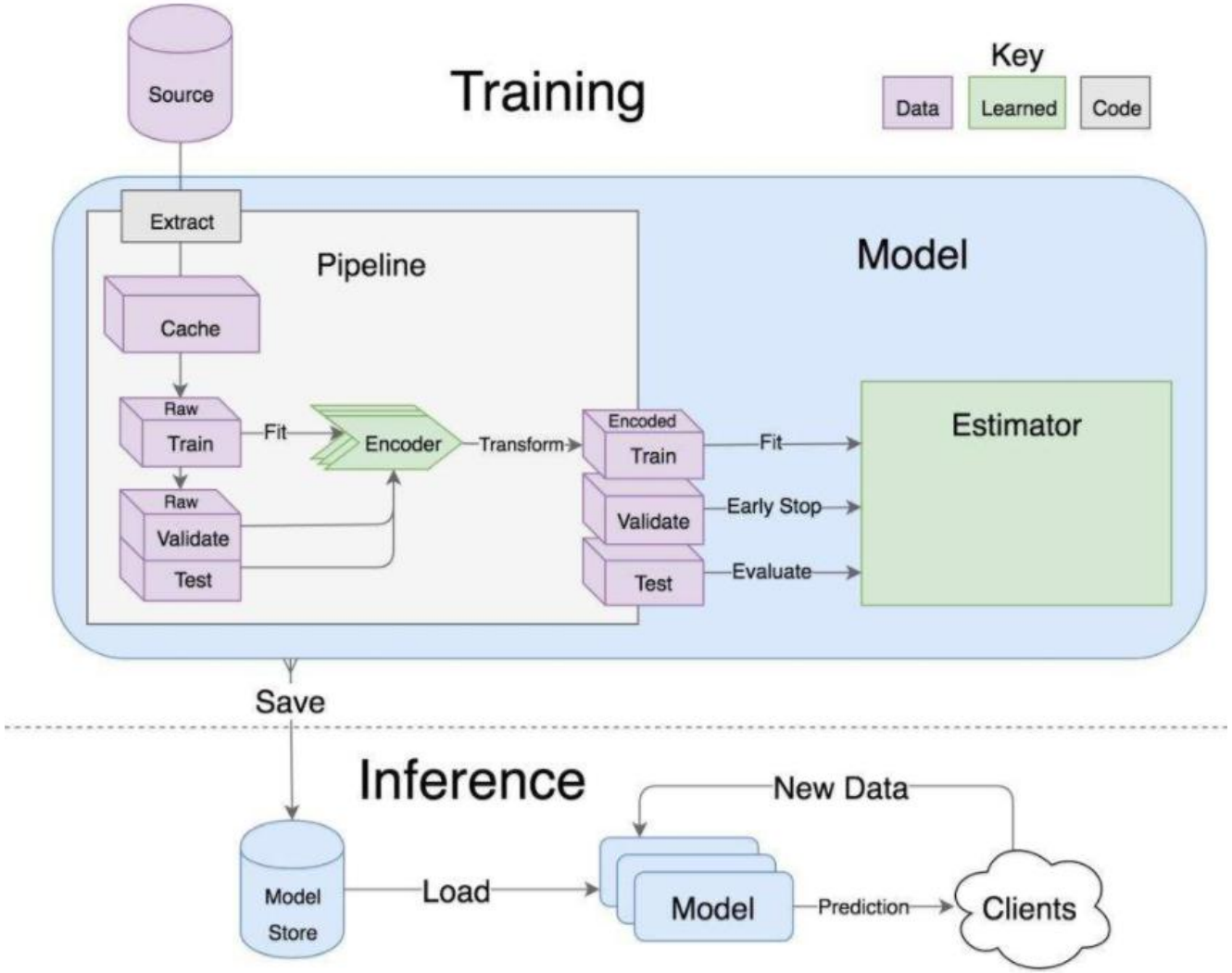
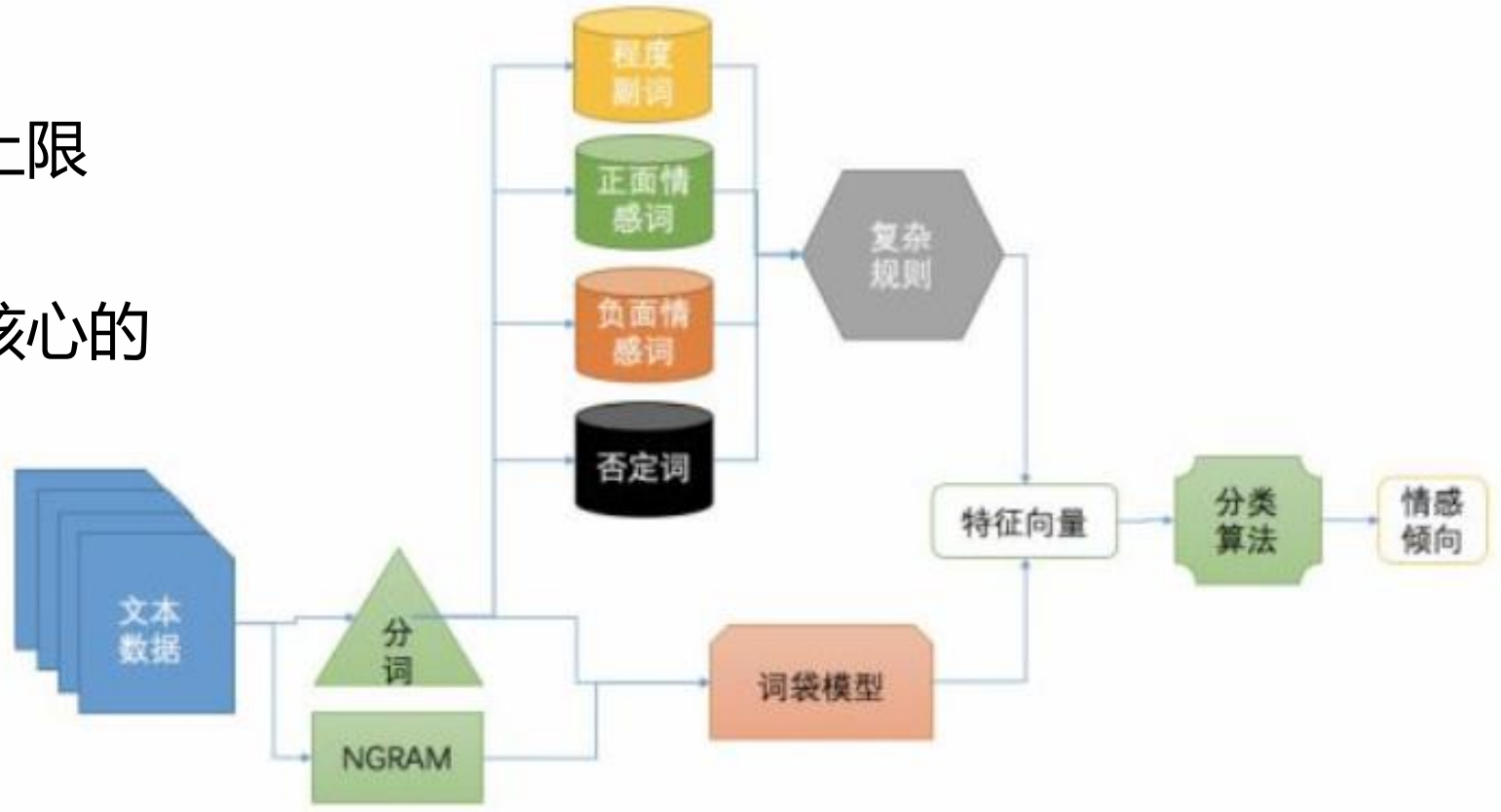
深度学习

- 机器学习流程：
  - 数据获取
  - 特征工程
  - 建立模型
  - 评估与应用



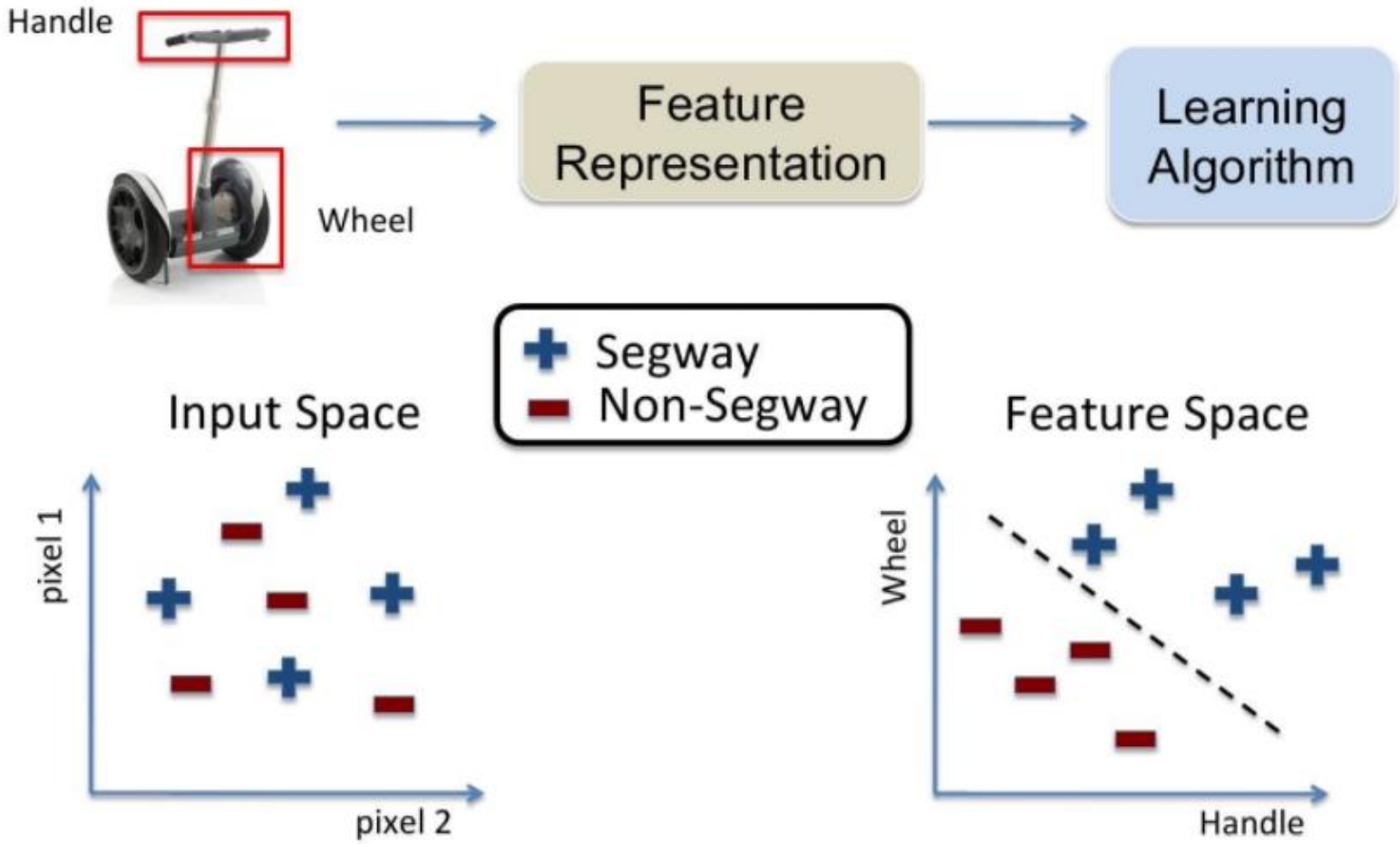
深度学习

- 特征工程的作用：
  - 数据特征决定了模型的上限
  - 预处理和特征提取是最核心的
  - 算法与参数选择决定了如何逼近这个上限



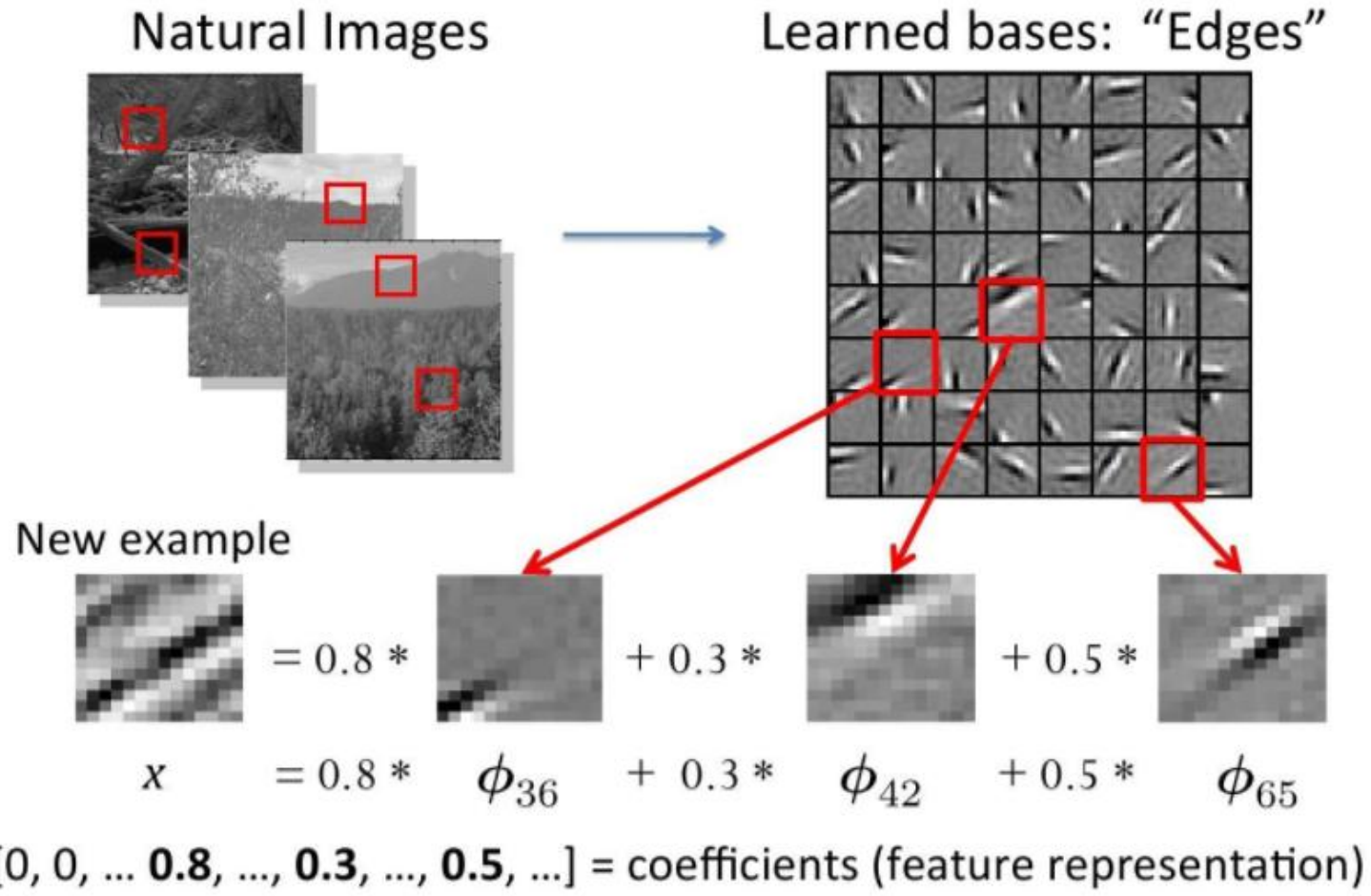
深度学习

- 特征如何提取：


Handle
Wheel
Feature Representation
Learning Algorithm
Segway
Non-Segway
Input Space
pixel 1
pixel 2
Feature Space
Wheel
Handle

