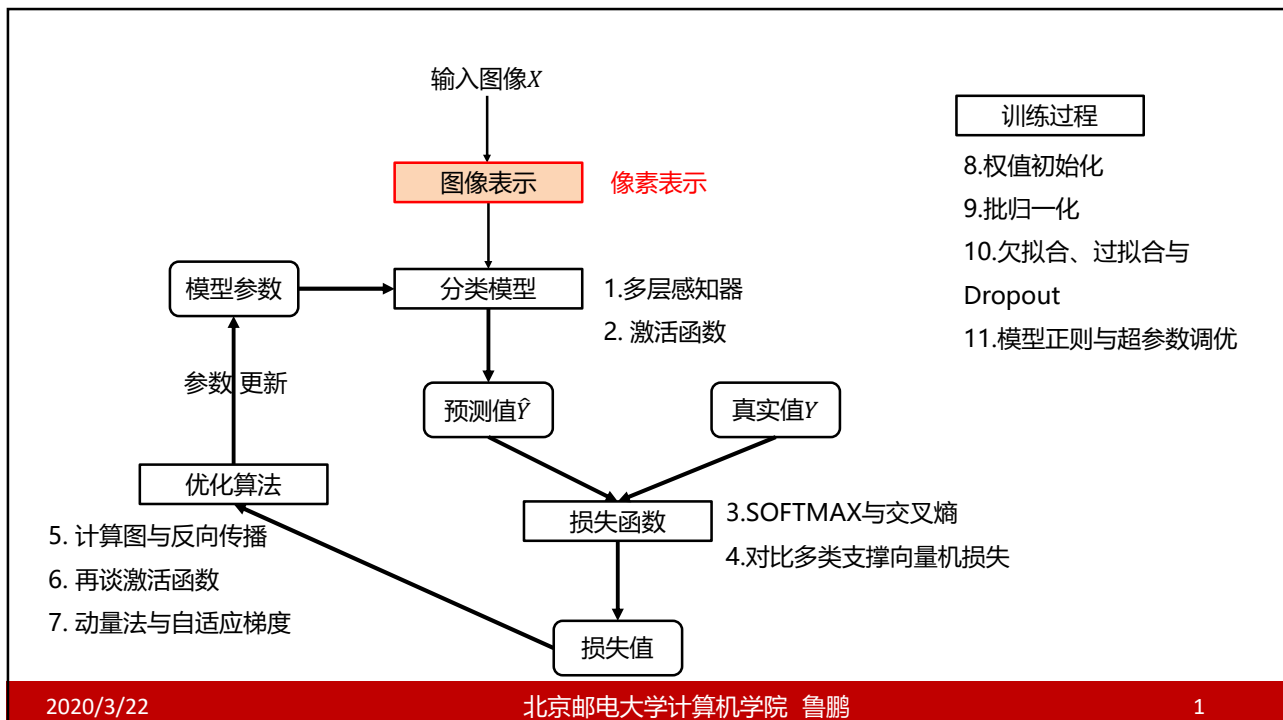


# 全连接神经网络

鲁 鹏

北京邮电大学 计算机学院 智能科学与技术中心

0



2020/3/22

北京邮电大学计算机学院 鲁鹏

1

1

## 如何表示图像？

直接利用原始像素作为特征，展开为列向量。

$$\begin{bmatrix} 76 & 32 \\ 4 & 157 \end{bmatrix} \xrightarrow{\text{将矩阵转成列向量}} x = \begin{bmatrix} 76 \\ 32 \\ 4 \\ 157 \end{bmatrix}$$

图像

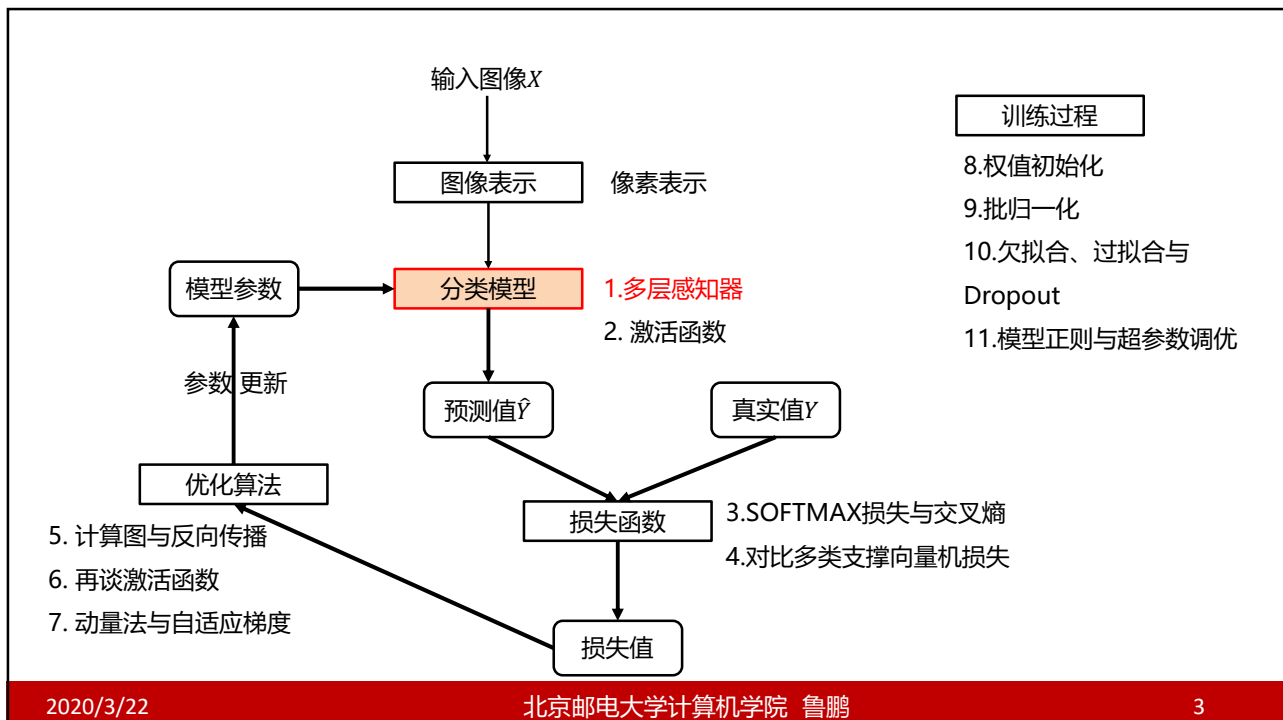
cifar10中每个图像可表示为一个3072 (32\*32\*3) 维的向量

2020/3/22

北京邮电大学计算机学院 鲁鹏

2

2



2020/3/22

北京邮电大学计算机学院 鲁鹏

3

3

## 线性分类器

$$f(x, W) = Wx + b$$

其中,  $x$  代表输入图像, 其维度为  $d$  ;

$f$  为分数向量, 其维度等于类别个数  $c$

$W = [w_1 \cdots w_c]^T$  为权值矩阵,  $w_i = [w_{i1} \cdots w_{id}]^T$  为第  $i$  个类别的权值向量

$b = [b_1 \cdots b_c]^T$  为偏置向量,  $b_i$  为第  $i$  个类别的偏置

2020/3/22

北京邮电大学计算机学院 鲁鹏

4

4

## 全连接神经网络

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

5

5

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

6

6

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

7

7

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

8

8

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

9

9

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

三层全连接网络  $f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2)$

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

10

10

## 全连接神经网络

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

三层全连接网络  $f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2)$

注意：非线性操作是不可以去掉

全连接神经网络级联多个变换来实现输入到输出的映射。

2020/3/22

北京邮电大学计算机学院 鲁鹏

11

11

## 全连接神经网络的权值

线性分类器:  $f(x, W) = \boxed{W}x + b$



权值模板



➤ 线性分类器中的  $W$  可看作模板，模板个数由类别个数决定

2020/3/22

北京邮电大学计算机学院 鲁鹏

12

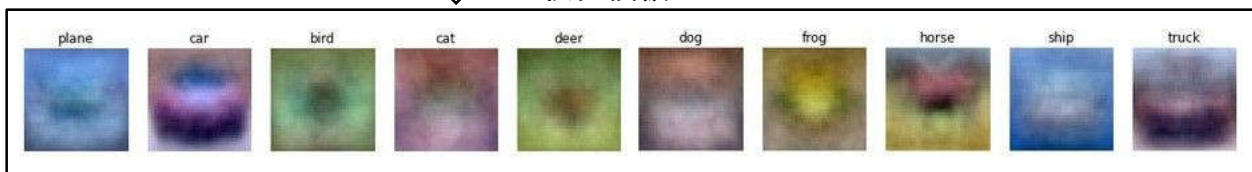
12

## 全连接神经网络的权值

线性分类器:  $f(x, W) = \boxed{W}x + b$



权值模板



两层全连接网络:

$$f = W_2 \max(0, \boxed{W_1}x + b_1) + b_2$$

➤ 线性分类器中的  $W$  可看作模板，模板个数由类别个数决定

➤ 全连接神经网络中:

◆  $W_1$  也可看作模板；模板个数人为指定

2020/3/22

北京邮电大学计算机学院 鲁鹏

13

13

## 全连接神经网络的权值

线性分类器:  $f(x, W) = \boxed{W}x + b$



权值模板



两层全连接网络:

$$f = W_2 \max(0, \boxed{W_1}x + b_1) + b_2$$

➤ 线性分类器中的  $W$  可看作模板，模板个数由类别个数决定

➤ 全连接神经网络中:

◆  $W_1$  也可看作模板；模板个数人为指定

◆  $W_2$  融合这多个模板的匹配结果来实现最终类别打分

2020/3/22

北京邮电大学计算机学院 鲁鹏

14

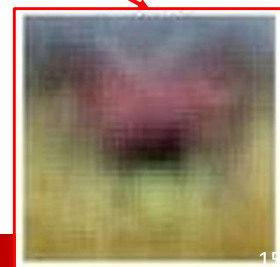
14

## 全连接神经网络的权值

线性分类器:  $f(x, W) = \boxed{W}x + b$



权值模板



两个马头

2020/3/22

北京邮电大学计算机学院 鲁鹏

15

15



## 全连接神经网络的权值

线性分类器:  $f(x, W) = Wx + b$

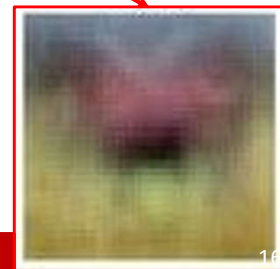


权值模板



两层全连接网络:

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



两个马头

2020/3/22

北京邮电大学计算机学院 鲁鹏

16

## 全连接神经网络的权值

线性分类器:  $f(x, W) = Wx + b$



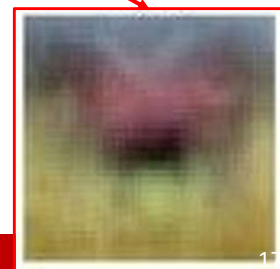
权值模板



两层全连接网络:

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$

全连接神经网络的描述能力更强。因为调整  $W_1$  行数等于增加模板个数, 分类器有机会学到两个不同方向的马的模板。



两个马头

2020/3/22

北京邮电大学计算机学院 鲁鹏

17

## 全连接神经网络与线性不可分

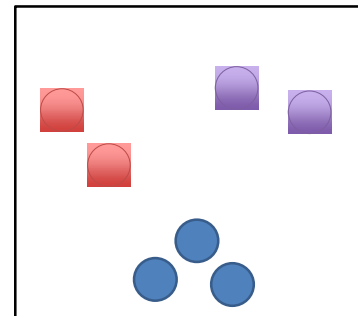
两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

线性分类器

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

线性可分——至少存在一个线性分界面  
能把两类样本没有错误的分开。

线性可分



2020/3/22

北京邮电大学计算机学院 鲁鹏

18

18

## 全连接神经网络与线性不可分

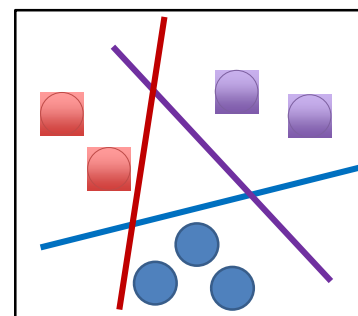
两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

线性分类器

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

线性可分——至少存在一个线性分界面  
能把两类样本没有错误的分开。

线性可分



2020/3/22

北京邮电大学计算机学院 鲁鹏

19

19

## 全连接神经网络与线性不可分

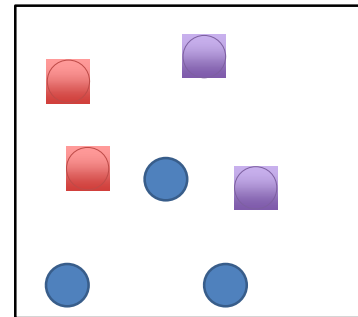
两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

线性分类器

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

线性可分——至多存在一个线性分界面  
能把两类样本错误的分开。

线性不可分



2020/3/22

北京邮电大学计算机学院 鲁鹏

20

20

## 全连接神经网络与线性不可分

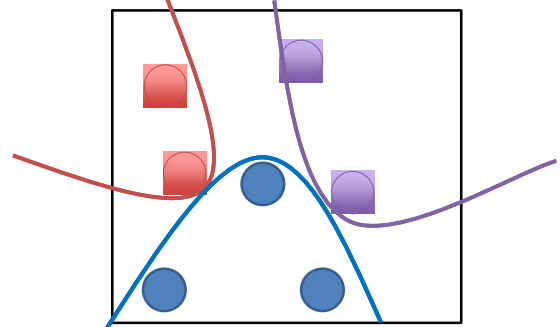
两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$

线性分类器

$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

线性可分——至多存在一个线性分界面  
能把两类样本错误的分开。

线性不可分



2020/3/22

北京邮电大学计算机学院 鲁鹏

21

21

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$

2020/3/22

北京邮电大学计算机学院 鲁鹏

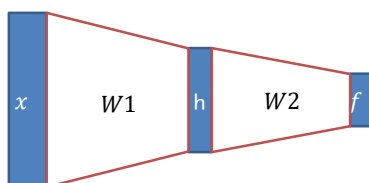
22

22

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

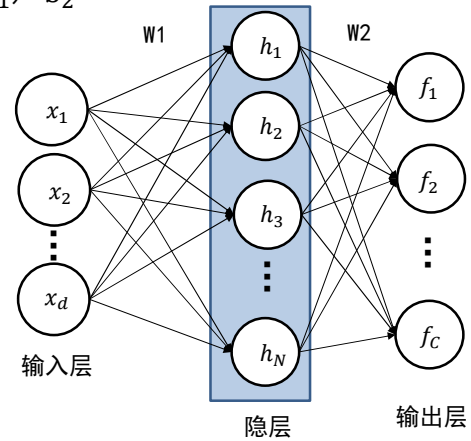
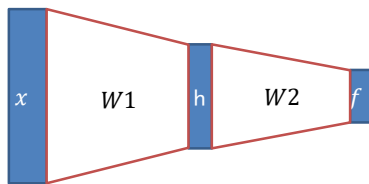
北京邮电大学计算机学院 鲁鹏

23

23

## 全连接神经网络绘制与命名

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$



2020/3/22

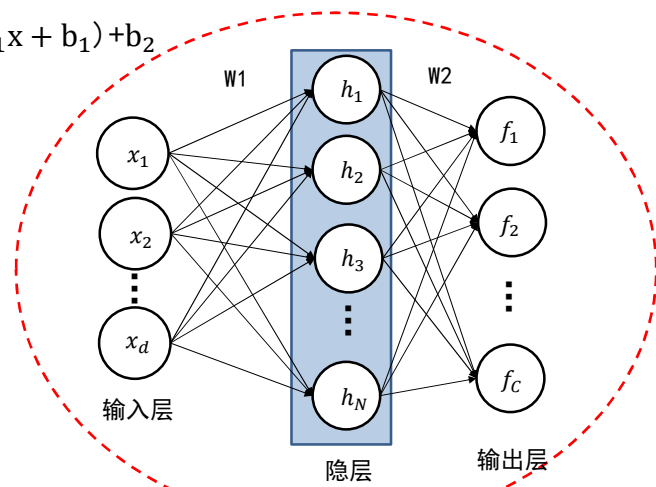
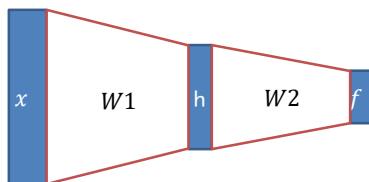
北京邮电大学计算机学院 鲁鹏

