



温州大學  
WENZHOU UNIVERSITY

# 深度学习-优化算法

黄海广 副教授

2023年04月

- 01** 小批量梯度下降
- 02** 优化算法
- 03** 超参数调整和**BatchNorm**
- 04** **Softmax**

# 1.小批量梯度下降

3

**01 小批量梯度下降**

**02 优化算法**

**03 超参数调整和BatchNorm**

**04 Softmax**

# 小批量梯度下降

4

## 小批量梯度下降 (Mini-Batch Gradient Descent)

梯度下降的每一步中，用到了一定批量的训练样本

每计算常数  $b$  次训练实例，便更新一次参数  $w$

### 参数更新

$$w_j := w_j - \alpha \frac{1}{b} \sum_{k=i}^{i+b-1} (h(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

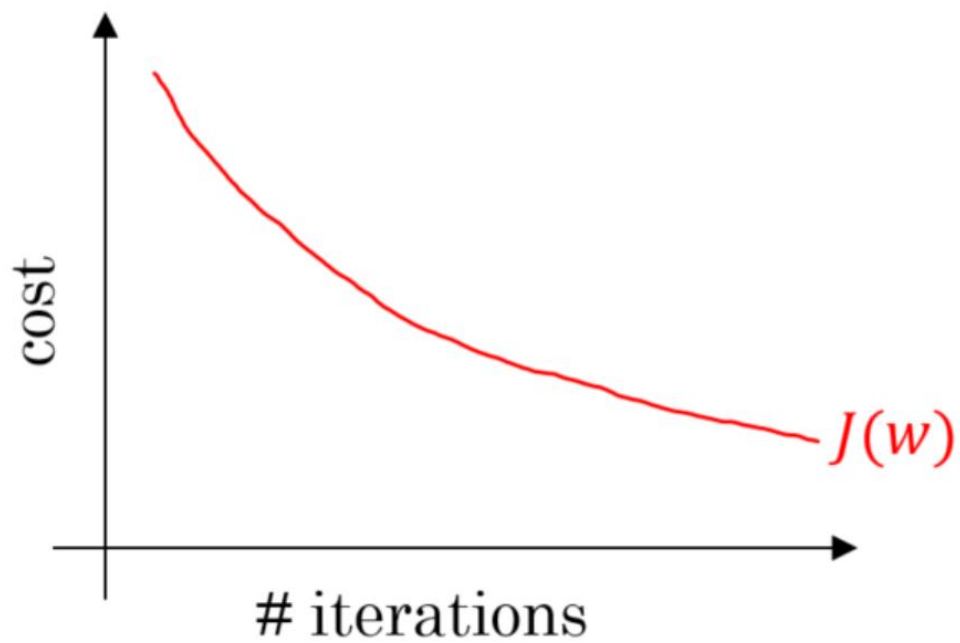
(同步更新  $w_j$  ,  $(j=0,1,...,n)$  )

$b=1$  (随机梯度下降,SGD)  
 $b=m$  (批量梯度下降,BGD)  
 $b=batch\_size$ , 通常是2的指数倍, 常见有32,64,128等。  
(小批量梯度下降,MBGD)

