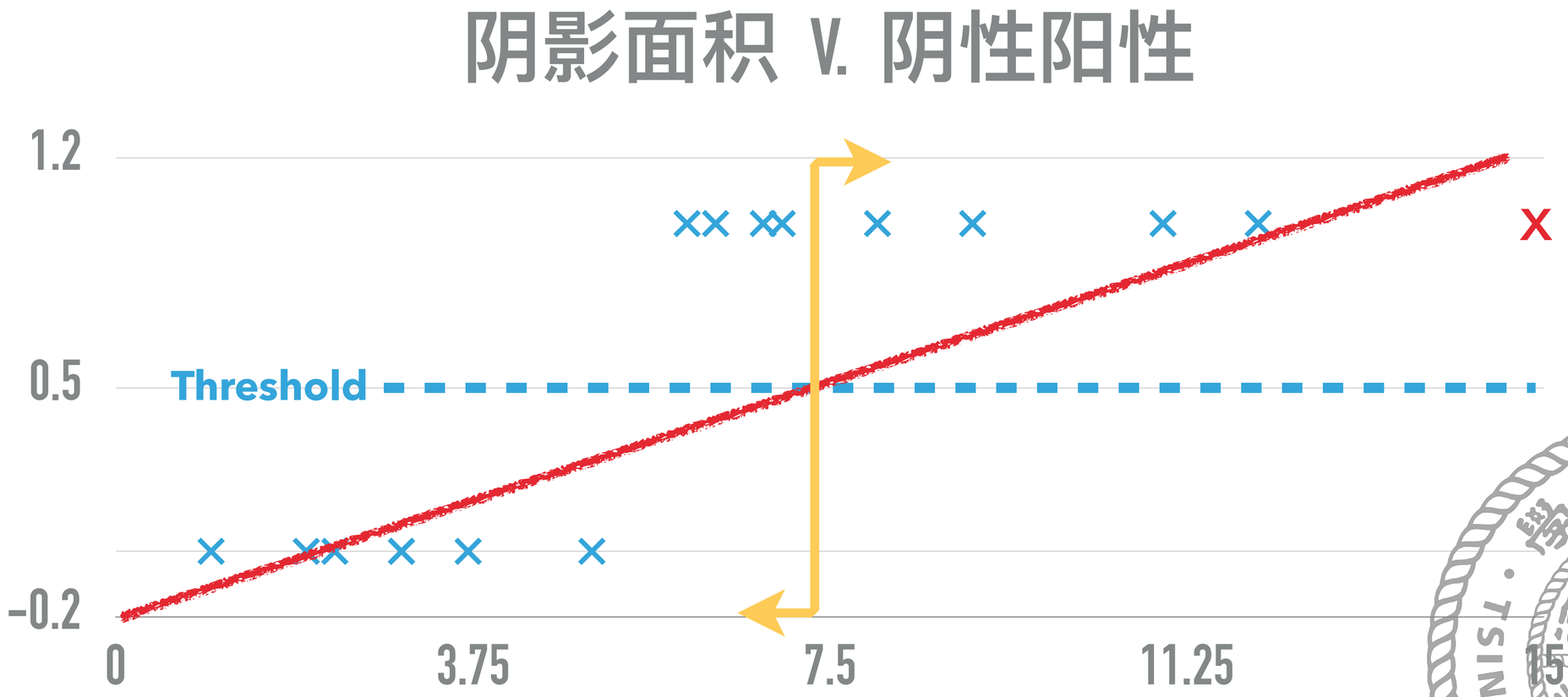
$$S(x) = \frac{1}{1 + e^{-x}}$$



CLASSIFICATION

LOGISTIC 回归

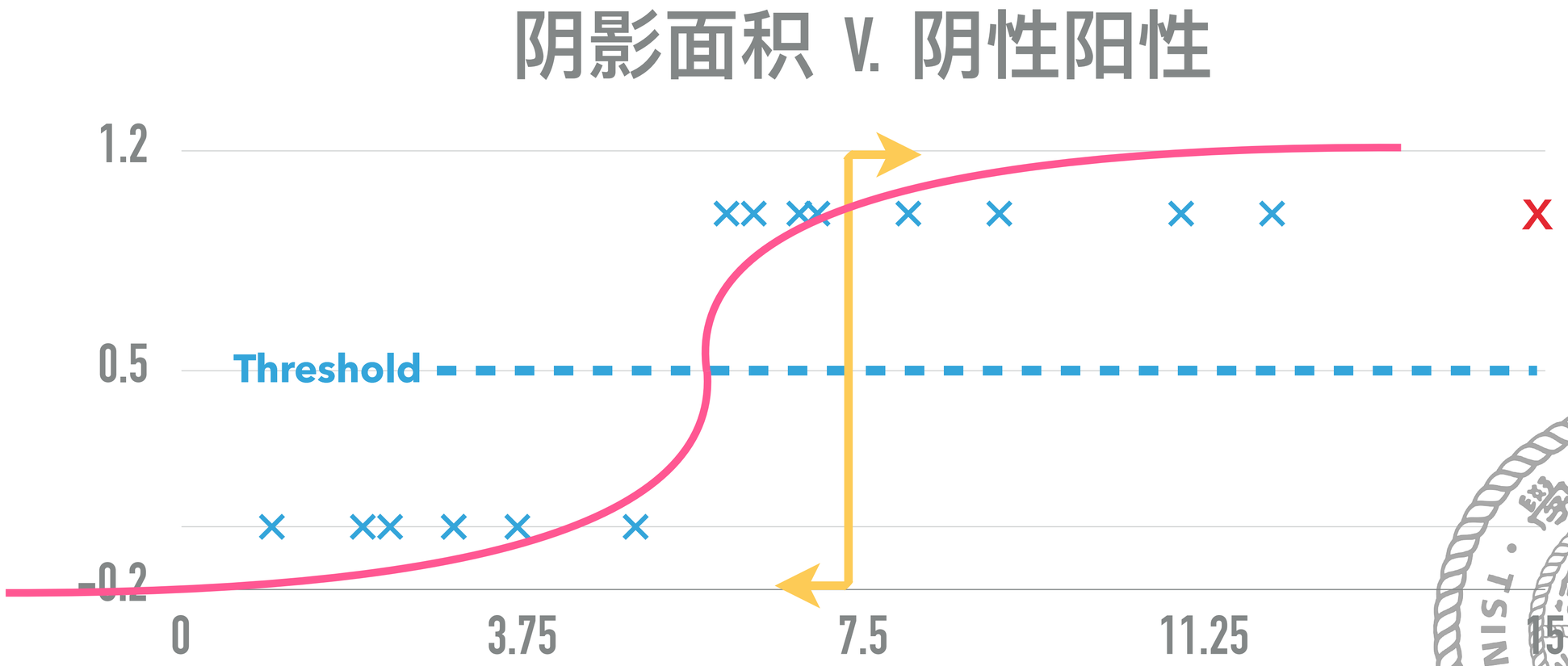
$$S(x) = \frac{1}{1 + e^{-x}}$$



CLASSIFICATION

LOGISTIC 回归

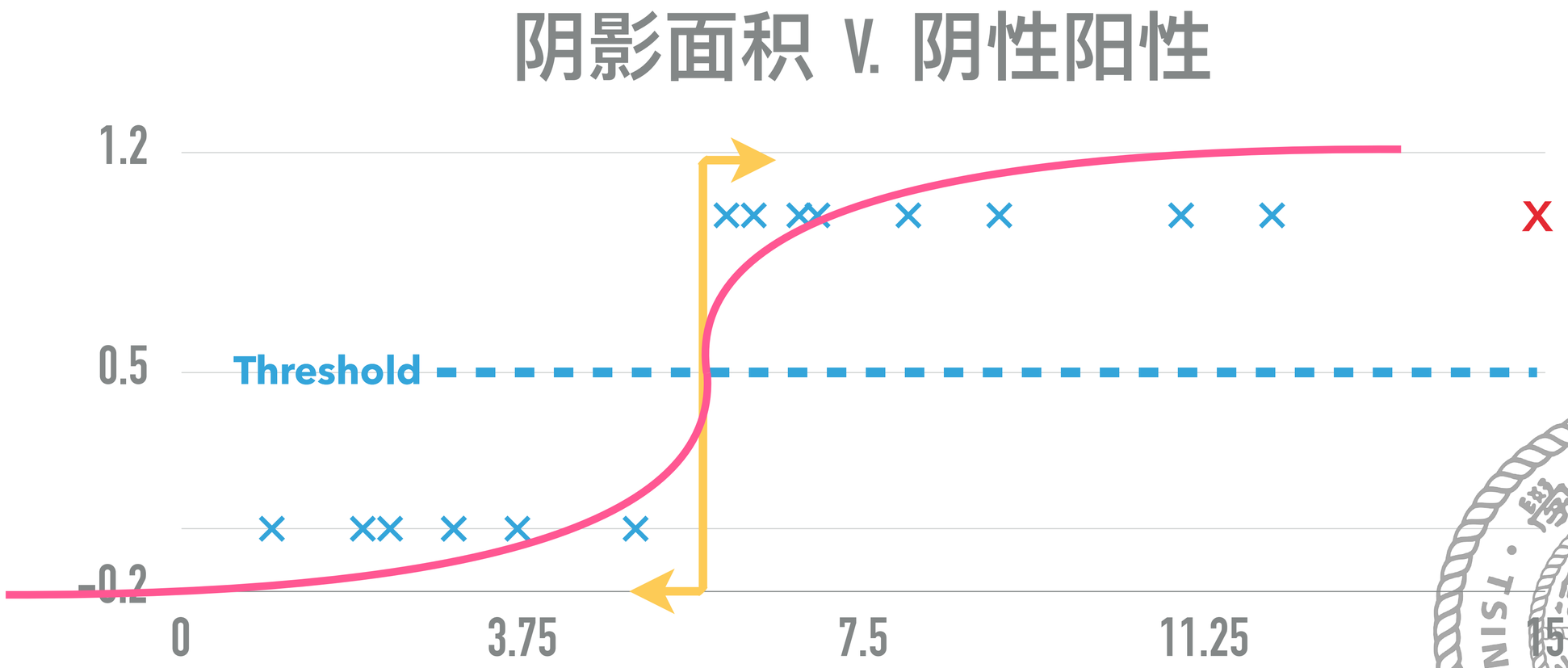
$$S(x) = \frac{1}{1 + e^{-x}}$$



CLASSIFICATION

LOGISTIC 回归

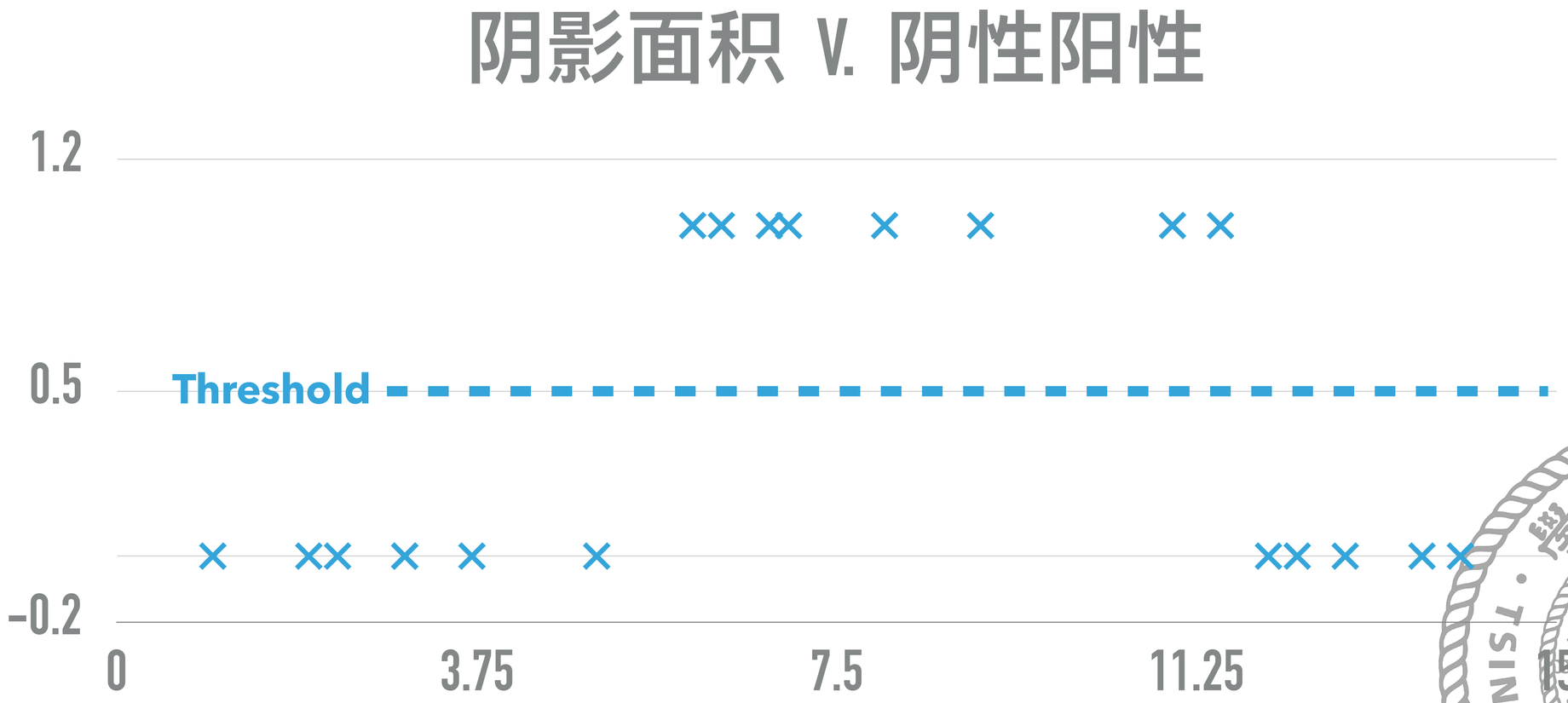
$$S(x) = \frac{1}{1 + e^{-x}}$$



CLASSIFICATION

LOGISTIC 回归

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

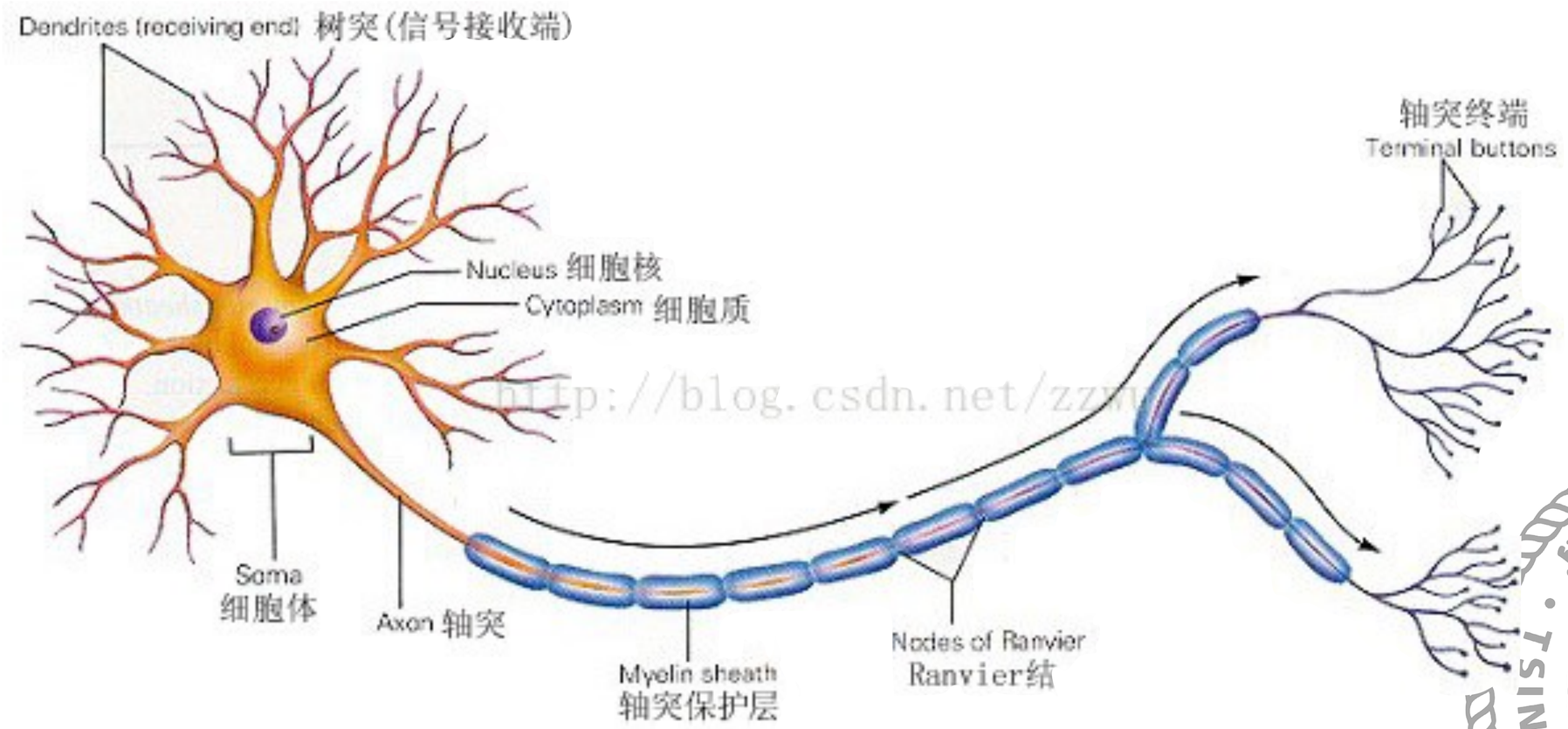


MULTI-LAYER PERCEPTRON

多层感知器网络

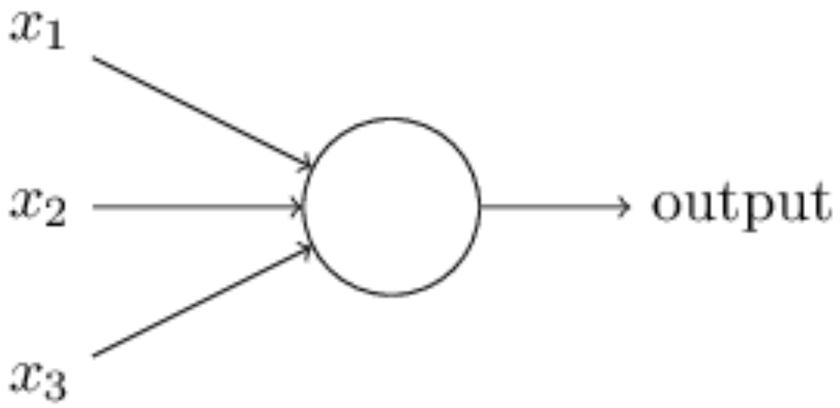
MULTI-LAYER PERCEPTRON

感知器



MULTI-LAYER PERCEPTRON

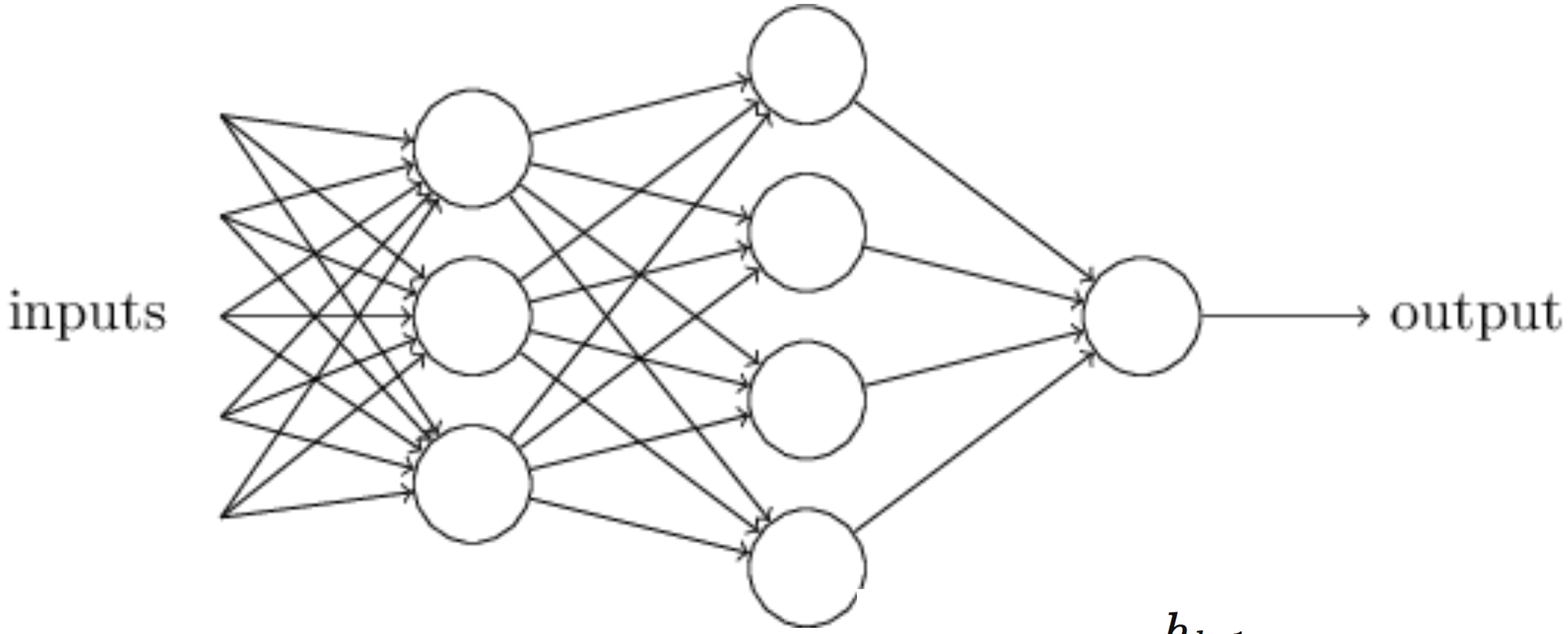
感知器



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

MULTI-LAYER PERCEPTRON

多层神经网络

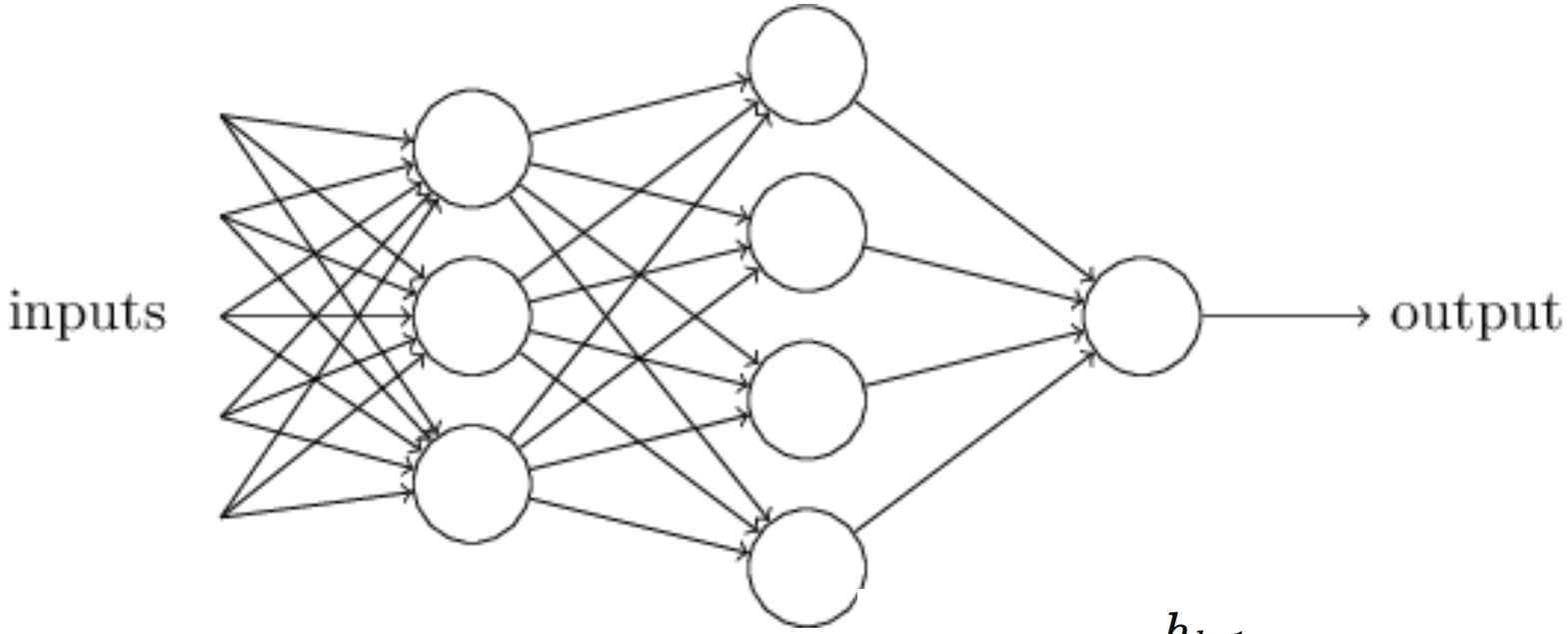