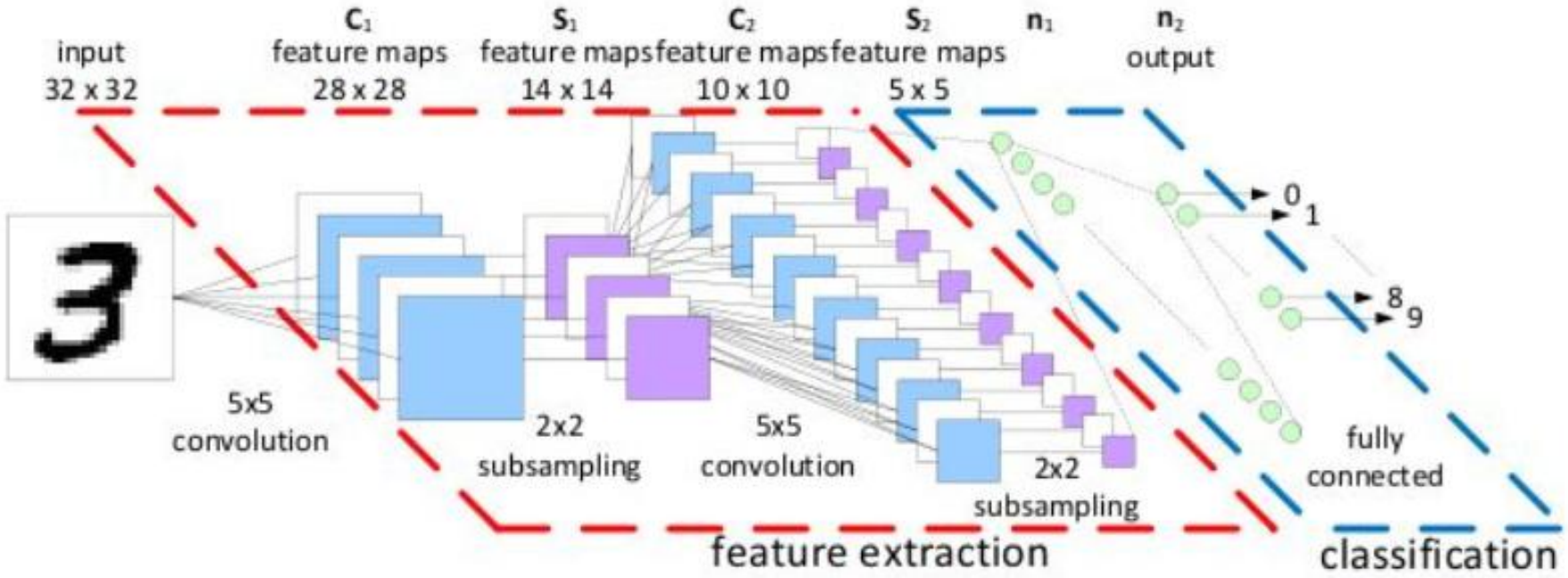
深度学习

- 传统特征提取方法：



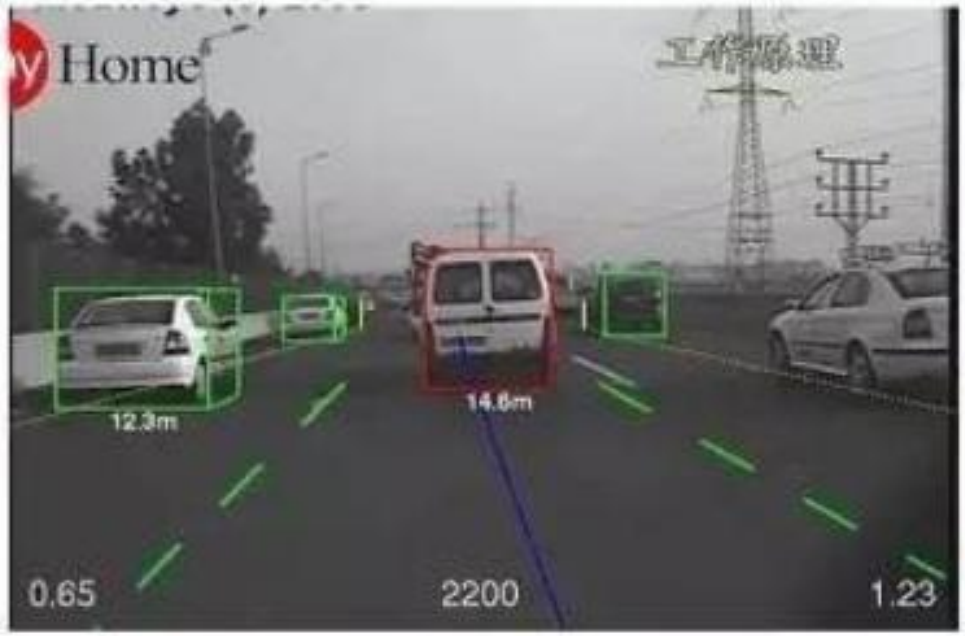
深度学习

- 为什么需要深度学习：



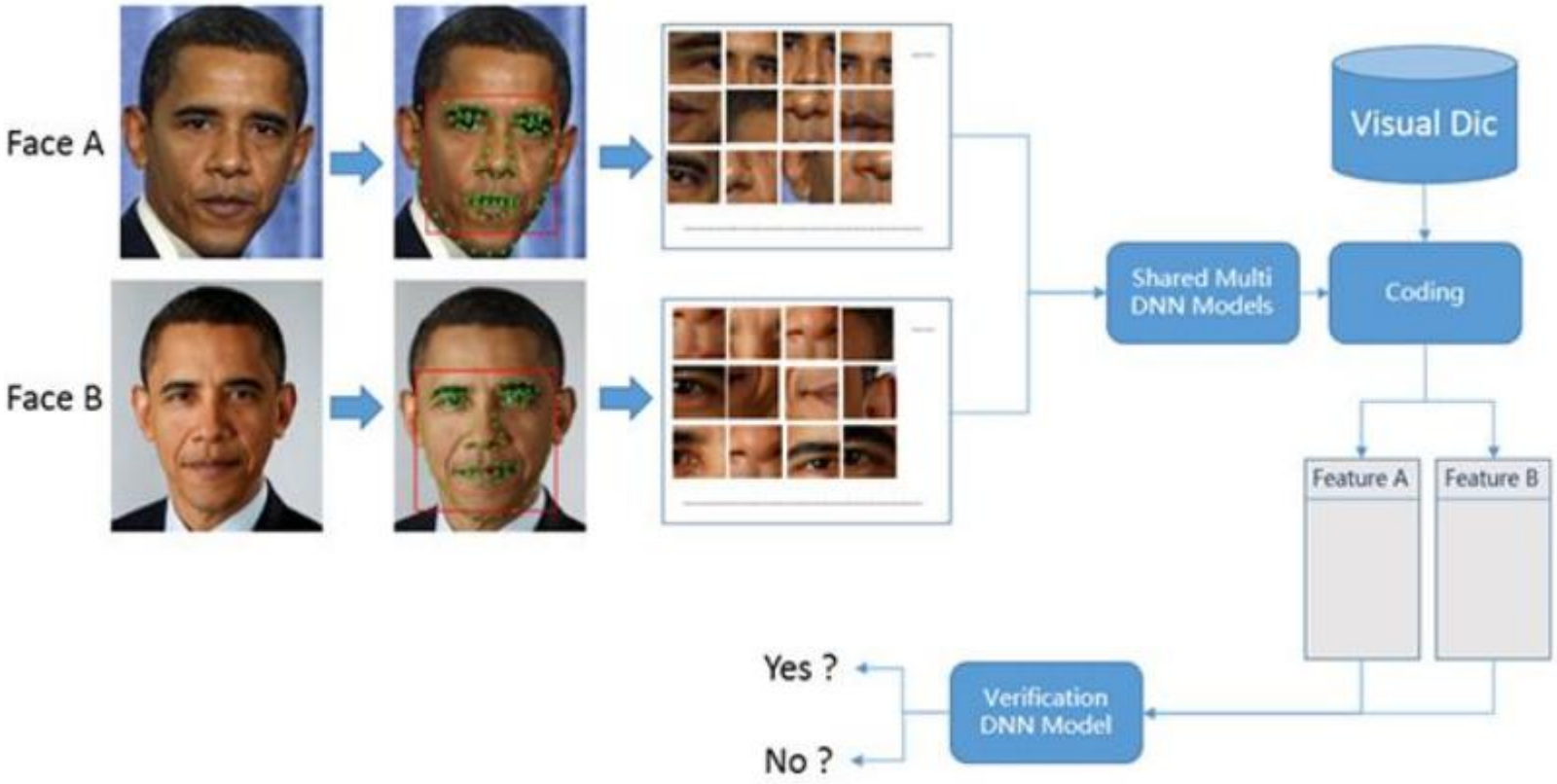
深度学习

- 深度学习应用：



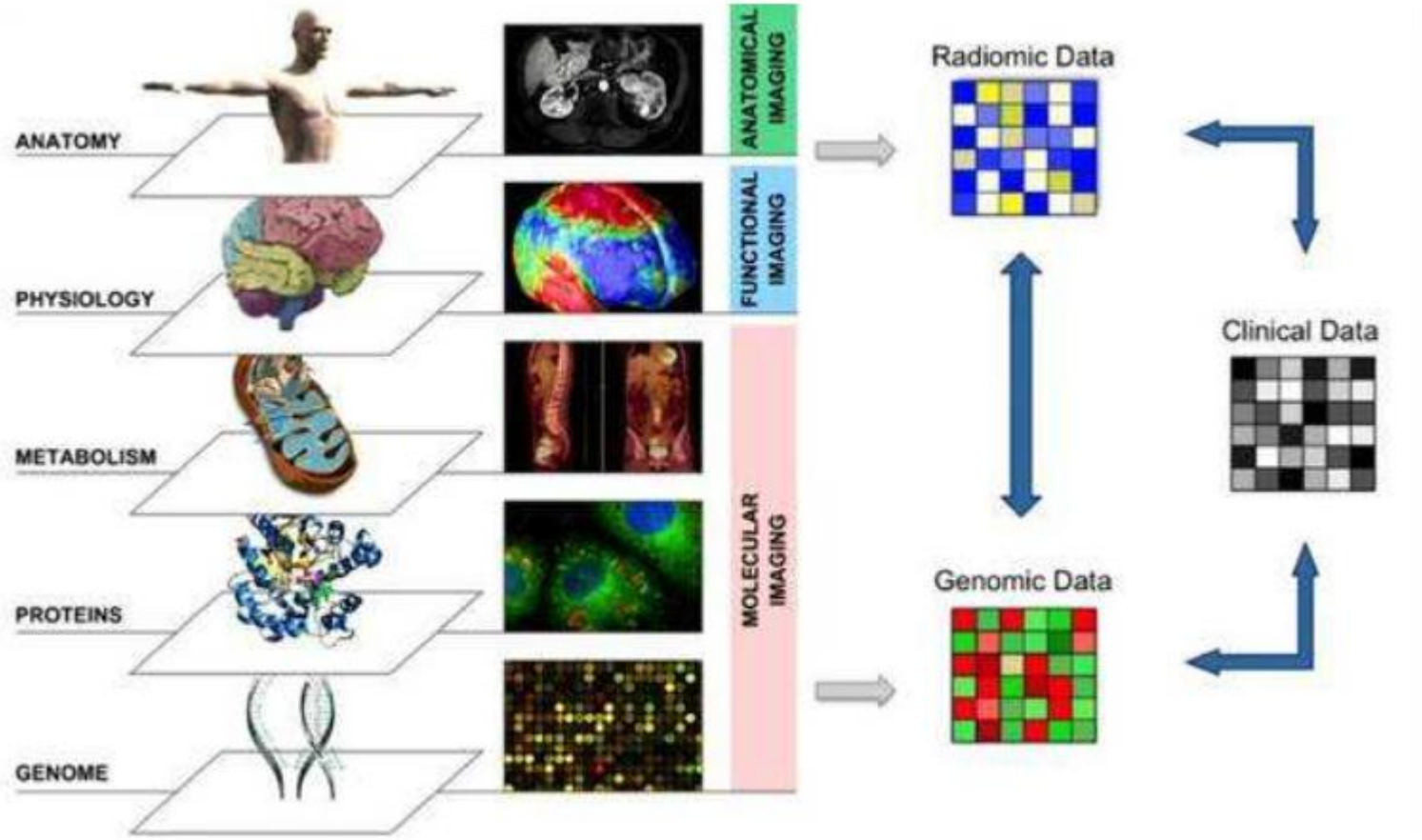
深度学习

- 深度学习应用：


Face A
Face B
Visual Dic
Shared Multi DNN Models
Coding
Feature A
Feature B
Yes ?
No ?
Verification DNN Model

深度学习

- 深度学习应用：


ANATOMY
PHYSIOLOGY
METABOLISM
PROTEINS
GENOME
ANATOMICAL IMAGING
FUNCTIONAL IMAGING
MOLECULAR IMAGING
Radiomic Data
Clinical Data
Genomic Data