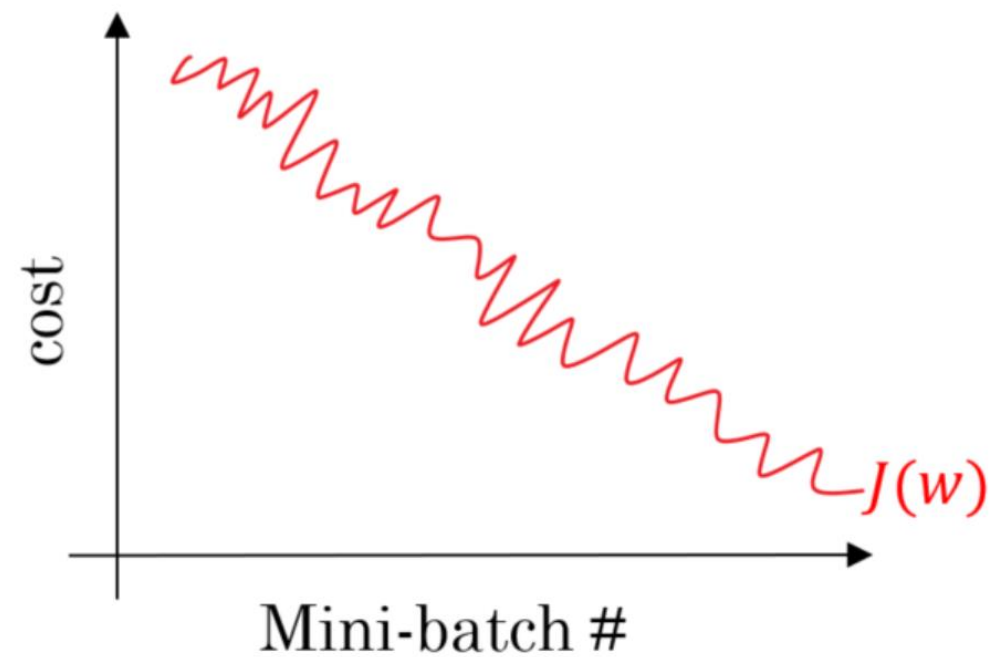
# 小批量梯度下降

5

Batch gradient descent



Mini-batch gradient descent



## 2.优化算法

6

**01** 小批量梯度下降

**02** 优化算法

**03** 超参数调整和BatchNorm

**04** Softmax

# 伦敦温度的例子

7

$$\theta_1 = 40^\circ\text{F}$$

$$\theta_2 = 49^\circ\text{F}$$

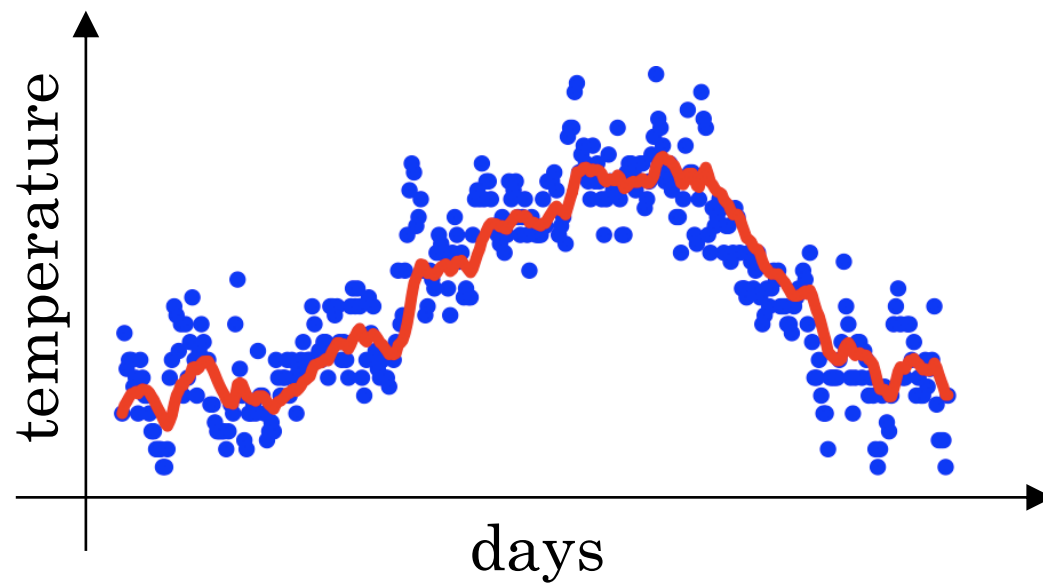
$$\theta_3 = 45^\circ\text{F}$$

$\vdots$

$$\theta_{180} = 60^\circ\text{F}$$

$$\theta_{181} = 56^\circ\text{F}$$

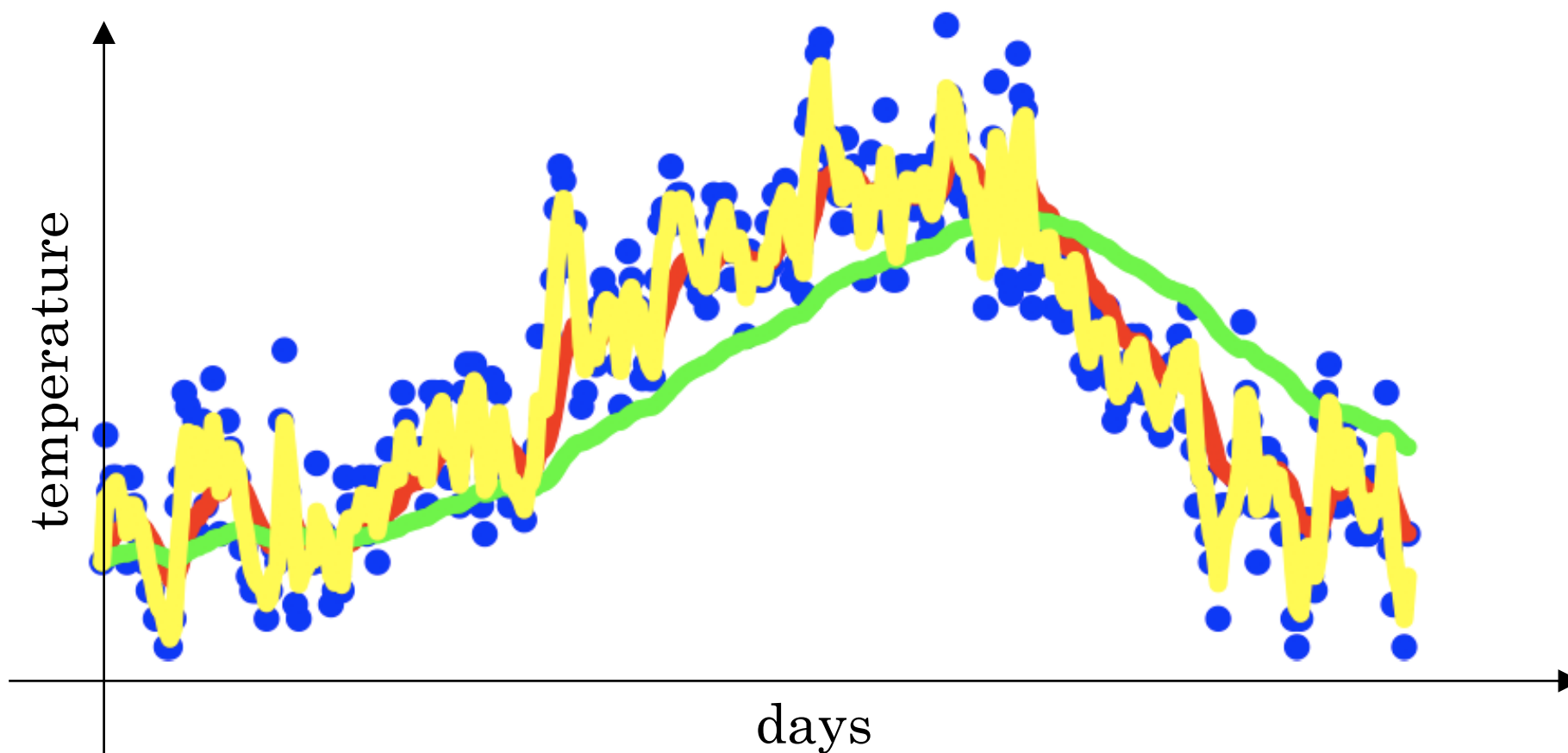
$\vdots$



# 指数加权平均数

8

$$v_t = \beta v_{t-1} + (1 - \beta)\theta_t$$





# 指数加权平均数

9

$$v_0 = 0$$

$$v_1 = \beta v_0 + (1 - \beta) \theta_1$$

$$v_2 = \beta v_1 + (1 - \beta) \theta_2$$

$$v_3 = \beta v_2 + (1 - \beta) \theta_3$$

...

$$v_t = \beta v_{t-1} + (1 - \beta) \theta_t$$

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$

$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$

$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

...

# Momentum

10

$$v_{dW} = \beta v_{dW} + (1 - \beta) dW$$

$$v = \beta v + (1 - \beta) \theta_t,$$

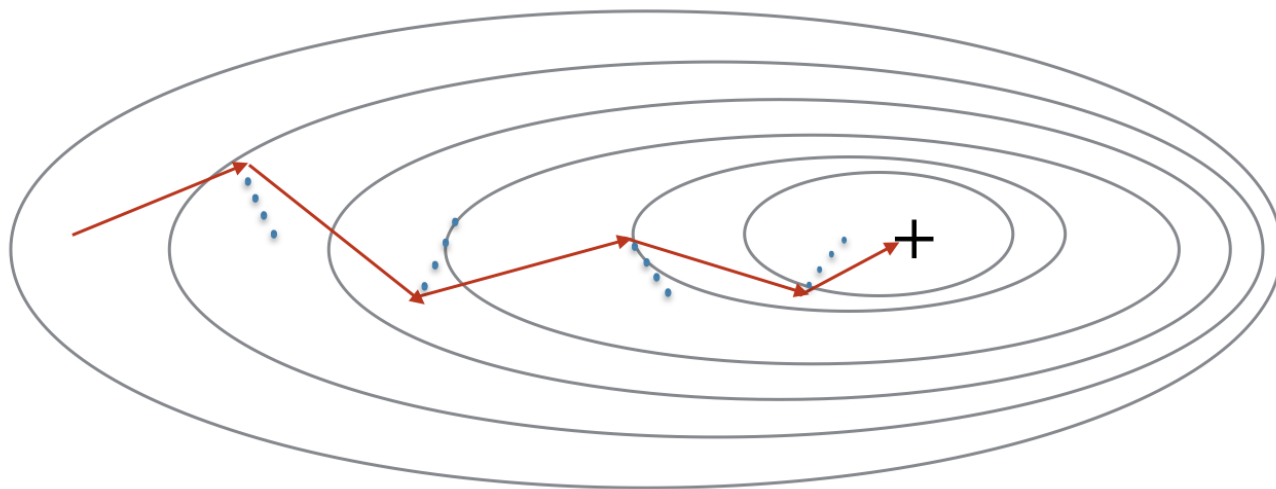
$$v_{db} = \beta v_{db} + (1 - \beta) db,$$