$$a_h = \sum_{h'=1}^{h_{l-1}} w_{h'h} b_{h'}$$

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

MULTI-LAYER PERCEPTRON

多层神经网络



$$a_h = \sum_{h'=1}^{h_{l-1}} w_{h'h} b_{h'}$$

$$b_h = f(a_h)$$

MULTI-LAYER PERCEPTRON

反向传播算法

MULTI-LAYER PERCEPTRON

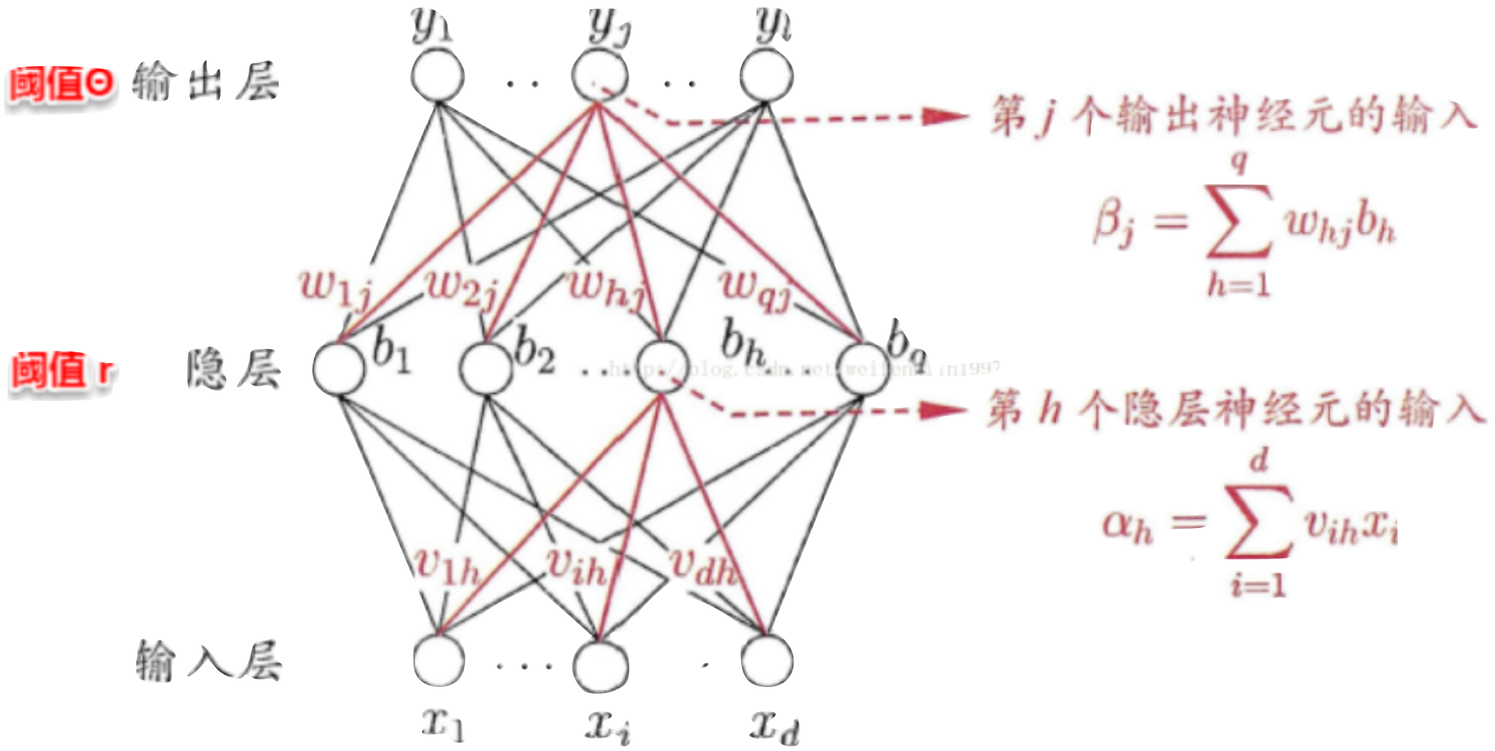
反向传播算法

MULTI-LAYER PERCEPTRON

BP(back propagation)神经网络一种按照误差逆向传播算法训练的多层前馈神经网络，是目前应用最广泛的神经网络

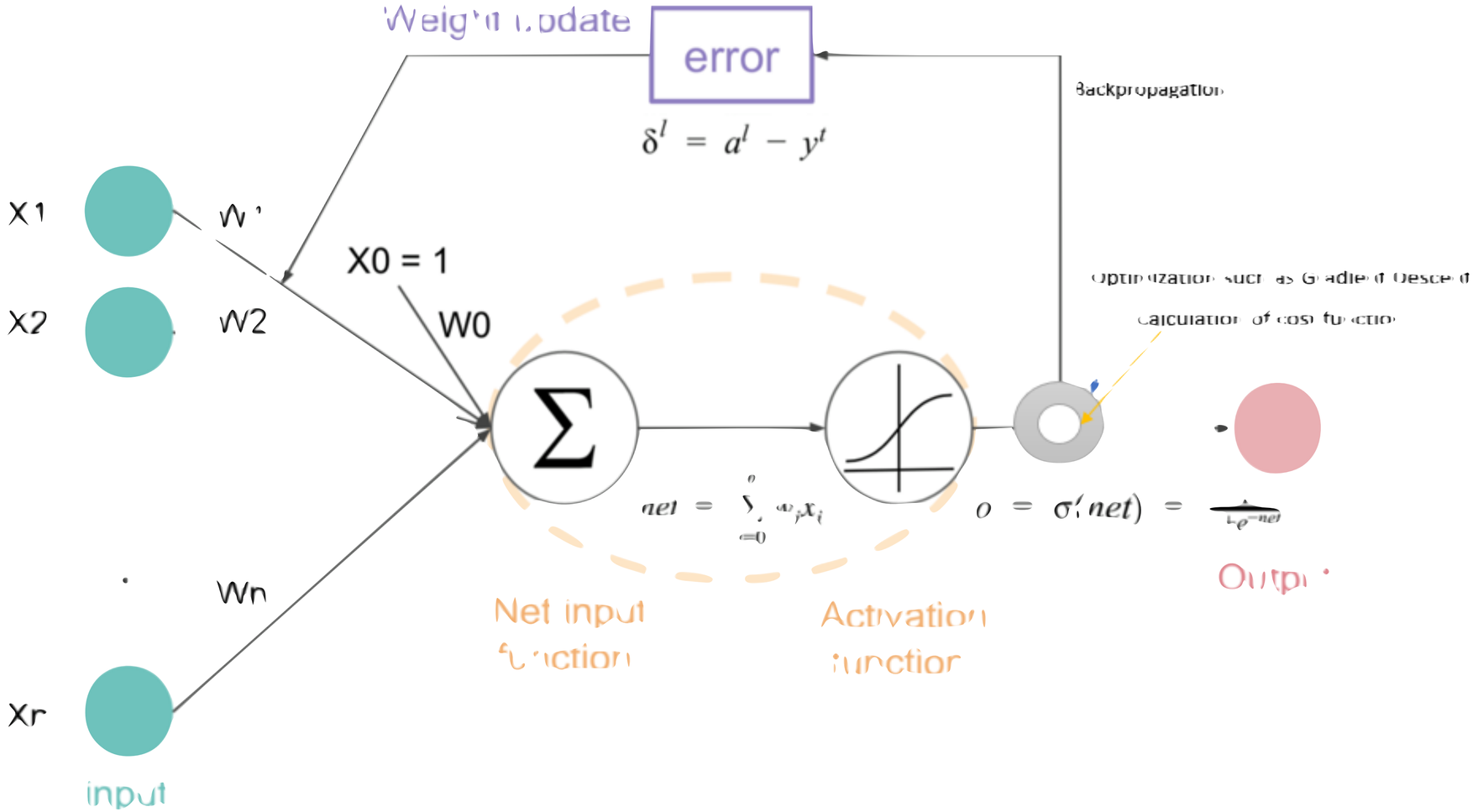
MULTI-LAYER PERCEPTRON

反向传播算法



MULTI-LAYER PERCEPTRON

反向传播算法



1 分钟神经网络实战

Tensorflow Playground

MULTI-LAYER PERCEPTRON

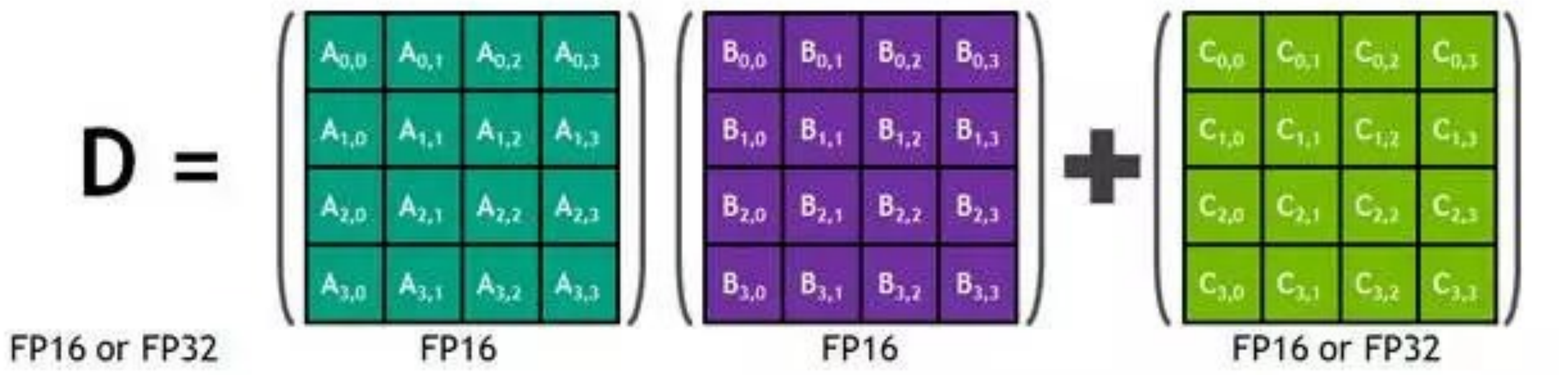
VECTORIZATION

VECTORIZATION

矩阵加速运算

	1024x1024	2048x2048	4096x4096
CUDA C (ms)	43.11	391.05	3407.99
C++ (ms)	6137.10	64369.29	551390.93
C# (ms)	10509.00	300684.00	2527250.00
Java (ms)	9149.90	92562.28	838357.94
MATLAB (ms)	75.01	423.10	3133.90