深度学习

- 深度学习应用：

深度学习

- 深度学习应用：

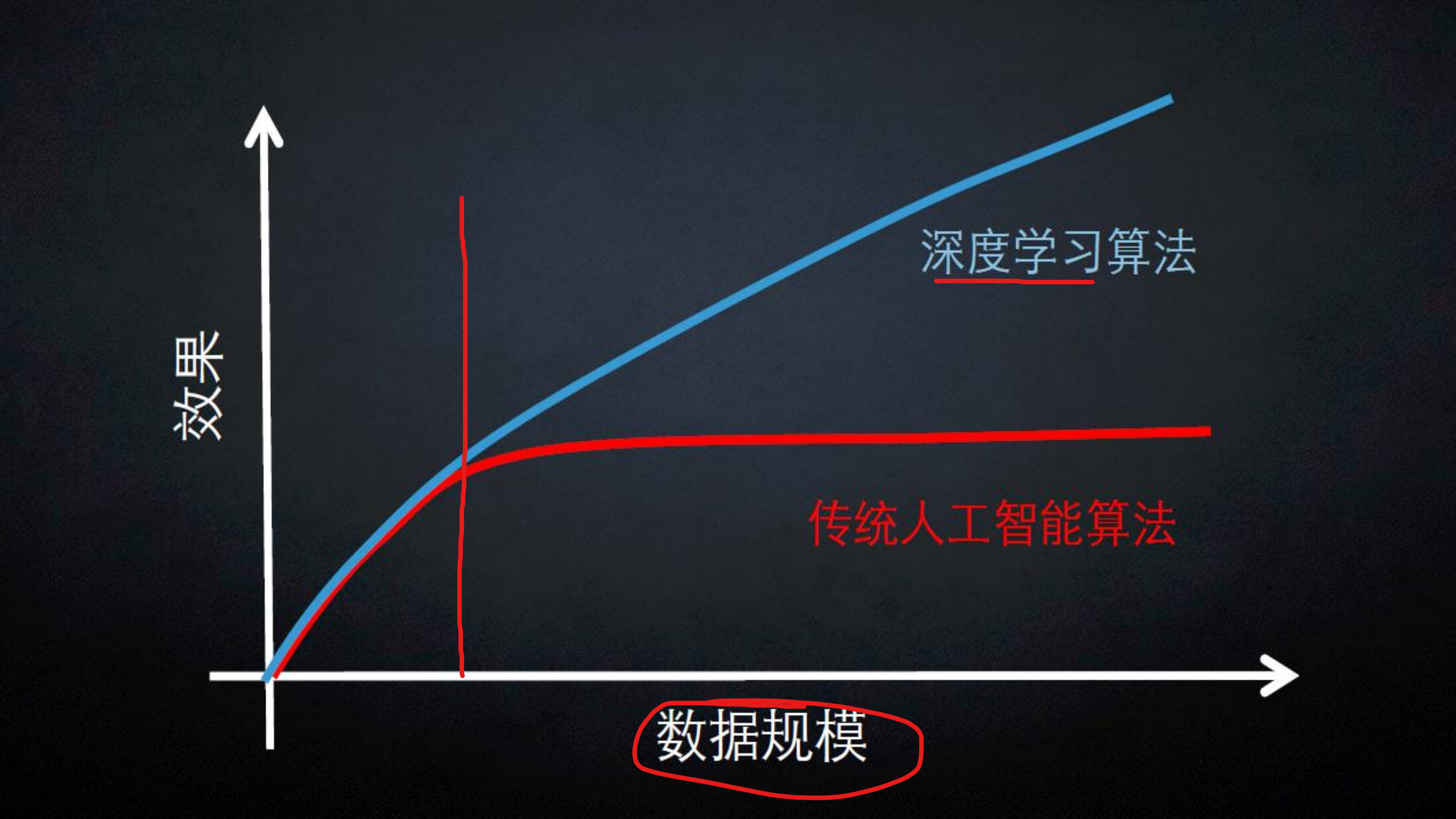
IMAGENET

www.image-net.org

ALEX

22K categories and **14M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formation
- Sport Activities


效果
深度学习算法
传统人工智能算法
数据规模

图像分类

- 计算机视觉：

  - 图像分类任务

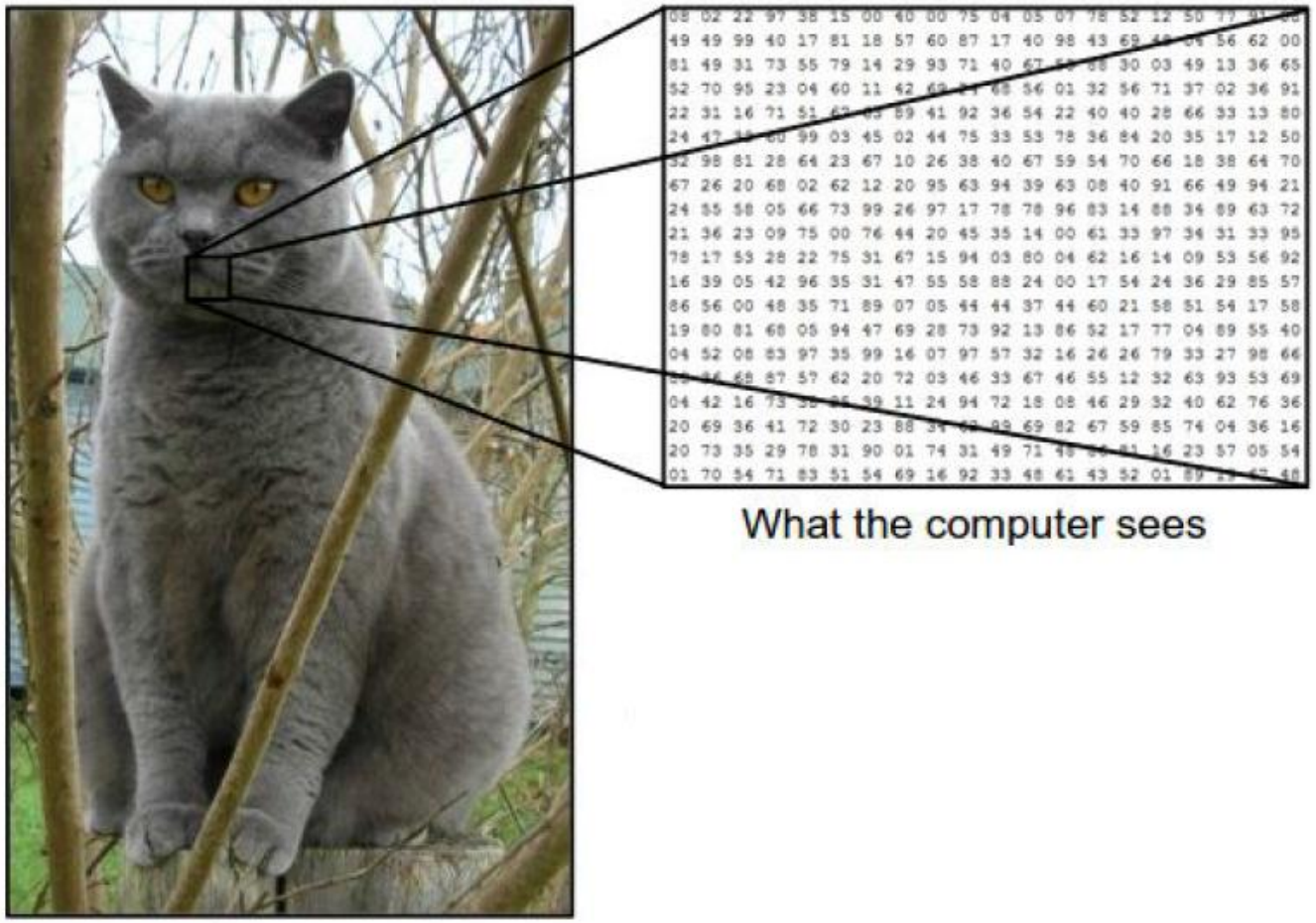
（假设我们有一系列的标签：狗，猫，汽车，飞机。。。）

→ 猫

图像分类

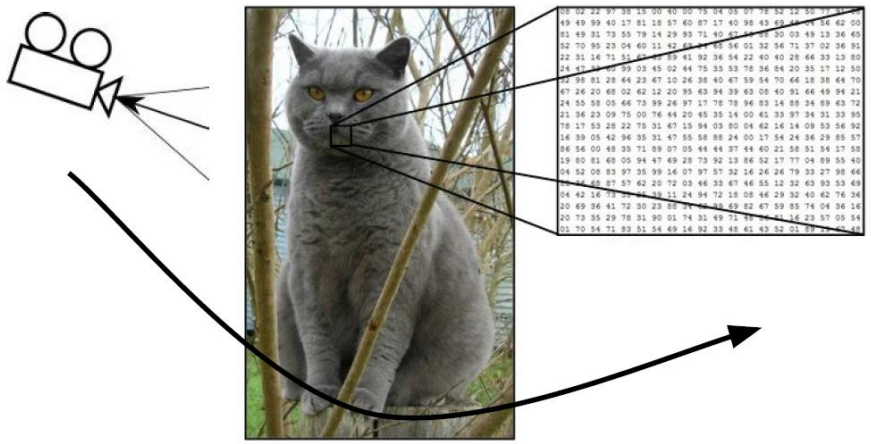
- 计算机视觉：
  - 图像表示：计算机眼中的图像
  - 一张图片被表示成三维数组的形式，每个像素的值从0到255

