第5讲 深度学习的基本概念

神经网络与深度学习 2026

信息科学技术学院 吴瀚霖

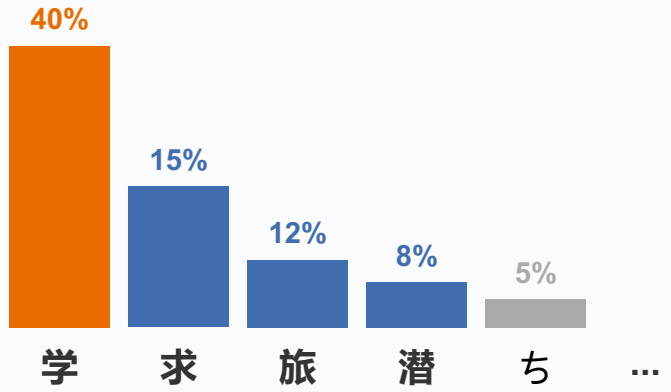
hlwu@bfsu.edu.cn

回顾：LLM内部工作原理

神经网络与深度

x

语言模型 f_θ

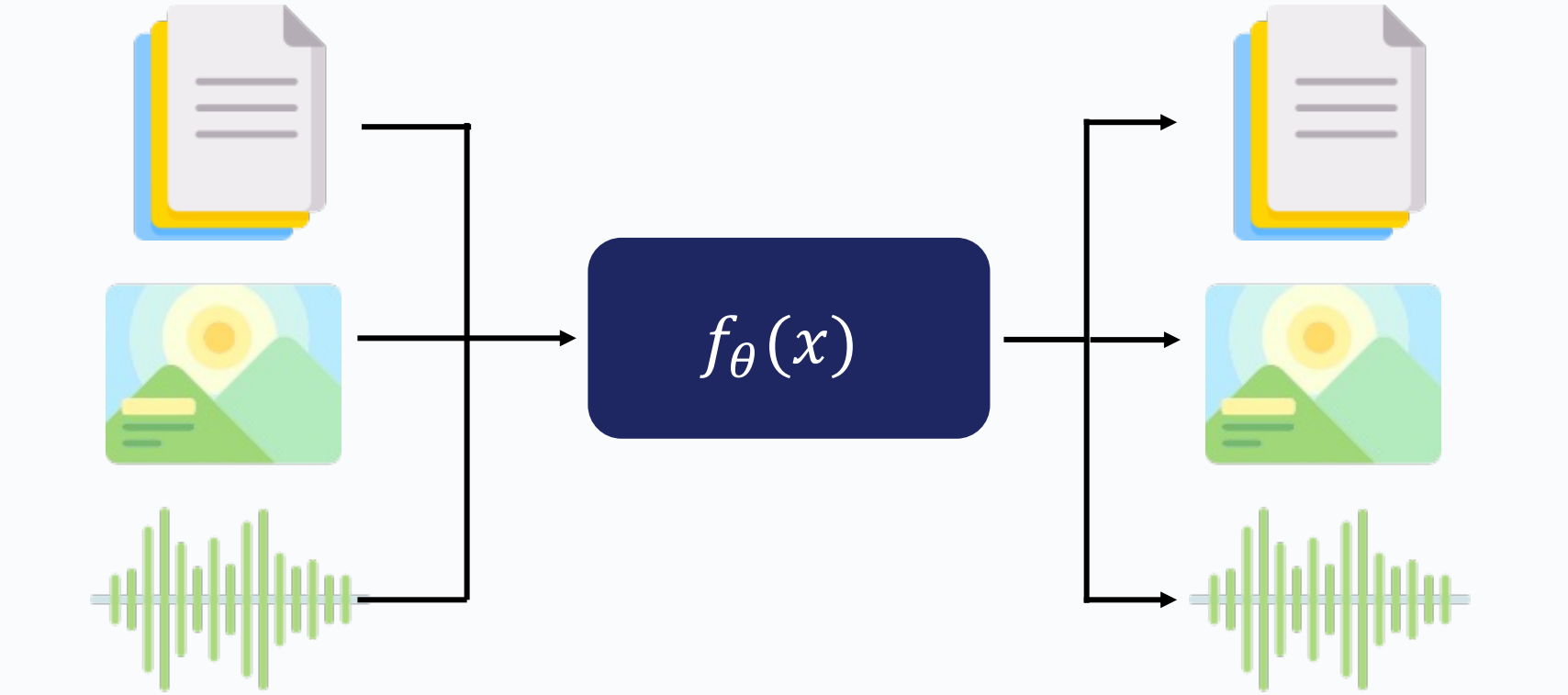


Token 概率分布

如何找到这个f_θ

机器学习（Machine Learning）

AI≈各种各样的函数

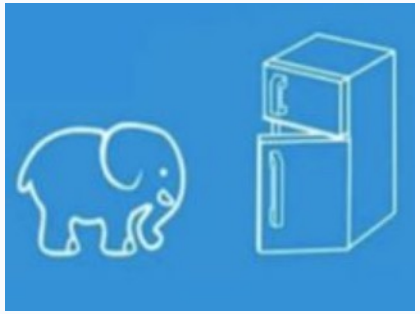


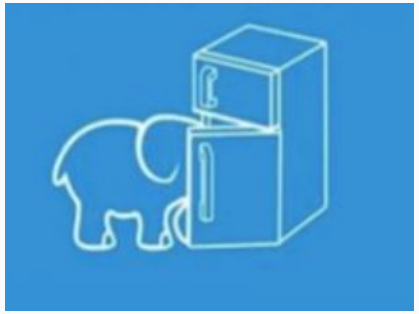
什么是机器学习？

机器学习 ≈

基于数据，让机器自动找到这个函数

找函数：同把大象放进冰箱一样简单





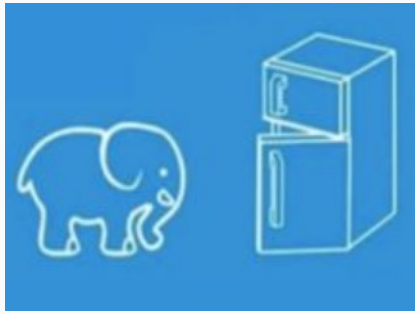
1 我有什么选择

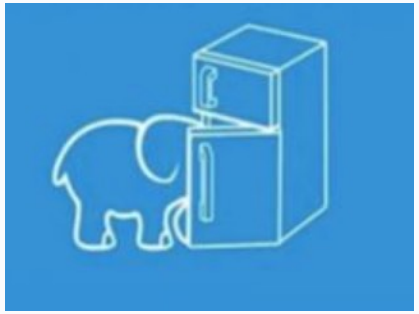
2 什么样的f更好

3 选一个最好的

训练

找函数：同把大象放进冰箱一样简单





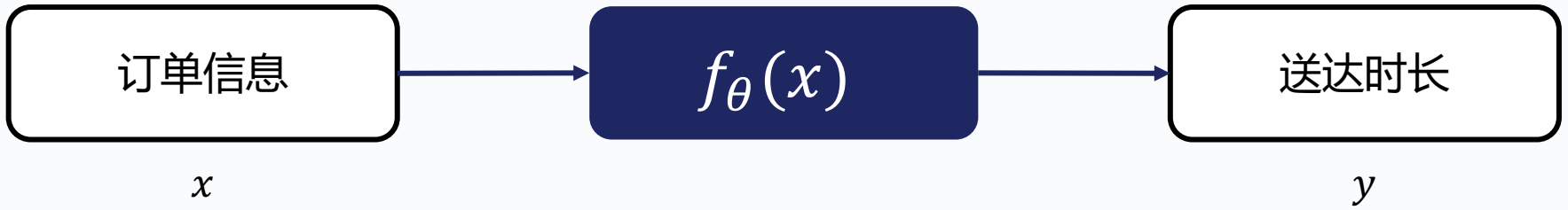
1 **我有什么选择**

2 **什么样的f更好**

3 **选一个最好的**

确定一个候选的函数集合

我的外卖什么时候送达



函数的输入只能是数字

Feature

距离	x_1
下单时间	x_2
交通指数	x_3
是否节假日	x_4
	(0 or 1)

线性回归

$$y = w_1 x_1 + b$$

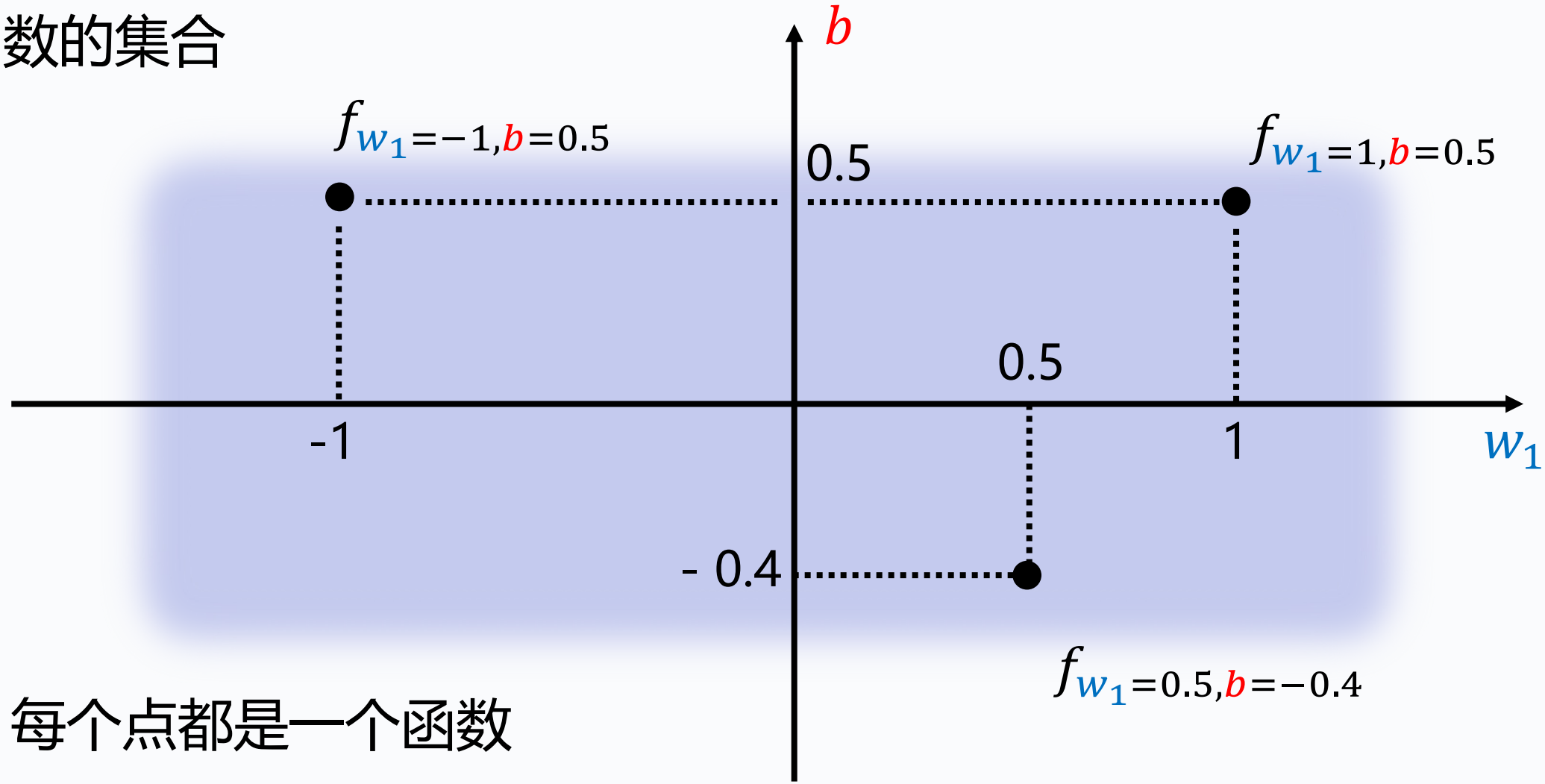
w_1：送达时间与距离呈某种比例关系

b：固定的时间（出餐）

w_1, b 数值未知 → **参数（Parameter）**

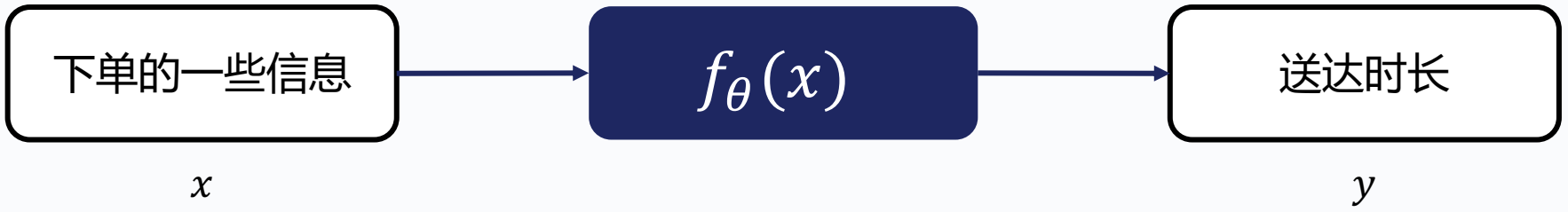
$y = w_1 x_1 + b$　　w_1, b 数值未知　　**参数（Parameter）**

函数的集合



每个点都是一个函数

我的外卖什么时候送达



$$y = w_1 x_1 + b$$

出于人类对于这个任务的理解

模型 (Model)

$$y = w_1 x_1 + w_2 x_2 + b$$

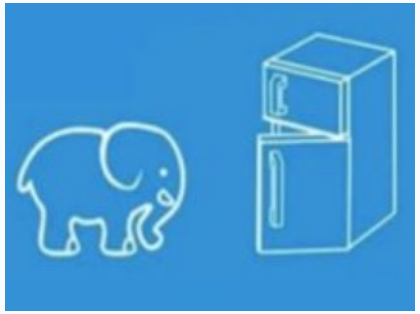
$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

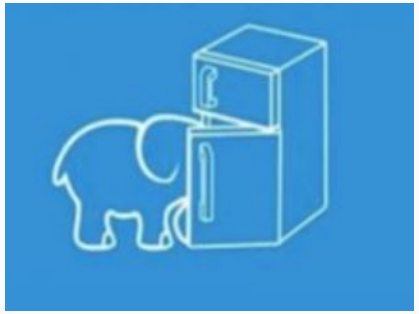
$$y = w_1 x_1 + w_2 x_2 x_4 + w_3 x_3{}^2 + b$$

有无穷的可能

距离	x_1
下单时间	x_2
交通指数	x_3
是否节假日	x_4
	(0 or 1)