24

24

## 全连接神经网络绘制与命名

两层全连接网络  $f = W_2 \max(0, W_1 x + b_1) + b_2$



2020/3/22

北京邮电大学计算机学院 鲁鹏

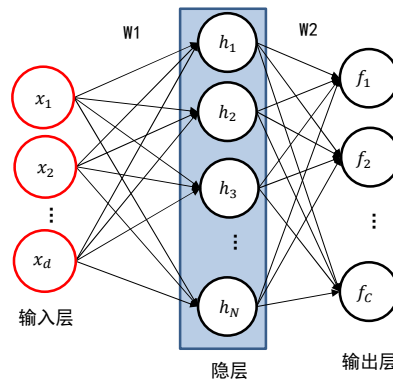
25

25

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

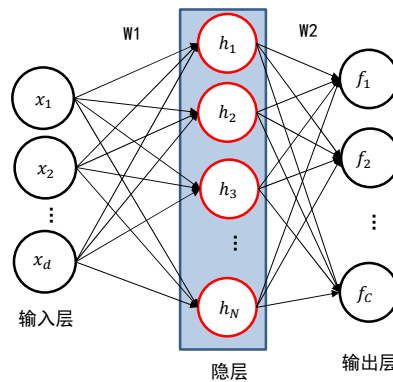
26

26

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

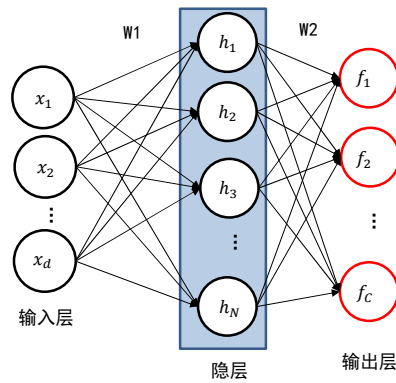
27

27

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

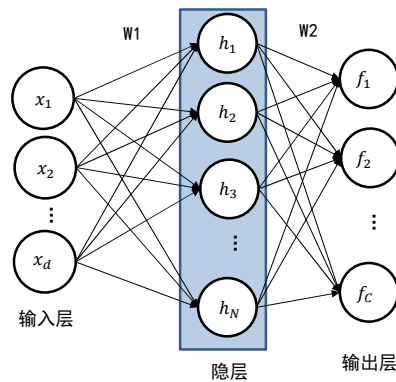
28

28

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

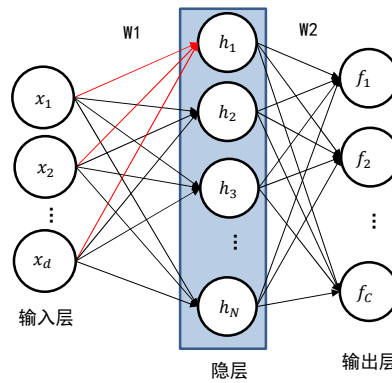
29

29

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

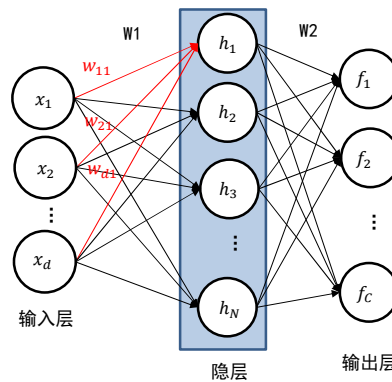
30

30

## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

31

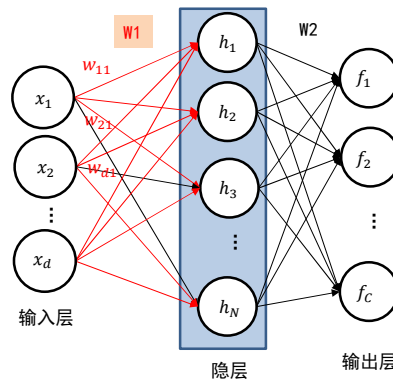
31



## 全连接神经网络绘制与命名

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

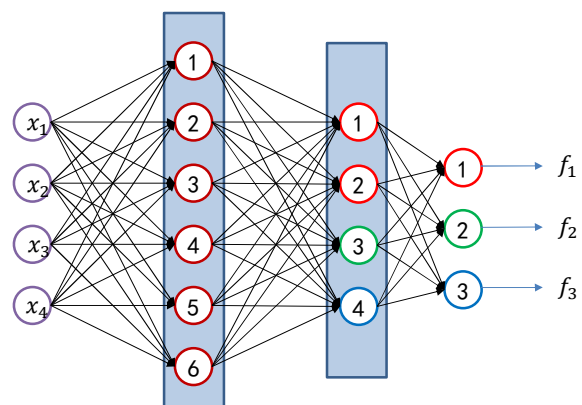
32

32

## 全连接神经网络绘制与命名

N层全连接神经网络——除输入

层之外其他层的数量为N的网络



$$f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$$

“3层神经网络”

2020/3/22

北京邮电大学计算机学院 鲁鹏

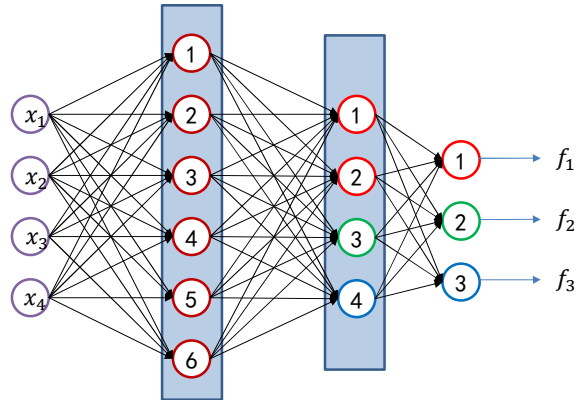
33

33

## 全连接神经网络绘制与命名

N层全连接神经网络——除输入层之外其他层的数量为N的网络

N个隐层的全连接神经网络——  
网络隐层的数量为N的网络



$$f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$$

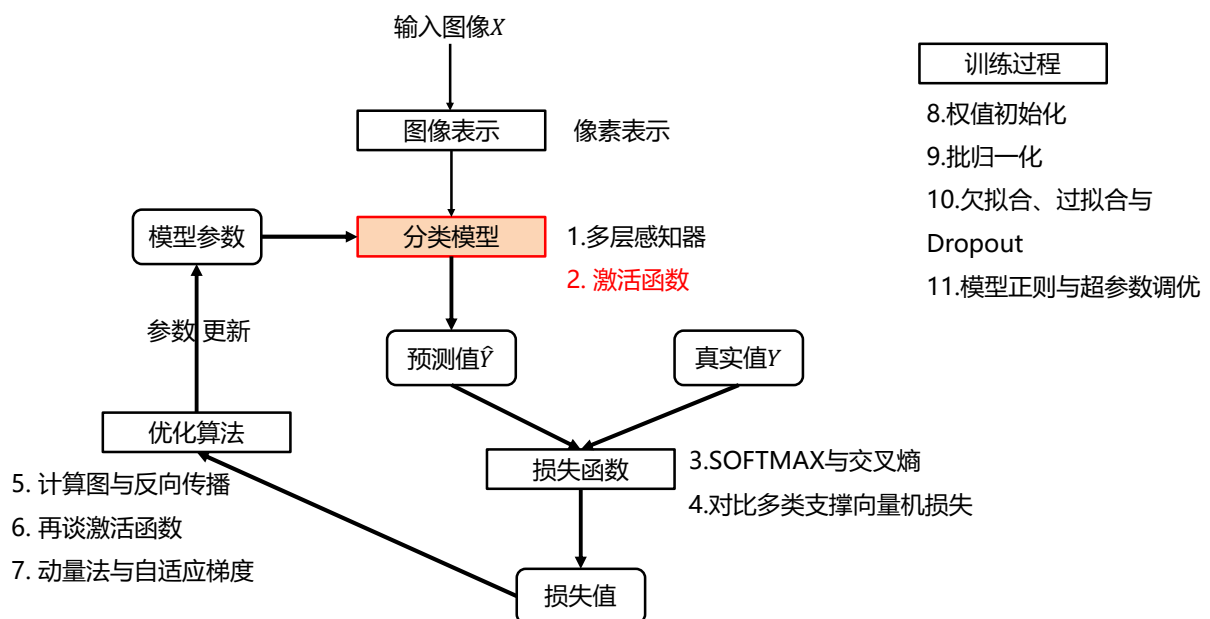
“3层神经网络”，或者“2隐层神经网络”

2020/3/22

北京邮电大学计算机学院 鲁鹏

34

34



2020/3/22

北京邮电大学计算机学院 鲁鹏

35

35

## 激活函数

- 为什么需要非线性操作？

三层全连接网络  $f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$

2020/3/22

北京邮电大学计算机学院 鲁鹏

36

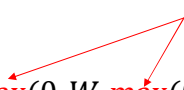
36

## 激活函数

- 为什么需要非线性操作？

三层全连接网络  $f = W_3 \text{max}(0, W_2 \text{max}(0, W_1 x + b_1) + b_2) + b_3$

激活函数



2020/3/22

北京邮电大学计算机学院 鲁鹏

37

37

## 激活函数

- 为什么需要非线性操作？

三层全连接网络  $f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$

激活函数

答：如果网络中缺少了激活函数，全连接神经网络将变成一个线性分类器。

2020/3/22

北京邮电大学计算机学院 鲁鹏

38

38

## 激活函数

- 为什么需要非线性操作？

三层全连接网络  $f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$

激活函数

答：如果网络中缺少了激活函数，全连接神经网络将变成一个线性分类器。

去掉激活函数

$$\begin{aligned}
 f &= W_3(W_2(W_1 x + b_1) + b_2) + b_3 \\
 &= W_3 W_2 W_1 x + (W_3 W_2 b_1 + W_3 b_2 + b_3) \\
 &= W' x + b'
 \end{aligned}$$

2020/3/22

北京邮电大学计算机学院 鲁鹏

39

39

# 常用的激活函数

2020/3/22

北京邮电大学计算机学院 鲁鹏

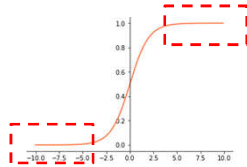
40

40

# 常用的激活函数

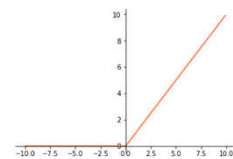
Sigmoid

$$1/(1 + e^{-x})$$



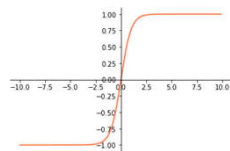
ReLU

$$\max(0, x)$$



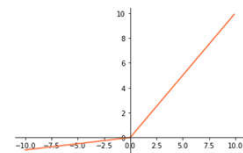
tanh

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Leaky ReLU

$$\max(0.1x, x)$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

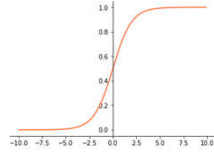
41

41

## 常用的激活函数

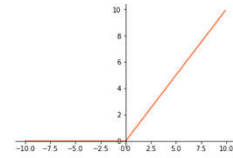
Sigmoid

$$1/(1 + e^{-x})$$



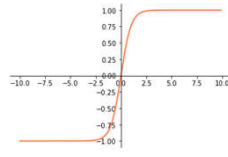
ReLU

$$\max(0, x)$$



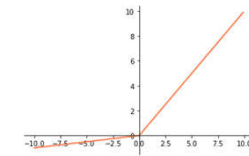
tanh

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Leaky ReLU

$$\max(0.1x, x)$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

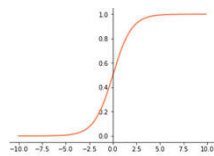
42

42

## 常用的激活函数

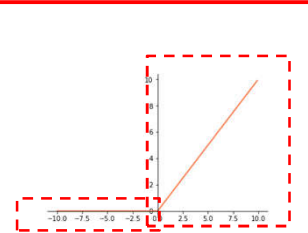
Sigmoid

$$1/(1 + e^{-x})$$



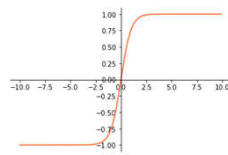
ReLU

$$\max(0, x)$$



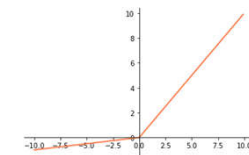
tanh

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Leaky ReLU

$$\max(0.1x, x)$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

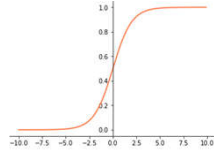
43

43

## 常用的激活函数

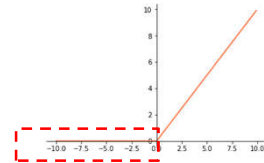
Sigmoid

$$1/(1 + e^{-x})$$



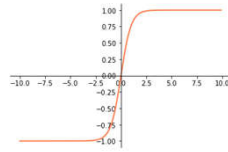
ReLU

$$\max(0, x)$$



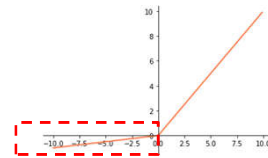
tanh

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Leaky ReLU

$$\max(0.1x, x)$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

44

44

## 网络结构设计

1. 用不用隐层，用一个还是用几个隐层？（深度设计）
2. 每隐层设置多少个神经元比较合适？（宽度设计）

没有统一的答案！

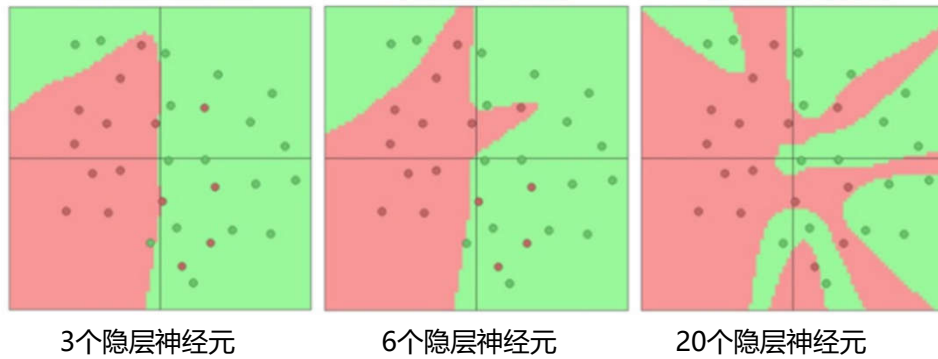
2020/3/22

北京邮电大学计算机学院 鲁鹏

45

45

## 网络结构设计



结论：神经元个数越多，分界面就可以越复杂，在这个集合上的分类能力就越强。

2020/3/22

北京邮电大学计算机学院 鲁鹏

46

46

## 网络结构设计

依据分类任务的难易程度来调整神经网络模型的复杂程度。分类任务越难，我们设计的神经网络结构就应该越深、越宽。但是，需要注意的是对训练集分类精度最高的全连接神经网络模型，在真实场景下识别性能未必是最好的（过拟合）。

2020/3/22

北京邮电大学计算机学院 鲁鹏

47

47



## 全连接神经网络小结

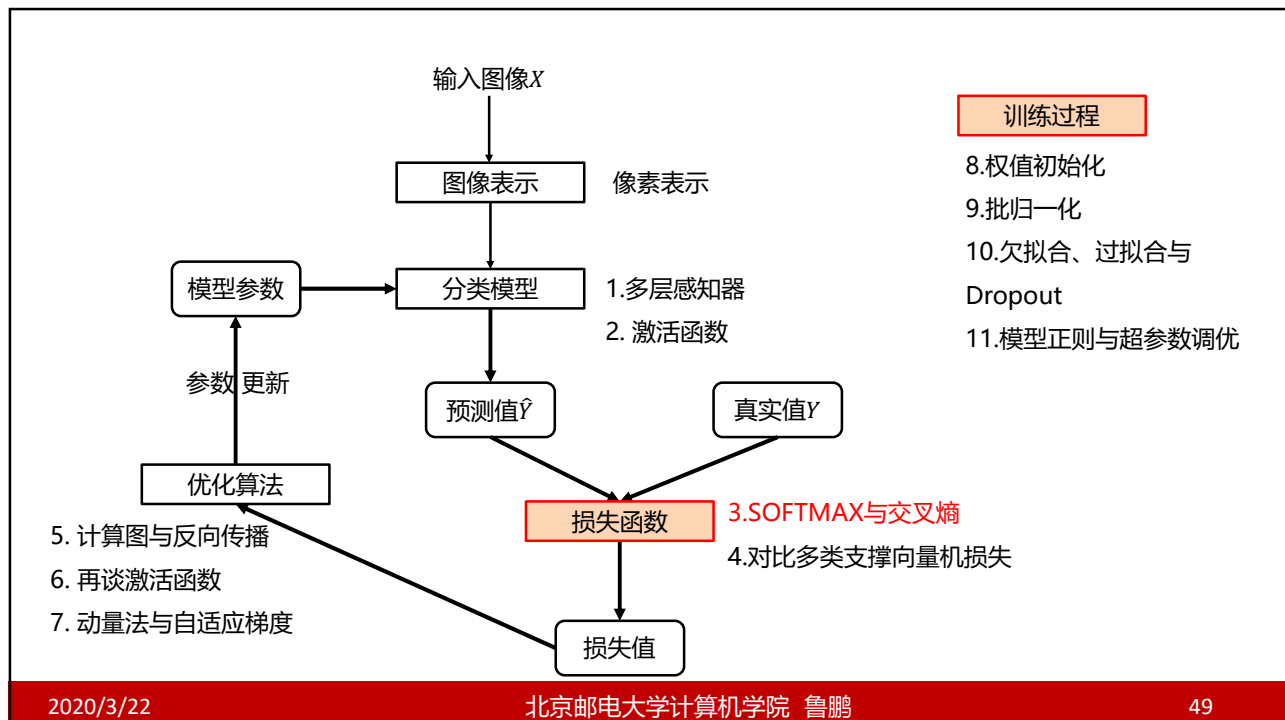
- 全连接神经网络组成：一个输入层、一个输出层及多个隐层；
- 输入层与输出层的神经元个数由任务决定，而隐层数量以及每个隐层的神经元个数需要人为指定；
- 激活函数是全连接神经网络中的一个重要部分，缺少了激活函数，全连接神经网络将退化为线性分类器。

2020/3/22

北京邮电大学计算机学院 鲁鹏

48

48



2020/3/22

北京邮电大学计算机学院 鲁鹏

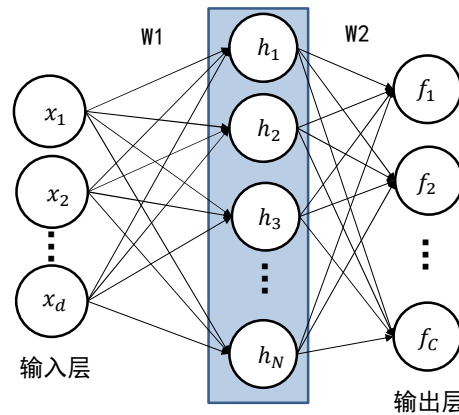
49

49

# SOFTMAX

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$



2020/3/22

北京邮电大学计算机学院 鲁鹏

50

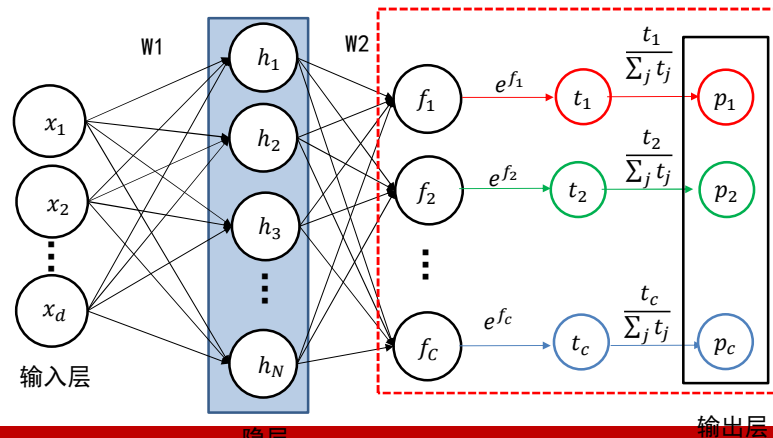
50

# SOFTMAX

两层全连接网络

$$f = W_2 \max(0, W_1 x + b_1) + b_2$$

Softmax



2020/3/22

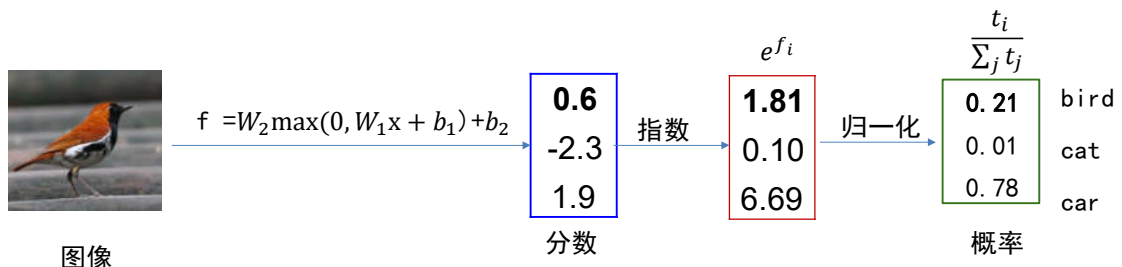
北京邮电大学计算机学院 鲁鹏

51

51

# SOFTMAX

## • 示例



2020/3/22

北京邮电大学计算机学院 鲁鹏

52

52

# 交叉熵损失

0.21
0.01
0.78

分类器预测分布 $q(x)$ 

2020/3/22

北京邮电大学计算机学院 鲁鹏

53

53

## 交叉熵损失

0.21
0.01
0.78

分类器预测分布 $q(x)$

1.00	bird
0.00	cat
0.00	car

真实分布 $p(x)$

2020/3/22

北京邮电大学计算机学院 鲁鹏

54

54

## 交叉熵损失

- 如何度量现在的分类器输出与预测值之间的距离？

0.21
0.01
0.78

分类器预测分布 $q(x)$

比较?

