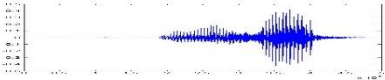


深度学习与非凸优化

AI100学院
2017年9月

- 深度学习简介
- 非凸优化问题
- 深度学习训练技巧
- 卷积神经网络

机器学习中的函数拟合

- 语音识别 $f(\text{语音波形}) = \text{“How are you”}$

- 图像识别 $f(\text{猫咪照片}) = \text{“猫”}$

- 围棋对战 $f(\text{棋局状态}) = \text{“5-5” (下一步棋)}$

- 对话系统 $f(\text{“Hi” (用户询问)}) = \text{“Hello” (系统反馈)}$



图像识别框架

图像:

$$f(\text{image of cat}) = \text{"cat"}$$

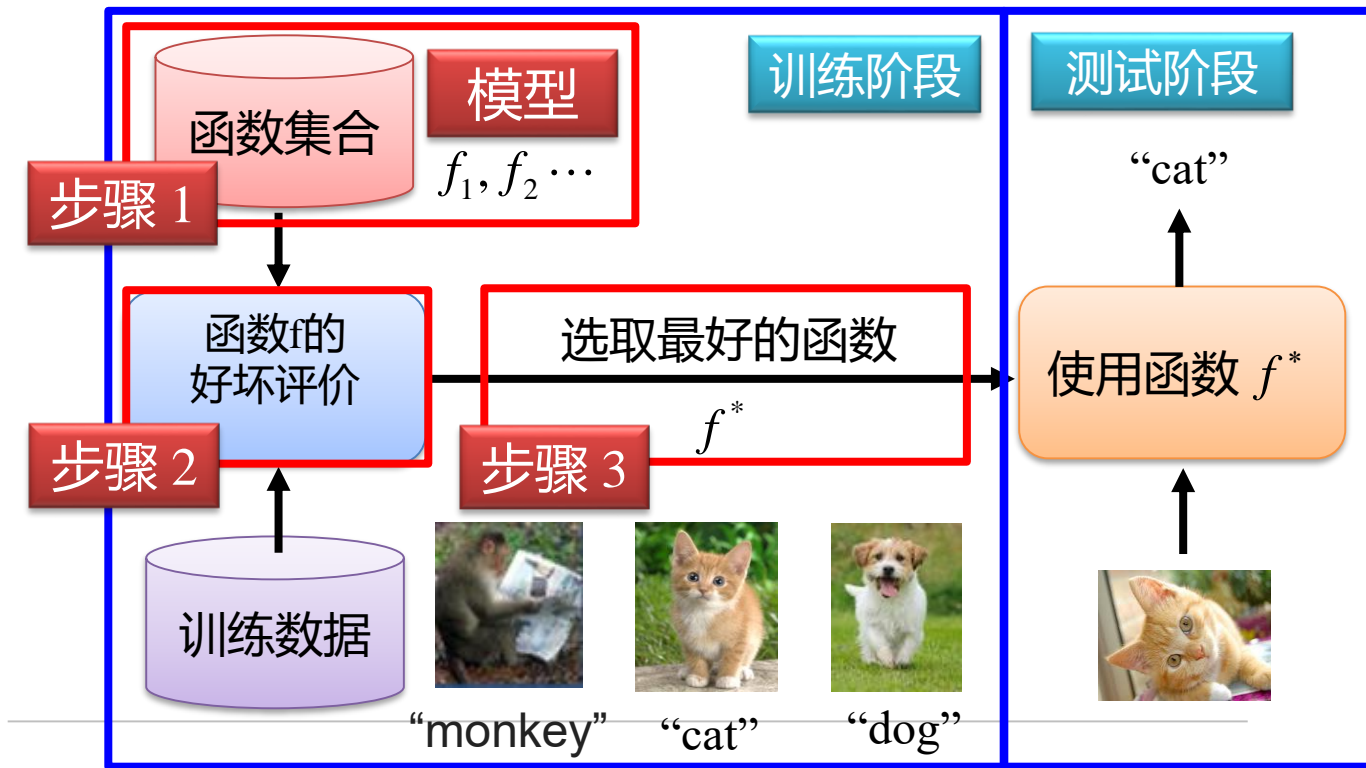


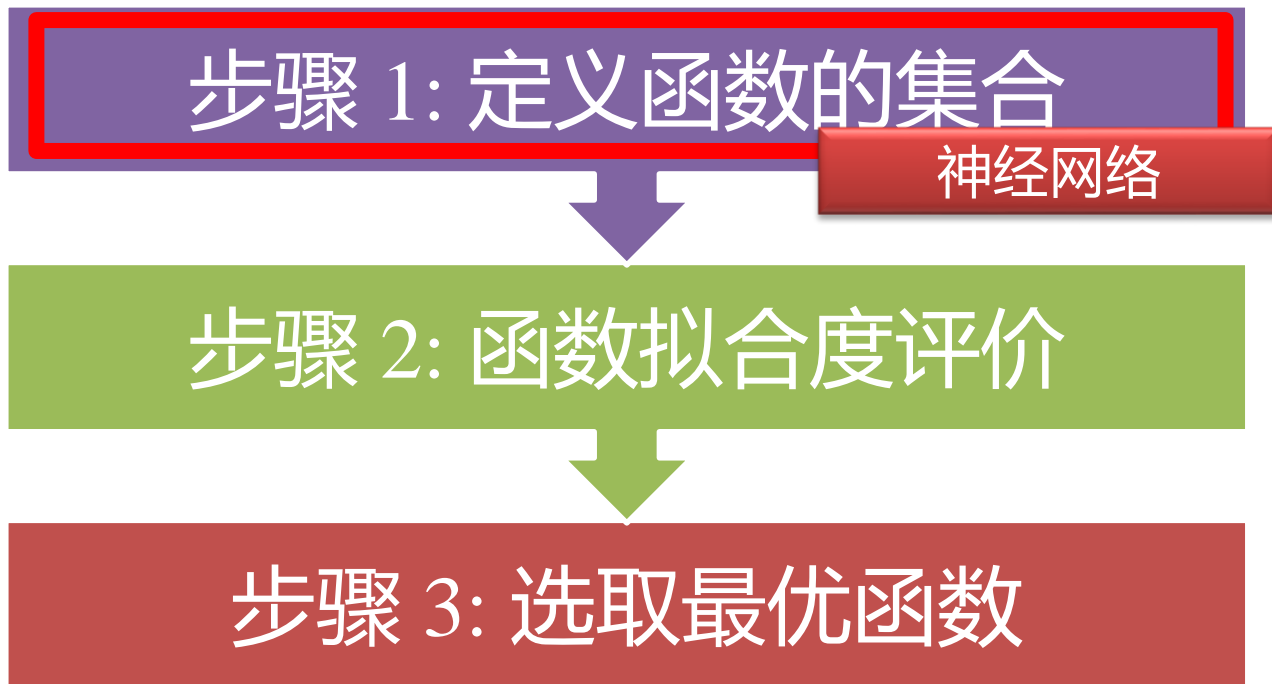


图像识别步骤

图像识别:

$$f(\text{image of cat}) = \text{"cat"}$$



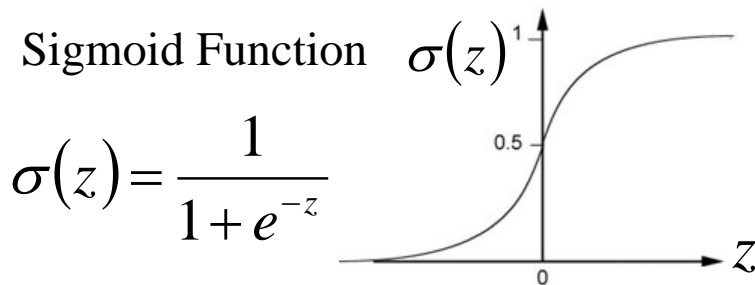
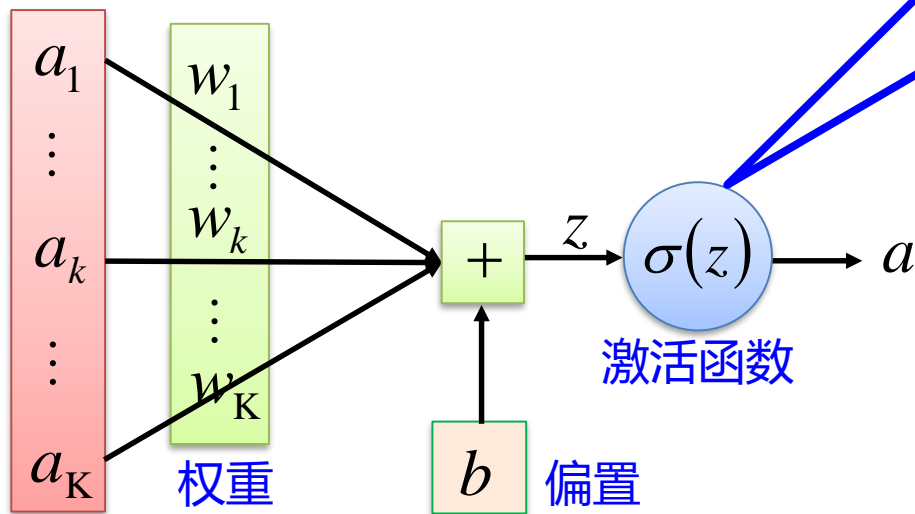




神经网络简介

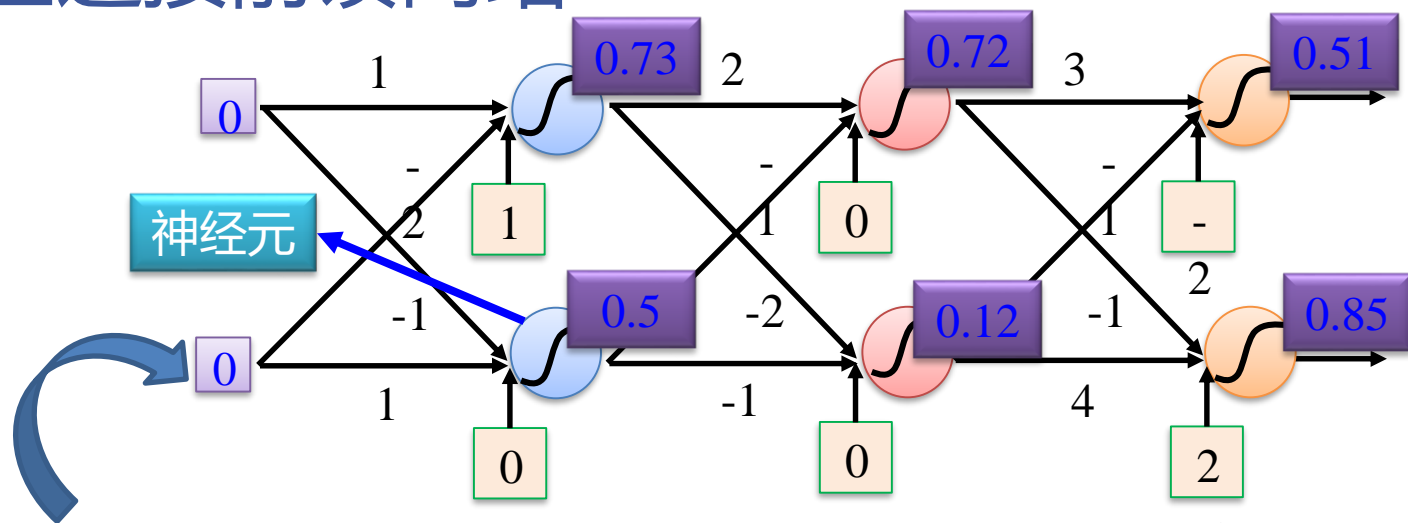
神经元

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



权重与偏置记为网络参数 θ

全连接前馈网络



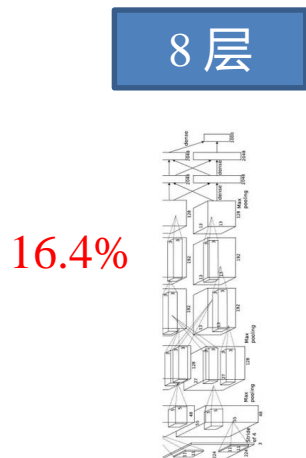
输入、输出均为向量的函数 $f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$ $f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$