找函数：同把大象放进冰箱一样简单





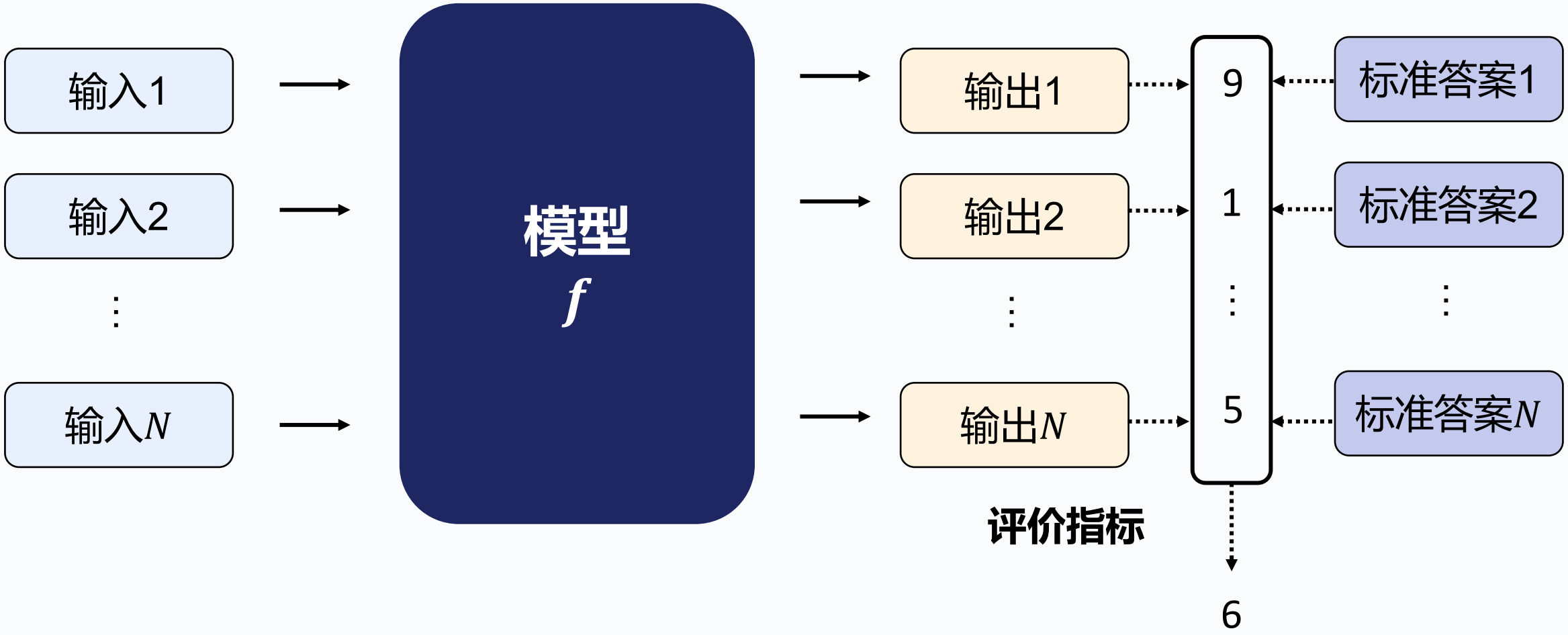
1 我有什么选择

2 什么样的f更好

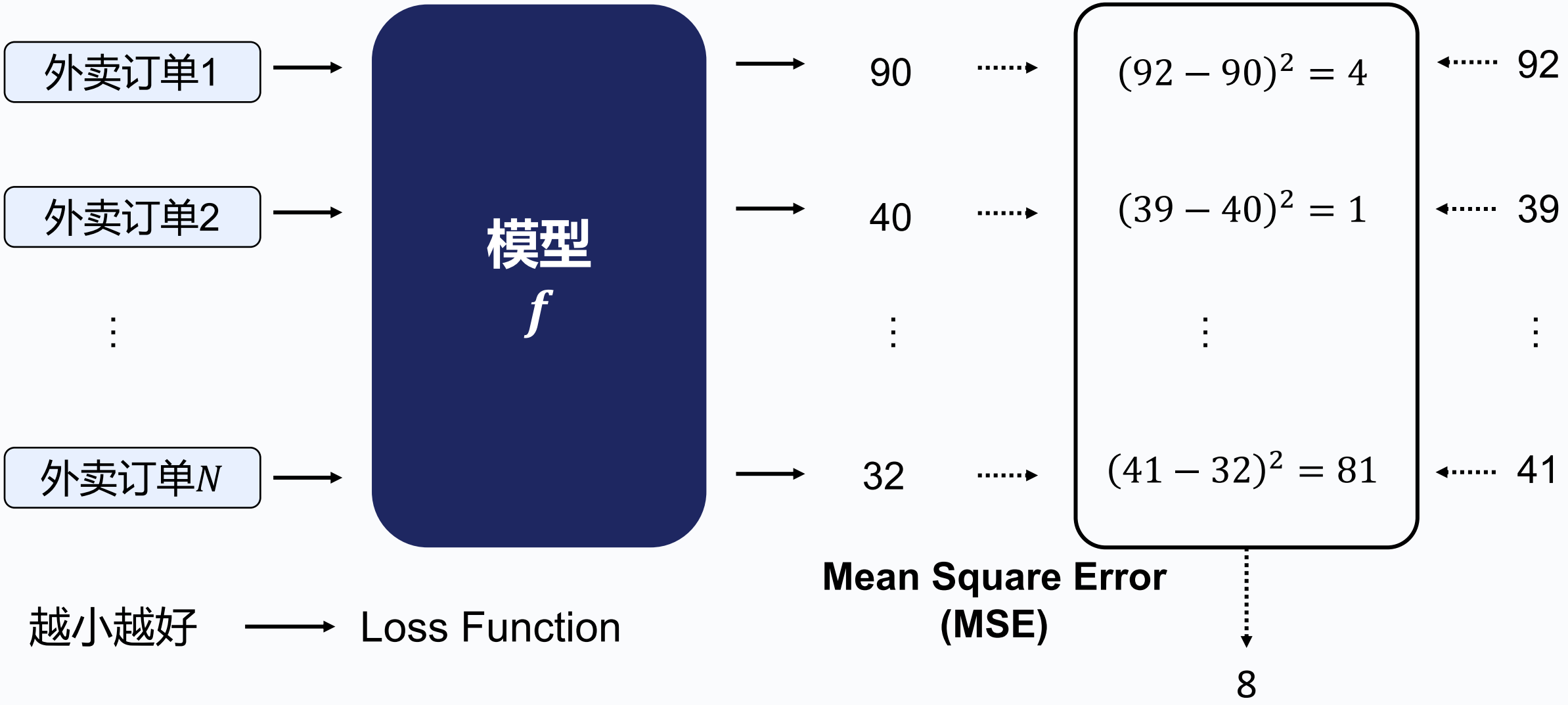
3 选一个最好的

给我一个f，我要知道它是不是我要的

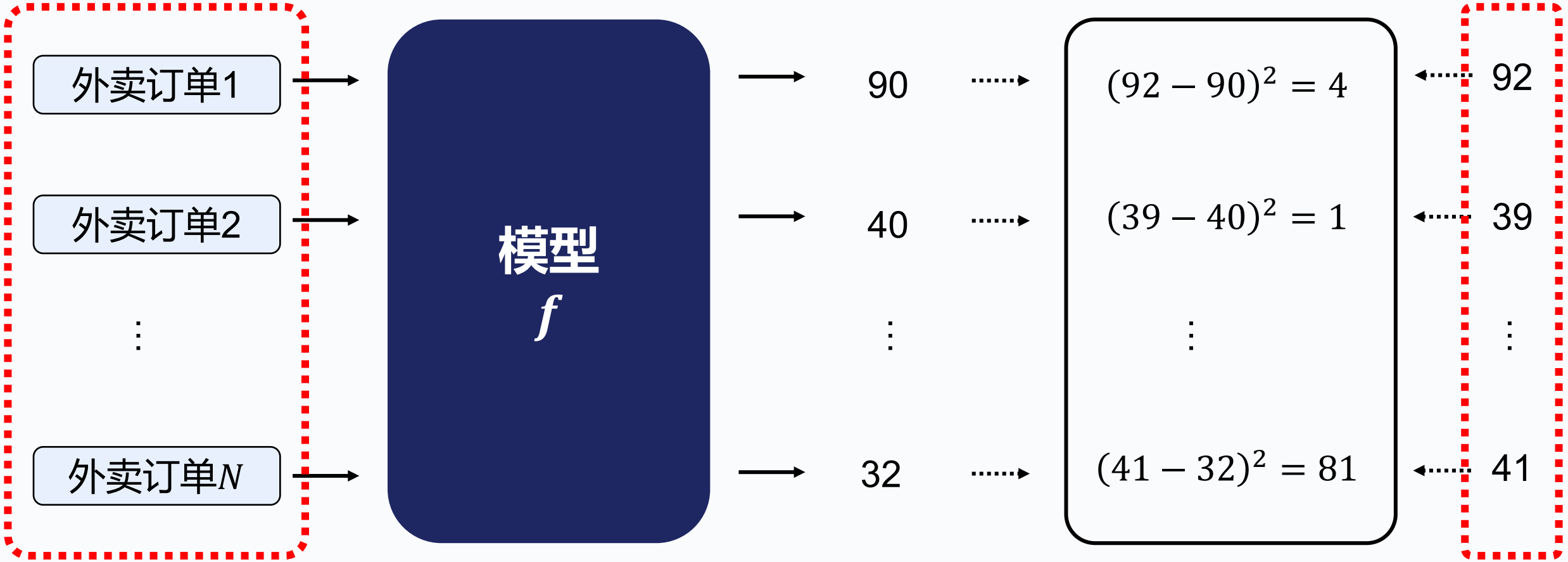
定一个评价标准



定一个评价标准


外卖订单1
外卖订单2
外卖订单N
模型
f
90
40
32
92
39
41
$(92-90)^2=4$
$(39-40)^2=1$
$(41-32)^2=81$
Mean Square Error
(MSE)
8
越小越好
Loss Function

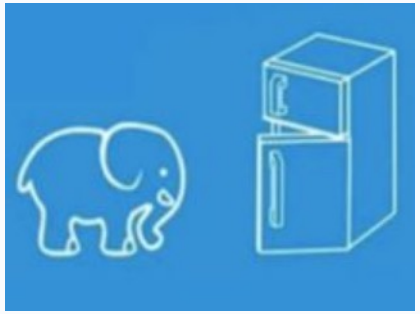
定一个评价标准

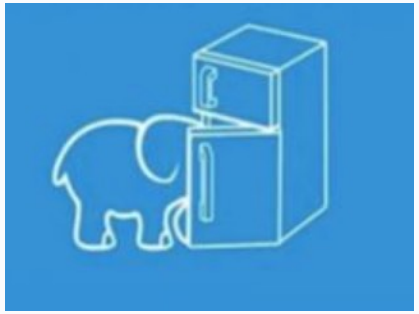


- 训练数据集（Training Dataset）
- 样本（Sample）

$$\mathcal{D}=\{(x^i,y^i)\}_{i=1}^{N}$$

找函数：同把大象放进冰箱一样简单





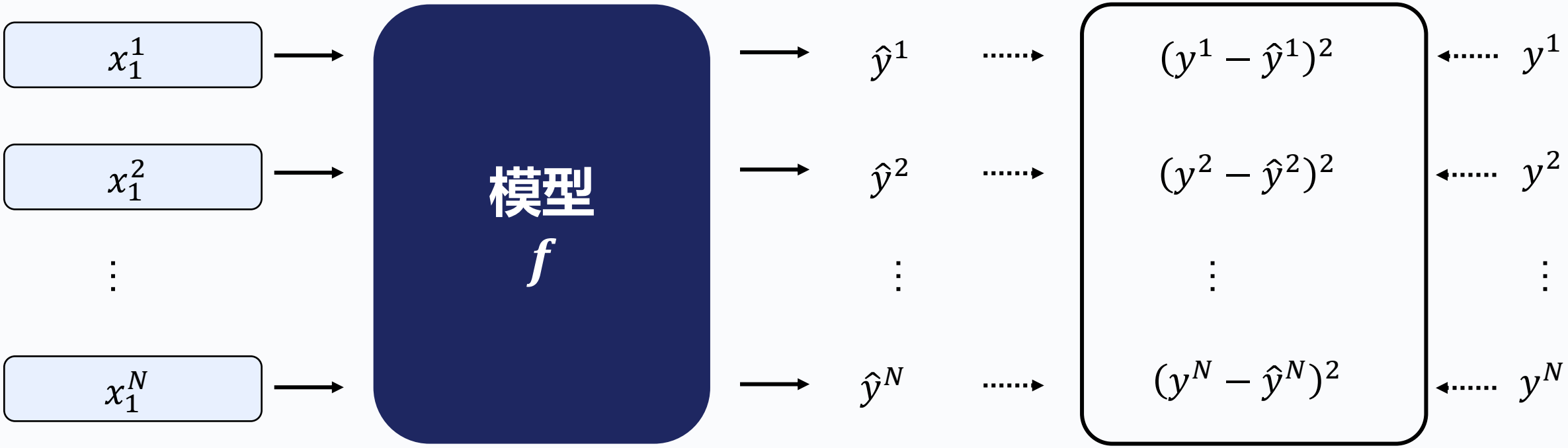
1 我有什么选择

2 什么样的f更好

3 选一个最好的

把Loss最低的找出来

选一个最好的出来



先把 Loss 的公式写出来

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(w_1 x_1^i + b - y^i\right)^2$$

$$y = w_1 x_1 + b$$

选一个最好的出来

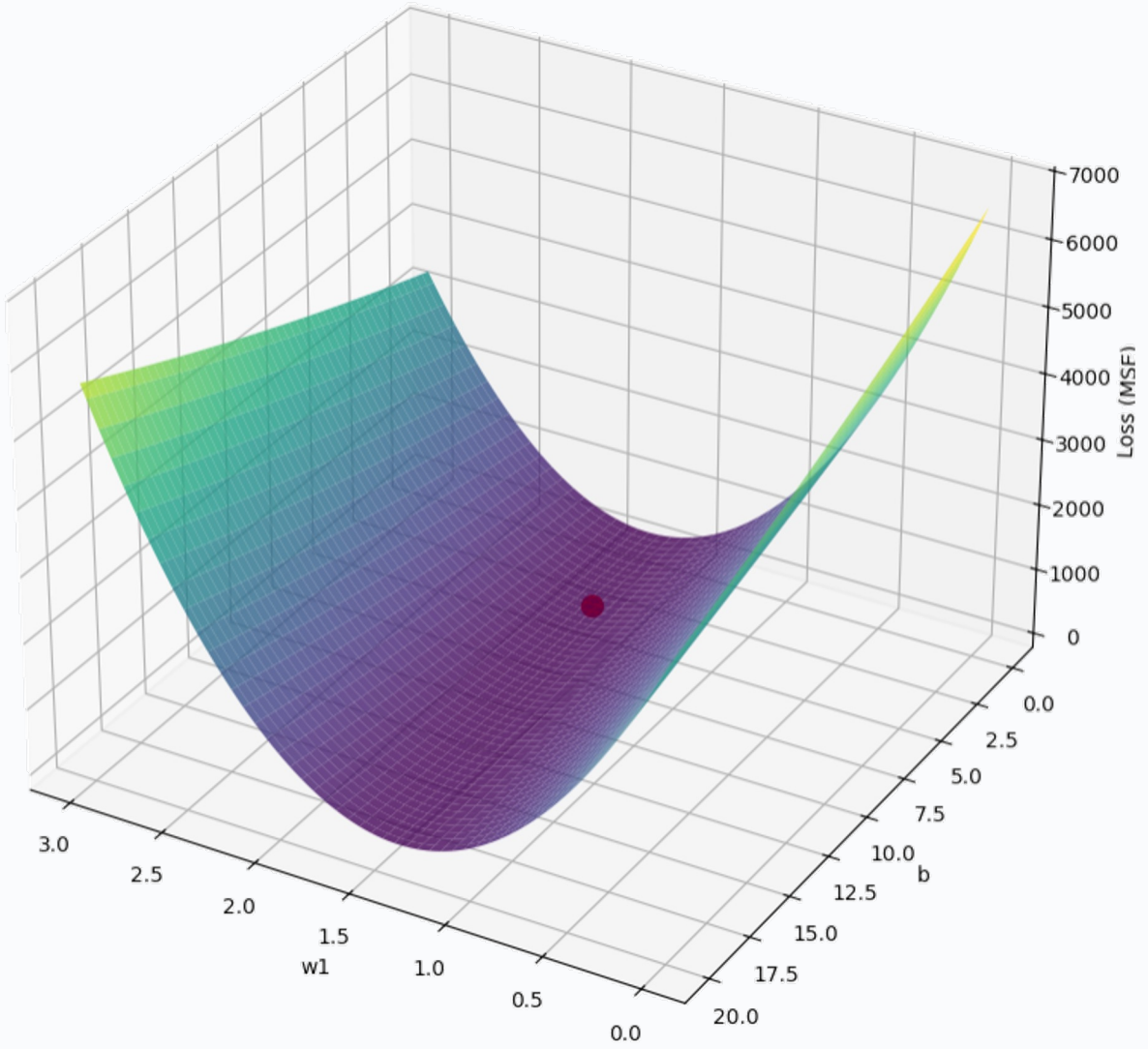
$$L(w_1, b) = \frac{1}{N}\sum_{i=1}^{N}\left(w_1 x_1^i + b - y^i\right)^2$$

最优化问题

$$w_1^*, b^* = \arg\min_{w_1, b} L(w_1, b)$$

暴力算出所有的loss

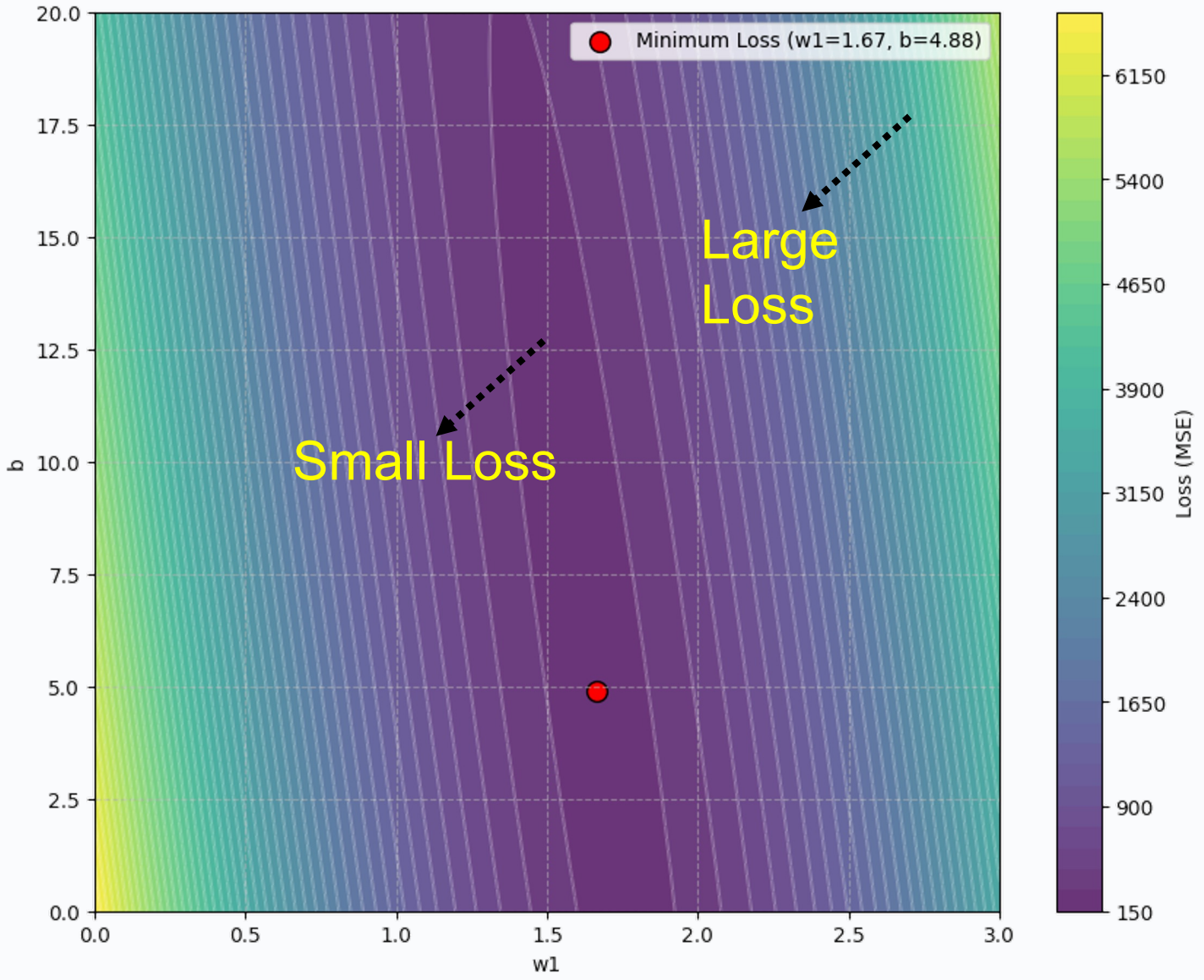
$$w_1^*, b^* = \arg\min_{w_1, b} L(w_1, b)$$