1.00	bird
0.00	cat
0.00	car

真实分布 $p(x)$

2020/3/22

北京邮电大学计算机学院 鲁鹏

55

55

## 交叉熵

熵:  $H(p) = -\sum_x p(x) \log p(x)$

交叉熵:  $H(p, q) = -\sum_x p(x) \log q(x)$

相对熵:  $KL(p||q) = -\sum_x p(x) \log \frac{q(x)}{p(x)}$

相对熵 (relative entropy) 也叫KL散度 (KL divergence); 用来度量两个分布之间的不相似性 (dissimilarity)。

三者之间关系:

$$\begin{aligned} H(p, q) &= -\sum_x p(x) \log q(x) \\ &= -\sum_x p(x) \log p(x) - \sum_x p(x) \log \frac{q(x)}{p(x)} \\ &= H(p) + KL(p||q) \end{aligned}$$

2020/3/22

北京邮电大学计算机学院 鲁鹏

56

56

## 交叉熵损失

- 如何度量现在的分类器输出与预测值之间的距离?

0.21	比较?	1.00	bird
0.01		0.00	cat
0.78		0.00	car

分类器预测分布  $q(x)$

真实分布  $p(x)$

如何衡量两个随机分布的差异

交叉熵:  $H(p, q) = -\sum_{i=1}^c p(x_i) \log(q(x_i))$

2020/3/22

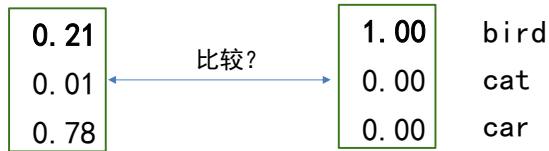
北京邮电大学计算机学院 鲁鹏

57

57

## 交叉熵损失

- 如何度量现在的分类器输出与预测值之间的距离？



分类器预测分布 $q(x)$

真实分布 $p(x)$

注：真实分布为one-hot形式时

交叉熵损失简化为：

$L_i = -\log(q_j)$ ，其中j为真实类别

如何衡量两个随机分布的差异

交叉熵： $H(p, q) = -\sum_{i=1}^C p(x_i) \log(q(x_i))$

2020/3/22

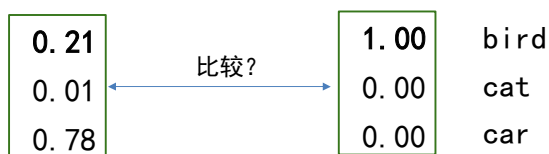
北京邮电大学计算机学院 鲁鹏

58

58

## 交叉熵损失

- 如何度量现在的分类器输出与预测值之间的距离？



分类器预测分布 $q(x)$

真实分布 $p(x)$

注：真实分布为one-hot形式时

交叉熵损失简化为：

$L_i = -\log(q_j)$ ，其中j为真实类别

如何衡量两个随机分布的差异

交叉熵： $H(p, q) = -\sum_{i=1}^C p(x_i) \log(q(x_i))$

交叉熵损失：

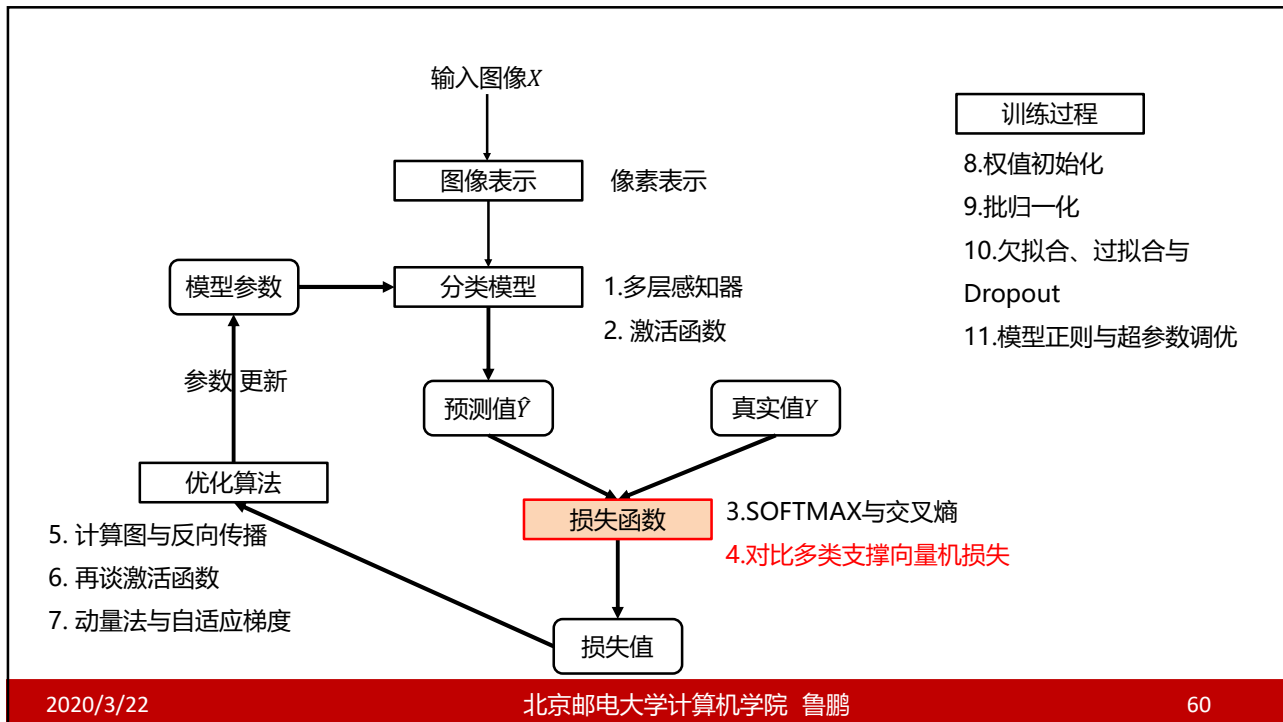
$$L_i = -\sum_{i=1}^C p(x_i) \log(q(x_i))$$

2020/3/22

北京邮电大学计算机学院 鲁鹏

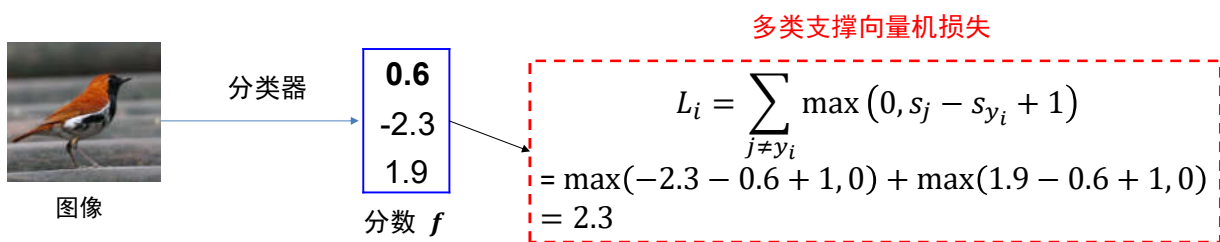
59

59



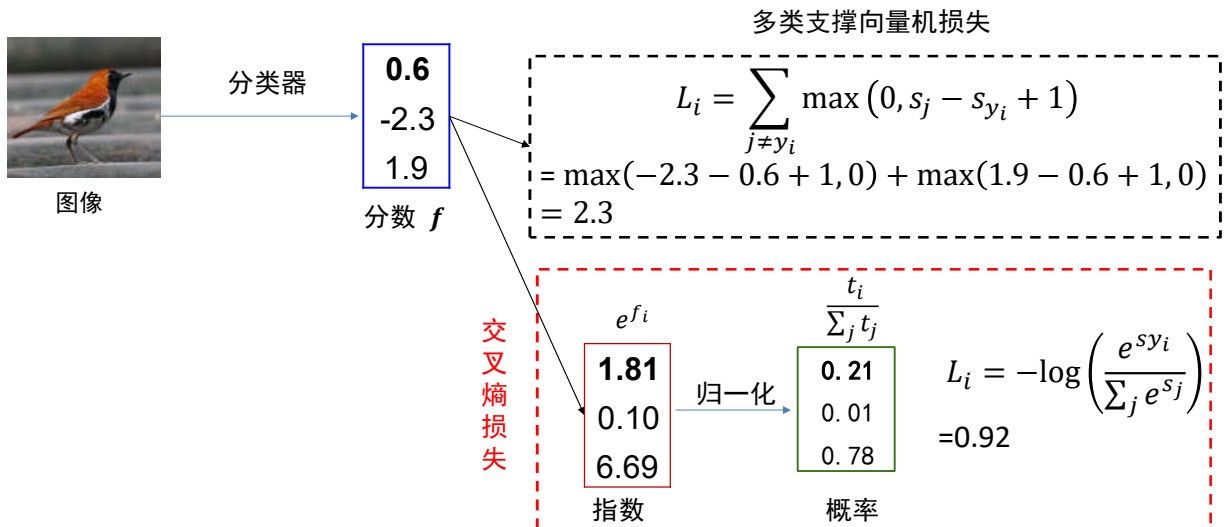
60

## 交叉熵损失 vs 多类支撑向量机损失



61

## 交叉熵损失 vs 多类支撑向量机损失



2020/3/22

北京邮电大学计算机学院 鲁鹏

62

62

## 交叉熵损失 vs 多类支撑向量机损失

假设分数  $f$   
 $y_i = 0$

$$L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

问： 相同分数下两种分类器的损失有什么区别？

2020/3/22

北京邮电大学计算机学院 鲁鹏

63

63



## 交叉熵损失 vs 多类支撑向量机损失

假设分数  $f$   
 $y_i = 0$

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

[10, -2, 3]

0.0004

0

[10, 9, 9]

0.2395

0

[10, -100, -100]

$1.5 \times 10^{-48}$

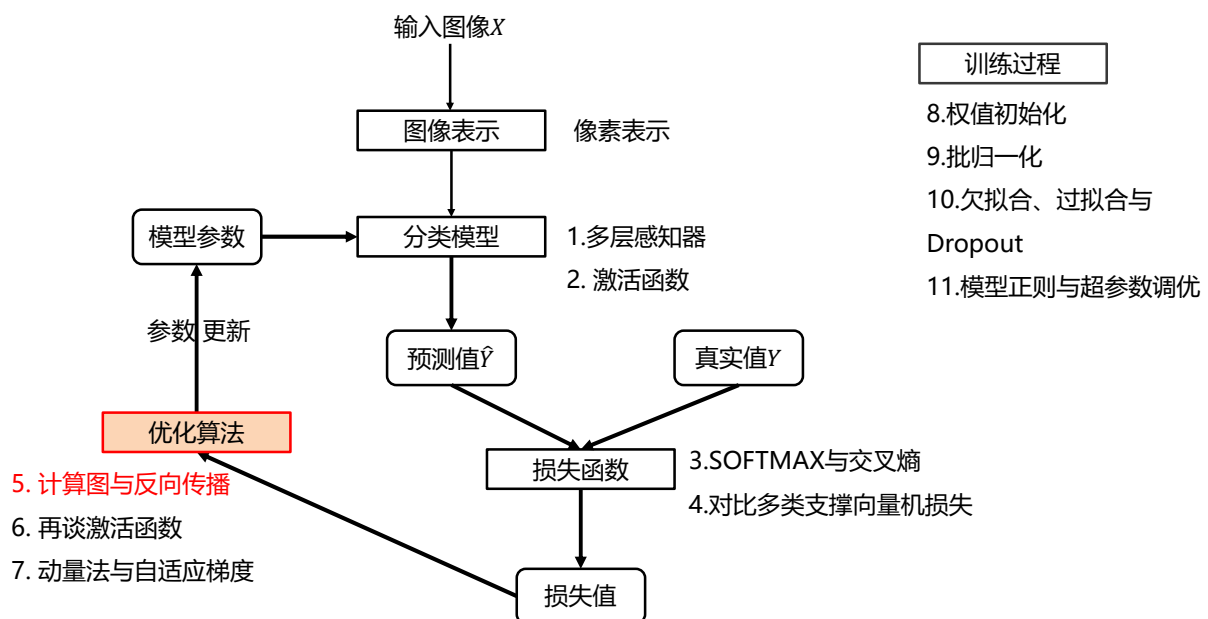
0

2020/3/22

北京邮电大学计算机学院 鲁鹏

64

64



2020/3/22

北京邮电大学计算机学院 鲁鹏

65

65

## 什么是计算图?

计算图是一种有向图，它用来表达输入、输出以及中间变量之间的计算关系，图中的每个节点对应着一种数学运算。

2020/3/22

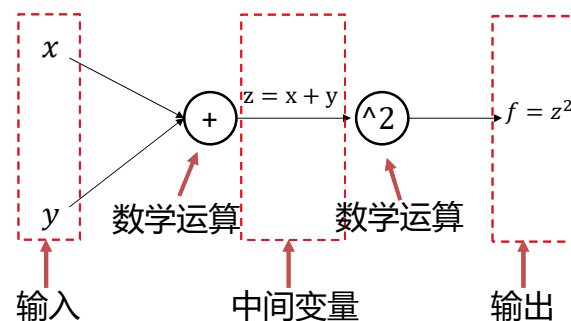
北京邮电大学计算机学院 鲁鹏

66

66

## 计算图与反向传播算法

函数  $f = (x + y)^2$  的计算图



2020/3/22

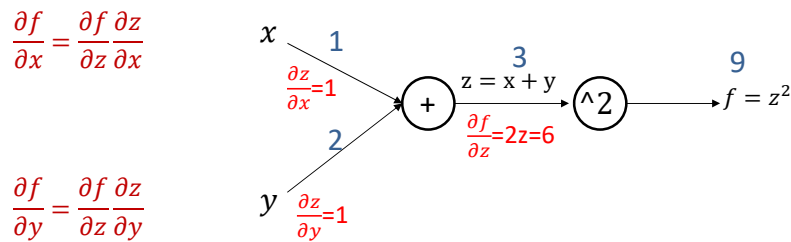
北京邮电大学计算机学院 鲁鹏

67

67

## 计算图的前反向计算

函数  $f = (x + y)^2$  的计算图



2020/3/22

北京邮电大学计算机学院 鲁鹏

68

68

## 计算图总结

- 任意复杂的函数，都可以用计算图的形式表示
- 在整个计算图中，每个门单元都会得到一些输入，然后，进行下面两个计算：
  - a) 这个门的输出值
  - b) 其输出值关于输入值的局部梯度。
- 利用链式法则，门单元应该将回传的梯度乘以它对其的输入的局部梯度，从而得到整个网络的输出对该门单元的每个输入值的梯度。