给定参数 θ , 函数即定义好

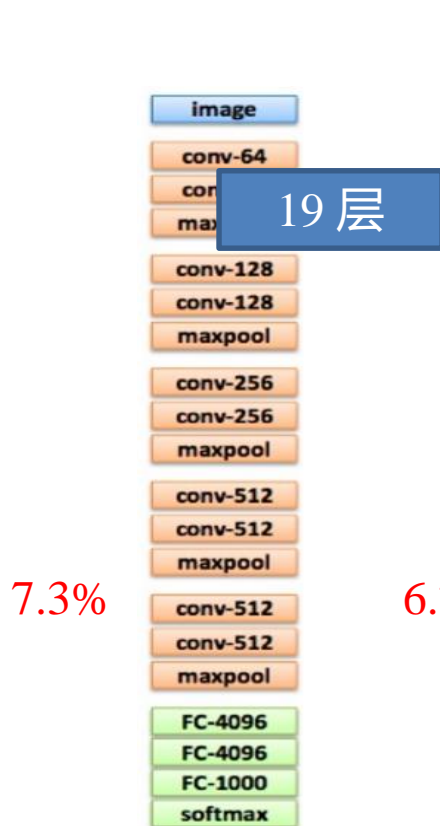
给定函数网络结构, 即定义函数集合

深度神经网络具有非常多的隐藏层

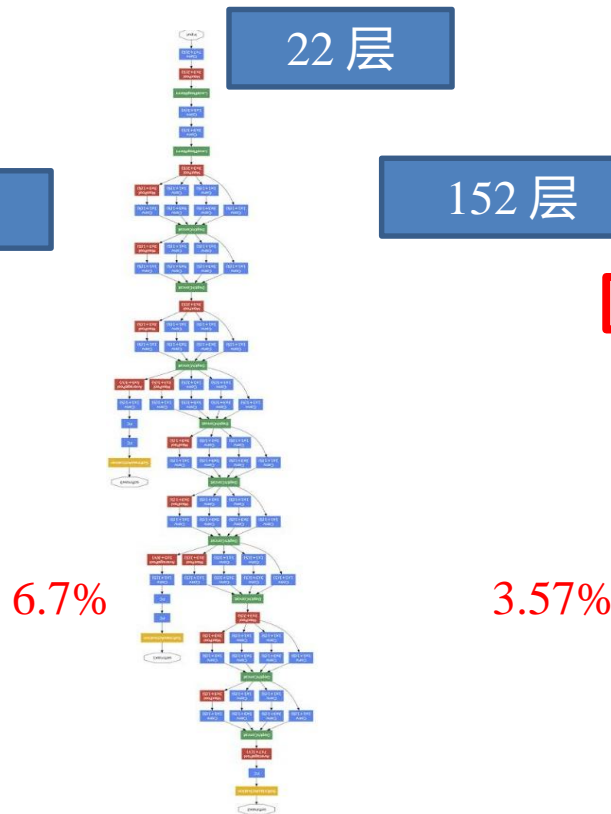
典型深度神经网络



AlexNet (2012)



VGG (2014)



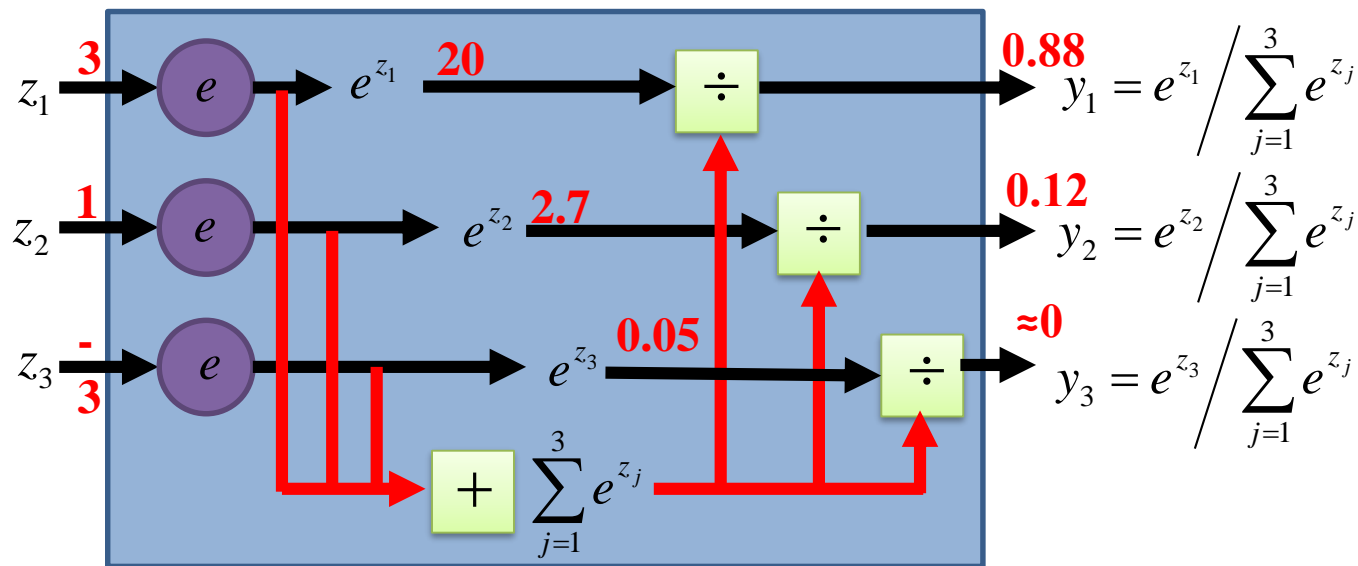
GoogleNet (2014)

Residual Net (2015)

► 输出层示例：Softmax层

- Softmax作为输出层

Softmax Layer



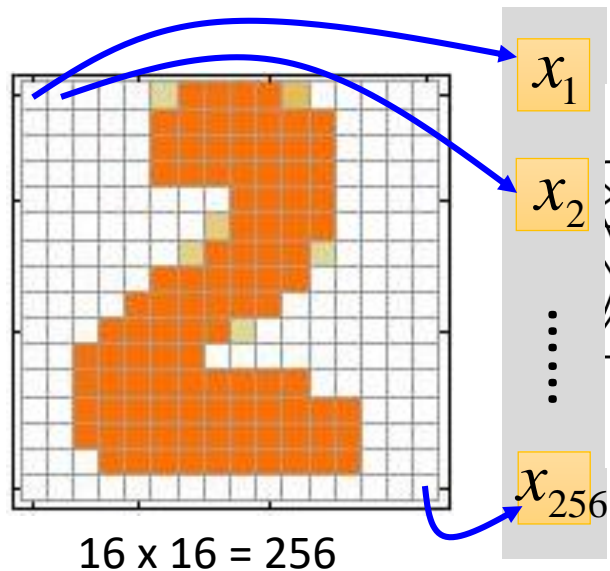
概率输出:

- $1 > y_i > 0$
- $\sum_i y_i = 1$

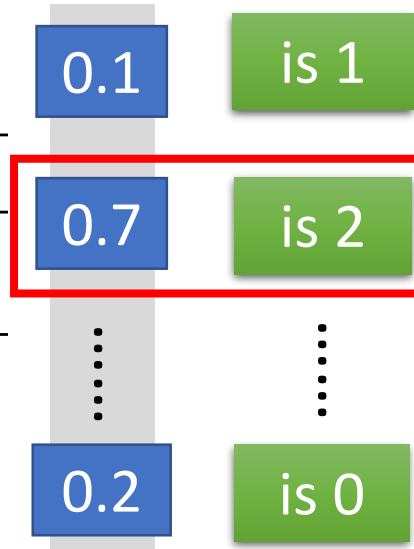
应用示例



• 输入



• 输出



图像中字符为“2”