暴力算出所有的loss

等高线图

$$y = w_1 x_1 + b$$



选一个loss最低的函数

Loss是MSE

$$L(w_1, b) = \frac{1}{N}\sum_{i=1}^{N}\left(w_1 x_1^i + b - y^i\right)^2$$

模型定义成这样

$$y = w_1 x_1 + b$$

线性回归
Linear Regression

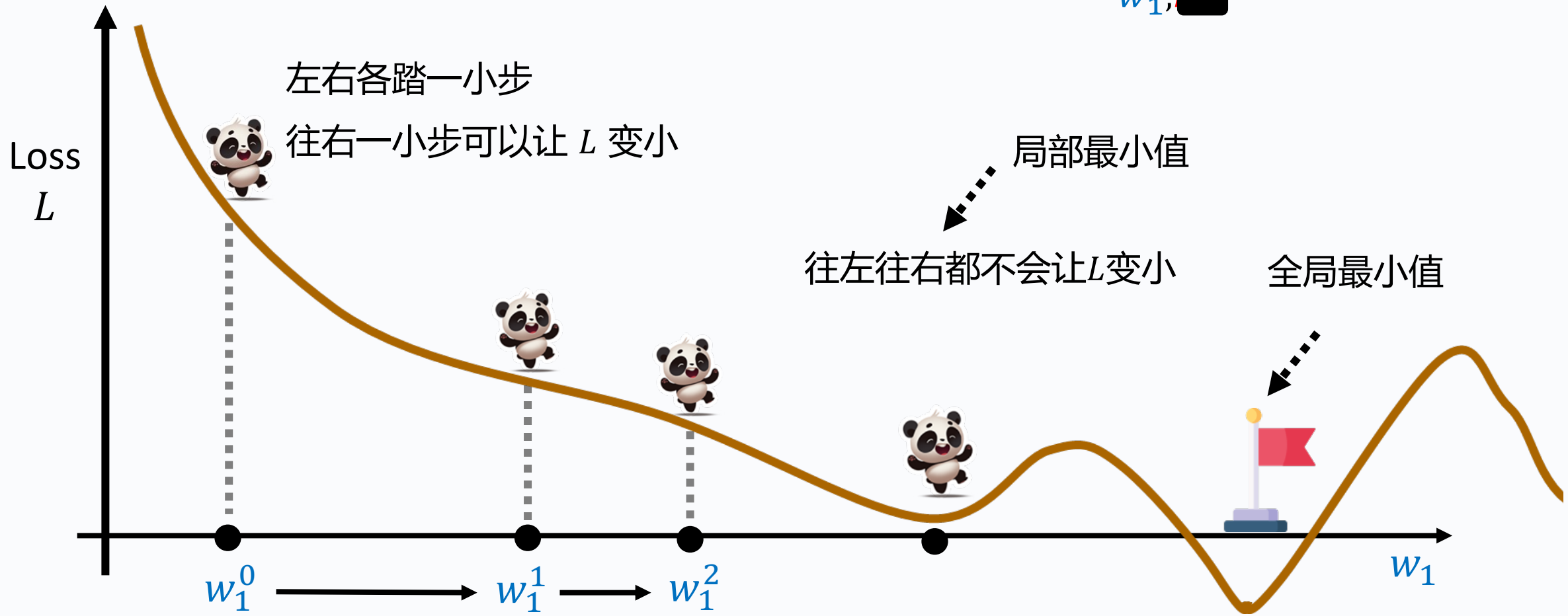
$$w_1^*, b^* = arg\min_{w_1, b} L(w_1, b)$$

这个问题很幸运，有公式解

但我们需要更通用的做法

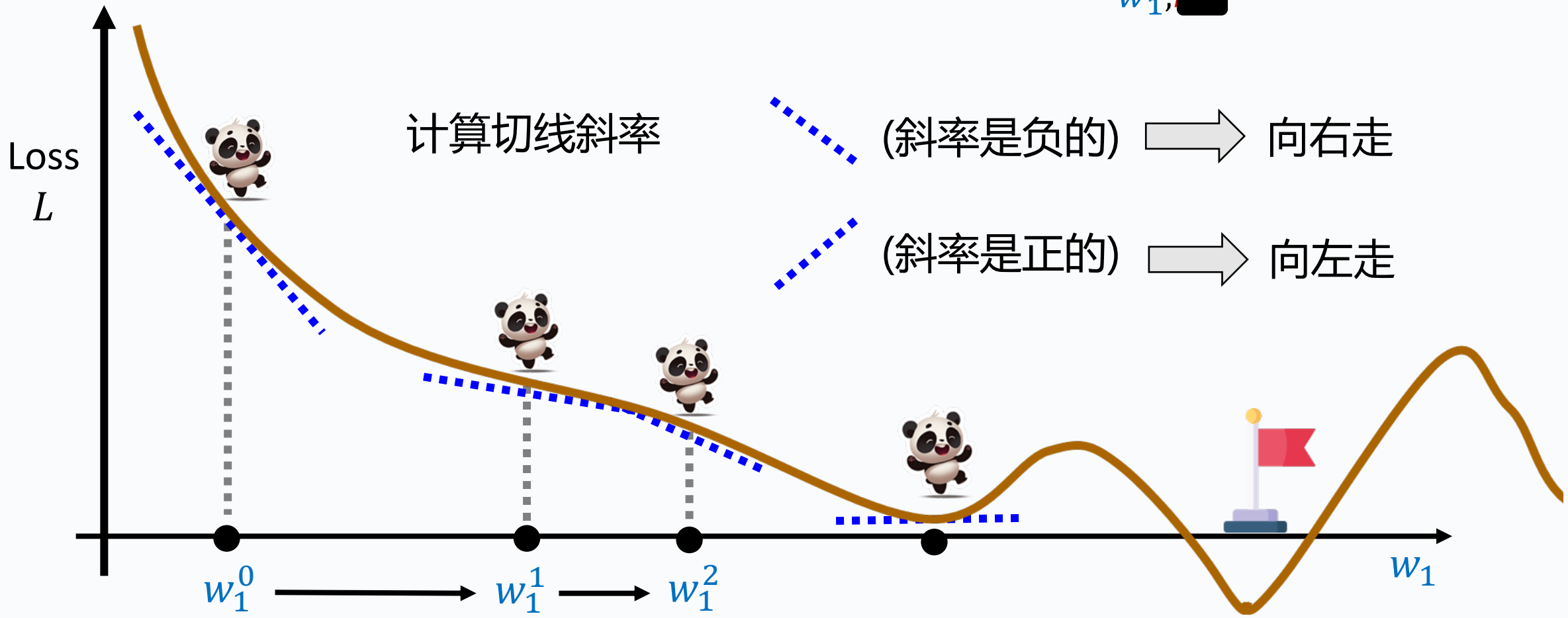
梯度下降 Gradient Descent

$$w_1^*, \blacksquare = \arg\min_{w_1, \blacksquare} L(w_1, \blacksquare)$$



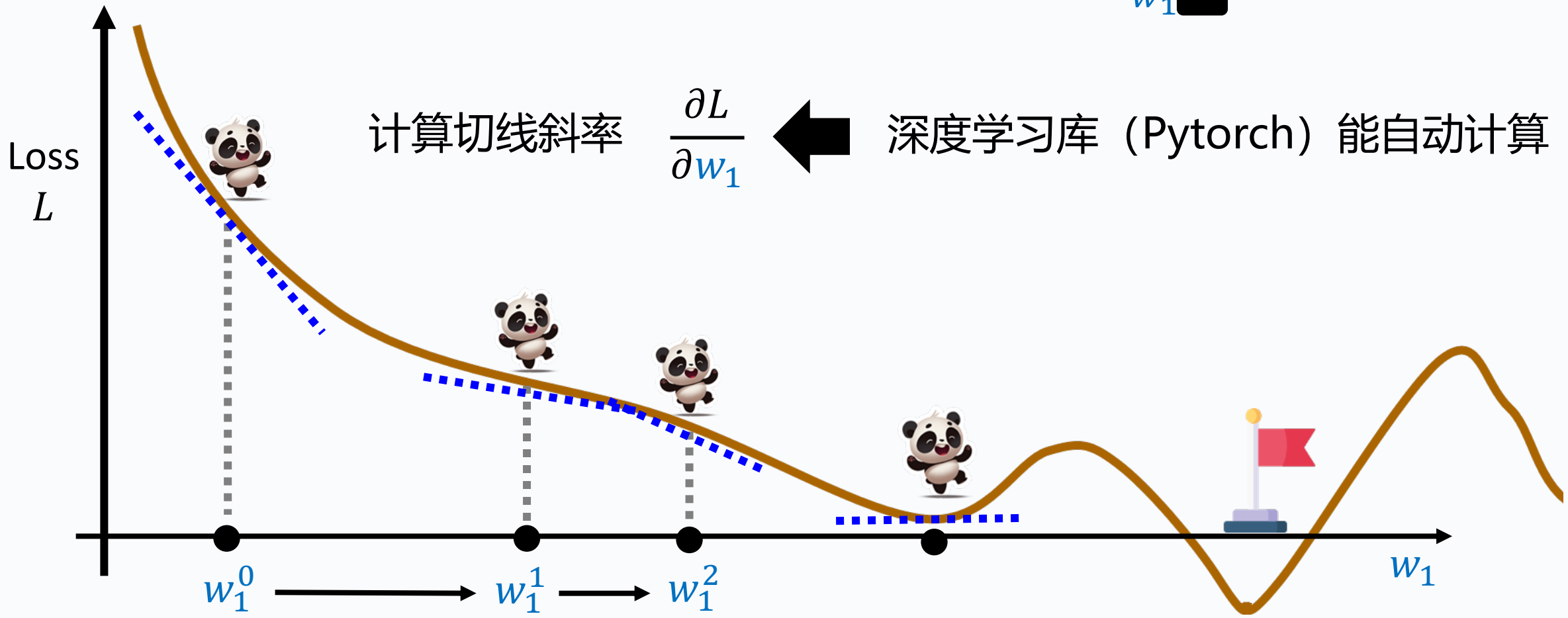
梯度下降 Gradient Descent

$$w_1^*, \blacksquare = \arg\min_{w_1, \blacksquare} L(w_1, \blacksquare)$$

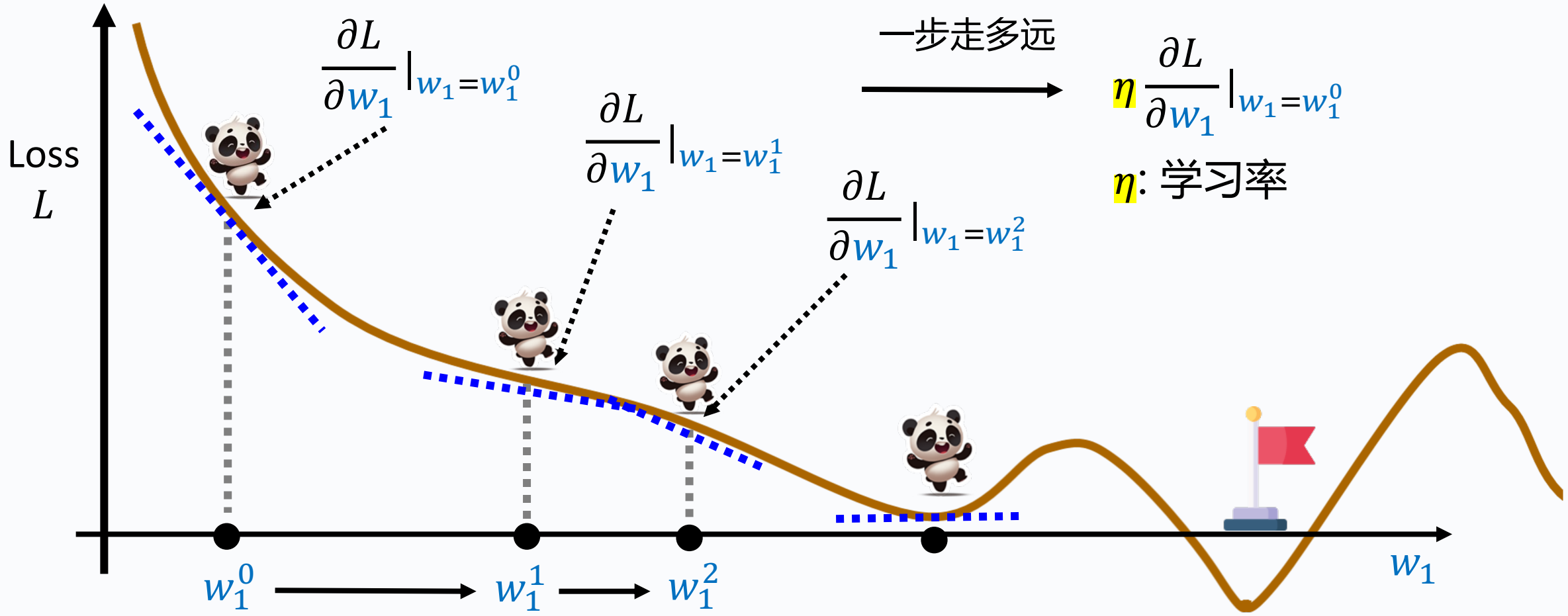


梯度下降 Gradient Descent

$$w_1^*, \blacksquare = arg \min_{w_1 \blacksquare} L(w_1, \blacksquare)$$



梯度下降 Gradient Descent



梯度下降 Gradient Descent

$$w_1^*, b^* = \arg\min_{w_1, b} L(w_1, b)$$

- (随机地) 选取初始值 w_1^0, b^0
- 计算

梯度

$$\frac{\partial L}{\partial w_1}\Big|_{w_1=w_1^0, b=b^0}$$

$$\frac{\partial L}{\partial b}\Big|_{w_1=w_1^0, b=b^0}$$

$$w_1^1 \leftarrow w_1^0 - \eta \frac{\partial L}{\partial w_1}\Big|_{w_1=w_1^0, b=b^0}$$

$$b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}\Big|_{w_1=w_1^0, b=b^0}$$

深度学习框架会帮我们自动计算（Autograd）

- 更新 w_1 和 b

梯度下降 Gradient Descent

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

- (随机地) 选取初始值 $\boldsymbol{\theta}^0$

梯度

$$\boldsymbol{g}^0 = \begin{bmatrix} \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \vdots \end{bmatrix} - \begin{bmatrix} \eta \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \eta \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\boldsymbol{g}^0 = \nabla L(\boldsymbol{\theta}^0)$$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$$

梯度下降 Gradient Descent

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

- (随机地) 选取初始值 $\boldsymbol{\theta}^0$
- 计算梯度 $\boldsymbol{g}^0 = \nabla L(\boldsymbol{\theta}^0)$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$$ ⬅ 1 次更新

- 计算梯度 $\boldsymbol{g}^1 = \nabla L(\boldsymbol{\theta}^1)$

$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^1$$

- 计算梯度 $\boldsymbol{g}^2 = \nabla L(\boldsymbol{\theta}^2)$

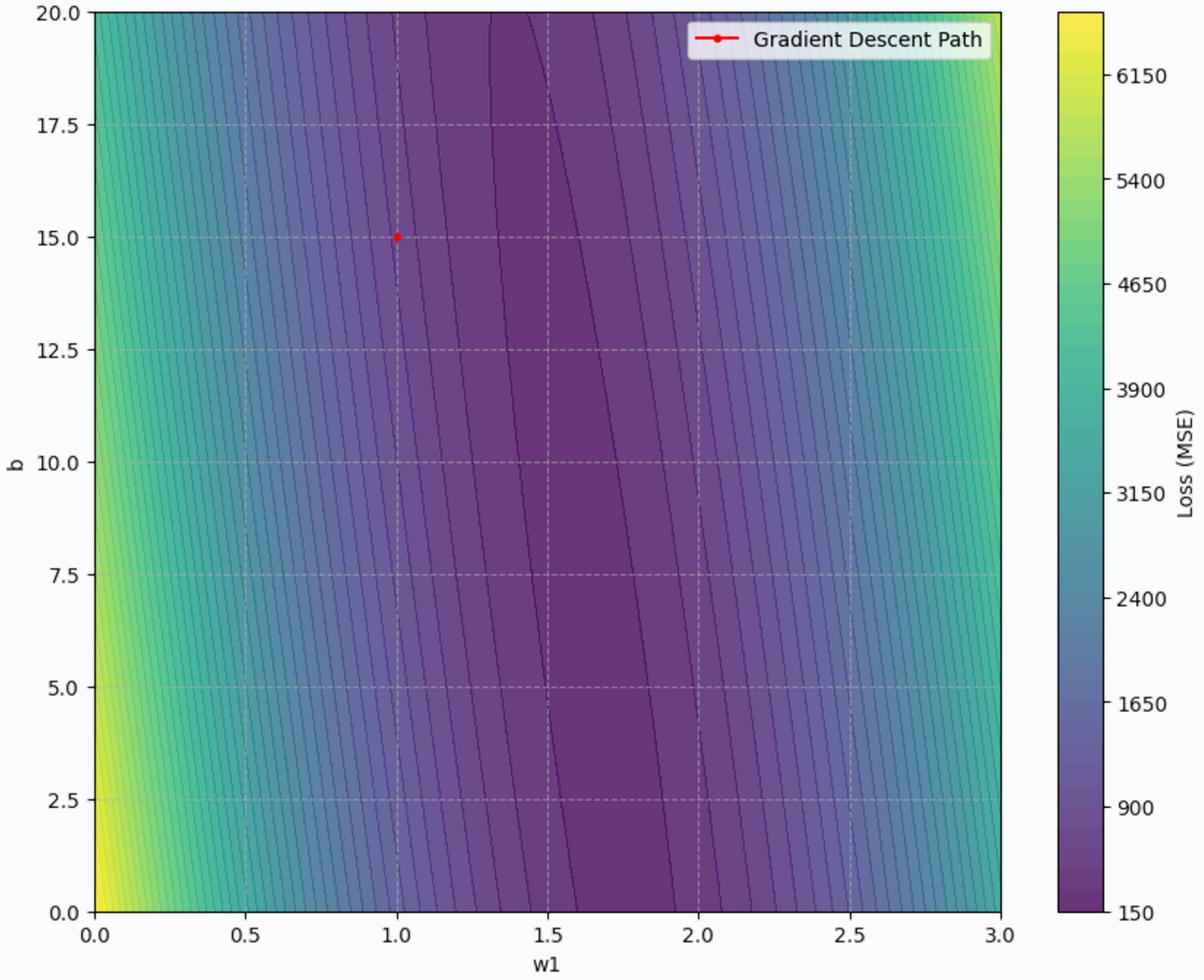
$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}^2$$

梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.001$

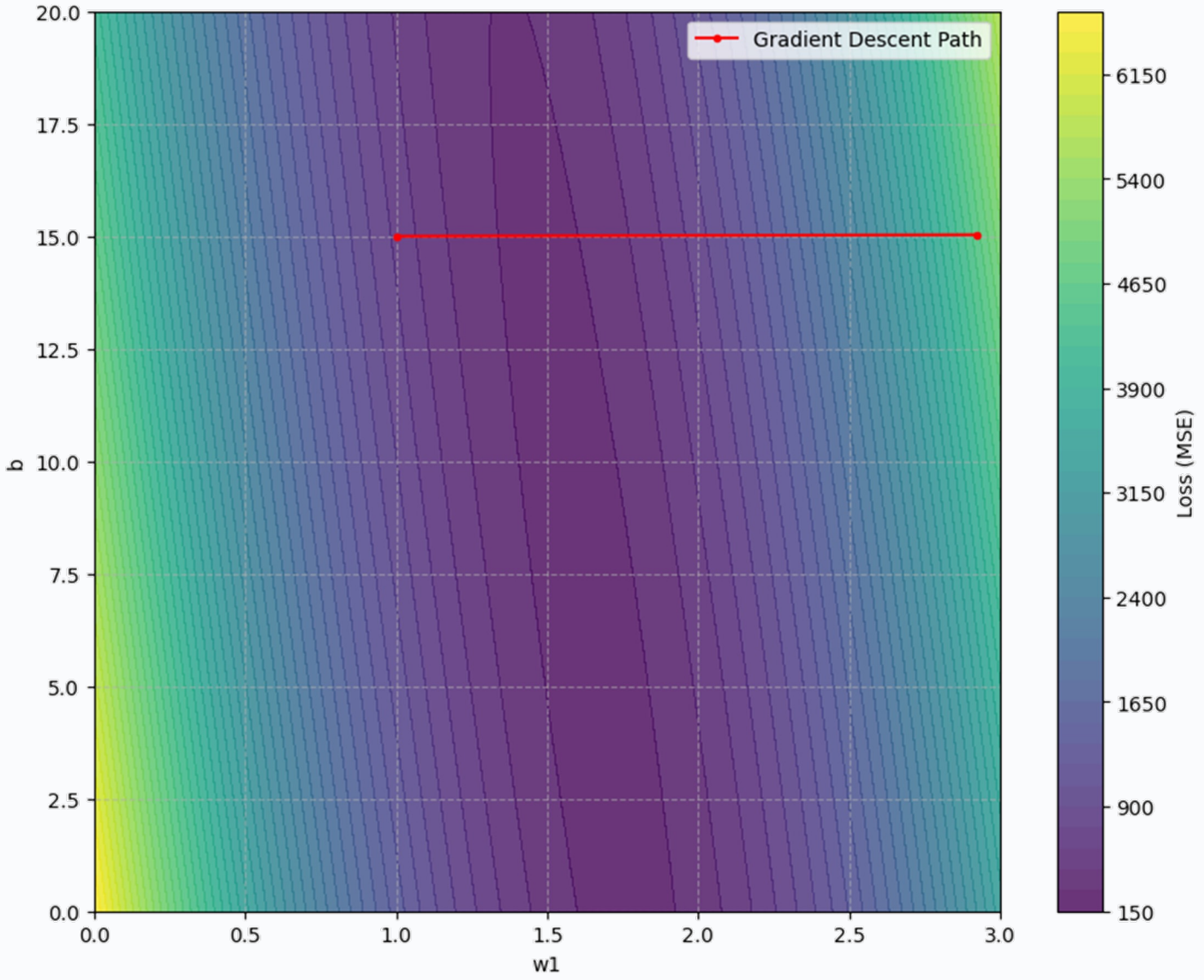


梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.001$

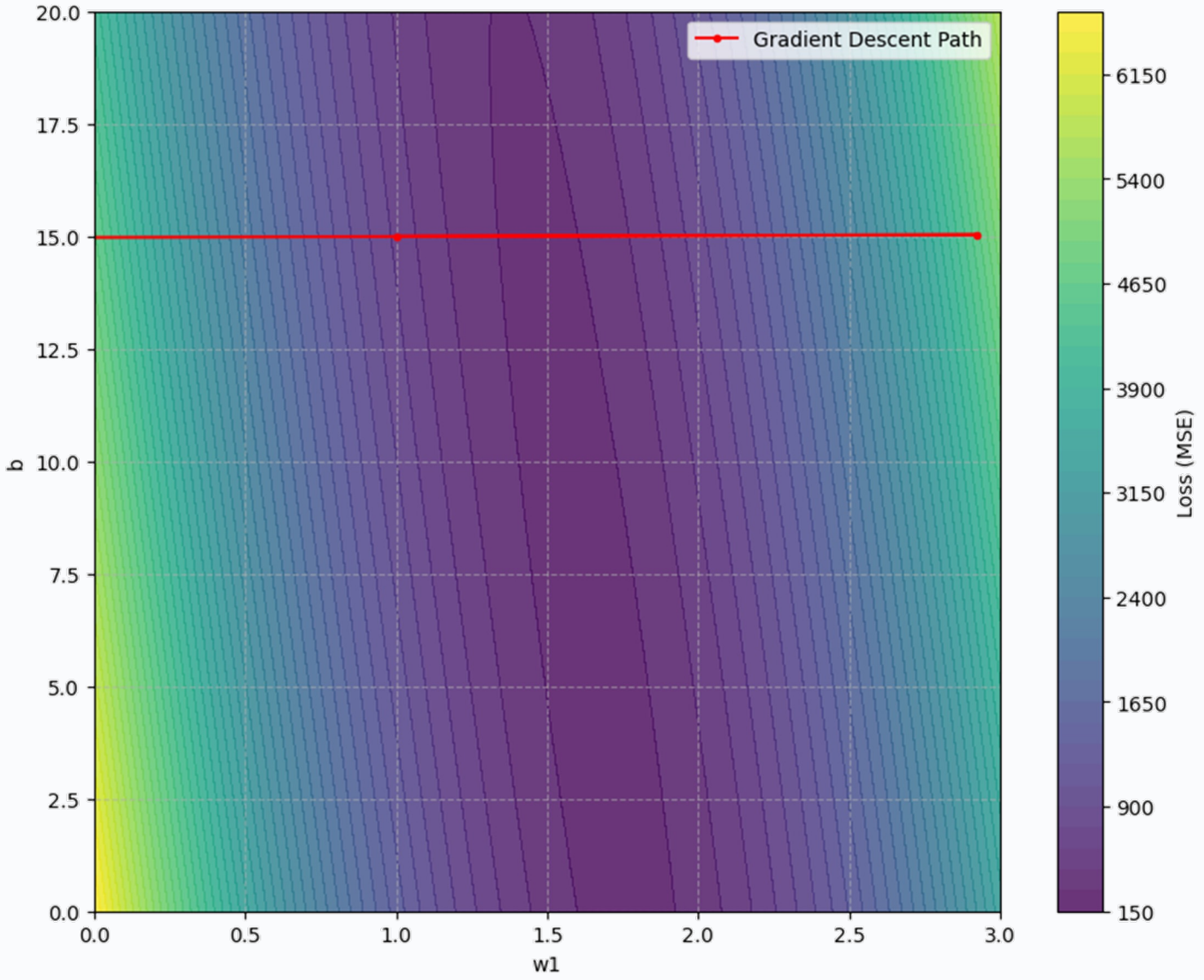


梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.001$

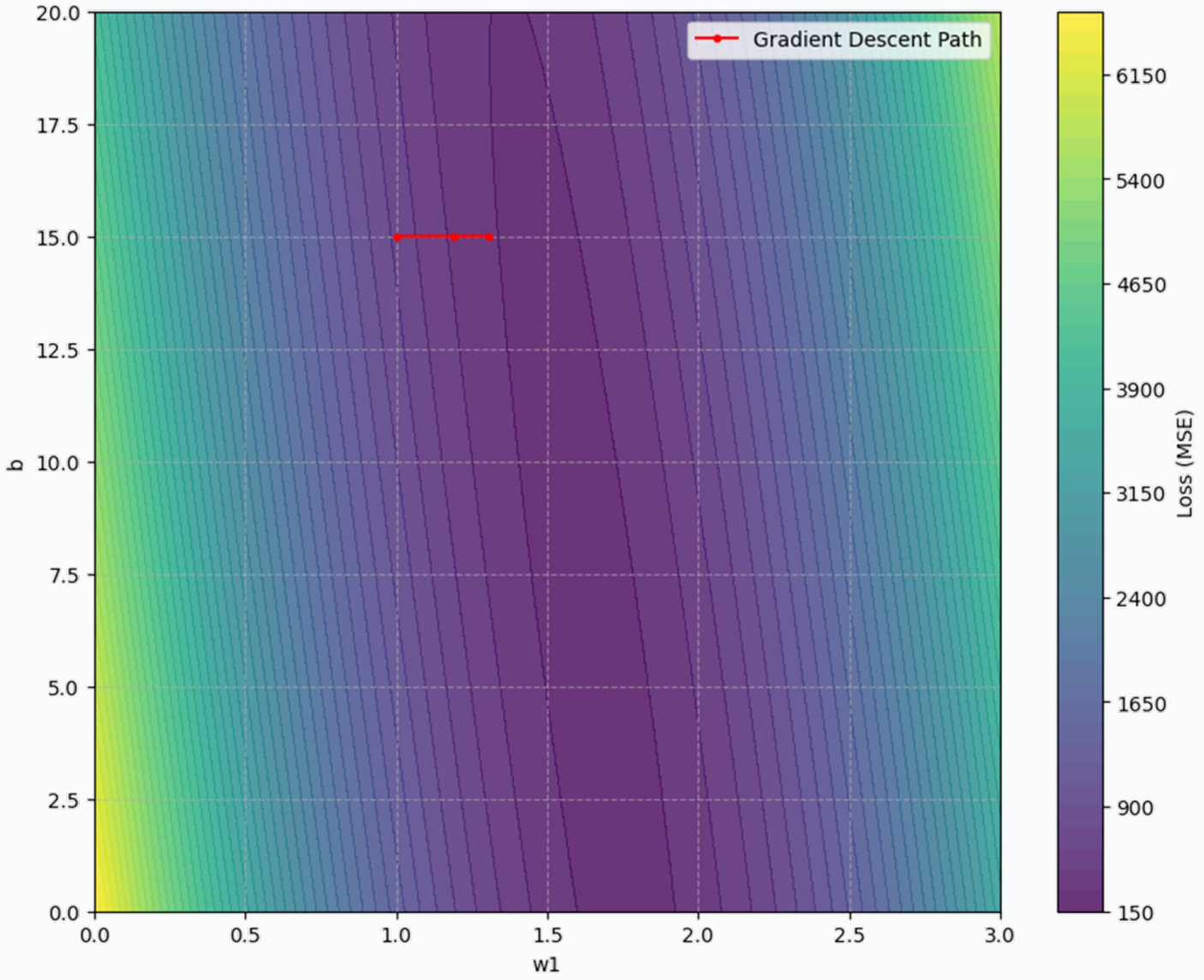


梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.0001$



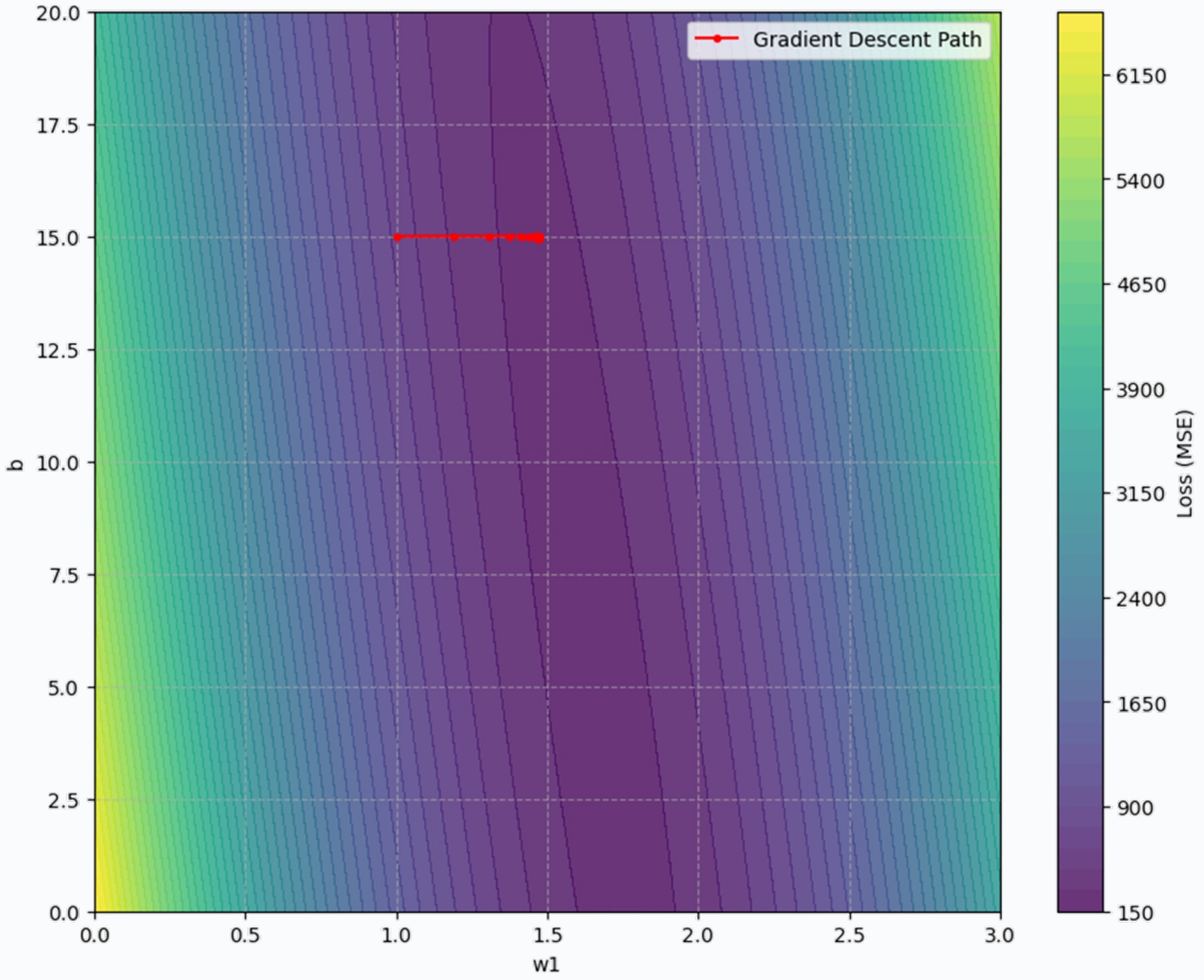
梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.0001$

Update 100 times



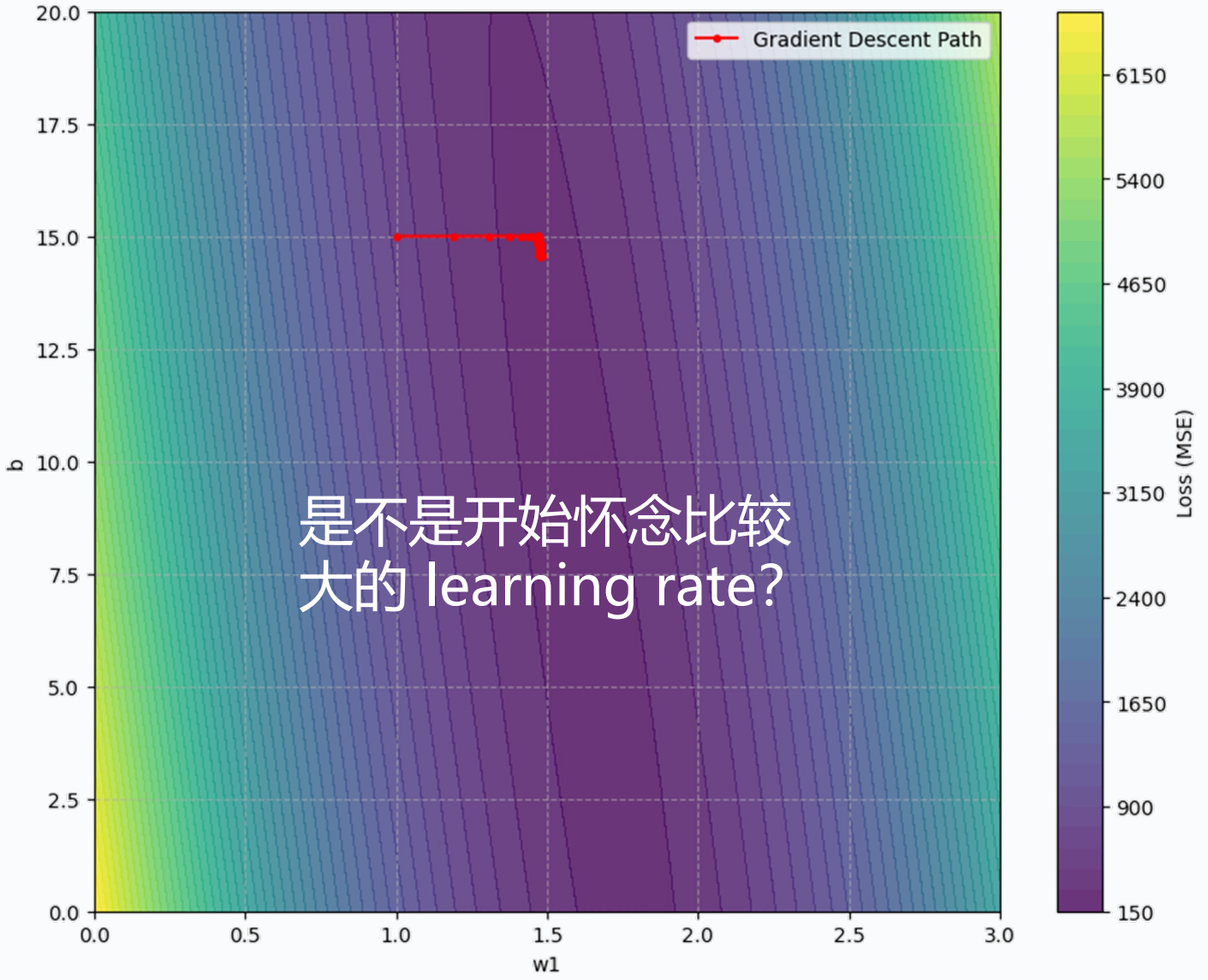
梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.0001$