2020/3/22

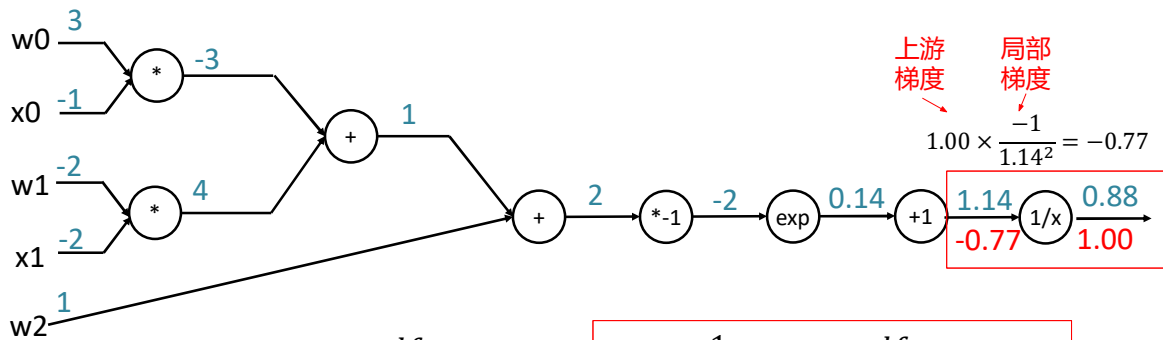
北京邮电大学计算机学院 鲁鹏

69

69

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

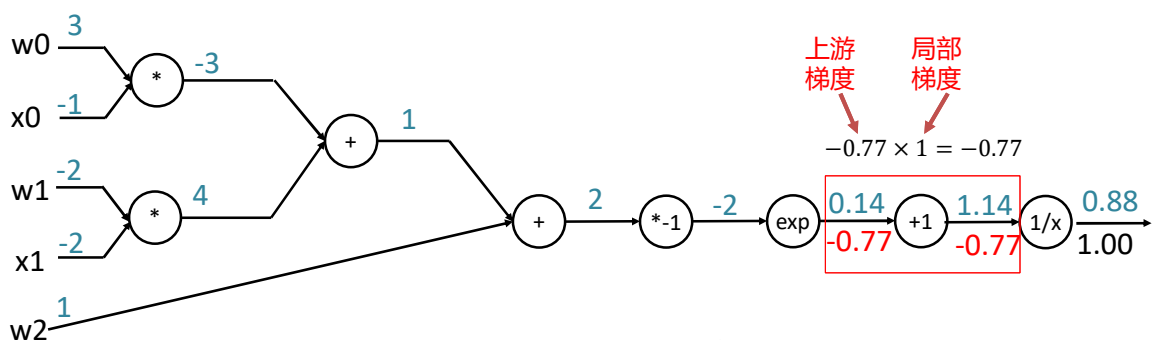
北京邮电大学计算机学院 鲁鹏

70

70

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

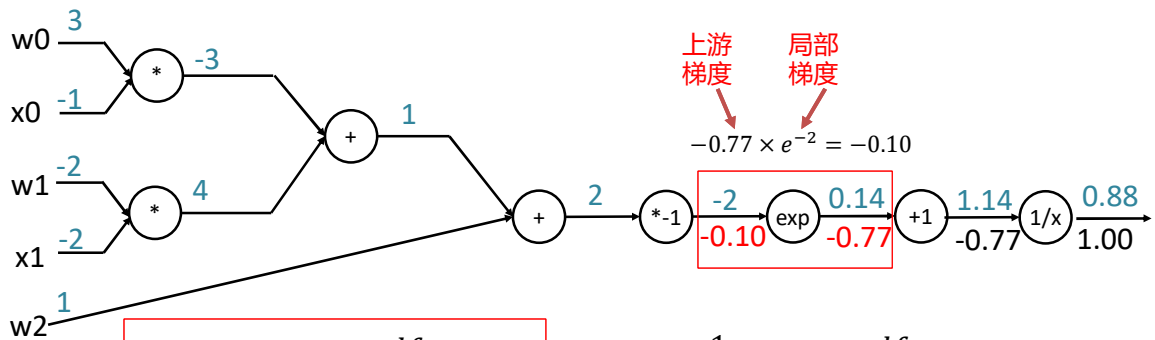
北京邮电大学计算机学院 鲁鹏

71

71

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

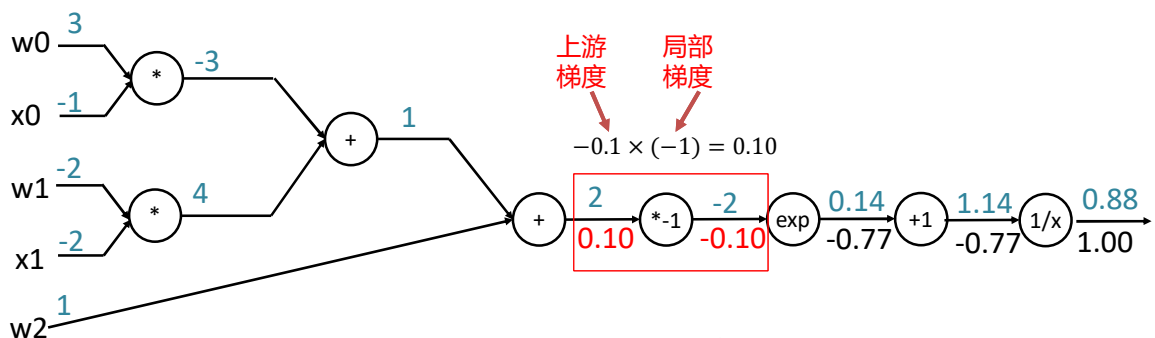
北京邮电大学计算机学院 鲁鹏

72

72

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

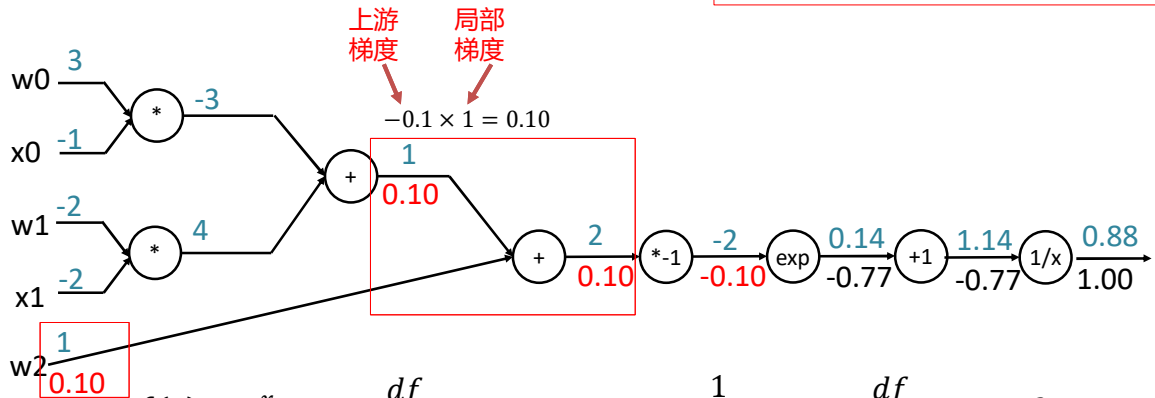
北京邮电大学计算机学院 鲁鹏

73

73

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a \quad f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

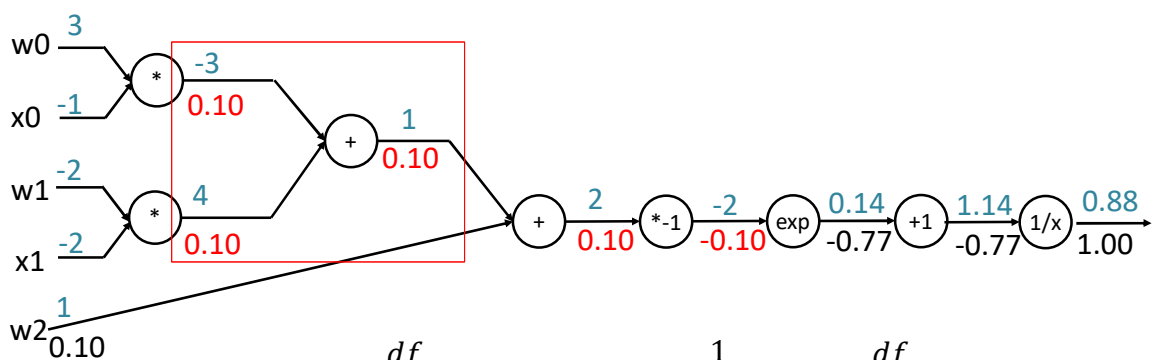
北京邮电大学计算机学院 鲁鹏

74

74

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x \quad f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a \quad f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