各维度表示字符信度

步骤 1: 定义函数的集合

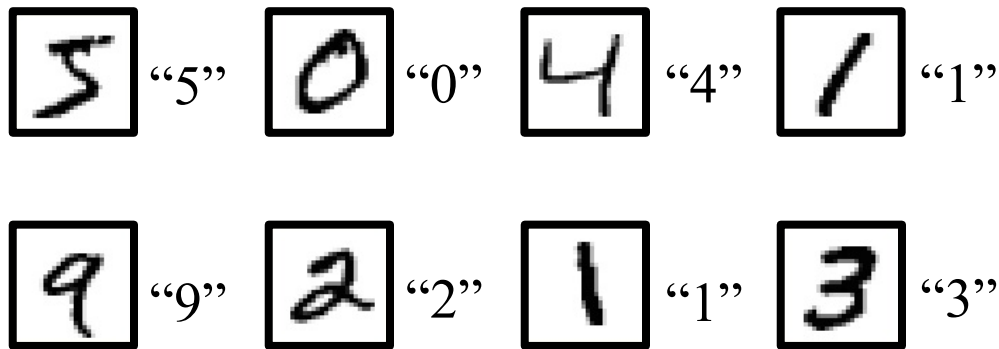


步骤 2: 函数拟合度评价



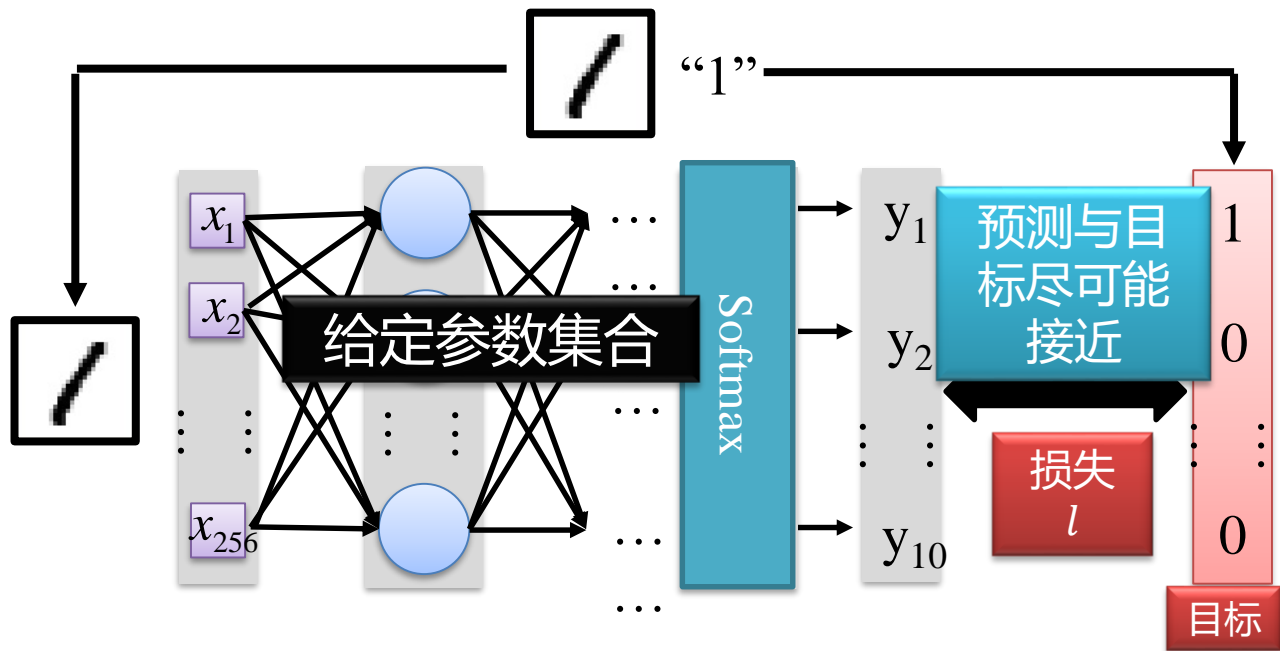
步骤 3: 选取最优函数

- 准备训练数据：图像及其标签



学习目标在训练集上定义

优秀的神经网络使得所有样本的损失越小越好



目标：总体损失越小越好

手段：寻找最小化总体损失的函数

算法：寻找最佳网络参数 θ^*

损失函数可以定义为网络输出与目标的均方误差或交叉熵

步骤 1: 定义函数的集合



步骤 2: 函数拟合度评价



步骤 3: 选取最优函数

► 如何选择最优函数

寻找最小化总体损失 L 的最优网络参数 θ^*

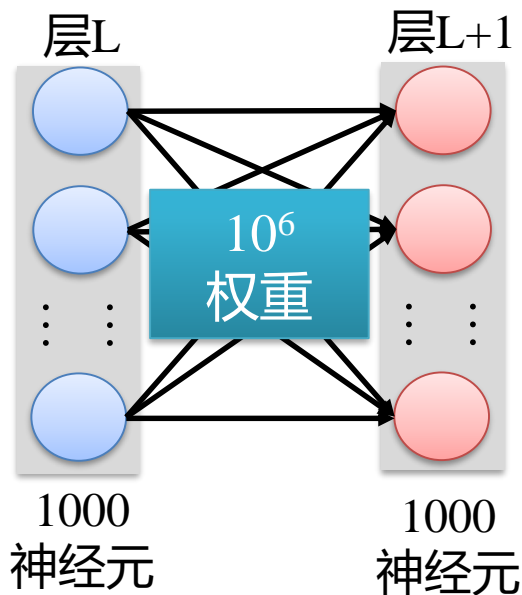
穷举所有可能数值

网络参数 $\theta =$

$\{w_1, w_2, w_3, \dots, b_1, b_2, b_3, \dots\}$

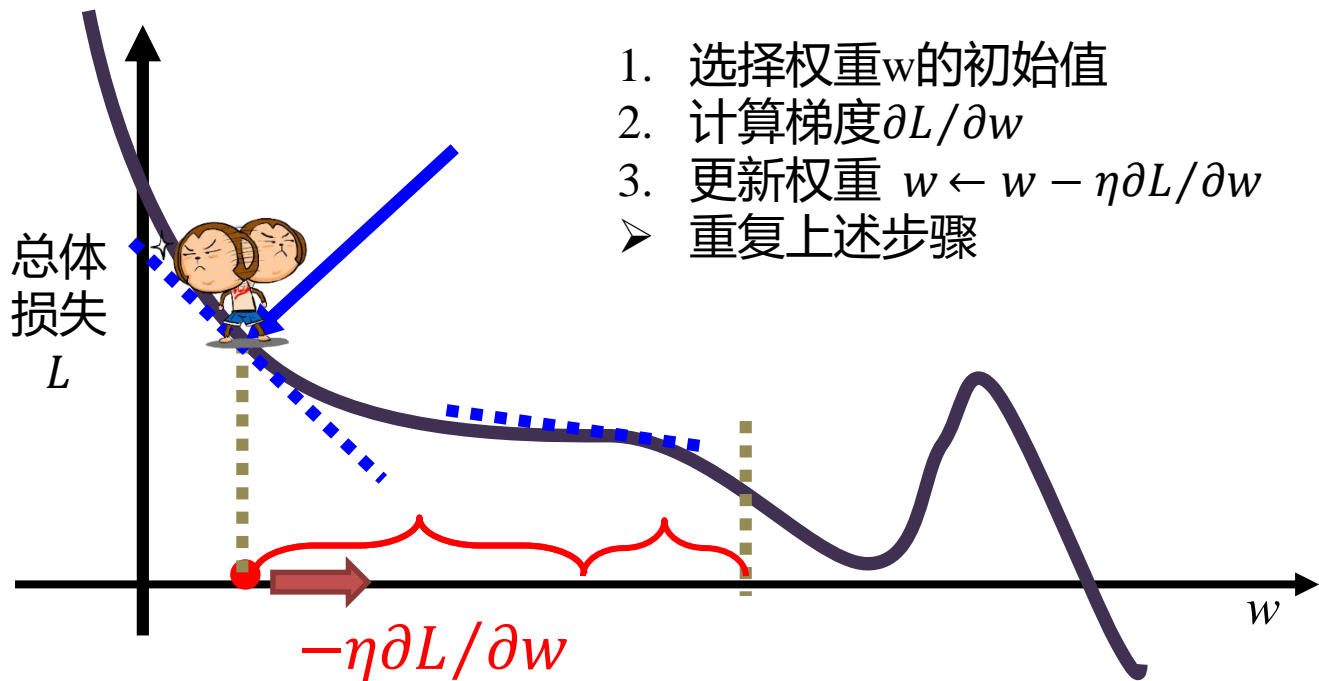
上百万参数

例如语音识别使用 8 层神经元，每层1000个神经元



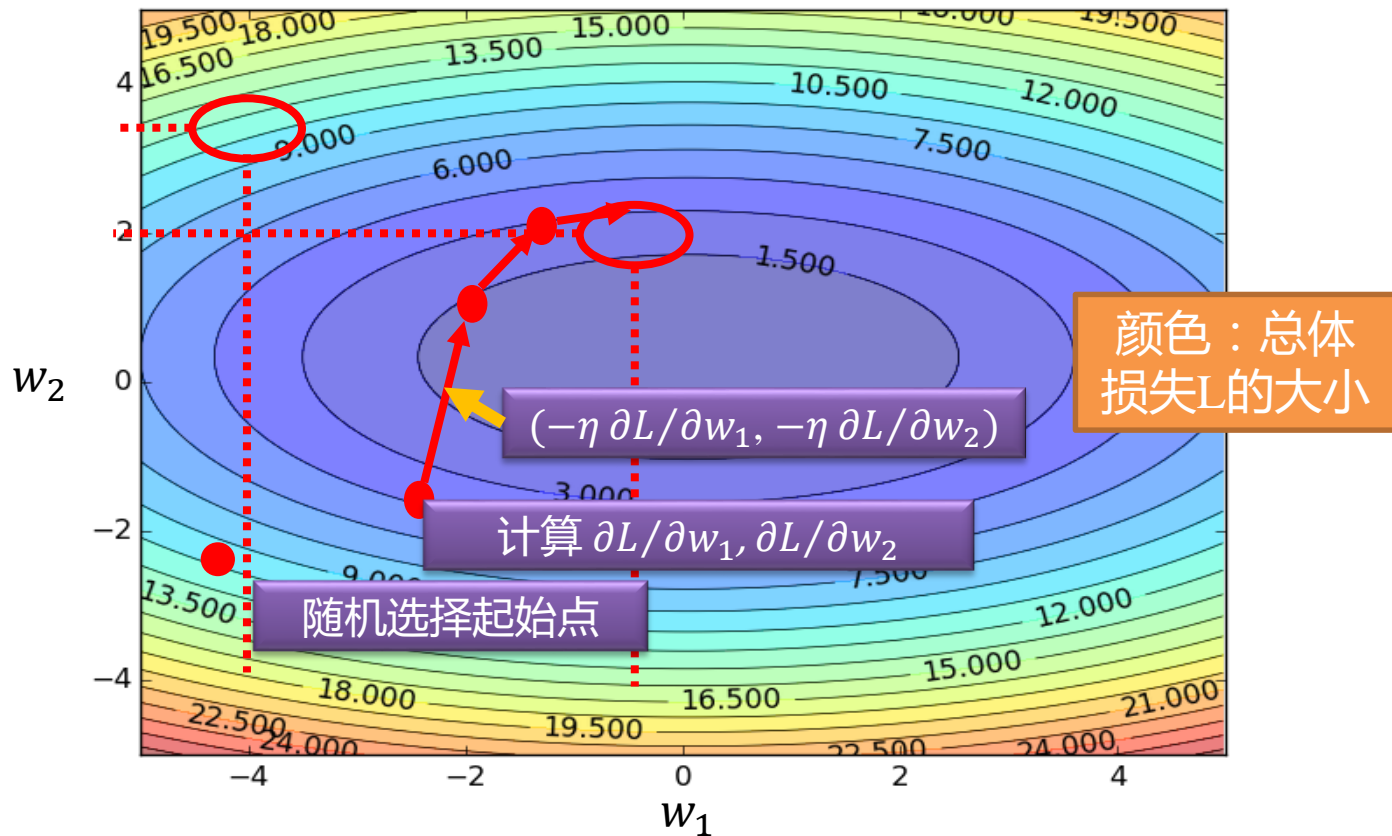
神经网络训练方法——梯度下降

网络参数 $\theta = \{w_1, w_2, \dots, b_1, b_2, \dots\}$

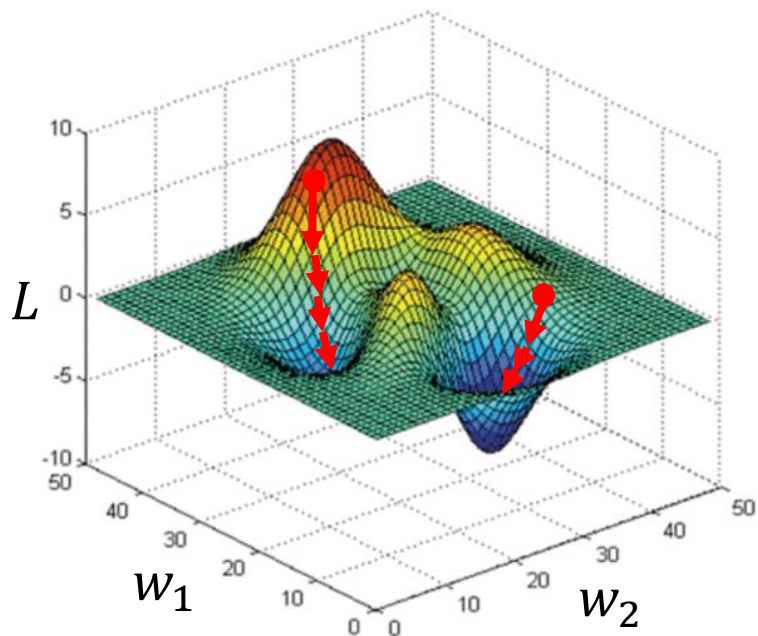
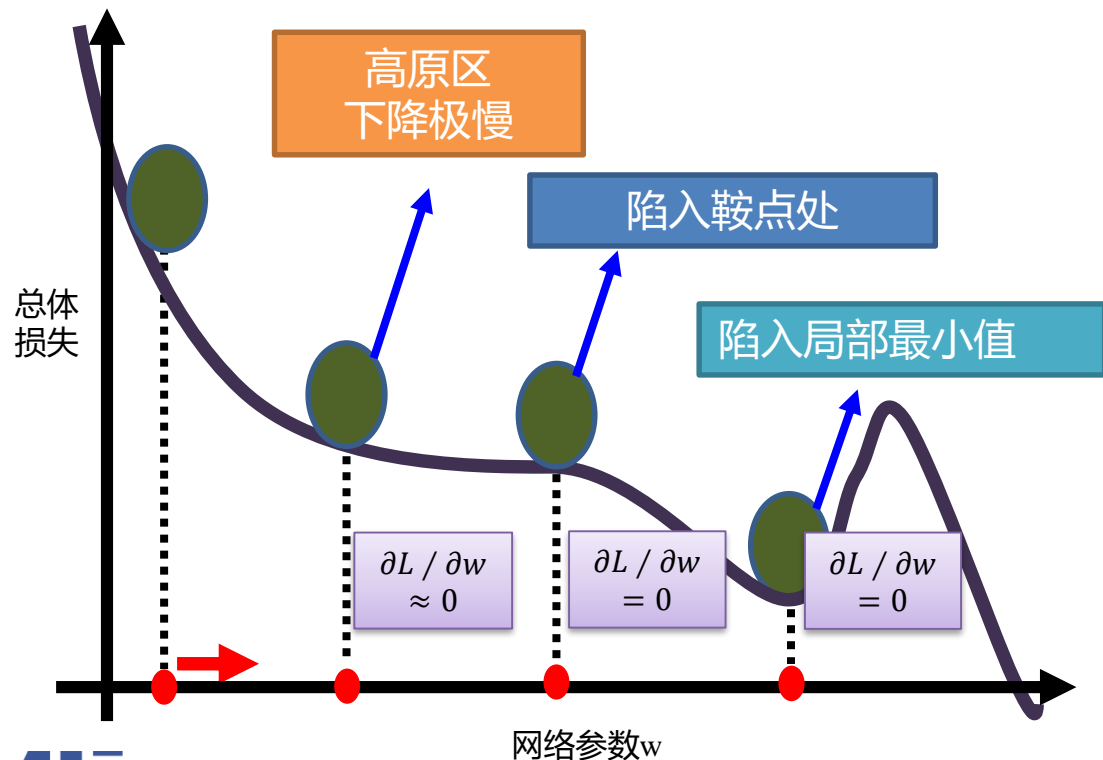