$$W := W - a v_{dW},$$

$$b := b - a v_{db},$$

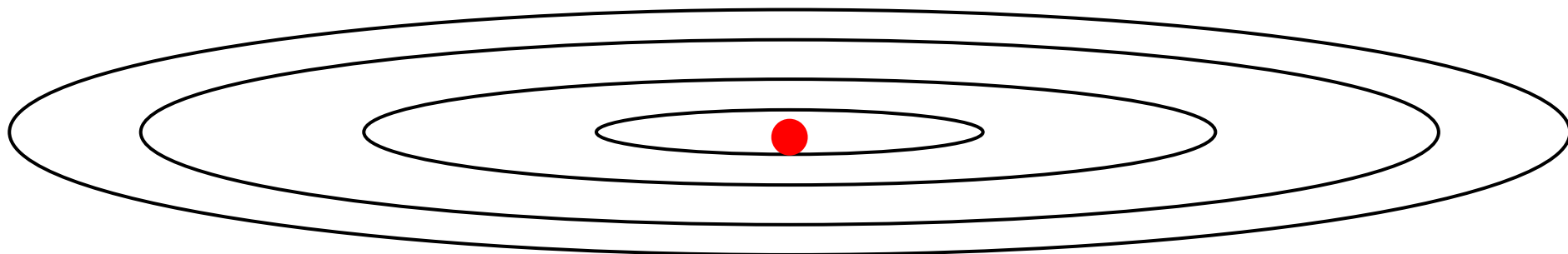
这样就可以减缓梯度下降的  
幅度。

通常情况下:  $\beta = 0.9$



# RMSprop

11



在第 $t$ 次迭代中，该算法会照常计算当下**mini-batch**的微分 $dW$ ,  $db$ ，所以我会保留这个指数加权平均数，我们用到新符号 $S_{dW}$ ，而不是 $v_{dW}$ ，因此 $S_{dW} = \beta S_{dW} + (1 - \beta)dW^2$ ，澄清一下，这个平方的操作是针对这一整个符号的，这样做能够保留微分平方的加权平均数，同样 $S_{db} = \beta S_{db} + (1 - \beta)db^2$ ，再说一次，平方是针对整个符号的操作。

接着**RMSprop**会这样更新参数值， $W := W - a \frac{dW}{\sqrt{S_{dW}}}$ ， $b := b - \alpha \frac{db}{\sqrt{S_{db}}}$ ，

# ADAM

12

**Adam**优化算法基本上就是将**Momentum**和**RMSprop**结合在一起

最后更新权重，所以 $W$ 更新后是 $W := W - \frac{\alpha v_{dW}^{\text{corrected}}}{\sqrt{S_{dW}^{\text{corrected}} + \epsilon}}$ （如果你只是用

**Momentum**，使用 $v_{dW}$ 或者修正后的 $v_{dW}$ ，但现在我们加入了**RMSprop**的部分，所以我们要除以修正后 $S_{dW}$ 的平方根加上 $\epsilon$ ）。

根据类似的公式更新 $b$ 值， $b := b - \frac{\alpha v_{db}^{\text{corrected}}}{\sqrt{S_{db}^{\text{corrected}} + \epsilon}}$ 。

# 学习率衰减

13

加快学习算法的一个办法就是随时间慢慢减少学习率，我们将之称为学习率衰减

可以将 $a$ 学习率设为
$$a = \frac{1}{1 + \text{decayrate} * \text{epoch-num}} a_0$$

(**decay-rate**称为衰减率，**epoch-num**为代数， $\alpha_0$ 为初始学习率)

# Pytorch的优化器

14

# 超参数

LR = 0.01

opt\_SGD = torch.optim.SGD(net\_SGD.parameters(), lr=LR)

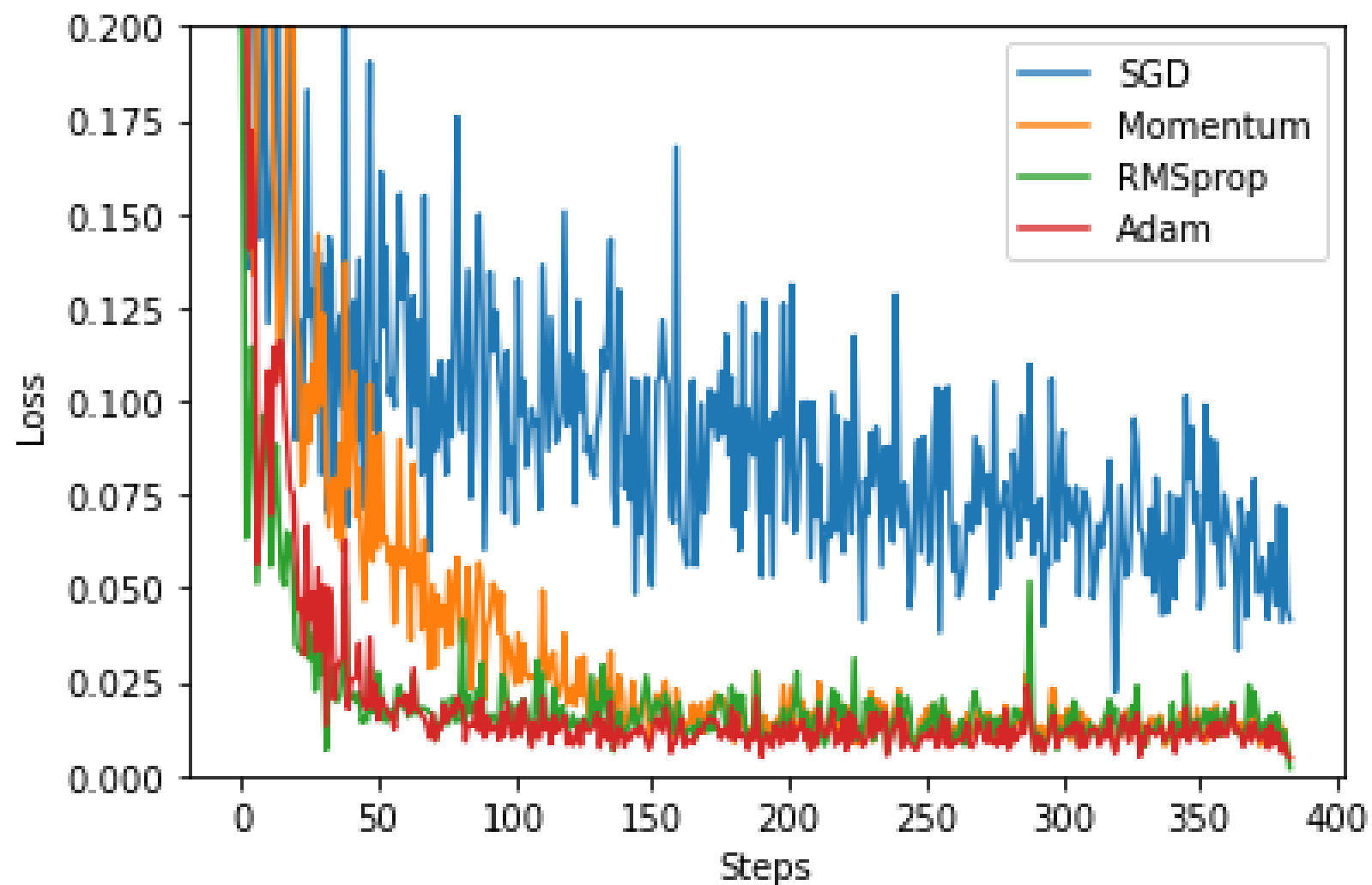
opt\_Momentum = torch.optim.SGD(net\_Momentum.parameters(), lr=LR,  
momentum=0.9)

opt\_RMSProp = torch.optim.RMSprop(net\_RMSProp.parameters(), lr=LR,  
alpha=0.9)

opt\_Adam = torch.optim.Adam(net\_Adam.parameters(), lr=LR, betas=(0.9, 0.99))