VECTORIZATION



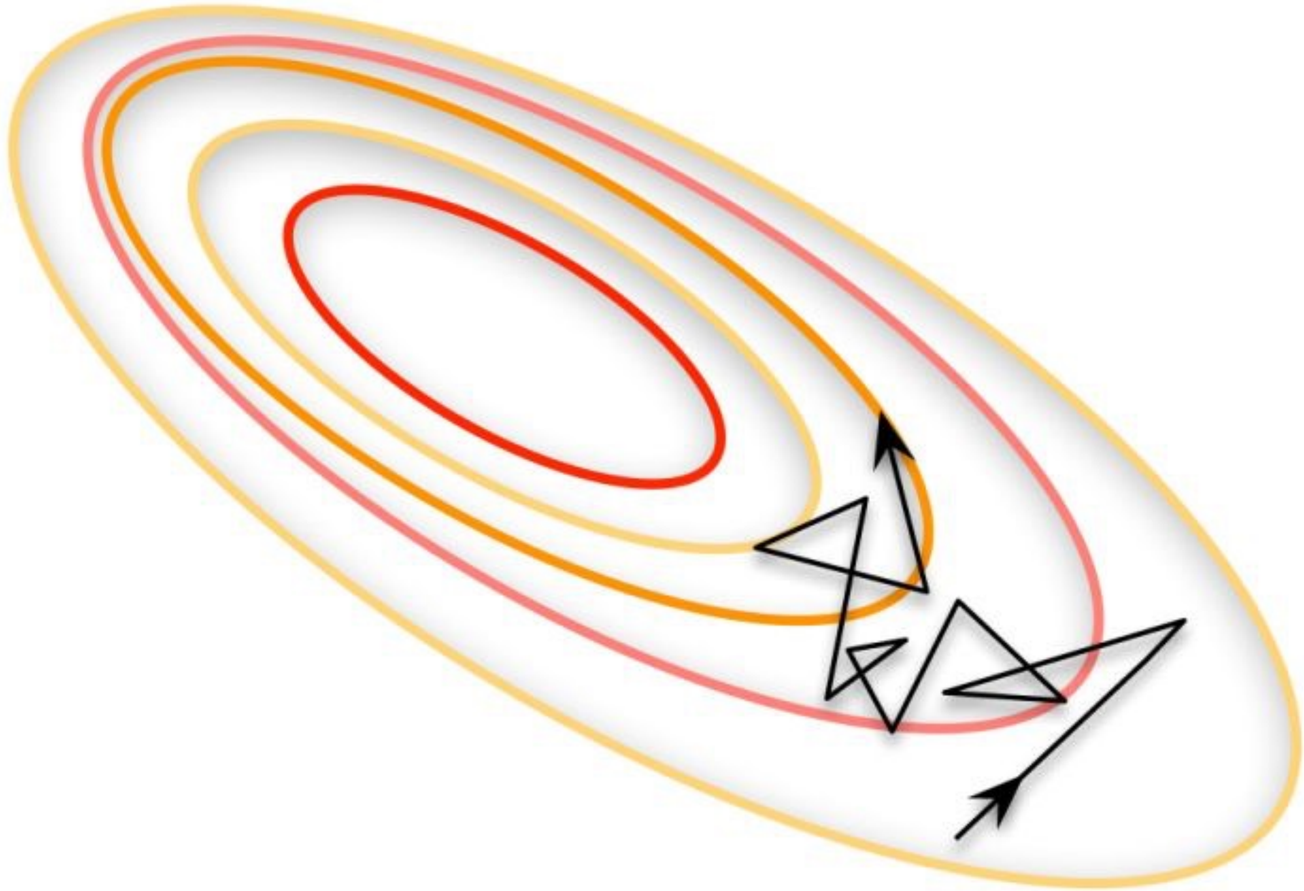
每个 Tensor Core 每个时钟可执行 64 次浮点 FMA 混合精度运算（FP16 乘法与 FP32 累加），一个 SM 单元中的 8 个 Tensor Core 每个时钟可执行共计 1024 次浮点运算。相比于使用标准 FP32 计算的 Pascal GP100 而言，单个 SM 下的每个深度学习应用的吞吐量提升了 8 倍，所以这最终使得 Volta V100 GPU 相比于 Pascal P100 GPU 的吞吐量一共提升了 12 倍。

MULTI-LAYER PERCEPTRON

BATCH TRAINING

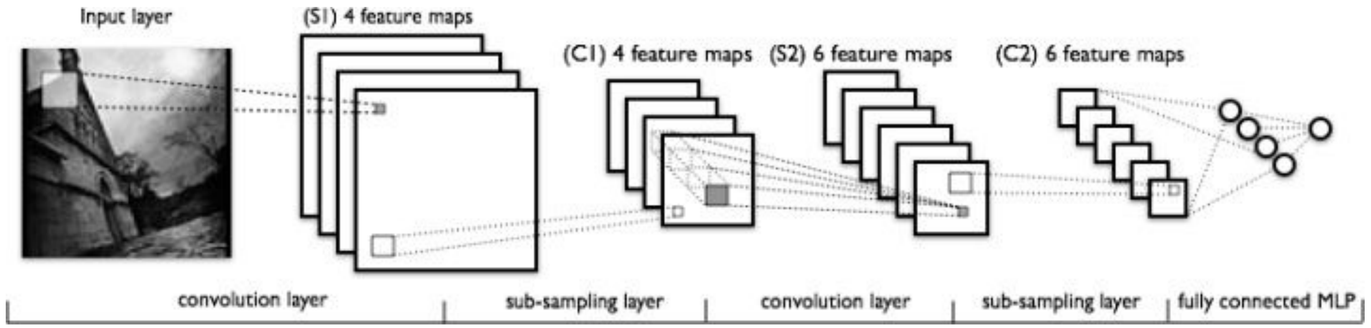
BATCH TRAINING

为什么要进行BATCH TRAINING



BATCH TRAINING

如何确定BATCH SIZE



Batch_Size	5000	2000	1000	500	256	100	50	20	10	5	2	1
Total Epoches	200	200	200	200	200	200	200	200	200	200	200	200
Total Iterations	1999	4999	9999	19999	38999	99999	199999	499999	999999	1999999	cannot converge	
Time of 200 Epoches	1	1.068	1.16	1.38	1.75	3.016	5.027	8.513	13.773	24.055		
Achieve 0.99 Accuracy at Epoch	-	-	135	78	41	45	24	9	9	-		
Time of Achieve 0.99 Accuracy	-	-	2.12	1.48	1	1.874	1.7	1.082	1.729	-		
Best Validation Score	0.015	0.011	0.01	0.01	0.01	0.009	0.0098	0.0084	0.01	0.032		
Best Score Achieved at Epoch	182	170	198	100	93	111	38	49	51	17		
Best Test Score	0.014	0.01	0.01	0.01	0.01	0.008	0.0083	0.0088	0.008	0.0262		
Final Test Error (200 epoches)	0.0134	0.01	0.01	0.01	0.01	0.009	0.0082	0.0088	0.008	0.0662		

神经网络遇到的问题

PROBLEMS

PROBLEMS

遇到的问题

PROBLEMS

遇到的问题

- 无历史性（记不住）

PROBLEMS

遇到的问题

- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）

PROBLEMS

遇到的问题

- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）
- 过拟合（Overfitting）

PROBLEMS

遇到的问题

- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）
- 过拟合（Overfitting）
- 局部最优（非凸优化）

PROBLEMS

遇到的问题

- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）
- 过拟合（Overfitting）
- 局部最优（非凸优化）
- 需要太多的标记数据

PROBLEMS

遇到的问题

- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）
- 过拟合（Overfitting）
- 局部最优（非凸优化）
- 需要太多的标记数据
- 超参数“看脸”

PROBLEMS

遇到的问题

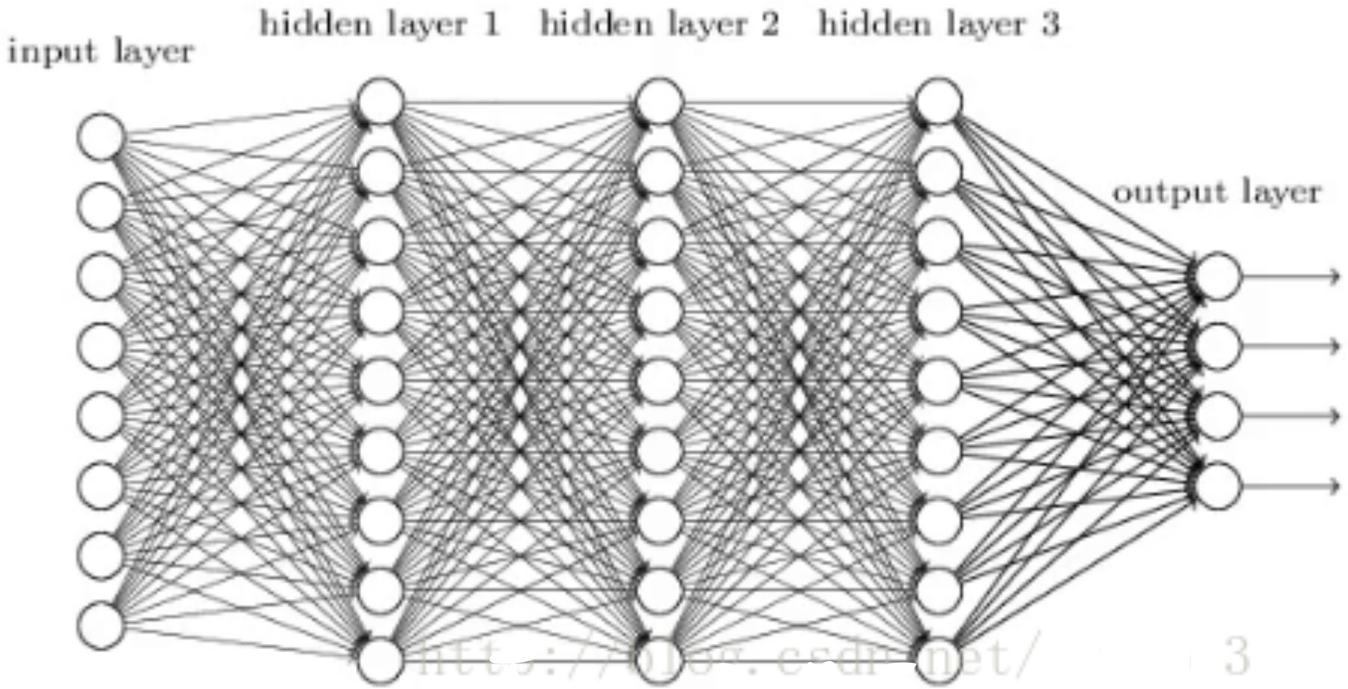
- 无历史性（记不住）
- 梯度消失、梯度爆炸 （深度上不去）
- 过拟合（Overfitting）
- 局部最优（非凸优化）
- 需要太多的标记数据
- 超参数“看脸”
- etc.

GRADIENT VANISH

梯度消失·爆炸

权值消失

GRADIENT VANISH

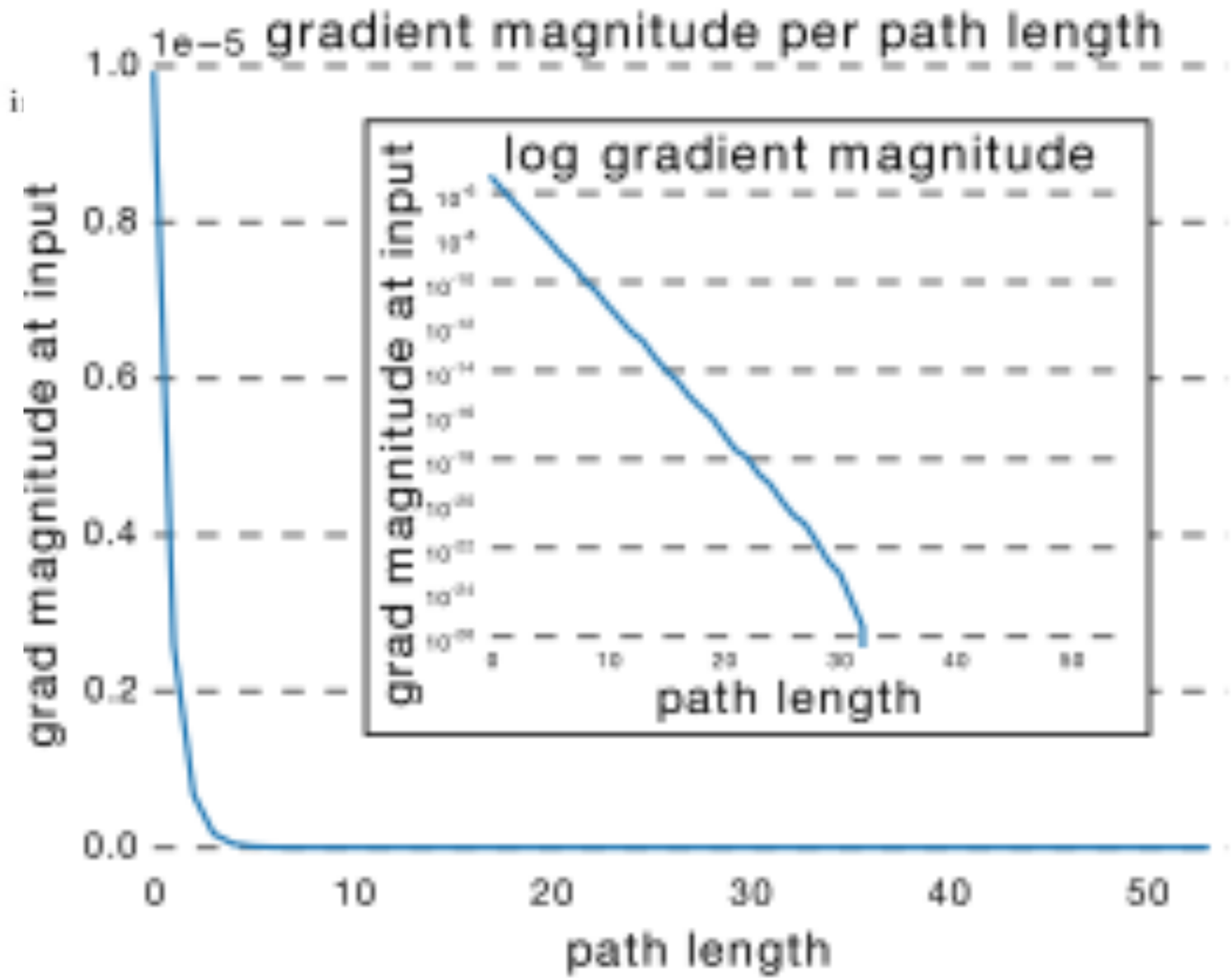


$$\Delta w_1 = \frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial f_4}\frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial w_2}$$