- 深度学习简介
- 非凸优化问题
- 深度学习训练技巧
- 卷积神经网络

梯度下降算法示意图



► 梯度下降难题：局部极小值等



梯度下降无法保证最小值，不同起始点出发，到达不同的最小值



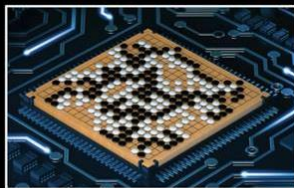
反向传播算法

- 反向传播: 神经网络中计算梯度 $\partial L / \partial w$, 更新权重参数的有效方法。
- 主流深度学习软件包中均采用反向传播算法



theano

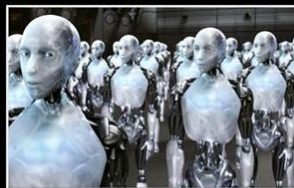
Caffe



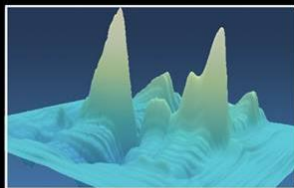
朋友覺得我在



我媽覺得我在



大眾覺得我在



指導教授覺得我在



我以為我在

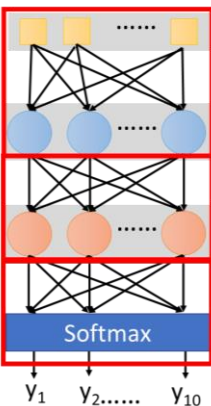


事實上我在

Step 1:
define a set
of function

Step 2:
goodness of
function

Step 3: pick
the best
function

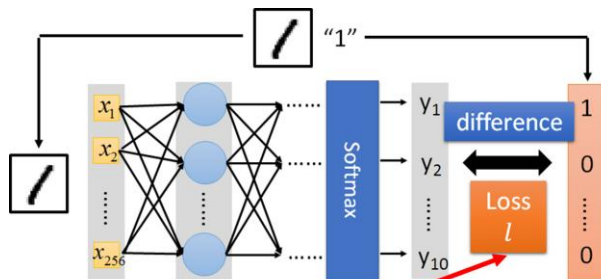


```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,  
                  output_dim=500 ) )  
model.add( Activation('sigmoid') )
```

```
model.add( Dense( output_dim=500 ) )  
model.add( Activation('sigmoid') )
```

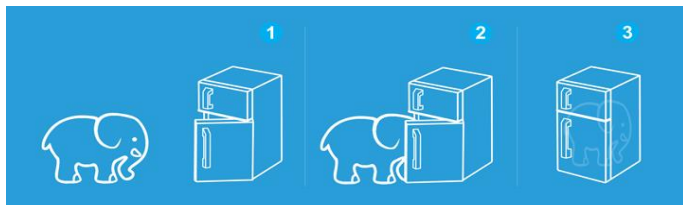
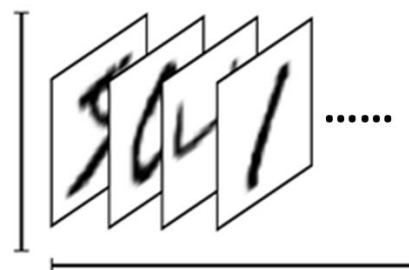
```
model.add( Dense( output_dim=10 ) )  
model.add( Activation('softmax') )
```



```
model.compile( loss='mse',  
               optimizer=SGD(lr=0.1),  
               metrics=['accuracy'] )
```

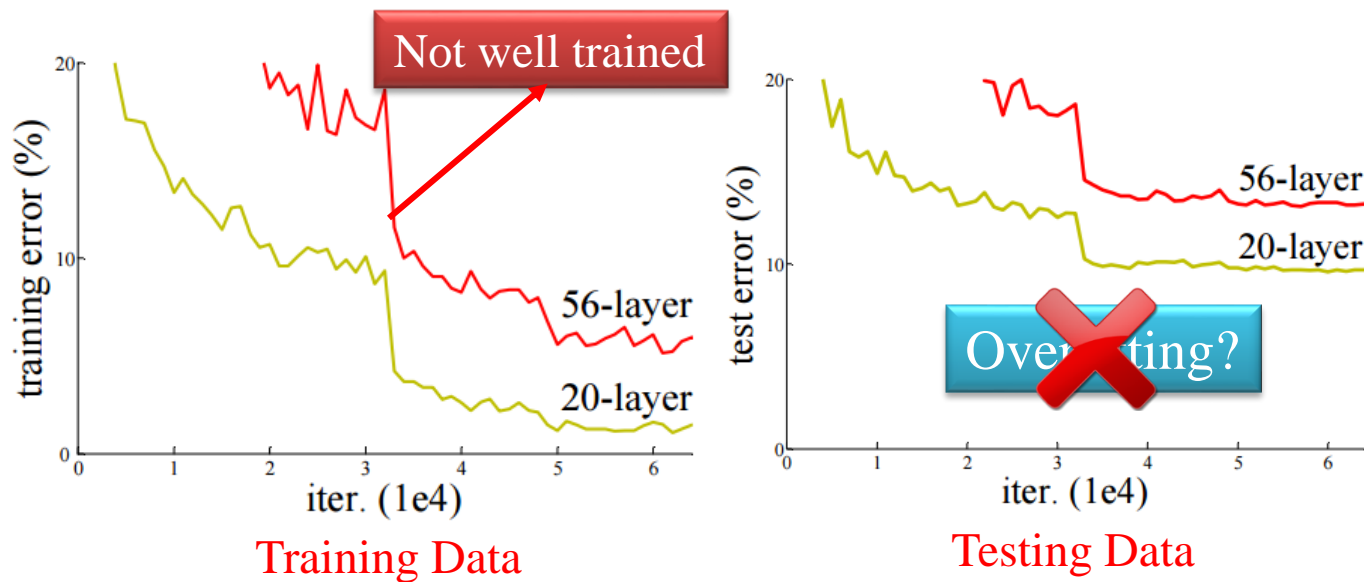
```
model.compile(loss='mse',  
              optimizer=SGD(lr=0.1),  
              metrics=['accuracy'])
```