2020/3/22

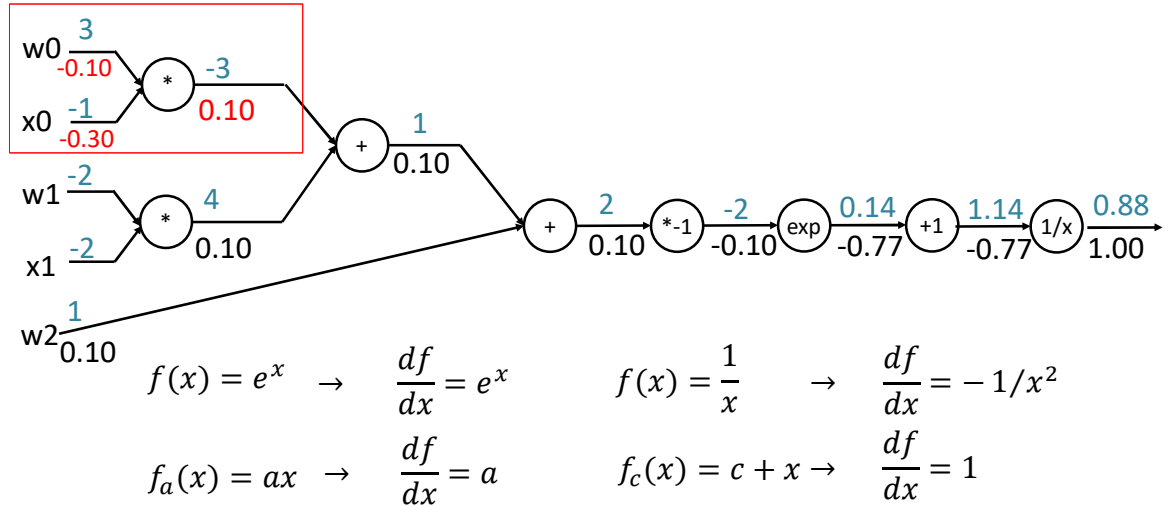
北京邮电大学计算机学院 鲁鹏

75

75

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



2020/3/22

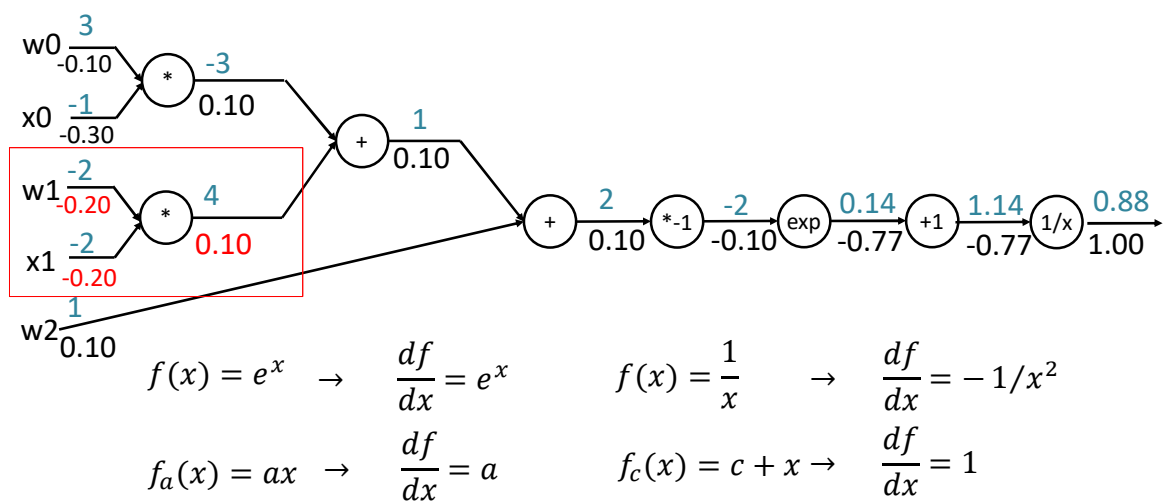
北京邮电大学计算机学院 鲁鹏

76

76

## 反向传播示例2

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



2020/3/22

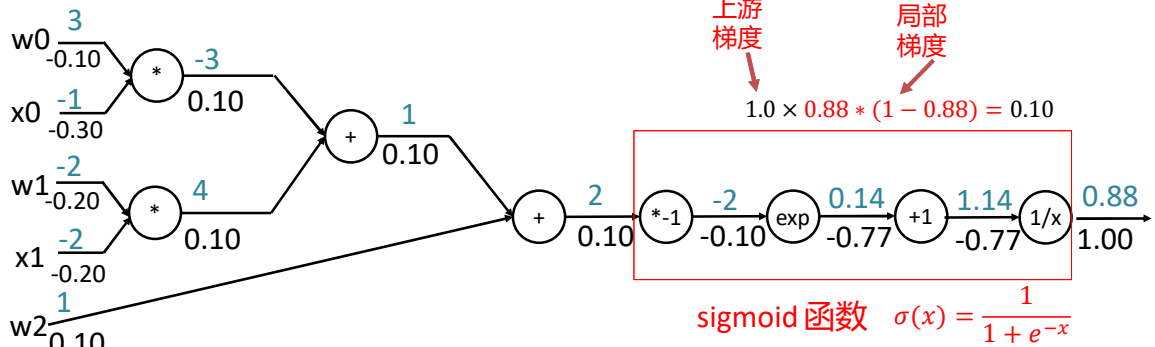
北京邮电大学计算机学院 鲁鹏

77

77

## 计算图的颗粒度

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2x_2)}}$$



$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$

2020/3/22

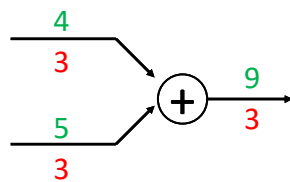
北京邮电大学计算机学院 鲁鹏

78

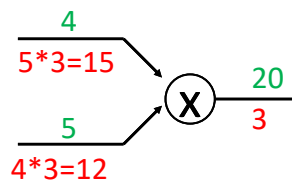
78

## 计算图中常见的门单元

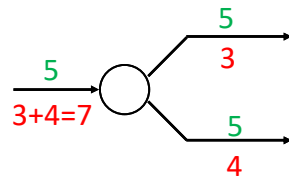
加法门



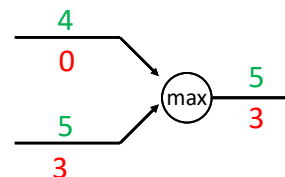
乘法门



拷贝门



max门



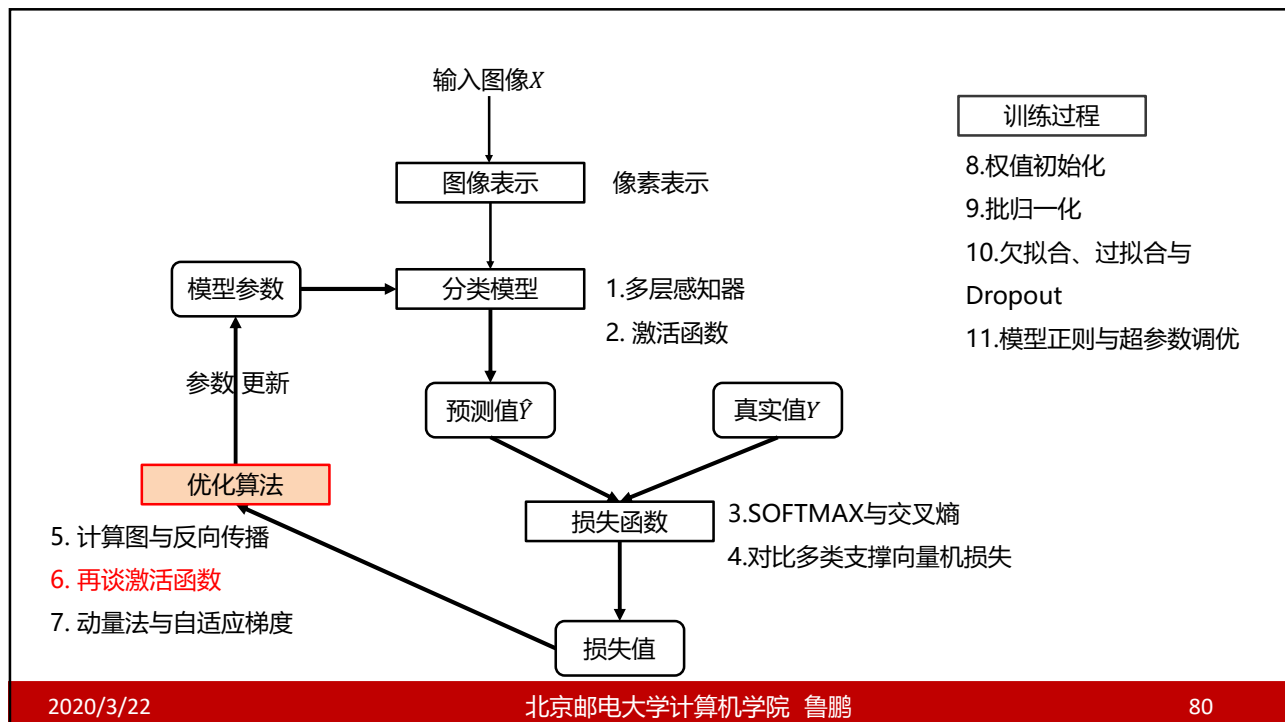
2020/3/22

北京邮电大学计算机学院 鲁鹏

79

79





80

## 再看激活函数

2020/3/22

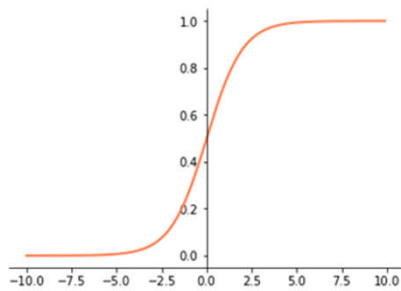
北京邮电大学计算机学院 鲁鹏

81

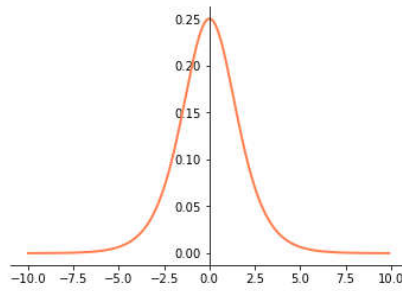
81

## 再看激活函数

Sigmoid激活函数:  $\sigma(x) = 1/(1 + e^{-x})$



sigmoid函数



Sigmoid函数的导数函数

当输入值大于10或者小于-10时局部梯度都是0；非常不利于网络的梯度流传递的。

2020/3/22

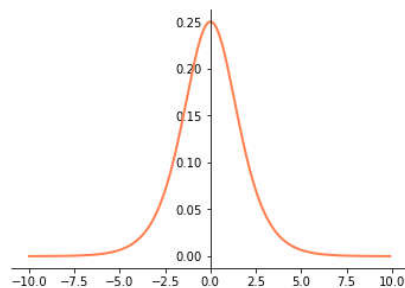
北京邮电大学计算机学院 鲁鹏

82

82

## 梯度消失

- 梯度消失是神经网络训练中非常致命的一个问题，其本质是由于链式法则的乘法特性导致的。



Sigmoid函数的导数函数

2020/3/22

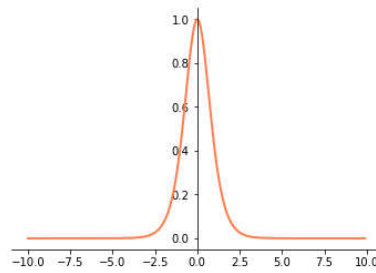
北京邮电大学计算机学院 鲁鹏

83

83

## 激活函数

双曲正切激活函数： $\tanh(x) = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



与sigmoid类似，局部梯度特性  
不利于网络梯度流的反向传递

tanh函数

tanh函数的导数函数

2020/3/22

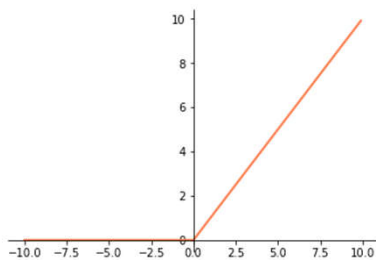
北京邮电大学计算机学院 鲁鹏

84

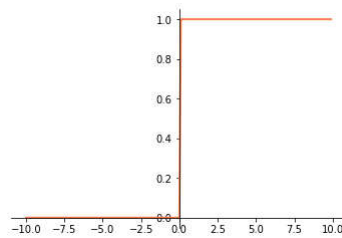
84

## 激活函数选择

ReLU激活函数： $f(x) = \max(0, x)$



ReLU函数



ReLU函数的导数函数

当输入大于0时，局部梯度永远  
不会为0，比较有利于梯度流的  
传递。

2020/3/22

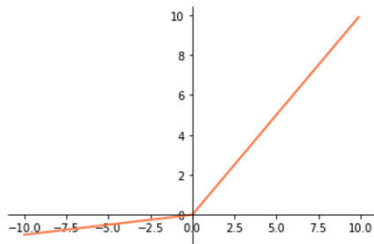
北京邮电大学计算机学院 鲁鹏

85

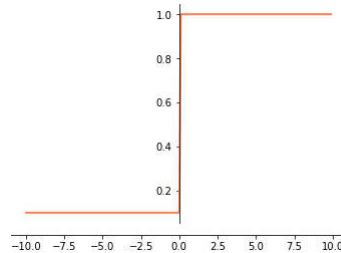
85

## 激活函数选择

Leaky ReLU激活函数:  $f(x) = \max(0.01x, x)$



ReLU函数



ReLU函数的导数函数

基本没有“死区”，也就是梯度永远不会为0。之所以说“基本”，是因为函数在0处没有导数。

2020/3/22

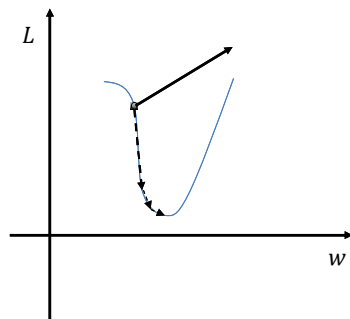
北京邮电大学计算机学院 鲁鹏

86

86

## 梯度爆炸

- 梯度爆炸也是由于链式法则的乘法特性导致的。



**梯度爆炸**: 断崖处梯度乘以学习率后

会是一个非常大得值，从而“飞”出了合理区域，最终导致算法不收敛；

**解决方案**: 把沿梯度方向前进的步长

限制在某个值内就可以避免“飞”出

了，这个方法也称为梯度裁剪。

2020/3/22

北京邮电大学计算机学院 鲁鹏

87

87

## 激活函数选择总结

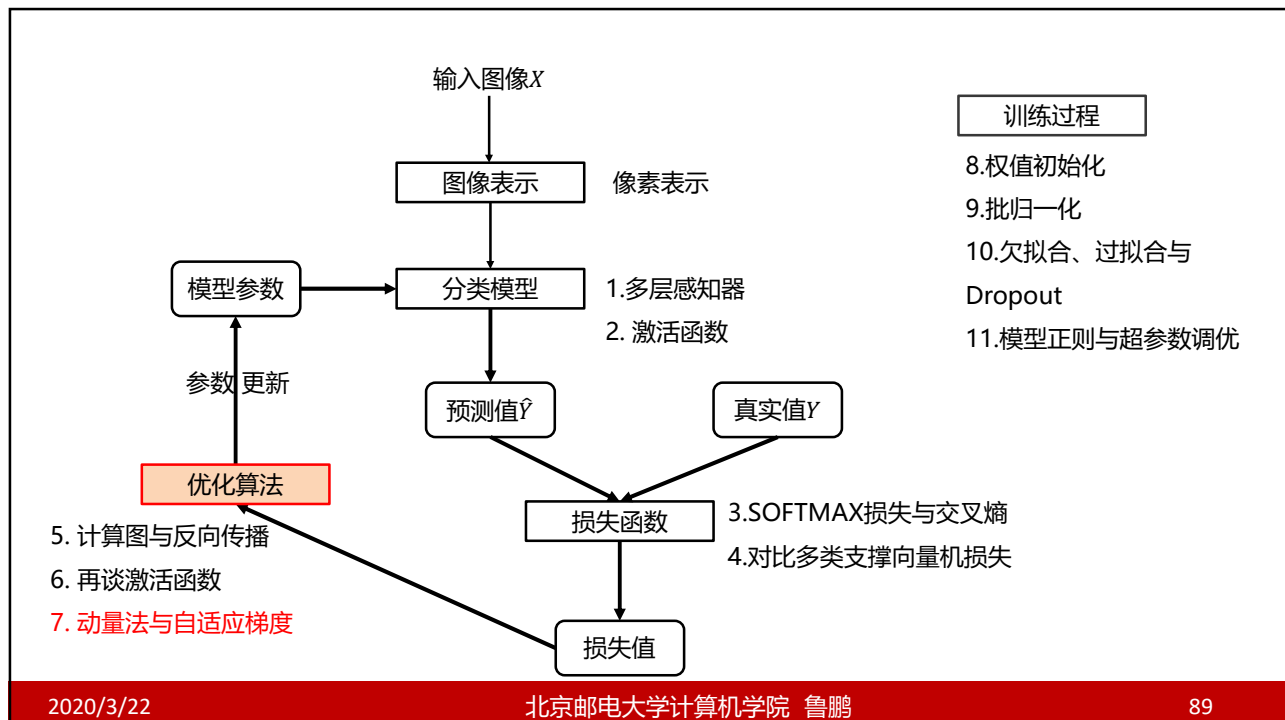
尽量选择ReLU函数或者Leaky ReLU函数，相对于Sigmoid/tanh，ReLU函数或者Leaky ReLU函数会让梯度流更加顺畅，训练过程收敛得更快。

2020/3/22

北京邮电大学计算机学院 鲁鹏

88

88



2020/3/22

北京邮电大学计算机学院 鲁鹏

89

89

## 梯度算法改进

- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

北京邮电大学计算机学院 鲁鹏

90

90

## 梯度算法改进

- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

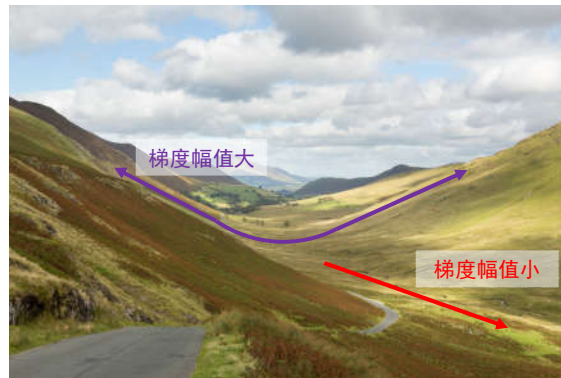
北京邮电大学计算机学院 鲁鹏

91

91

## 梯度下降法存在的问题

- 损失函数特性：一个方向上变化迅速而在另一个方向上变化缓慢。



2020/3/22

北京邮电大学计算机学院 鲁鹏

92

92

## 梯度下降法存在的问题

- 损失函数特性：一个方向上变化迅速而在另一个方向上变化缓慢。
- 优化目标：从起点处走到笑脸处



2020/3/22

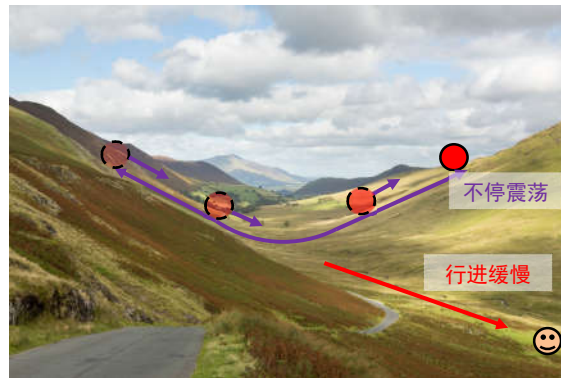
北京邮电大学计算机学院 鲁鹏

93

93

## 梯度下降法存在的问题

- 损失函数特性：一个方向上变化迅速而在另一个方向上变化缓慢。
- 优化目标：从起点处走到底端笑脸处
- 梯度下降算法存在的问题：山壁间震荡，往谷低方向的行进较慢。



2020/3/22

北京邮电大学计算机学院 鲁鹏

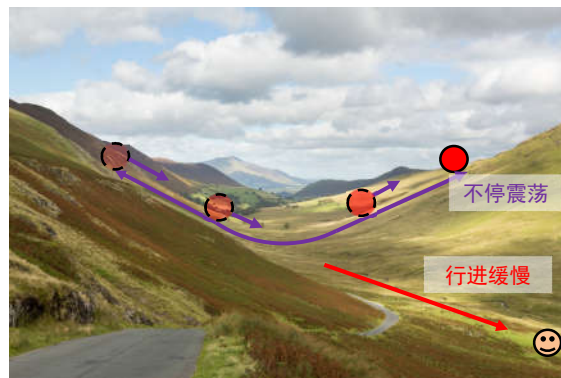
94

94

## 梯度下降法存在的问题

- 损失函数特性：一个方向上变化迅速而在另一个方向上变化缓慢。
- 优化目标：从起点处走到底端笑脸处
- 梯度下降算法存在的问题：山壁间震荡，往谷低方向的行进较慢。

仅增大步长并  
不能加快算法收敛  
速度！



2020/3/22

北京邮电大学计算机学院 鲁鹏

95

95



## 梯度算法改进

- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

北京邮电大学计算机学院 鲁鹏

96

96

## 动量法

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低

2020/3/22

北京邮电大学计算机学院 鲁鹏

97

97

## 动量法

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

2020/3/22

北京邮电大学计算机学院 鲁鹏

98

98

## 动量法

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

### 使用动量的小批量梯度下降算法

Require: 学习率  $\epsilon$ , 动量系数  $\mu$ Require: 初始参数  $\theta$ 初始化速度  $v = 0$ 

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 速度更新:  $v = \mu v + (1 - \mu)g$
4. 更新权值:  $\theta \leftarrow \theta - \epsilon v$

end while

### 小批量梯度下降算法

Require: 学习率  $\epsilon$ Require: 初始参数  $\theta$ 

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

99

99

## 动量法

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

### 使用动量的小批量梯度下降算法

Require: 学习率  $\epsilon$ , 动量系数  $\mu$

Require: 初始参数  $\theta$

初始化速度  $v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 速度更新:  $v = \mu v + (1 - \mu)g$
4. 更新权值:  $\theta \leftarrow \theta - \epsilon v$

end while

### 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

100

100

## 动量法

为什么有效?

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

### 使用动量的小批量梯度下降算法

Require: 学习率  $\epsilon$ , 动量系数  $\mu$

Require: 初始参数  $\theta$

初始化速度  $v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 速度更新:  $v = \mu v + (1 - \mu)g$
4. 更新权值:  $\theta \leftarrow \theta - \epsilon v$

end while

### 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

101

101

## 动量法

为什么有效？

累加过程中震荡方向相互抵消，  
平坦方向得到加强

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

### 使用动量的小批量梯度下降算法

Require: 学习率  $\epsilon$ ，动量系数  $\mu$

Require: 初始参数  $\theta$

初始化速度  $v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 速度更新:  $v = \mu v + (1 - \mu)g$
4. 更新权值:  $\theta \leftarrow \theta - \epsilon v$

end while

### 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

102

102

## 动量法

为什么有效？

累加过程中震荡方向相互抵消，  
平坦方向得到加强

- 目标：改进梯度下降算法存在的问题，即减少震荡，加速通往谷低
- 改进思想：利用累加历史梯度信息更新梯度

### 使用动量的小批量梯度下降算法

Require: 学习率  $\epsilon$ ，动量系数  $\mu$

Require: 初始参数  $\theta$

初始化速度  $v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 速度更新:  $v = \mu v + (1 - \mu)g$
4. 更新权值:  $\theta \leftarrow \theta - \epsilon v$

end while

$\mu$  取值范围  $[0, 1)$

$\mu = 0$  时等价于梯度下降算法

建议设置: 0.9

### 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

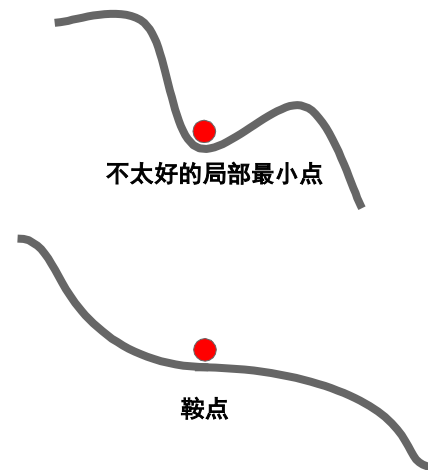
北京邮电大学计算机学院 鲁鹏

103

103

## 动量法还有什么效果？

现象：损失函数常具有不太好的**局部最小值**  
或**鞍点**（高维空间非常常见）



2020/3/22

北京邮电大学计算机学院 鲁鹏

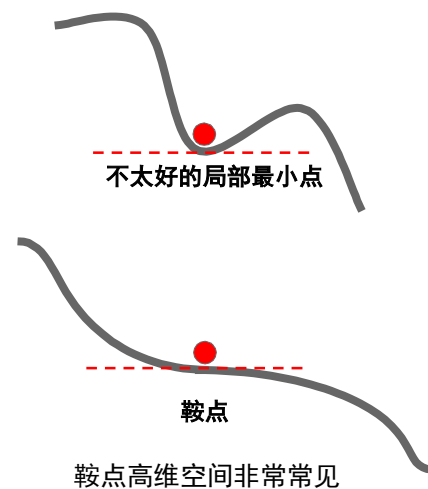
104

104

## 动量法还有什么效果？

现象：损失函数常具有不太好的**局部最小值**  
或**鞍点**（高维空间非常常见）

**梯度下降算法存在的问题：**局部最小处与鞍点处梯度为0，算法无法通过。



2020/3/22

北京邮电大学计算机学院 鲁鹏

105

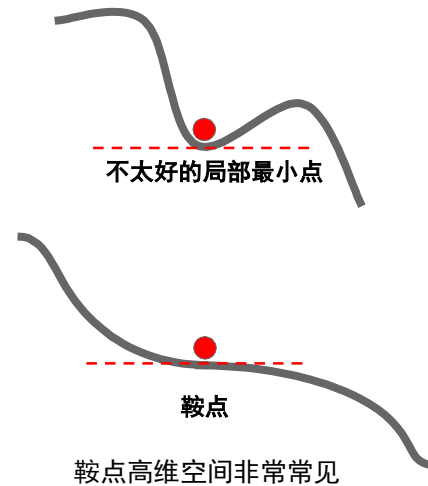
105

## 动量法还有什么效果？

现象：损失函数常具有不太好的**局部最小值**或**鞍点**（高维空间非常常见）

**梯度下降算法存在的问题：**局部最小处与鞍点处梯度为0，算法无法通过。

**动量法的优势：**由于动量的存在，算法可以冲出局部最小点以及鞍点，找到更优的解。



2020/3/22

北京邮电大学计算机学院 鲁鹏

106

106

## 梯度算法改进

- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

北京邮电大学计算机学院 鲁鹏

107

107

## 自适应梯度法



2020/3/22

北京邮电大学计算机学院 鲁鹏

108

108

## 自适应梯度法

- 自适应梯度法通过减小震荡方向步长，增大平坦方向步长来减小震荡，加速通往谷底方向；



2020/3/22

北京邮电大学计算机学院 鲁鹏

109

109

## 自适应梯度法

- 自适应梯度法通过减小震荡方向步长，增大平坦方向步长来减小震荡，加速通往谷底方向；
- 如何区分震荡方向与平坦方向？



2020/3/22

北京邮电大学计算机学院 鲁鹏

110

110

## 自适应梯度法

- 自适应梯度法通过减小震荡方向步长，增大平坦方向步长来减小震荡，加速通往谷底方向；
- 如何区分震荡方向与平坦方向？

回答：梯度幅度的平方较大的方向是震荡方向；梯度幅度的平方较小的方向是平坦方向。



2020/3/22

北京邮电大学计算机学院 鲁鹏

111

111



# RMSProp

RMSProp方法是一种自适应梯度方法

## RMSProp

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为  $10^{-5}$ )

初始化累积变量  $r = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$

2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

3. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho) g * g$

4. 更新权值:  $w \leftarrow w - \frac{\epsilon}{\sqrt{r} + \delta} g$

end while

## 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$

对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$

2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

112

112

# RMSProp

RMSProp方法是一种自适应梯度方法

## RMSProp

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为  $10^{-5}$ )

初始化累积变量  $r = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$   
对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$

2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

3. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho) g * g$

4. 更新权值:  $w \leftarrow w - \frac{\epsilon}{\sqrt{r} + \delta} g$

end while

## 小批量梯度下降算法

Require: 学习率  $\epsilon$

Require: 初始参数  $\theta$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$

对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$

2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

3. 更新权值:  $\theta \leftarrow \theta - \epsilon g$

end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

113

113

## RMSProp

RMSProp方法是一种自适应梯度方法

RMSProp	$\rho$ 取值范围 $[0, 1)$ $\rho = 0$ 时仅考虑当前梯度的强度 建议设置: 0.999	小批量梯度下降算法
Require: 学习率 $\epsilon$ , 衰减速率 $\rho$ Require: 初始参数 $\theta$ Require: 小常数 $\delta$ , 用于被小数除时的数值稳定。(通常设为 $10^{-5}$ ) 初始化累积变量 $r = 0$ while 停止标准未满足 do: 1. 从训练集中采样 $m$ (批量大小) 个样本 $\{x^{(1)}, \dots, x^{(m)}\}$ 对应的目标为 $\{y^{(1)}, \dots, y^{(m)}\}$ 2. 计算梯度: $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$ 3. 累积平方梯度: $r \leftarrow \rho r + (1 - \rho) g * g$ 4. 更新权值: $w \leftarrow w - \frac{\epsilon}{\sqrt{r} + \delta} g$ end while		Require: 学习率 $\epsilon$ Require: 初始参数 $\theta$ while 停止标准未满足 do: 1. 从训练集中采样 $m$ (批量大小) 个样本 $\{x^{(1)}, \dots, x^{(m)}\}$ 对应的目标为 $\{y^{(1)}, \dots, y^{(m)}\}$ 2. 计算梯度: $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$ 3. 更新权值: $\theta \leftarrow \theta - \epsilon g$ end while

2020/3/22

北京邮电大学计算机学院 鲁鹏

114

114

## 梯度算法改进

- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

北京邮电大学计算机学院 鲁鹏

115

115

# Adam

- 同时使用动量与自适应梯度思想

2020/3/22

北京邮电大学计算机学院 鲁鹏

116

116

# Adam

## Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。(建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为  $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

- 同时使用动量与自适应梯度思想

2020/3/22

北京邮电大学计算机学院 鲁鹏

117

117

## Adam

### Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。 (建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。 (通常设为 $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