Update 1,000 times



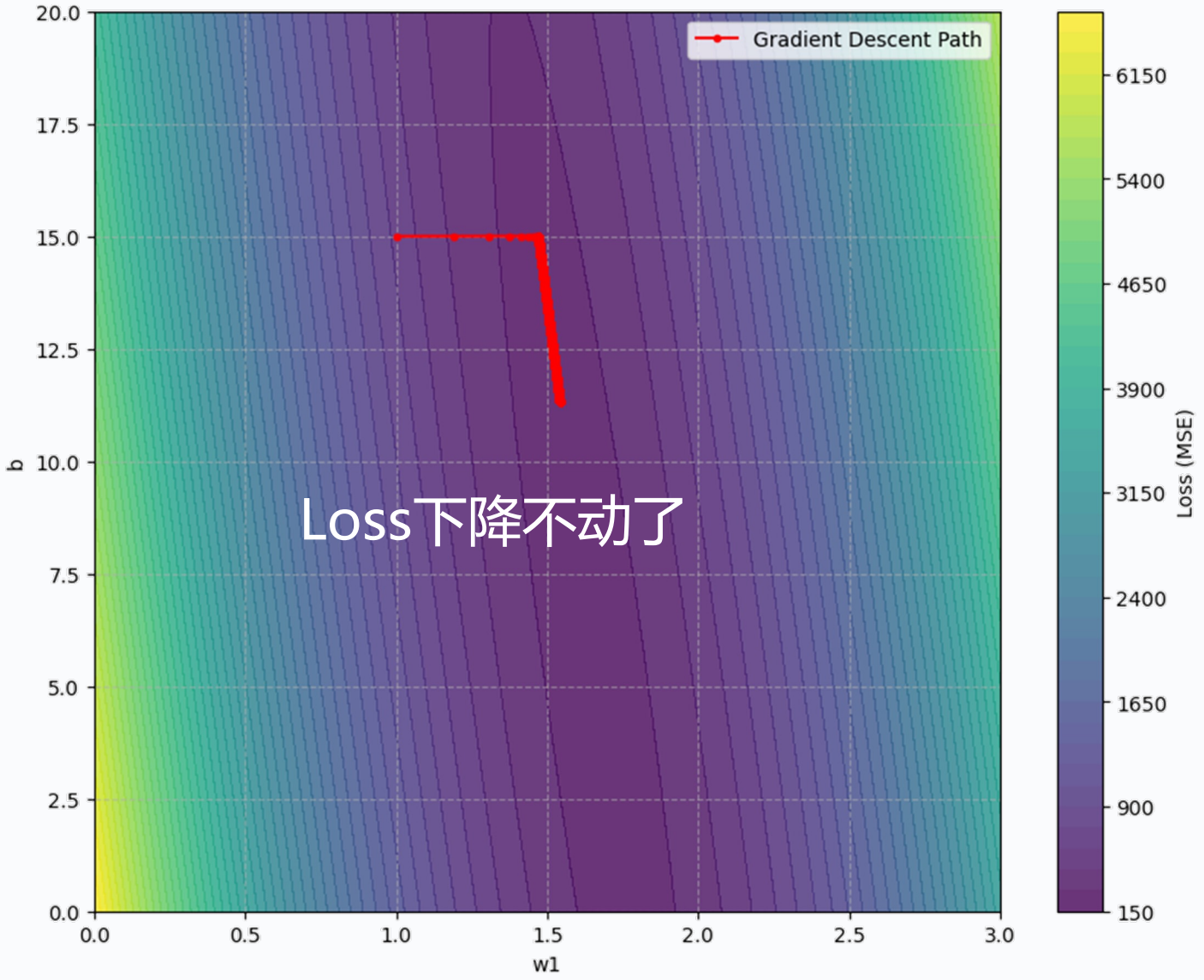
梯度下降 Gradient Descent

学习率的选择并不容易

$w_1^0 = 1.0, b^0 = 15.0$

$\eta = 0.0001$

Update 10,000 times



参数更新太慢

- (随机地) 选取初始值 $\boldsymbol{\theta}^0$
- 计算梯度 $\boldsymbol{g}^0 = \nabla L(\boldsymbol{\theta}^0)$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$$

- 计算梯度 $\boldsymbol{g}^1 = \nabla L(\boldsymbol{\theta}^1)$

$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^1$$

- 计算梯度 $\boldsymbol{g}^2 = \nabla L(\boldsymbol{\theta}^2)$

$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}^2$$

$$L = \frac{1}{N}\sum_{i=1}^{N} \cdots\cdots$$

如果训练数据集很大，要等很久才能更新一次参数

如何快点更新参数

- (随机地) 选取初始值 $\boldsymbol{\theta}^0$
- 计算梯度 $\boldsymbol{g}^0 = \nabla L(\boldsymbol{\theta}^0)$

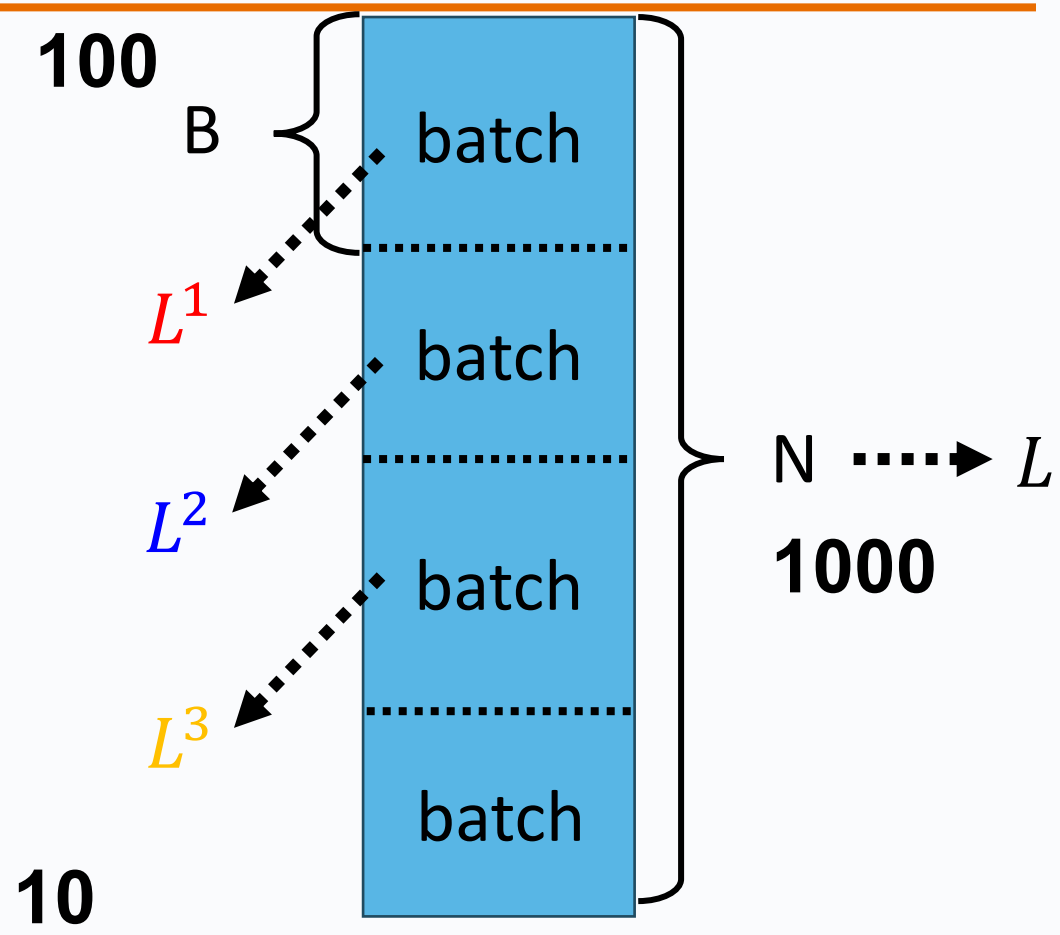
$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$$

- 计算梯度 $\boldsymbol{g}^1 = \nabla L(\boldsymbol{\theta}^1)$

$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^1$$

- 计算梯度 $\boldsymbol{g}^2 = \nabla L(\boldsymbol{\theta}^2)$

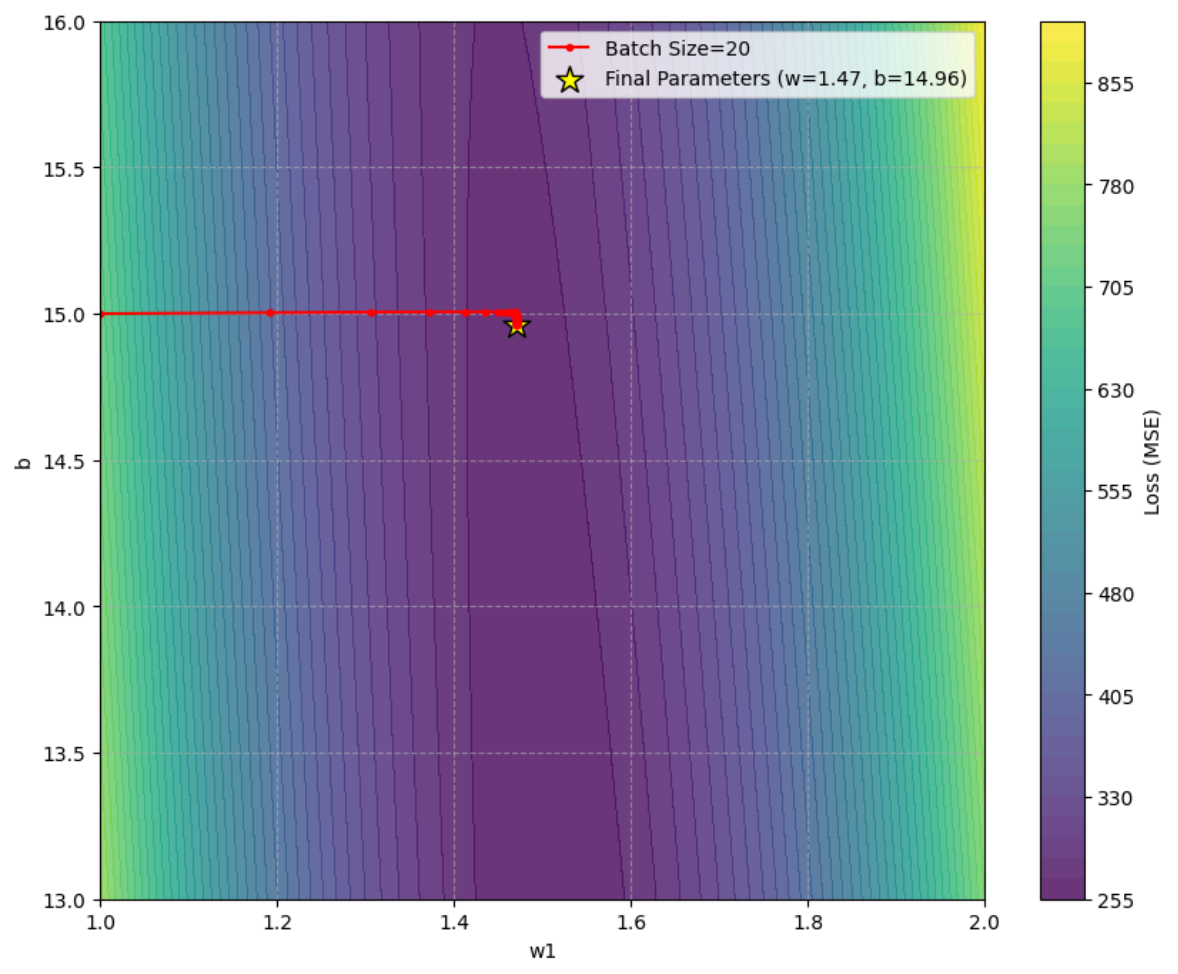
$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}^2$$



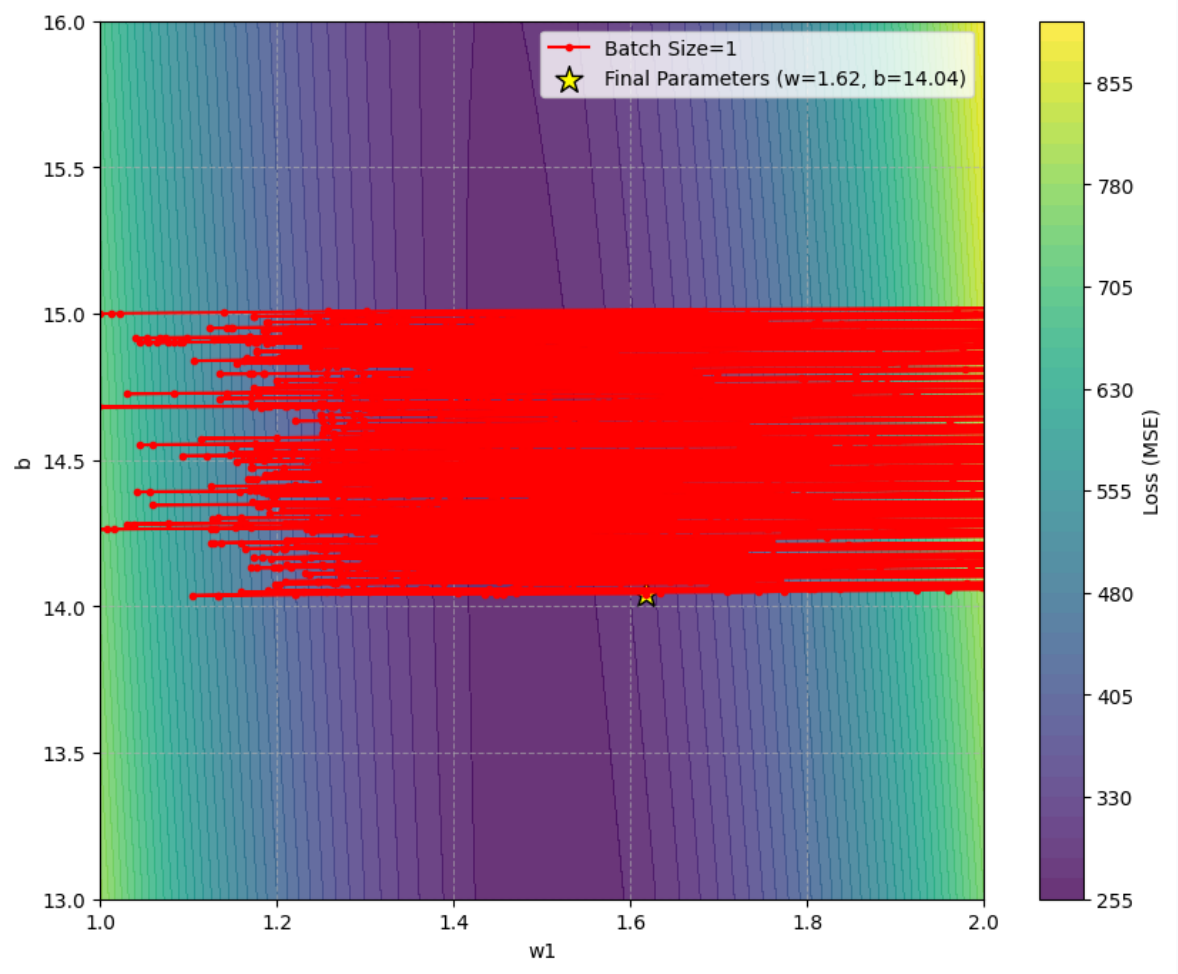
更新 N/B 次参数
每次叫做一个iteration

1 **Epoch** = 看过数据集的所有样本

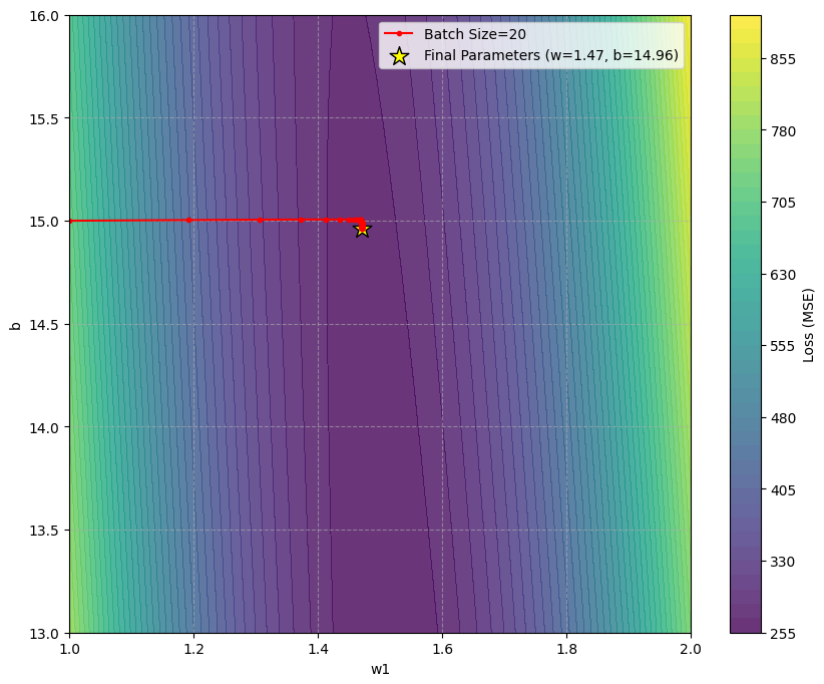
epochs = 100



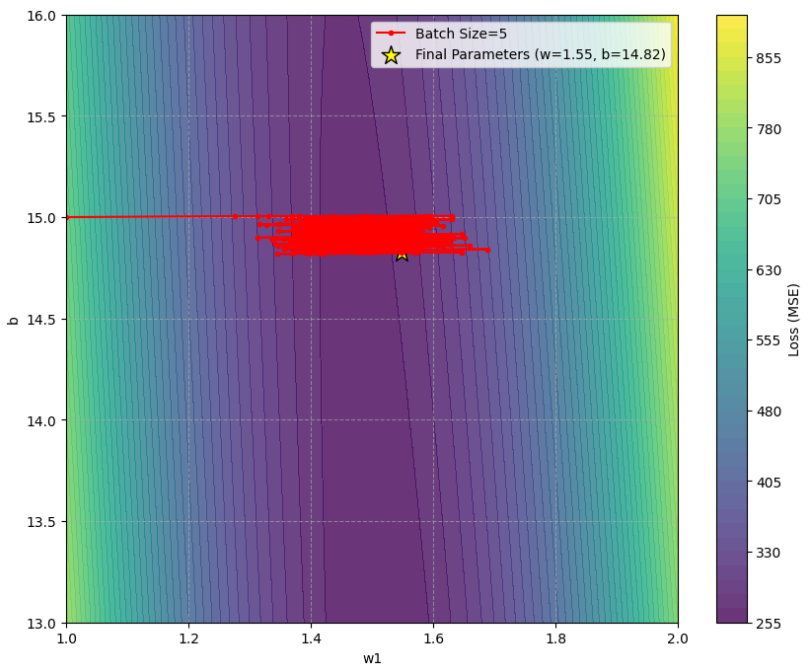
Batch size = all training data
(Full Batch)



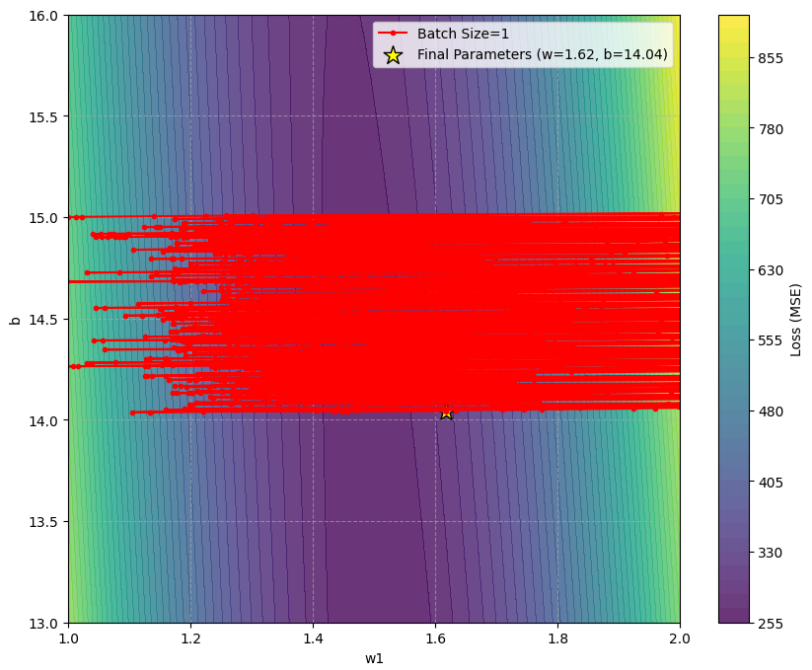
Batch size = 1
(Stochastic Gradient Descent, SGD)