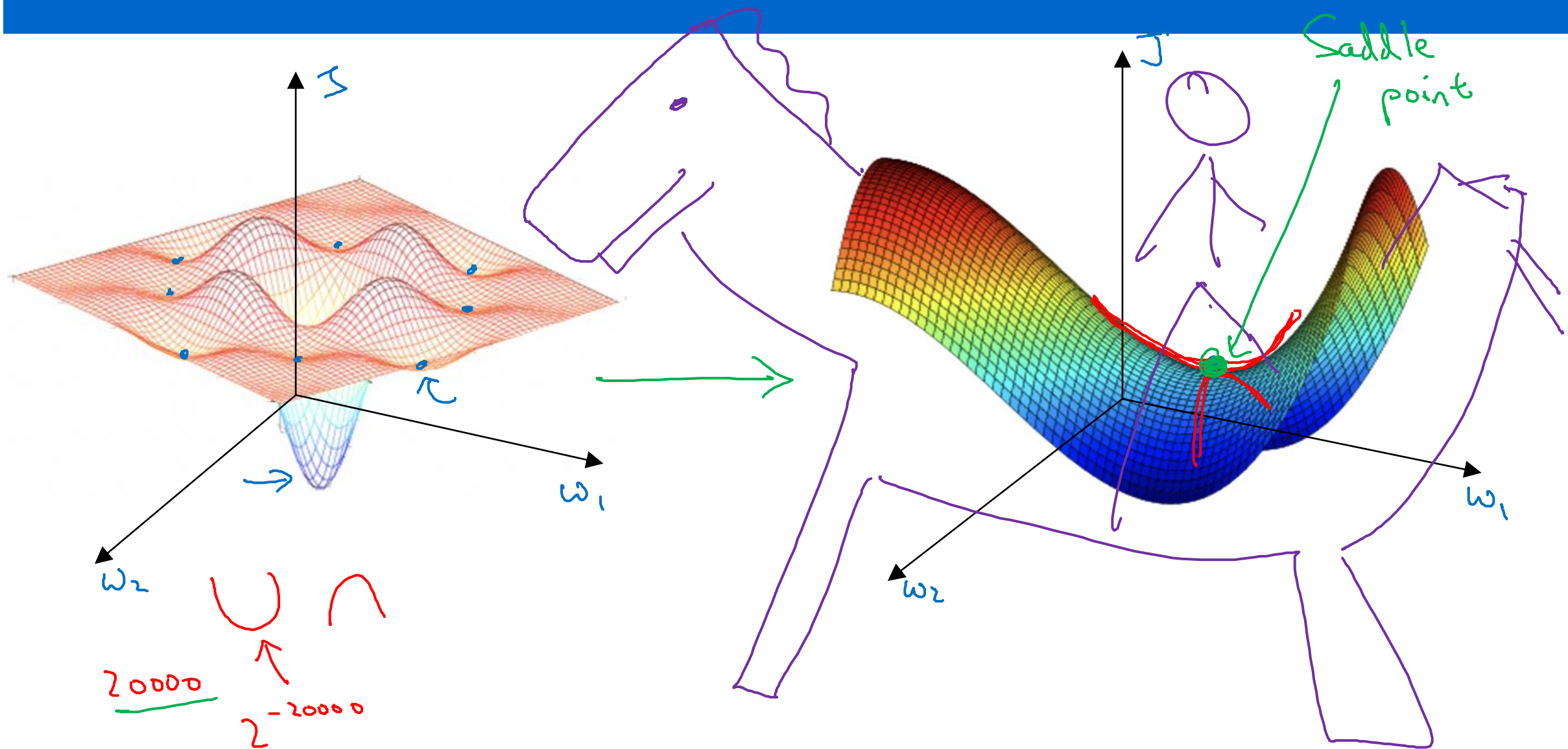
# Pytorch的优化器

15



# 神经网络的局部最优问题

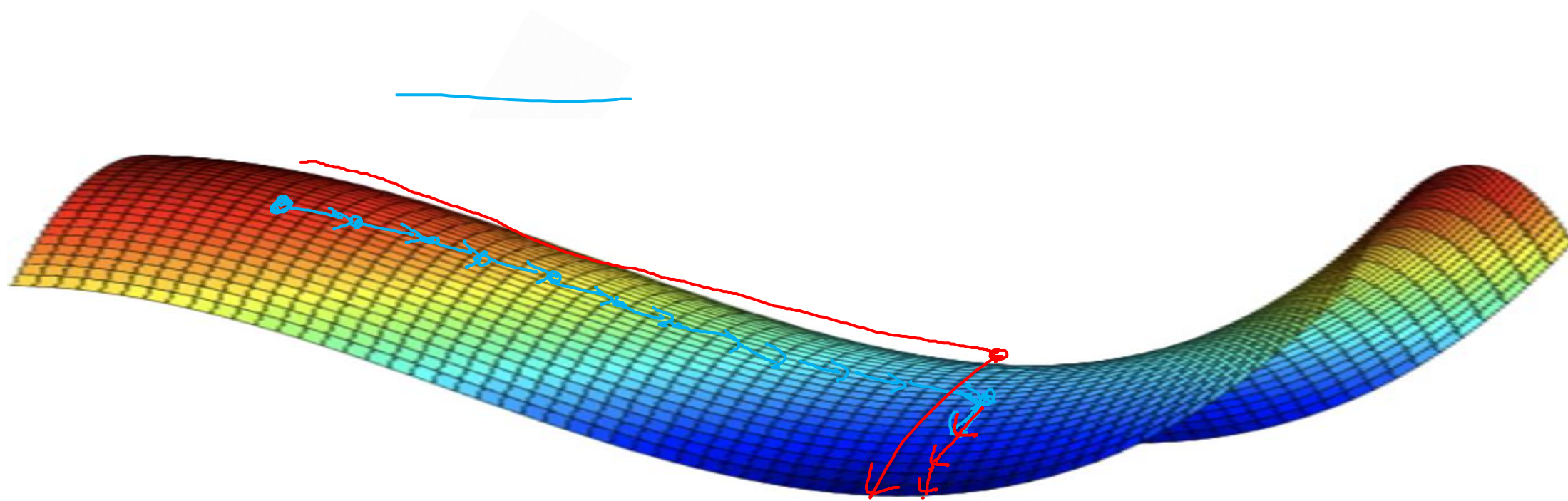
16





# 局部最优问题

17



# 3.BatchNorm

18

**01** 小批量梯度下降

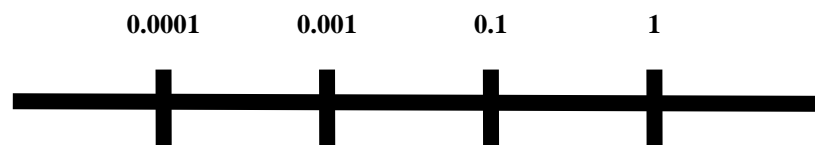
**02** 优化算法

**03** 超参数调整和BatchNorm

**04** Softmax

# 超参数调整的方法

19

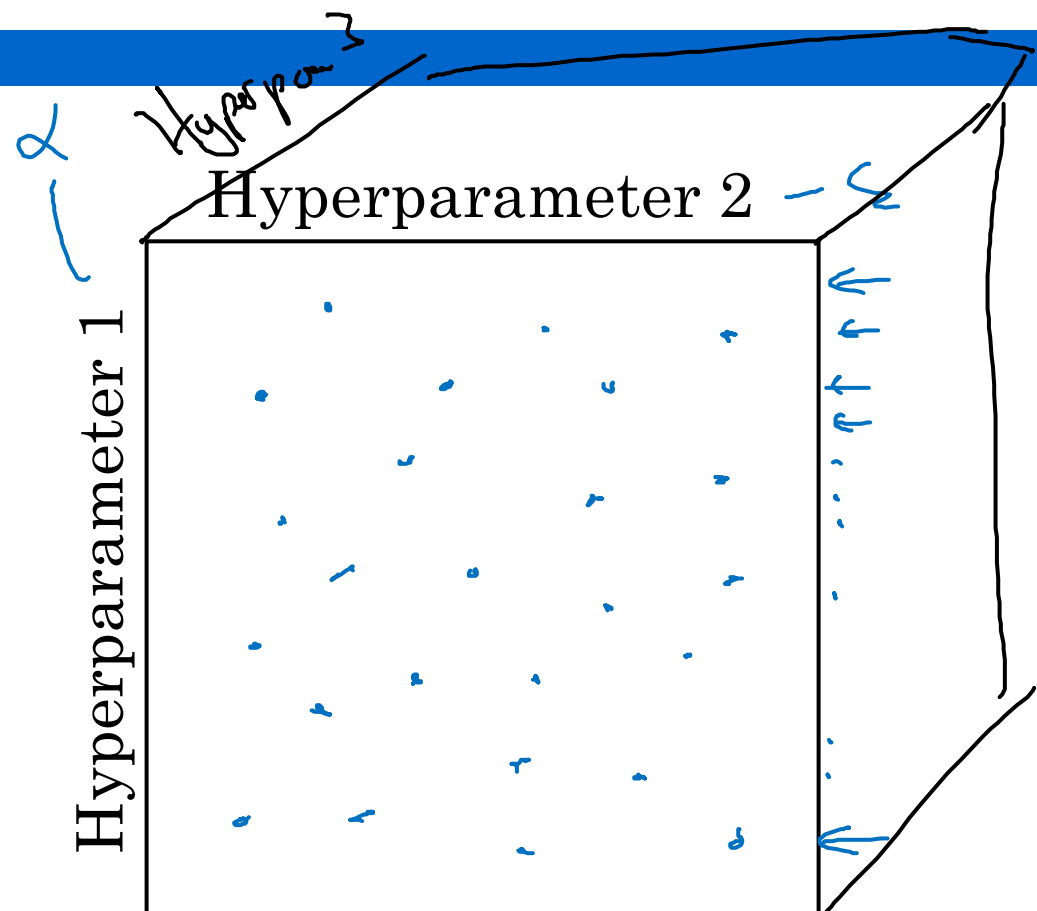