numpy array

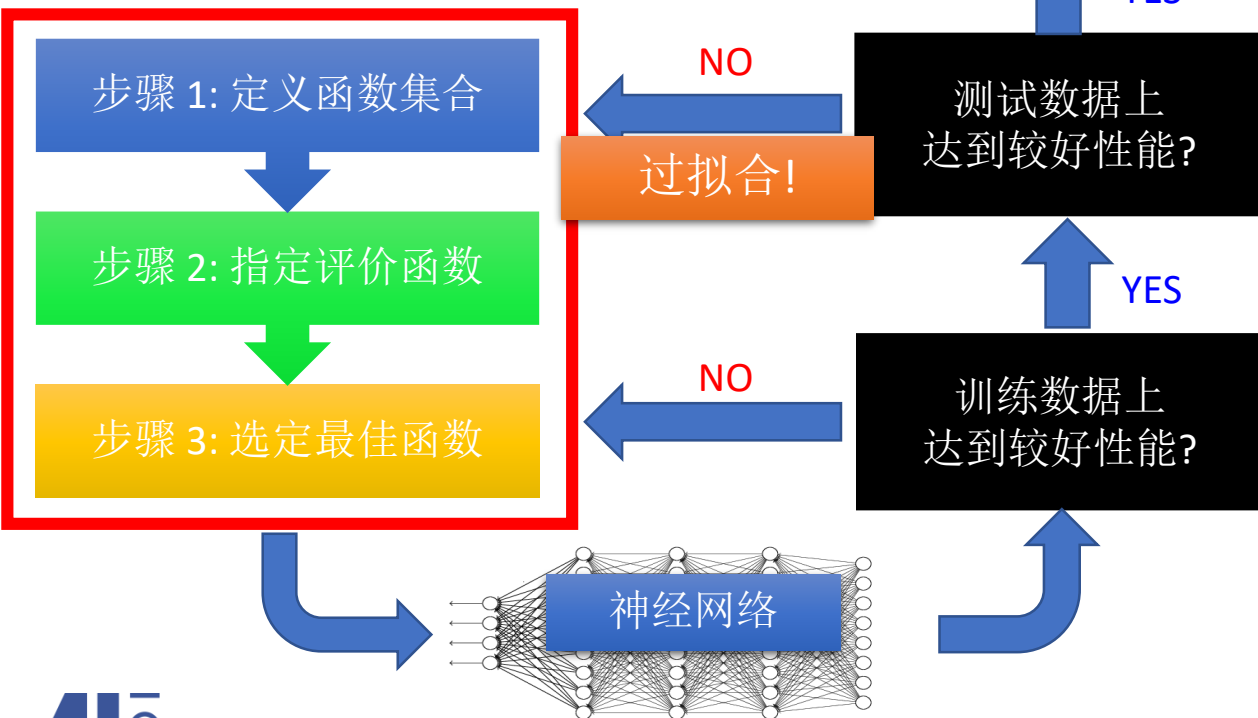


```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Do not always blame Overfitting



- 深度学习简介
- 非凸优化问题
- 深度学习训练技巧
- 卷积神经网络



选择合适的损失函数

Mini-batch训练

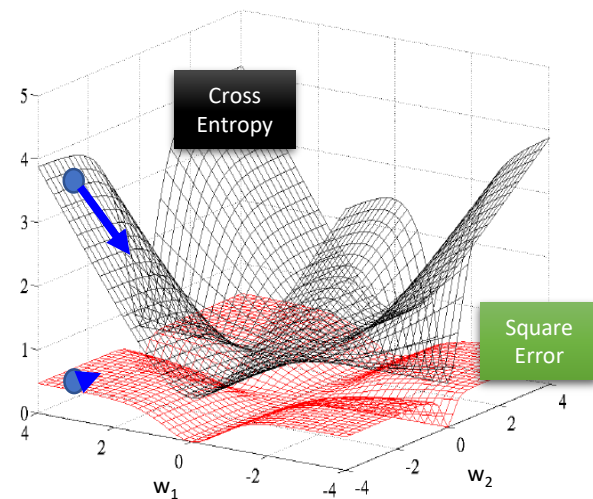
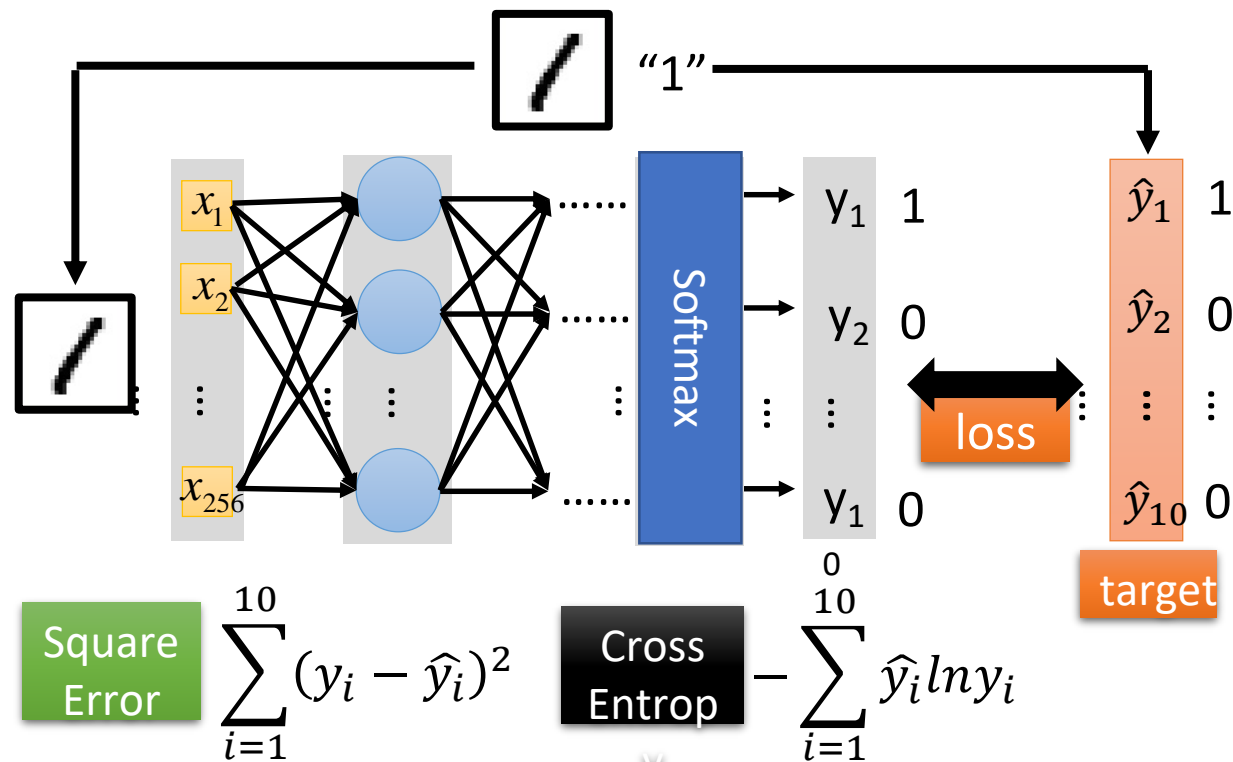
新的激活函数

自适应学习率

含动量的梯度下降

训练技巧

选择合适的损失函数

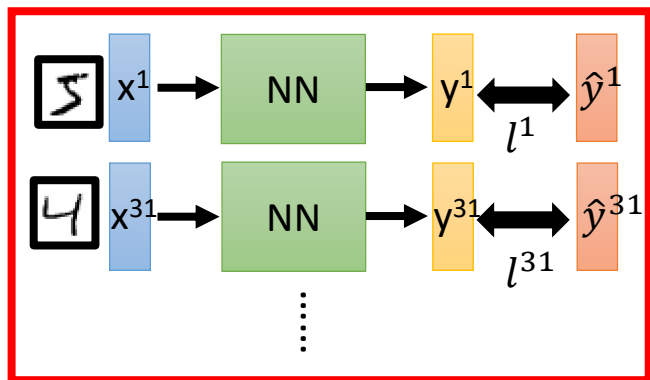


采用Softmax输出层时，
选择交叉熵更合适

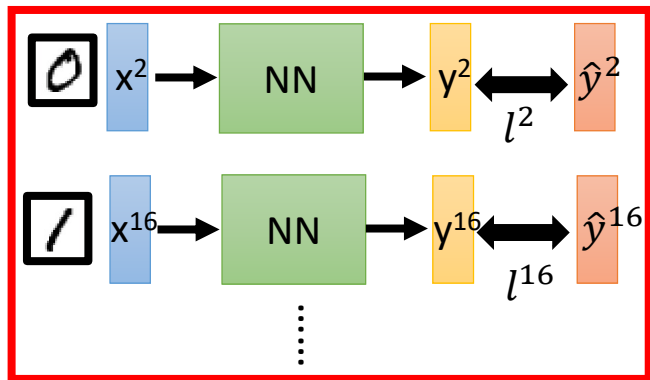
平方误差 vs 交叉熵

Mini-batch (小批量) 训练

Mini-batch



Mini-batch



➤ 随机初始化网络参数

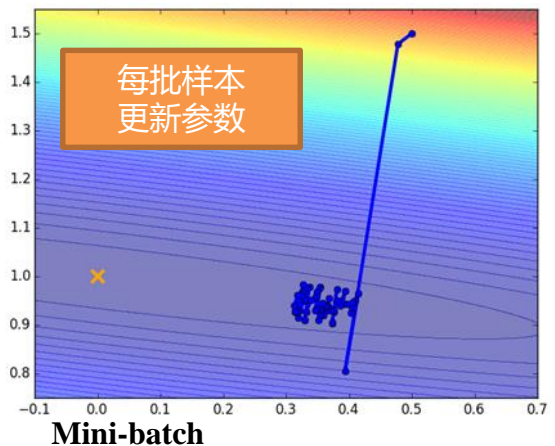
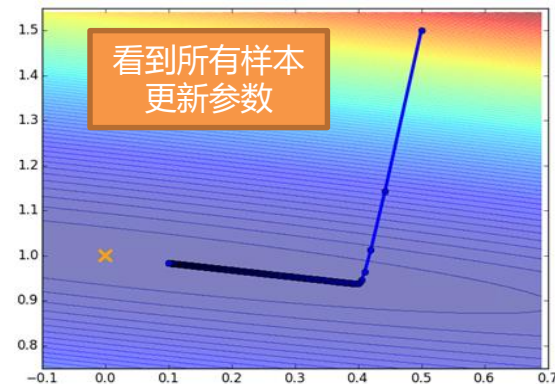
➤ 选择第一批样本更新参数

$$L' = l^1 + l^{31} + \dots$$

➤ 选定第二批样本更新参数

$$L'' = l^2 + l^{16} + \dots$$

➤ 重复参数更新，直到所有mini-batch被选定



Mini-batch

难以发挥深度的优势

选择合适的损失函数

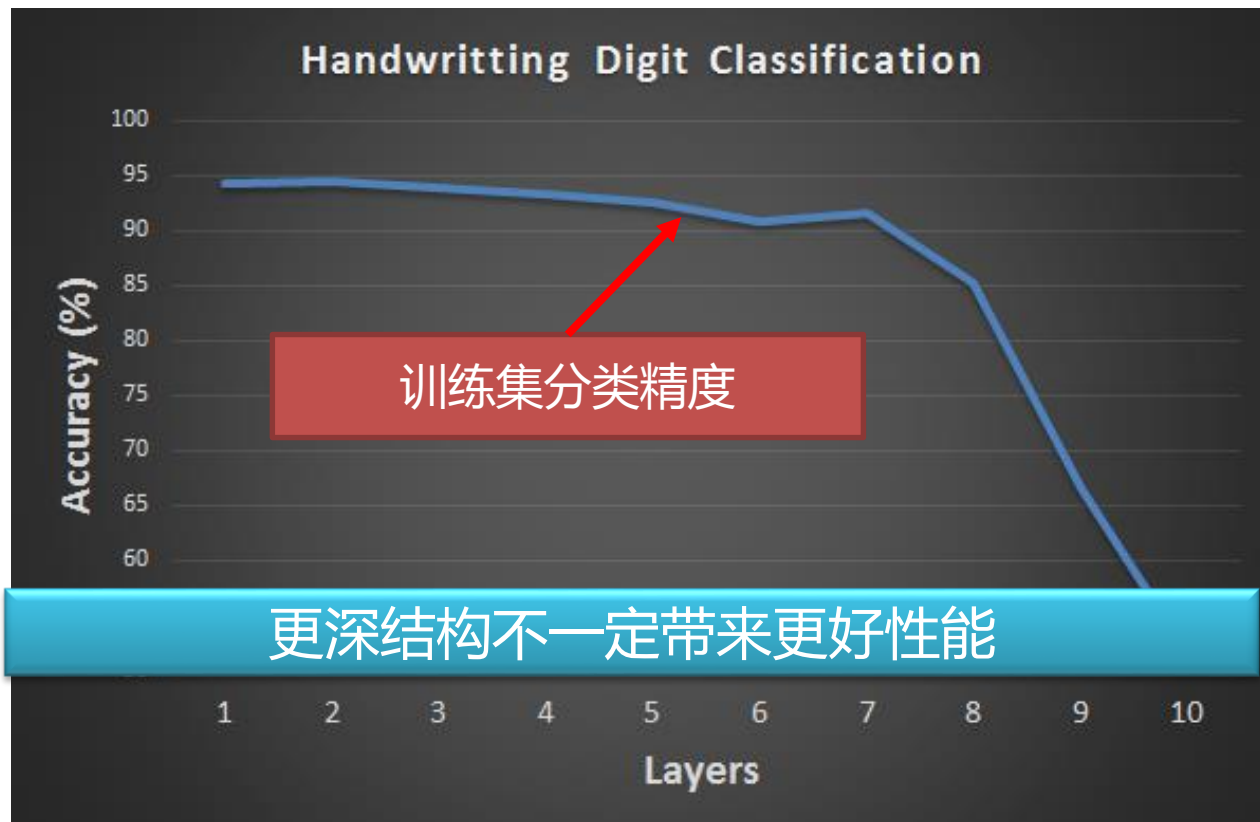
Mini-batch训练

新的激活函数

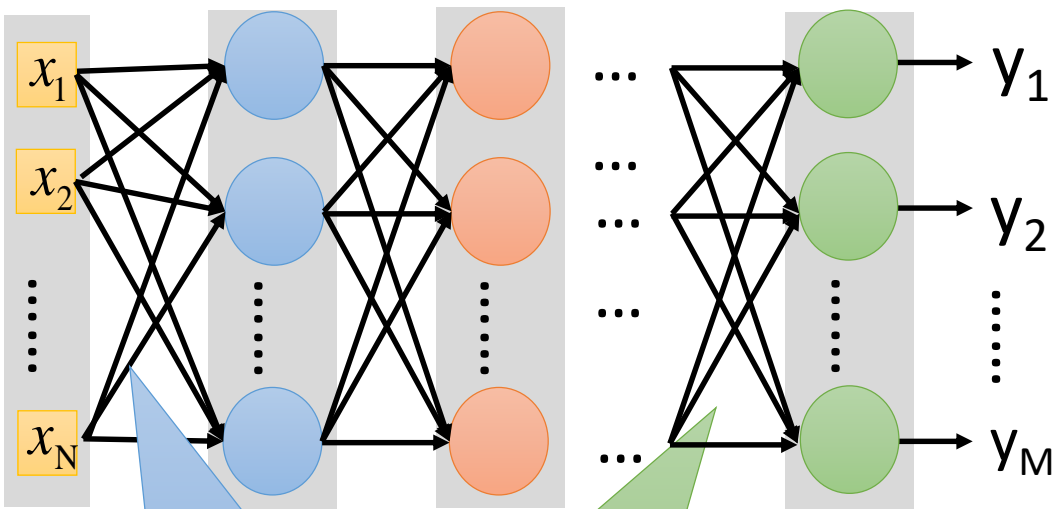
自适应学习率

含动量的梯度下降

训练技巧



▶ 梯度消失问题

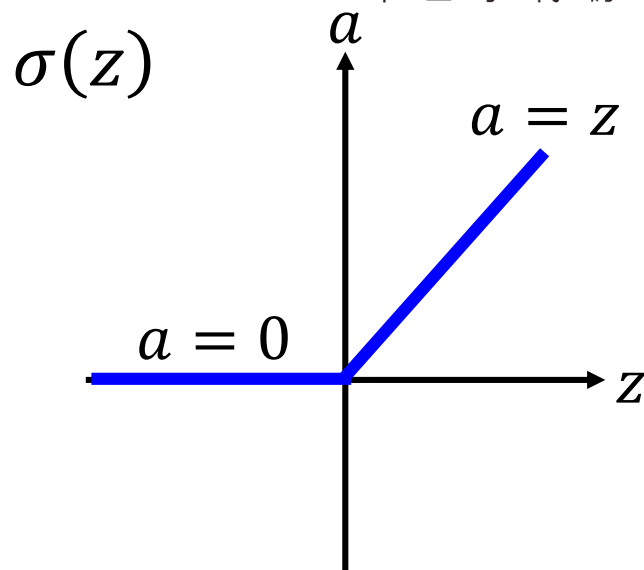


梯度小学习慢

接近初始随机

梯度大学习快