➤ 同时使用动量与自适应梯度思想

2020/3/22

北京邮电大学计算机学院 鲁鹏

118

118

## Adam

### Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。 (建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。 (通常设为 $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

➤ 同时使用动量与自适应梯度思想

2020/3/22

北京邮电大学计算机学院 鲁鹏

119

119

## Adam

### Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。(建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为 $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

➤ 同时使用动量与自适应梯度思想

2020/3/22

北京邮电大学计算机学院 鲁鹏

120

120

## Adam

### Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。(建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为 $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

➤ 同时使用动量与自适应梯度思想

➤ 修正偏差步骤可以极大缓解算法初期的冷启动问题

2020/3/22

北京邮电大学计算机学院 鲁鹏

121

121

## Adam

### Adam算法

Require: 学习率  $\epsilon$ , 衰减速率  $\rho$ , 动量系数  $\mu$ 。(建议值分别为0.999, 0.9)

Require: 初始参数  $\theta$

Require: 小常数  $\delta$ , 用于被小数除时的数值稳定。(通常设为 $10^{-5}$ )

初始化累积变量  $r = 0, v = 0$

while 停止标准未满足 do:

1. 从训练集中采样  $m$  (批量大小) 个样本  $\{x^{(1)}, \dots, x^{(m)}\}$  对应的目标为  $\{y^{(1)}, \dots, y^{(m)}\}$
2. 计算梯度:  $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
3. 累积梯度:  $v \leftarrow \mu v + (1 - \mu)g$
4. 累积平方梯度:  $r \leftarrow \rho r + (1 - \rho)g * g$
5. 修正偏差:  $\tilde{v} = \frac{v}{1 - \mu^t}$ ;  $\tilde{r} = \frac{r}{1 - \rho^t}$
6. 更新权值:  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\tilde{r} + \delta}} \tilde{v}$

end while

- 同时使用动量与自适应梯度思想
- 修正偏差步骤可以极大缓解算法初期的冷启动问题

2020/3/22

北京邮电大学计算机学院 鲁鹏

122

122

## 梯度算法改进

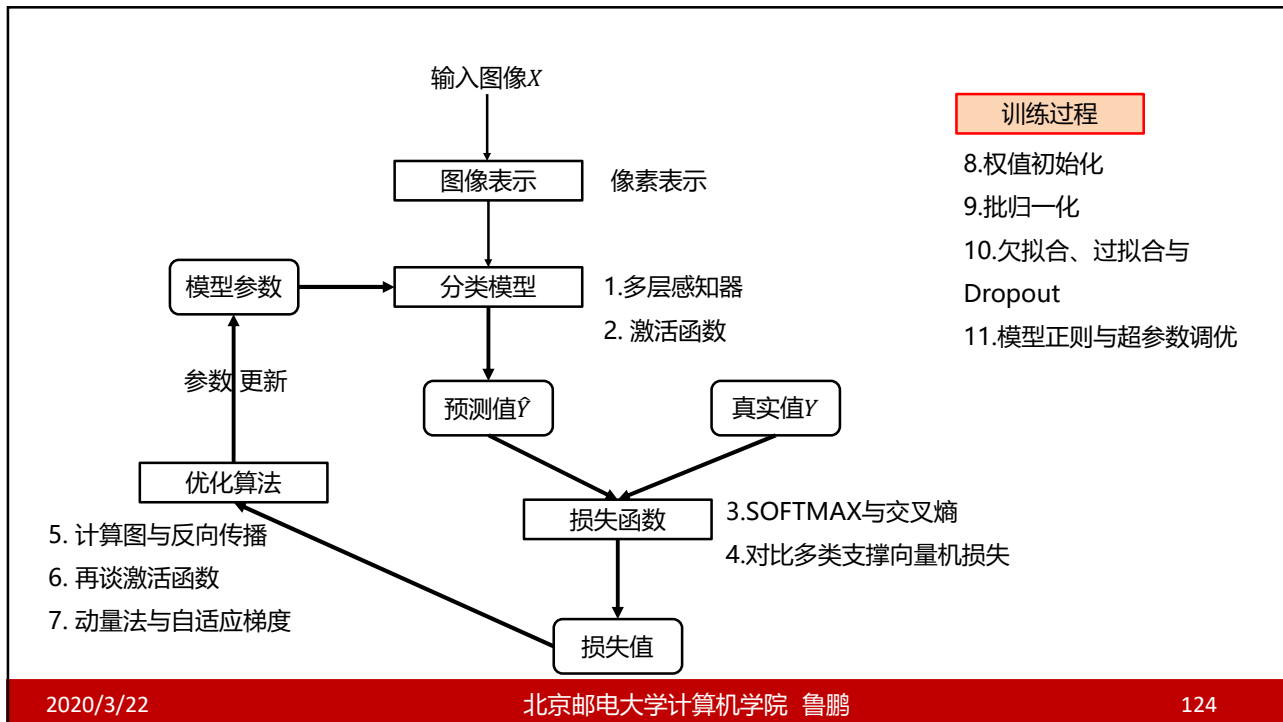
- 梯度下降算法存在的问题
- 动量法
- 自适应梯度与RMSProp
- ADAM
- 总结

2020/3/22

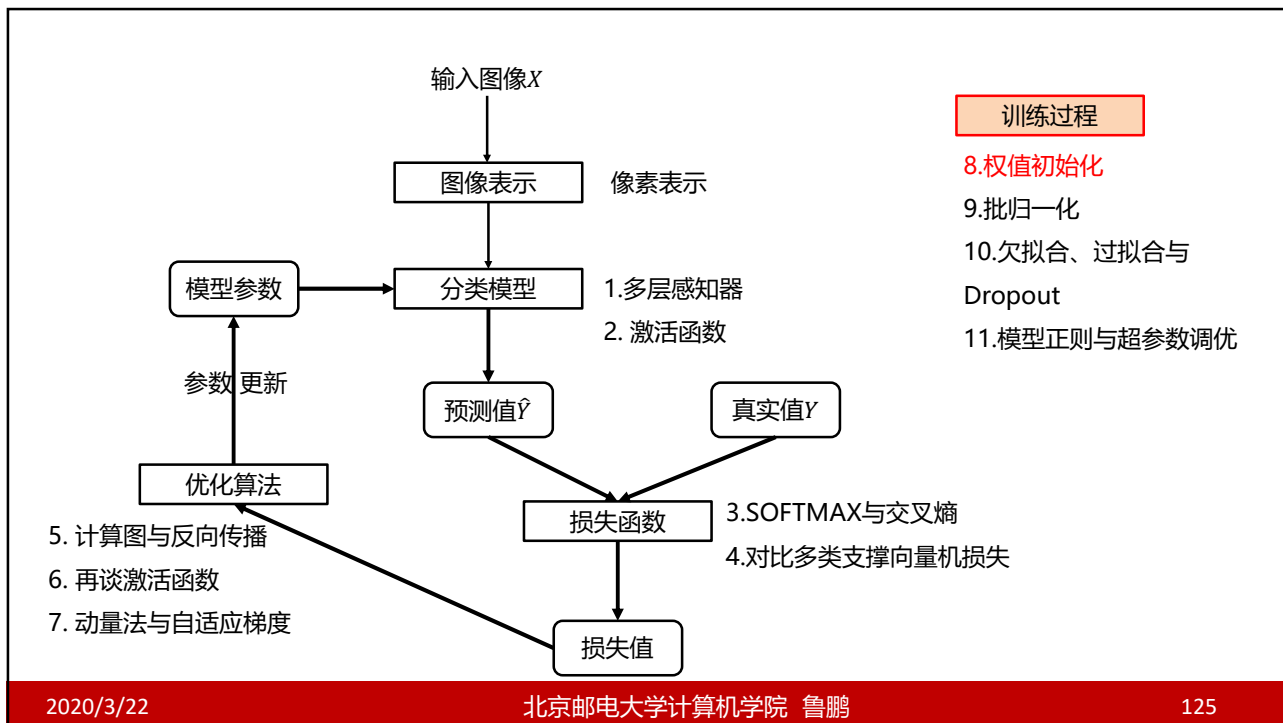
北京邮电大学计算机学院 鲁鹏

123

123

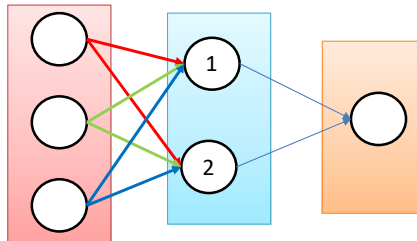


124



125

## 权值初始化



全零初始化：网络中不同的神经元有相同的输出，进行同样的参数更新；  
因此，这些神经元学到的参数都一样，等价于一个神经元。

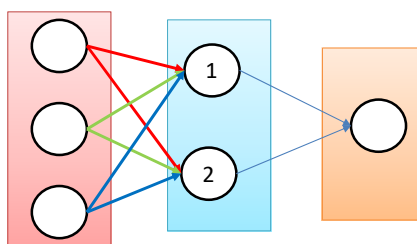
2020/3/22

北京邮电大学计算机学院 鲁鹏

126

126

## 全零初始化



建议：采用随机初始化，  
避免全零初始化！

全零初始化：网络中不同的神经元有相同的输出，进行同样的参数更新；  
因此，这些神经元学到的参数都一样，等价于一个神经元。

2020/3/22

北京邮电大学计算机学院 鲁鹏

127

127



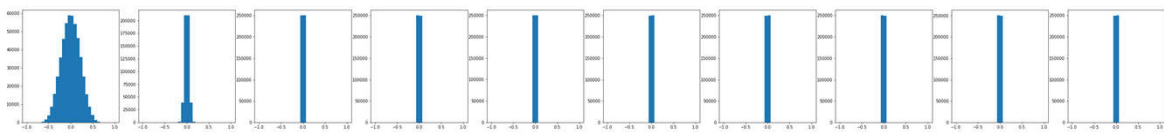
## 随机权值初始化

网络结构：10个隐层，1个输出层，每个隐层包含500个神经元，使用的双曲正切激活函数。

随机初始化：

权值采样自  $N(0,0.01)$  的高斯分布

```
input layer had mean 0.000097 and std 0.999691
hidden layer 1 mean -0.000134 and std 0.214021
hidden layer 2 mean 0.000025 and std 0.047655
hidden layer 3 mean -0.000030 and std 0.010683
hidden layer 4 mean -0.000002 and std 0.002390
hidden layer 5 mean -0.000000 and std 0.000532
hidden layer 6 mean -0.000000 and std 0.000119
hidden layer 7 mean -0.000000 and std 0.000027
hidden layer 8 mean 0.000000 and std 0.000006
hidden layer 9 mean 0.000000 and std 0.000001
hidden layer 10 mean 0.000000 and std 0.000000
```



除了前两层，后续所有层的激活值为0；此时，输入信息传递不到输出层；最终，网络得不到训练。

2020/3/22

北京邮电大学计算机学院 鲁鹏

128

128

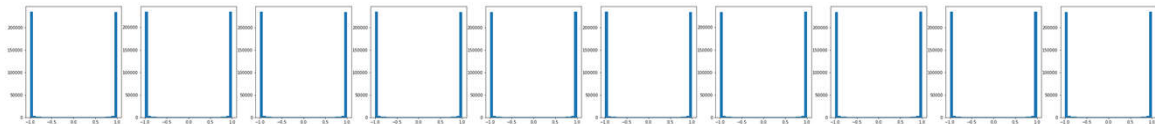
## 随机权值初始化

网络结构：10个隐层，1个输出层，每个隐层包含500个神经元，使用的双曲正切激活函数。

随机初始化：

权值采样自  $N(0,1)$  的高斯分布

```
input layer had mean 0.000240 and std 1.000121
hidden layer 1 mean -0.003320 and std 0.982045
hidden layer 2 mean 0.002391 and std 0.981602
hidden layer 3 mean 0.000433 and std 0.981700
hidden layer 4 mean 0.000040 and std 0.981740
hidden layer 5 mean 0.000358 and std 0.981791
hidden layer 6 mean -0.000324 and std 0.981612
hidden layer 7 mean 0.002807 and std 0.981484
hidden layer 8 mean -0.001022 and std 0.981578
hidden layer 9 mean 0.000497 and std 0.981684
hidden layer 10 mean -0.001396 and std 0.981915
```



几乎所有的神经元都饱和了（不是-1就是1）；此时，神经元局部梯度都是零，网络没有反向梯度流；最终，所有的参数得不到更新

2020/3/22

北京邮电大学计算机学院 鲁鹏

129

129

## 随机权值初始化

- 实验结论：初始化时让权值不相等，并不能保证网络能够正常的被训练。
- 有效的初始化方法：使网络各层的激活值和局部梯度的方差在传播过程中尽量保持一致；以保持网络中正向和反向数据流动。

2020/3/22

北京邮电大学计算机学院 鲁鹏

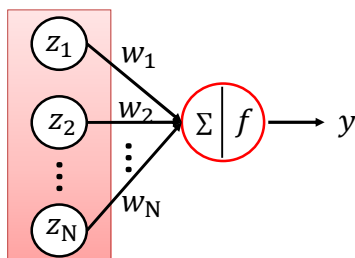
130

130

## Xavier初始化

一个神经元，其输入为 $z_1, z_2, \dots, z_N$ ，这 $N$ 个输入是独立同分布的；其权值为 $w_1, \dots, w_N$ ，它们也是独立同分布的，且 $w$ 与 $z$ 是独立的；其激活函数为 $f$ ；其最终输出 $y$ 的表达式：

$$y = f(w_1 * z_1 + \dots + w_N * z_N)$$



目标：使网络各层的激活值和局部梯度的方差在传播过程中尽量保持一致，即寻找  $w$  的分布使得输出  $y$  与输入  $z$  的方差一致

2020/3/22

北京邮电大学计算机学院 鲁鹏

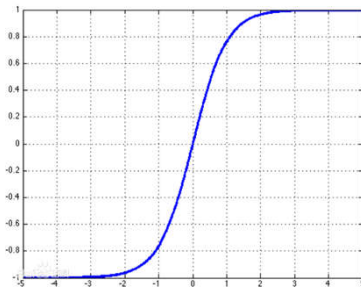
131

131

## Xavier初始化

一个神经元，其输入为 $z_1, z_2, \dots, z_N$ ，这 $N$ 个输入是独立同分布的；其权值为 $w_1, \dots, w_N$ ，它们也是独立同分布的，且 $w$ 与 $z$ 是独立的；其激活函数为 $f$ ；其最终输出 $y$ 的表达式：

$$y = f(w_1 * z_1 + \dots + w_N * z_N)$$



假设 $f$ 为双曲正切函数， $w_1, \dots, w_N$ 独立同分布， $z_1, \dots, z_N$ 独立同分布，随机变量 $w$ 与 $z$ 独立，且均值都为0，则有：

$$\begin{aligned} \text{Var}(y) &= \text{Var}\left(\sum_{i=1}^n w_i z_i\right) = \sum_{i=1}^n \text{Var}(w_i z_i) \\ &= \sum_{i=1}^n [E(w_i)]^2 \text{Var}(z_i) + [E(z_i)]^2 \text{Var}(w_i) + \text{Var}(w_i) \text{Var}(z_i) \\ &= \sum_{i=1}^n \text{Var}(w_i) \text{Var}(z_i) \\ &= n \text{Var}(w_i) \text{Var}(z_i) \end{aligned}$$

$\text{var}(w) = 1/N$ 时， $y$ 的方差与 $z$ 的方差一致。

2020/3/22

北京邮电大学计算机学院 鲁鹏

132

132

## Xavier初始化

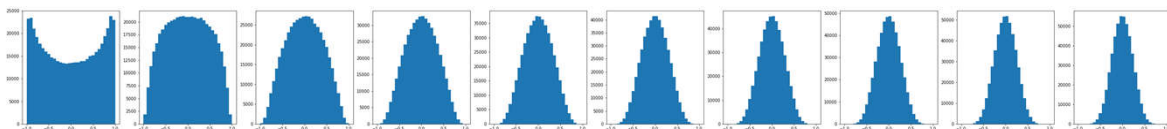
网络结构：10个隐层，1个输出层，每个隐层包含500个神经元，使用的双曲正切激活函数。

随机初始化：

权值采样自  $\mathcal{N}(0, 1/N)$  的高斯分布， $N$ 为输入神经元个数

达到目的：

每层神经元激活值的方差基本相同！



2020/3/22

北京邮电大学计算机学院 鲁鹏

133

133

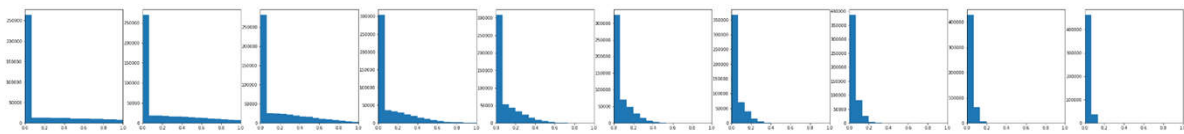
## Xavier初始化

网络结构：10个隐层，1个输出层，每个隐层包含500个神经元，使用的ReLU激活函数。

随机初始化：

权值采样自  $\mathcal{N}(0, 1/N)$  的高斯分布,  $N$ 为输入神经元个数

Xavier初始化方法不太适合ReLU激活函数！



2020/3/22

北京邮电大学计算机学院 鲁鹏

134

134

## HE初始化(MSRA)

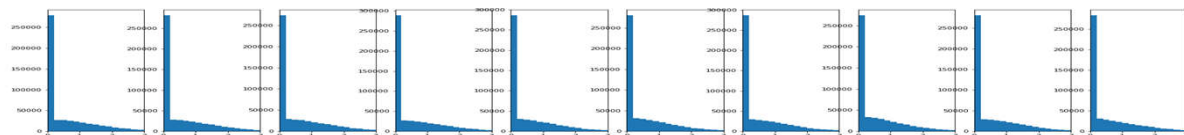
网络结构：10个隐层，1个输出层，每个隐层包含500个神经元，使用的ReLU激活函数。