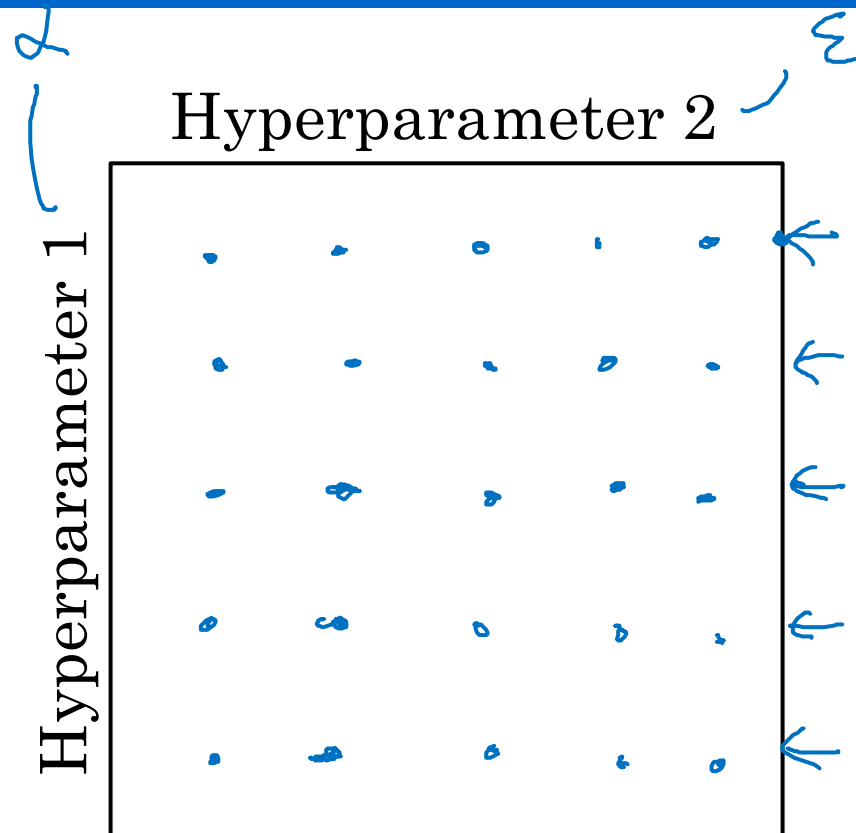


假设你在搜索超参数（学习率），假设你怀疑其值最小是0.0001或最大是1。如果你画一条从0.0001到1的数轴，沿其随机均匀取值，那90%的数值将会落在0.1到1之间，结果就是， $\alpha$ 在0.1到1之间，应用了90%的资源，而 $\alpha$ 在0.0001到0.1之间，只有10%的搜索资源。

反而，用对数标尺搜索超参数的方式会更合理，因此这里不使用线性轴，分别依次取0.0001，0.001，0.01，0.1，1，在对数轴上均匀随机取点，这样，在0.0001到0.001之间，就会有更多的搜索资源可用，还有在0.001到0.01之间等等。