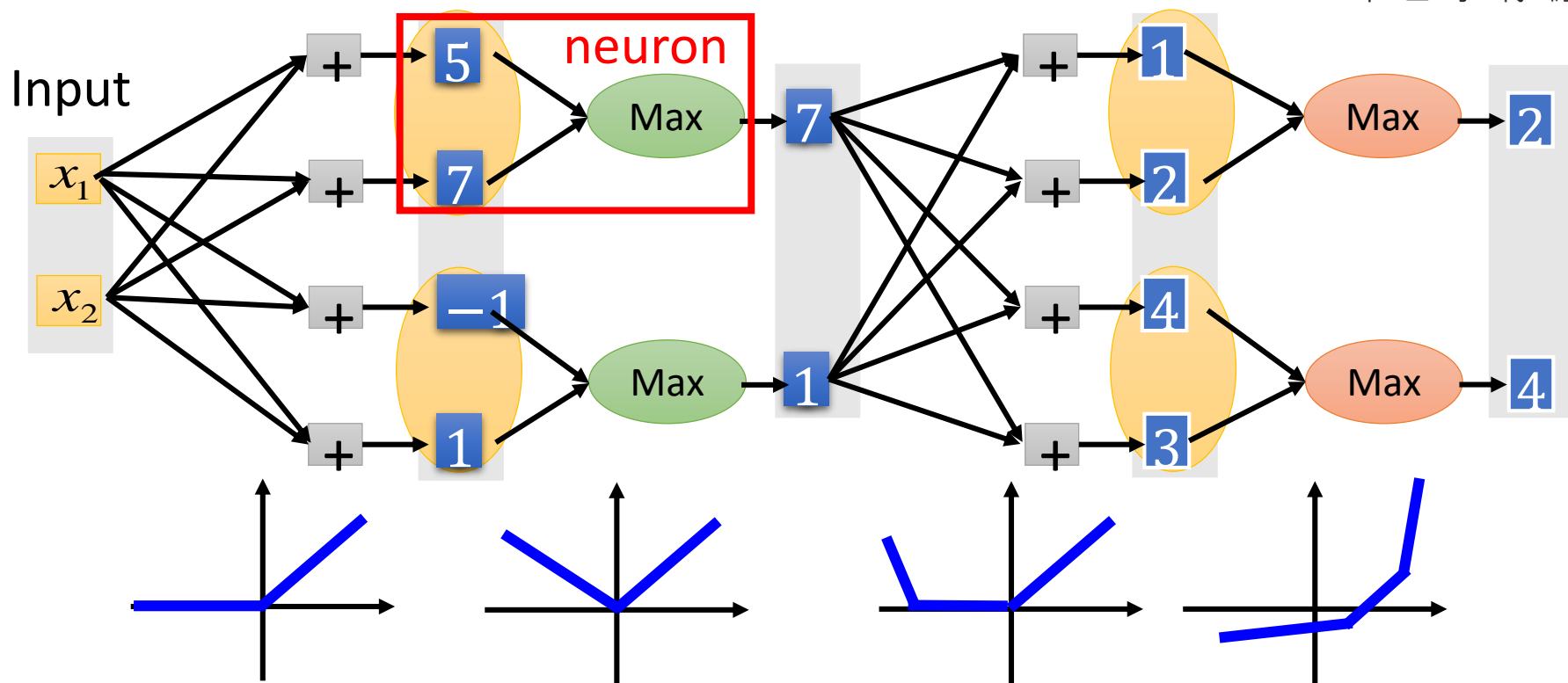
接近收敛



ReLU激活函数：

1. 计算快速
2. 生物启发
3. 克服梯度消失问题

Maxout激活函数



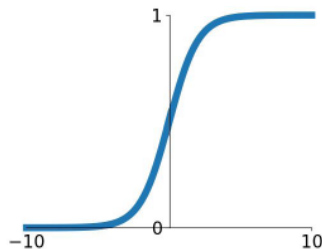
ReLU是Maxout的一个特例

优缺点：可学习的激活函数（+），激活参数加倍（-）

常用激活函数

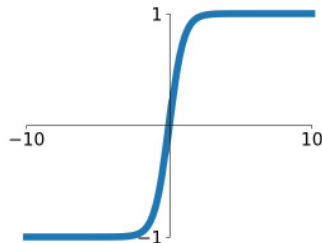
Sigmoid

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



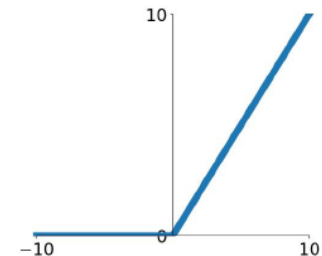
tanh

$$\tanh(x)$$



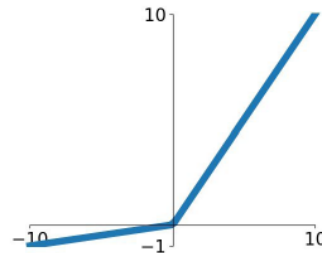
ReLU

$$\max(0, x)$$



Leaky ReLU

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

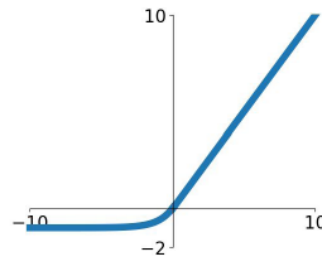


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



自适应学习率

选择合适的损失函数

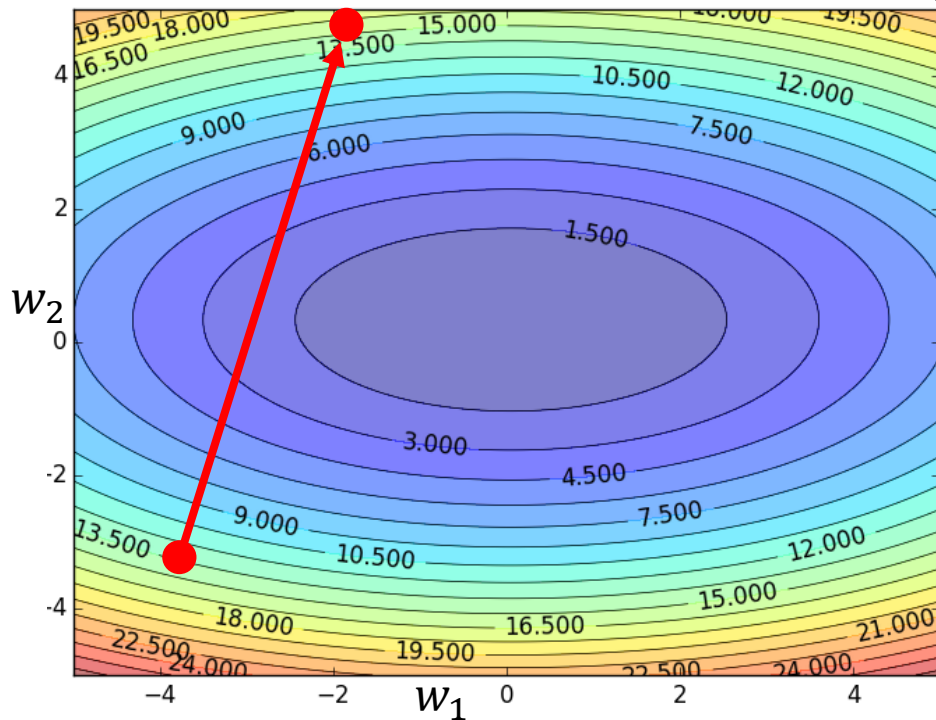
Mini-batch训练

新的激活函数

自适应学习率

含动量的梯度下降

训练技巧



问题：学习率太高，每回合总损失不下降；学习率太低，训练速度太慢

解决方案：训练过程中逐渐降低学习率

$$\eta^t = \eta / \sqrt{t + 1}$$

Adagrad方法 $w \leftarrow w - \eta_w \partial L / \partial w$

$$w_1 \begin{array}{|c|} \hline g^0 \\ \hline 0.1 \\ \hline \end{array}$$

学习率:

$$\frac{\eta}{\sqrt{0.1^2}}$$