随机初始化：

权值采样自  $\mathcal{N}(0, 2/N)$  的高斯分布,  $N$ 为输入神经元个数

ReLU激活函数初始化方法

修改此处



2020/3/22

北京邮电大学计算机学院 鲁鹏

135

135

## 权值初始化小结

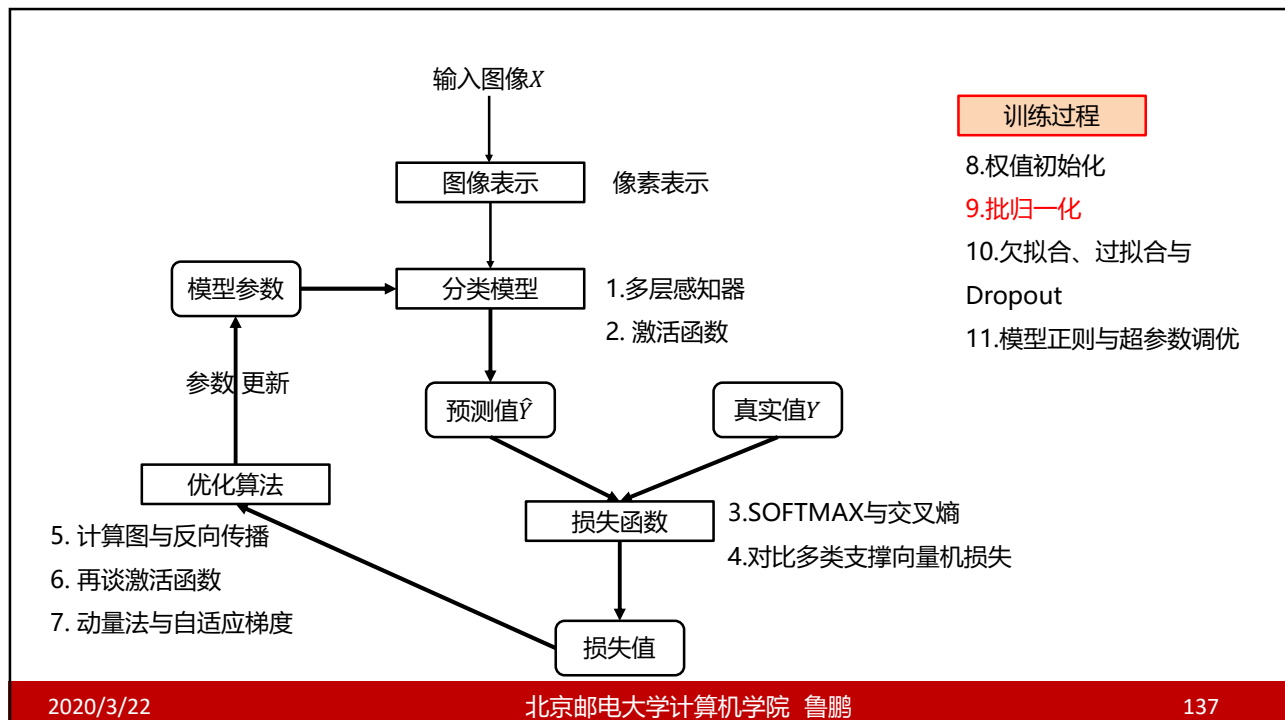
- 好的初始化方法可以防止前向传播过程中的信息消失，也可以解决反向传递过程中的梯度消失。
- 激活函数选择双曲正切或者Sigmoid时，建议使用Xaizer初始化方法；
- 激活函数选择ReLU或Leakly ReLU时，推荐使用He初始化方法。

2020/3/22

北京邮电大学计算机学院 鲁鹏

136

136



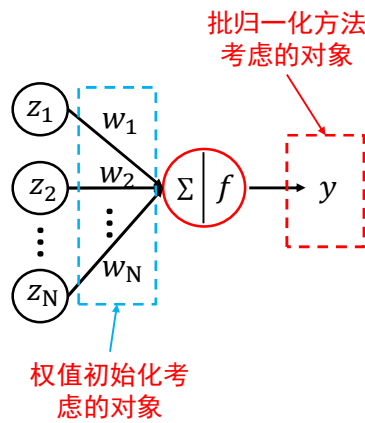
2020/3/22

北京邮电大学计算机学院 鲁鹏

137

137

## 批归一化



方法：直接对神经元的输出进行批归一化

方法：调整权值分布使得输出与输入具有相同的分布。

如果每一层的每个神经元进行批归一化，就能解决前向传递过程中的信号消失问题。

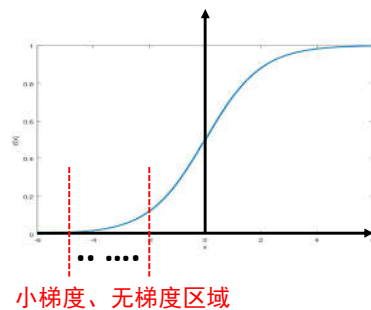
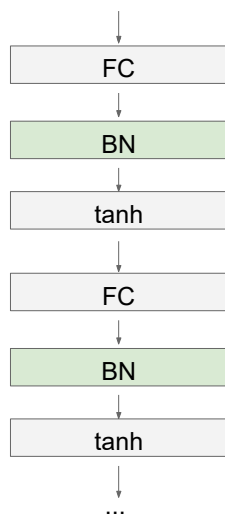
2020/3/22

北京邮电大学计算机学院 鲁鹏

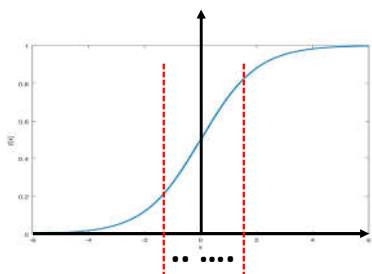
138

138

## 批归一化与梯度消失



减均值  
除方差



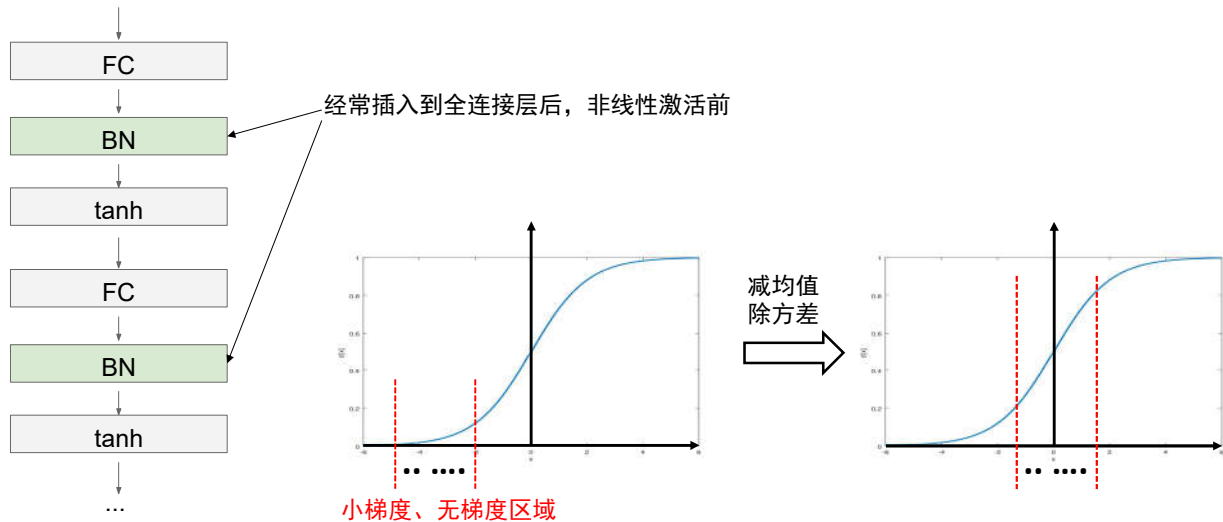
2020/3/22

北京邮电大学计算机学院 鲁鹏

139

139

## 批归一化与梯度消失



2020/3/22

北京邮电大学计算机学院 鲁鹏

140

140

## 批归一化

### 批归一化算法

输入:  $B = \{x_1, \dots, x_m\}$ ;

学习参数:  $\gamma, \beta$

输出:  $\{y_1, \dots, y_m\}$

1. 计算小批量均值:  $\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$
2. 计算小批量方差:  $\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$
3. 归一化:  $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
4. 平移缩放:  $y_i \leftarrow \gamma \hat{x}_i + \beta$

问题: 输出的0均值1方差的正态分布

是最有利于网络分类的分布吗?

问题: 单张样本测试时, 均值和方差

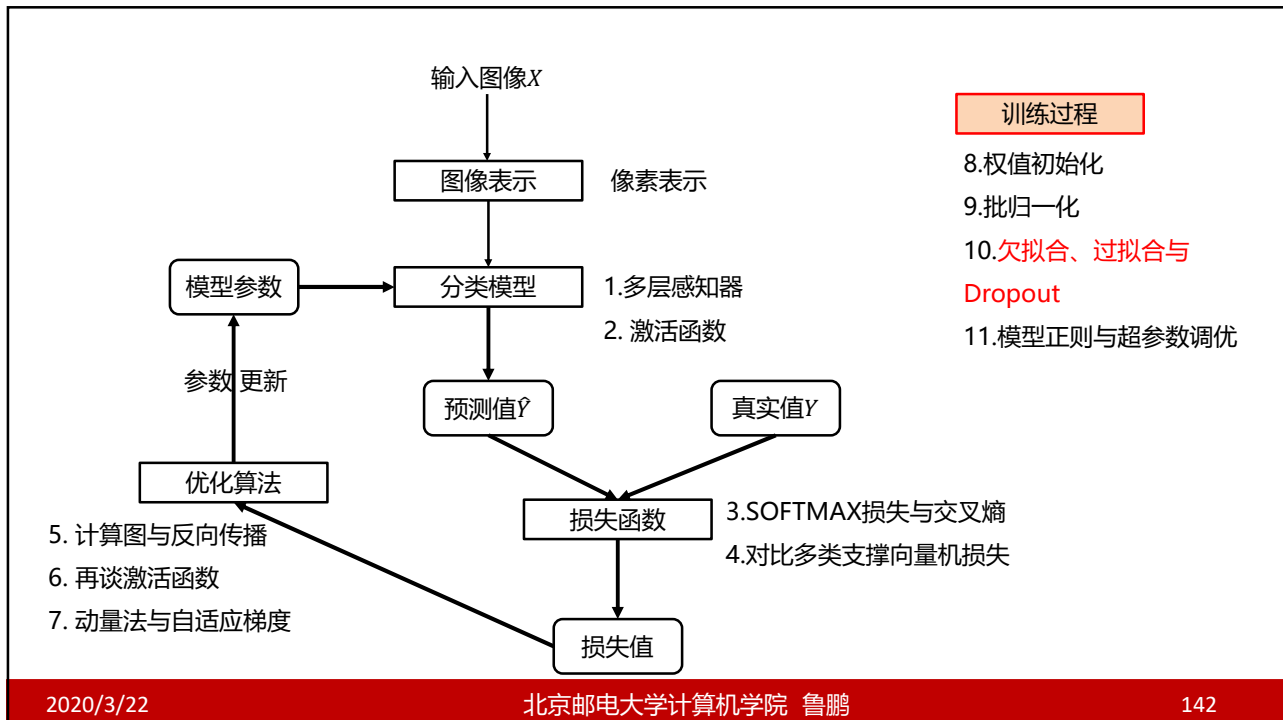
怎么设置?

2020/3/22

北京邮电大学计算机学院 鲁鹏

141

141



142

## 过拟合现象

出现过拟合，得到的模型在训练集上的准确率很高，但在真实的场景中识别率确很低。

2020/3/22

北京邮电大学计算机学院 鲁鹏

143

143



## 过拟合与欠拟合

**过拟合**——是指学习时选择的模型所包含的参数过多，以至于出现这一模型对已知数据预测的很好，但对未知数据预测得很差的现象。这种情况下模型可能只是记住了训练集数据，而不是学习到了数据特征。

**欠拟合**——模型描述能力太弱，以至于不能很好地学习到数据中的规律。

产生欠拟合的原因通常是模型过于简单。

2020/3/22

北京邮电大学计算机学院 鲁鹏

144

144

## 过拟合与欠拟合

**过拟合**——是指学习时选择的模型所包含的参数过多，以至于出现这一模型对已知数据预测的很好，但对未知数据预测得很差的现象。这种情况下模型可能只是记住了训练集数据，而不是学习到了数据特征。

**欠拟合**——模型描述能力太弱，以至于不能很好地学习到数据中的规律。

产生欠拟合的原因通常是模型过于简单。

2020/3/22

北京邮电大学计算机学院 鲁鹏

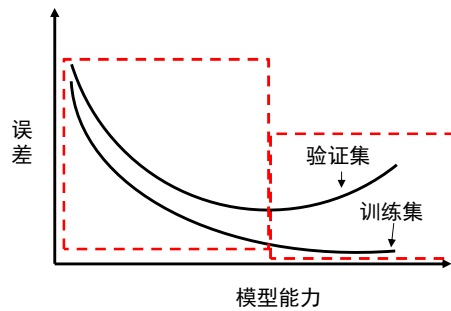
145

145

## 学习过程中的过拟合

- 机器学习的根本问题是优化和泛化的问题。
- 优化——是指调节模型以在训练数据上得到最佳性能；
- 泛化——是指训练好的模型在前所未见的的数据上的性能好坏。

**训练初期：**优化和泛化是相关的：训练集上的误差越小，验证集上的误差也越小，模型的泛化能力逐渐增强



**训练后期：**模型在验证集上的错误率不再降低，转而开始变高。模型出现过拟合，开始学习仅和训练数据有关的模式。

2020/3/22

北京邮电大学计算机学院 鲁鹏

146

146

## 应对过拟合

最优方案

次优方案

2020/3/22

北京邮电大学计算机学院 鲁鹏

147

147

## 应对过拟合

最优方案——获取更多的训练数据

次优方案

2020/3/22

北京邮电大学计算机学院 鲁鹏

148

148

## 应对过拟合

最优方案——获取更多的训练数据

次优方案——调节模型允许存储的信息量或者对模型允许存储的信息加以约束，该类方法也称为正则化。

2020/3/22

北京邮电大学计算机学院 鲁鹏

149

149

## 应对过拟合

最优方案——获取更多的训练数据

次优方案——调节模型允许存储的信息量或者对模型允许存储的信息加以约束，该类方法也称为正则化。

➤ 调节模型大小

2020/3/22

北京邮电大学计算机学院 鲁鹏

150

150

## 应对过拟合

最优方案——获取更多的训练数据

次优方案——调节模型允许存储的信息量或者对模型允许存储的信息加以约束，该类方法也称为正则化。

➤ 调节模型大小

➤ 约束模型权重，即权重正则化（常用的有L1、L2正则化）

2020/3/22

北京邮电大学计算机学院 鲁鹏

151

151

## L2正则化

$$L(W) = \underbrace{\frac{1}{N} \sum_i L_i(f(x_i, W), y_i)}_{\text{数据损失}} + \underbrace{\lambda R(W)}_{\text{权重正则损失}}$$

L2正则损失:  $R(W) = \sum_k \sum_l W_{k,l}^2$

2020/3/22

北京邮电大学计算机学院 鲁鹏

152

152

## L2正则化

$$L(W) = \underbrace{\frac{1}{N} \sum_i L_i(f(x_i, W), y_i)}_{\text{数据损失}} + \underbrace{\lambda R(W)}_{\text{权重正则损失}}$$

L2正则损失:  $R(W) = \sum_k \sum_l W_{k,l}^2$

L2正则损失对于大数值的权值向量进行严厉惩罚，鼓励更加分散的权重向量，

使模型倾向于使用所有输入特征做决策，此时的模型泛化性能好！

2020/3/22

北京邮电大学计算机学院 鲁鹏

153

153

## 应对过拟合

最优方案——获取更多的训练数据

次优方案——调节模型允许存储的信息量或者对模型允许存储的信息加以约束，该类方法也称为正则化。

- 调节模型大小
- 约束模型权重，即权重正则化（常用的有L1、L2正则化）
- 随机失活（Dropout）

2020/3/22

北京邮电大学计算机学院 鲁鹏

154

154

## 随机失活（Dropout）

- 随机失活：让隐层的神经元以一定的概率不被激活。

2020/3/22

北京邮电大学计算机学院 鲁鹏

155

155

## 随机失活（Dropout）

- 随机失活：让隐层的神经元以一定的概率不被激活。
- 实现方式：训练过程中，对某一层使用Dropout，就是随机将该层的一些输出舍弃（输出值设置为0），这些被舍弃的神经元就好像被网络删除了一样。

2020/3/22

北京邮电大学计算机学院 鲁鹏

156

156

## 随机失活（Dropout）

- 随机失活：让隐层的神经元以一定的概率不被激活。
- 实现方式：训练过程中，对某一层使用Dropout，就是随机将该层的一些输出舍弃（输出值设置为0），这些被舍弃的神经元就好像被网络删除了一样。
- 随机失活比率（Dropout ratio）：是被设为0的特征所占的比例，通常在0.2~0.5范围内。

2020/3/22

北京邮电大学计算机学院 鲁鹏

157

157

## 随机失活（Dropout）

- 随机失活：让隐层的神经元以一定的概率不被激活。
- 实现方式：训练过程中，对某一层使用Dropout，就是随机将该层的一些输出舍弃（输出值设置为0），这些被舍弃的神经元就好像被网络删除了一样。
- 随机失活比率（Dropout ratio）：是被设为 0 的特征所占的比例，通常在 0.2——0.5 范围内。

例：假设某一层对给定输入样本的返回值应该是向量：[0.2, 0.5, 1.3, 0.8, 1.1]。

使用Dropout后，这个向量会有几个随机的元素变成：[0, 0.5, 1.3, 0, 1.1]。

2020/3/22

北京邮电大学计算机学院 鲁鹏

158

158

## 随机失活（Dropout）

随机失活为什么能够防止过拟合呢？

2020/3/22

北京邮电大学计算机学院 鲁鹏

159

159



## 随机失活 (Dropout)

随机失活为什么能够防止过拟合呢？

解释1：随机失活使得每次更新梯度时参与计算的网路参数减少了，降低了模型容量，所以能防止过拟合。

2020/3/22

北京邮电大学计算机学院 鲁鹏

160

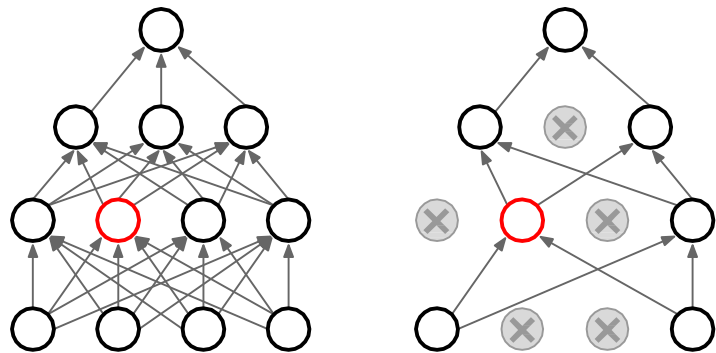
160

## 随机失活 (Dropout)

随机失活为什么能够防止过拟合呢？

解释1：随机失活使得每次更新梯度时参与计算的网路参数减少了，降低了模型容量，所以能防止过拟合。

解释2：



2020/3/22

北京邮电大学计算机学院 鲁鹏

161

161

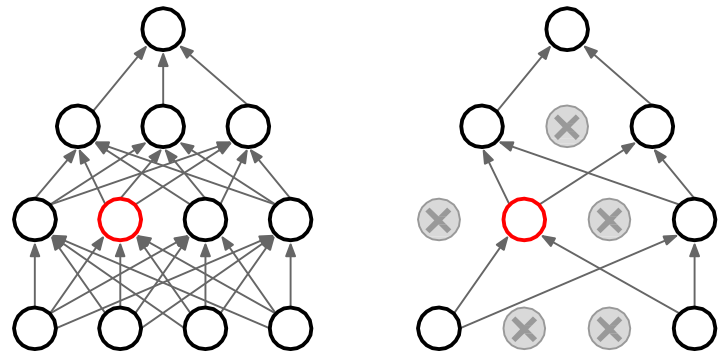
## 随机失活 (Dropout)