Batch size
= all training data



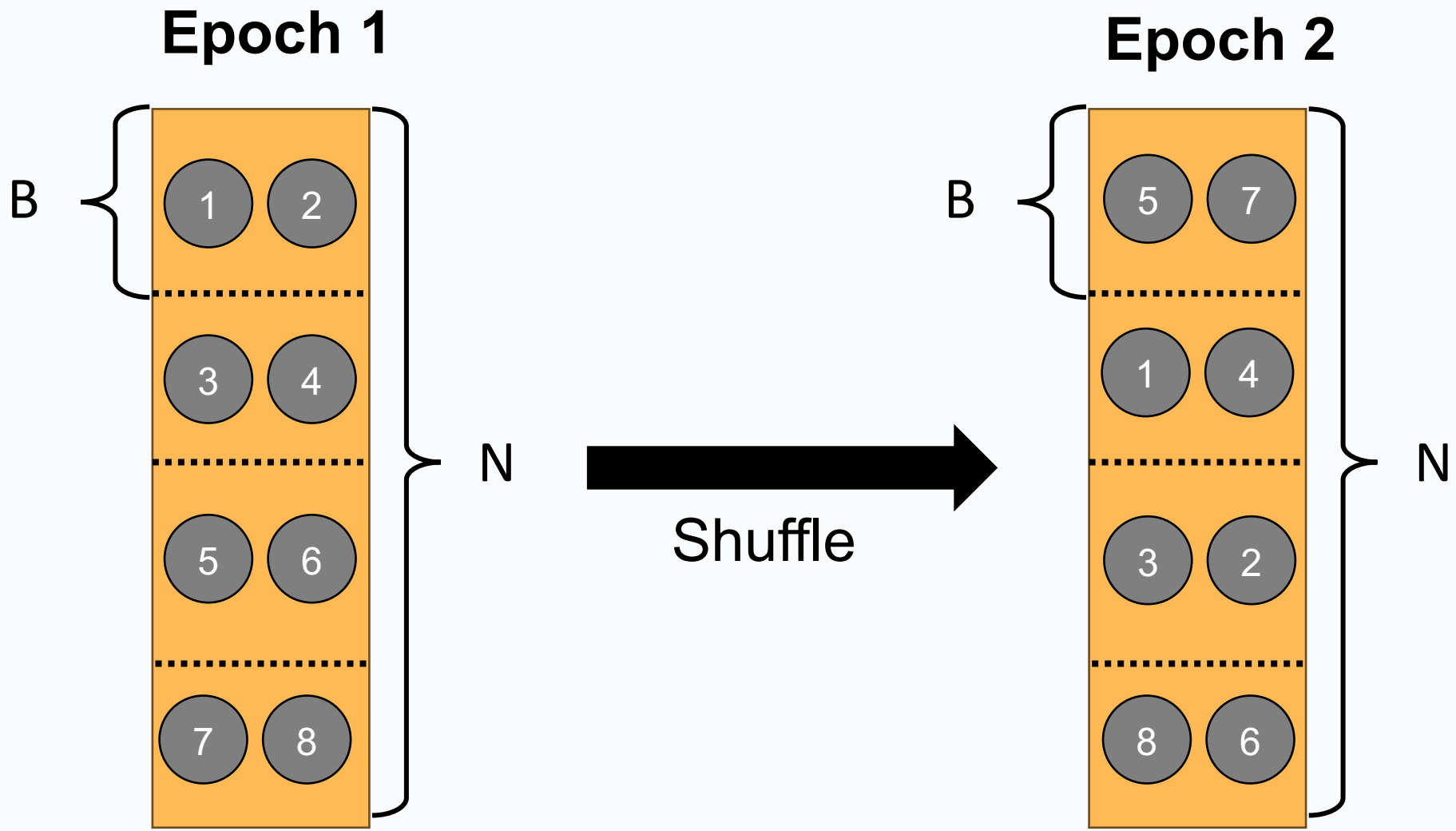
Batch size
= 5



Batch size
= 1

又多了一个可以调的超参数

Shuffle



1 我有什么选择

2 什么样的f更好

3 选一个最好的

$$y = w_1 x_1 + b$$

$$y = 1.67 x_1 + 4.85$$

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2$$

$$L(w_1^*, b^*) = 240$$

$$w_1^*, b^* = \arg\min_{w_1, b} L(w_1, b)$$

$$w_1^* = 1.67, b^* = 4.85$$

测试 (Testing)

1 我有什么选择

2 什么样的f更好

3 选一个最好的

$$y = w_1 x_1 + b$$

$$y = 1.67 x_1 + 4.85$$

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - \hat{y}^i\right)^2$$

$$L(w_1^*, b^*) = 240$$

$$w_1^*, b^* = \arg\min_{w_1, b} L(w_1, b)$$

$$w_1^* = 1.67, b^* = 4.85$$

验证 (Validation)

测试 (Testing)