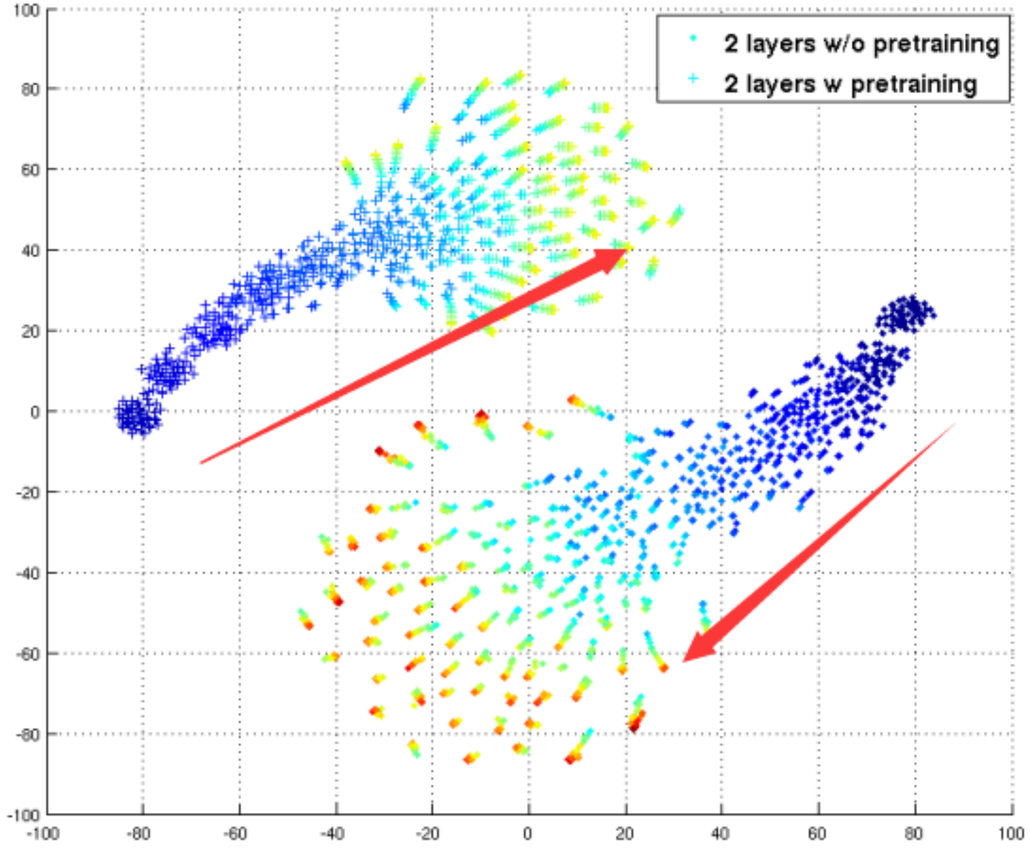
$$S'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = S(x)(1-S(x))$$

权值消失

GRADIENT VANISH

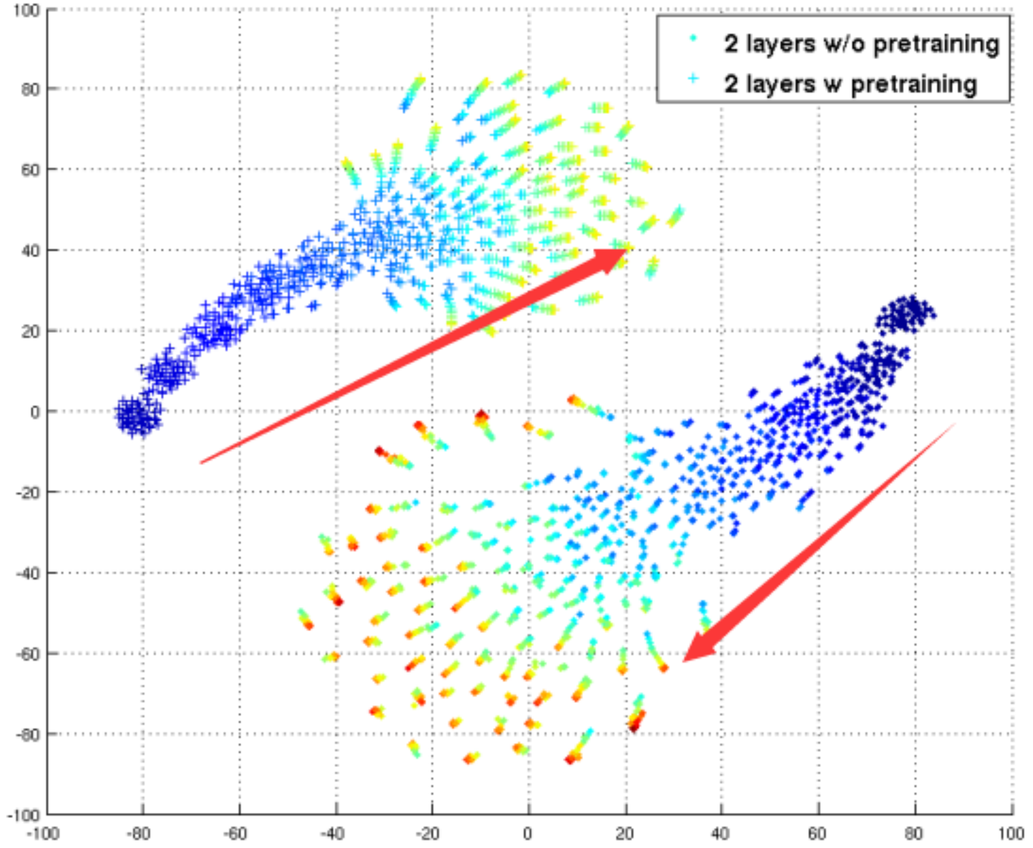


权值消失

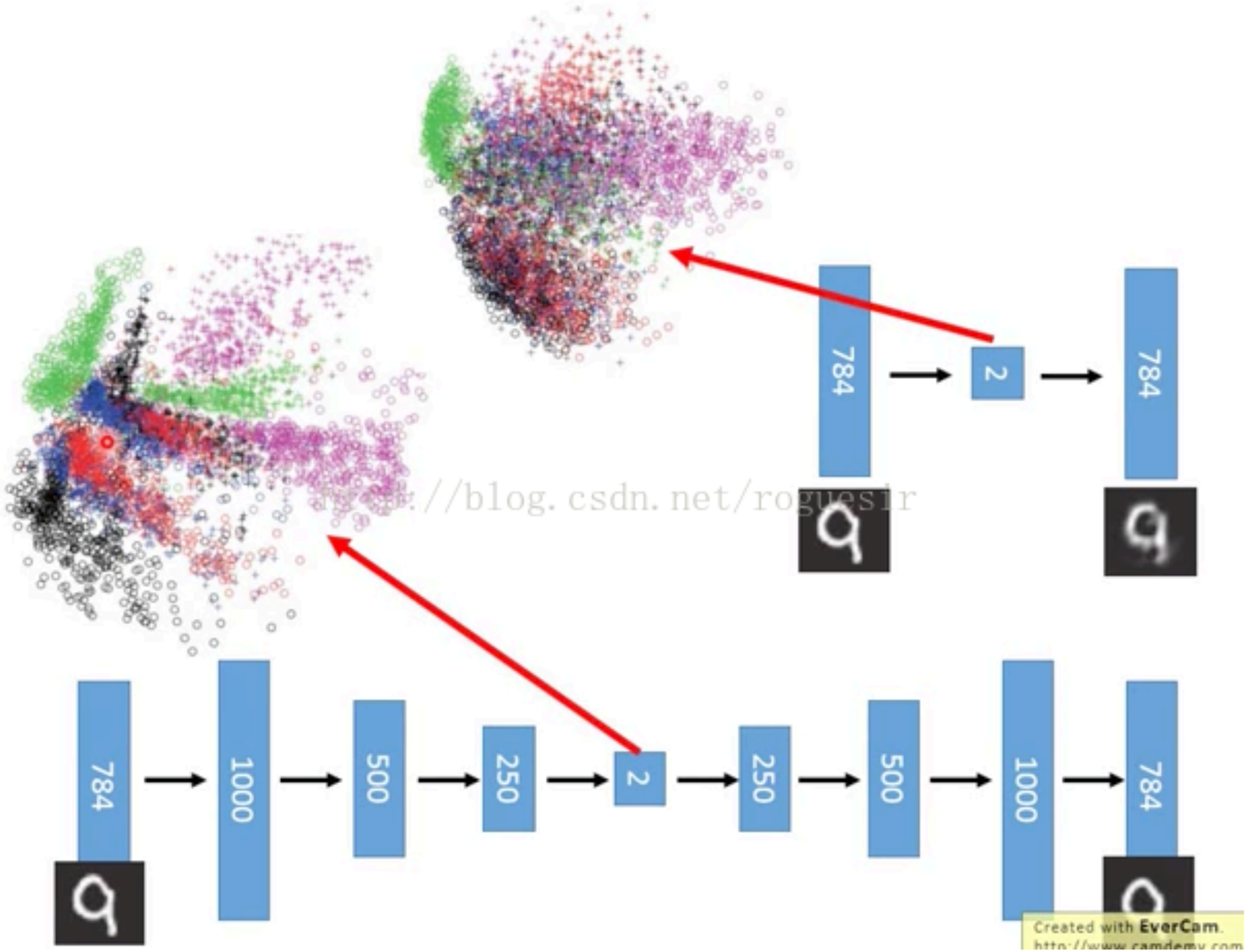

2 layers w/o pretraining
2 layers w pretraining
100
80
60
40
20
0
-20
-40
-60
-80
-100
-100
-80
-60
-40
-20
0
20
40
60
80
100

权值消失

2006年的时候Hilton使用受限玻尔兹曼自动机对模型进行预训练，利用无监督逐层训练方法，其基本思想是每次训练一层隐节点，训练时将上一层隐节点的输出作为输入，而本层隐节点的输出作为下一层隐节点的输入，将神经网络加深到了7层。

