

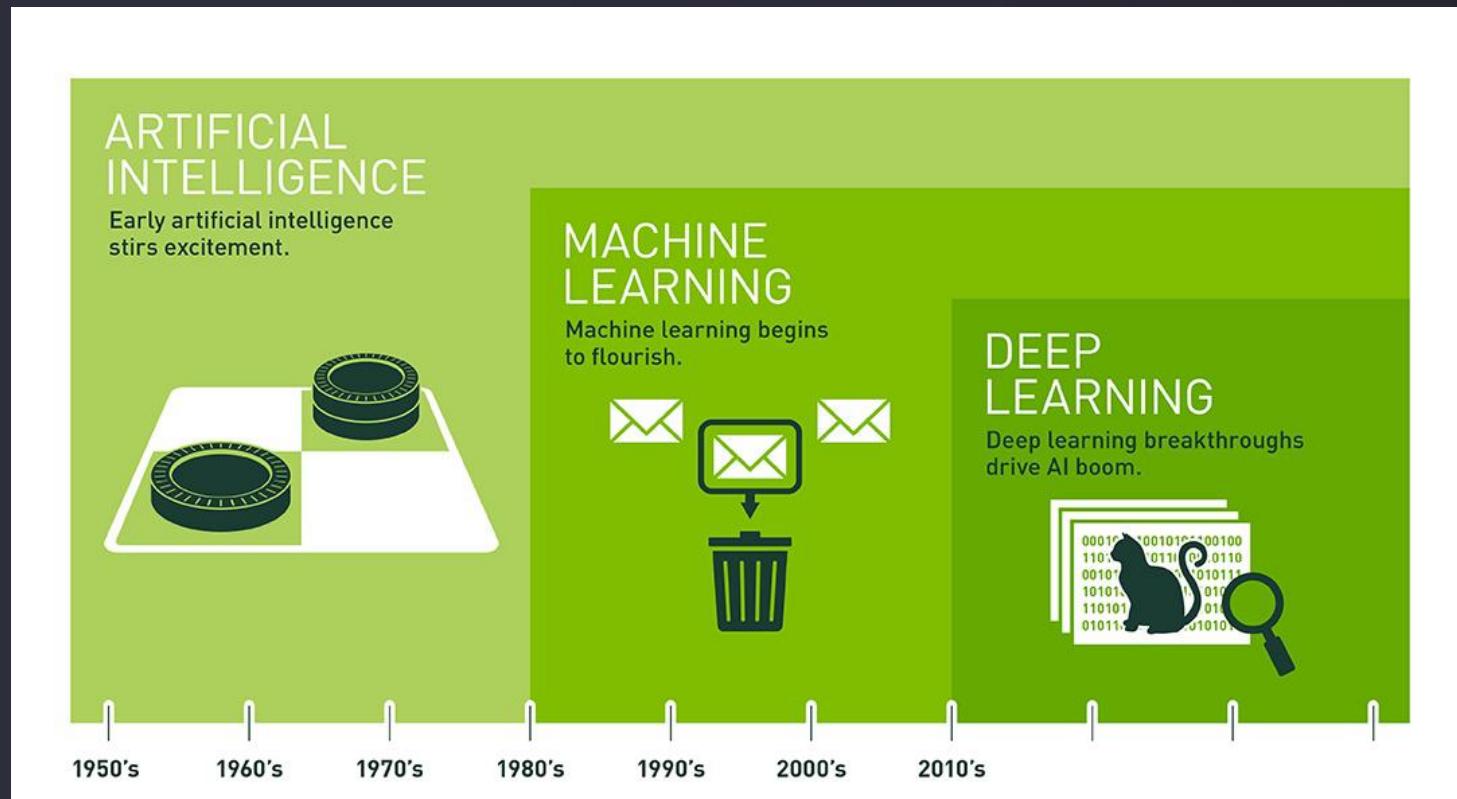


Python3人工智能入门+实战提升：深度学习

Chapter 1 课程介绍及环境配置

赵辛

| Python3人工智能入门+实战提升：深度学习



人工智能

机器学习

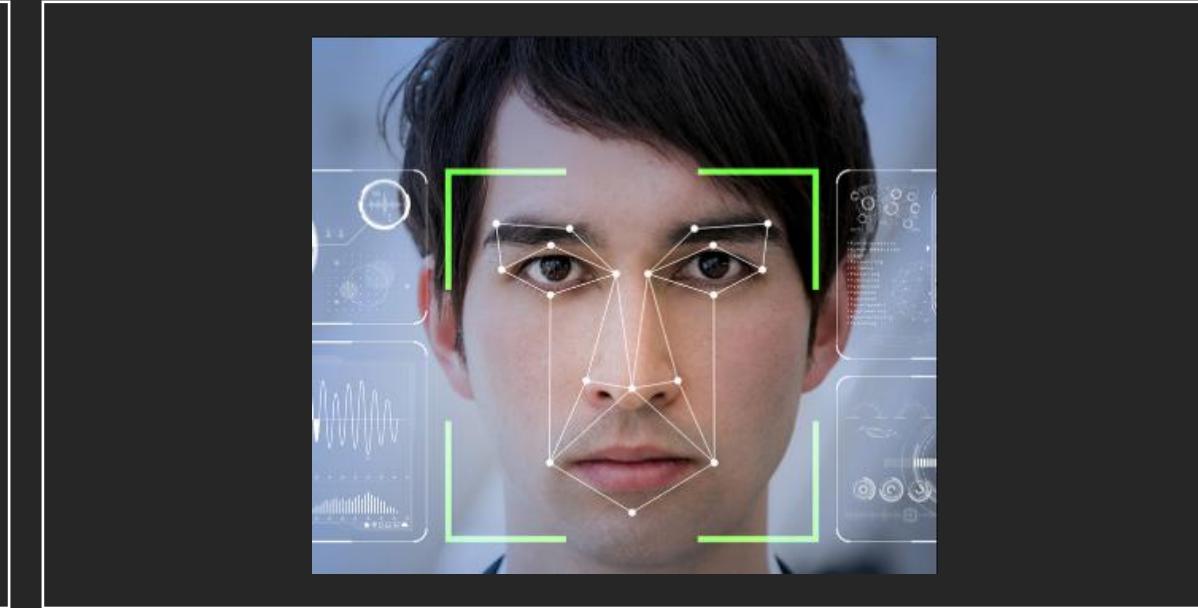
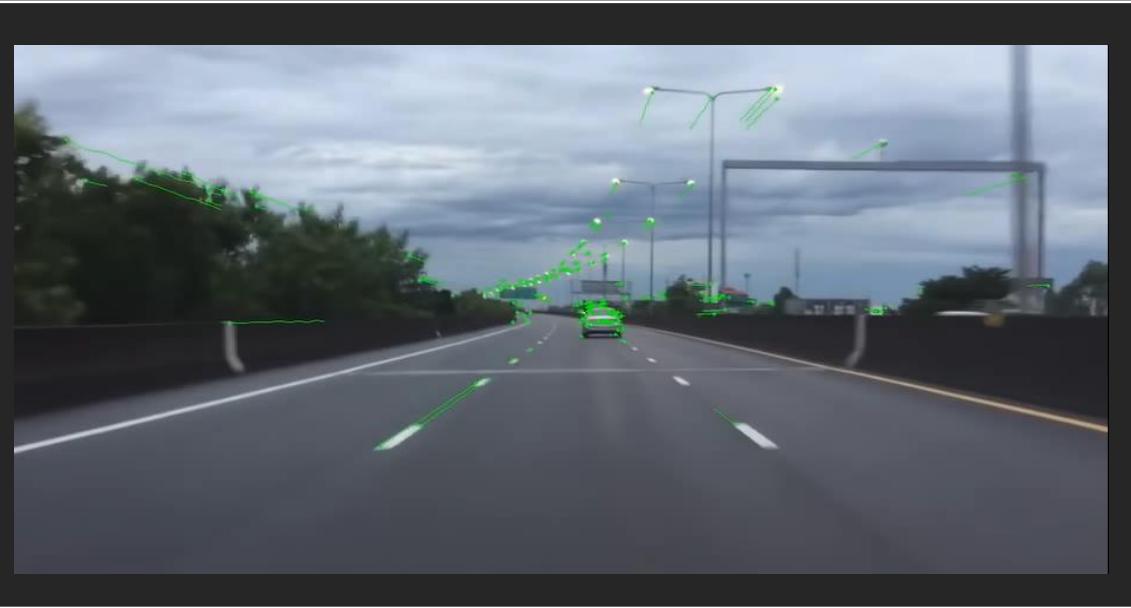
回归分析之房价预测

Kmeans之图像分割

高斯分布之概率检测

朴素贝叶斯之奖学金预测

深度学习



本门课程的特点

容易懂、趣味案例、快速应用！



1、以AI知识与实战为主、降低学习难度
课程安排：现实案例+知识干货+实战

2、生活案例出发、覆盖主流技术，侧重
对不同技术的应用理解

3、现实案例驱动知识讲解、手把手带你
编程，帮助加深知识点理解

4、实战案例与现实场景相结合，帮助加
强解决实际问题的能力

讲师介绍



人工智能算法科学家

- 2019年福布斯精英科技榜U30
- 深圳市海外高层次人才
- 澳大利亚新南威尔士大学全奖博士
- 国际SCI收录学术文章十篇
- CSDN人工智能精英讲师（机器学习、深度学习）

Chapter 1 课程介绍及环境配置

-
- 1 --课程目标
 - 2 --课程内容概览
 - 3 --人工智能核心概念
 - 4 -- AI开发工具及实战基础

课程目标

掌握核心理论

MLP
CNN
RNN
...

熟练AI工具

Python
Jupyter
Keras
...

解决实际问题

数据预处理
模型训练与预测
模型评估
...

提升综合能力

特征提取
模型迁移
半监督学习
...

课程目标

1、掌握核心知识点



课程目标

2、熟练使用AI工具



课程目标

3、掌握利用AI解决实际场景问题的能力

全面、
规范的
实践训
练流程

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

部分
案例

非线性数据分类

Fashion_mnist服饰预测

CNN猫狗识别

贵州茅台股价预测

RNN语句生成

少样本特别草莓识别

课程目标

4、提AI综合能力

数据增强

结果矫正

机器学习
+
深度学习

经典模型
特征提取

监督
+
无监督学习

数据分离

PCA数据
降维

模型评估

结果可视化

| Python3人工智能入门+实战提升：深度学习

下一个改革是人工智能！本课程是一门为AI新手量身定做的入门课

简单易懂的AI课程！不同行业的学员均适合，让你具备用AI解决问题的技能！

学习资料

(资料) Python3 掌握人工智能：入门与综合

介绍

本文档为围绕五门核心课程(《零基础学Python》、《Python3人工智能入门+实战提升：机器学习》、《Python3人工智能入门+实战提升：深度学习》、《资深人工智能面试官教你拿下AI工作OFFER的5大秘决》、《深度学习与计算机视觉》)，整理的重要资料入口，不定期更新，如果无法访问，可加小助手微信反馈问题(微信号：ai_flare)。

备注：

- 1、一次给你一个压缩包，大概率会成垃圾。老师会整理出重要资料，附带简介，不定期更新，希望能帮到每位学习的同学
- 2、如有好的学习资料，欢迎添加小助手微信投稿

行业发展与就业情况

人工智能产业人才发展报告（2020工信部最新）

简介：国家工信部的AI行业发展报告，包含行业发展概览、现状、就业情况（薪资、能力要求）、趋势等，帮你明确行业格局，先人一步。

链接：<https://pan.baidu.com/s/1QnDrdLu5nTGz0DoEzY7nQ>

目录

- 介绍
- 行业发展与就业情况
 - 人工智能产业人才发展报告（2020工信部最新）
- 工程开发
 - 环境配置及软件安装常见问题汇总
 - 常用pip安装源：https://blog.csdn.net/dfly_zx/article/details/108060652
- 知识学习
 - 课程>>Python3人工智能入门+实战提升：机器学习
 - 模型评估、选择与优化
 - 具体算法
 - 行业应用
- 常用链接

亲自整理与使用的学习资料

内容不定期更新，包含：

- a) 行业状况
- b) 知识干货
- c) 实战资料

|学习交流

立刻扫码
入群学习





Python3人工智能入门+实战提升：深度学习

Chapter 1 课程介绍及环境配置

赵辛

Chapter 1 课程介绍及环境配置

-
- 1 --课程目标
 - 2 --课程内容概述
 - 3 --人工智能核心概念
 - 4 -- AI开发工具及实战基础

课程概述

1、课程介绍及环境配置

课程目标

课程内容概览的定义

核心工具介绍

手把手实战



课程概述

2、MLP多层感知器

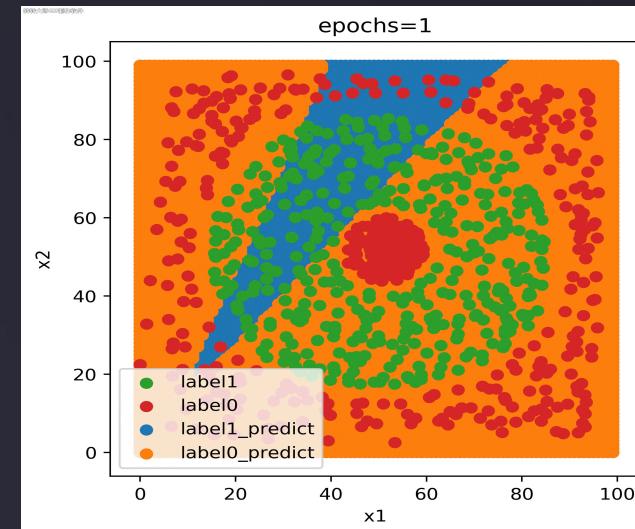
逻辑回归与多层感知器

非线性分类与多分类

激活函数

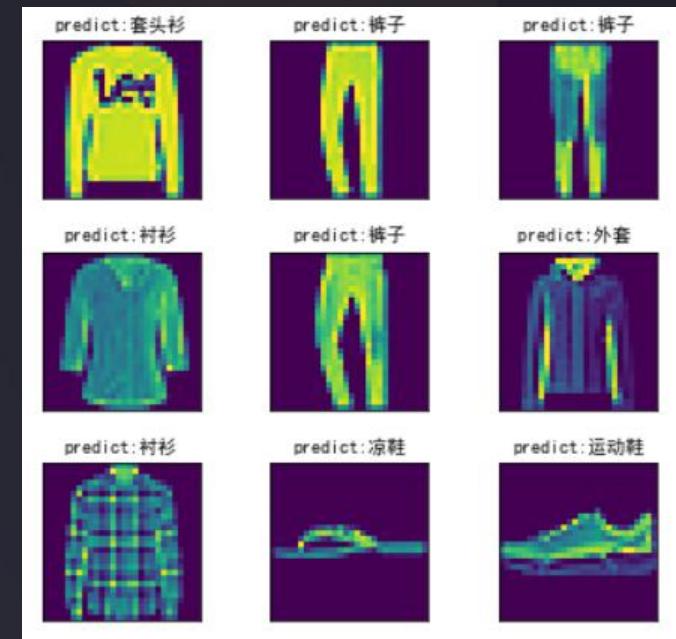
损失函数与反向传播算法

手把手实战



MLP非线性边界数据分类

Fashion_mnist服饰分类



课程概述

3、CNN卷积神经网络

普通MLP局限性

图像卷积

卷积神经网络

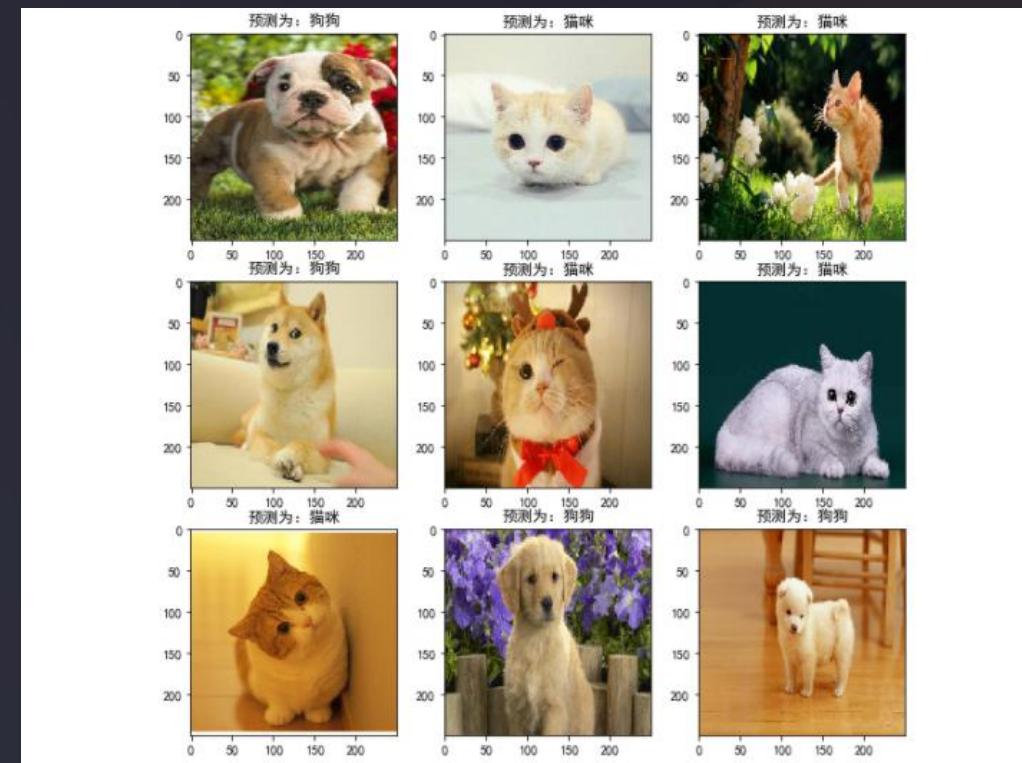
池化与填充

经典CNN模型

手把手实战

① 搭建全新CNN模型完成猫狗识别

② VGG16+MLP快速准确预测



课程概述

4、RNN循环神经网络

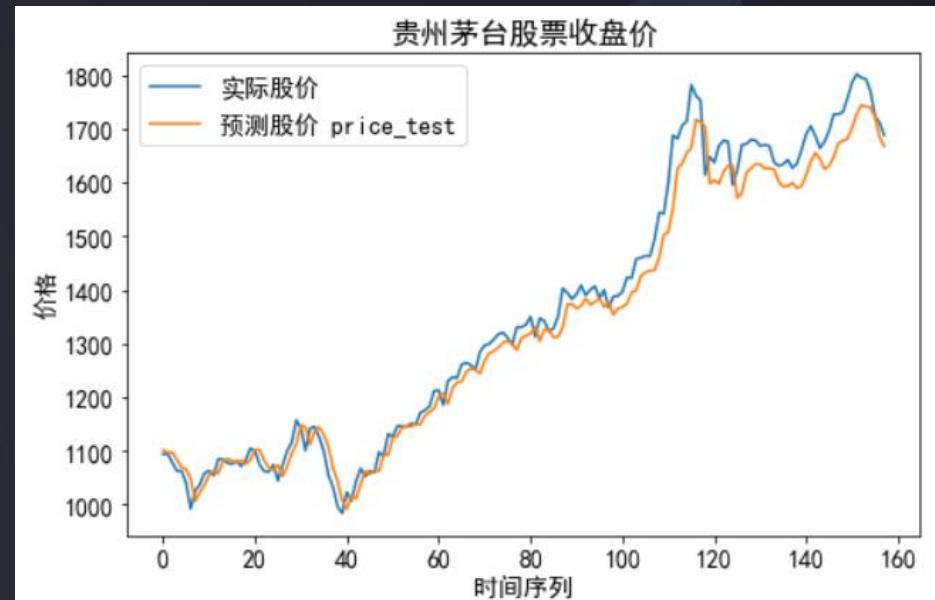
序列数据预测

循环神经网络

多样的RNN网络

LSTM、BRNN、DRNN

手把手实战



RNN预测贵州
茅台股价

Artificial intelligence (AI), --predict next letter is--- s
rtificial intelligence (AI), s --predict next letter is--- o
tificial intelligence (AI), so --predict next letter is--- m
ificial intelligence (AI), som --predict next letter is--- e
ficial intelligence (AI), some --predict next letter is--- t
icial intelligence (AI), somet --predict next letter is--- i
cial intelligence (AI), someti --predict next letter is--- m
ial intelligence (AI), sometim --predict next letter is--- e
al intelligence (AI), sometime --predict next letter is--- s
l intelligence (AI), sometimes --predict next letter is---
intelligence (AI), sometimes --predict next letter is--- c
intelligence (AI), sometimes c --predict next letter is--- a
ntelligence (AI), sometimes ca --predict next letter is--- l
telligence (AI), sometimes cal --predict next letter is--- l
elligence (AI), sometimes call --predict next letter is--- e
lligence (AI), sometimes calle --predict next letter is--- d