    例如：300*100*3



图像分类

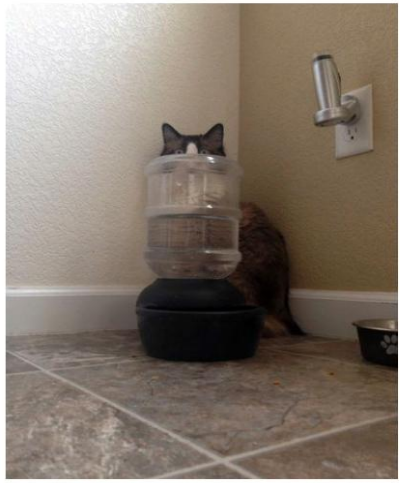
- 计算机视觉面临的挑战：
  - 照射角度：



  - 形状改变：

图像分类

- 计算机视觉面临的挑战：
  - 部分遮蔽：


  - 背景混入：

图像分类

- 机器学习常规套路：

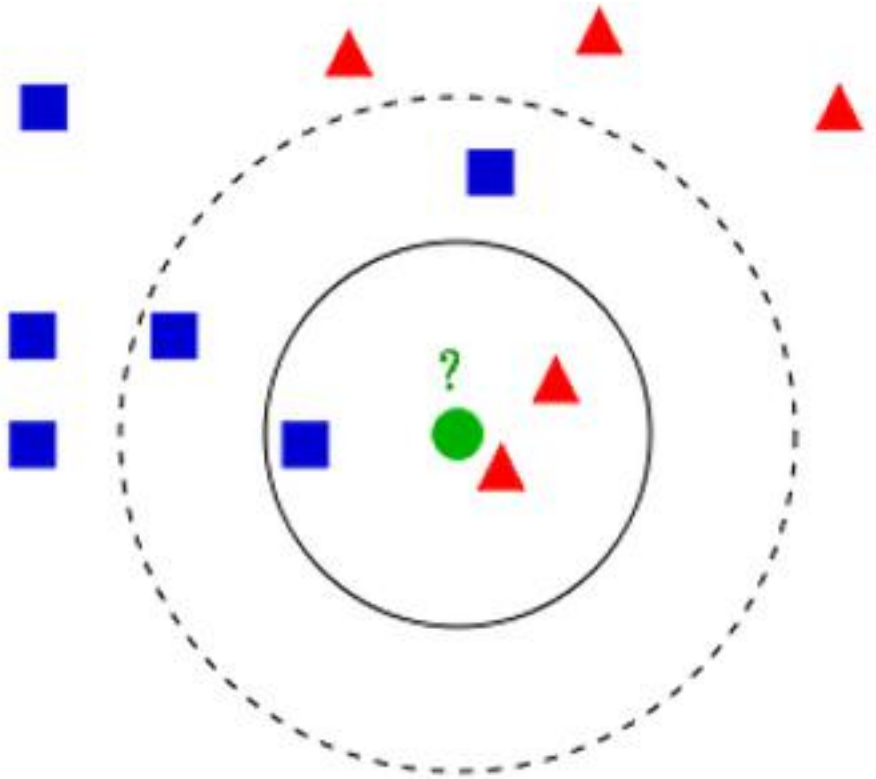
1.收集数据并给定标签
2.训练一个分类器
3.测试，评估

```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

K近邻

- K近邻算法：
  - 数据：两类点方块和三角
  - 绿色的点属于方块还是三角呢?
  - K=3还是K=5？结果一样吗?



K近邻

- K近邻计算流程：

1.计算已知类别数据集中的点与当前点的距离

2.按照距离依次排序

3.选取与当前点距离最小的K个点

4.确定前K个点所在类别的出现概率

5.返回前K个点出现频率最高的类别作为当前点预测分类

K近邻

K近邻分析

- KNN 算法本身简单有效，它是一种 lazy-learning 算法。
- 分类器不需要使用训练集进行训练，训练时间复杂度为0。
- KNN 分类的计算复杂度和训练集中的文档数目成正比，
  也就是说，如果训练集中文档总数为 n，那么 KNN 的分类时间复杂度为O(n)。
- K 值的选择，距离度量和分类决策规则是该算法的三个基本要素

数据库样例：CIFAR-10

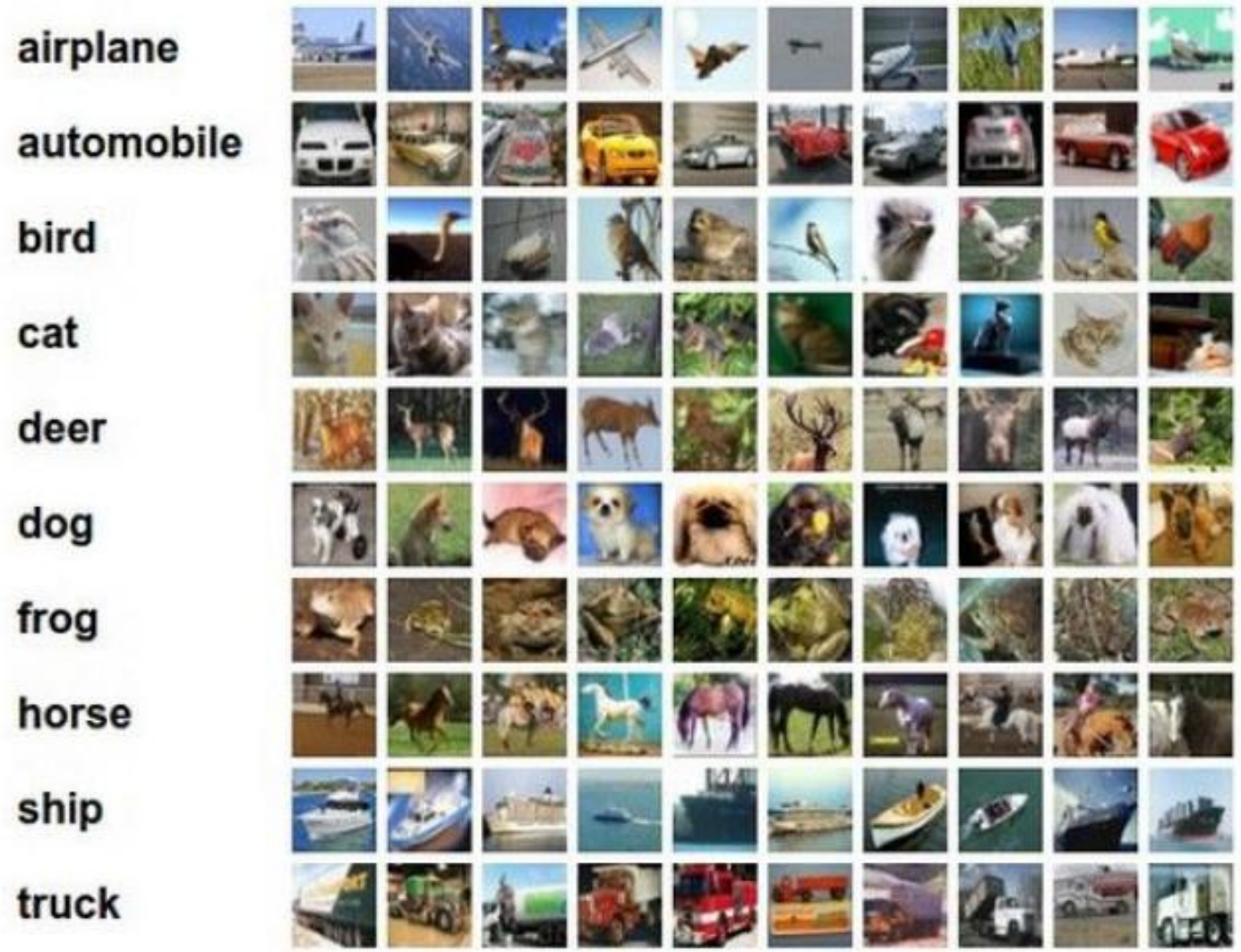
- 数据库简介：

  - 10类标签

    50000个训练数据

    10000个测试数据

    大小均为32*32



图像分类

- 距离的选择： **L1 distance:** $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

- 图像距离计算方式：

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

add → 456

图像分类：

- 测试结果：
  - 部分结果还可以的
  - 没有分类对的图像，

    问题出在哪里呢？

图像分类：

- 参数会对结果有影吗？

- 距离的定义：　L1 (Manhattan) distance　　L2 (Euclidean) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

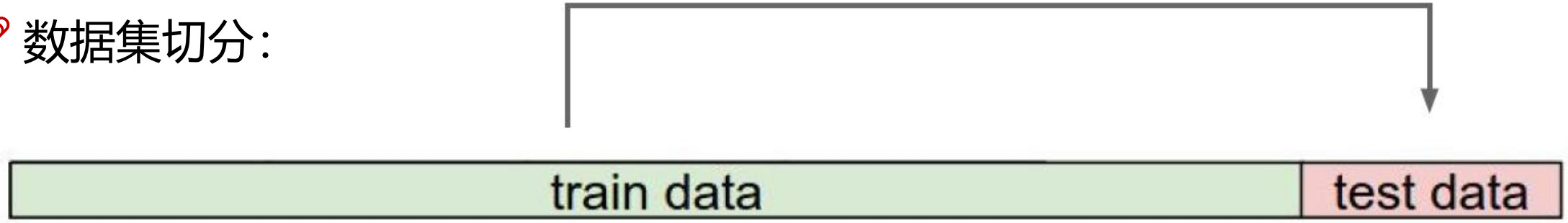
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

- 对于K近邻的K该如何选择？

- 如果有的话，其它的超参数该怎么设定呢？

图像分类：

- 选择最好的参数
  - 交叉验证：实践来检验真理
  - 数据集切分：

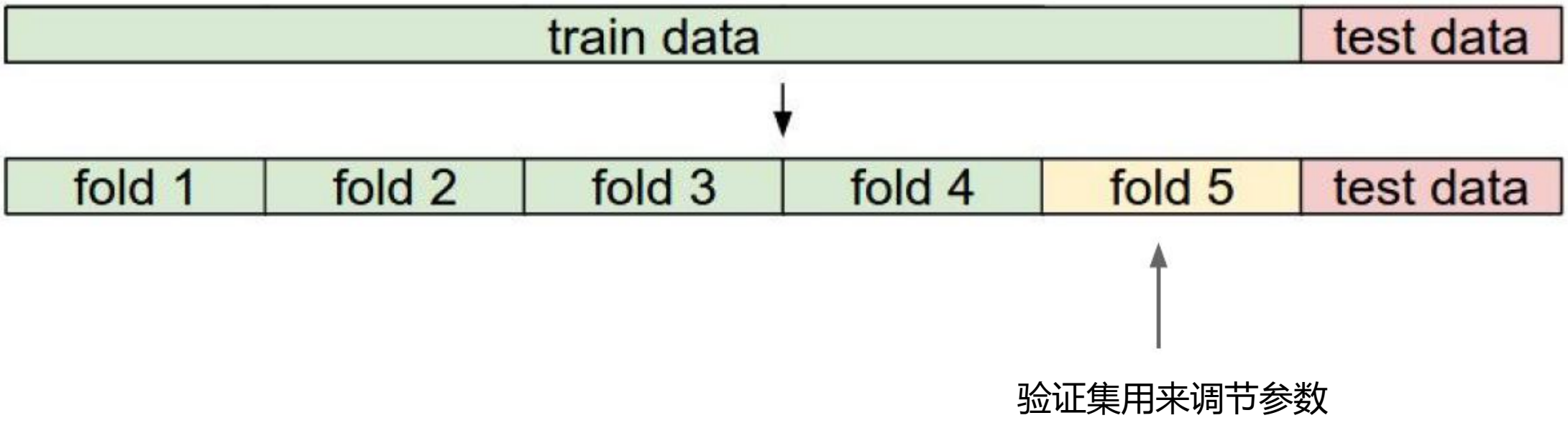