$$= \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}}$$

$$= \frac{\eta}{0.22}$$

$$w_2 \begin{array}{|c|} \hline g^0 \\ \hline 20.0 \\ \hline \end{array}$$

学习率:

$$\frac{\eta}{\sqrt{20^2}}$$

$$= \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}}$$

$$= \frac{\eta}{22}$$

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^t (g^i)^2}}$$

$g^i = \partial L / \partial w$, 第*i*次更新时获得;
分母为历次梯度的平方和

特性:

1. 所有参数的学习率越来越小
2. 导数越小, 梯度越大

动量梯度下降

选择合适的损失函数

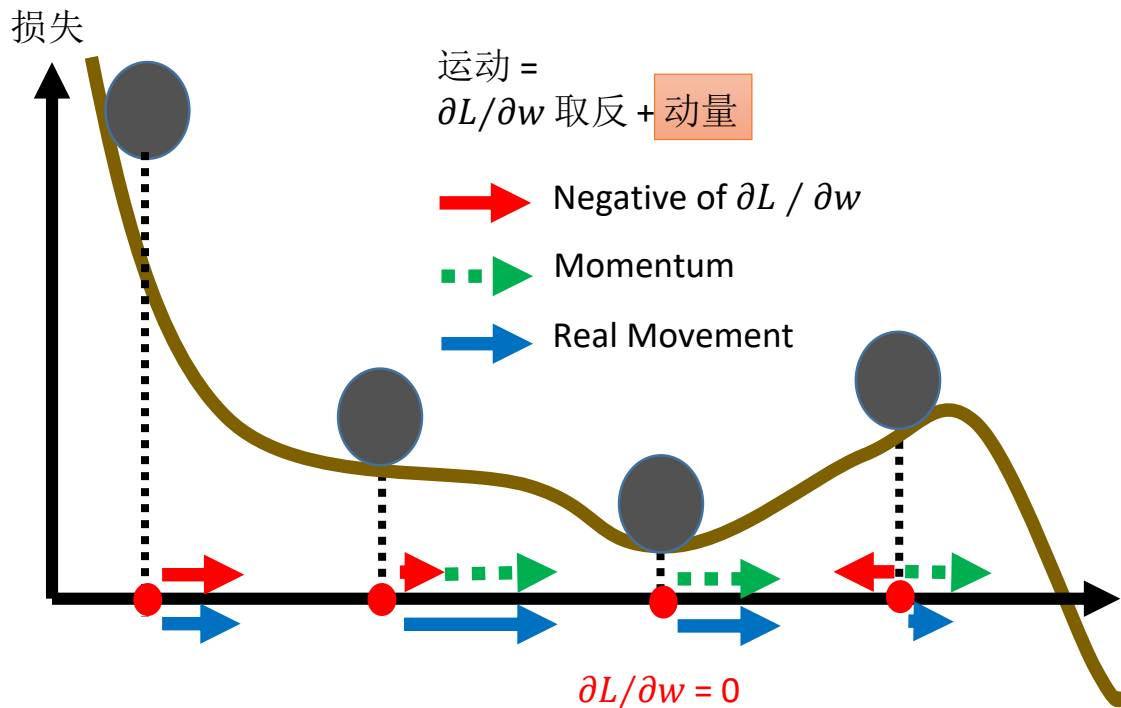
Mini-batch训练

新的激活函数

自适应学习率

含动量的梯度下降

训练技巧



Adam = RMSProp (Advanced Adagrad) + Momentum

其他训练技巧

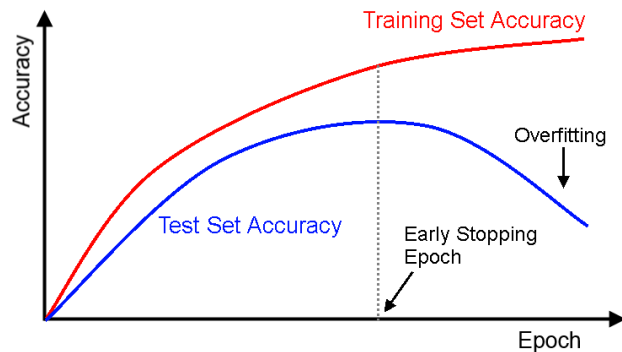
Early Stopping

正则化

Dropout

网络结构

样本扩增



训练技巧

Dropout

Early Stopping

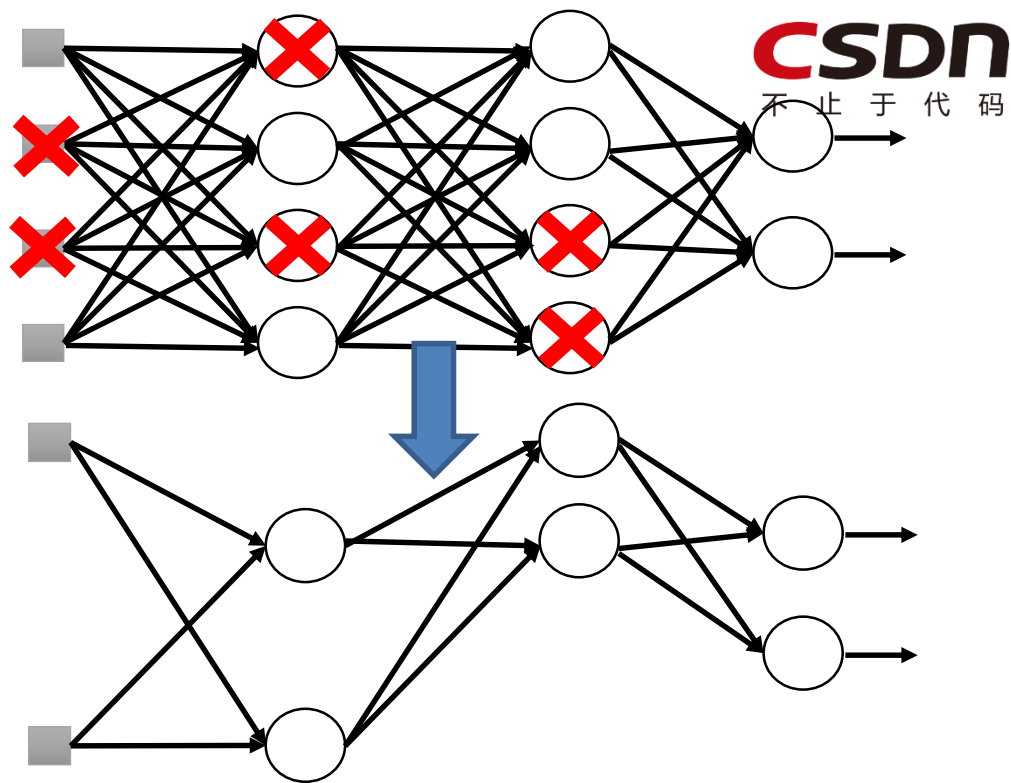
正则化

Dropout

网络结构

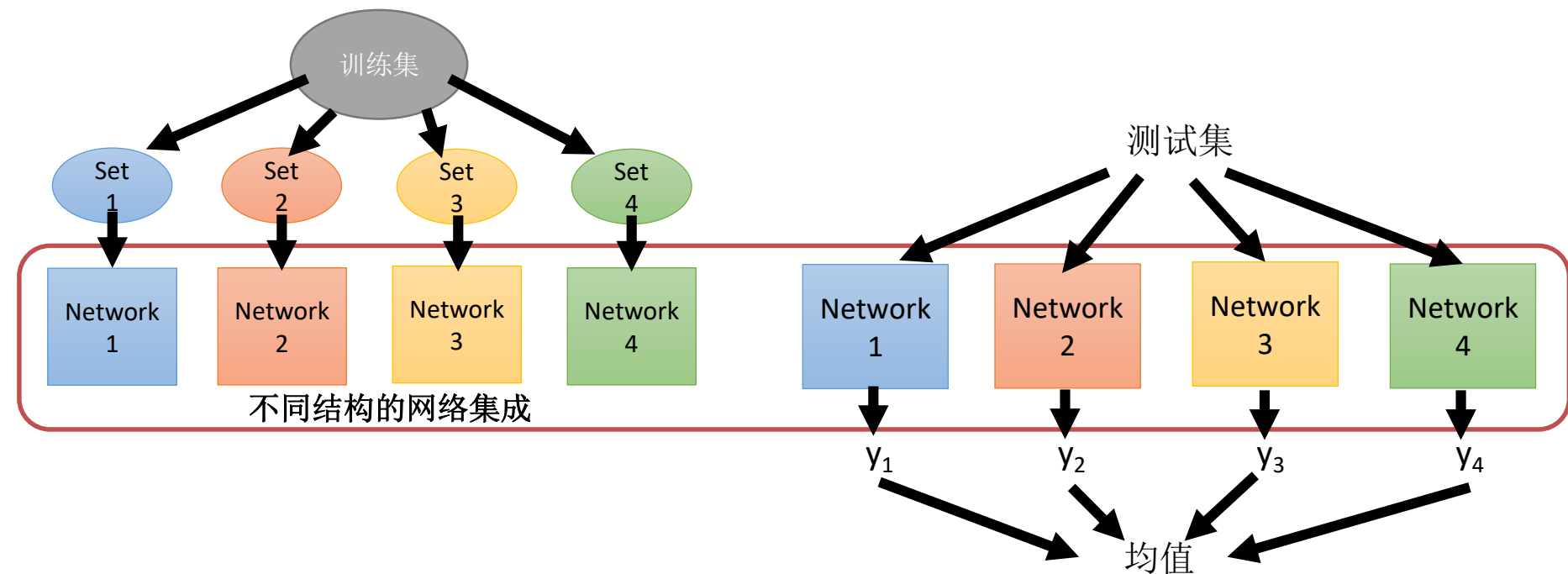
样本扩增

训练技巧

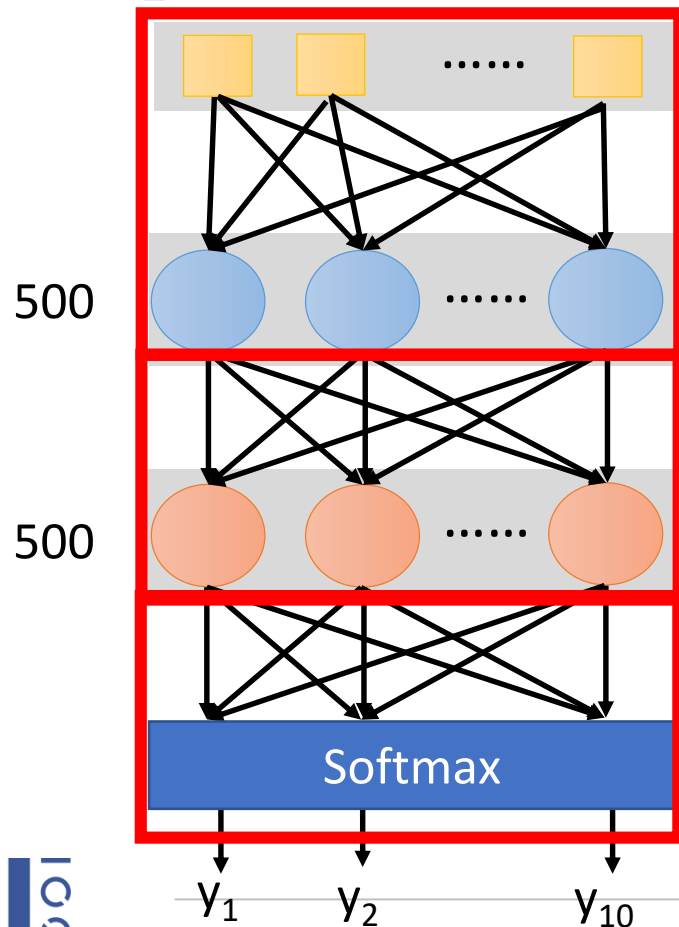


每次参数更新前，每个神经元有 $p\%$ dropout, 网络结构发生变化，使用新网络训练。每个mini-batch重采样dropout神经元

- Dropout作为集成学习方法



Dropout演示