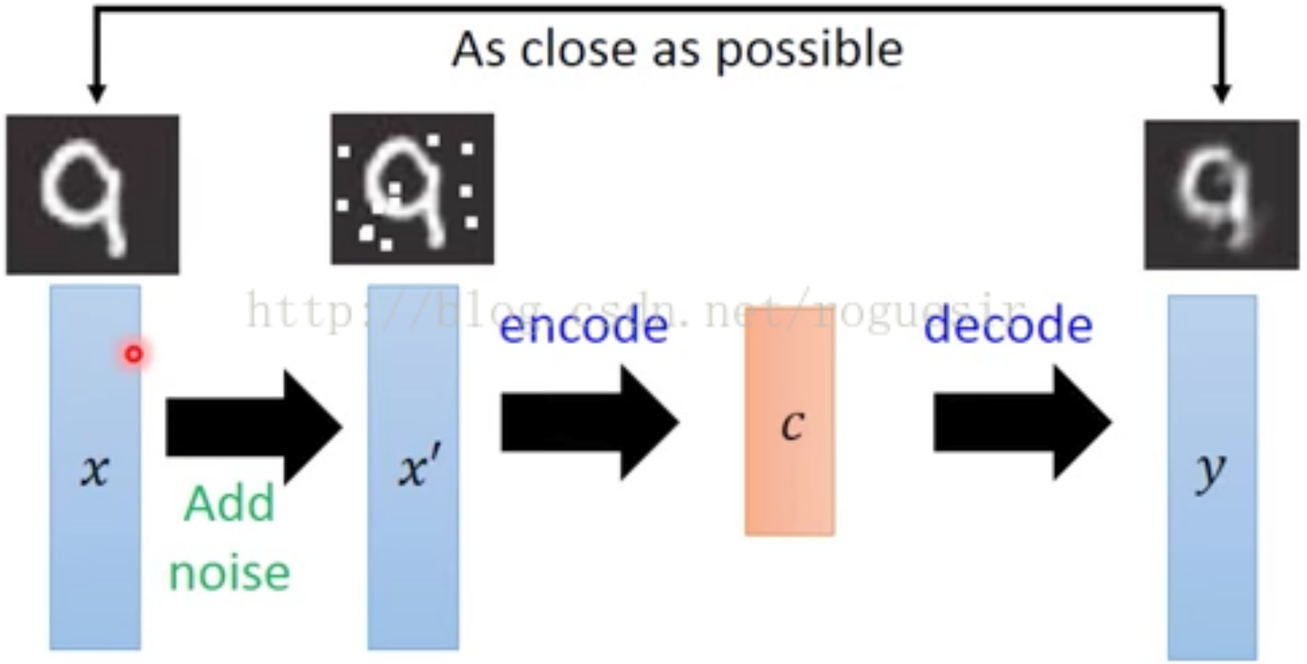


权值消失



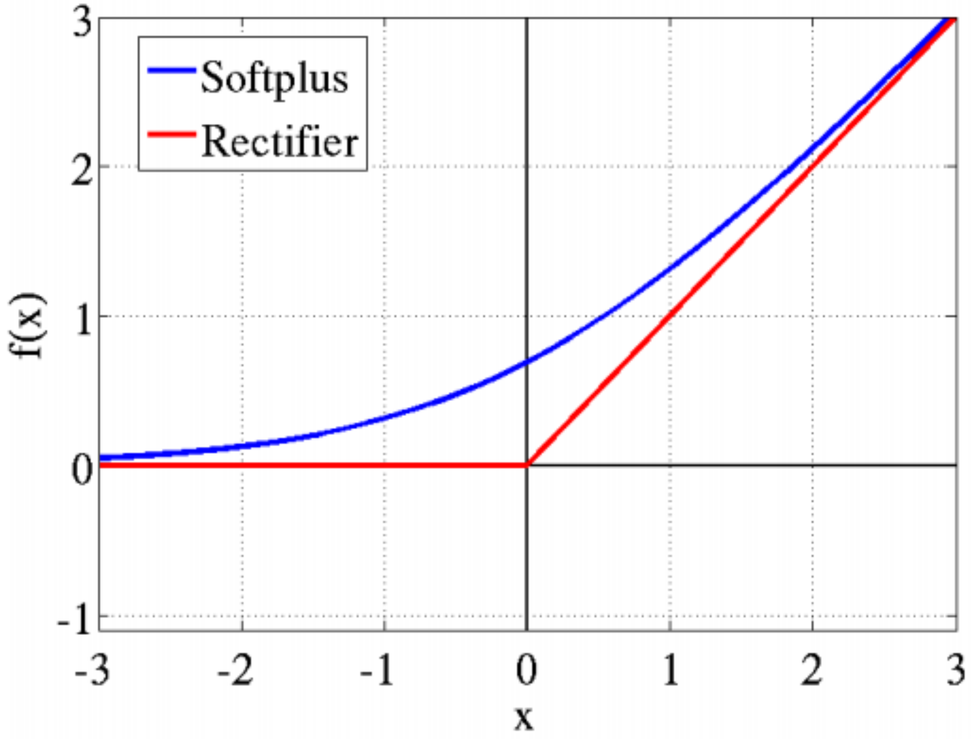
权值消失

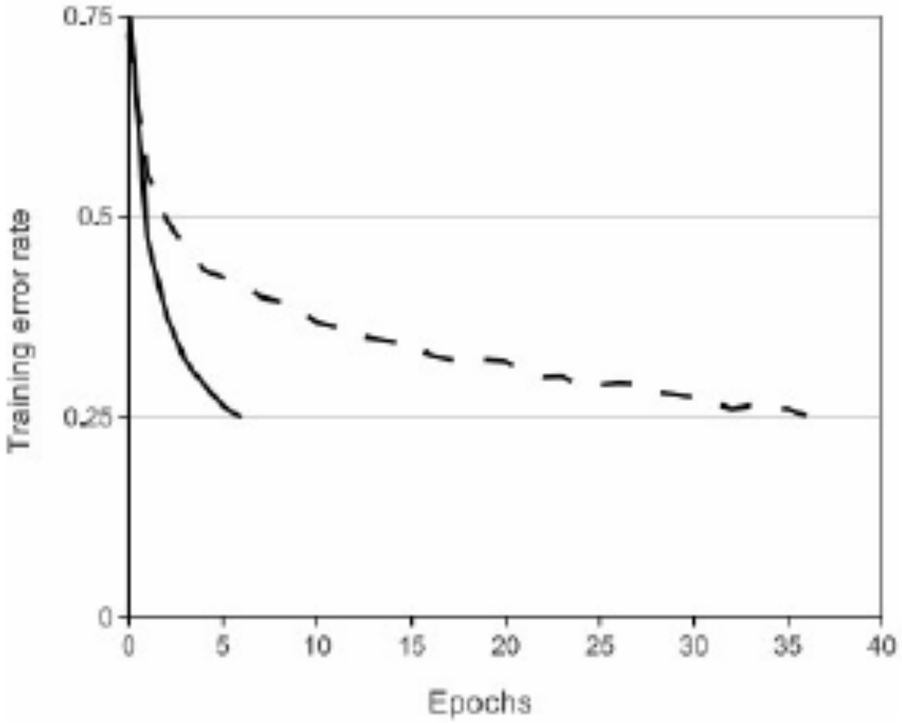
- De-noising auto-encoder



权值消失

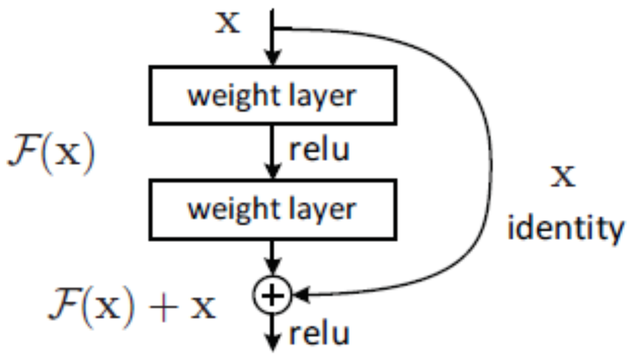
ReLU（Rectified Linear Unit）

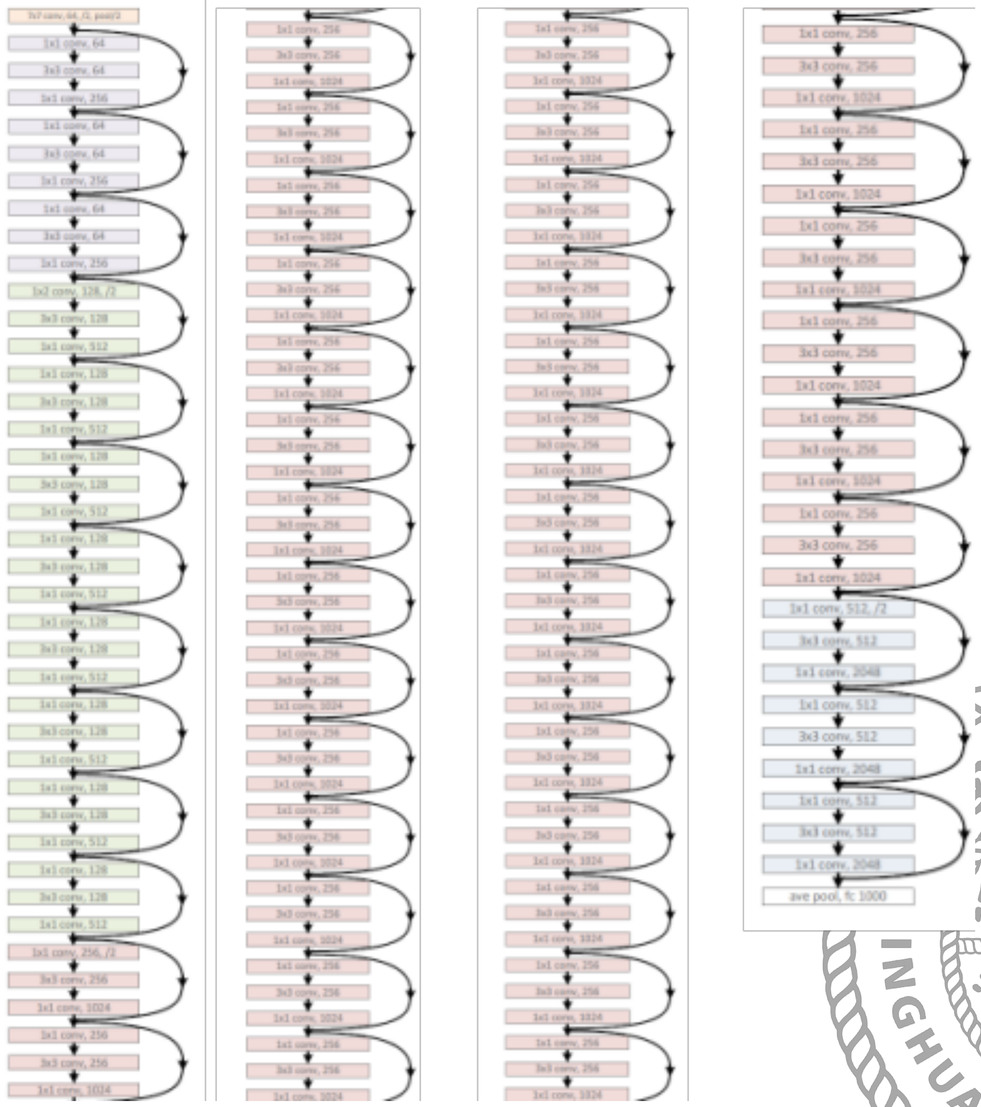




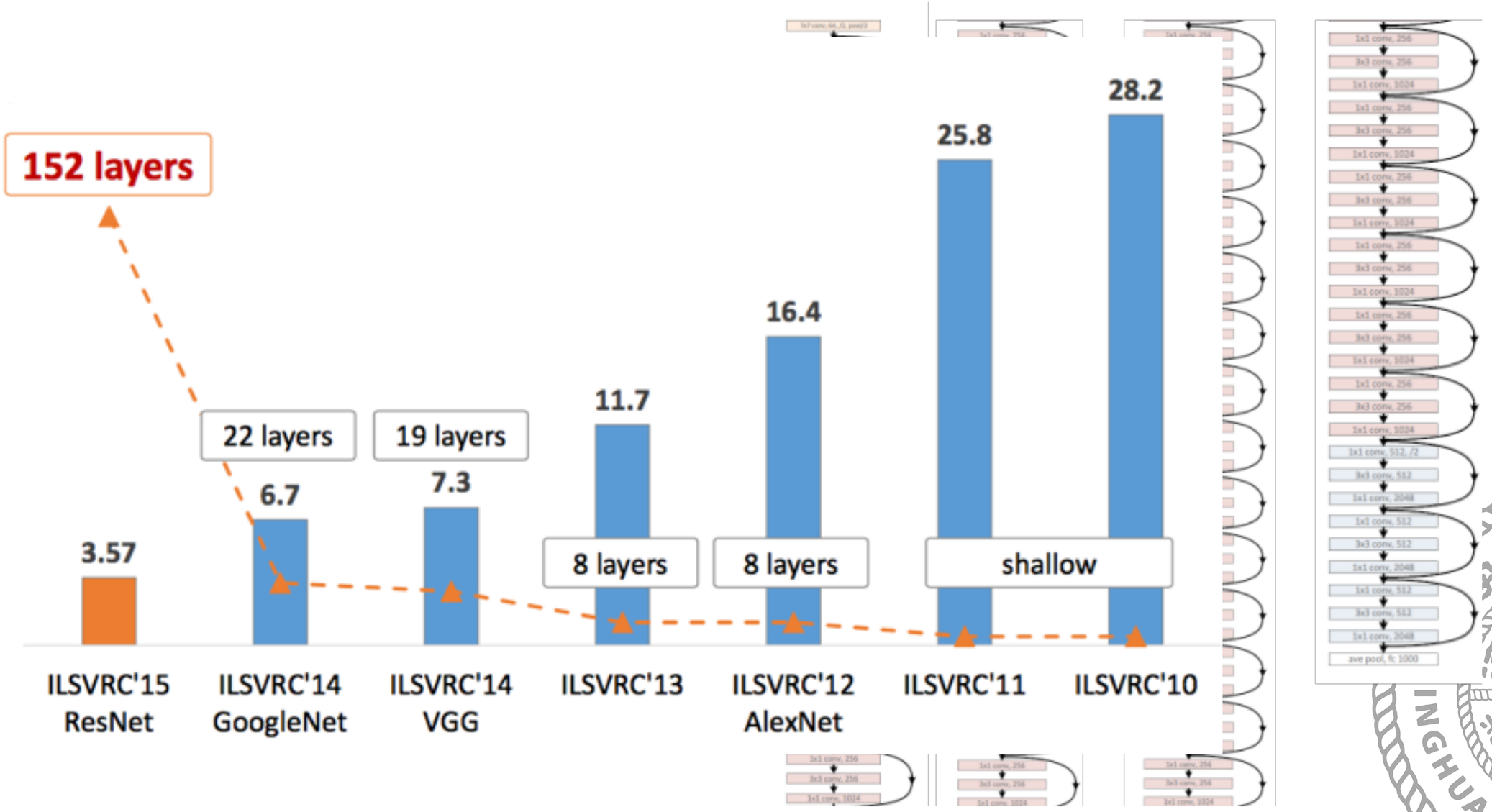
$$f(x) = max(0, x)$$

权值消失


x
weight layer
$\mathcal{F}(x)$
relu
weight layer
x
identity
$\mathcal{F}(x) + x$
relu



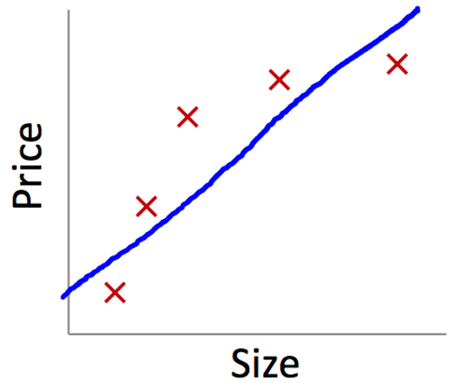
权值消失


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

OVER-FIT

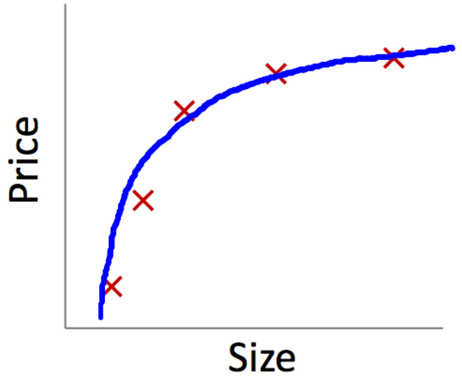
过拟合的问题

过拟合



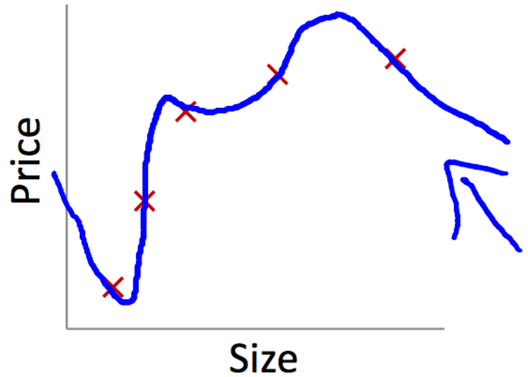
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

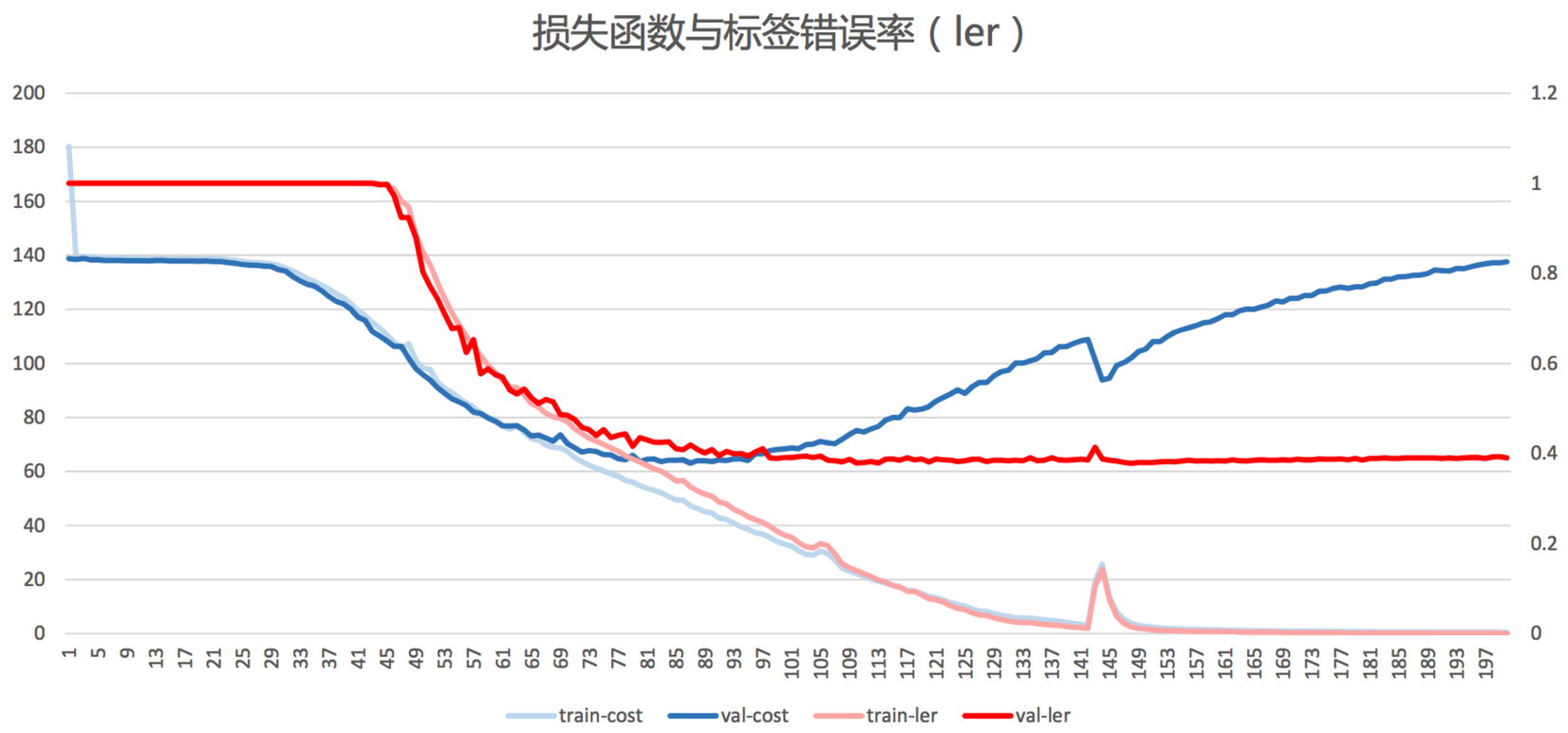


$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" "High variance"