LSTM文本生成

课程概述

5、迁移混合模型与综合提升

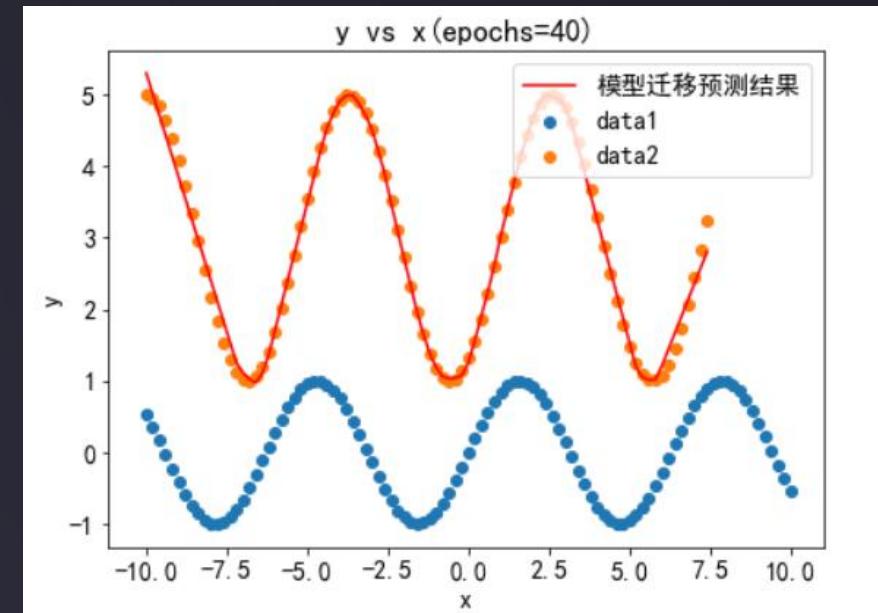
迁移学习

半监督学习

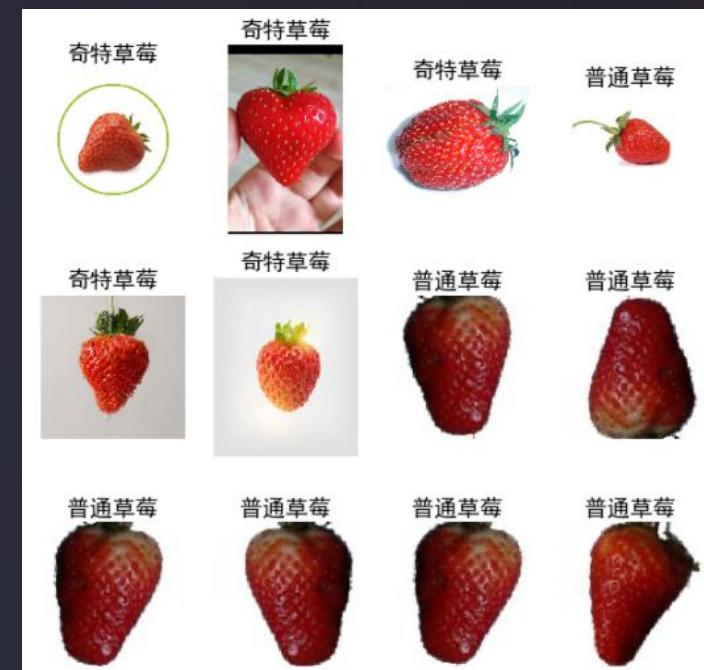
机器+深度学习

综合实战准备

手把手实战



迁移学习高效
回归分析



少样本寻找奇
特草莓



Python3人工智能入门+实战提升：深度学习

Chapter 1 课程介绍及环境配置

赵辛

Chapter 1 课程介绍及环境配置

-
- 1 --课程目标
 - 2 --课程内容概览
 - 3 --人工智能核心概念
 - 4 -- AI开发工具及实战基础

| 核心开发环境

核心开发
环境



| Python

Python是一种解释型的、面向对象的、移植性强的高级程序设计语言。

开发者：吉多·范罗苏姆 (Guido van Rossum)

解释型：不需要编译成二进制代码，直接从源代码运行程序

面向对象：Python同时支持面向过程和面向对象编程。

可移植性：Python可以跨操作平台无差别的运行

高层语言：无须考虑诸如如何管理程序使用的内存一类的底层细节

www.python.org/



Python



1. 简单易用，学习成本低

- 开发效率高
- 高级语言，功能强大
- 解释型语言，能跨平台
- 可扩展性强
- 可嵌入性

● 运行速度慢
代码加密困难

缺陷

Rank	Language	Type	Score
1	Python	🌐💻⚙️	100.0
2	Java	🌐📱💻	96.3
3	C	📱💻⚙️	94.4
4	C++	📱💻⚙️	87.5
5	R	📱	81.5
6	JavaScript	🌐	79.4
7	C#	🌐📱💻⚙️	74.5
8	Matlab	📱	70.6
9	Swift	📱💻	69.1
10	Go	🌐💻	68.0

IEEE Spectrum (电气和电子工程师协会)
编程语言 Top 10

Anaconda

Anaconda是一个方便的python包管理和环境管理软件

- 支持 Linux, Mac, Windows
- 可以很方便地实现多版本python并存、切换以及各种第三方包的快速安装



特点:

- 使用方便、安装过程简单
- 兼容不同系统、可同时实现包管理、环境管理的功能

www.anaconda.com/

Jupyter notebook

Jupyter Notebook (此前被称为 IPython notebook) 是一个开源的 Web 应用程序，允许开发者方便的创建、共享和执行代码。

- 可以实时写代码、运行代码、查看结果，并可视化数据

特点：

- 极其适合数据分析（分块执行、方便调试）
- 远程运行
- 交互式展现

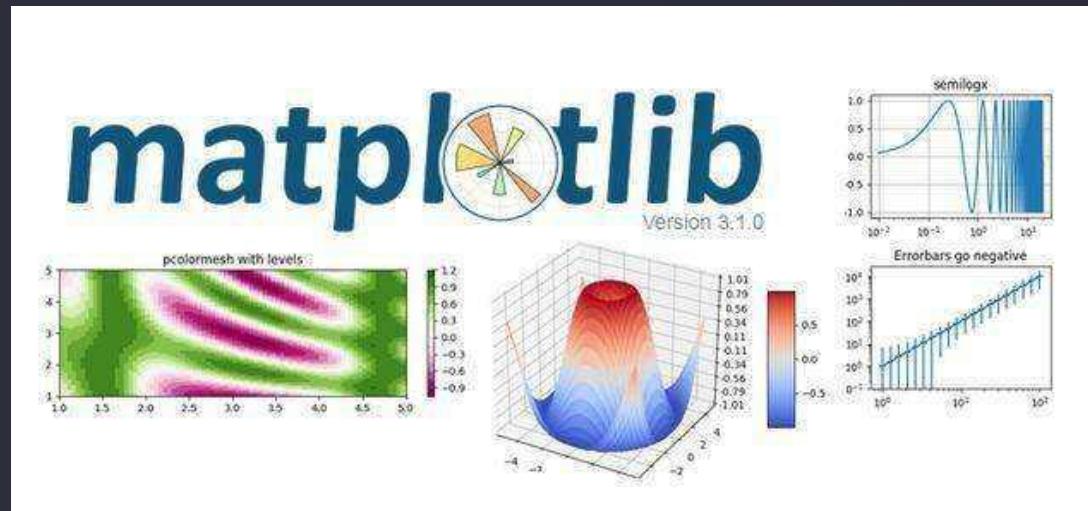
<https://jupyter.org/>



|基础工具包



Pandas提供了大量能使我们快速便捷地处理数据的函数和方法，可用于快速实现数据导入/出、索引。
www.pypandas.cn/



通过 Matplotlib，开发者可以仅需要几行代码，便可以生成绘图、散点图、曲线/曲面图、直方图等。
www.matplotlib.org.cn/



可用来存储和处理大型矩阵，支持大量的维度数组与矩阵运算，对数组运算提供大量的数学函数库。
www.numpy.org.cn/

|Scikit-learn

针对机器学习应用而开发的算法库

编程语言：python

常用功能：数据预处理、分类、回归、降维、模型选择等常用的机器学习算法

四大优点：

- 丰富的算法模块；
- 易于安装与使用；
- 样例丰富、教程文档详细

官网：<https://scikit-learn.org/stable/index.html>





Python3人工智能入门+实战提升：深度学习

Chapter 1 课程介绍及环境配置

赵辛

Chapter 1 课程介绍及环境配置

-
- 1 --课程目标
 - 2 --课程内容概览
 - 3 --人工智能核心概念
 - 4 -- AI开发工具及实战基础

|深度学习AI工具



Keras

Keras 是一个用 Python 编写的用于神经网络开发的应用接口，调用开接口可以实现神经网络、卷积神经网络、循环神经网络等常用深度学习算法的开发



特点：

- 集成了深度学习中各类成熟的算法，容易安装和使用，样例丰富，教程和文档也非常详细
- 能够以 TensorFlow, 或者 Theano 作为后端运行

<https://keras.io/zh/>

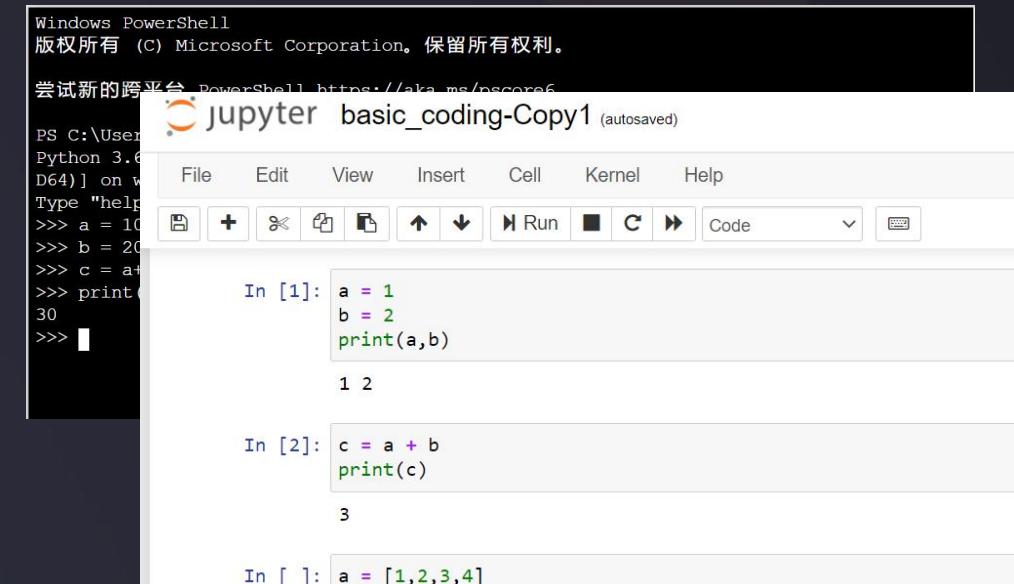
<https://keras.io/>

Keras or Tensorflow

Tensorflow是一个采用数据流图，用于数值计算的开源软件库，可自动计算模型相关的微分导数：非常适合用于神经网络模型的求解。

Keras可看作为tensorflow封装后的一个接口（Keras作为前端，TensorFlow作为后端。

Keras为用户提供了一个易于交互的外壳，方便进行深度学习的快速开发



The image shows a Jupyter Notebook interface and a Windows PowerShell window side-by-side.

Windows PowerShell:

```
Windows PowerShell
版权所有 (C) Microsoft Corporation. 保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Users\Python 3.6
D64] on w
Type "help"
>>> a = 10
>>> b = 20
>>> c = a+
>>> print(c)
30
>>> 
```

Jupyter Notebook:

```
jupyter basic_coding-Copy1 (autosaved)
```

File Edit View Insert Cell Kernel Help

In [1]:
a = 1
b = 2
print(a,b)
1 2

In [2]:
c = a + b
print(c)
3

In []: a = [1,2,3,4]

Keras建立MLP模型

```
#建立一个Sequential顺序模型
from keras.models import Sequential
model = Sequential()
```

```
#通过.add()叠加各层网络
from keras.layers import Dense
model.add(Dense(units=3, activation='sigmoid', input_dim=3))
model.add(Dense(units=1, activation='sigmoid'))
```

```
#通过.compile()配置模型求解过程参数
model.compile(loss='binary_crossentropy', optimizer='sgd' )
```

```
#训练模型
model.fit(x_train, y_train, epochs=10)
```

30s上手Keras: <https://keras-cn.readthedocs.io/en/latest/#30skeras>



|深度学习开发实战基础

1. 下载、安装Anaconda
2. 新建开发环境、指定安装python版本
 - 配置: `conda create -n env_name python=3.7.0`
3. 安装jupyter-notebook
4. numpy、pandas、matplotlib、scikit-learn安装与测试
 - 配置: `pip/conda install package_name`
5. Tensorflow、keras安装与测试

国内镜像源: https://blog.csdn.net/dfly_zx/article/details/108060652



Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

赵辛

Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

逻辑回归实现图像分类

400行

```
[[186. 190. 193. ... 232. 233. 233.]  
[185. 190. 194. ... 230. 233. 233.]  
[187. 190. 192. ... 229. 228. 228.]  
...  
[110. 103. 100. ... 197. 207. 231.]  
[106. 104. 103. ... 204. 226. 241.]  
[142. 136. 123. ... 218. 238. 249.]]
```

500列

$$\begin{bmatrix} x_1 & \cdots & x_{500} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ x_{199501} & \cdots & x_{200000} \end{bmatrix}$$

用于模型的二次项($x_i \times x_j$)数量：超过百亿个！(10^{10})

如果不增加特征项，是否有其他办法？

逻辑回归实现非线性边界二分类

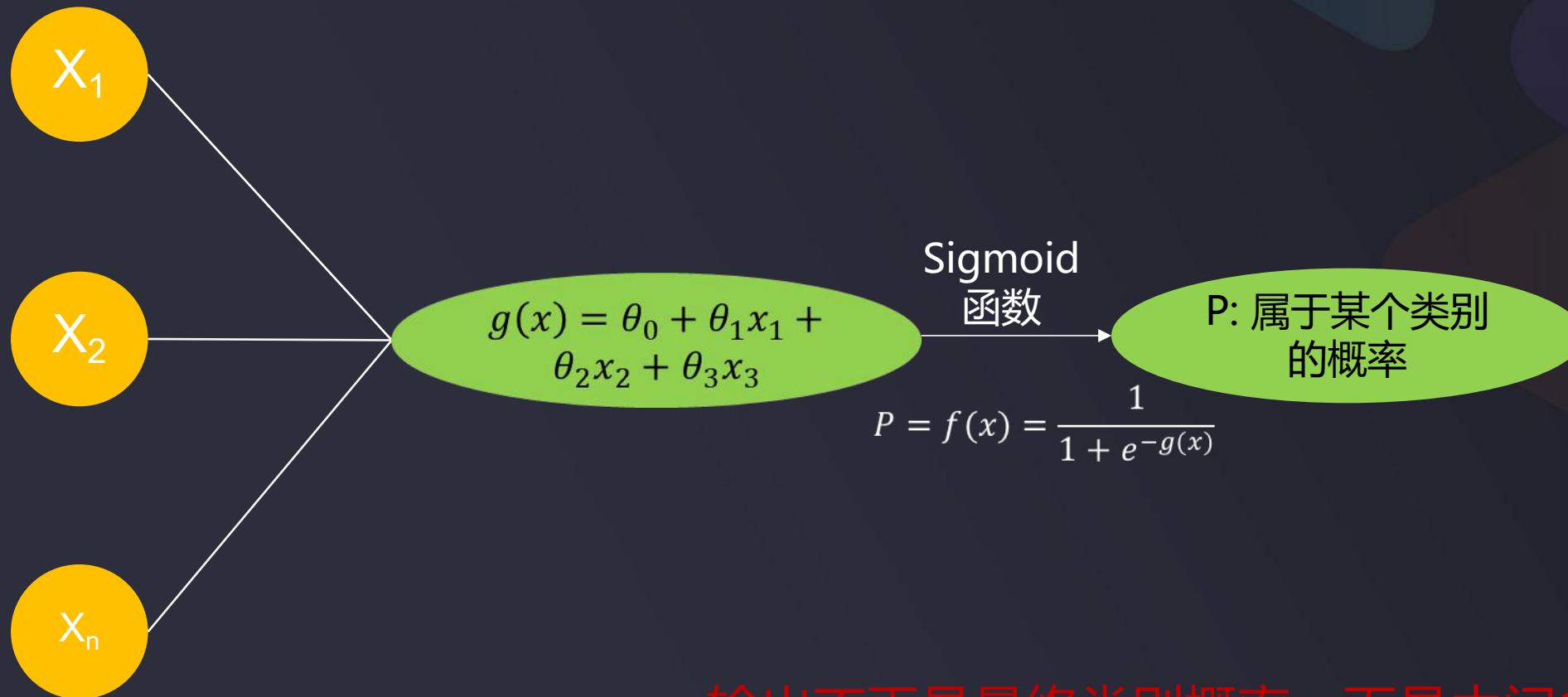
$$P(x) = \frac{1}{1 + e^{-x}}$$

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



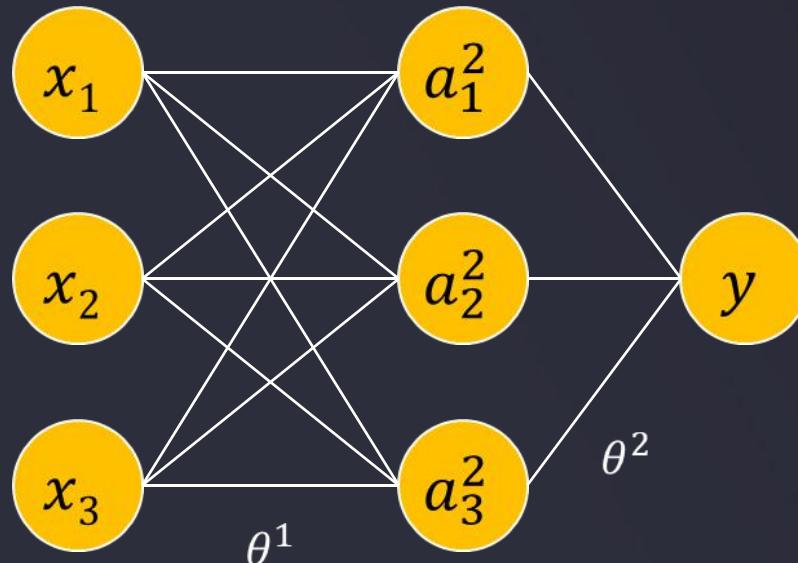
(2) 把多个逻辑回归以一定的方式连起来，组成更复杂的分类器！

逻辑回归模型框架



输出不再是最终类别概率，而是中间参数

逻辑回归组合在一起实现复杂逻辑判断



$$a_1^2 = f(\theta_{10}^1 x_0 + \theta_{11}^1 x_1 + \theta_{12}^1 x_2 + \theta_{13}^1 x_3)$$

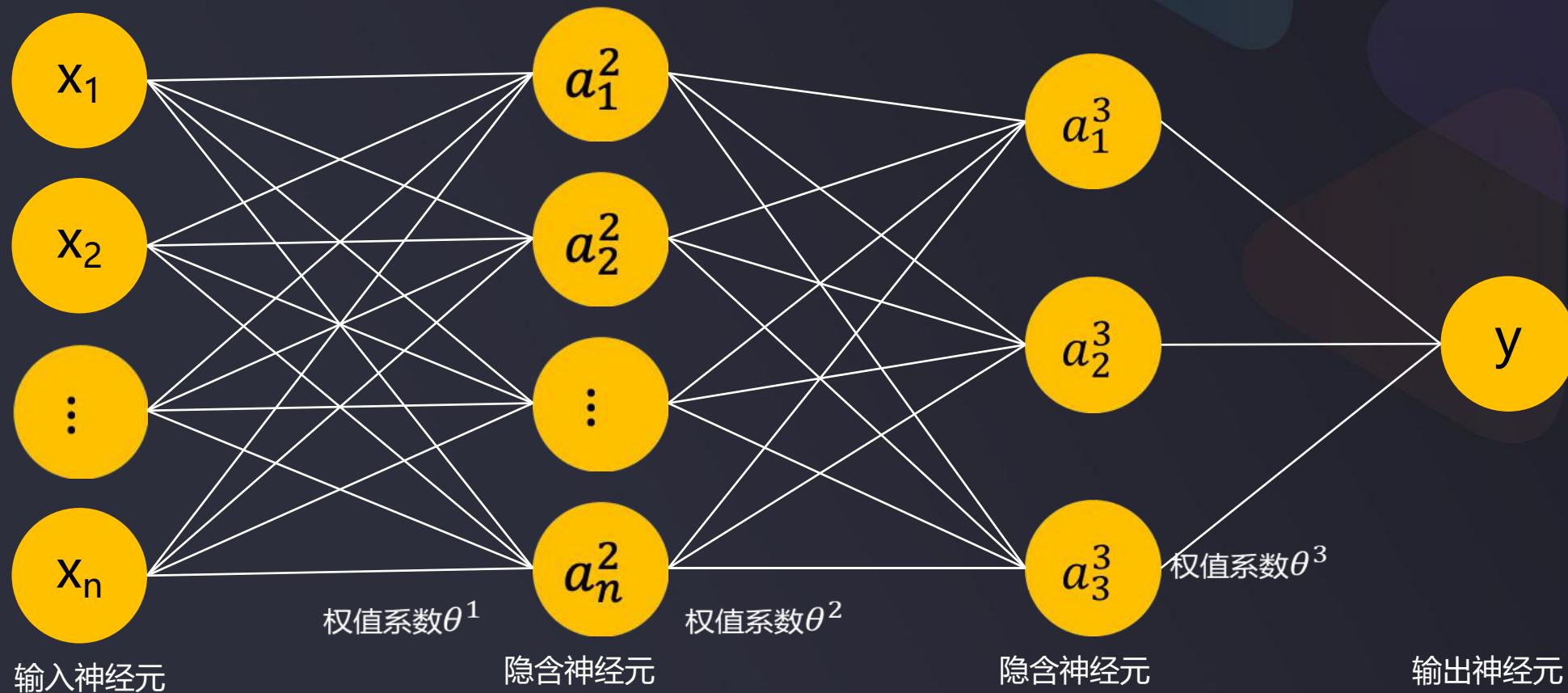
$$a_2^2 = f(\theta_{20}^1 x_0 + \theta_{21}^1 x_1 + \theta_{22}^1 x_2 + \theta_{23}^1 x_3)$$

$$a_3^2 = f(\theta_{30}^1 x_0 + \theta_{31}^1 x_1 + \theta_{32}^1 x_2 + \theta_{33}^1 x_3)$$

$$p = f(\theta_{10}^2 a_0^2 + \theta_{11}^2 a_1^2 + \theta_{12}^2 a_2^2 + \theta_{13}^2 a_3^2)$$