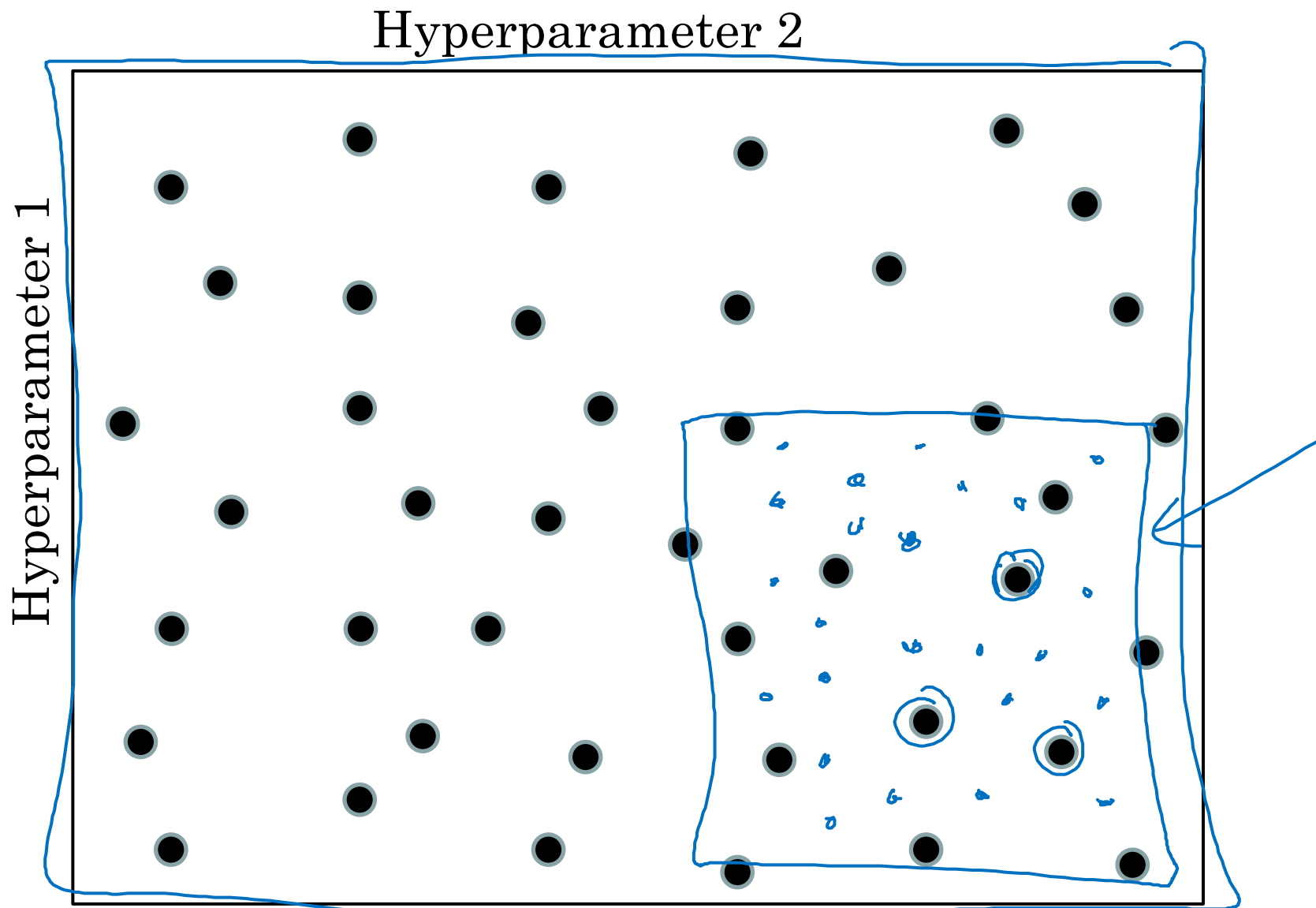
# 超参数调整的方法

20



# 由粗到细调整超参数

21

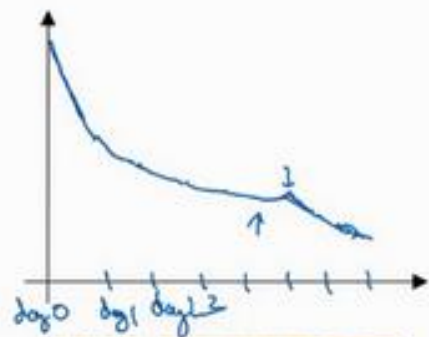


# 熊猫方式与鱼子酱方式

22

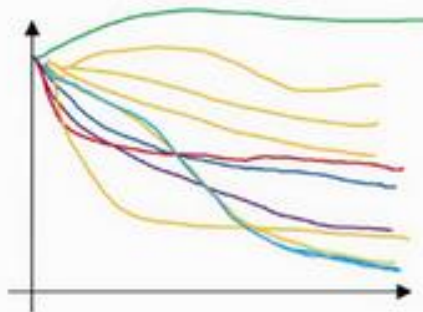
由计算资源决定

Babysitting one  
model



Panda

Training many  
models in parallel



Caviar

Andrew Ng

# Batch Norm

23

Batch Normalization是2015年一篇论文中提出的数据归一化方法，往往用在深度神经网络中激活层之前。其作用可以加快模型训练时的收敛速度，使得模型训练过程更加稳定，避免梯度爆炸或者梯度消失。并且起到一定的正则化作用，几乎代替了Dropout。

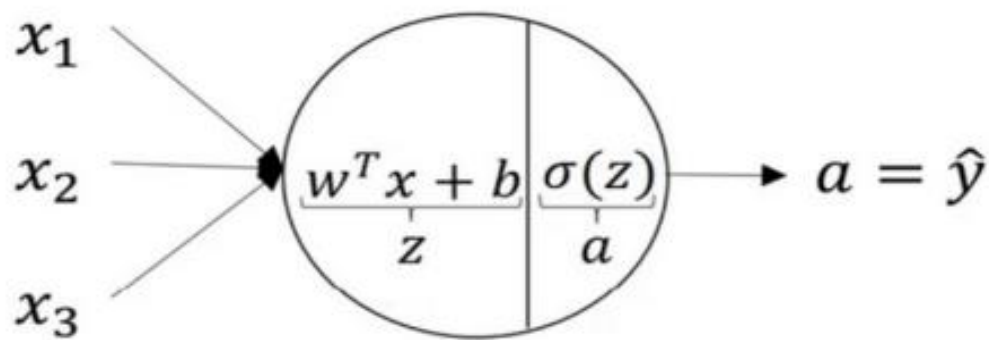
在深度学习中，由于采用full batch的训练方式对内存要求较大，且每一轮训练时间过长；我们一般都会采用对数据做划分，用mini-batch对网络进行训练。因此，Batch Normalization也就在**mini-batch的基础上**进行计算。

**让每个特征都有均值为0，方差为1的分布**

# Batch Norm

24

发生在计算 $z$ 和 $a$ 之间的，将每一批次数据的输入值减去这一批次均值然后除以其标准差。



$$z = w^T x + b$$

$$a = \sigma(z)$$

$$\gamma = \sqrt{\sigma^2 + \varepsilon}, \beta = \mu$$

$$\mu = \frac{1}{m} \sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_i (z^{(i)} - \mu)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta$$



# Batch Norm

25

## 作用

- (1) BN使得网络中每层输入数据的分布相对稳定，加速模型学习速度**
- (2) BN使得模型对网络中的参数不那么敏感，简化调参过程，使得网络学习更加稳定**
- (3) BN允许网络使用饱和性激活函数（例如sigmoid, tanh等），缓解梯度消失问题**
- (4) BN具有一定的正则化效果**