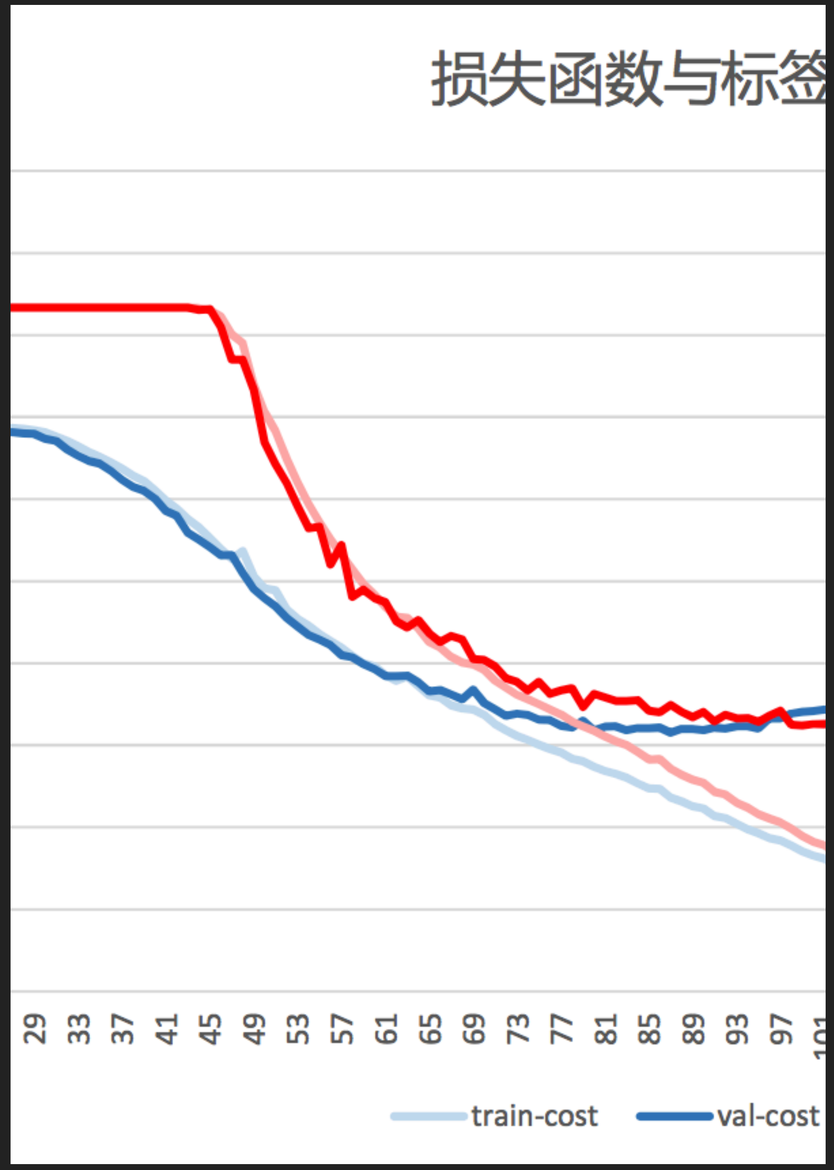
过拟合

参数维度越高过拟合可能越严重



过拟合的解决办法

- More Data
- Early Stopping
- 减少参数数目
- Regularization
- Noise
- Droupout

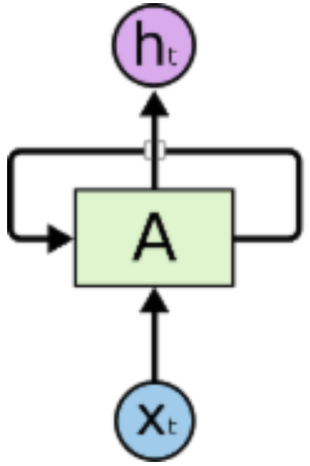


RECURRENT NEURAL NETWORKS

记忆

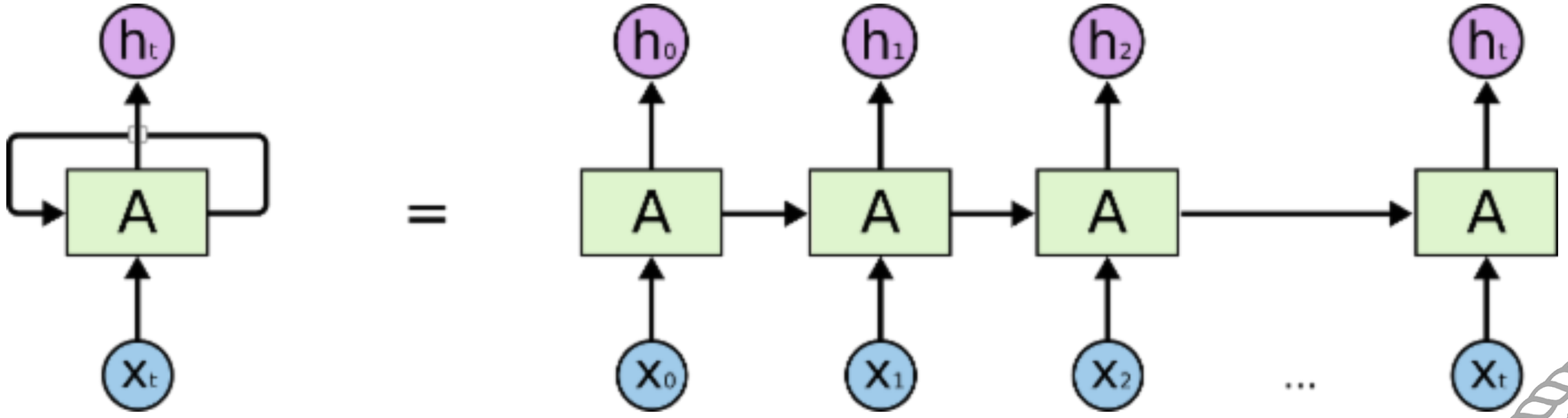
RECURRENT NEURAL NETWORKS

RNN的结构



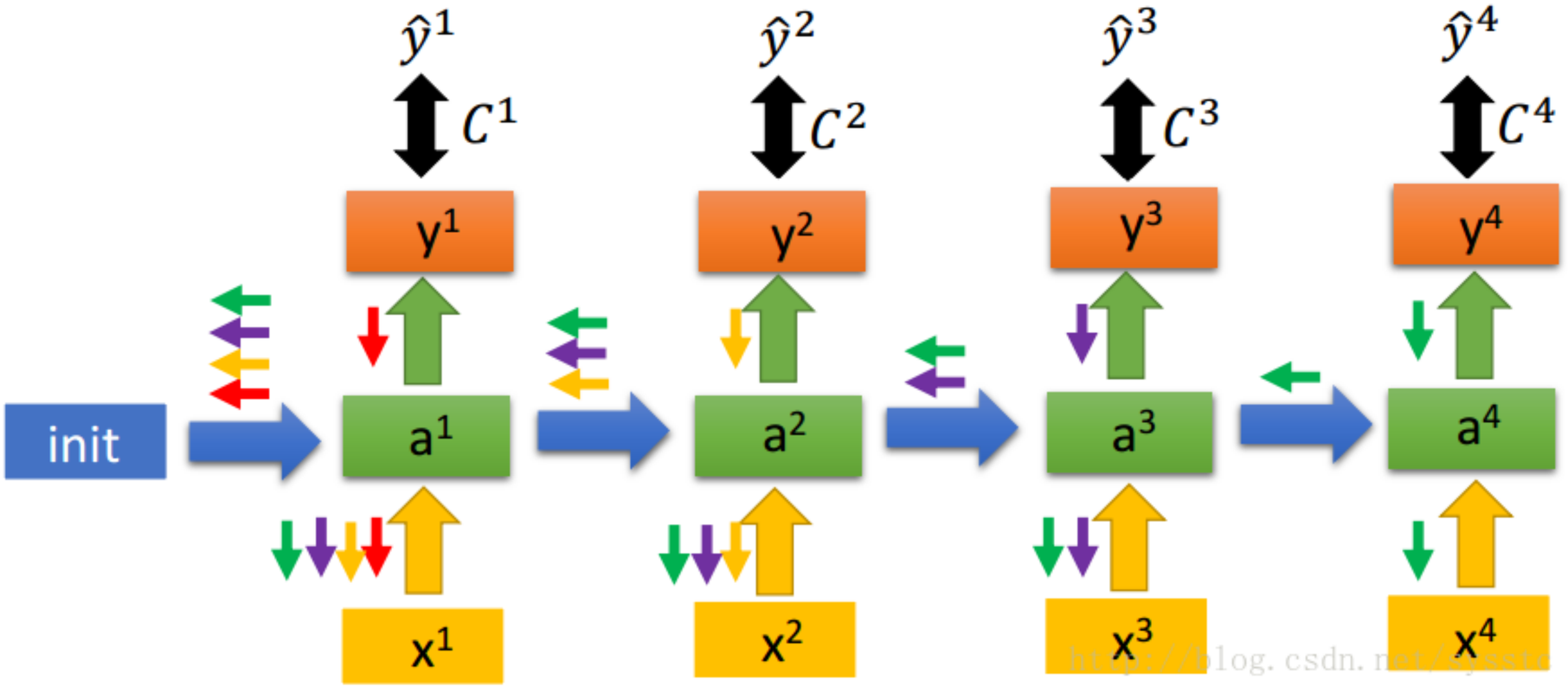
RECURRENT NEURAL NETWORKS

RNN的结构



RECURRENT NEURAL NETWORKS

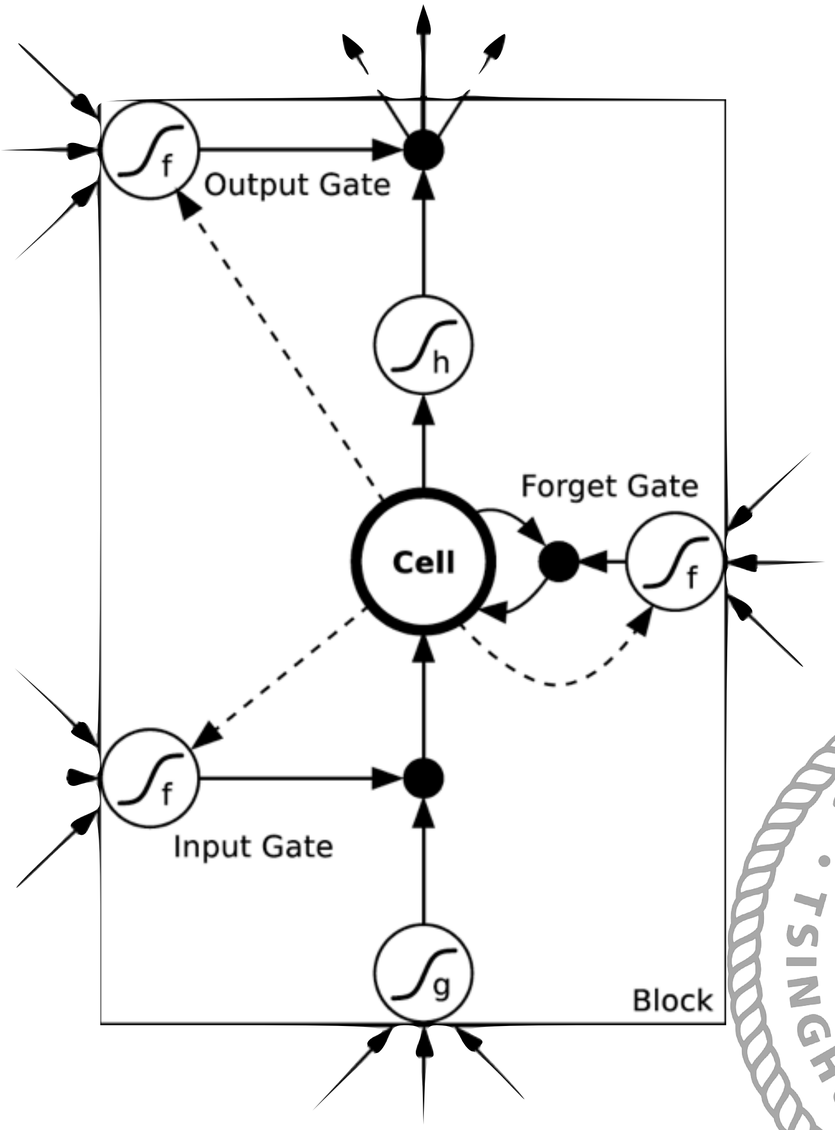
RNN的结构



LONG SHORT-TERM MEMORY

LSTM区别于RNN的地方，主要就在于它在算法中加入了一个记忆细胞“Cell"。

一个cell当中被放置了三扇门，分别叫做输入门、遗忘门和输出门。一个信息进入LSTM的网络当中，可以根据规则来判断是否有用。只有符合算法认证的信息才会留下，不符的信息则通过遗忘门被遗忘。



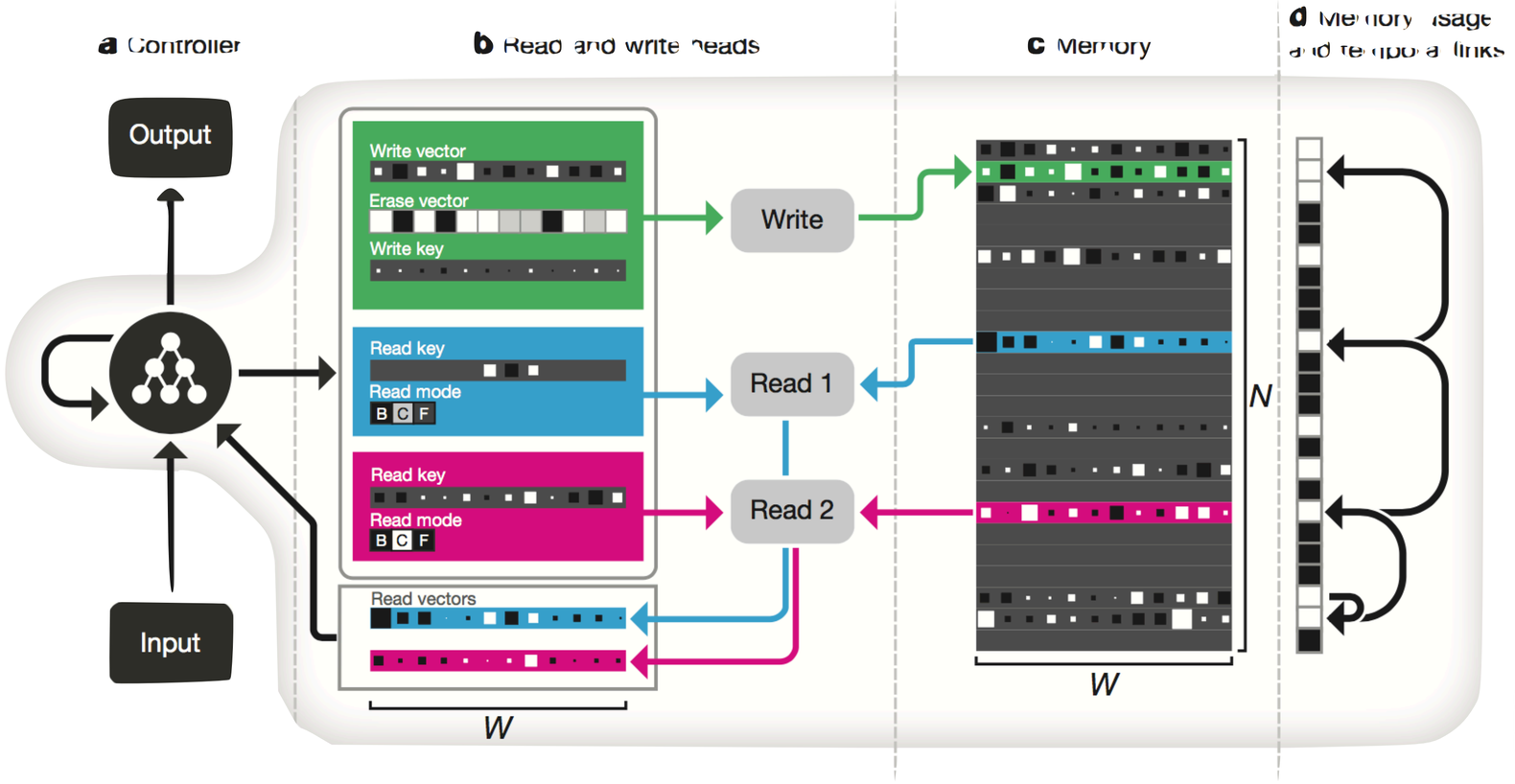
MEMORY

记忆神经网络的三个主流方向

- 门控制结构（RNN，LSTM）
- 基于外部学习模块
  - MemNet （Facebook AI Lab）
  - NTM->DNC（DeepMind）
- 面向连续学习
  - Elastic Weight Consolidation, EWC（DeepMind）
  - Progressive Neural Networks, PNN （DeepMind）