根据 p , 确定输出类别 y

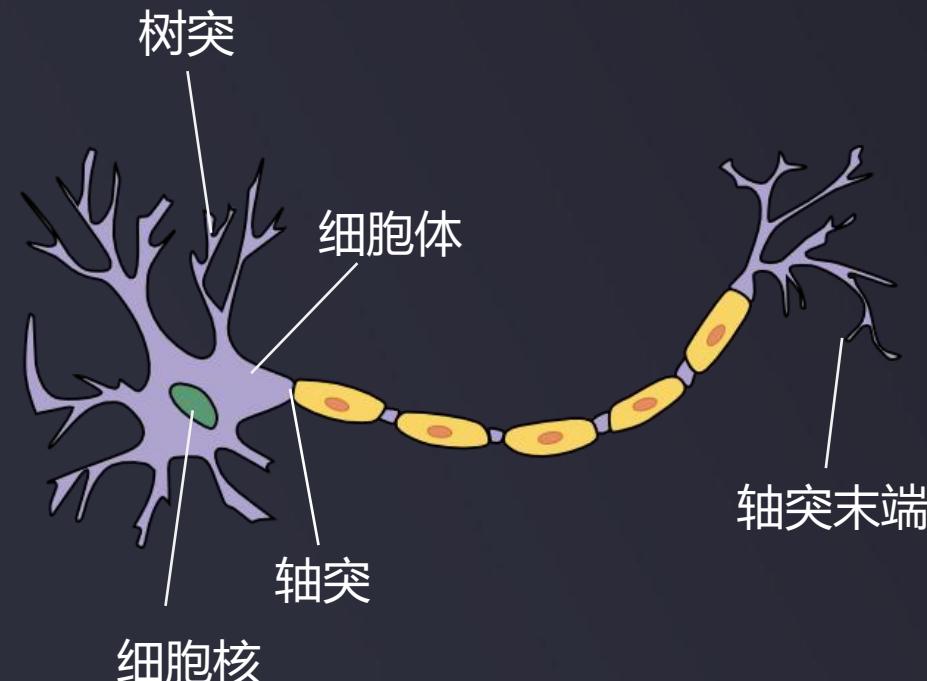
多层次感知器模型框架



|多层感知器（人工神经网络）

人工神经网络 (Artificial Neural Networks, 简写为ANN) : 一种类似于大脑神经突触联接的结构进行信息处理的数学模型

树突：
其作用是**接受**其它神经元传来的冲动，并将冲动传向胞体。

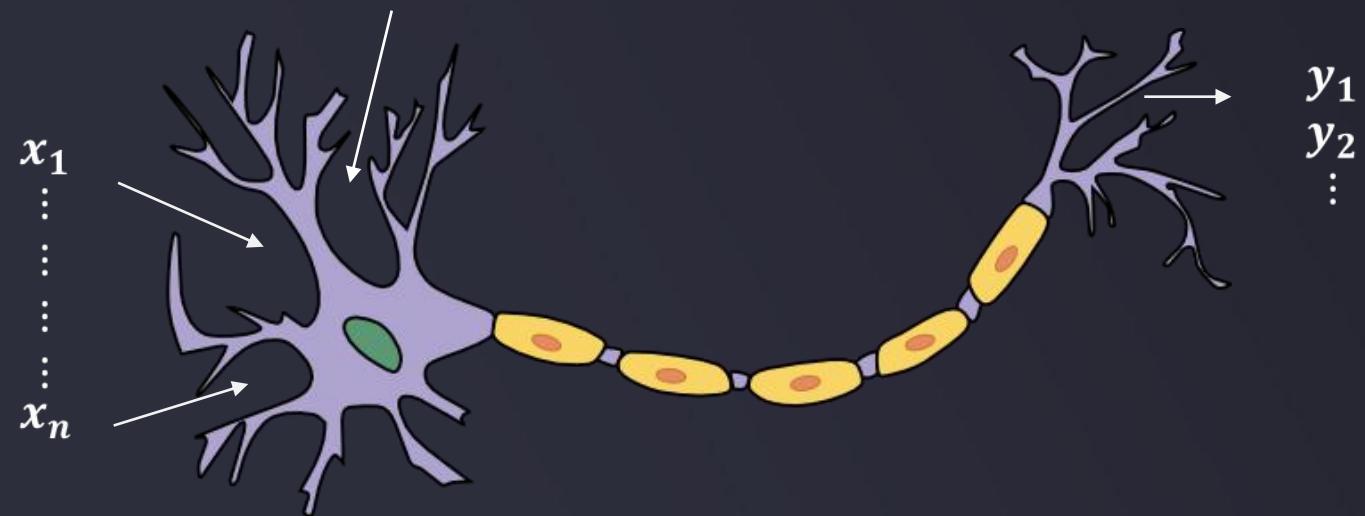


神经元结构

轴突：其主要作用是将神经元胞体所产生的兴奋冲动传至其它神经元或效应器，实现**信息传递**

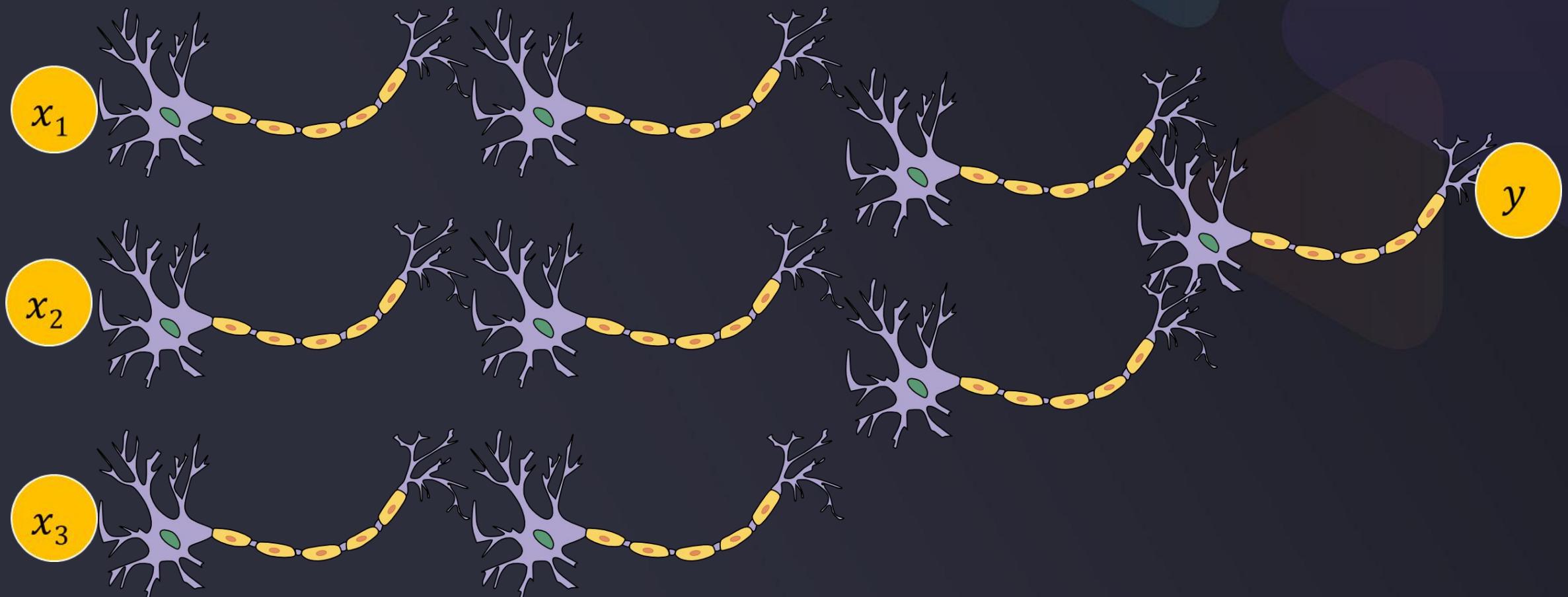
多层感知器（人工神经网络）

人工神经网络 (Artificial Neural Networks, 简写为ANN)：一种类似于大脑神经突触联接的结构进行信息处理的数学模型



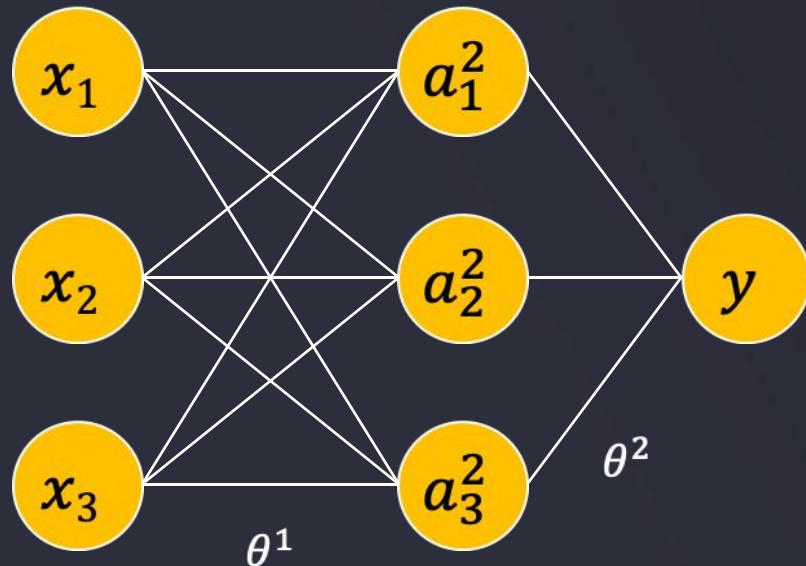
神经元信号数值化

|多层感知器（人工神经网络）



知识巩固

$x_1 = 3, x_2 = -2, x_3 = 1$, 计算y



$$a_1^2 = f(2x_1 - x_2 + 3x_3)$$

$$a_2^2 = f(2 + 2x_1 + 3x_2 - 4x_3)$$

$$a_3^2 = f(-3 + 3x_1 + x_2 + 2x_3)$$

$$p = f(3 + 2a_1^2 + a_2^2 - 3a_3^2)$$



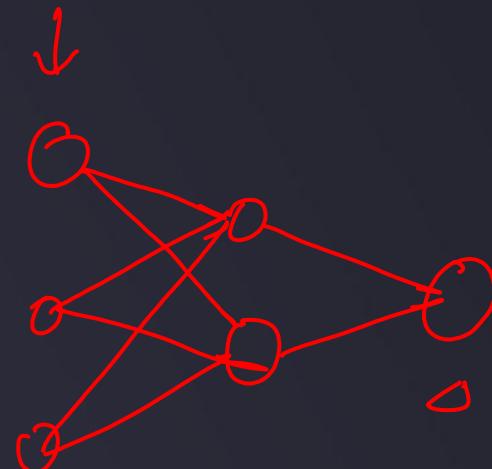
Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

赵辛

Chapter 2 深度学习之多层感知器

- 1 --逻辑回归实现非线性边界分类
- 2 --逻辑回归到多层感知器MLP
- 3 --MLP实现非线性分类
- 4 --MLP实现多分类
- 5 --激活函数
- 6 --MLP损失函数与反向传播算法
- 7 --实战准备
- 8 --实战(一)MLP实现非线性边界数据分类
- 9 --实战(二)Fashion_mnist服饰预测

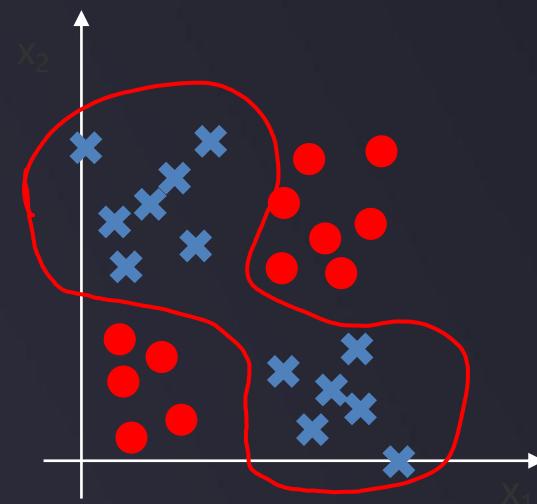
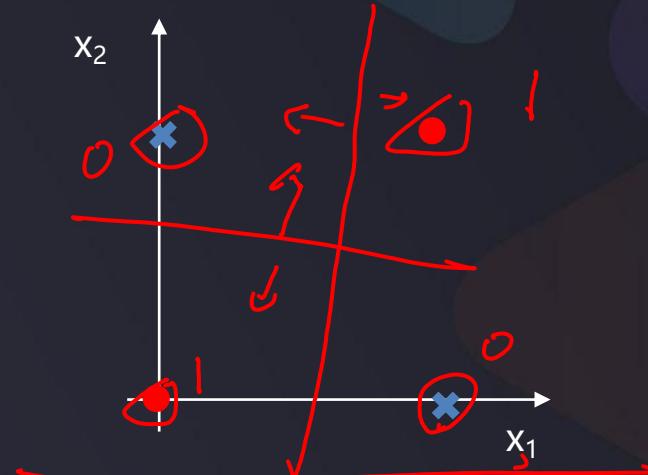


MLP实现同或门

⇒

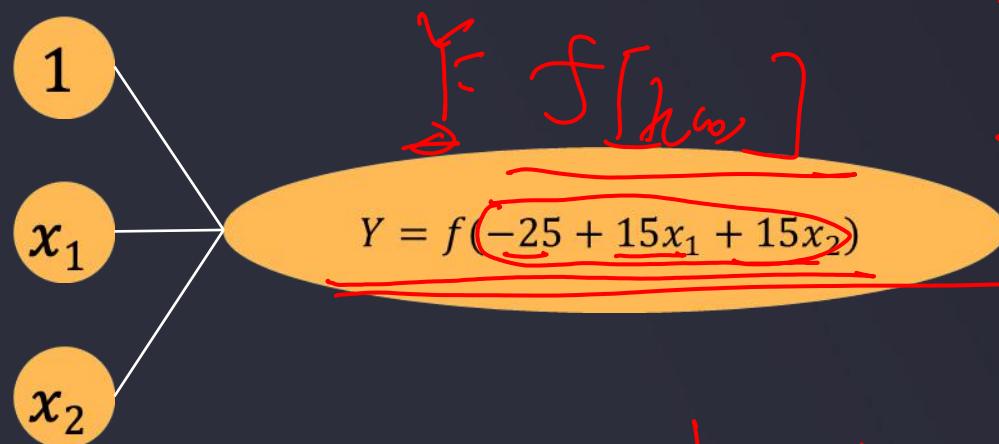
x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

$y : x_1 XNOR x_2$
 y 为 x_1 和 x_2 的同或门



逻辑回归实现与门

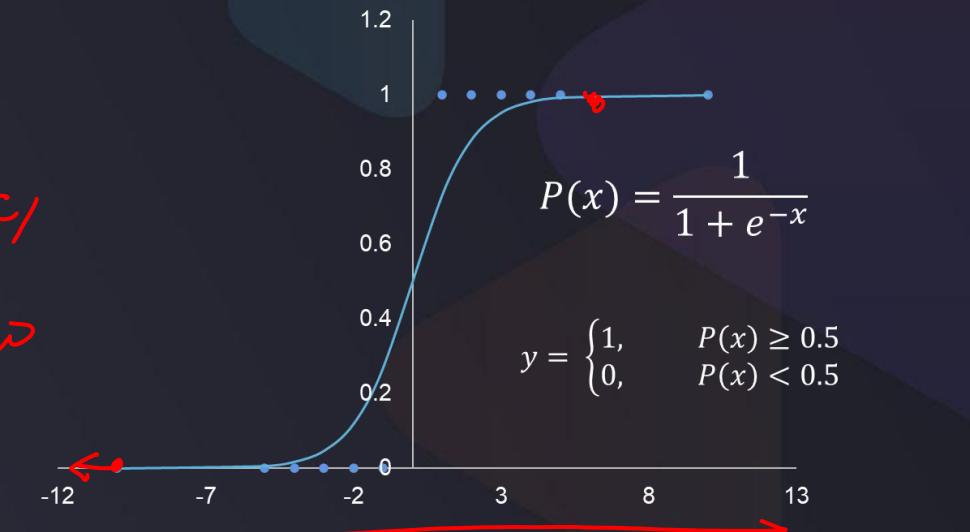
$y : x_1 \text{ [AND]} x_2, y$ 为 x_1 和 x_2 的与门



x_1	x_2	$h(x)$	y
0	0	-25	0
0	1	-15	0
1	0	-15	0
1	1	5	1

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow y = 1$$

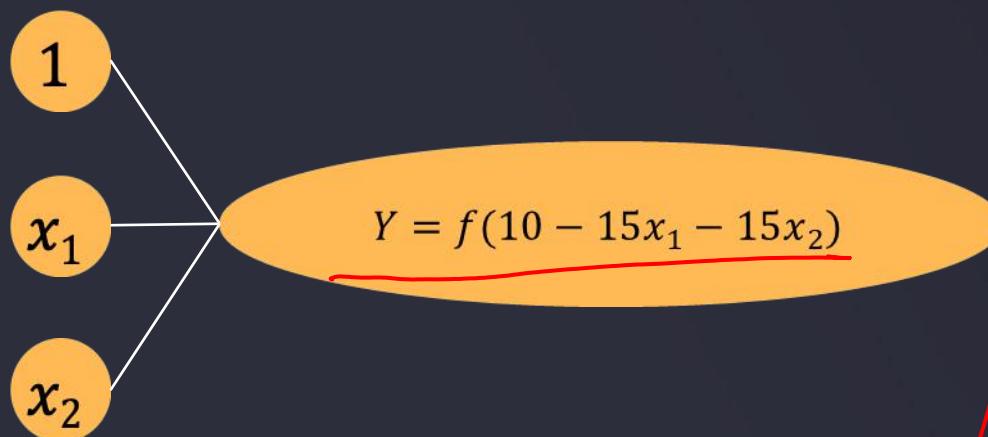
$$\begin{aligned} &\geq 0.5 \Rightarrow y=1 \\ &< 0.5 \Rightarrow y=0 \end{aligned}$$



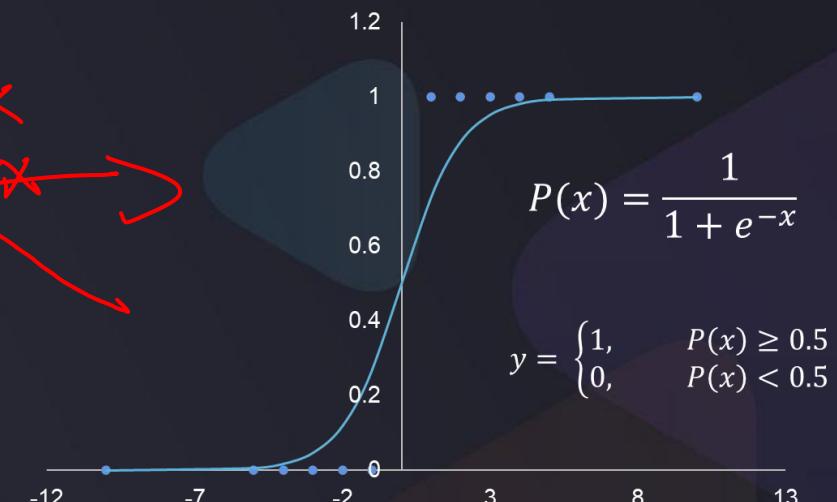
$$\begin{aligned} &x_1 = 0, x_2 = 0 \Rightarrow h(0, 0) = -25 + 0 + 0 = -25 \Rightarrow T = 0 < 0.5 \Rightarrow y = 0 \\ &x_1 = 0, x_2 = 1 \Rightarrow h(0, 1) = -25 + 0 + 15 = -10 \Rightarrow T = 0 < 0.5 \Rightarrow y = 0 \\ &x_1 = 1, x_2 = 0 \Rightarrow h(1, 0) = -25 + 15 + 0 = -10 \Rightarrow T = 0 < 0.5 \Rightarrow y = 0 \\ &x_1 = 1, x_2 = 1 \Rightarrow h(1, 1) = -25 + 15 + 15 = 5 \Rightarrow T = 5 > 0.5 \Rightarrow y = 1 \end{aligned}$$

逻辑回归实现非与门

$$y: (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$$



x_1	x_2	$h(\theta, x)$	y
0	0	10	1
0	1	-5	0
1	0	-5	0
1	1	-20	0



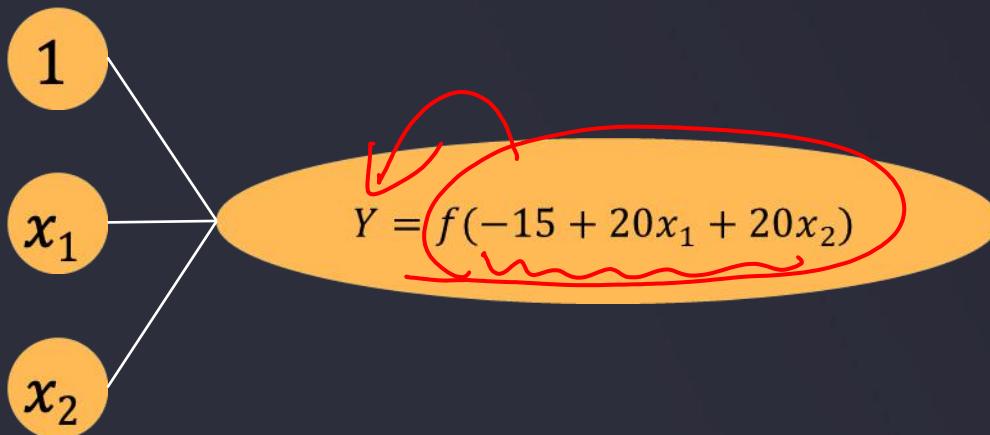
对于 $x_1 = 0, x_2 = 0$, $h = 10 - 0 - 0 = 10$
 $\Rightarrow Y \sim 1 \Rightarrow out \Rightarrow y = 1$