  - 用测试集来选择最好的参数吗？

图像分类：

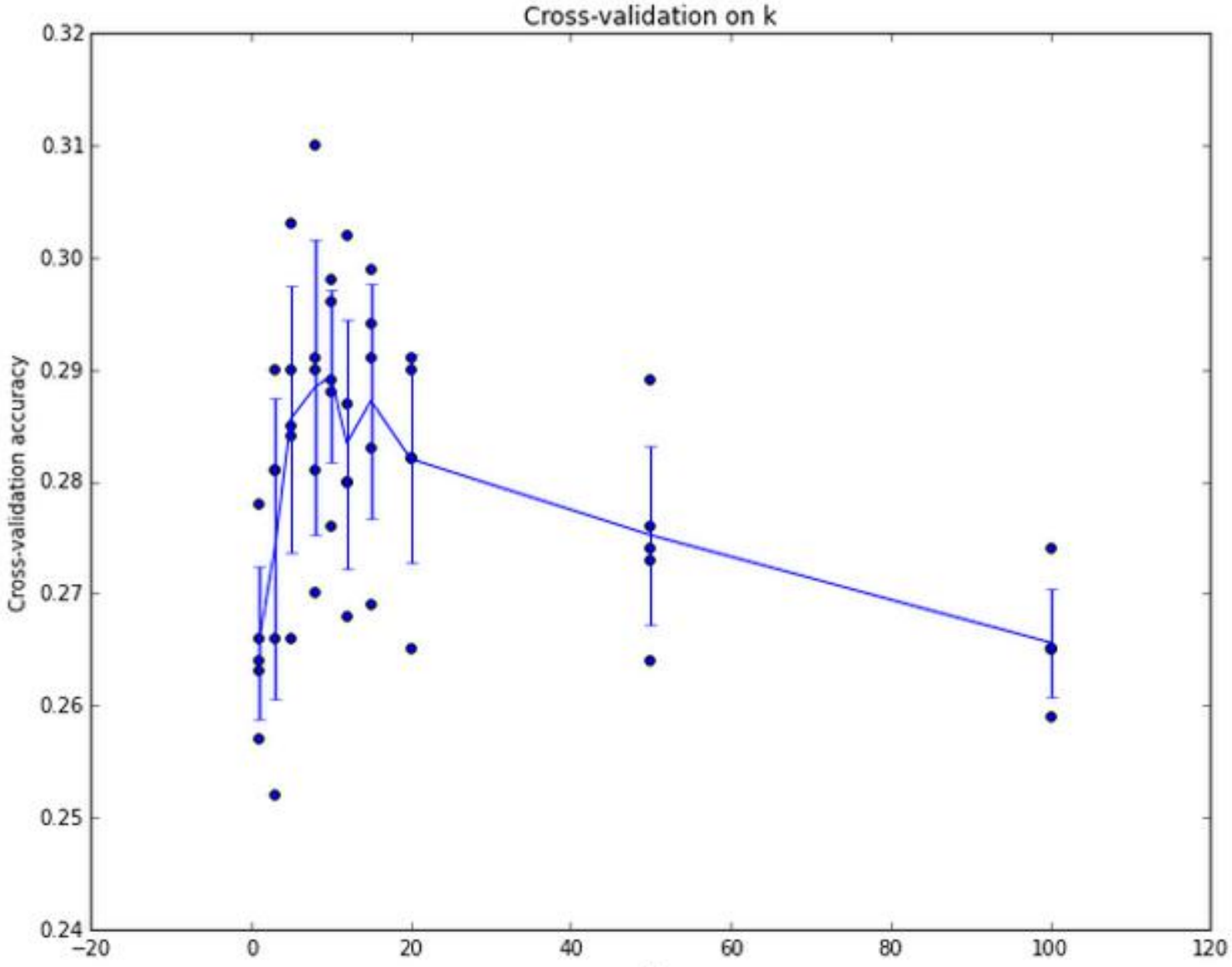
- 选择最好的参数

  - 交叉验证：得到更可靠的验证结果



图像分类：

- K近邻交叉验证结果
  - 参数对结果的影响有多大？
  - 交叉验证是最合适的参数选择方法，比经验值靠谱的多！



图像分类：

- 为什么K近邻不能用来图像分类？
  - 背景主导是一个最大的问题，我们关注的却是主体（主要成分）
  - 如何才能让机器学习到哪些是重要的成分呢？

神经网络基础

- 线性函数

  - 从输入-->输出的映射

(32x32x3)

image parameters

$$f(x,W)$$

每个类别的得分

神经网络基础

- 线性函数

- 数学表示

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

3072x1

(+b) 10x1

10x1

10x3072

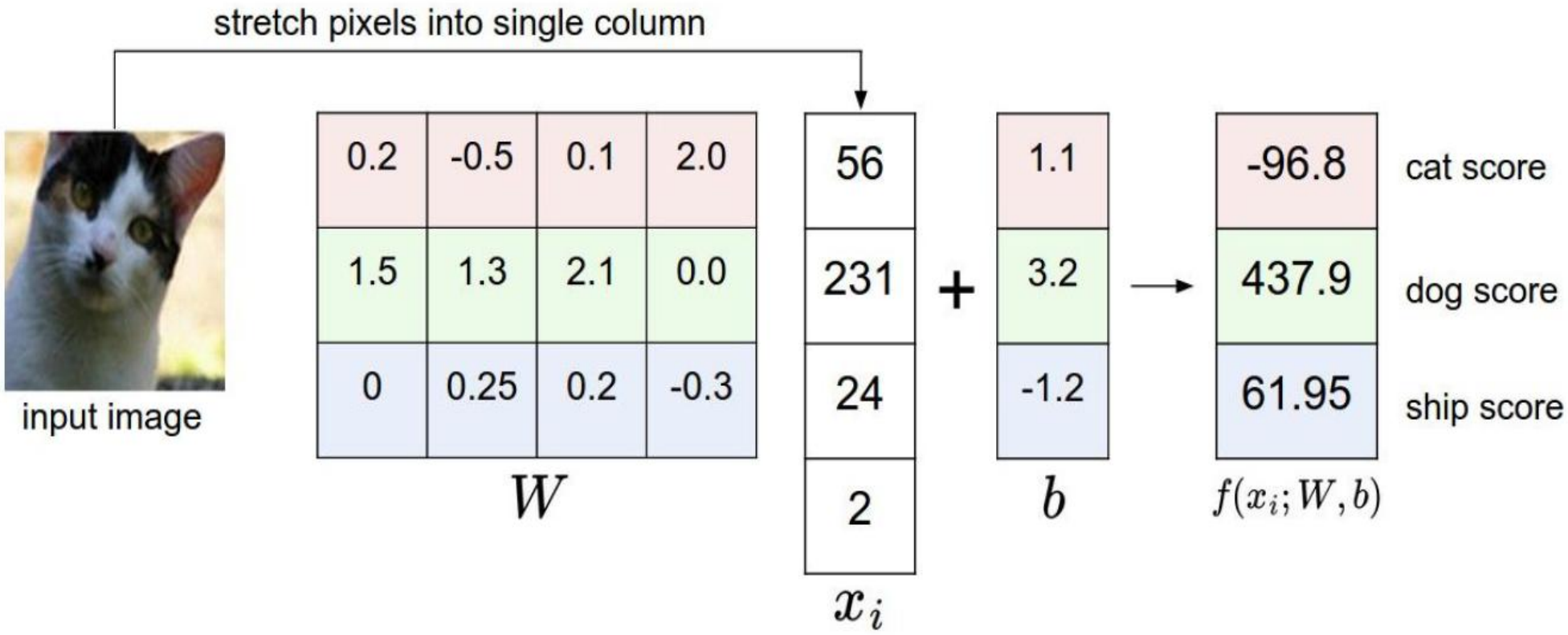
[32x32x3]

每个类别的得分

神经网络基础

- 线性函数

- 计算方法

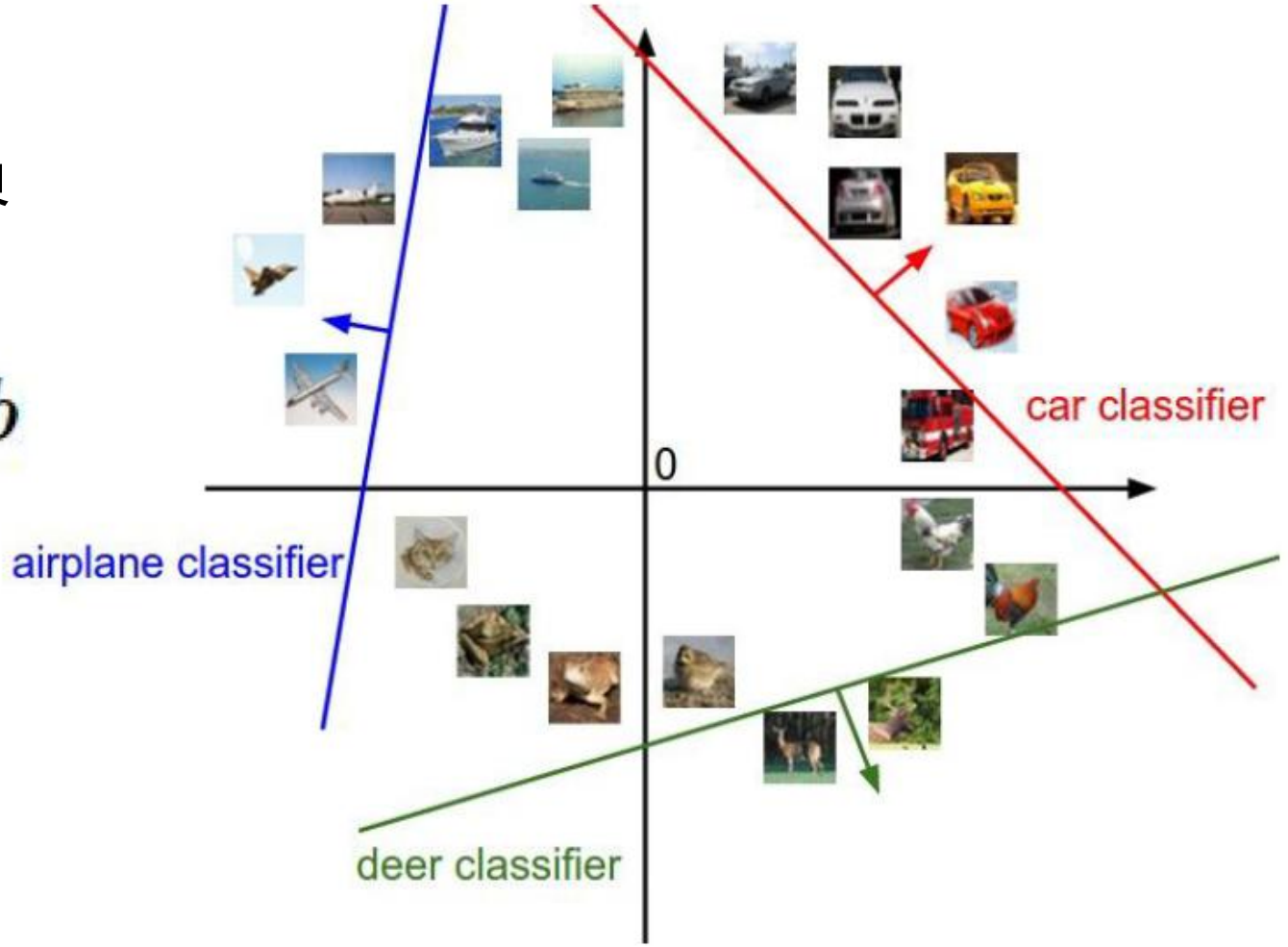


神经网络基础

- 线性函数

  - 多组权重参数构成了决策边界

$$f(x_i, W, b) = Wx_i + b$$



神经网络基础

- 损失函数

  - 如何衡量分类的结果呢?

  - 结果的得分值有着明显的差异，我们需要明确的指导模型的当前效果，有多好或是多差!



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

神经网络基础

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

- 损失函数其实有很多种，我们来实验一个

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

= max(0, 5.1 - 3.2 + 1)
  +max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

= max(0, 1.3 - 4.9 + 1)
  +max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

= max(0, 2.2 - (-3.1) + 1)
  +max(0, 2.5 - (-3.1) + 1)
= max(0, 5.3) + max(0, 5.6)
= 5.3 + 5.6
= 10.9

神经网络基础

损失函数

如何损失函数的值相同，那么意味着两个模型一样吗？

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