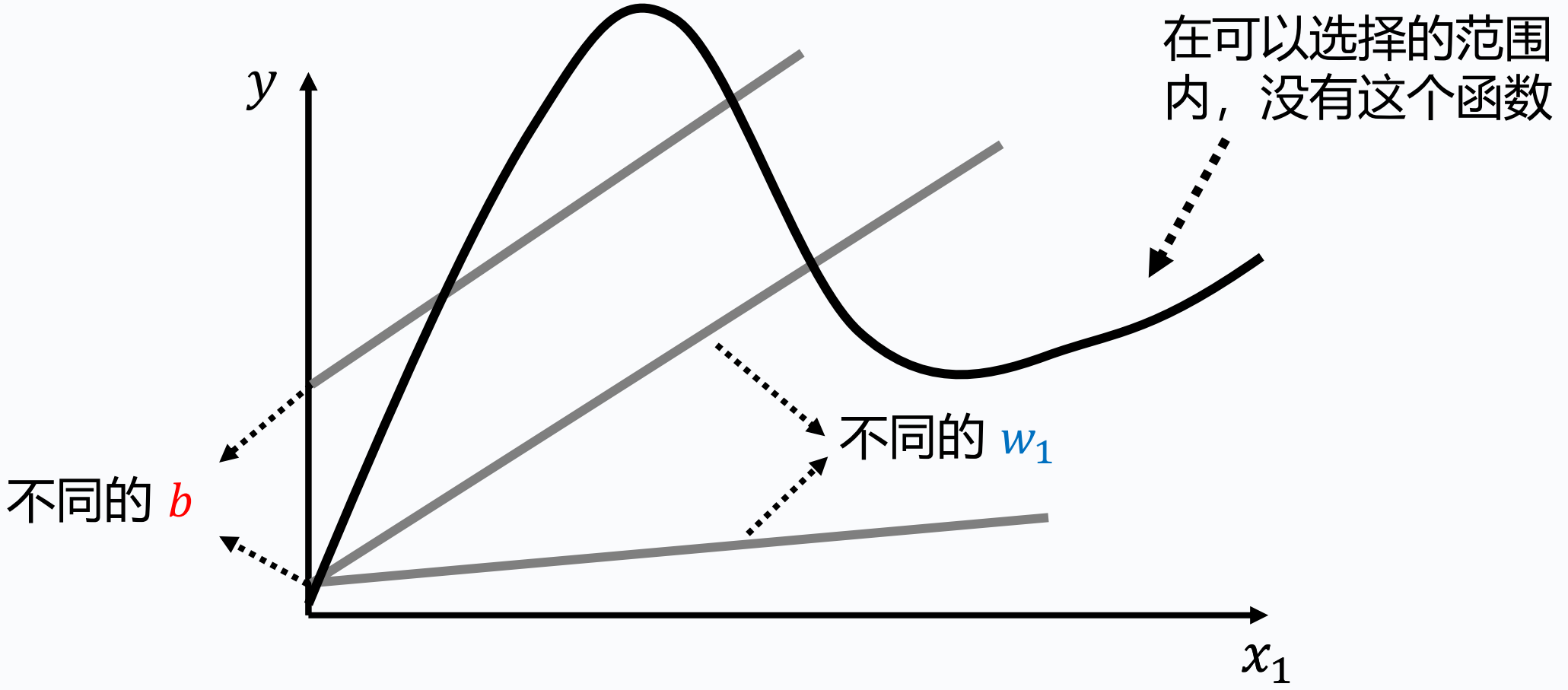
如果验证的结果不好

问题可能出在哪里呢？

1. 我有什么选择
2. 什么样的f更好
3. 选一个最好的

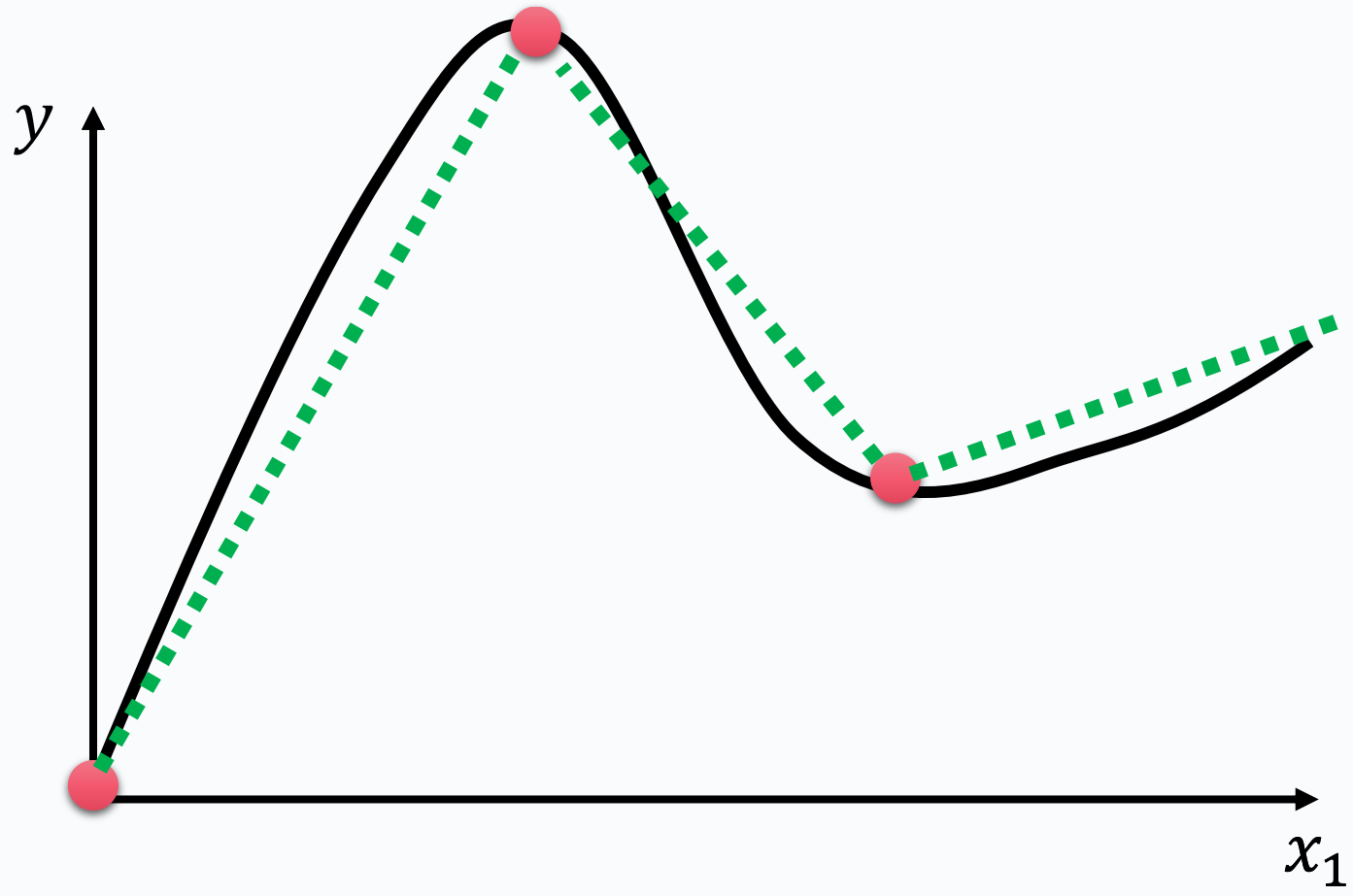
给f更多可能性

$$y = w_1 x_1 + b$$



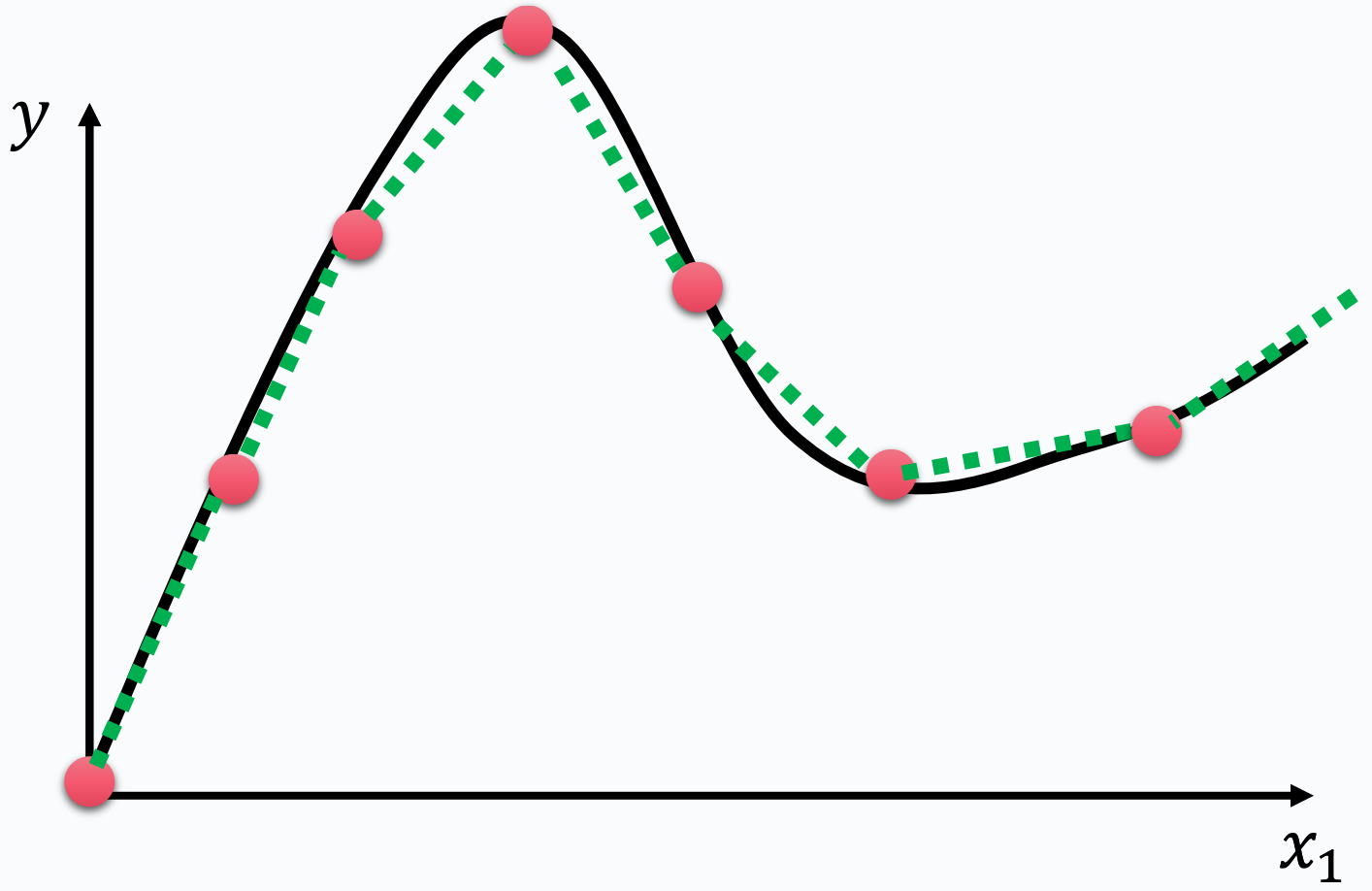
划定一个更大的函数范围

分段线性函数

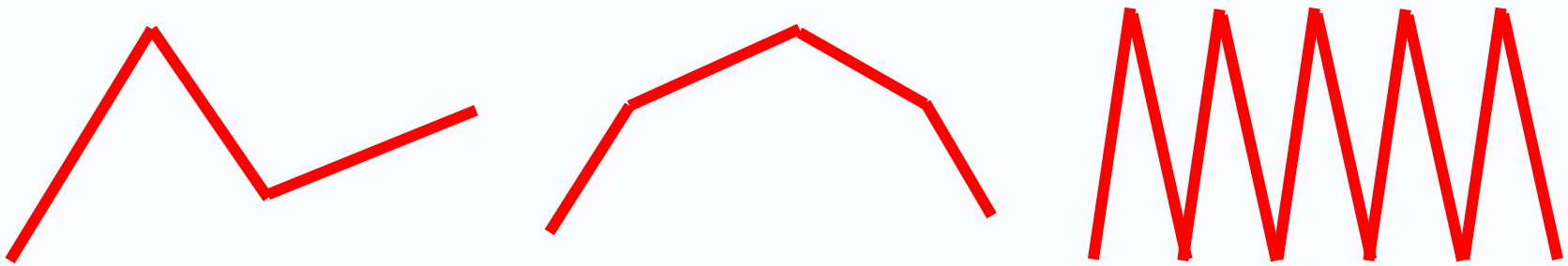


划定一个更大的函数范围

分段线性函数

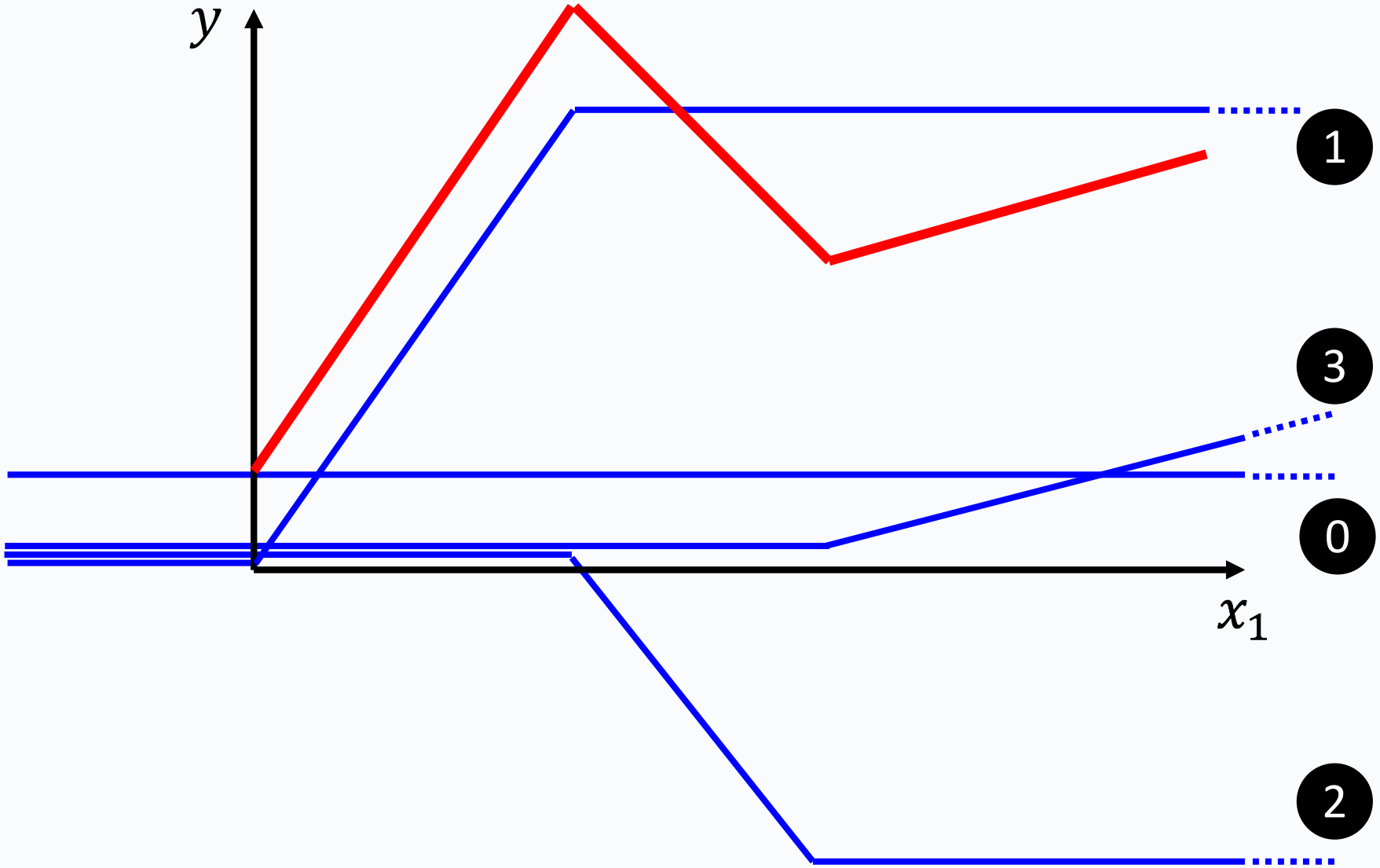


复杂的函数=许多  的组合

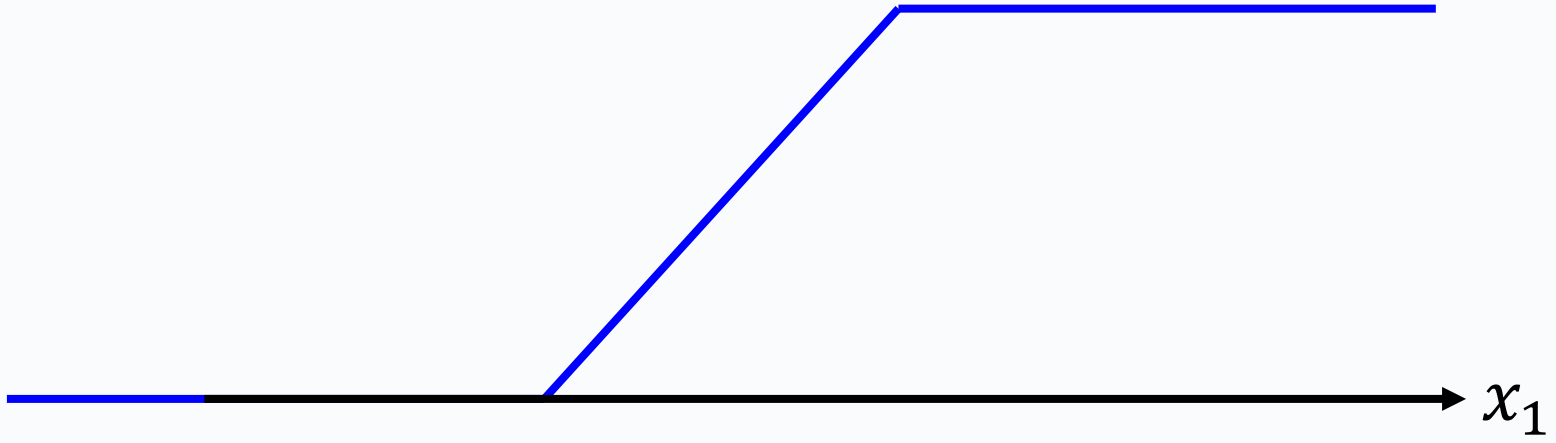


更多的分段，需要更多的 

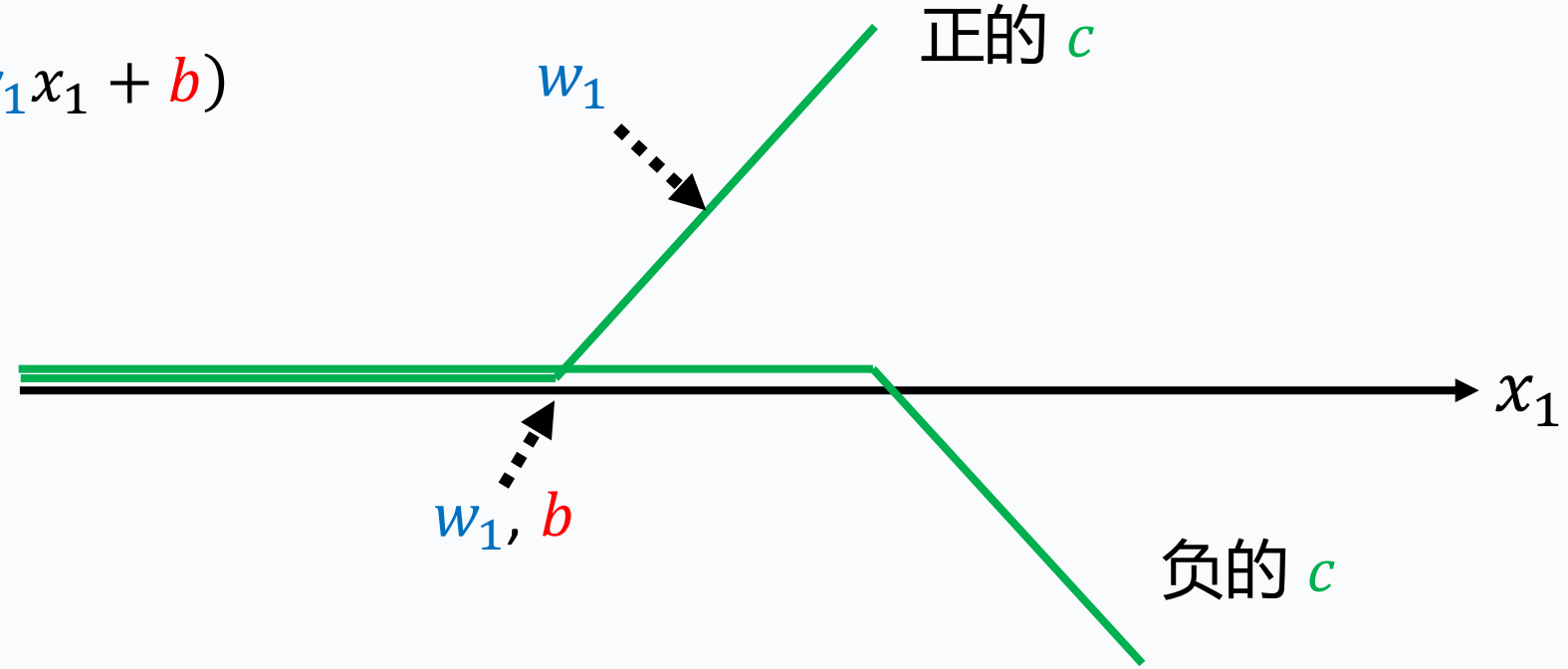
红色曲线 = 常数 + 许多  的组合



如何表示这个 Z 字曲线



$c \max(0, w_1 x_1 + b)$



任意复杂的函数 ≈ 分段线性的函数

= 常数 + 许多的

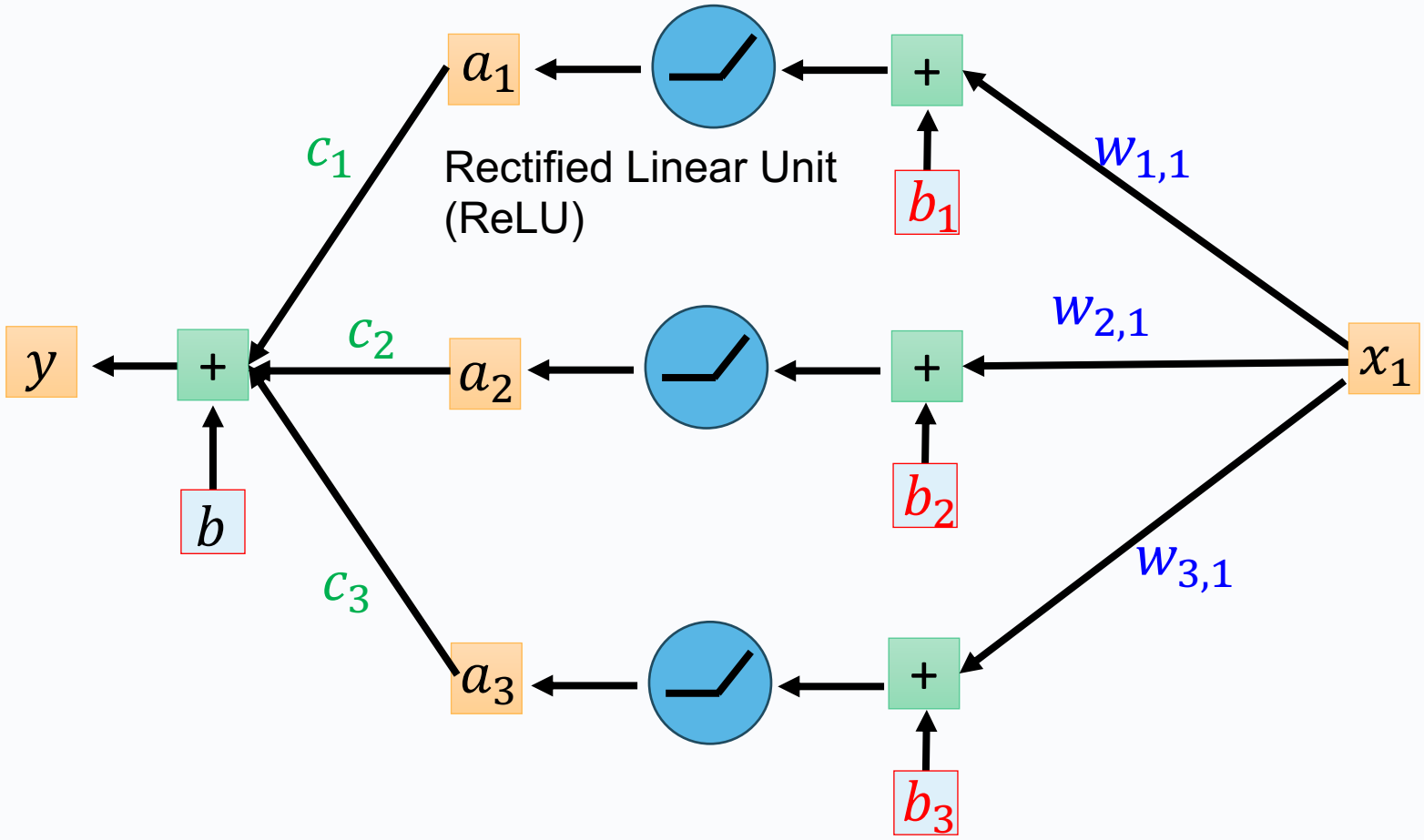
= 常数 + 许多多的

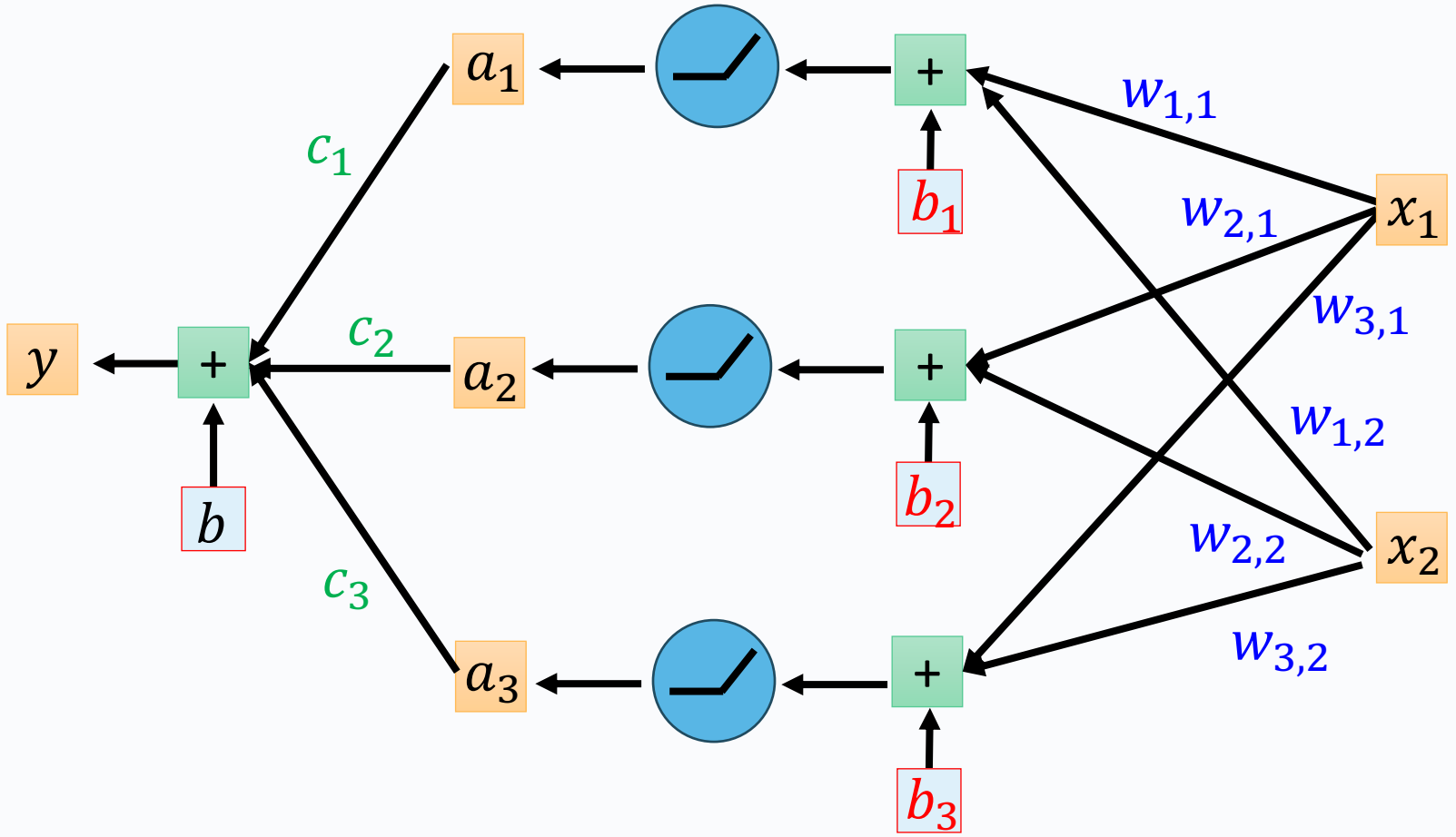
$c\max(0, w_1 x_1 + b)$

H

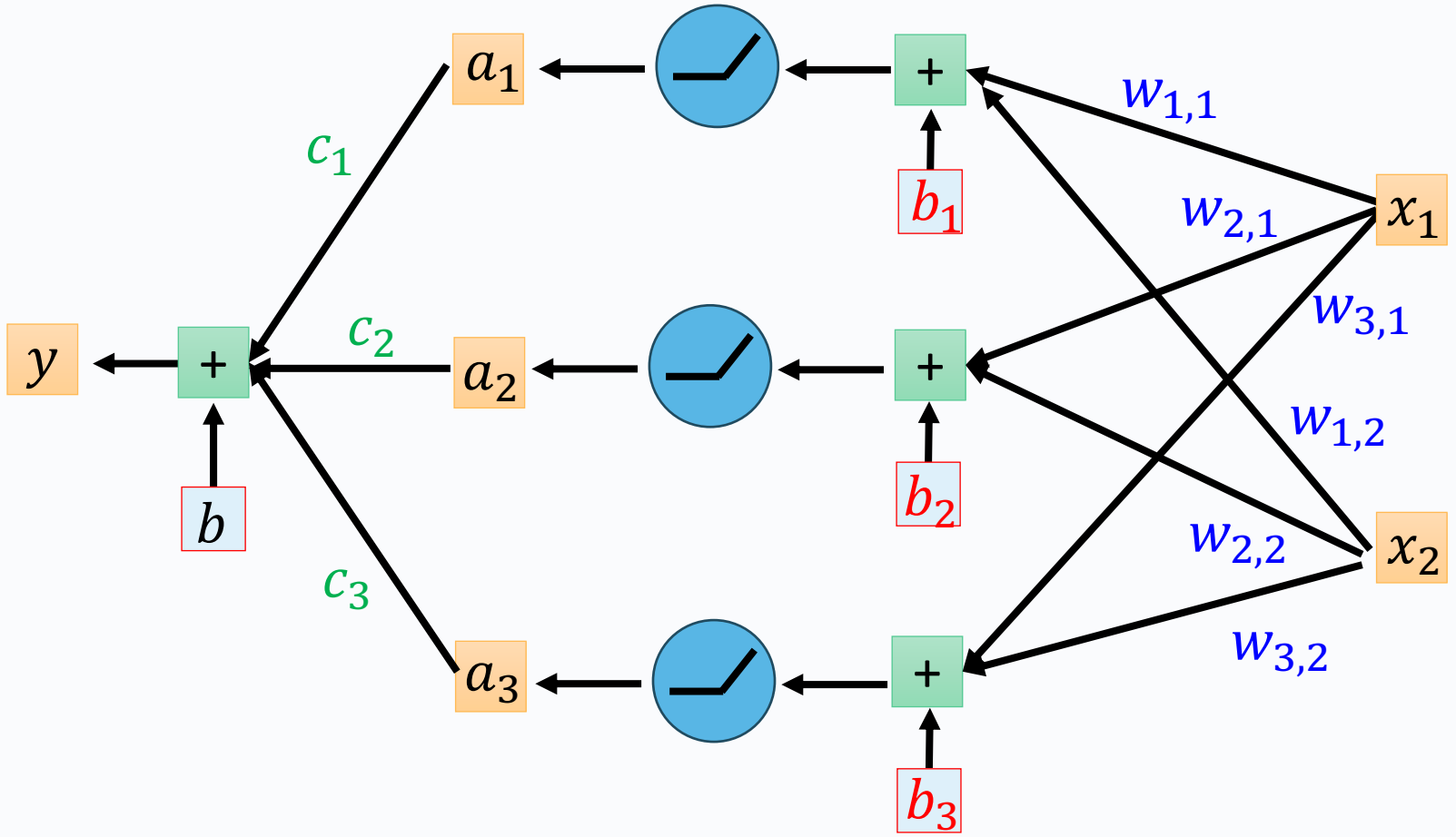
$$y = b + \sum_{i=1}^{H} c_i \max\left(0, w_{i,1} x_1 + b_i\right)$$