对于 $x_1 = 0, x_2 = 1$ $\rightarrow h = 10 - 15 - 0 = -5$
 $Y \sim 0 \Rightarrow y = 0$

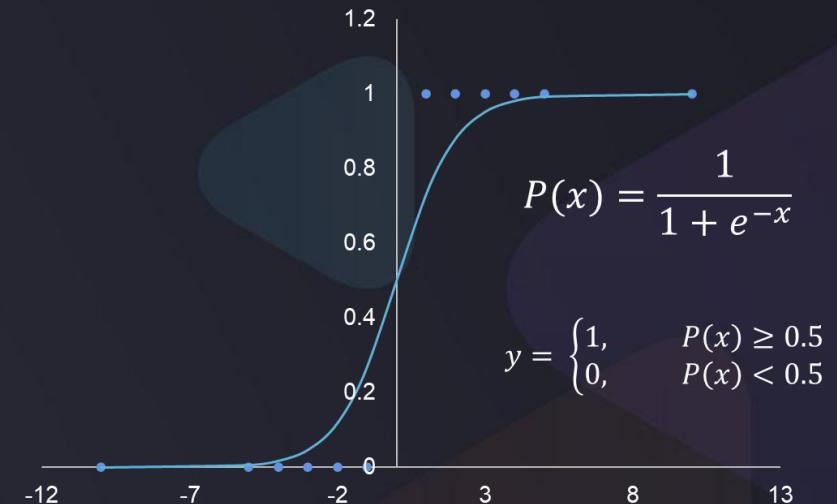
$10 - 15 - 15 = -20$

逻辑回归实现或门

$y : x_1 \text{ OR } x_2, y$ 为 x_1 和 x_2 的或门



x_1	x_2	$h(\theta, x)$	y
0	0	-15	0
0	1	5	1
1	0	5	1
1	1	25	1



$$\Rightarrow x_1, x_2 = 0 \Rightarrow h = -15 + 0 + 0 = -15$$

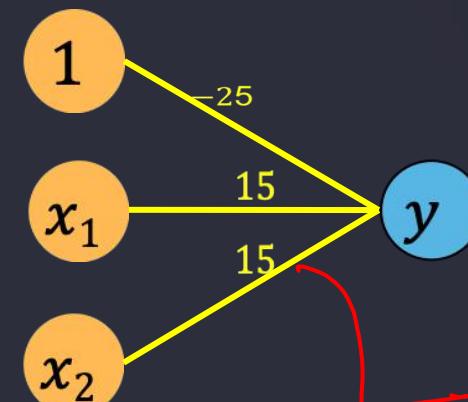
$$\Rightarrow \boxed{x_1 = 0} \cup \boxed{x_2 = 0} \Rightarrow \boxed{y = 0}$$

$$\Rightarrow x_1 = 0, x_2 = 1 \Rightarrow h = -15 + 20 = 5$$

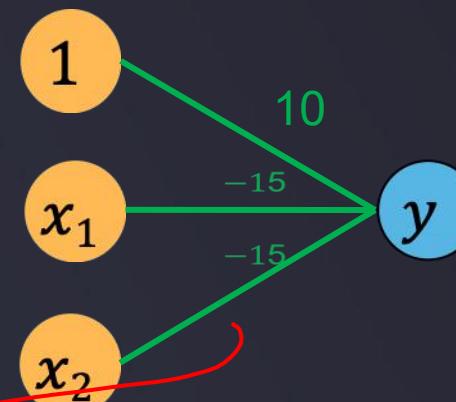
$$\Rightarrow \boxed{x_1 = 1} \cup \boxed{x_2 = 1} \Rightarrow \boxed{y = 1}$$

MLP实现同或门

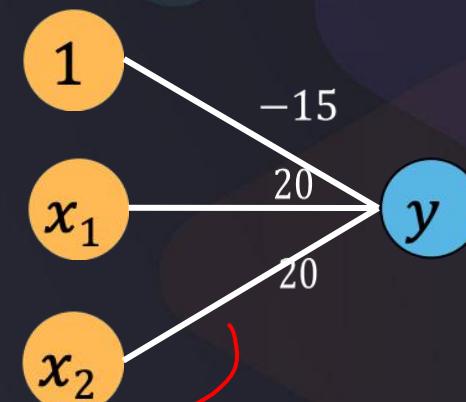
$$y : x_1 \text{ AND } x_2$$



$$y : (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$$



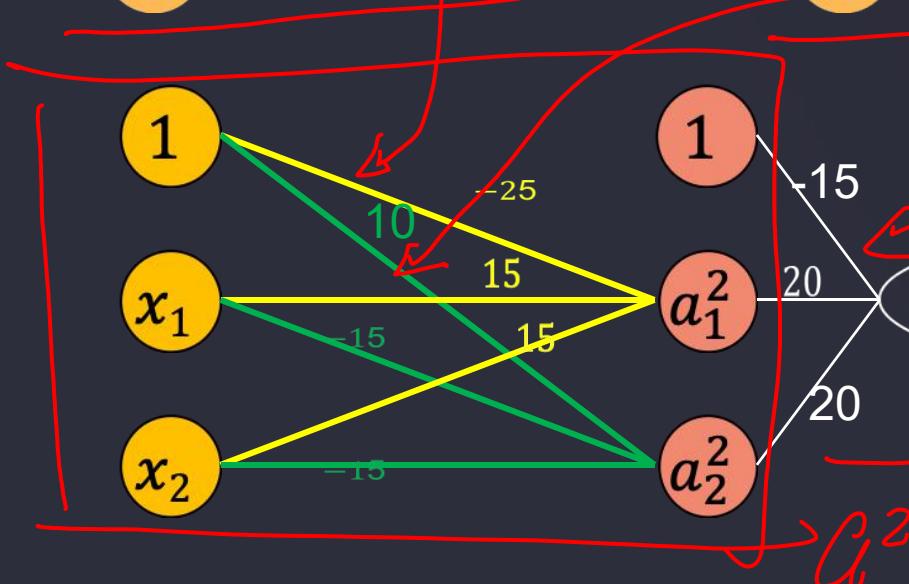
$$y : x_1 \text{ OR } x_2$$



$$y : x_1 \text{ XNOR } x_2$$

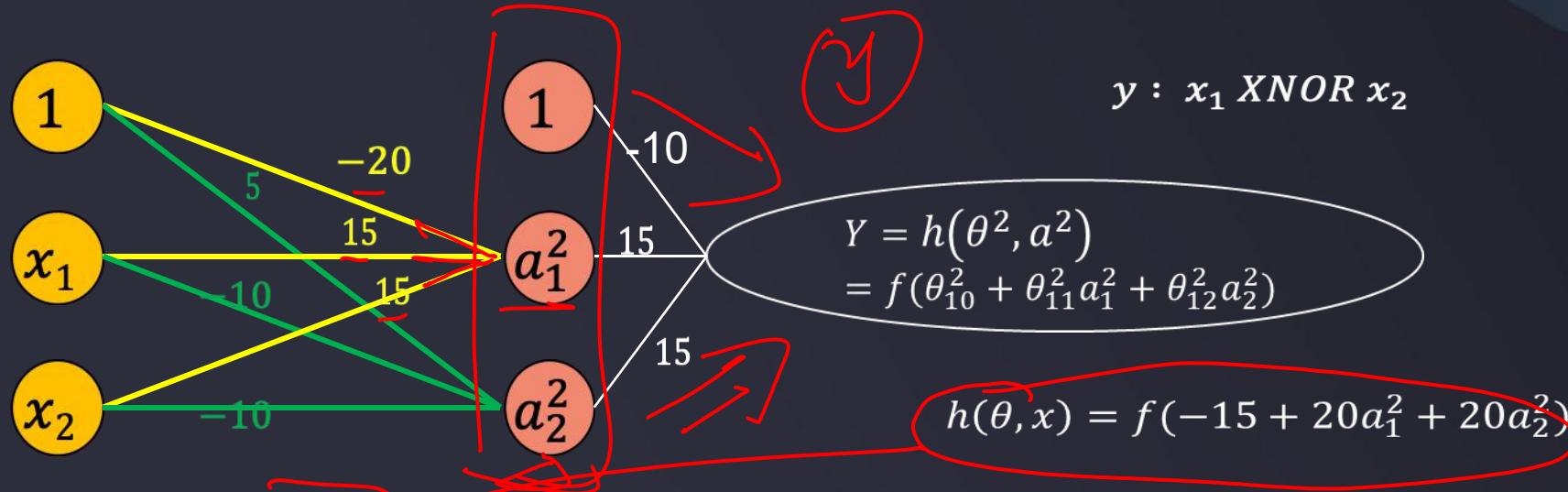
$$\begin{aligned} Y &= h(\theta^2, a^2) \\ &= f(\theta_{10}^2 + \theta_{11}^2 a_1^2 + \theta_{12}^2 a_2^2) \end{aligned}$$

$$h(\theta, x) = f(-15 + 20a_1^2 + 20a_2^2)$$



MLP实现同或门

$$y : x_1 \text{ XNOR } x_2$$



$x_1, x_2 = 0$

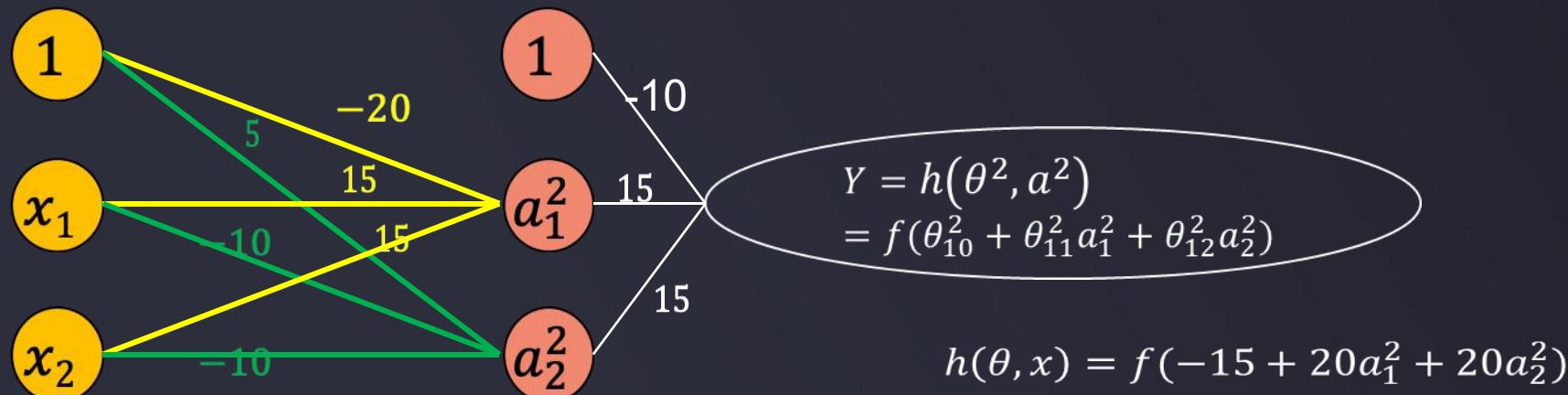
$$\begin{aligned} a_1^2 &= f(-20 + 15x_1 + 15x_2) = f(-20) = 0 \\ a_2^2 &= f(5 - 10x_1 - 10x_2) = f(5) = 1 \end{aligned}$$

$$Y = f(-15 + 20 \times 0 + 20 \times 1) = f(5) = 1$$

	a_1^2	a_2^2	$h(\theta, x)$	y
0	0	0	$f(5)$	1
0	1			
1	0			
1	1			

MLP实现同或门

$$y : x_1 \text{ XNOR } x_2$$

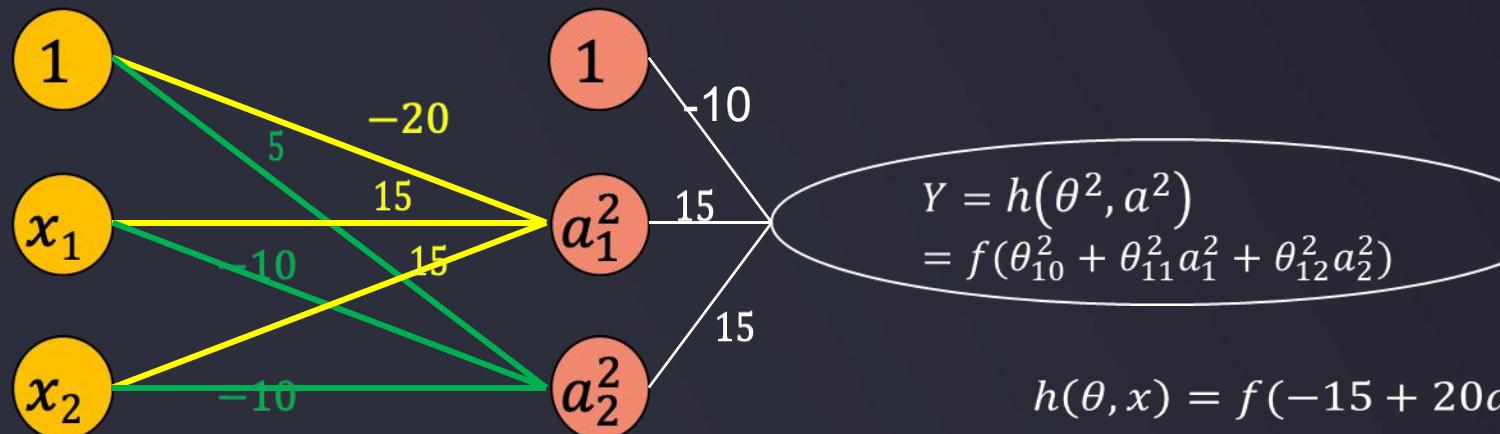


$$\begin{aligned} x_1=0, x_2=1 &\Rightarrow a_1^2 = f(-20+0+15) = f(5)=1 \\ a_2^2 &= f(5+0-10) = f(-5)=0 \\ \Rightarrow Y &= f(-5+20\times 0 - 20\times 0) = 0 \end{aligned}$$

x_1	x_2	a_1^2	a_2^2	$h(\theta, x)$	y
0	0	0	1	$f(5)$	1
0	1	0	0	$f(-15)$	0
1	0				
1	1				

MLP实现同或门

$$y : x_1 \text{ XNOR } x_2$$



$$x_1=1 \Rightarrow a_1^2 = f(-20 + 15 \times 1 + 0) = f(-5) = 0$$

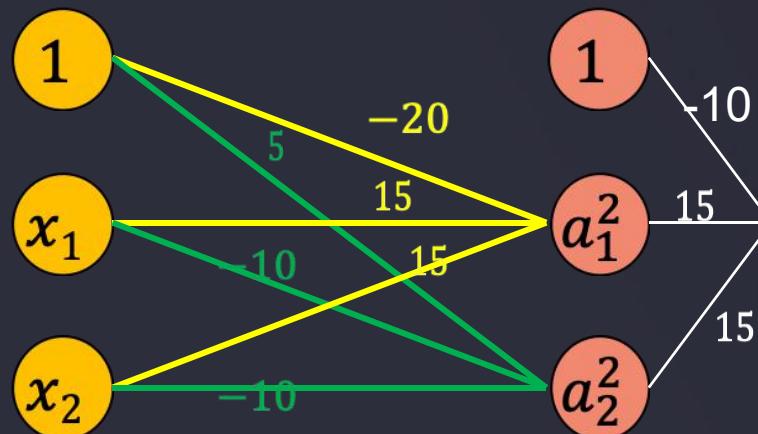
$$x_2=0 \quad a_2^2 = f(5 - 10 \times 1 + 0) = f(-5) = 0$$

$$\Rightarrow Y = f(-15 + 0 + 0) = f(-15) = 0$$

x_1	x_2	a_1^2	a_2^2	$h(\theta, x)$	y
0	0	0	1	$f(5)$	1
0	1	0	0	$f(-15)$	0
1	0	0	0	$f(-15)$	0
1	1				1

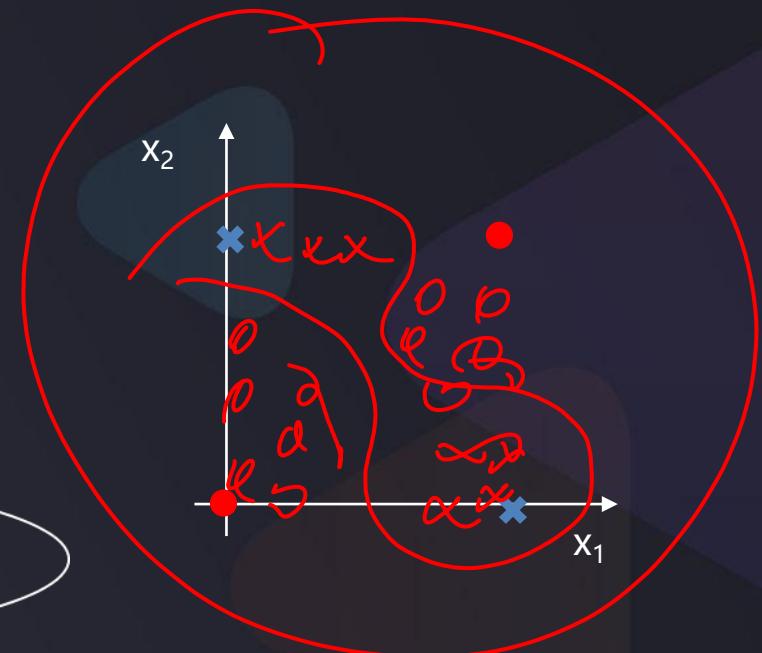
MLP实现同或门

$$y : x_1 \text{ XNOR } x_2$$



$$Y = h(\theta^2, a^2) \\ = f(\theta_{10}^2 + \theta_{11}^2 a_1^2 + \theta_{12}^2 a_2^2)$$

$$h(\theta, x) = f(-15 + 20a_1^2 + 20a_2^2)$$



$$x_1 = 1$$

$$a_1^2 = f(-20 + 15 + 15) = 1$$

$$x_2 = 1$$

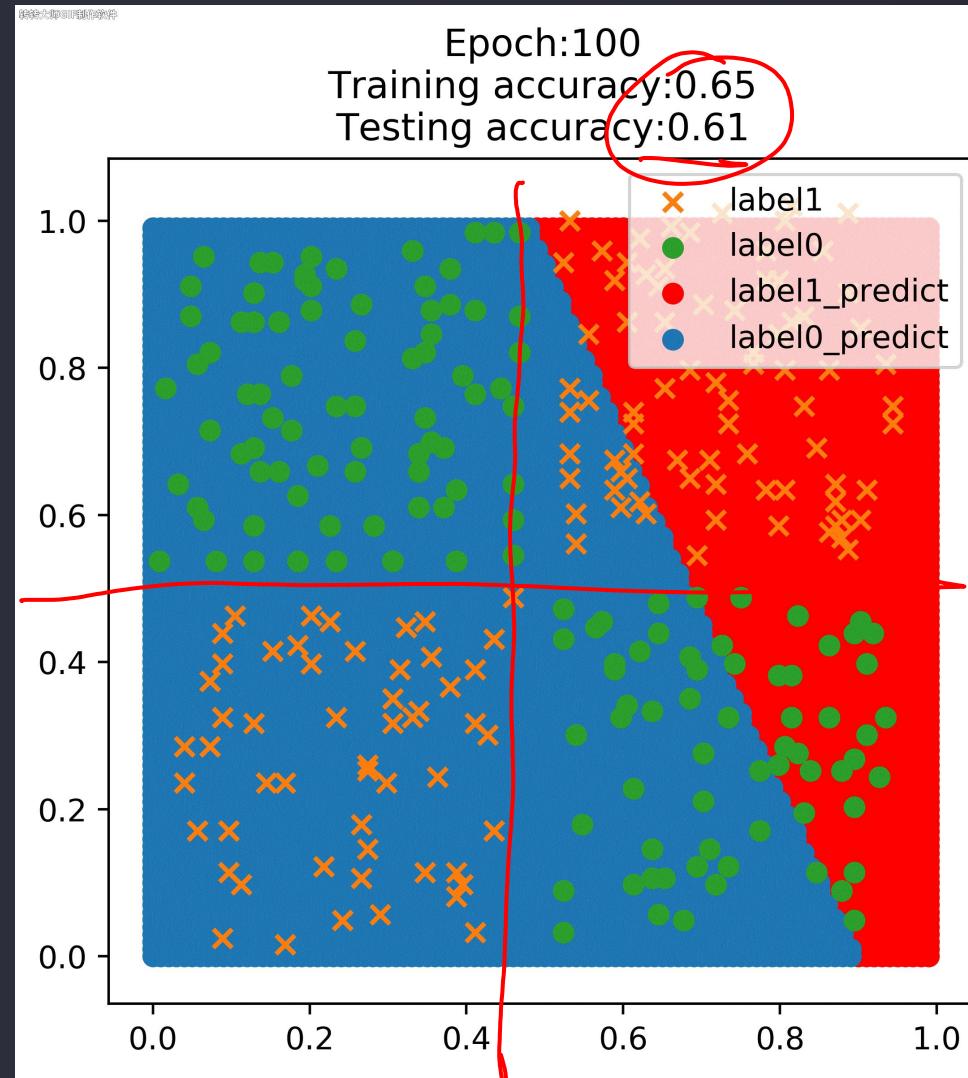
$$a_2^2 = f(5 - 10 - 10) = 0$$

$$\Rightarrow Y = f(-15 + 20x_1 + 0) = f(5) = 1$$



x_1	x_2	a_1^2	a_2^2	$h(\theta, x)$	y
0	0	0	1	$f(5)$	1
0	1	0	0	$f(-15)$	0
1	0	0	0	$f(-15)$	0
1	1	1	0	$f(5)$	1