# 4. Softmax

26

**01** 小批量梯度下降

**02** 优化算法

**03** 超参数调整和BatchNorm

**04** Softmax

# Softmax 层

27



3



1



2



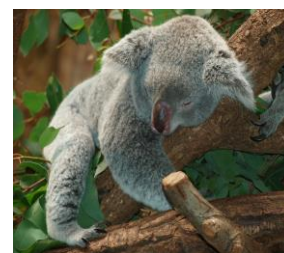
0



3



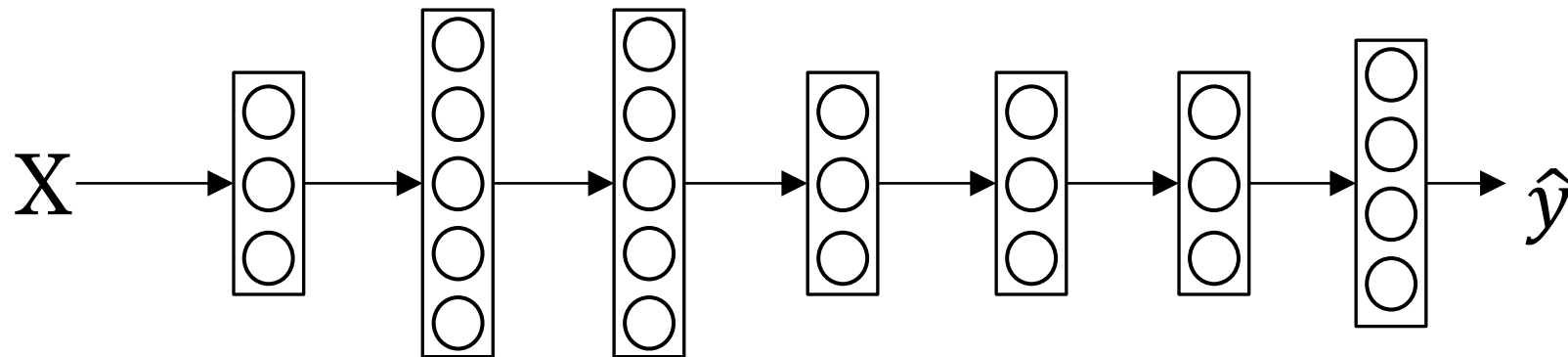
2



0

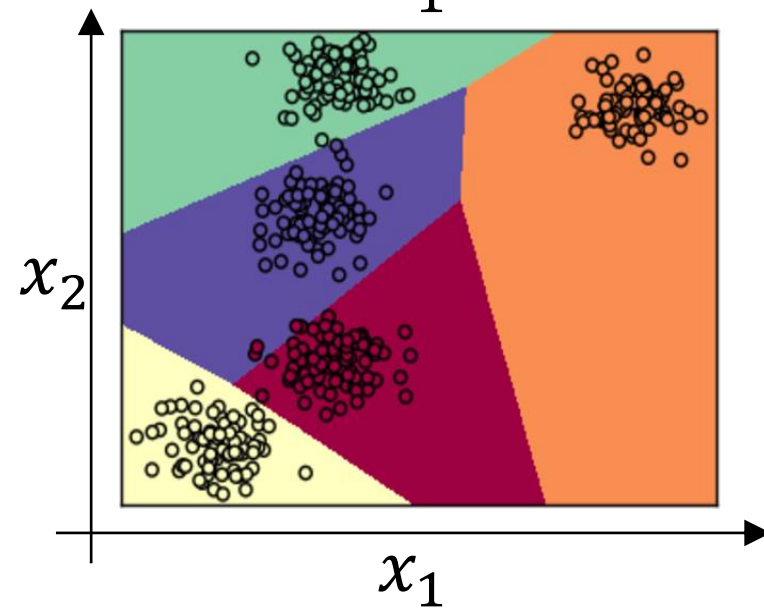
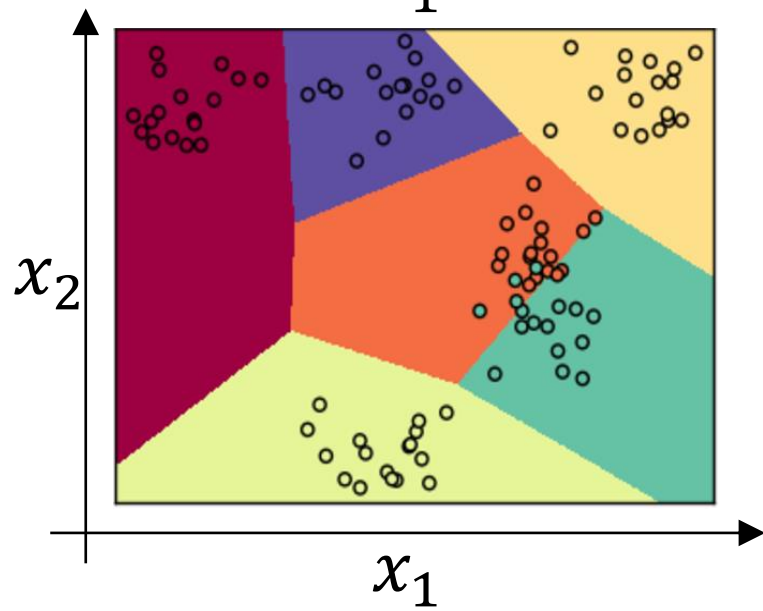
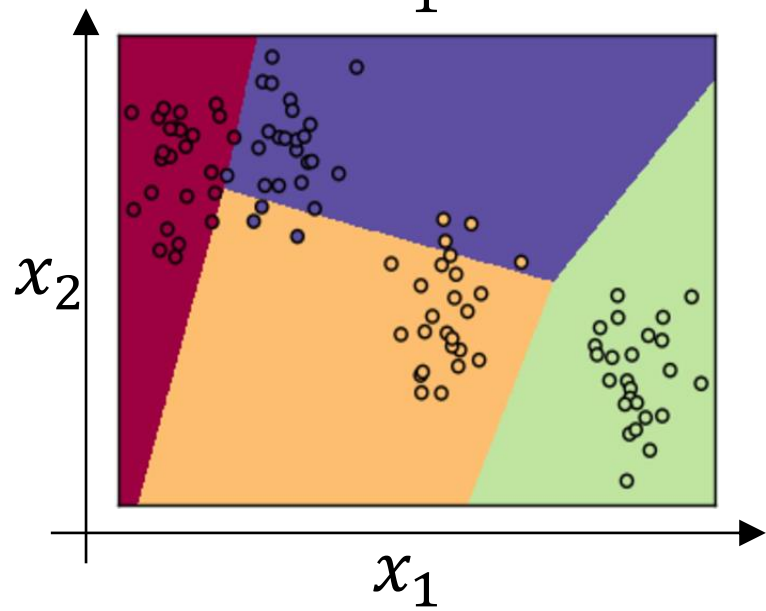
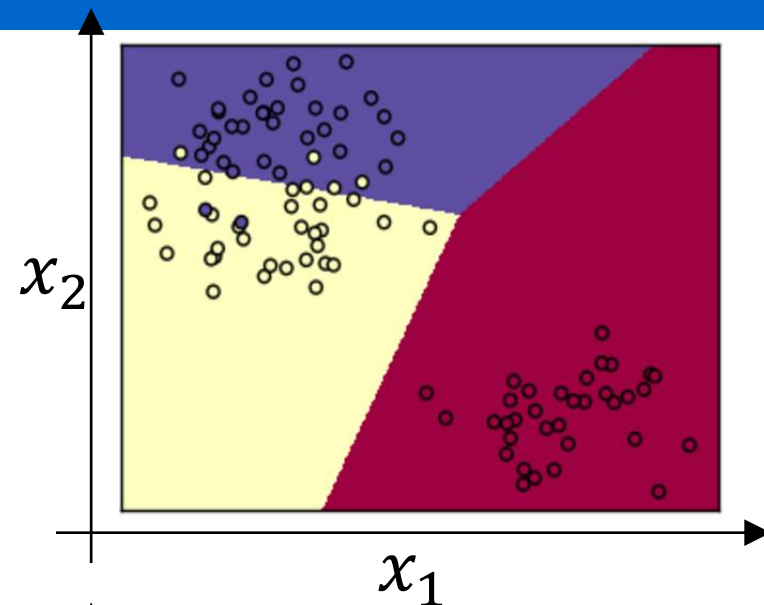
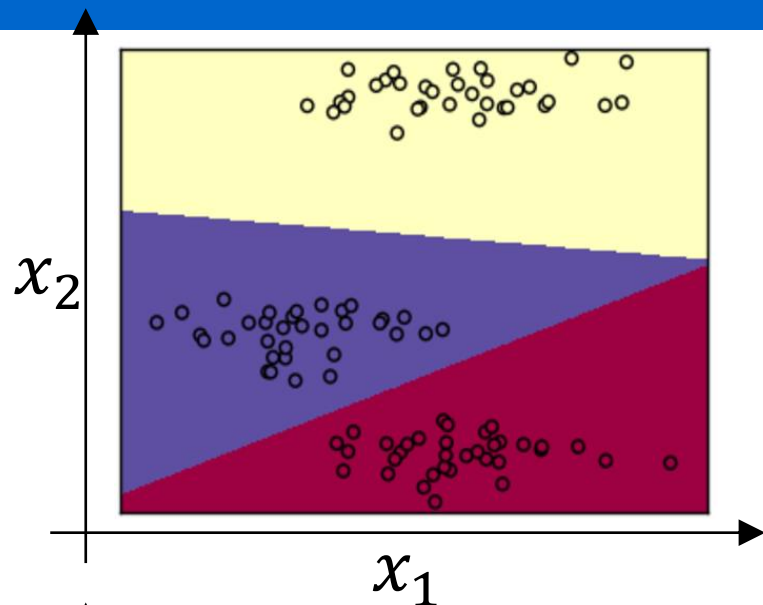
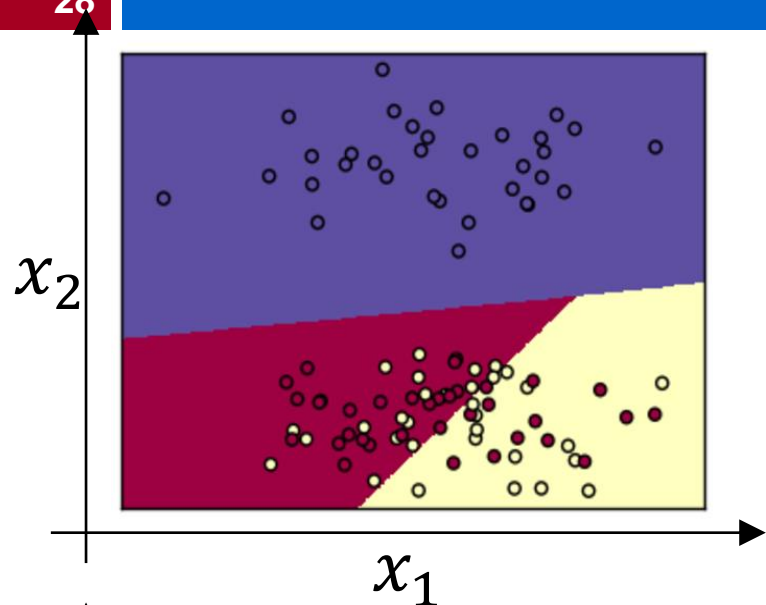