随机失活为什么能够防止过拟合呢？

解释1：随机失活使得每次更新梯度时参与计算的网路参数减少了，降低了模型容量，所以能防止过拟合。

解释2：

随机失活鼓励权重分散，从这个角度来看随机失活也能起到正则化的作用，进而防止过拟合。

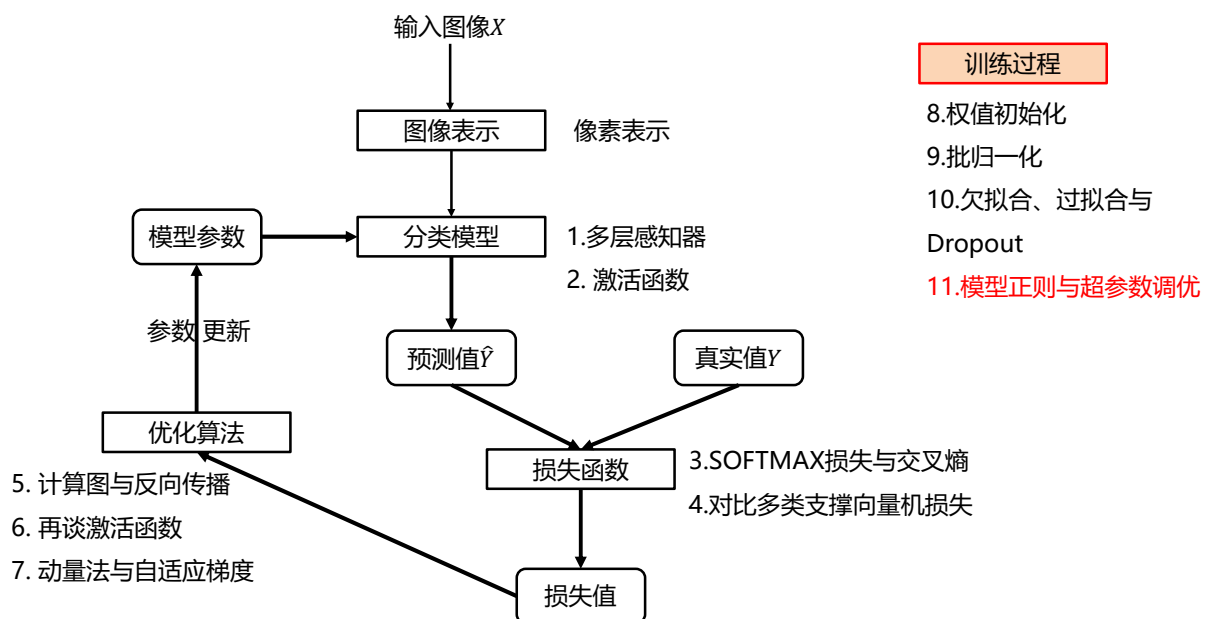


2020/3/22

北京邮电大学计算机学院 鲁鹏

162

162



2020/3/22

北京邮电大学计算机学院 鲁鹏

163

163

## 神经网络中的超参数

超参数：

- 网络结构——隐层神经元个数，网络层数，非线性单元选择等
- 优化相关——学习率、dropout比率、正则项强度等

超参数的重要性，如何找到合适的超参数？

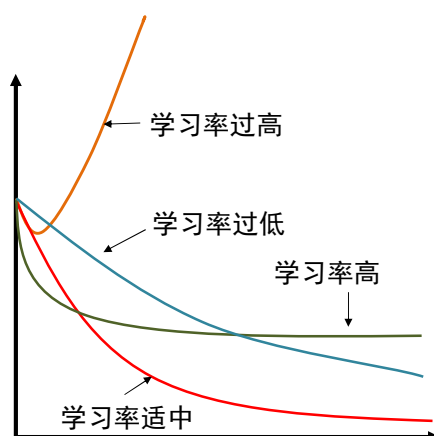
2020/3/22

北京邮电大学计算机学院 鲁鹏

164

164

## 学习率设置



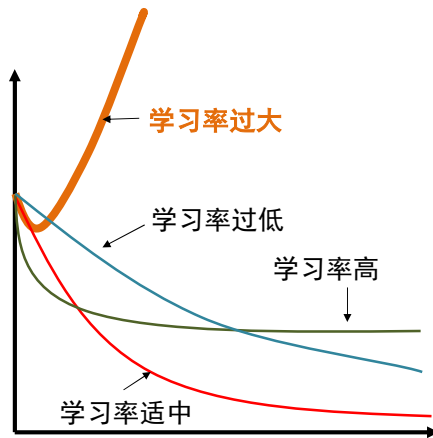
2020/3/22

北京邮电大学计算机学院 鲁鹏

165

165

## 学习率设置



➤ 学习率过大，训练过程无法收敛

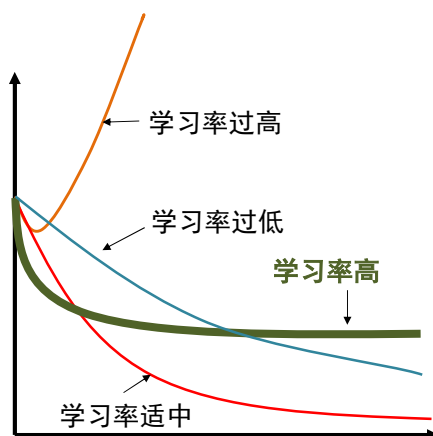
2020/3/22

北京邮电大学计算机学院 鲁鹏

166

166

## 学习率设置



➤ 学习率过大，训练过程无法收敛

➤ 学习率偏大，在最小值附近震荡，达不到最优。

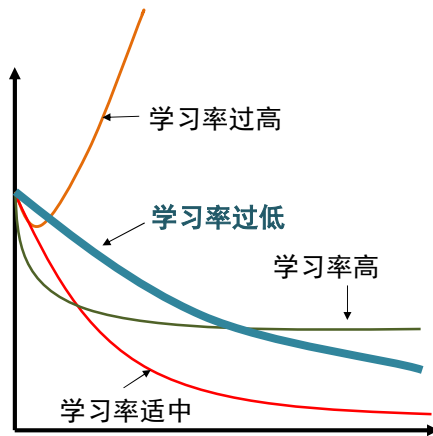
2020/3/22

北京邮电大学计算机学院 鲁鹏

167

167

## 学习率设置



- 学习率过大，训练过程无法收敛
- 学习率偏大，在最小值附近震荡，达不到最优。
- 学习率太小，收敛时间较长

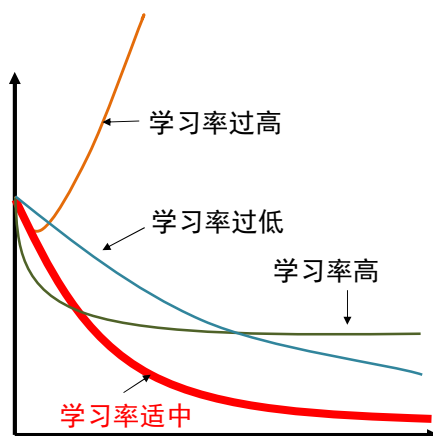
2020/3/22

北京邮电大学计算机学院 鲁鹏

168

168

## 学习率设置



- 学习率过大，训练过程无法收敛
- 学习率偏大，在最小值附近震荡，达不到最优
- 学习率太小，收敛时间较长
- 学习率适中，收敛快、结果好

2020/3/22

北京邮电大学计算机学院 鲁鹏

169

169

## 超参数优化方法

### 网格搜索法：

- ① 每个超参数分别取几个值，组合这些超参数值，形成多组超参数；
- ② 在验证集上评估每组超参数的模型性能；
- ③ 选择性能最优的模型所采用的那组值作为最终的超参数的值。

2020/3/22

北京邮电大学计算机学院 鲁鹏

170

170

## 超参数优化方法

### 网格搜索法：

- ① 每个超参数分别取几个值，组合这些超参数值，形成多组超参数；
- ② 在验证集上评估每组超参数的模型性能；
- ③ 选择性能最优的模型所采用的那组值作为最终的超参数的值。

### 随机搜索法：

- ① 参数空间内随机取点，每个点对应一组超参数；
- ② 在验证集上评估每组超参数的模型性能；
- ③ 选择性能最优的模型所采用的那组值作为最终的超参数的值。

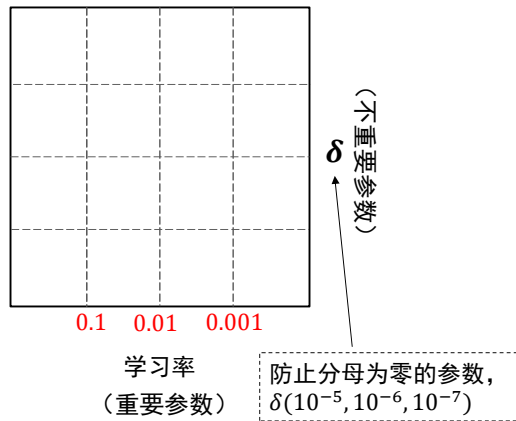
2020/3/22

北京邮电大学计算机学院 鲁鹏

171

171

## 超参数优化方法



2020/3/22

北京邮电大学计算机学院 鲁鹏

172

172

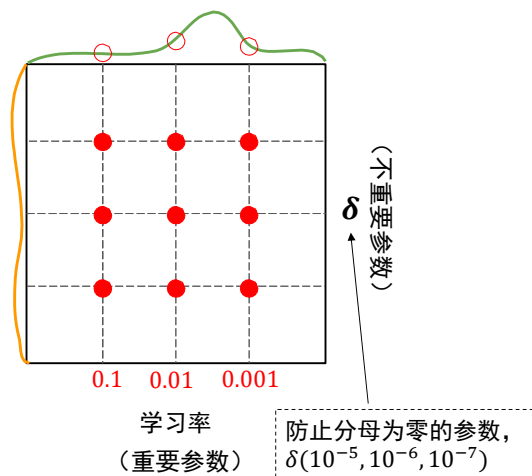
## 超参数优化方法

预算：9组实验

目标：最好的超参数组合

网格搜索

尝试了三个不同的学习率



2020/3/22

北京邮电大学计算机学院 鲁鹏

173

173

## 超参数优化方法

预算：9组实验

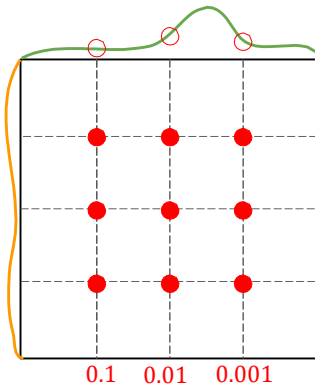
目标：最好的超参数组合

网格搜索

随机搜索



尝试了三个不同的学习率



学习率  
(重要参数)

防止分母为零的参数,  
 $\delta(10^{-5}, 10^{-6}, 10^{-7})$

学习率  
(重要参数)

(不重要参数)  
 $\delta$

尝试了九个不同的学习率!

2020/3/22

北京邮电大学计算机学院 鲁鹏

174

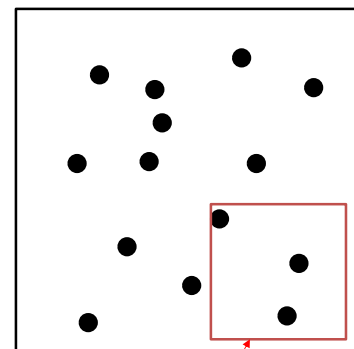
174

## 超参数搜索策略

超参数搜索策略：

- 粗搜索：利用随机法在较大范围里采样超参数，训练一个周期，依据验证集正确率缩小超参数范围。

参数空间



缩小后的  
参数空间

2020/3/22

北京邮电大学计算机学院 鲁鹏

175

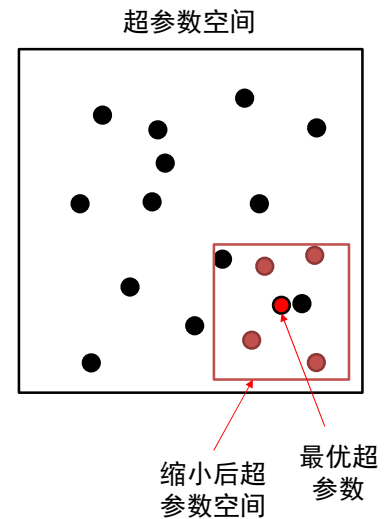
175



## 超参数搜索策略

超参数搜索策略：

- 粗搜索：利用随机法在较大范围里采样超参数，训练一个周期，依据验证集正确率缩小超参数范围。
- 精搜索：利用随机法在前述缩小的范围内采样超参数，运行模型五到十个周期，选择验证集上精度最高的那组超参数。



2020/3/22

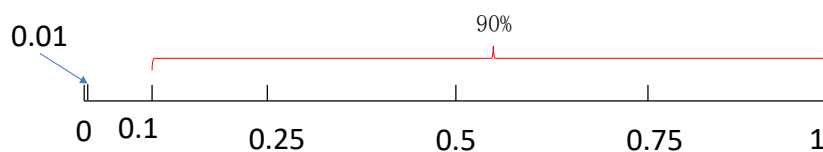
北京邮电大学计算机学院 鲁鹏

176

176

## 超参数的标尺空间

以学习率为例：



2020/3/22

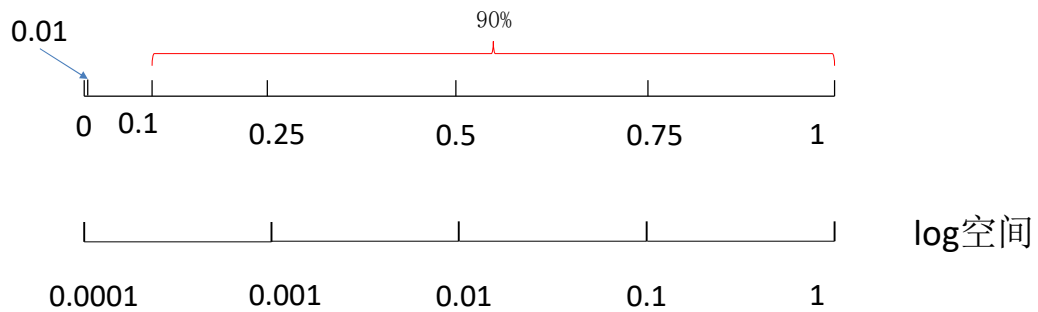
北京邮电大学计算机学院 鲁鹏

177

177

## 超参数的标尺空间

以学习率为例：



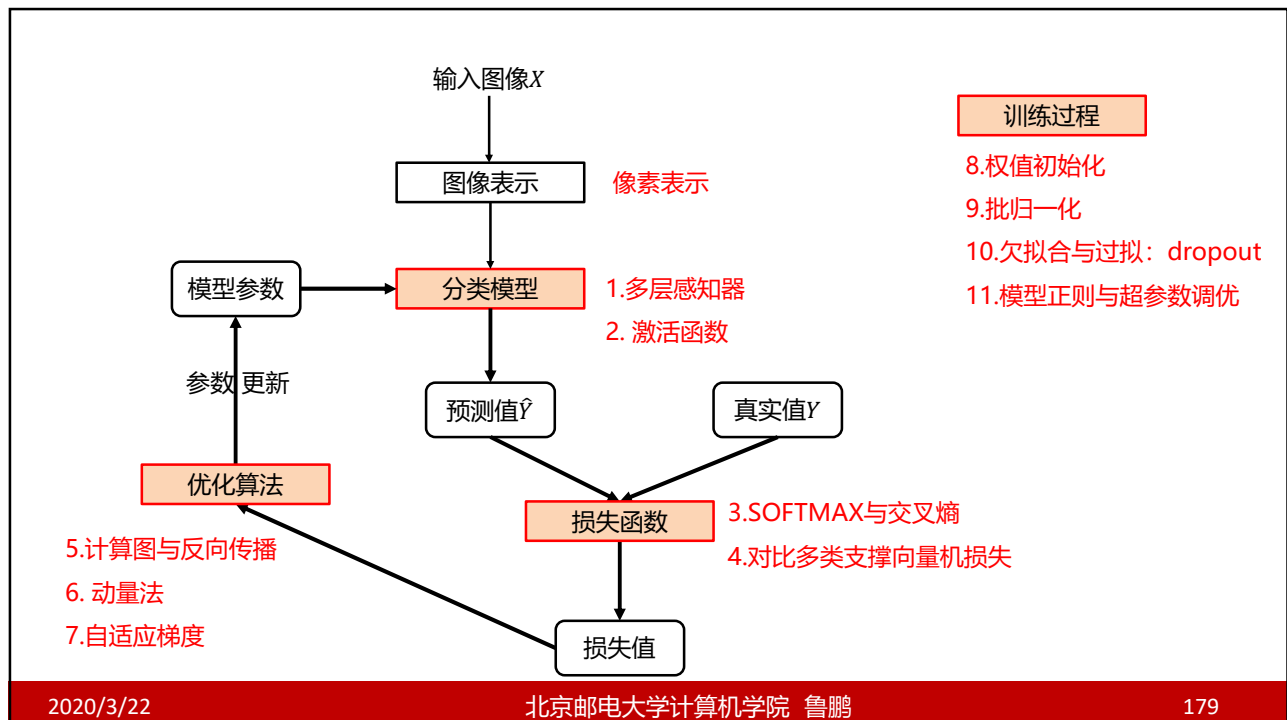
建议：对于学习率、正则项强度这类超参数，在对数空间上进行随机采样更合适！

2020/3/22

北京邮电大学计算机学院 鲁鹏

178

178



2020/3/22

北京邮电大学计算机学院 鲁鹏

179

179