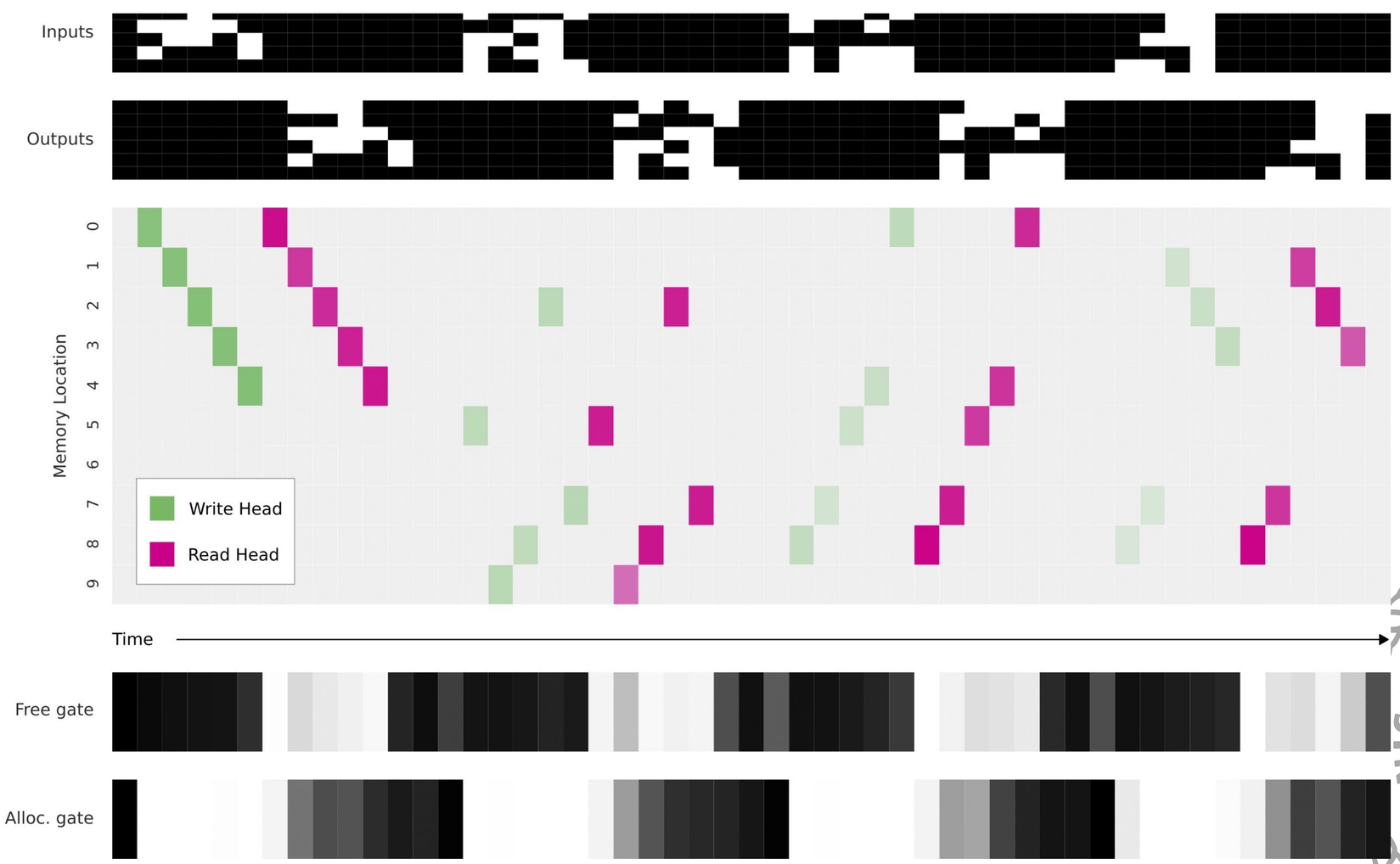
DNC

DIFFERENTIAL NEURAL COMPUTER



DNC

MEMORY ALLOCATION



SEQ2SEQ LEARNING

序列学习

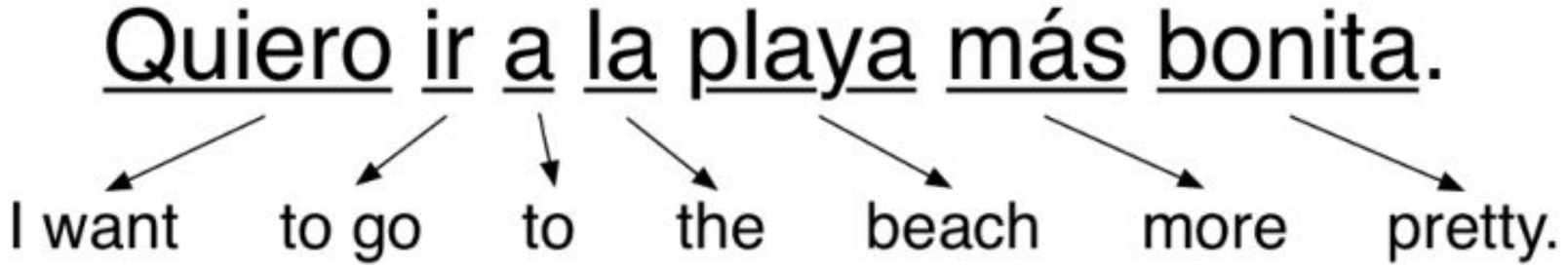
序列学习

机器翻译的问题

Quiero ir a la playa más bonita.