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j \neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

输入数据：$x = [1, 1, 1, 1]$

模型A：$w_1 = [1, 0, 0, 0]$

模型B：$w_2 = [0.25, 0.25, 0.25, 0.25]$

$$w_1^T x = w_2^T x = 1$$

神经网络基础

损失函数

损失函数 = 数据损失 + 正则化惩罚项

正则化惩罚项

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i;W)_j - f(x_i;W)_{y_i} + 1) + \lambda R(W)$$

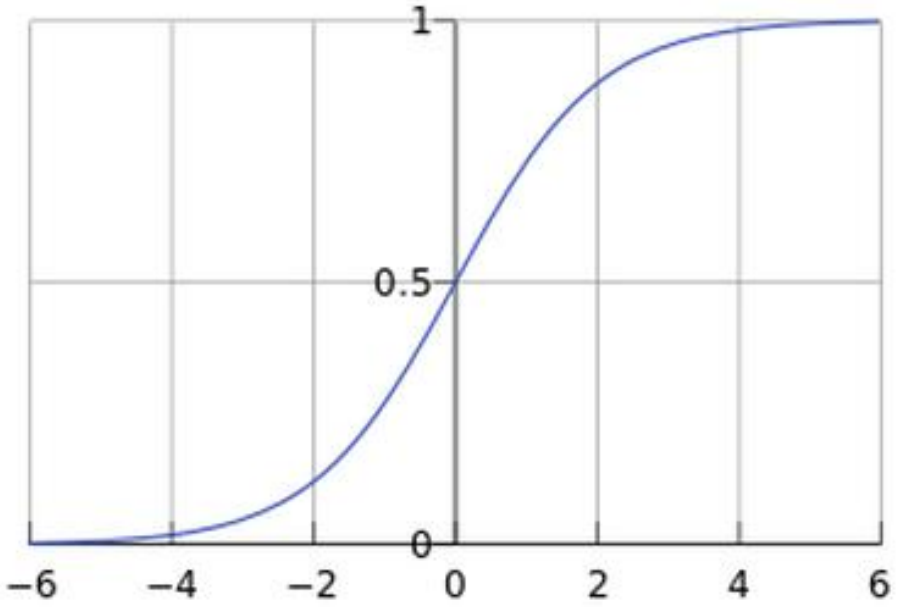
正则化惩罚项：$R(W) = \sum_k \sum_l W_{k,l}^2$

（我们总是希望模型不要太复杂，过拟合的模型是没用的）

神经网络基础

Softmax分类器

- 现在我们得到的是一个输入的得分值，但如果给我一个概率值岂不更好！
- 如何把一个得分值转换成一个概率值呢？



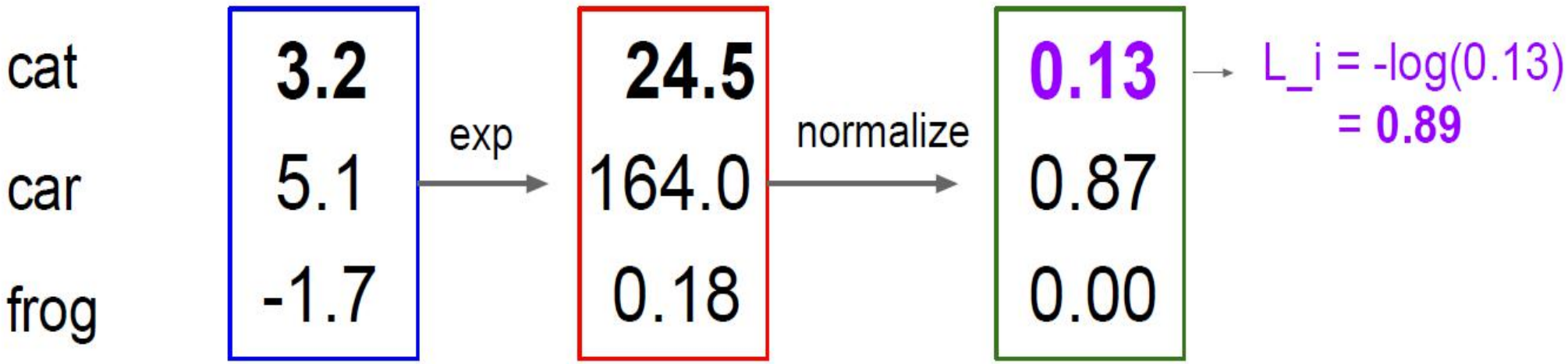
$$g(z)=\frac{1}{1+e^{-z}}$$

神经网络基础

- Softmax分类器

归一化：$$P(Y=k|X=x_i)=\frac{e^{s_k}}{\sum_j e^{s_j}}$$ where $$s=f(x_i;W)$$

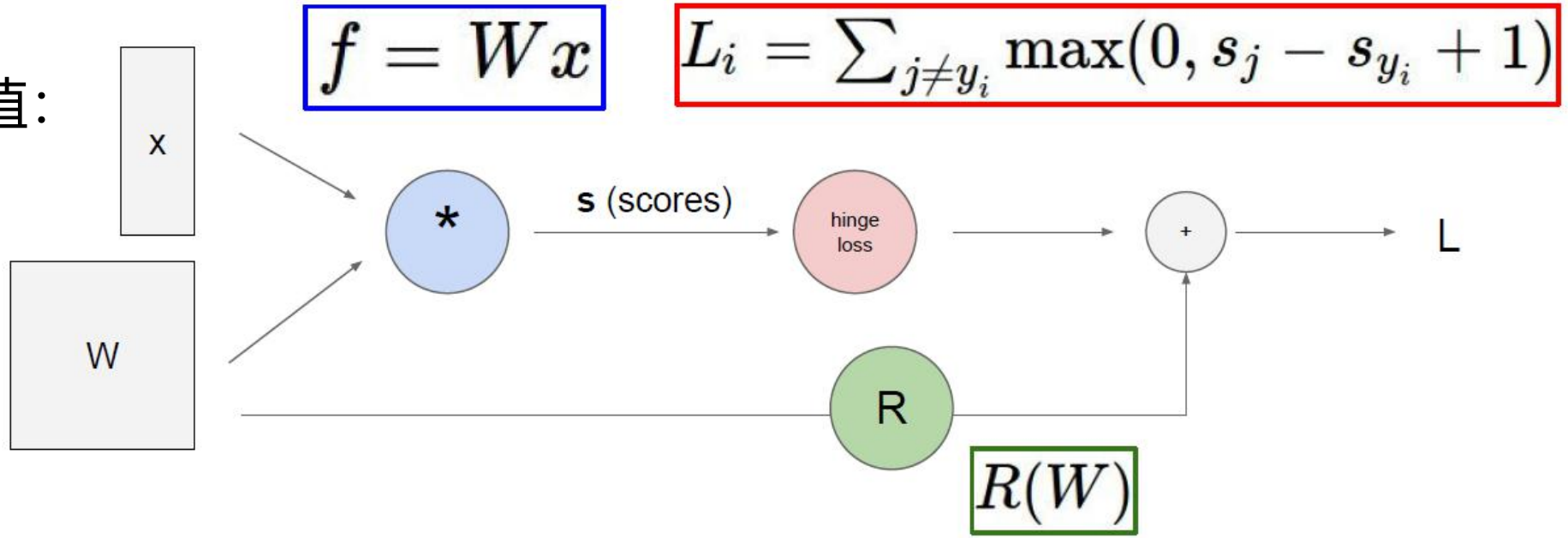
计算损失值：$$L_i=-\log P(Y=y_i|X=x_i)$$



神经网络基础

- 前向传播

得出损失值：



如何更新模型呢？这个就交给反向传播了（梯度下降）

线性回归

- 梯度下降

  - 引入：当我们得到了一个目标函数后，如何进行求解？
    直接求解？（并不一定可解，线性回归可以当做是一个特例）

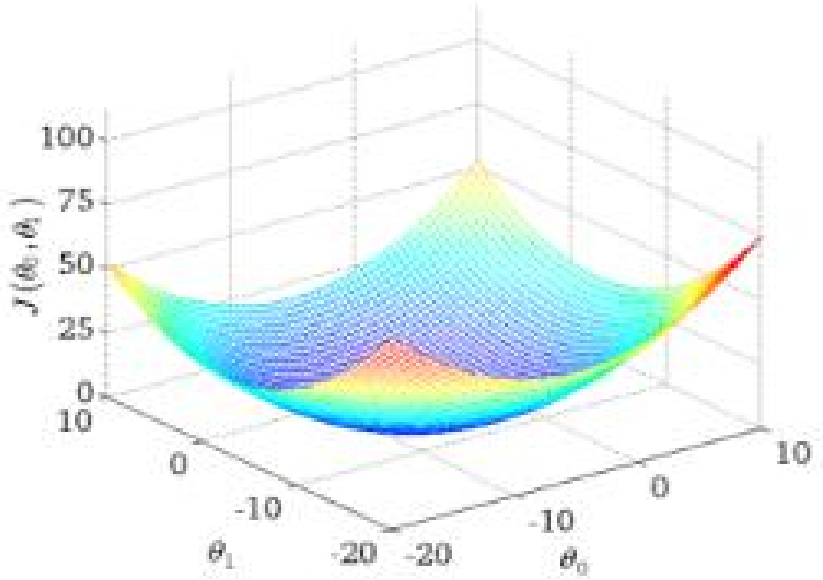
  - 常规套路：机器学习的套路就是我交给机器一堆数据，然后告诉它
    什么样的学习方式是对的（目标函数），然后让它朝着这个方向去做

  - 如何优化：一口吃不成个胖子，我们要静悄悄的一步步的完成迭代
    （每次优化一点点，累积起来就是个大成绩了）

线性回归

- 梯度下降

目标函数： $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$



寻找山谷的最低点，也就是我们的目标函数终点
（什么样的参数能使得目标函数达到极值点）

下山分几步走呢？（更新参数）

（1）：找到当前最合适的方向

（2）：走那么一小步，走快了该”跌倒“了

（3）：按照方向与步伐去更新我们的参数

线性回归

- 梯度下降，目标函数： $J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))^2$