```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,  
                  output_dim=500 ) )  
model.add( Activation('sigmoid') )
```

```
model.add( dropout(0.8) )
```

```
model.add( Dense( output_dim=500 ) )  
model.add( Activation('sigmoid') )
```

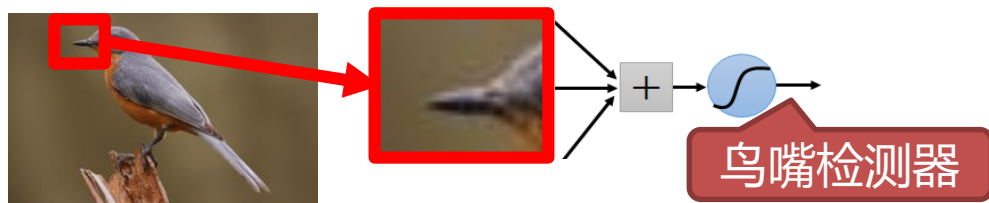
```
model.add( dropout(0.8) )
```

```
model.add( Dense( output_dim=10 ) )  
model.add( Activation('softmax') )
```

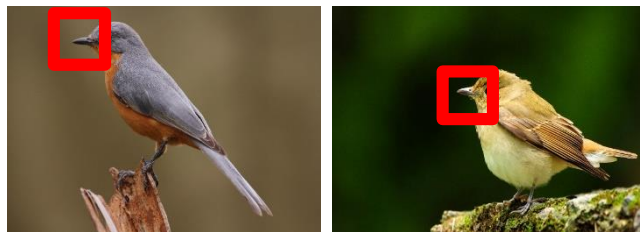
- 深度学习简介
- 非凸优化问题
- 深度学习训练技巧
- 卷积神经网络

卷积神经网络 (CNN) 用于图像分析

- 某些模式比全图小很多



- 相同模式出现在不同区域

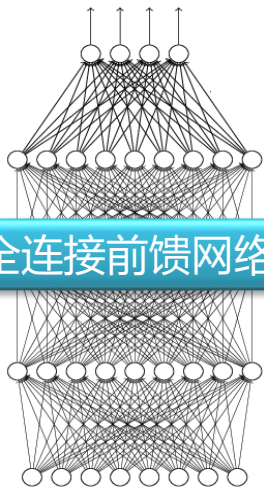


- 图像下采样不改变物体

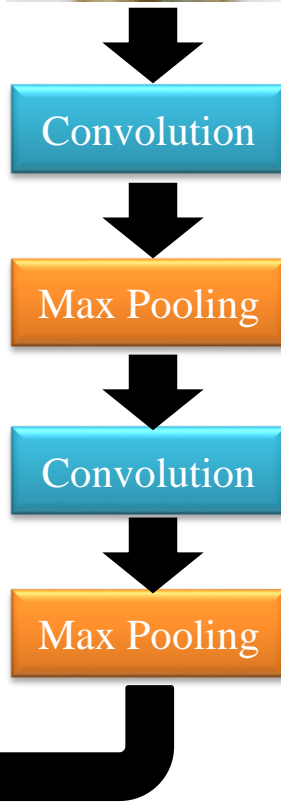
完整的CNN结构



cat dog



重复多次



性质 1

- 某些模式比全图小很多

性质 2

- 相同模式出现在不同区域

性质 3

- 子采样不改变物体

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 图像

1	-1	-1
-1	1	-1
-1	-1	1

滤波器 1

参数矩阵

-1	1	-1
-1	1	-1
-1	1	-1

滤波器 2

参数矩阵

⋮ ⋮

滤波器矩阵为待学习的
网络参数

特性 1

每个滤波器检测一个3 x 3大小的模式.

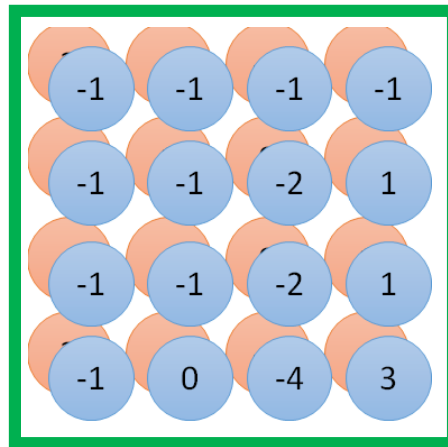
卷积神经网络 VS 全连接前馈网络

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

图像

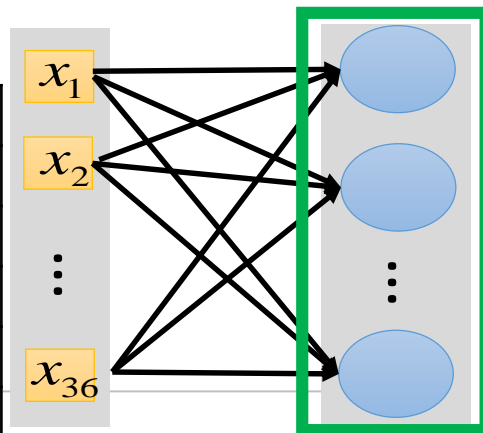
1	-1	-1
-1	1	-1
-1	-1	1

-1	1	-1
-1	1	-1
-1	1	-1



全连接

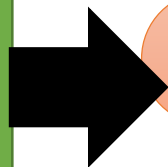
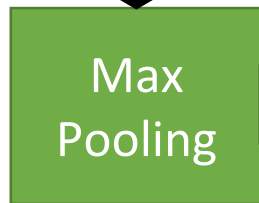
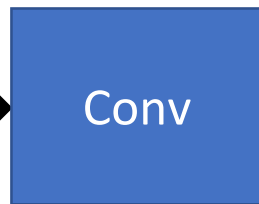
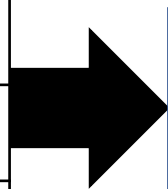
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0



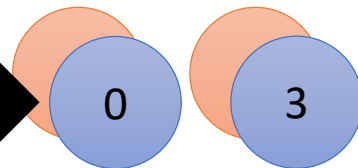
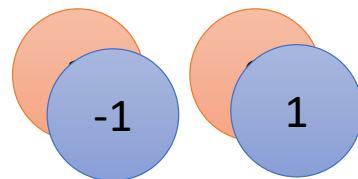
► CNN的Max Pooling

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 图像



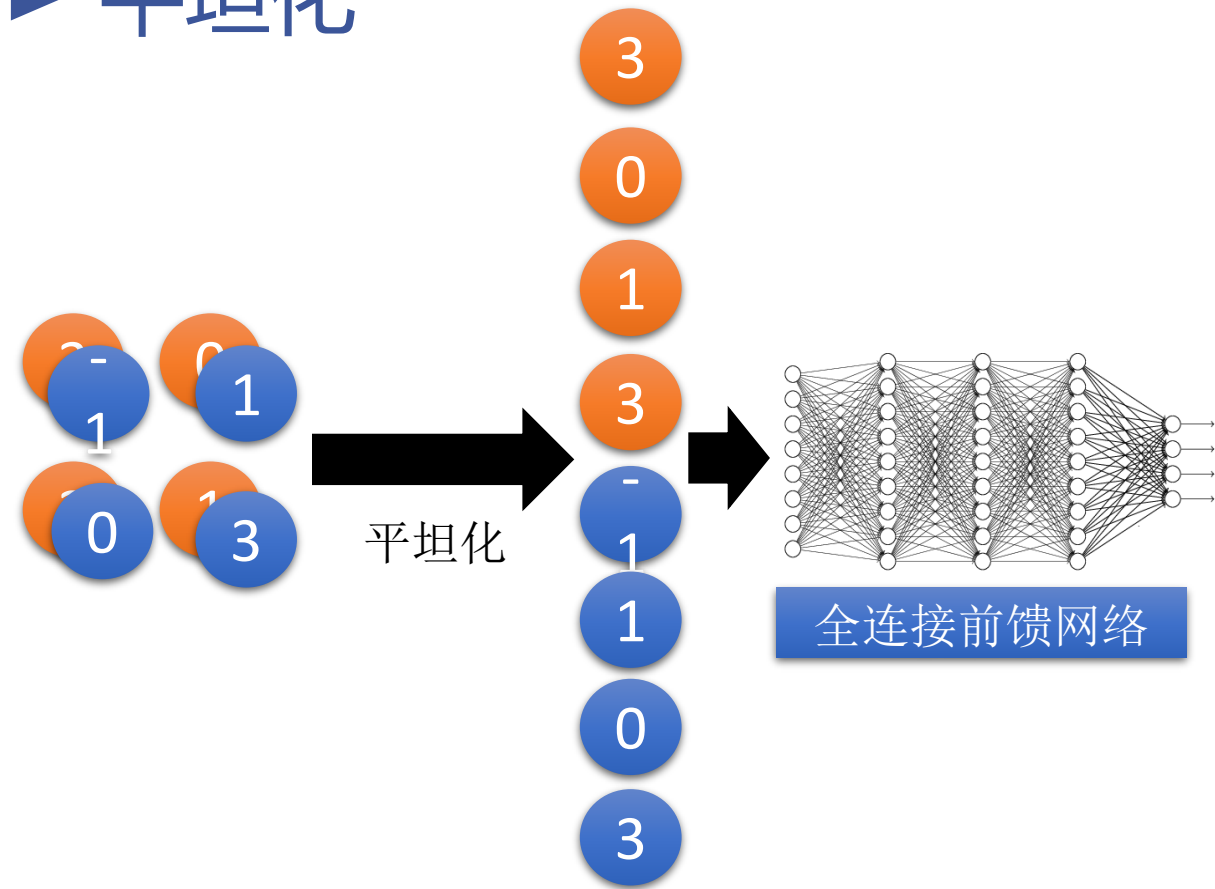
新图像尺寸变小



2 x 2 图像

每个滤波器对应
一个通道

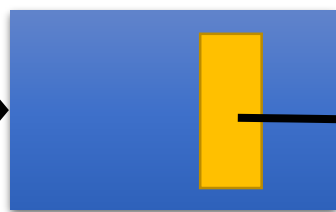
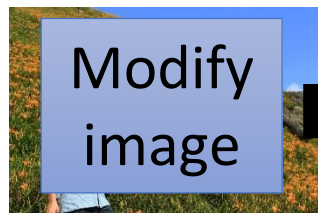
▶ 平坦化



CNN中的学习算法:

- 与其他神经网络学习算法相同，仍然使用梯度下降算法
- 由于卷积计算较高效，能够使用GPU加速计算

CNN热门应用——Deep Dream

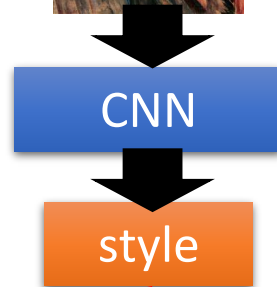
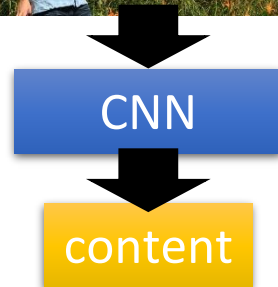
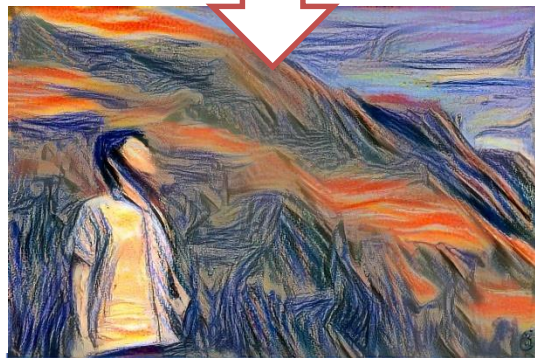
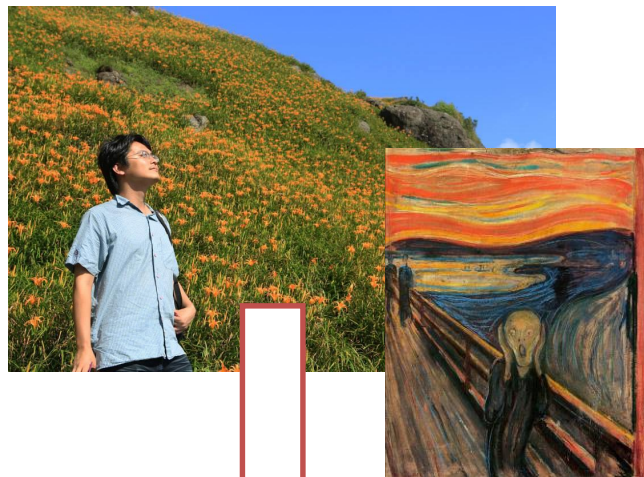


$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$



算法在原始图像上增加CNN
检测结果

CNN热门应用——Deep Style



需求：给定一张照片，使得
它的网格与著名绘画相似
实现：两套CNN网络分别
提取内容与网格的统计信息



更多应用：围棋对战

黑: 1
白: -1
无子: 0



19 x 19 矩阵
(图像)

神经网络

下一步棋
(19 x 19 个位置)

历史棋谱

CNN

Target:
“天元” = 1
else = 0

CNN

Target:
“五之 5” = 1
else = 0

THANK YOU



AI100