MLP实现非线性分类预测



输入数据: x_1, x_2

输出类别: $y \in \{0, 1\}$

要求: 不增加新的特征数据, 实现二分类

x_1, x_2, y^2



Python3人工智能入门+实战提升：深度学习

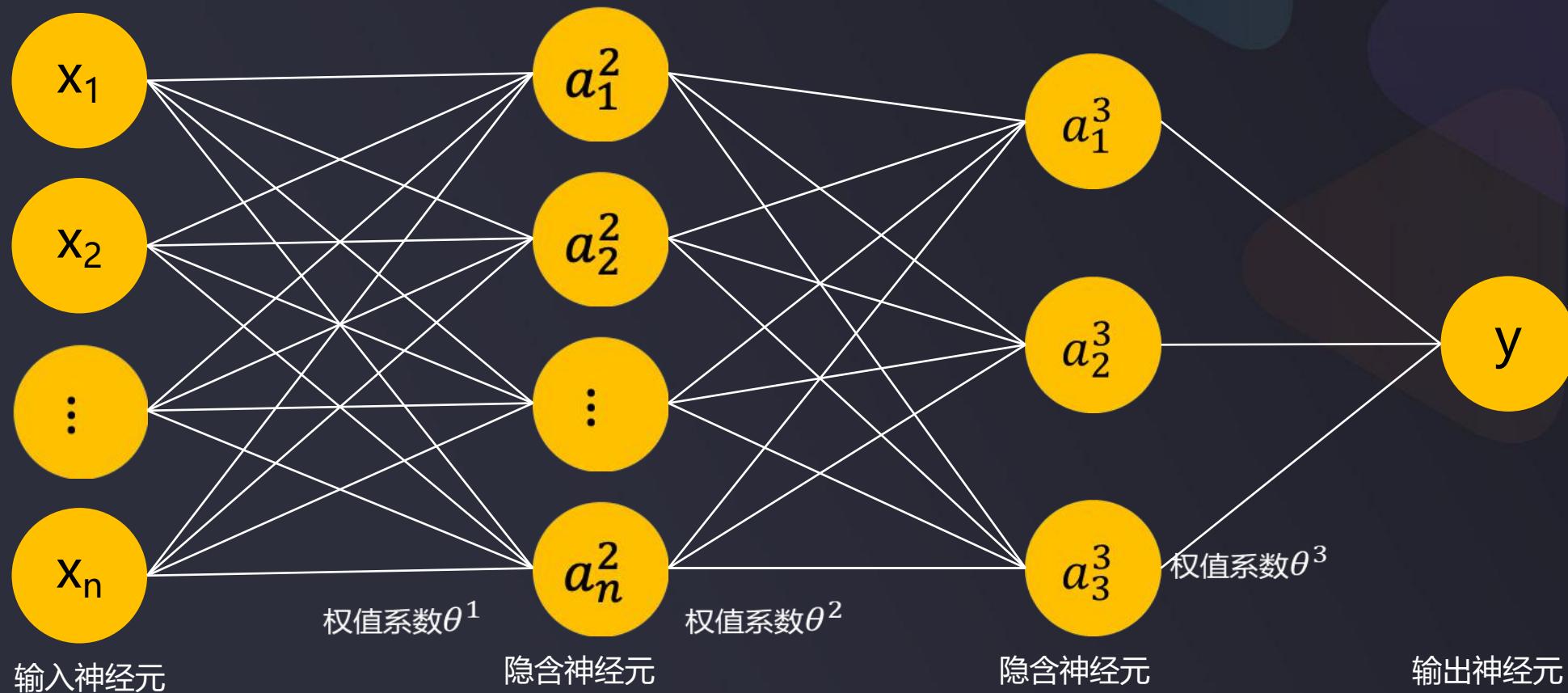
Chapter 2 深度学习之多层感知器

赵辛

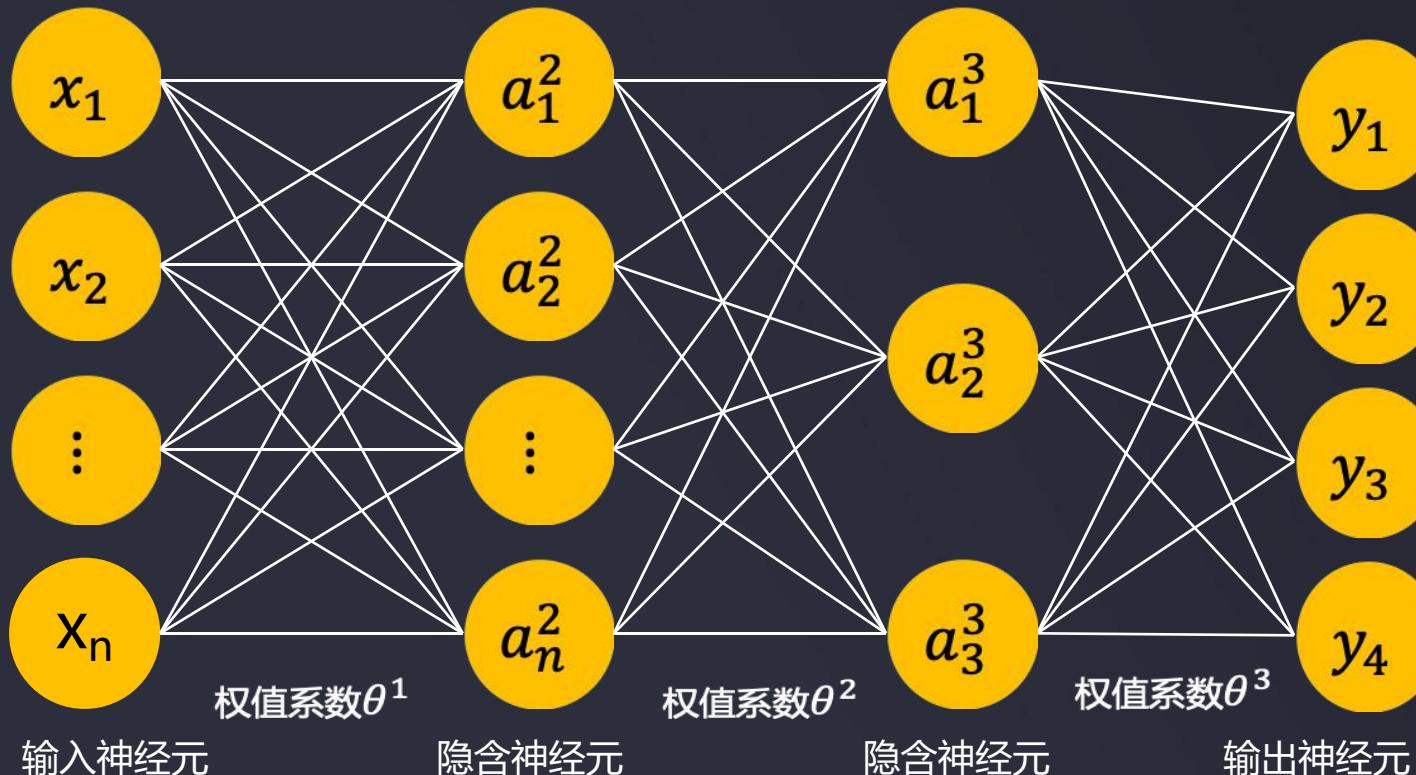
Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

多层次感知器模型框架



MLP实现多分类预测



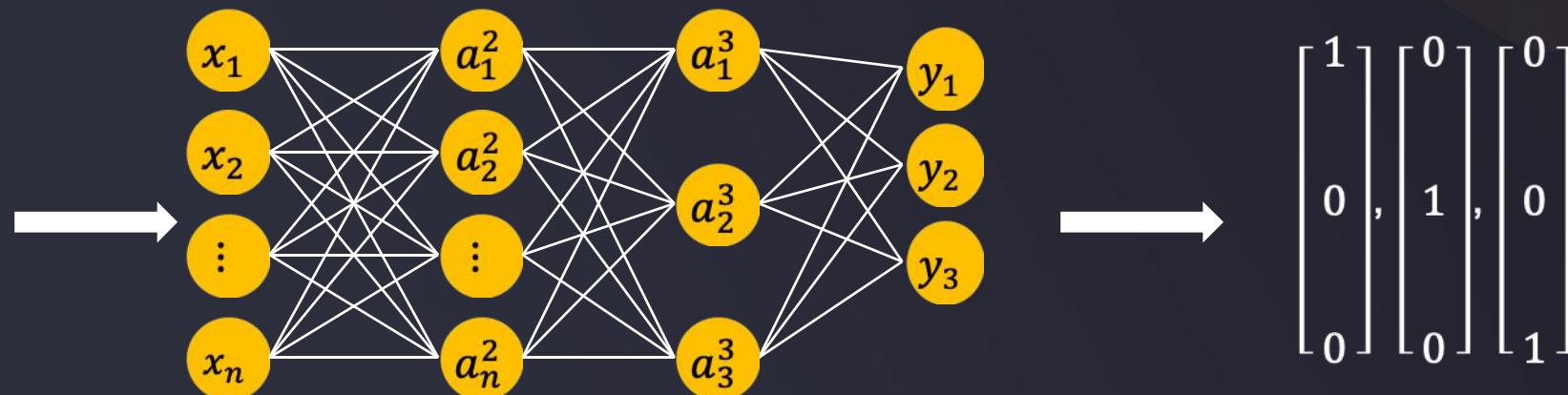
计算各类别预测概率

选取**概率最大的**，其所属类别即为模型预测类别

MLP实现多分类预测



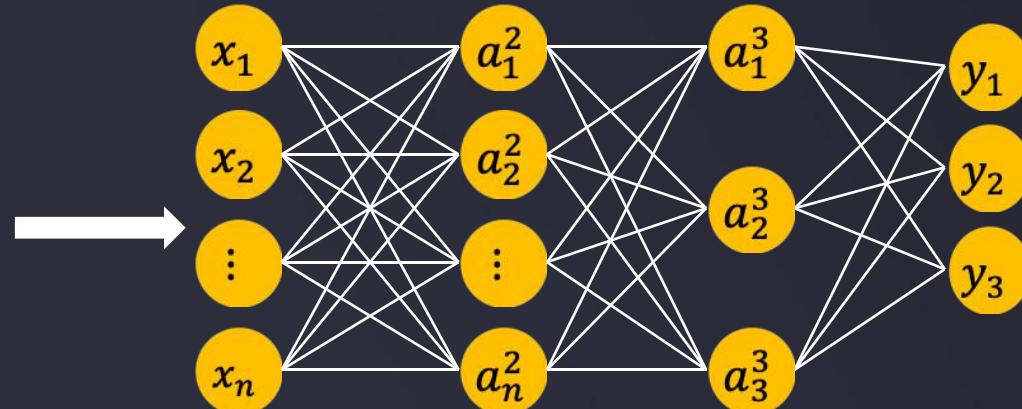
把图片每个像素点的灰度值输入
MLP模型



MLP实现多分类预测

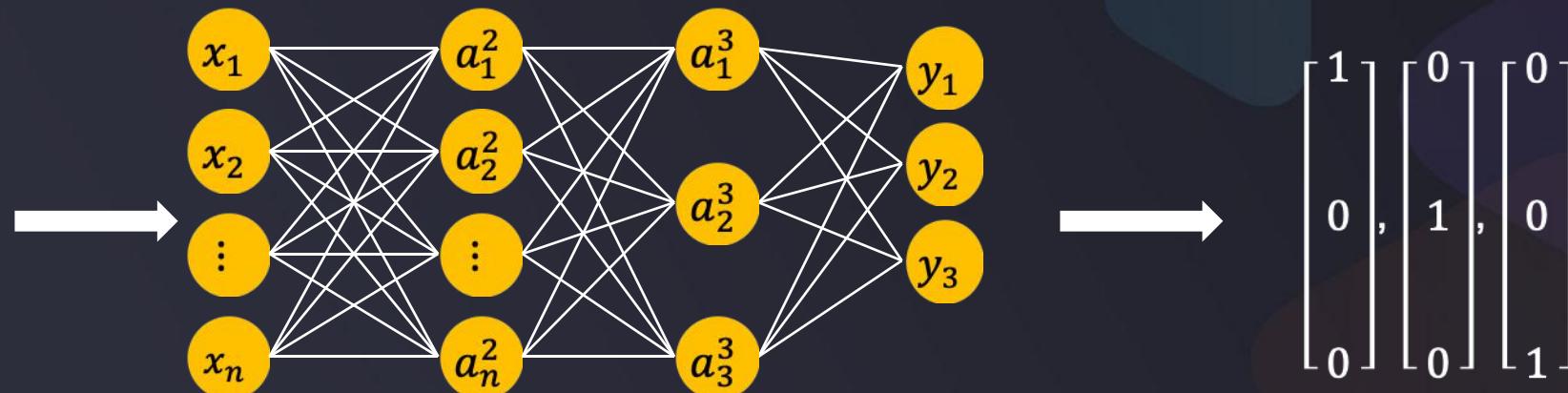


```
[[186. 190. 193. ... 232. 233. 233.]  
[185. 190. 194. ... 230. 233. 233.]  
[187. 190. 192. ... 229. 228. 228.]  
...  
[110. 103. 100. ... 197. 207. 231.]  
[106. 104. 103. ... 204. 226. 241.]  
[142. 136. 123. ... 218. 238. 249.]]
```



MLP实现多分类预测

把图片每个像素点的灰度值输入
MLP模型



需要对输出结果
进行预处理!

$$y \in \{1, 2, 3\}$$

$$y \in \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

知识巩固

问题：以下为MLP模型实现图像多分类预测的流程，对其进行正确排序。

- A、配置模型训练参数
- B、图片加载并将其转换为数组格式
- C、模型训练与预测
- D、对输出结果进行格式转化
- E、对输入数据进行维度转换与数据预处理
- F、建立MLP模型



Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

赵辛

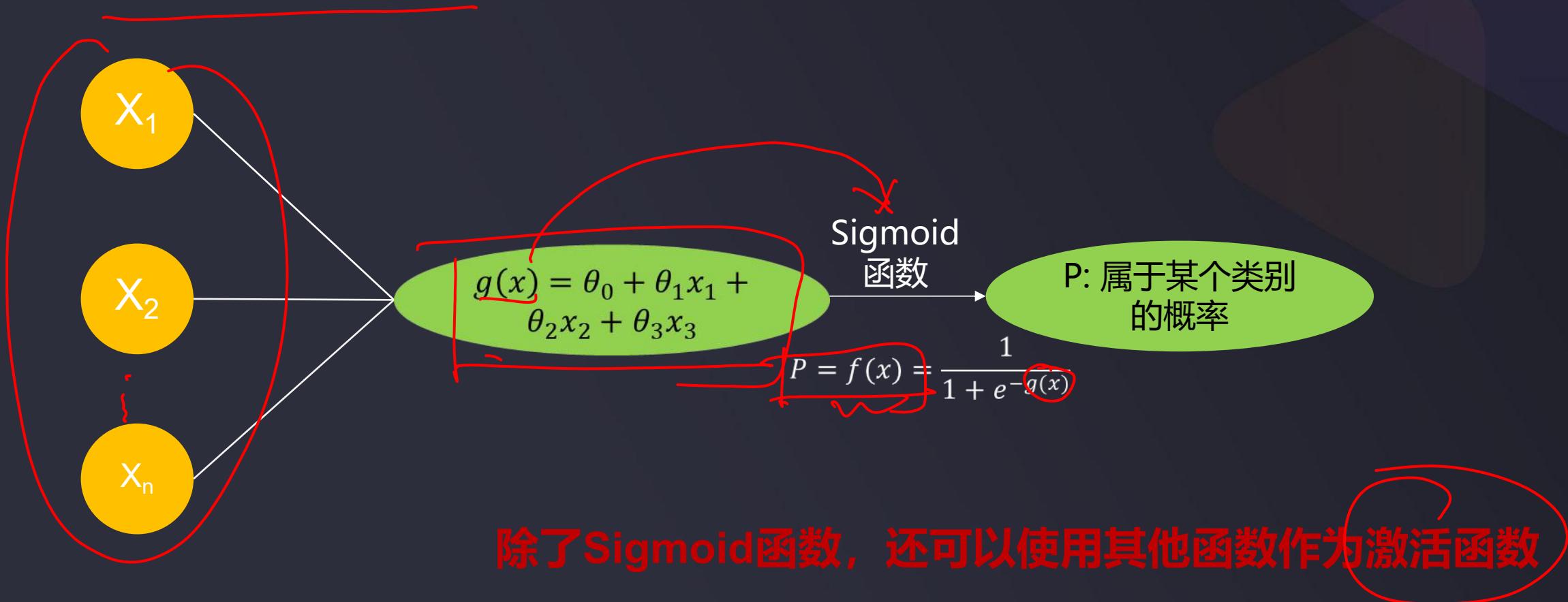
Chapter 2 深度学习之多层感知器

- 1 --逻辑回归实现非线性边界分类
- 2 --逻辑回归到多层感知器MLP
- 3 --MLP实现非线性分类
- 4 --MLP实现多分类
- 5 --激活函数 ✓
- 6 --MLP损失函数与反向传播算法
- 7 --实战准备
- 8 --实战(一)MLP实现非线性边界数据分类
- 9 --实战(二)Fashion_mnist服饰预测

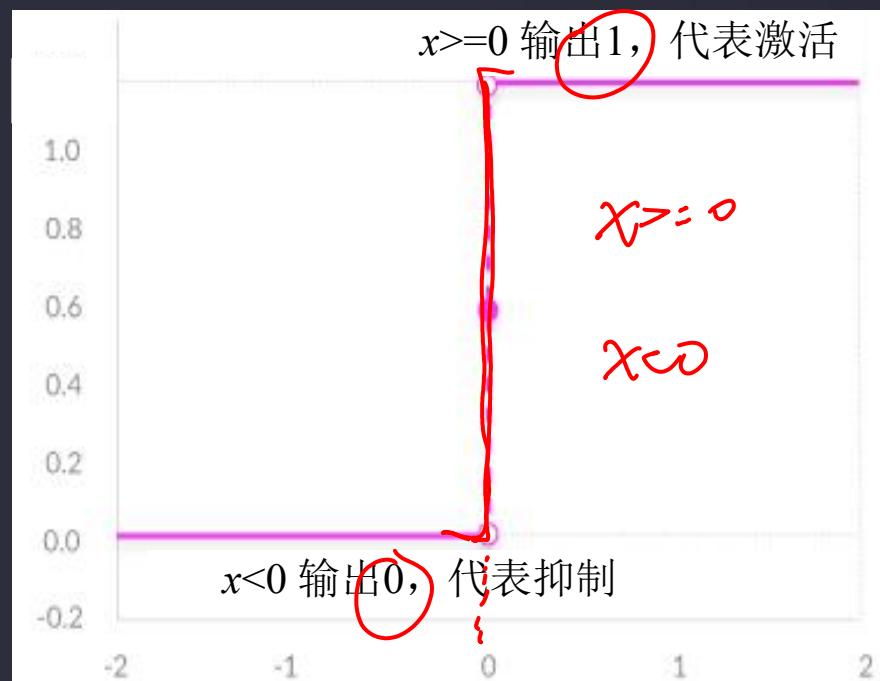
激活函数

$$\rightarrow \text{D} \rightarrow f(x) = \frac{1}{1+e^{-x}}$$

激活函数 (Activation Function)，就是在人工神经网络的神经元上运行的函数，负责将神经元的输入映射到输出端



阶跃函数



最基础的判断方法：阈值区分

$$f(x) = \begin{cases} 1, & \text{if } x \geq T \\ 0, & \text{if } x < T \end{cases}$$

T为阈值

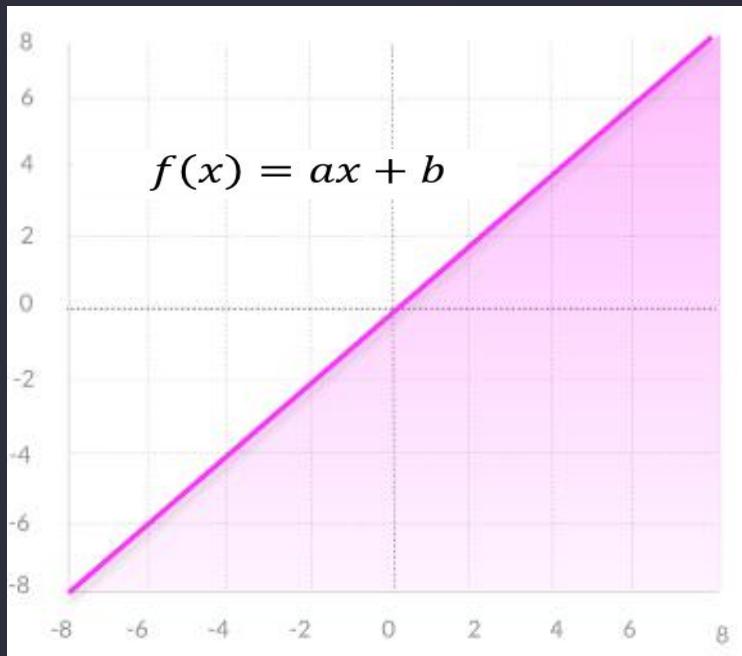
优点：简单易用

缺点：函数不光滑、不连续，不可导

线性函数

MLP

最简单的连续函数：



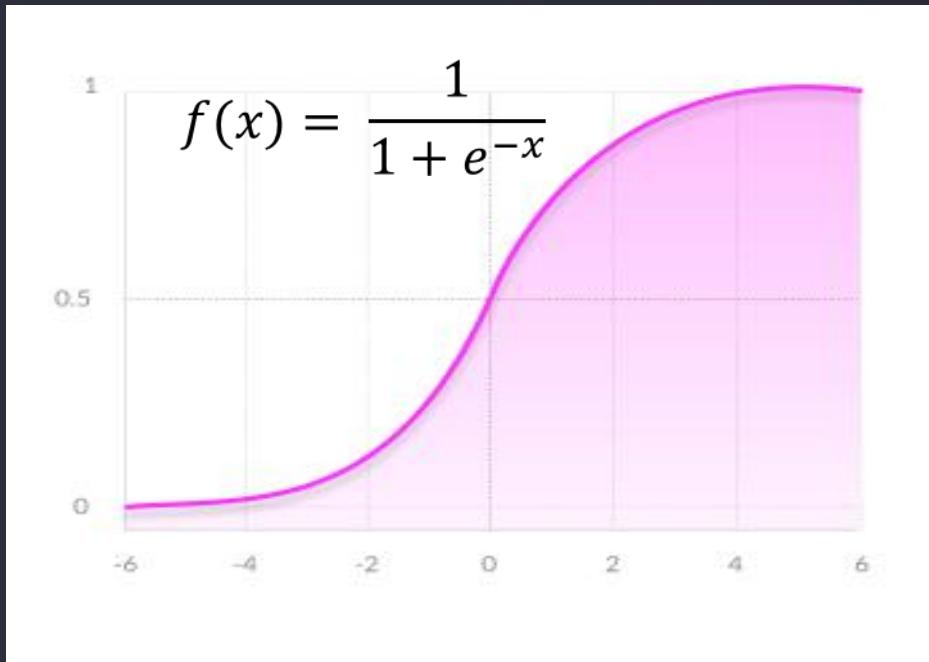
$$f(x) = ax + b$$

优点：多个输出，而不仅仅是“是”和“不是”（1/0）

缺点：

- (1) 无法使用梯度下降法来训练模型：导数是常数，并且与输入x无关，不利于模型求解过程中对权重的确定；
- (2) 神经网络的所有层都将折叠为线性激活关系：无论神经网络中有多少层，最后一层都是第一层的线性函数（因为线性函数的线性组合仍然是线性函数）