$$y = b + \sum_{i=1}^{H} c_i \underbrace{\max(0, w_{i,1} x_1 + b_i)}_{a_i}$$



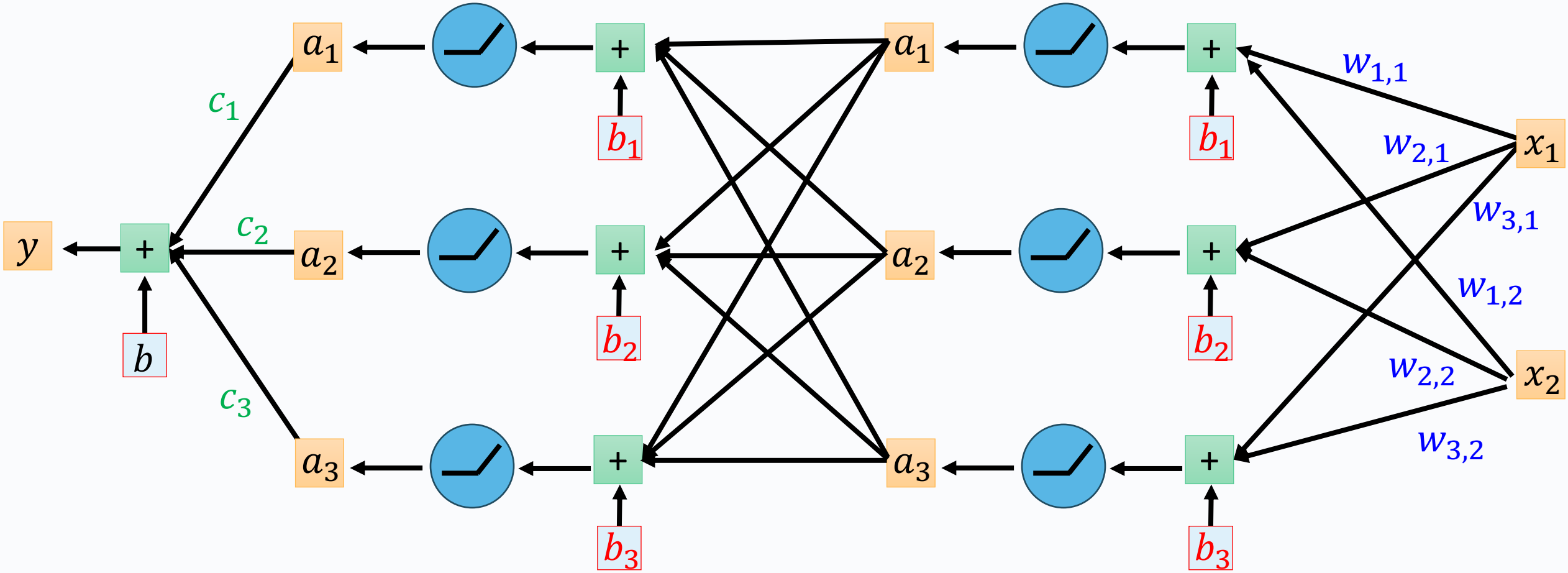


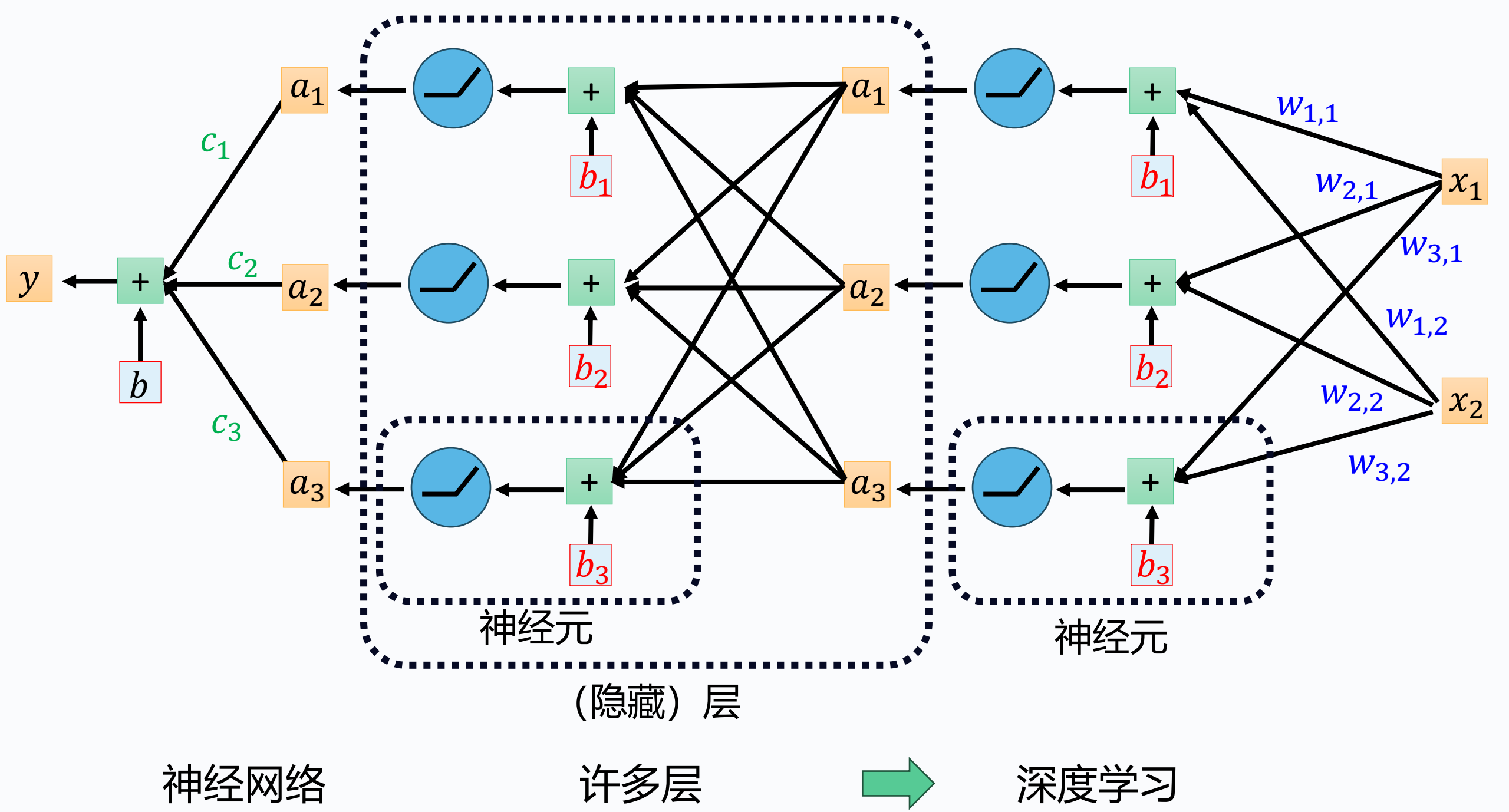
$$y = b + \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)$$



$$y = b + \boldsymbol{c}^T \boldsymbol{a} \qquad \boldsymbol{a} = \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

$$\boldsymbol{a}' = \sigma\left(\boldsymbol{b}' + W'\boldsymbol{a}\right) \qquad \boldsymbol{a} = \sigma\left(\boldsymbol{b} + W\boldsymbol{x}\right)$$



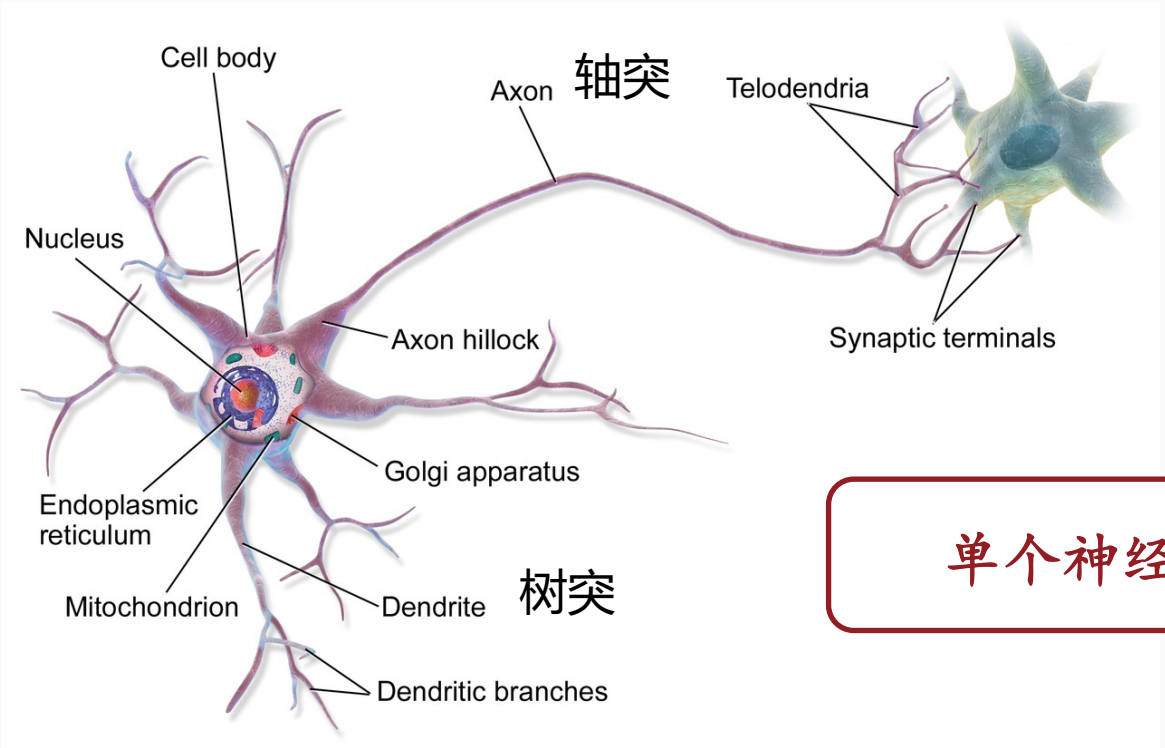


神经网络　　许多层　　深度学习

生物神经元

人脑神经系统有860亿个神经元

每个神经元有上千个突触与其他神经元相连

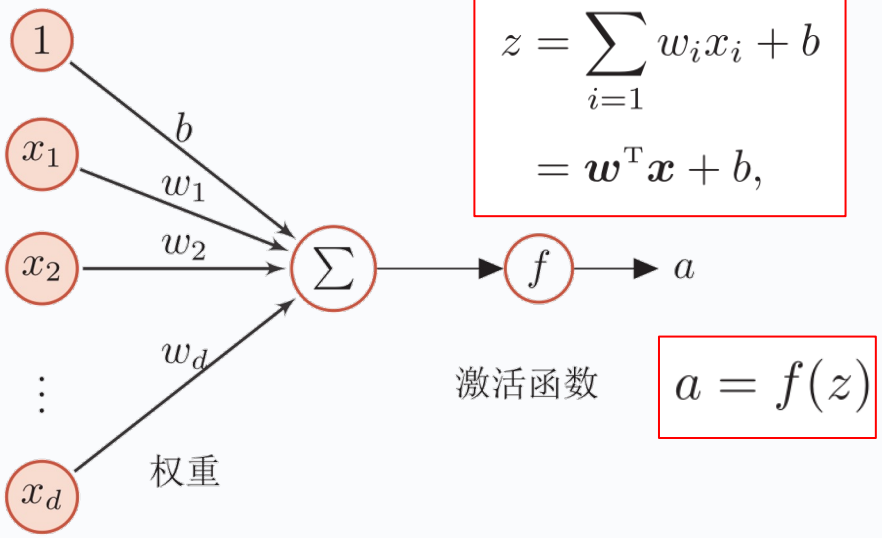


单个神经细胞只有两种状态：兴奋 / 抑制

人工神经网络

M-P神经元模型（McCulloch and Pitts, 1943）

- 神经元接收到来自其他d**个神经元**传递过来的**输入信号**
- 这些输入信号通过带**权重的连接**进行传递
- 神经元接收到的总输入值将与神经元的**阈值**（bias）进行比较。

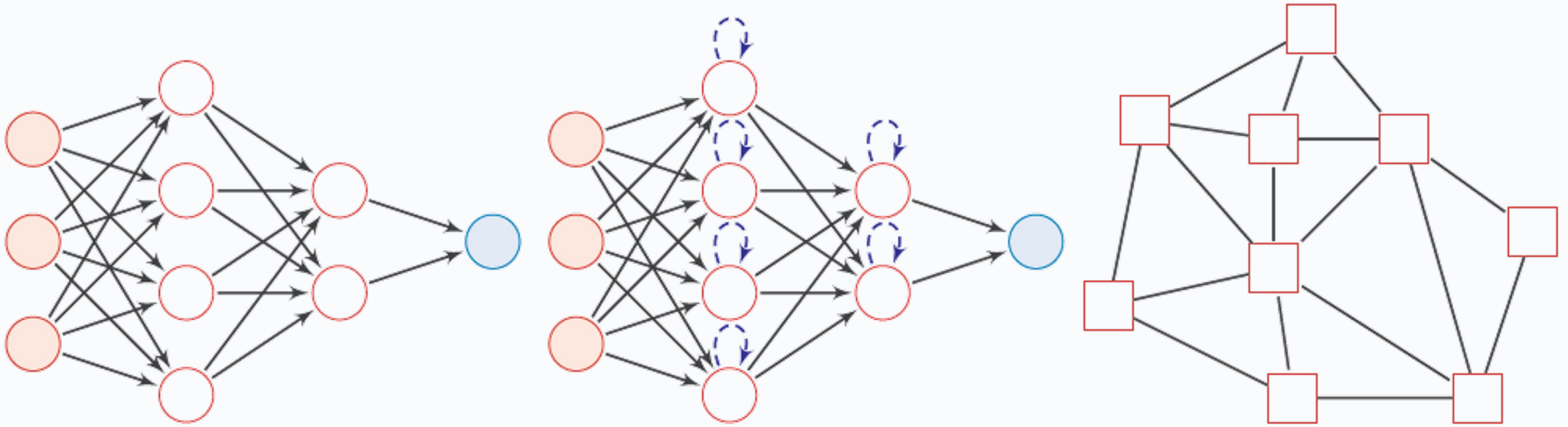


$$z = \sum_{i=1}^{d} w_i x_i + b$$
$$= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b,$$

$$a = f(z)$$

输出为0或1

各种各样的连接方式

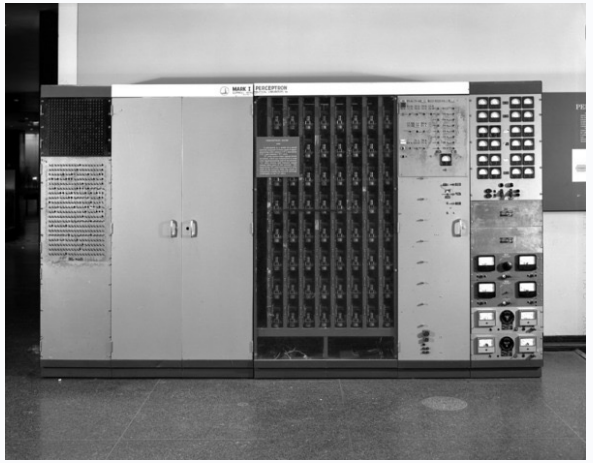


(a) 前馈网络　(b) 记忆网络　(c) 图网络

神经网络的早期发展

感知机（Perceptron）

- 1958, Rosenblatt
- 用于图像识别（▲■⭕）
- 400个光电管连接到1个神经元上
- 连接权重通过滑动变阻器控制



Mark I 感知机

- 1958，纽约时报：机器即将可以走路、说话、看见、有自我意识。
- 1966，Summer vision project： “奋斗一个夏天，解决模式识别”

失败的尝试 😢

AI的第一个寒冬：1974-1980

感知机无法求解XOR问题

感知机的训练非常困难

AI的实际表现很糟糕

- 最杰出的AI程序也只能解决它们尝试解决的问题中最简单的一部分，也就是说所有的AI程序都只是“玩具”。
- 各国政府表示对AI的研究进展失望，削减经费。

1965年，跳棋程序“MAC Hack VI”
始终无法战胜美国全国冠军

1973年 美国副总统
对计算机翻译模型进展表示失望

神经网络的复兴：1990s

Geoffery E. Hinton

- 1986年，Hinton等人将引入反向传播算法引入到多层感知机。（Nature）
  - 将纠错的运算量下降到只和**神经元数目成正比**。
- 贡献: Boltzman machine, mixtures of experts, AlexNet (2012)
- 2018图灵奖；2024年诺贝尔奖

一些鸡汤

- 1978-2012，经历两次AI寒冬
- 许多想法在几十年前就被提出了
- Research最重要的是什么?

第二次寒冬

超过3个隐藏层的神经网络通常不能训练出好的结果。

1989，一个隐藏层，就可以拟合任何函数。（通用近似定理）

在20世纪90年代中期，统计学习理论和以支持向量机（SVM）为代表的机器学习模型开始兴起。

- 相比之下，神经网络的理论基础不清晰、优化困难、可解释性差

深度学习的春天：2010s

GPU

- 2009年，斯坦福大学的Rajat Raina和吴恩达合作发表论文：*Large-scale Deep Unsupervised Learning using Graphic Processors* (ICML 09)
  - 使用GPU运行速度，比传统双核CPU快70倍。
- GPU擅长对**大批量数据进行并行处理**

数据

- ImageNet: A Large hierarchical image database发布，包含320万张图像。