  - 批量梯度下降： $\frac{\partial J(\theta)}{\partial \theta_j}=-\frac{1}{m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))x_j^i \quad \theta_j^{'}=\theta_j+\frac{1}{m}\sum_{i=1}^{m}(y^i-h_\theta(x^i))x_j^i$

    （容易得到最优解，但是由于每次考虑所有样本，速度很慢）

  - 随机梯度下降： $\theta_j^{'}=\theta_j+(y^i-h_\theta(x^i))x_j^i$

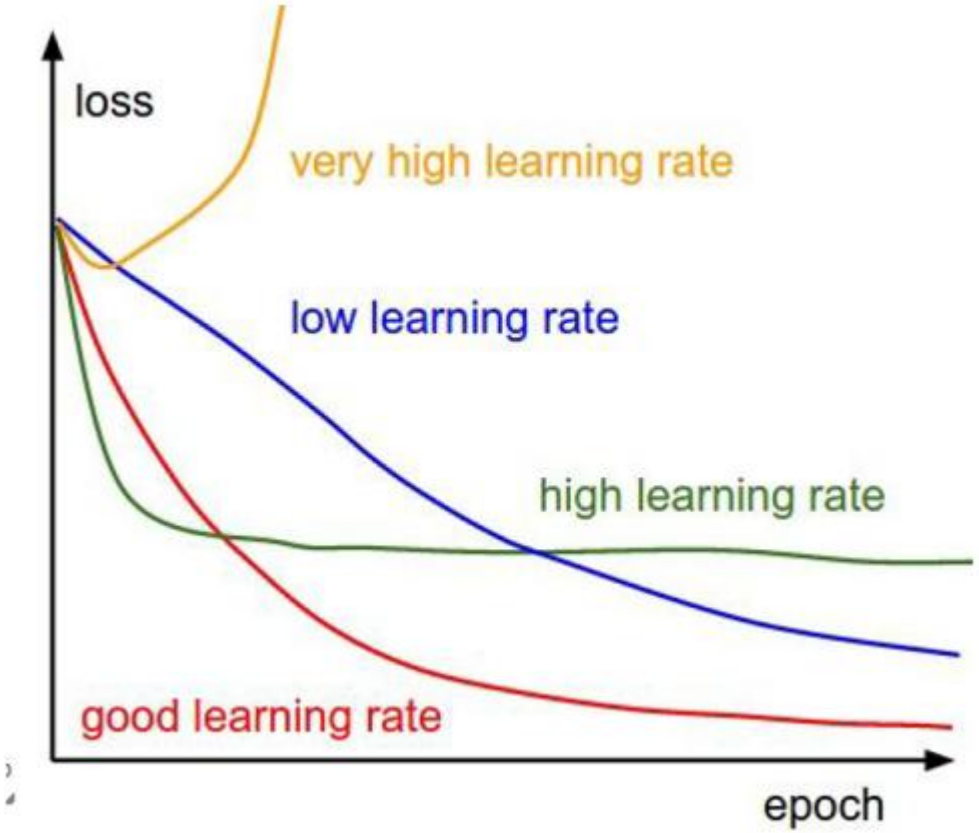
    （每次找一个样本，迭代速度快，但不一定每次都朝着收敛的方向）

  - 小批量梯度下降法： $\theta_j:=\theta_j-\alpha\frac{1}{10}\sum_{k=i}^{i+9}(h_\theta(x^{(k)})-y^{(k)})x_j^{(k)}$

    （每次更新选择一小部分数据来算，实用！）

线性回归

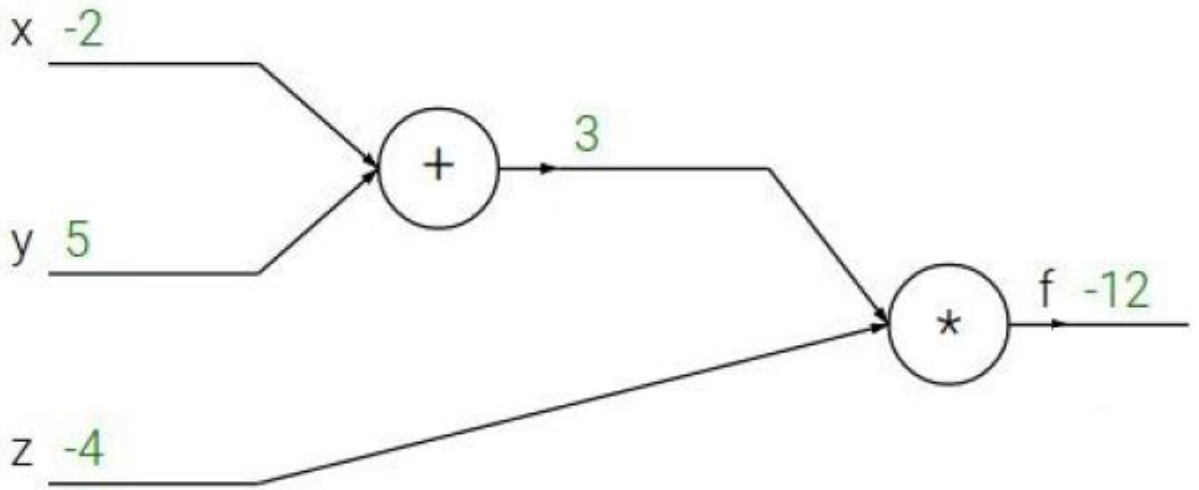
- 梯度下降
  - 学习率（步长）：对结果会产生巨大的影响，一般小一些
  - 如何选择：从小的时候，不行再小
  - 批处理数量：32，64，128都可以，很多时候还得考虑内存和效率



神经网络基础

- 反向传播

- 简单的栗子：



$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

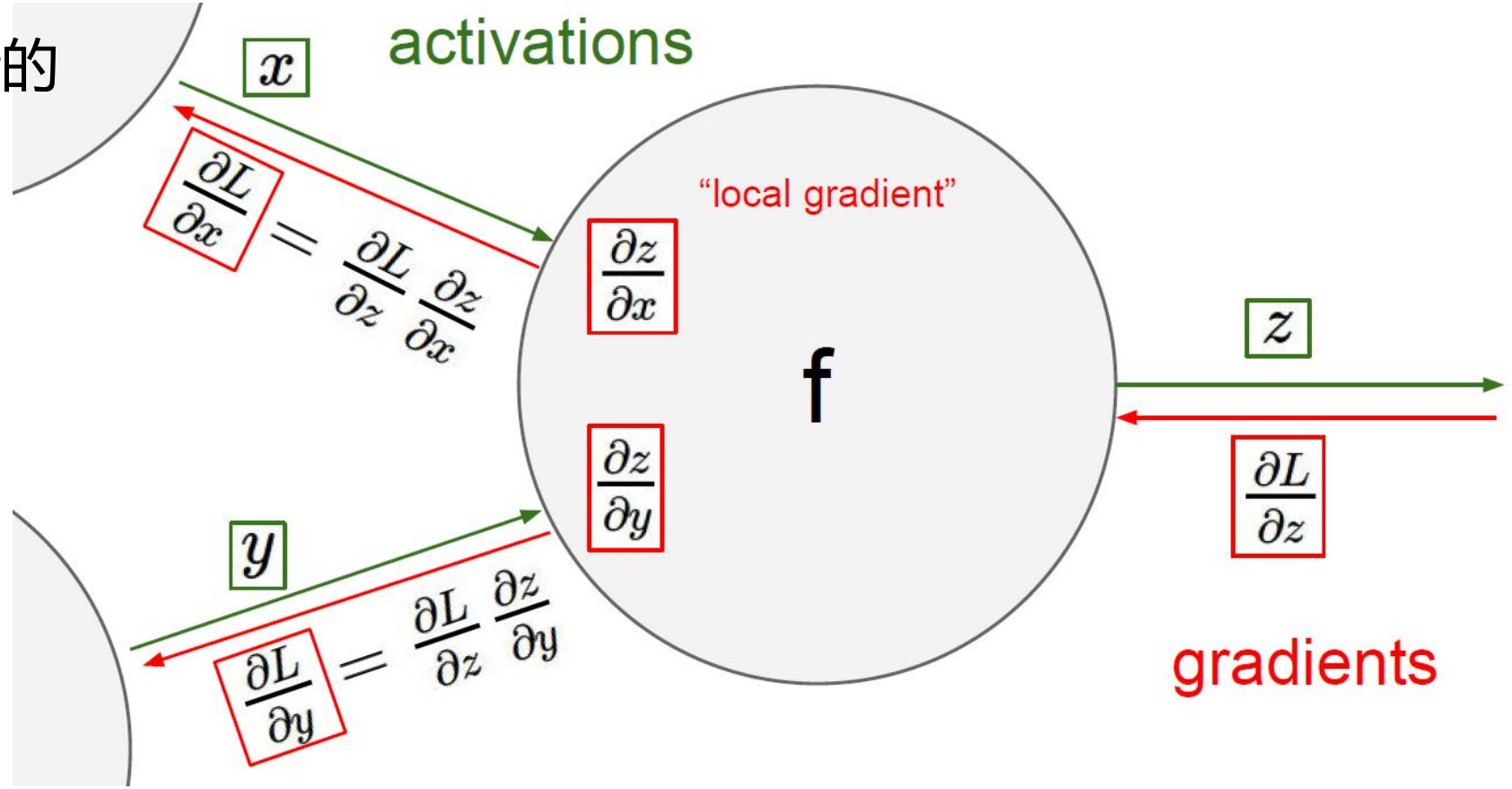
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

神经网络基础

- 链式法则

  - 梯度是一步一步传的

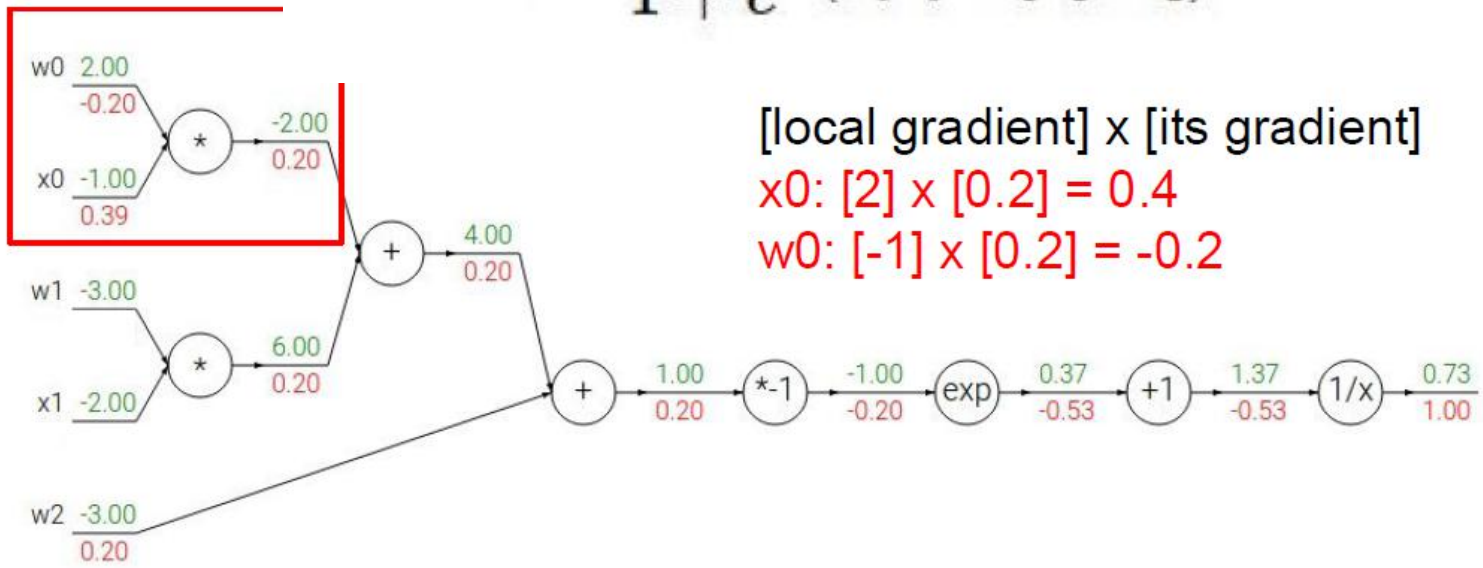


神经网络基础

反向传播

复杂的栗子：

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



[local gradient] x [its gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

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

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

神经网络基础

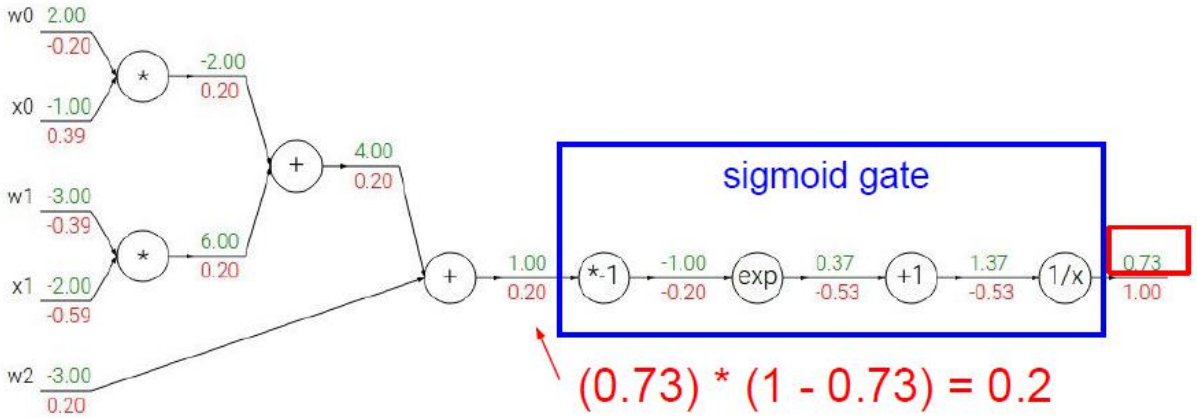
反向传播

可以一大块一大块的计算吗?

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

$$\sigma(x)=\frac{1}{1+e^{-x}}$$ sigmoid function