Sigmoid函数



优点:

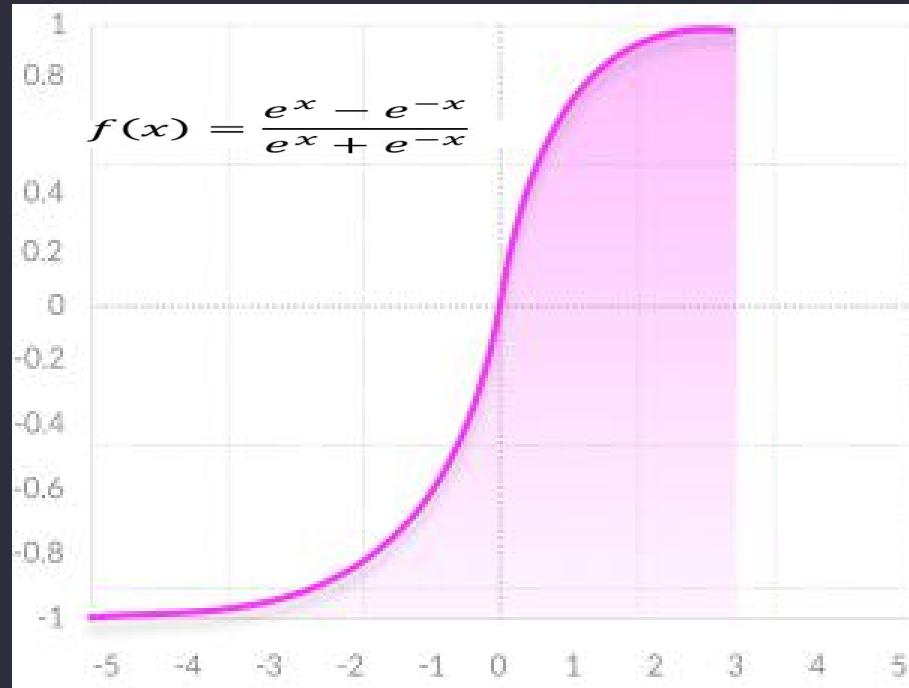
- (1) 平滑的渐变，防止输出值“跳跃”。
- (2) 输出值介于0和1之间，对每个神经元的输出进行标准化。
- (3) 清晰的预测:对于大于2或小于-2的x，趋向于将y值(预测)带到曲线的边缘，非常接近1或0

缺点:

- (1) 消失梯度:双边区域数值饱和(x很大或很小)导致随着x变化带来的y变化很小，导数趋于零，容易造成模型求解梯度消失问题。这可能导致网络求解过程中拒绝进一步学习，或者太慢而无法获得准确的预测。
- (2) 输出y中心不是零。

Tanh函数

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\text{sigmoid}(2x) - 1$$



优点:

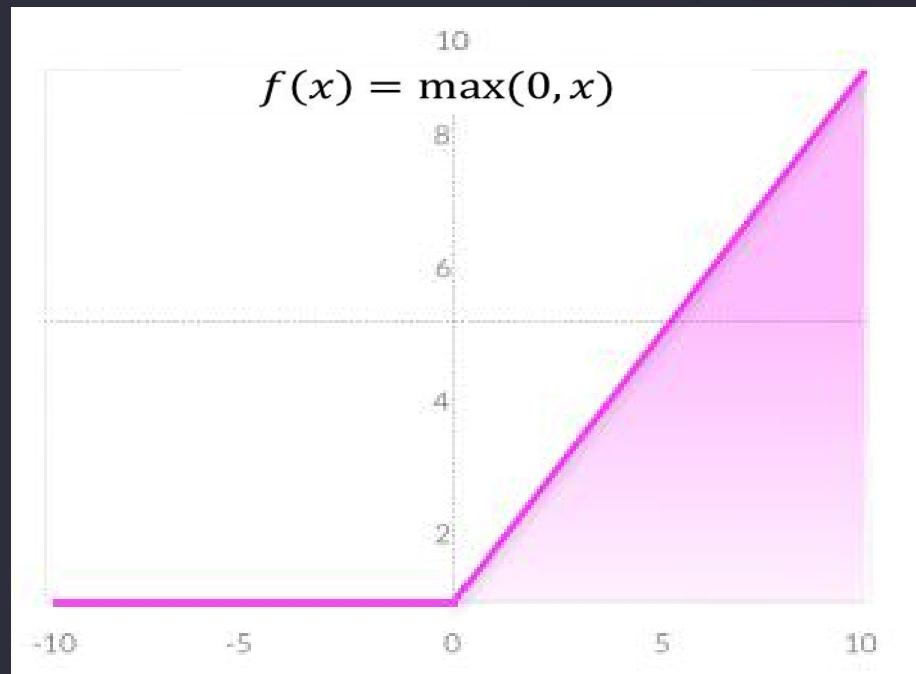
- (1) 正负方向以原点对称，输出均值是0（与很多样本的分布均值接近），使得其收敛速度要比sigmoid快，减少迭代次数。
- (2) 具有Sigmoid函数的优点。

缺点:

- (1) 与Sigmoid函数一样，也存在消失梯度问题

ReLU函数

$$f(x) = \max(0, x)$$



优点:

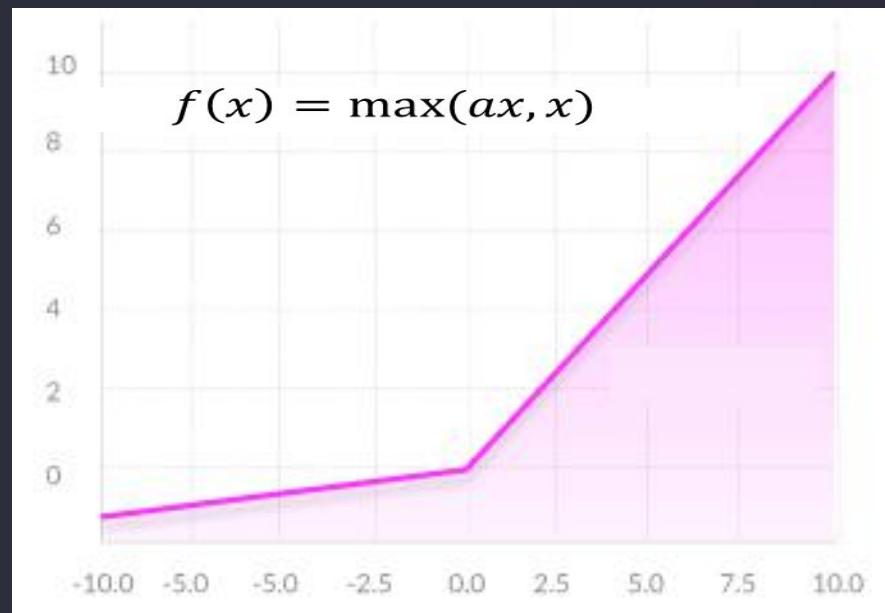
- (1) 计算效率高，允许网络快速收敛。
- (2) 非线性，尽管ReLU看起来像线性函数，但它具有导数函数并允许反向传播。

缺点:

- (1) 神经元死亡问题：当输入接近零或为负时，函数的梯度变为零，网络将无法执行反向传播，也无法学习。

Leaky Relu函数

$$f(x) = \max(ax, x)$$



优点:

- (1) 解决了Relu的神经元死亡问题， 在负区域具有小的正斜率，因此即使对于负输入值，它也可以进行反向传播
- (2) 具有Relu函数的优点

缺点:

- (1) 结果不一致，无法为正负输入值提供一致的关系预测
(不同区间函数不同)

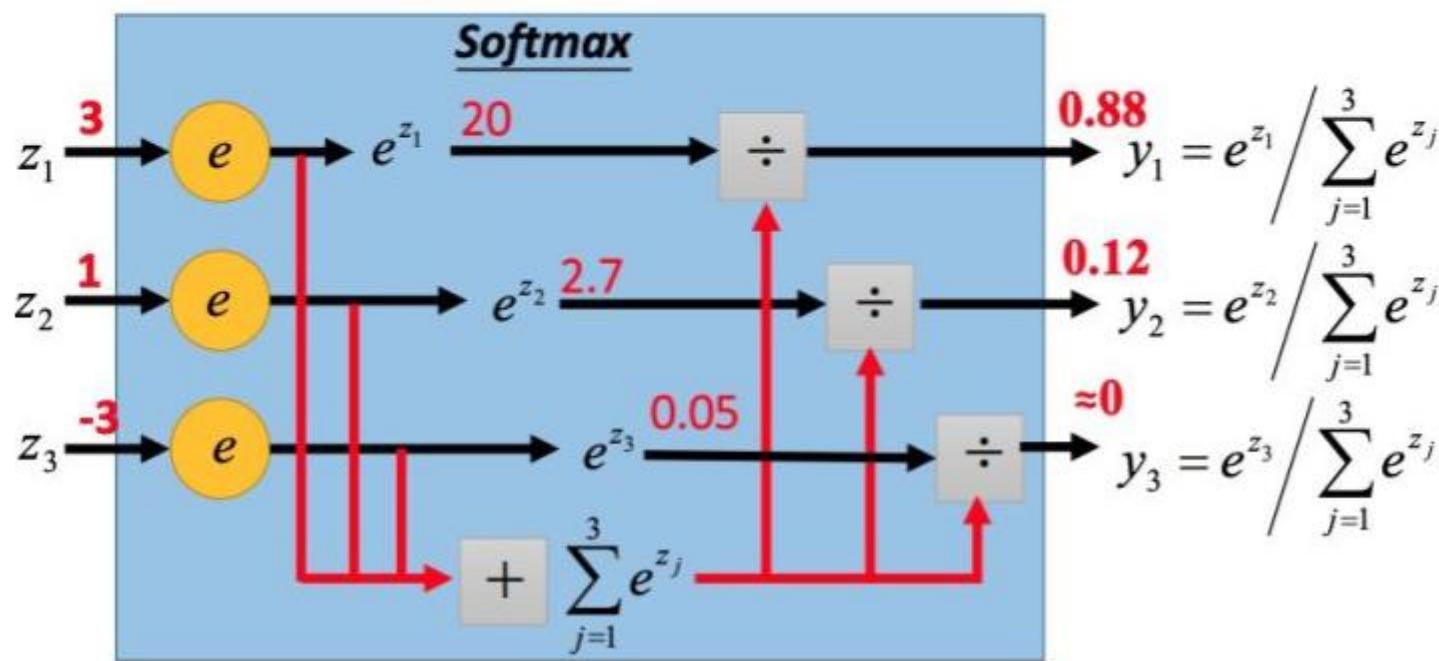
Softmax函数

$$\begin{array}{ll} C_1: w^1, b_1 & z_1 = w^1 \cdot x + b_1 \\ C_2: w^2, b_2 & z_2 = w^2 \cdot x + b_2 \\ C_3: w^3, b_3 & z_3 = w^3 \cdot x + b_3 \end{array}$$

Probability:

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

$$y_i = P(C_i | x)$$



$$f(x_i) = \text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$$

K代表输出类别总数

Softmax函数的作用：
把一堆实数的值映射到0-1区间，
并且使他们的和为1，可以理解
为对应每个类别对应的预测概率！

|小结：没有最好的激活函数，场景决定哪个合适

- Sigmoid、tanh：二分类任务输出层；模型隐藏层
- Relu、Leaky Relu：回归任务；卷积神经网络隐藏层
- Softmax：多分类任务输出层

学习链接：

https://blog.csdn.net/dfly_zx/article/details/104493048



Python3人工智能入门+实战提升：深度学习

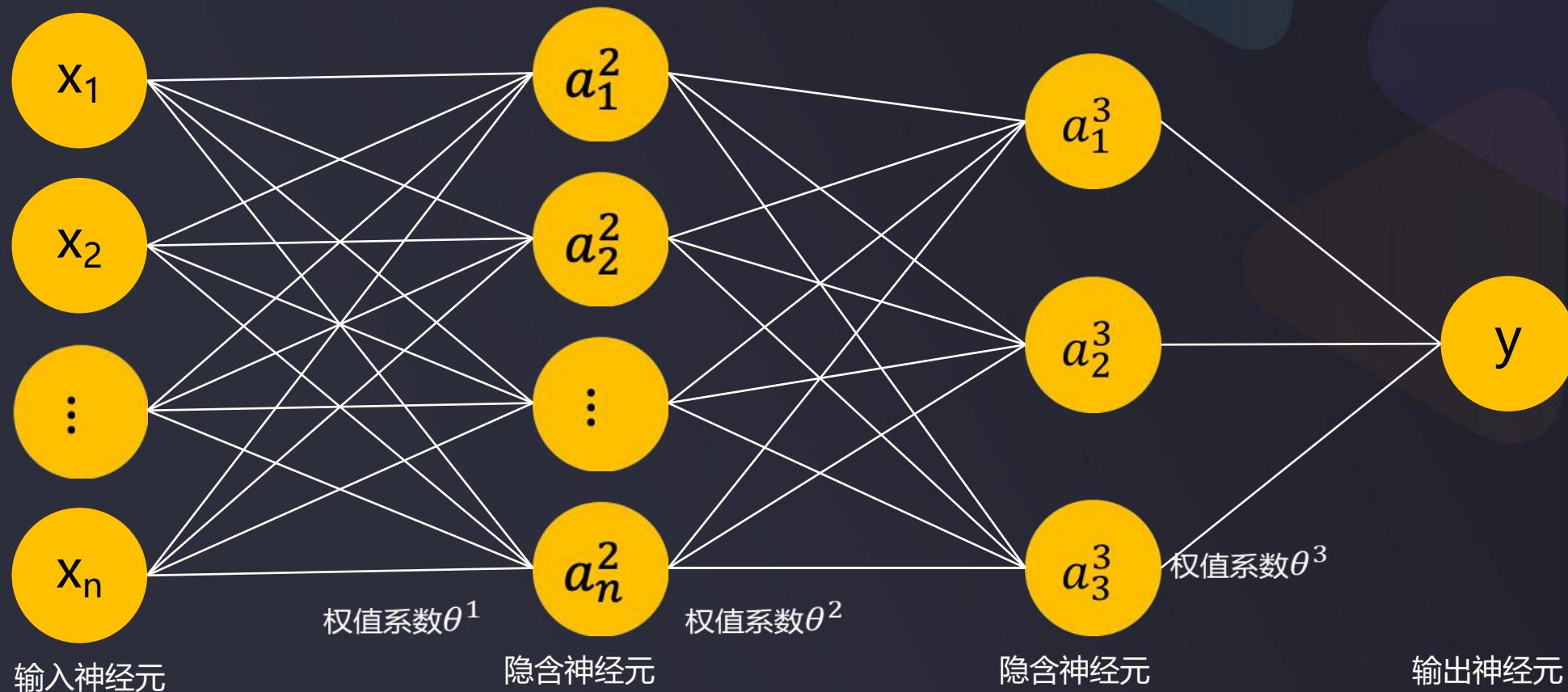
Chapter 2 深度学习之多层感知器

赵辛

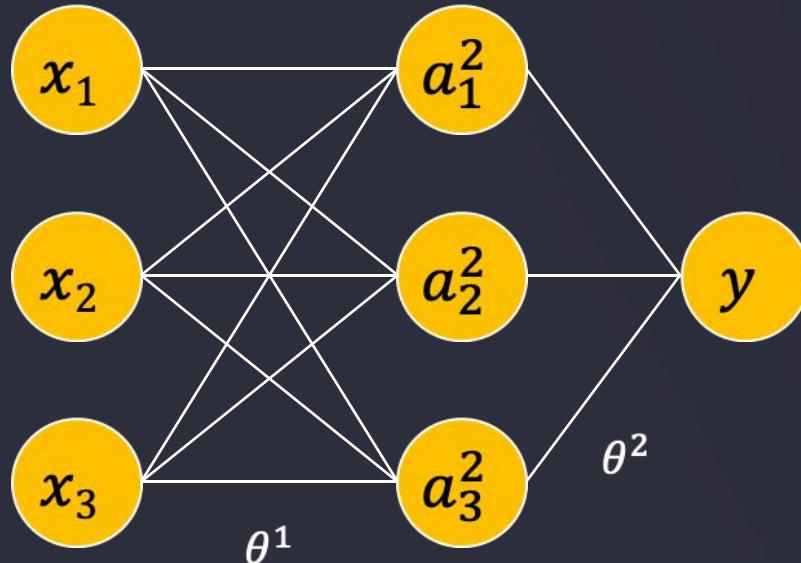
Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

多层次感知器模型框架



多层感知器正向传播 (forward propagation)



$$a_1^2 = f(\theta_{10}^1 x_0 + \theta_{11}^1 x_1 + \theta_{12}^1 x_2 + \theta_{13}^1 x_3) = f(\theta^1 x) = f(z_1^2)$$

$$a_2^2 = f(\theta_{20}^1 x_0 + \theta_{21}^1 x_1 + \theta_{22}^1 x_2 + \theta_{23}^1 x_3) = f(\theta^2 x) = f(z_2^2)$$

$$a_3^2 = f(\theta_{30}^1 x_0 + \theta_{31}^1 x_1 + \theta_{32}^1 x_2 + \theta_{33}^1 x_3) = f(\theta^3 x) = f(z_3^2)$$

$$y = f(\theta_{10}^2 a_0^2 + \theta_{11}^2 a_1^2 + \theta_{12}^2 a_2^2 + \theta_{13}^2 a_3^2) = f(\theta^2 a^2) = f(z_1^3)$$

模型的求解：寻找到合适的参数 Θ ，使总体样本预测 y 与实际 y 的误差总和最小。

损失函数：从逻辑回归到mlp

逻辑回归损失函数 (J) :

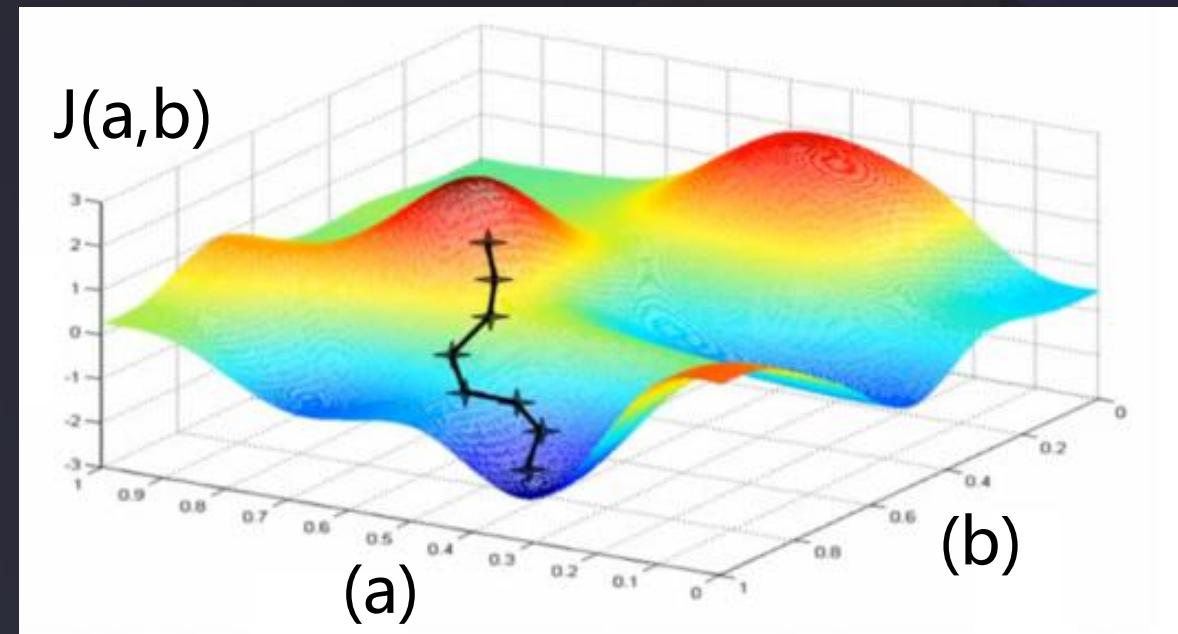
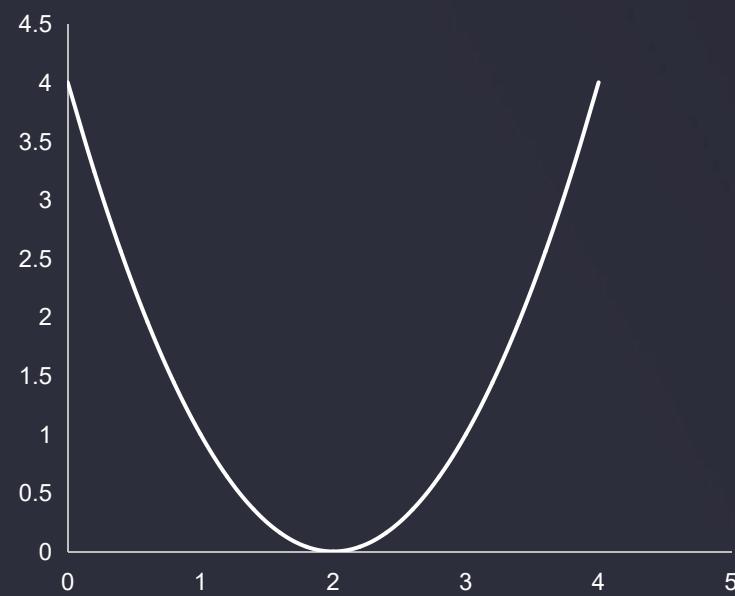
$$J = \frac{1}{m} \sum_{i=1}^m J^{(i)} = -\frac{1}{m} \left[\sum_{i=1}^m \left(y^{(i)} \log(h(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \right) \right]$$