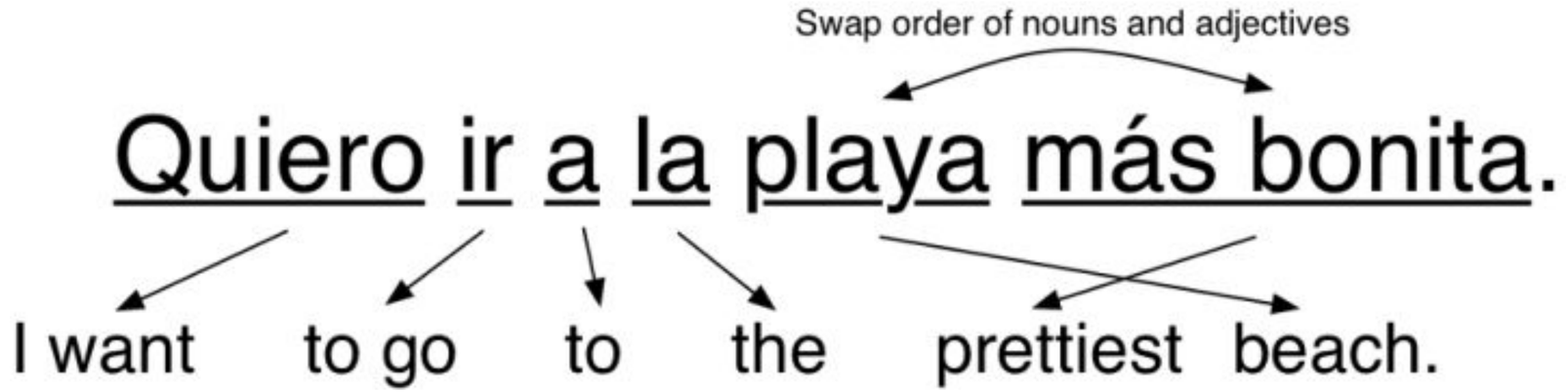
序列学习

机器翻译的问题



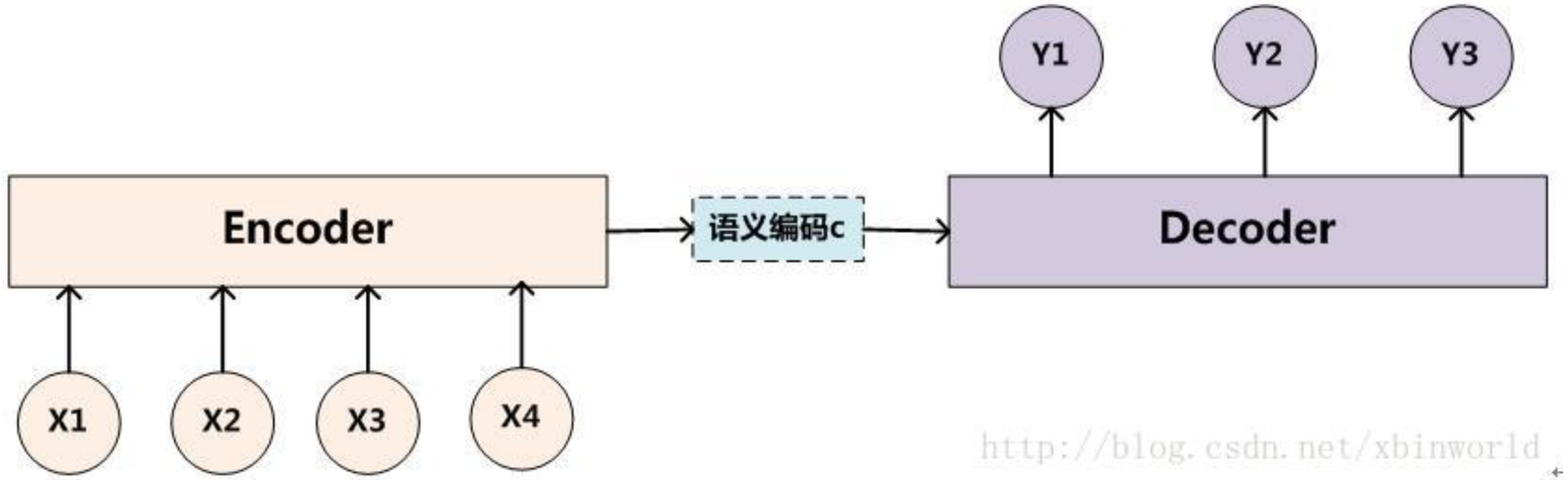
序列学习

机器翻译的问题

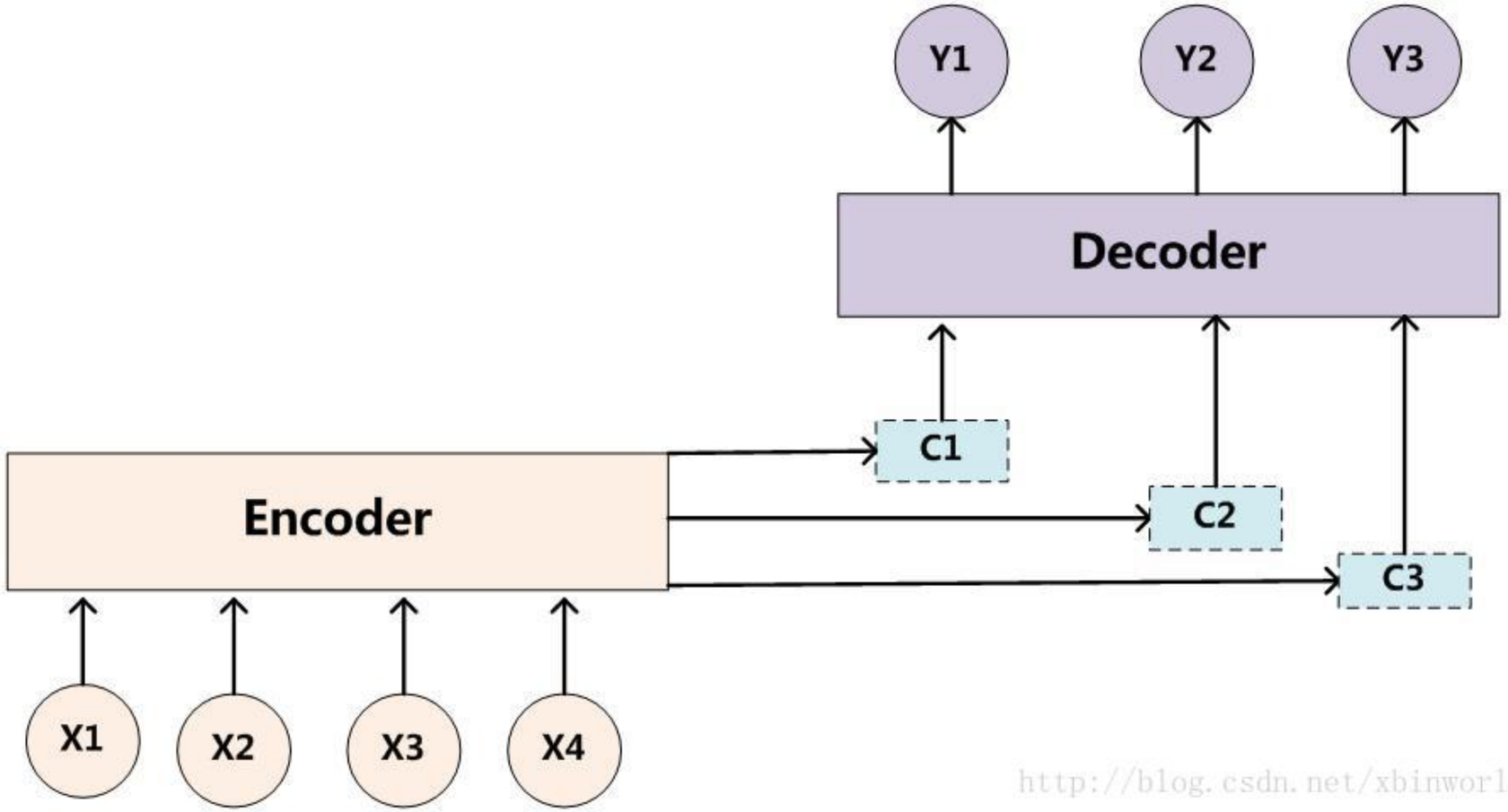


序列学习

Encoder-Decoder网络

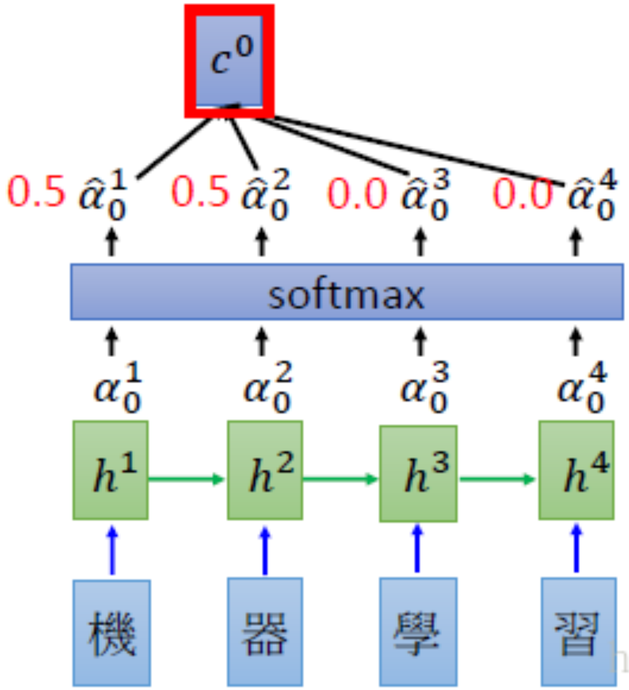


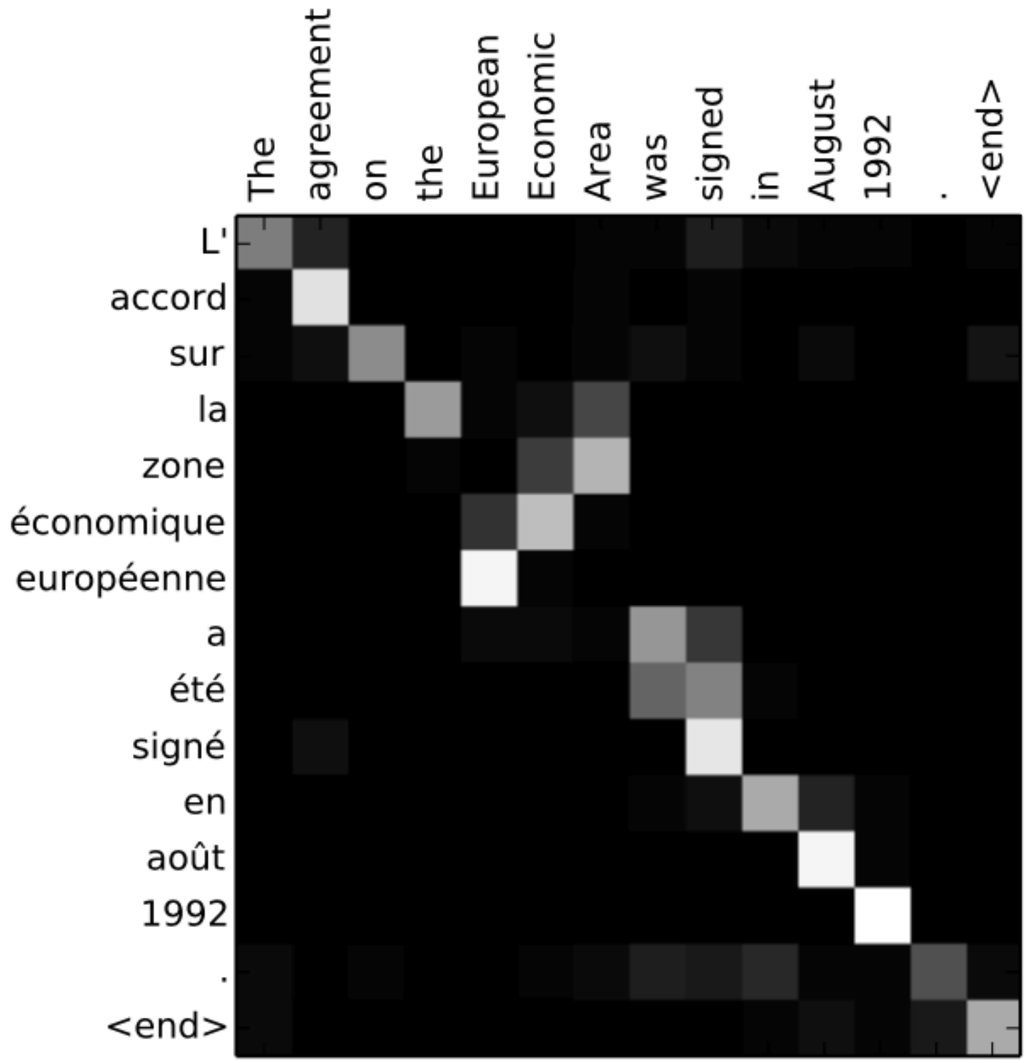
记忆网络



记忆网络

- Attention-based model





OPTIMIZER

优化算法

OPTIMIZER

SGD的主要问题

OPTIMIZER

- 学习率的选择

OPTIMIZER

SGD的主要问题

- 学习率的选择
- 数据集稀疏

OPTIMIZER

SGD的主要问题

- 学习率的选择
- 数据集稀疏
- 局部最小值

OPTIMIZER

SGD的主要问题

- 学习率的选择
- 数据集稀疏
- 局部最小值

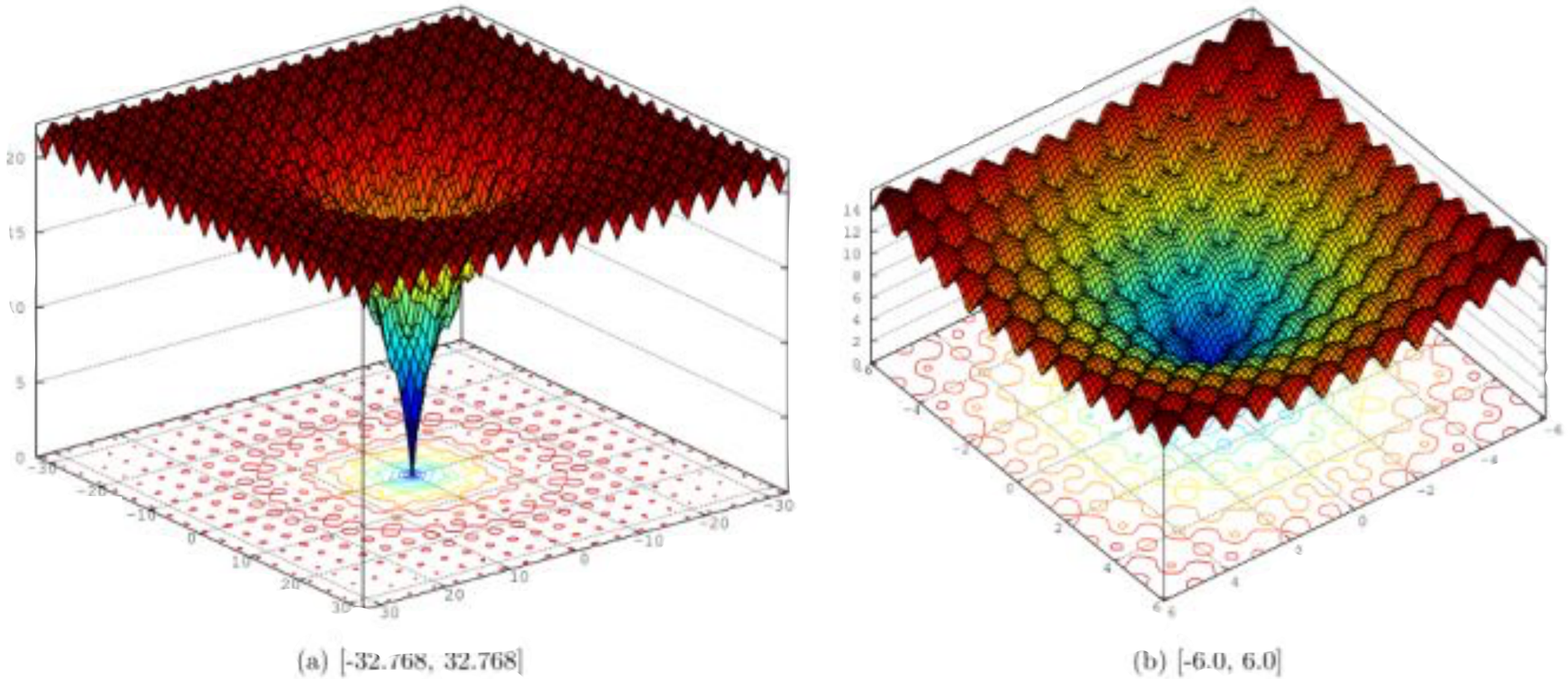
《Identifying and attacking the **saddle point** problem in high-dimensional non-convex optimization》2014

OPTIMIZER

局部最小値

$$f(x) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$$

Dimensions: n
Domain: $-32.768 \leq x_i \leq 32.768$
Global Optimum: $f(x) = 0.0$ at $x = (0.0, 0.0, \ldots, 0.0)$
Operator: AckleyEvaluator
Charts:



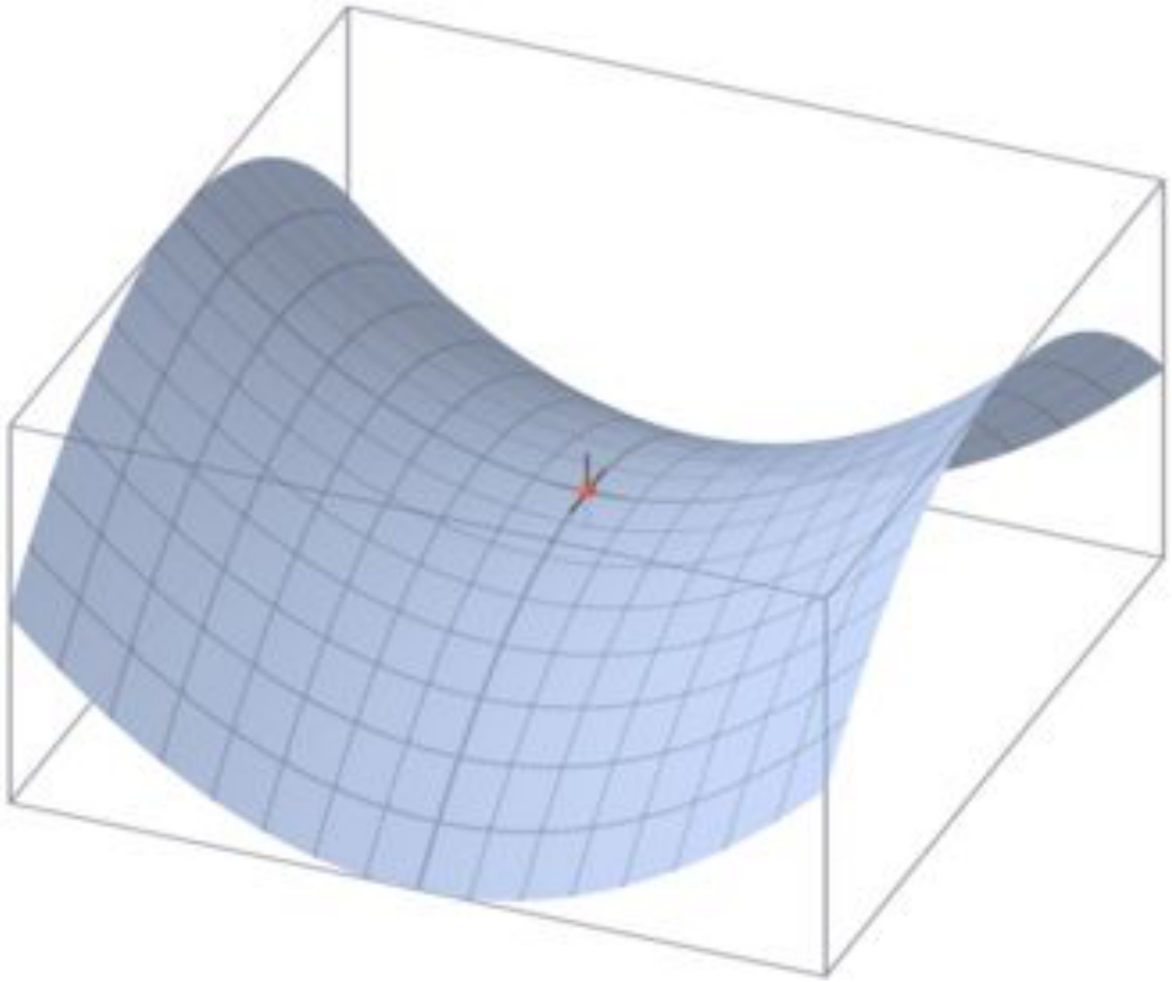
(a) [-32.768, 32.768]

(b) [-6.0, 6.0]

Figure 1: Ackley function plots.

OPTIMIZER

SADDLE POINT



OPTIMIZER

优化函数

OPTIMIZER

优化函数

- Momentum

OPTIMIZER

优化函数

- Momentum
- Nesterov

OPTIMIZER

优化函数

- Momentum
- Nesterov
- Adagrad

OPTIMIZER

优化函数

- Momentum
- Nesterov
- Adagrad
- AdaDelta

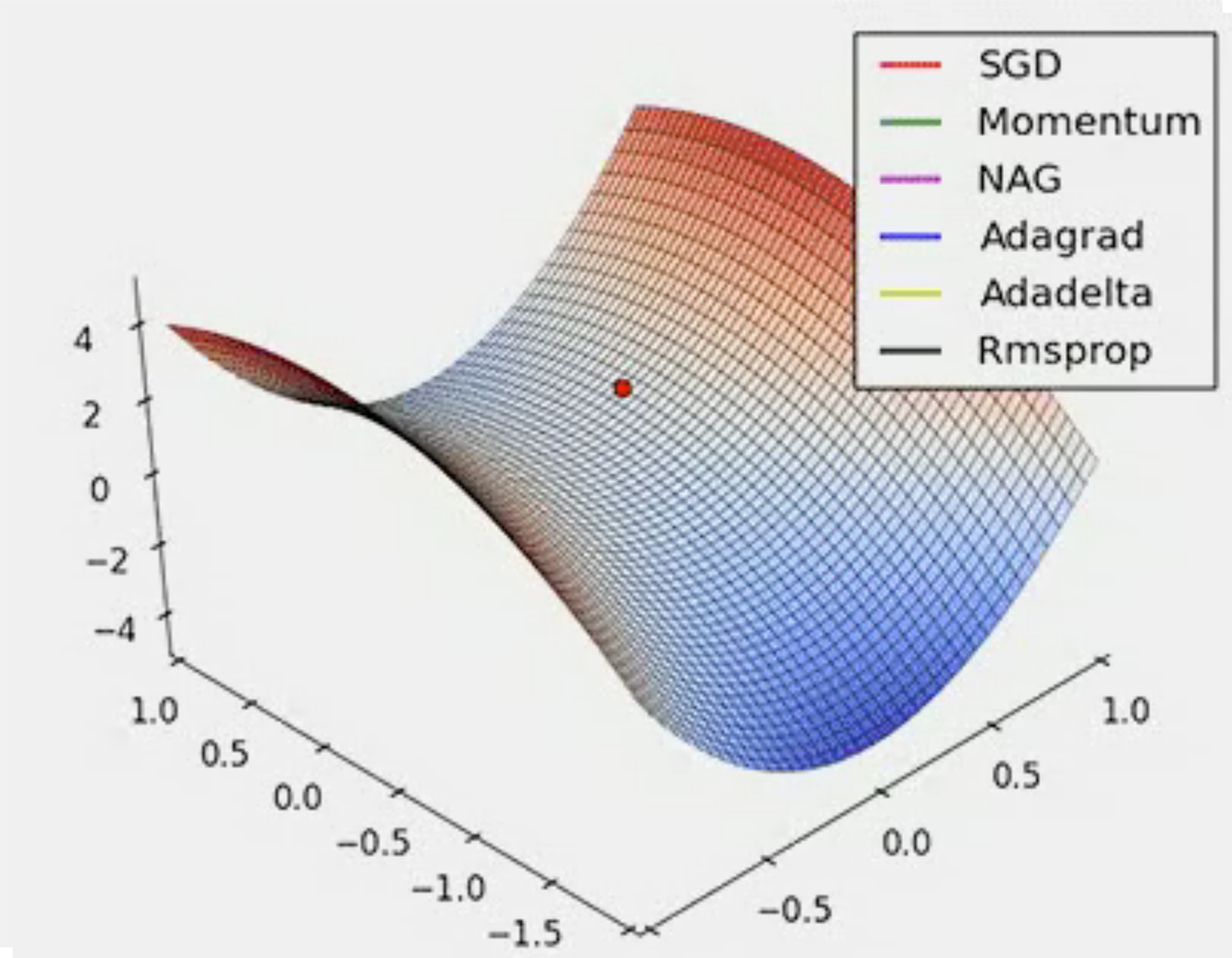
OPTIMIZER

优化函数

- Momentum
- Nesterov
- Adagrad
- AdaDelta
- ADAM (Adaptive Moment Estimation)

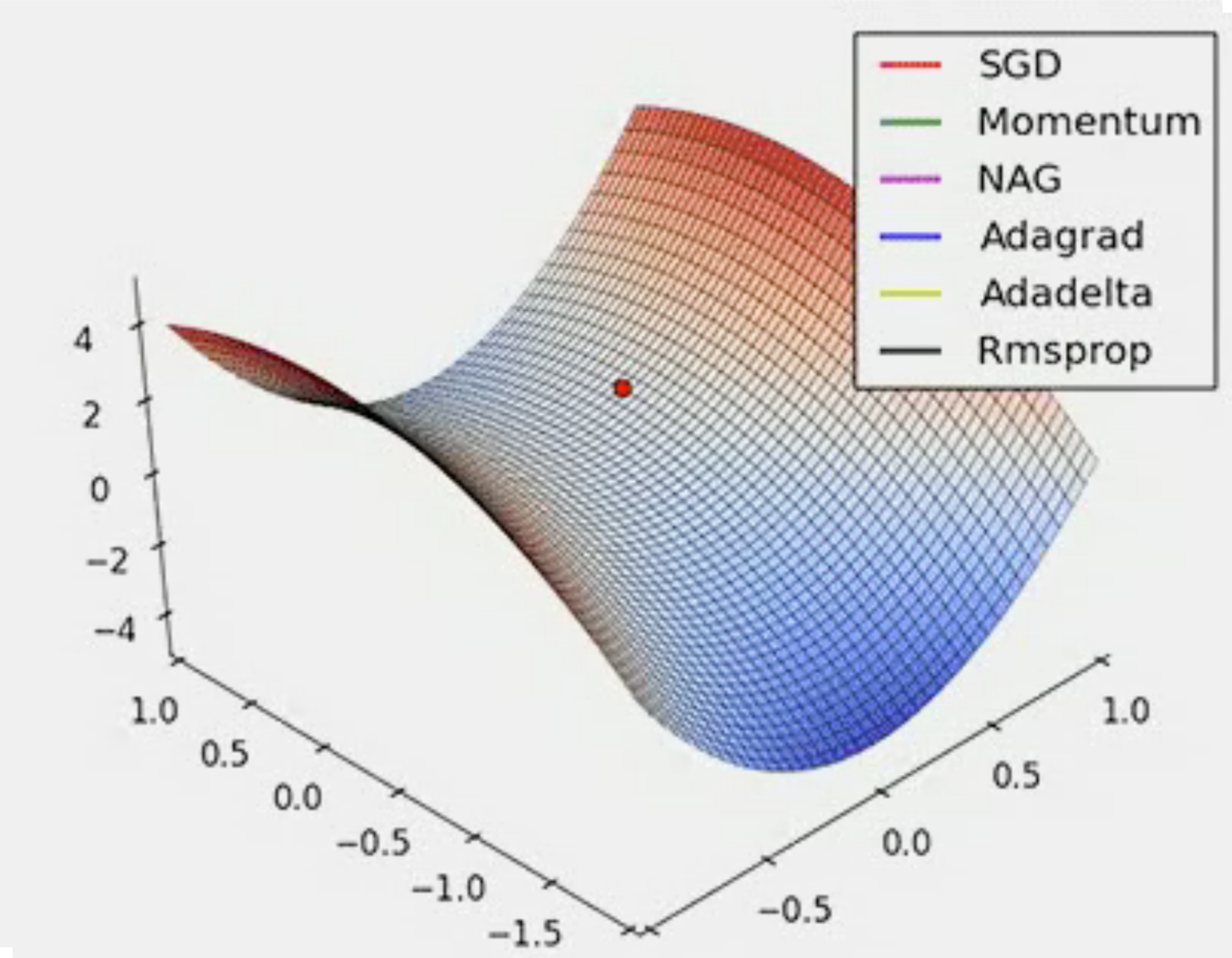
OPTIMIZER

直观对比



OPTIMIZER

直观对比



OPTIMIZER

直观对比