1



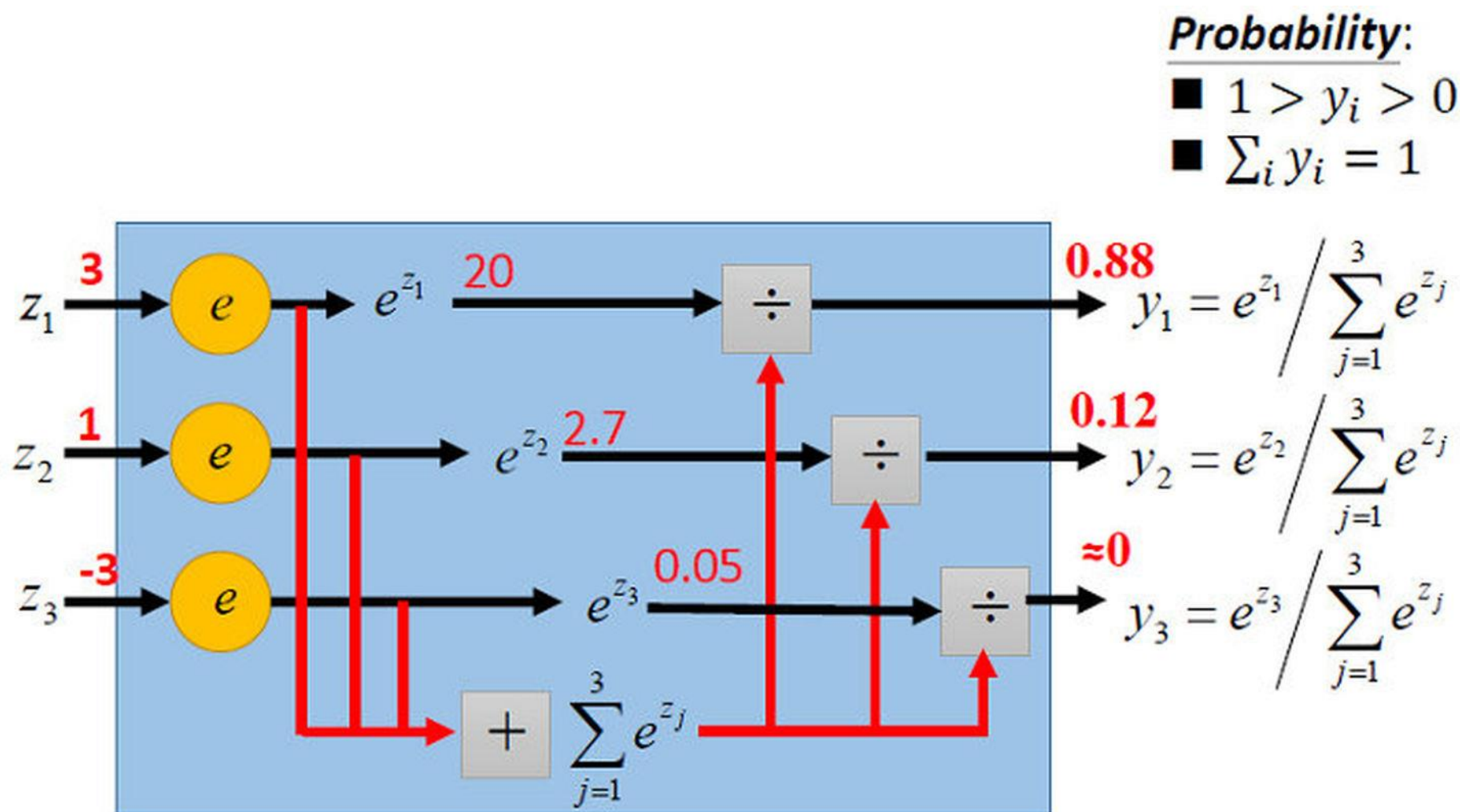
# Softmax 样本

28



# Softmax 回归

29



1. IAN GOODFELLOW等, 《深度学习》, 人民邮电出版社, 2017
2. Andrew Ng, <http://www.deeplearning.ai>



谢谢!