多层感知器损失函数 (J) :

$$J = \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K J^{(i)} = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K \left(y_k^{(i)} \log(h(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h(x^{(i)}))_k) \right) \right]$$

| 梯度下降法寻找参数

$$\left\{ \begin{array}{l} temp_{\theta_j} = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ \theta_j = temp_{\theta_j} \end{array} \right\}$$



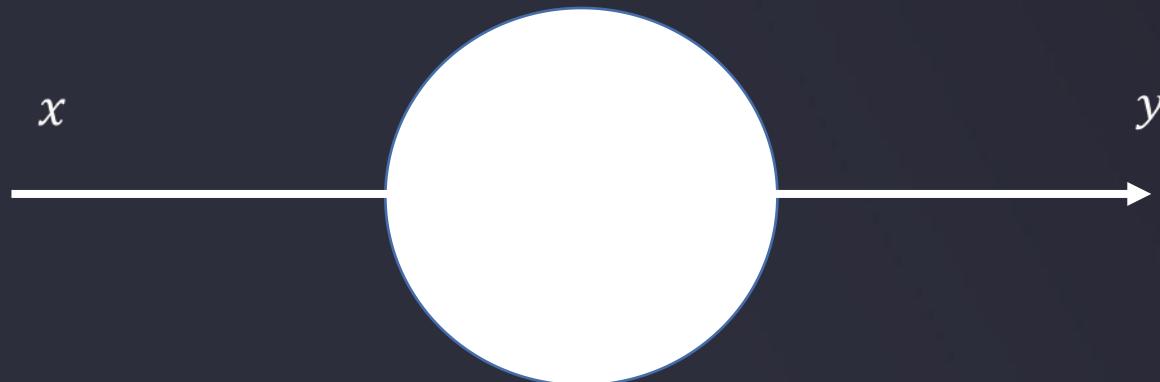
| 梯度下降法寻找参数

以一个神经元为例: $x = 3, Y = 10$

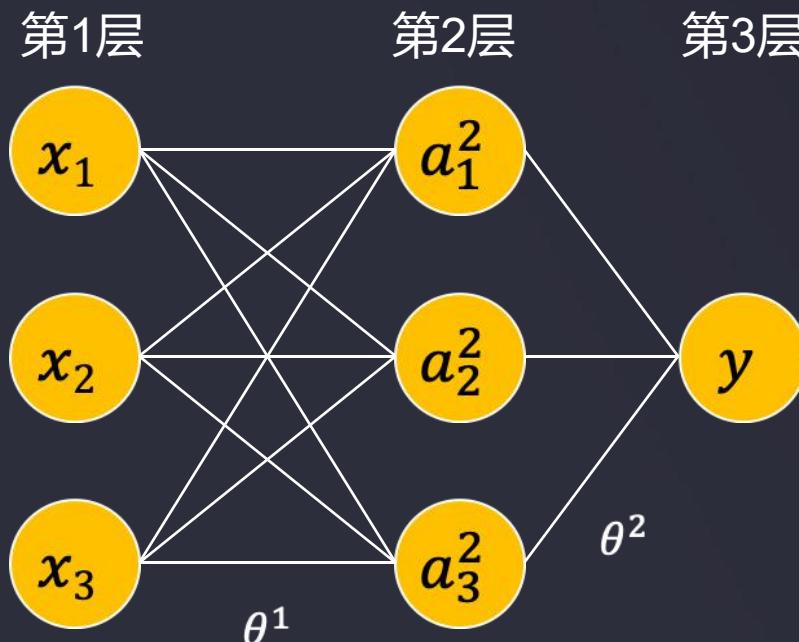
$$y = \theta_1 x + \theta_0$$

初始参数: $\theta_0 = 2, \theta_1 = 2, \alpha = 0.2$

$$J = |Y - y| = Y - (\theta_1 x + \theta_0)$$



多层感知器反向传播 (backpropagation)



δ_j^l 代表第 l 层第 j 个神经元的偏差

$$\delta_1^3 = a^3 - Y$$

$$\delta^2 = (\theta^2)^T \delta^3 \odot f'(z^3)$$

如果有更多的隐藏层: $\delta^l = (\theta^l)^T \delta^{l+1} \odot f'(z^{l+1})$

$$a_1^2 == f(\theta^1 x) = f(z_1^2)$$

$$a_2^2 == f(\theta^2 x) = f(z_2^2)$$

$$a_3^2 == f(\theta^3 x) = f(z_3^2)$$

$$y = a^3 = f(\theta^2 a^2) = f(z_1^3)$$

从后往前依次计算每层神经元的数值偏差，然后通过梯度下降法
寻找到使偏差最小的参数 θ ，完成模型求解



Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

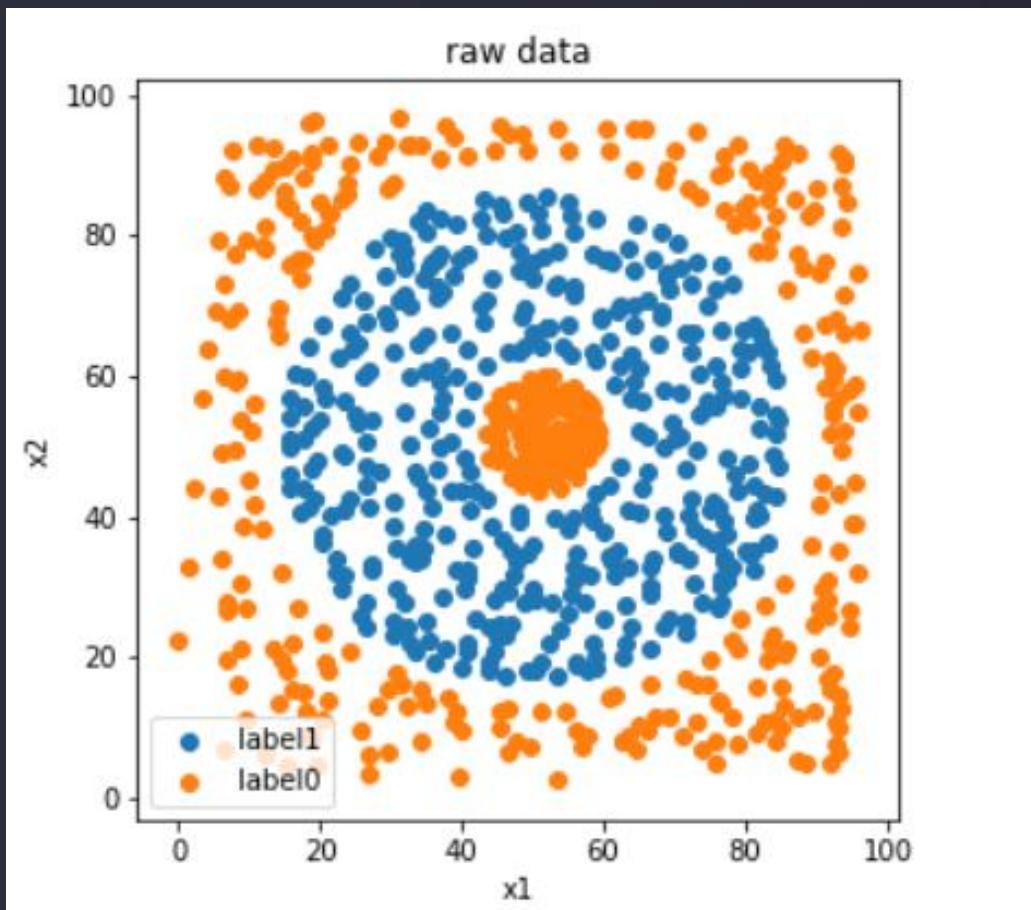
赵辛

Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

任务一：MLP快速搭建非线性二分类模型

基于task1_data数据，建立mlp模型，实现非线性边界二分类。



- 1、数据分离: `test_size=0.2, random_state=0;`
- 2、建模并训练模型（迭代1000次），计算训练集、测试集准确率；
- 3、可视化预测结果
- 4、继续迭代6000次，重复步骤2-3
- 5、迭代1-10000次（500为间隔），查看迭代过程中的变化(可视化结果、准确率)

模型结构：一层隐藏层，25个神经元，激活函数：
sigmoid

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#数据加载  
import pandas as pd  
import numpy as np  
data = pd.read_csv('task1_data.csv')  
data.head()
```

	x1	x2	y
0	46.4663	63.4143	1
1	43.5724	61.4890	1
2	41.4073	57.8710	1
3	40.4504	54.0086	1
4	37.3039	56.6732	1

MLP快速搭建非线性二分类模型

数据加载及展示

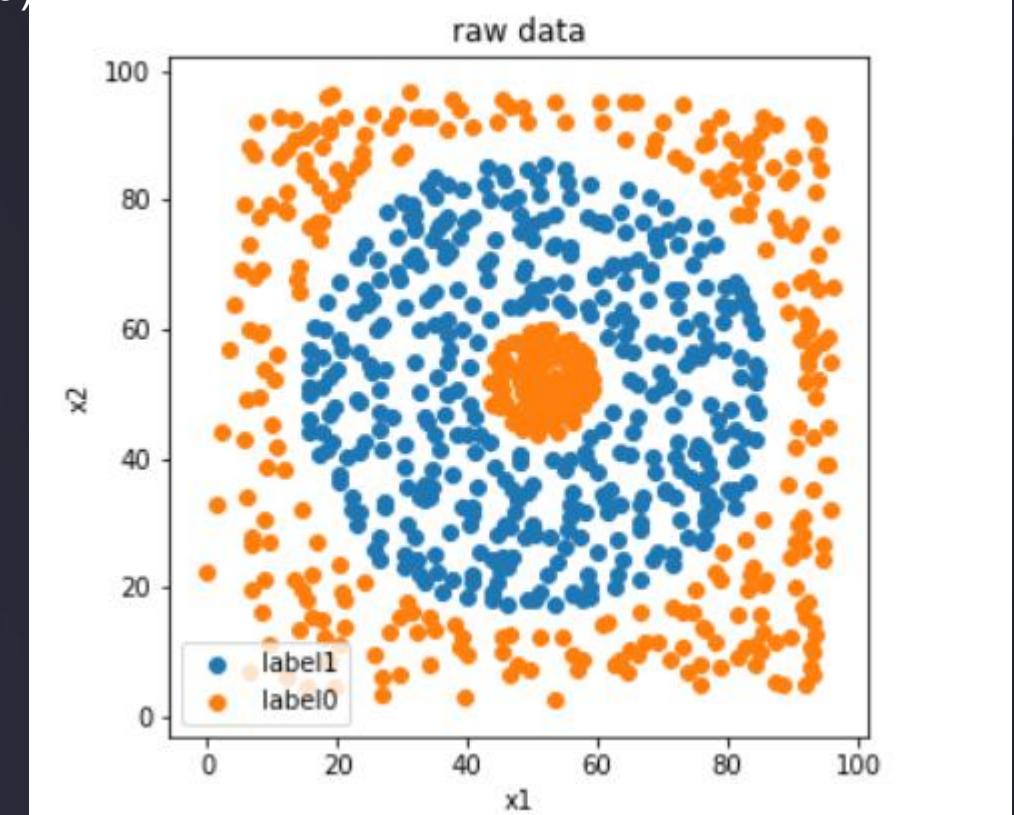
数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#数据可视化  
from matplotlib import pyplot as plt  
fig1 = plt.figure(figsize=(5,5))  
label1=plt.scatter(X.loc[:, 'x1'][y==1],X.loc[:, 'x2'][y==1])  
label0=plt.scatter(X.loc[:, 'x1'][y==0],X.loc[:, 'x2'][y==0])  
plt.legend((label1,label0),('label1','label0'))  
plt.xlabel('x1')  
plt.ylabel('x2')  
plt.title('raw data')  
plt.show()
```



MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#x y赋值

```
X = data.drop(['y'],axis=1)  
y = data.loc[:, 'y']
```

#数据分离

```
from sklearn.model_selection import train_test_split  
X_train,X_test,y_train,y_test =  
train_test_split(X,y,test_size=0.2,random_state=0)  
print(X_train.shape,X_test.shape,X.shape)
```

(630, 2) (158, 2) (788, 2)

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#模型建立

```
from keras.models import Sequential  
from keras.layers import Dense, Activation
```

```
mlp = Sequential()
```

```
mlp.add(Dense(units=25, input_dim=2, activation='sigmoid'))  
mlp.add(Dense(units=1,activation='sigmoid'))  
mlp.summary()
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
<hr/>		
dense_11 (Dense)	(None, 25)	75
dense_12 (Dense)	(None, 1)	26
<hr/>		
Total params: 101		
Trainable params: 101		
Non-trainable params: 0		
<hr/>		

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#模型求解参数配置

```
mlp.compile(optimizer='adam',loss='binary_crossentropy')
```

#模型训练

```
mlp.fit(X_train,y_train,epochs=1000)
```

```
Epoch 1/1000  
630/630 [=====] - 0s 235us/step - loss: 0.7799  
Epoch 2/1000  
630/630 [=====] - 0s 63us/step - loss: 0.7366  
Epoch 3/1000  
630/630 [=====] - 0s 59us/step - loss: 0.7059  
Epoch 4/1000  
630/630 [=====] - 0s 57us/step - loss: 0.6775  
Epoch 5/1000  
630/630 [=====] - 0s 60us/step - loss: 0.6477  
Epoch 6/1000  
630/630 [=====] - 0s 57us/step - loss: 0.6293  
Epoch 7/1000  
630/630 [=====] - 0s 57us/step - loss: 0.6110  
Epoch 8/1000  
630/630 [=====] - 0s 57us/step - loss: 0.5953
```

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测及表现评估

结果展示

```
#训练数据预测与评估  
y_train_predict = mlp.predict_classes(X_train)  
from sklearn.metrics import accuracy_score  
accuracy_train =  
accuracy_score(y_train,y_train_predict)  
print(accuracy_train)
```

0.8380952380952381

```
#测试数据预测与评估  
y_test_predict = mlp.predict_classes(X_test)  
accuracy_test =  
accuracy_score(y_test,y_test_predict)  
print(accuracy_test)
```

0.7848101265822784

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

#生成2D数据集

```
xx, yy = np.meshgrid(np.arange(0,100,1),np.arange(0,100,1))  
x_range = np.c_[xx.ravel(),yy.ravel()]  
print(x_range)
```



```
[[ 0  0]  
 [ 1  0]  
 [ 2  0]  
 ...  
 [97 99]  
 [98 99]  
 [99 99]]
```

MLP快速搭建非线性二分类模型

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

```
#生成数据结果预测  
y_range_predict = mlp.predict_classes(x_range)  
print(type(y_range_predict))  
print(y_range_predict.shape)
```

```
<class 'numpy.ndarray'>  
(10000, 1)
```

```
#格式转化  
y_range_predict_form = pd.Series(i[0] for i in  
y_range_predict)  
print(type(y_range_predict_form))  
print(y_range_predict_form)
```

```
<class 'pandas.core.series.Series'>  
0      0  
1      0  
2      0  
3      0
```

数据加载及展示

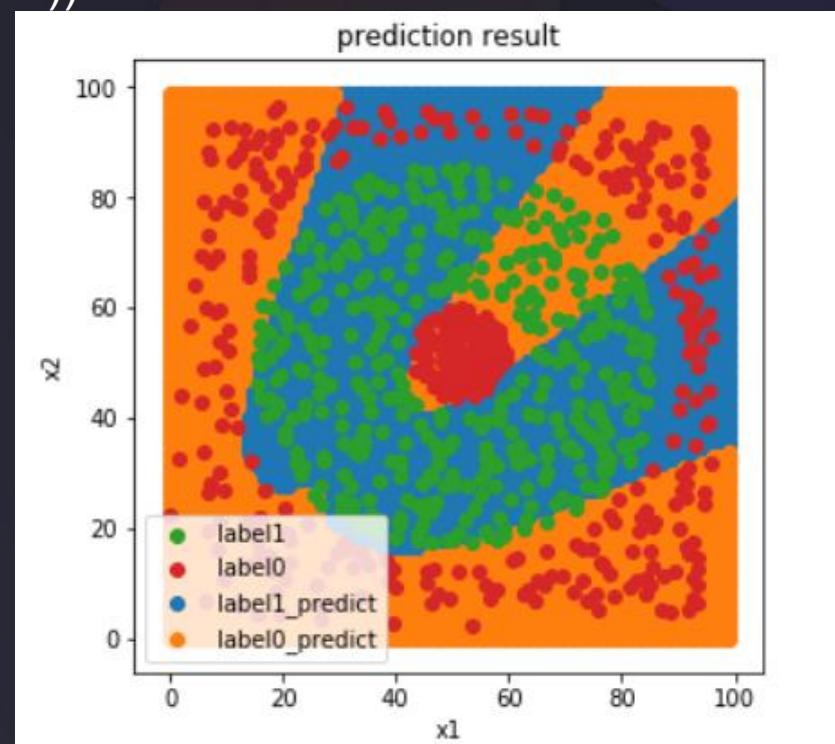
数据预处理

模型建立及训练

模型预测

结果展示

```
#结果可视化  
fig2 = plt.figure(figsize=(5,5))  
label1_predict=plt.scatter(x_range[:,0][y_range_predict_form==1],  
x_range[:,1][y_range_predict_form==1])  
label0_predict=plt.scatter(x_range[:,0][y_range_predict_form==0],  
x_range[:,1][y_range_predict_form==0])  
  
label1=plt.scatter(X.loc[:, 'x1'][y==1],X.loc[:, 'x2'][y==1])  
label0=plt.scatter(X.loc[:, 'x1'][y==0],X.loc[:, 'x2'][y==0])  
plt.legend((label1,label0,label1_predict,label0_predict),('label1','la  
bel0','label1_predict','label0_predict'))  
plt.xlabel('x1')  
plt.ylabel('x2')  
plt.title('prediction result')  
plt.show()
```



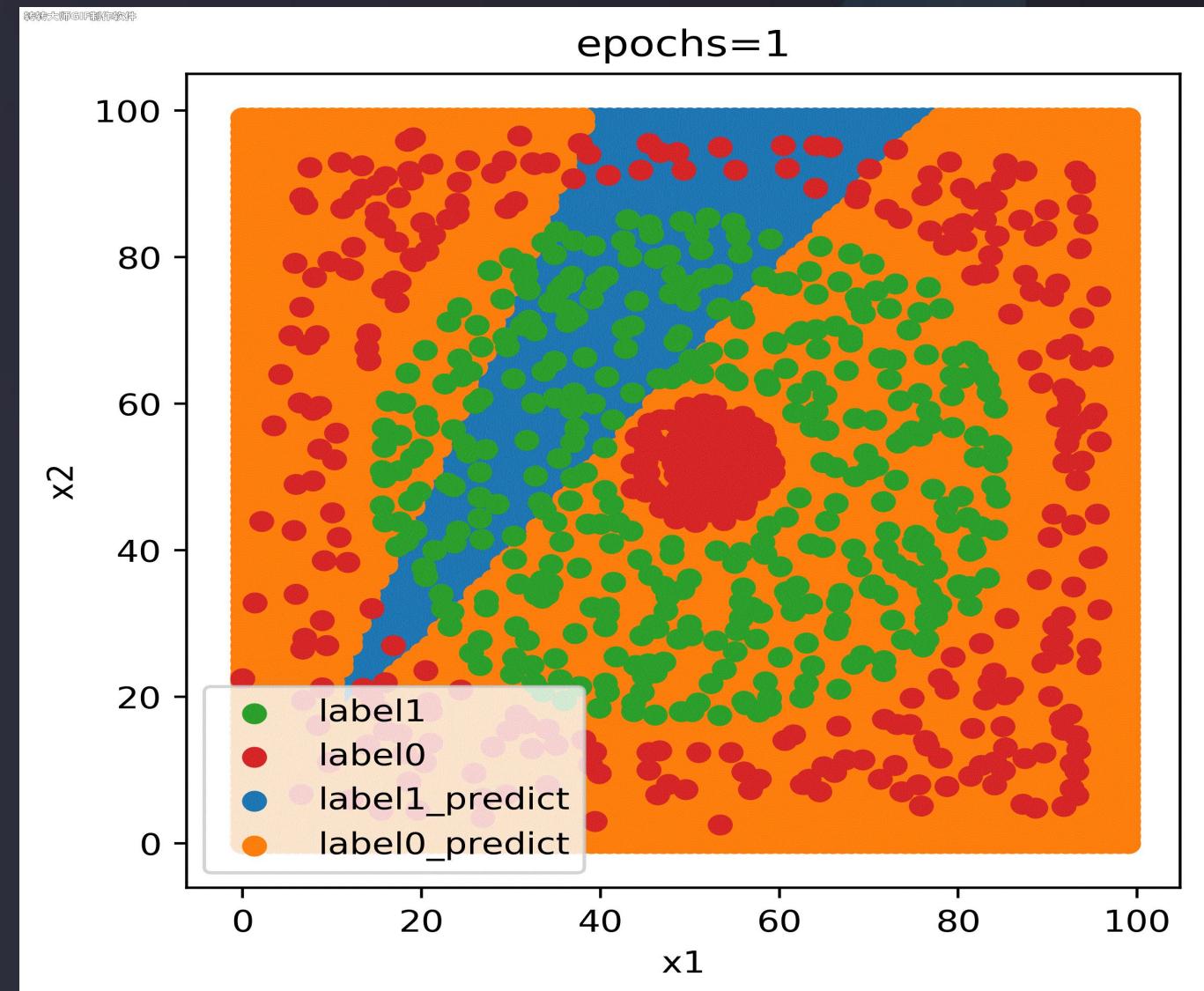
查看模型迭代过程中的结果变化



mlp_batch_code.ipynb