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



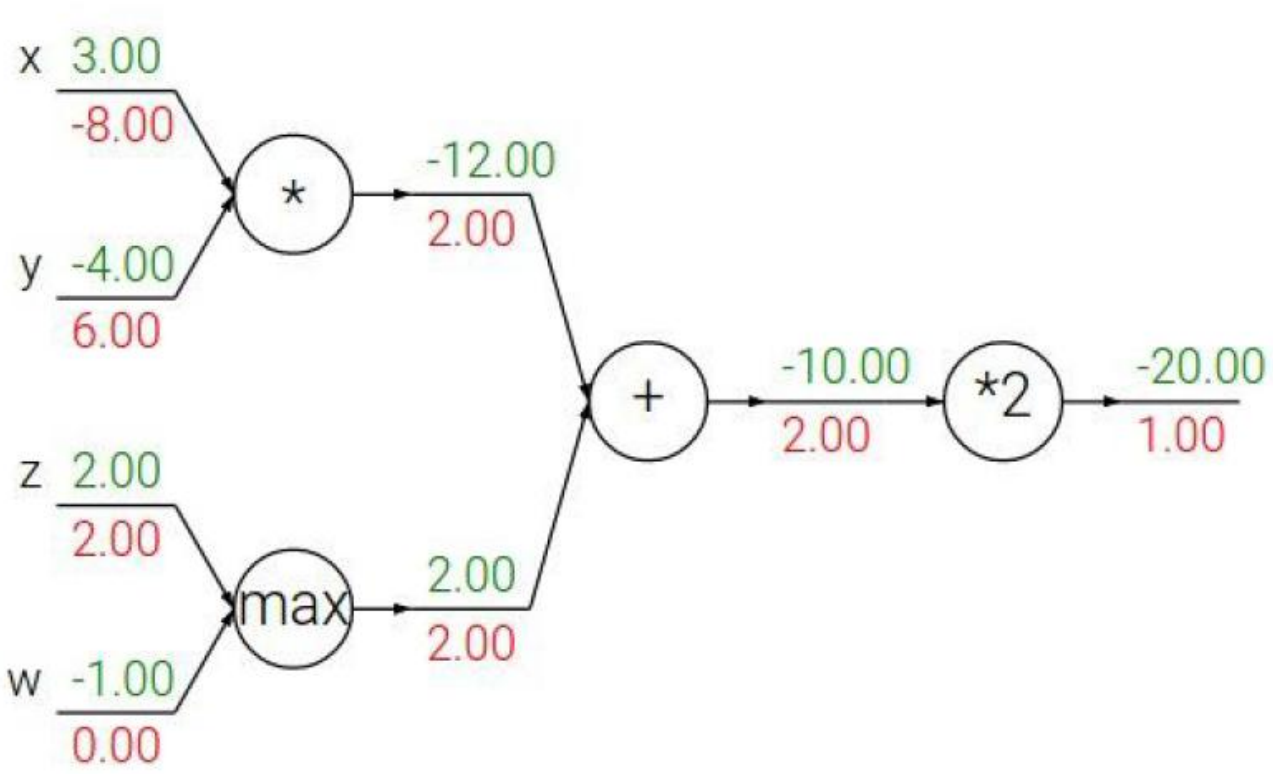
神经网络基础

- 反向传播

  - 加法门单元：均等分配

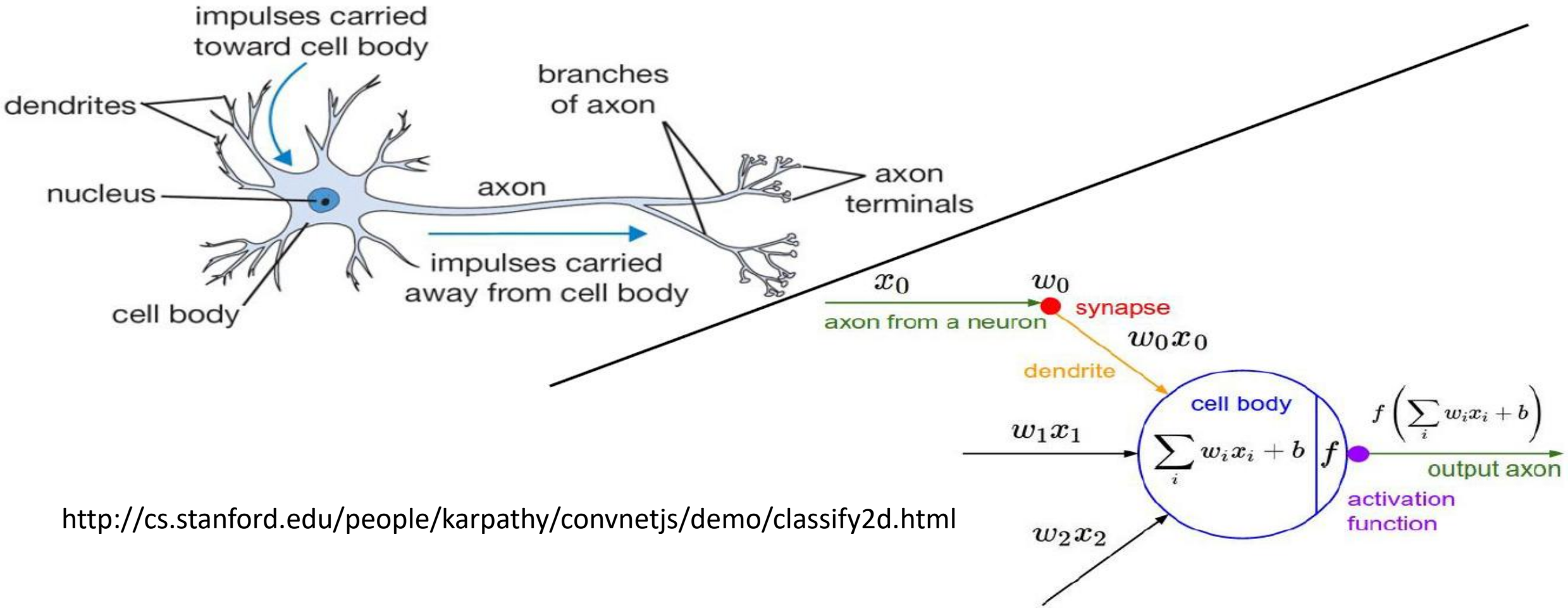
    MAX门单元：给最大的

    乘法门单元：互换的感觉



神经网络

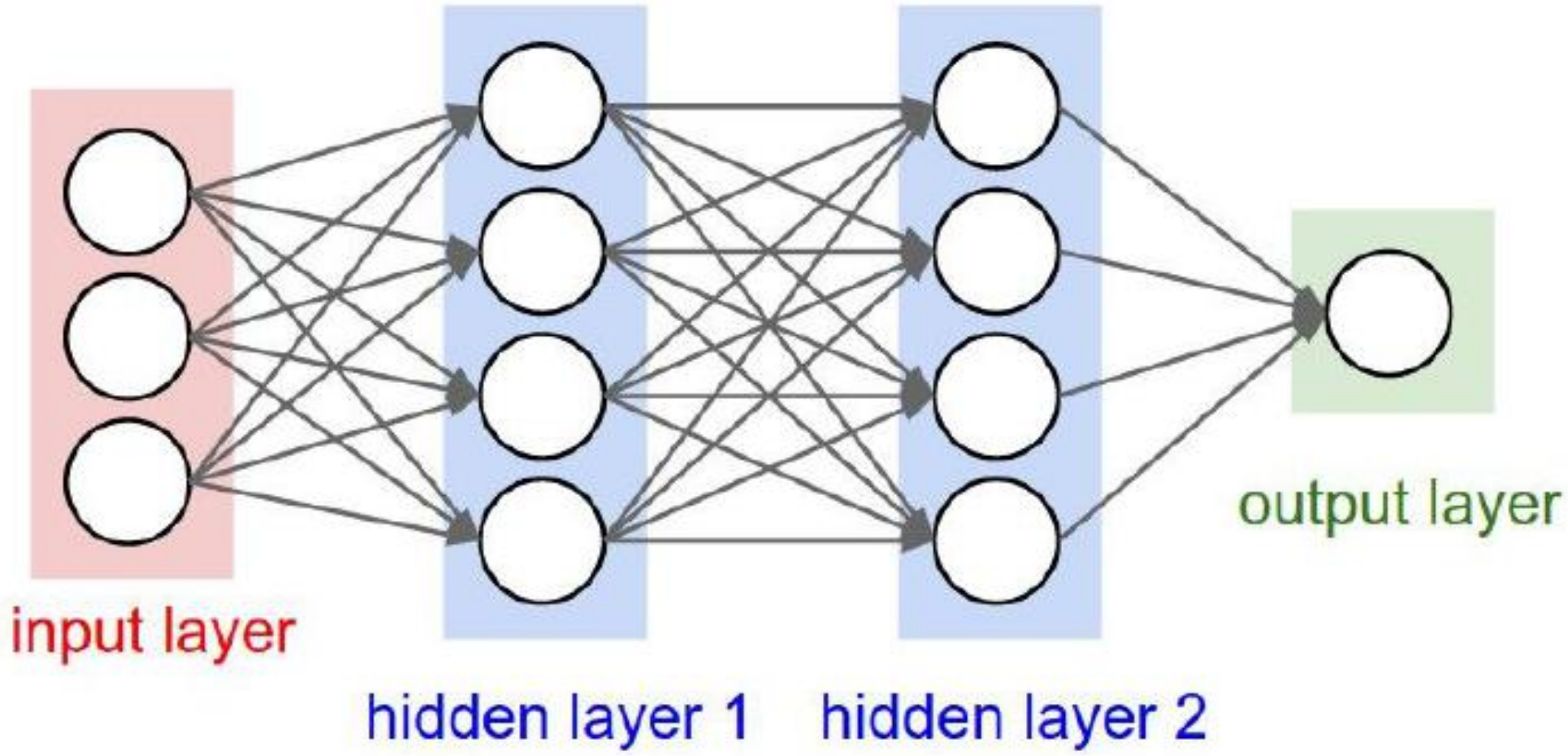
- 整体架构



http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

神经网络

- 整体架构
  - 层次结构
  - 神经元
  - 全连接
  - 非线性



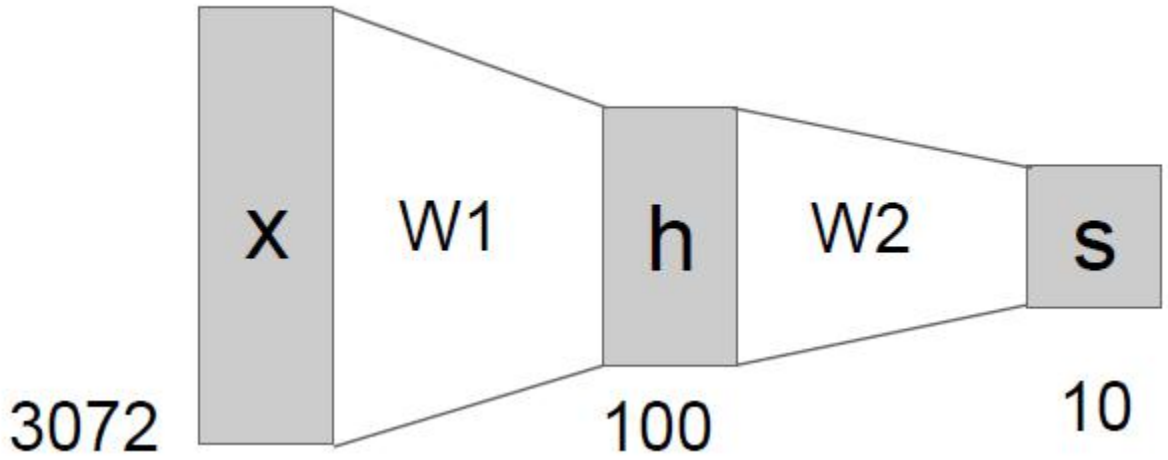
神经网络

- 整体架构

  - 线性方程：$f = Wx$

  - 非线性方程：$f = W_2 \max(0, W_1 x)$

  - 计算结果：



神经网络

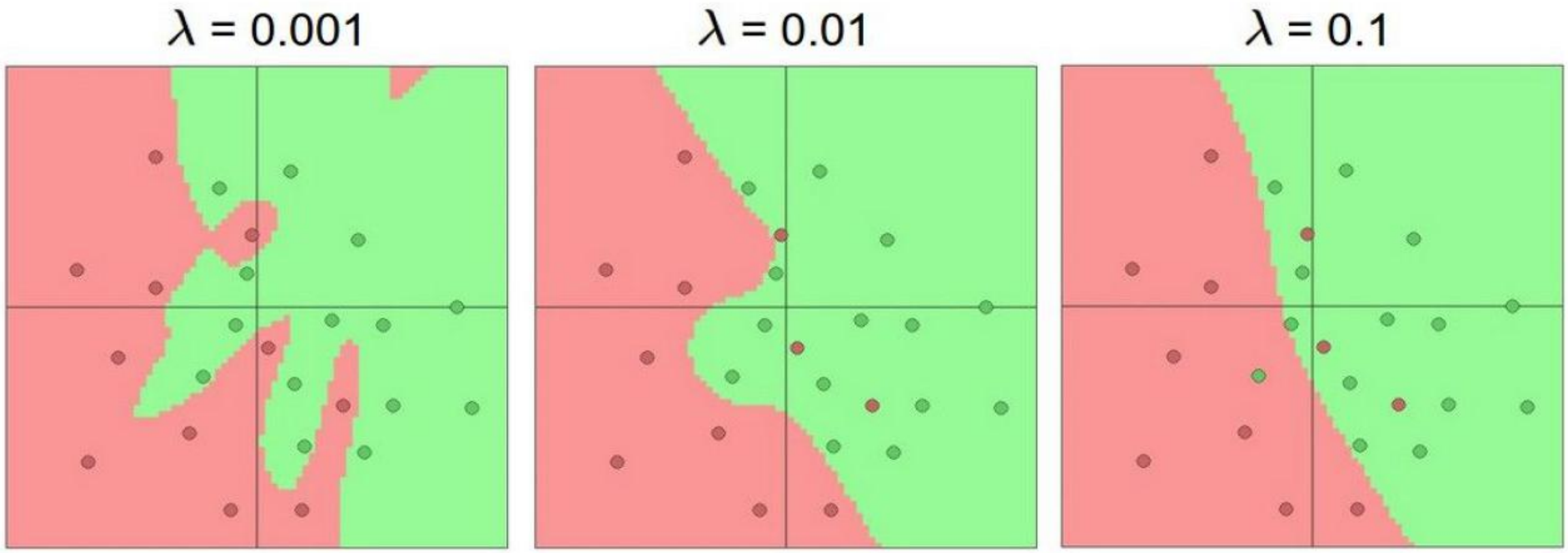
- 整体架构
  - 基本机构：$f = W_2 \max(0, W_1 x)$
  - 继续堆叠一层：$f = W_3 \max(0, W_2 \max(0, W_1 x))$
  - 神经网络的强大之处在于，用更多的参数来拟合复杂的数据

    （参数多到多少呢？百万级别都是小儿科了，但是参数越多越好吗？）

神经网络

- 正则化的作用

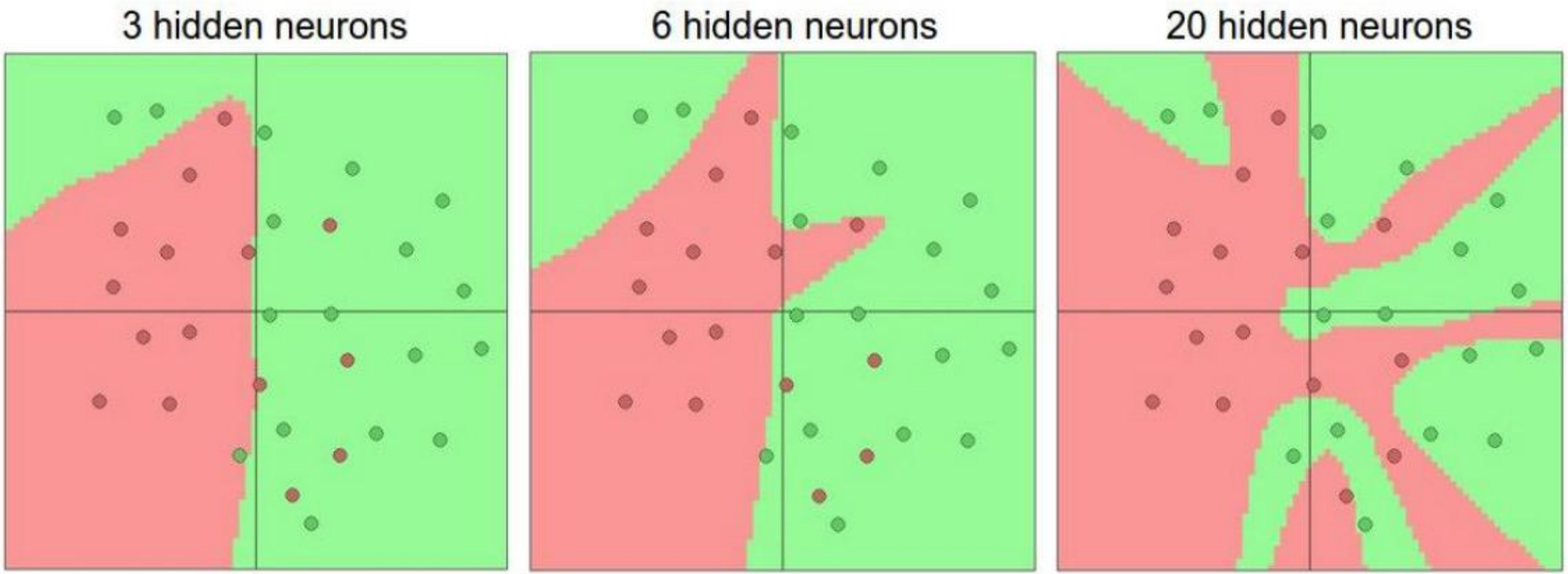
  - 惩罚力度对结果的影响：



神经网络

- 神经元

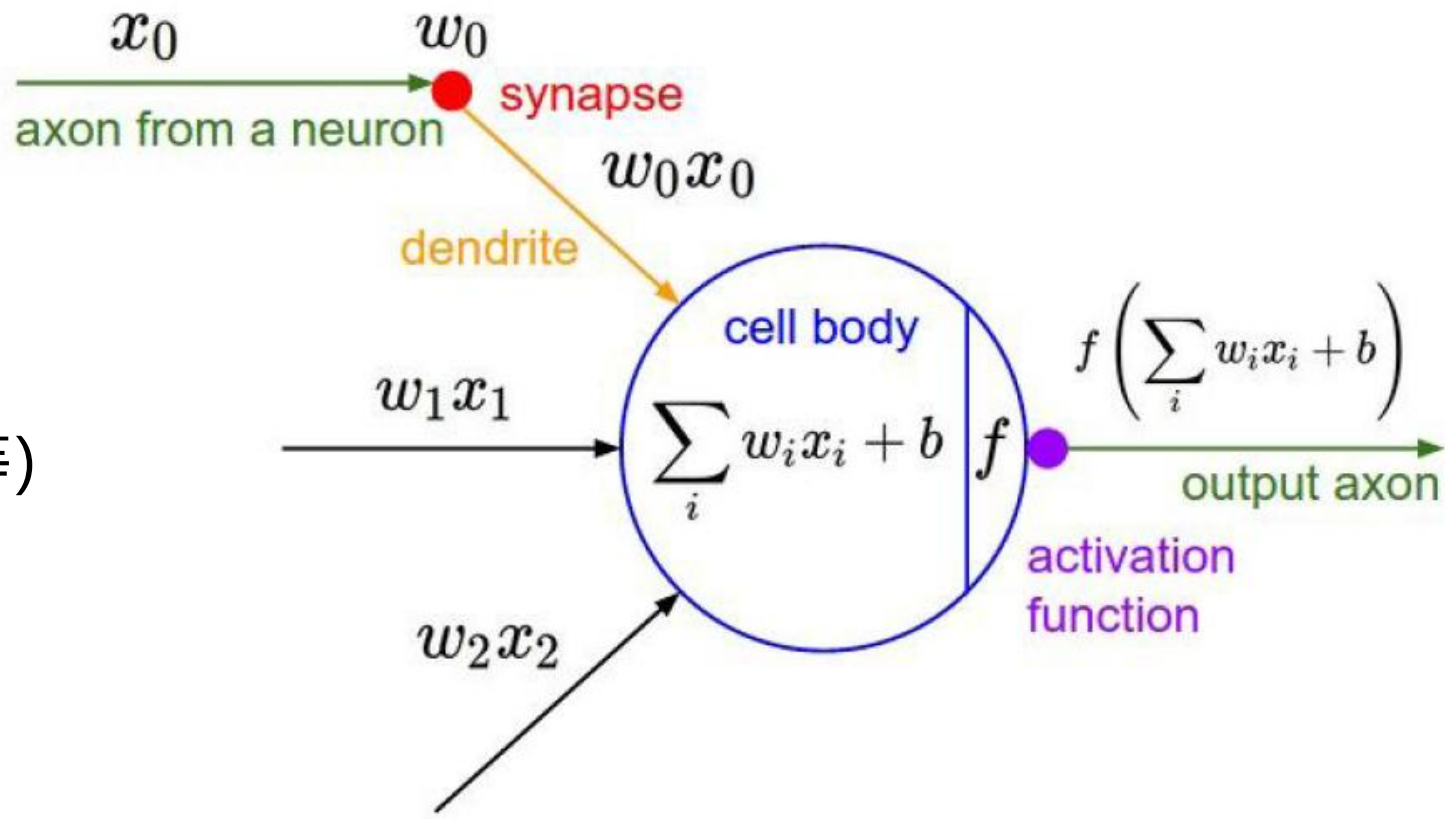
- 参数个数对结果的影响：



神经网络

- 激活函数
  - 非常重要的一部分
  - 常用的激活函数

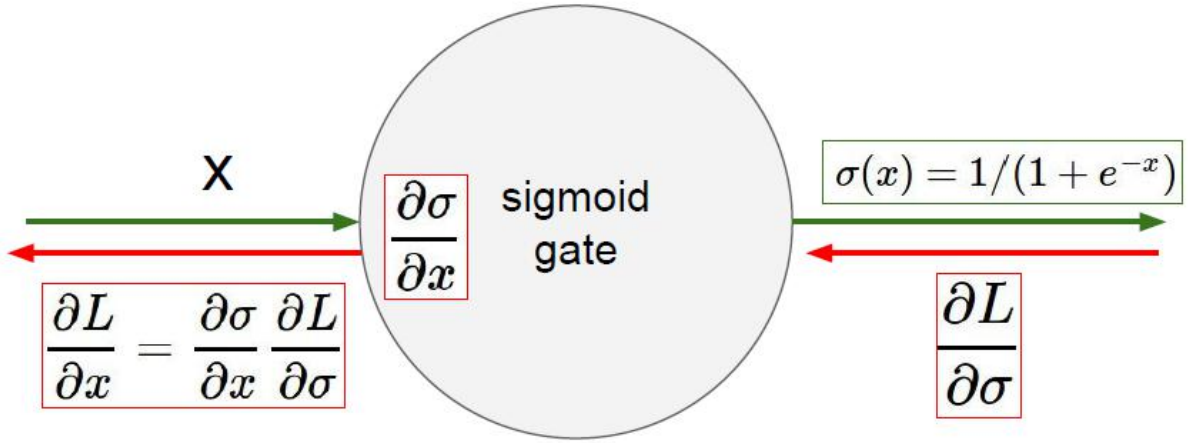
    (Sigmoid,Relu,Tanh 等)

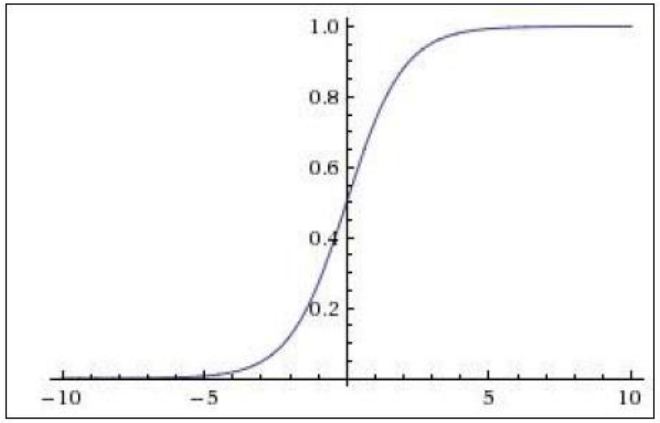


神经网络

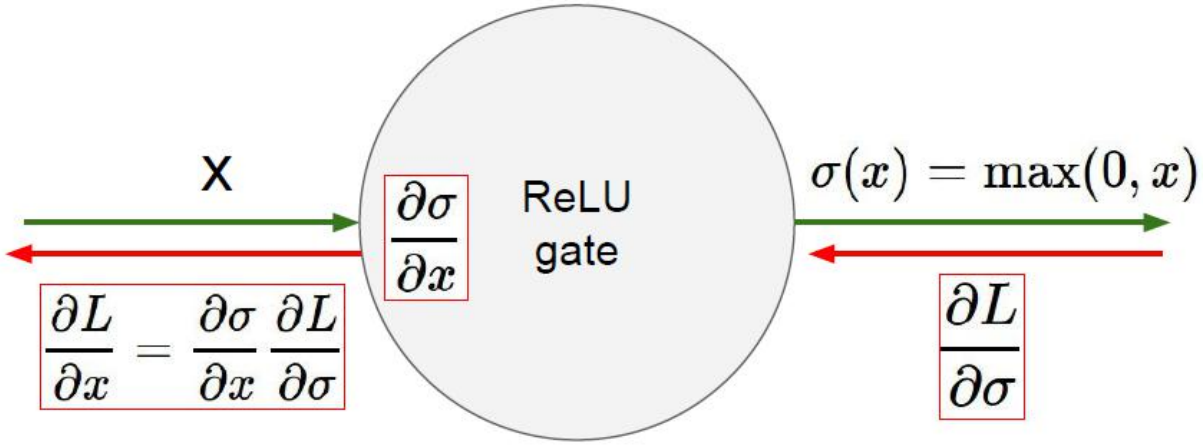
- 激活函数对比

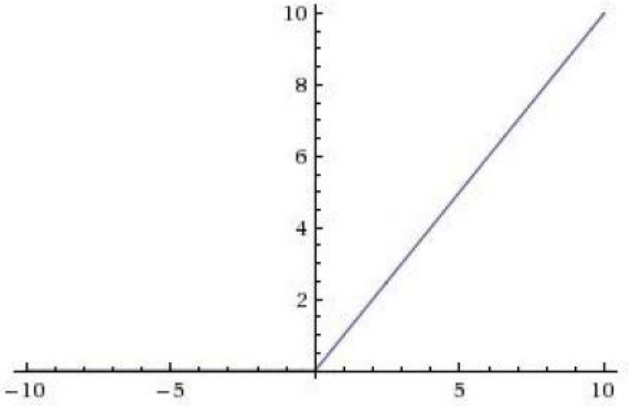
- Sigmoid:





- Relu:

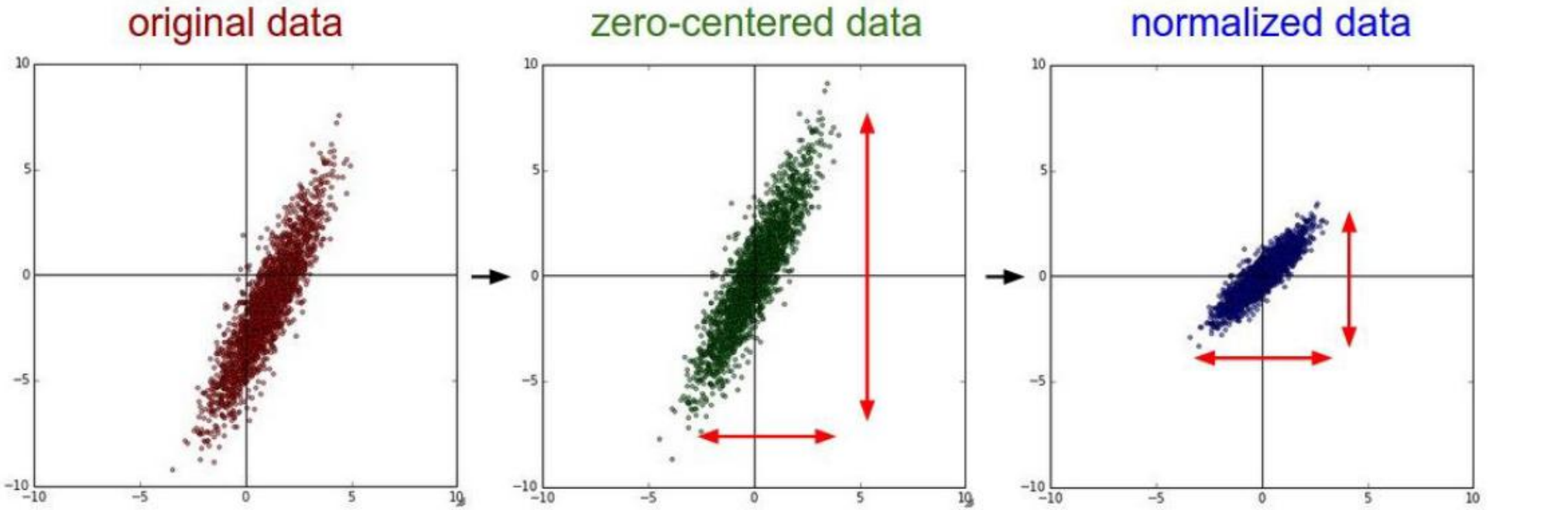




神经网络

- 数据预处理

- 不同的预处理结果会使得模型的效果发生很大的差异!



神经网络

- 参数初始化

  - 参数初始化同样非常重要!

  - 通常我们都使用随机策略来进行参数初始化

```
W = 0.01* np.random.randn(D,H)
```

神经网络

- DROP-OUT（传说中的七伤拳）

  - 过拟合是神经网络非常头疼的一个大问题！