```
: #逐步迭代及结果可视化
accuracy_train = []
accuracy_test = []
for i in range(0,21):
    #train the model
    if i == 0:
        mlp.fit(X_train,y_train,epochs=1)
    else:
        mlp.fit(X_train,y_train,epochs=500)
    #make prediction and calculate the accuracy
    y_train_predict = mlp.predict_classes(X_train)
    accuracy_train_i = accuracy_score(y_train,y_train_predict)
    #make prediction based on the test data
    y_test_predict = mlp.predict_classes(X_test)
    accuracy_test_i = accuracy_score(y_test,y_test_predict)
    accuracy_train.append(accuracy_train_i)
    accuracy_test.append(accuracy_test_i)
```

MLP快速搭建非线性二分类模型



#准确率变化

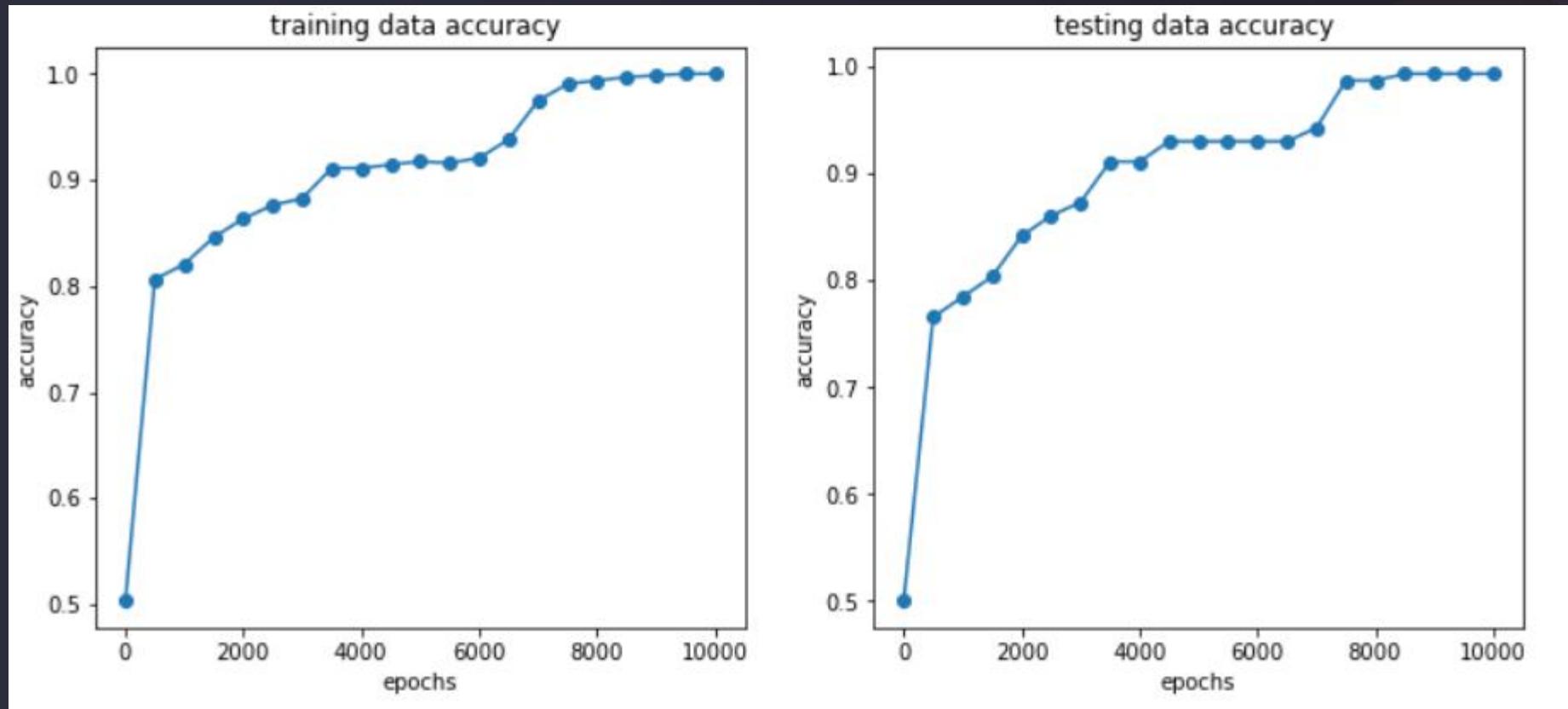
```
n = [1+i*500 for i in range(0,21)]
```

#k值变化对准确率影响的结果可视化

```
fig10 = plt.figure(figsize=(12,5))
```

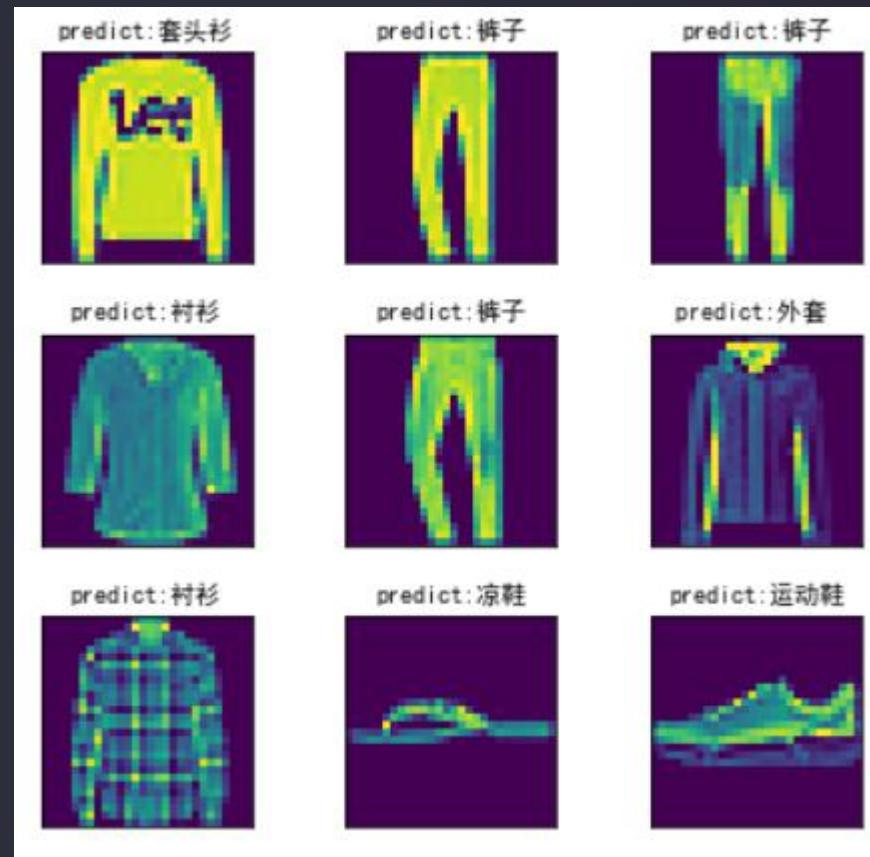
```
plt.subplot(121)
```

```
plt.plot(n,accuracy_train,marker='o')
```



任务二：Fashion_mnist服饰预测

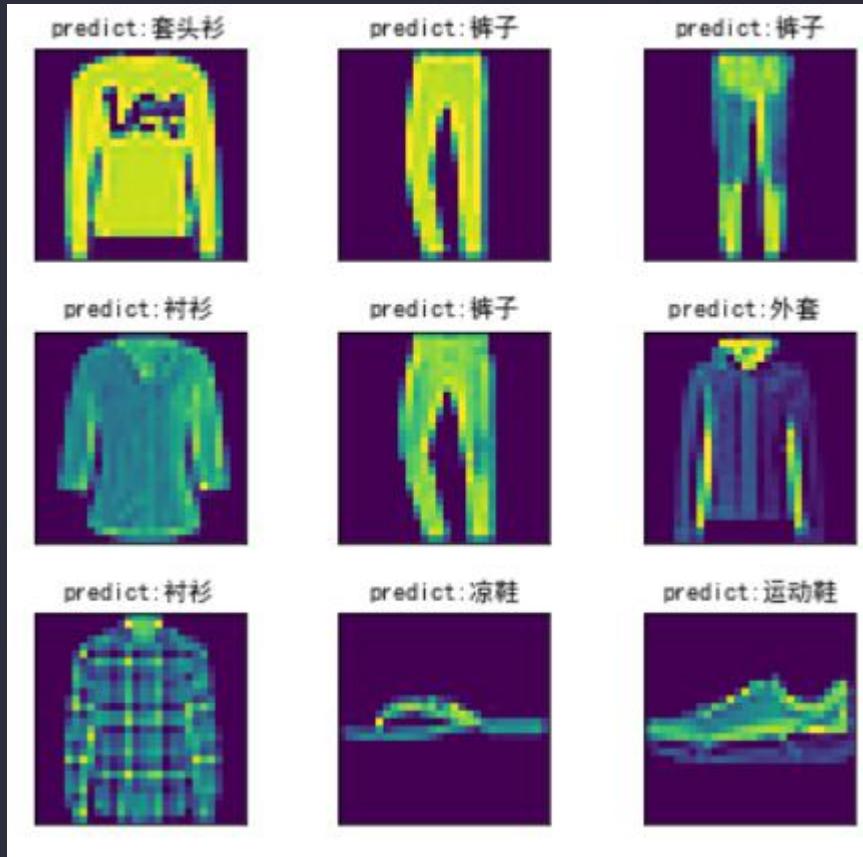
基于fashion_mnist数据集，建立mlp模型，实现服饰图片十分类



- 1、实现图像数据加载、可视化
- 2、进行数据预处理：维度转化，归一化、输出结果格式转化
- 3、建立mlp模型，进行模型训练与预测，计算模型在训练、测试数据集的准确率
- 4、选取一个测试样本，预测其类别
- 5、选取测试集前10个样本，分别预测其类别

模型结构：两层隐藏层（激活函数：relu），分别有392、196个神经元；输出层10类，激活函数softmax

Fashion_mnist数据集



一个替代MNIST手写数字集的图像数据集，涵盖了来自10种类别的共7万个不同服饰商品的正面图片，由60000个训练样本和10000个测试样本组成，每个样本都是一张 $28 * 28$ 像素的灰度图片。

- 官方网站：
<https://github.com/zalandoresearch/fashion-mnist>
- 一共4个文件，训练集、训练集标签、测试集、测试集标签

Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#图像加载与展示

```
from keras.datasets import fashion_mnist  
(X_train, y_train), (X_test, y_test) =  
fashion_mnist.load_data()  
print(type(X_train),'\n',X_train.shape)
```

```
<class 'numpy.ndarray'>  
(60000, 28, 28)
```

|Fashion_mnist服饰预测

数据加载及展示

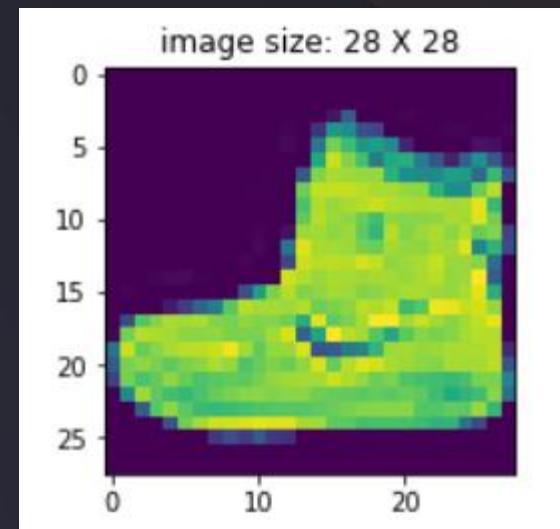
数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#样本可视化  
img1 = X_train[0]  
%matplotlib inline  
from matplotlib import pyplot as plt  
fig1 = plt.figure(figsize=(3,3))  
plt.imshow(img1)  
plt.title('image size: 28 X 28')  
plt.show()
```



Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#输入数据维度转化  
feature_size = img1.shape[0]*img1.shape[1]  
X_train_format = X_train.reshape(X_train.shape[0],feature_size)  
X_test_format = X_test.reshape(X_test.shape[0],feature_size)  
  
print(X_train.shape)  
print(X_train_format.shape)
```

(60000, 28, 28)
(60000, 784)

Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#数据归一化处理  
X_train_normal = X_train_format/255  
X_test_normal = X_test_format/255  
print(X_train_normal[0])
```

0	1	1	0	0	0	0
36	136	127	62	54	0	0
0	0	0	0	0	0	0
0	0	0	12	10	0	0
0	0	155	236	207	178	107
0	0	0	0	0	0	0
127	121	122	146	141	88	172
1	1	0	200	232	232	233
0	0	0	0	0	0	0

0.	0.	0.	0.
0.00392157	0.	0.	0.0
0.	0.00392157	0.01568627	0.
0.	0.00392157	0.00392157	0.
0.	0.	0.	0.
0.	0.	0.	0.
0.14117647	0.53333333	0.49803922	0.2
0.	0.	0.00392157	0.0
0.	0.01176471	0.	0.

Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#格式化输出结果(labels)
from keras.utils import to_categorical
y_train_format = to_categorical(y_train)
y_test_format = to_categorical(y_test)
print(y_train[0])
print(y_train_format[0])
```

```
9
[0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

```
print(y_train.shape,y_train_format.shape)
```

```
(60000,) (60000, 10)
```

Fashion_mnist服饰预测

#模型建立

```
from keras.models import Sequential  
from keras.layers import Dense, Activation
```

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
mlp = Sequential()  
mlp.add(Dense(units=392,activation='relu',input_dim=784))  
mlp.add(Dense(units=196,activation='relu'))  
mlp.add(Dense(units=10,activation='softmax'))  
mlp.summary()
```

Model: "sequential_1"		
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 392)	307720
dense_2 (Dense)	(None, 196)	77028
dense_3 (Dense)	(None, 10)	1970
=====		
Total params: 386,718		
Trainable params: 386,718		
Non-trainable params: 0		

Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#参数配置

```
mlp.compile(loss='categorical_crossentropy',optimizer='adam',  
metrics=['categorical_accuracy'])
```

#模型训练

```
mlp.fit(X_train_normal,y_train_format,epochs=10)
```

```
Epoch 1/10  
60000/60000 [=====] - 17s 285us/step - loss: 0.4682 - categorical_accuracy:  
0.8297  
Epoch 2/10  
60000/60000 [=====] - 18s 300us/step - loss: 0.3550 - categorical_accuracy:  
0.8699  
Epoch 3/10  
60000/60000 [=====] - 20s 337us/step - loss: 0.3192 - categorical_accuracy:  
0.8813  
Epoch 4/10  
60000/60000 [=====] - 22s 363us/step - loss: 0.2978 - categorical_accuracy:  
0.8903
```

|Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#结果预测  
y_train_predict =  
mlp.predict_classes(X_train_normal)  
print(type(y_train_predict))  
print(y_train_predict[0:10])
```

```
<class 'numpy.ndarray'>  
[9 0 6 3 3 2 7 4 5 5]
```

Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#训练数据集预测准确率  
from sklearn.metrics import accuracy_score  
accuracy_train =  
accuracy_score(y_train,y_train_predict)  
print(accuracy_train)
```

0.9275

```
#测试数据集预测准确率  
y_test_predict = mlp.predict_classes(X_test_normal)  
accuracy_test =  
accuracy_score(y_test,y_test_predict)  
print(accuracy_test)
```

0.8913

|Fashion_mnist服饰预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#创建结果对应标签字典

```
label_dic = {0:'T恤',1:'裤子',2:'套头衫',3:'裙子',4:'外套',5:'凉鞋',6:'衬衫',7:'运动鞋',8:'包',9:'踝靴'}  
print(label_dic)
```

```
{0: 'T恤', 1: '裤子', 2: '套头衫', 3: '裙子', 4:  
'外套', 5: '凉鞋', 6: '衬衫', 7: '运动鞋', 8: '包'  
, 9: '踝靴'}
```

Fashion_mnist服饰预测

数据加载及展示

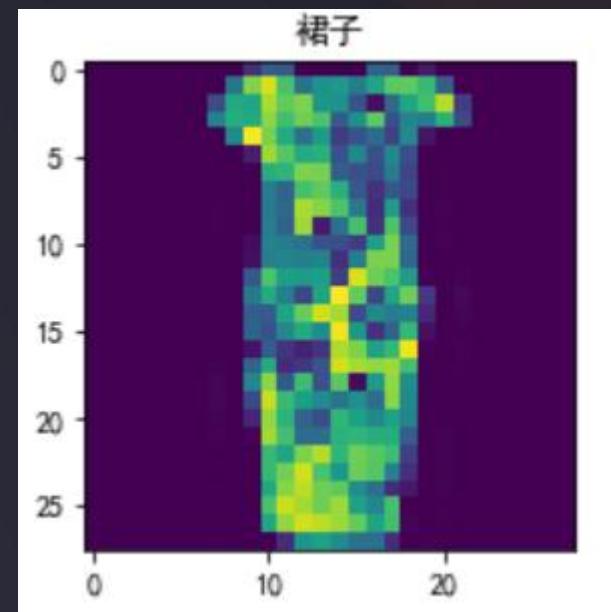
数据预处理

模型建立及训练

模型预测

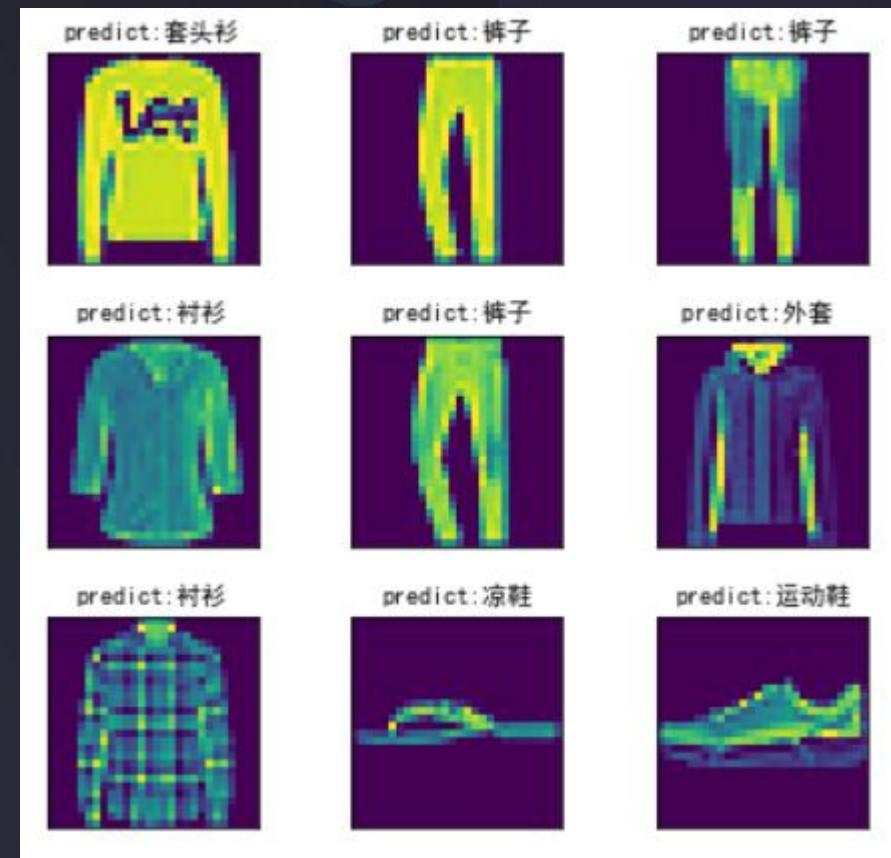
结果展示及表现评估

```
#结果可视化  
img2 = X_test[100]  
fig2 = plt.figure(figsize=(3,3))  
plt.imshow(img2)  
plt.title(label_dic[y_test_predict[100]])  
plt.show()
```



|Fashion_mnist服饰预测

```
#可视化前九张图的预测结果  
a = [i for i in range(1,10)]  
fig4 = plt.figure(figsize=(5,5))  
for i in a:  
    plt.subplot(3,3,i)  
    plt.tight_layout()  
    plt.imshow(X_test[i])  
  
    plt.title('predict:{}'.format(label_dic[y_test_predict[i]]),font2)  
    plt.xticks([])  
    plt.yticks([])
```





Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

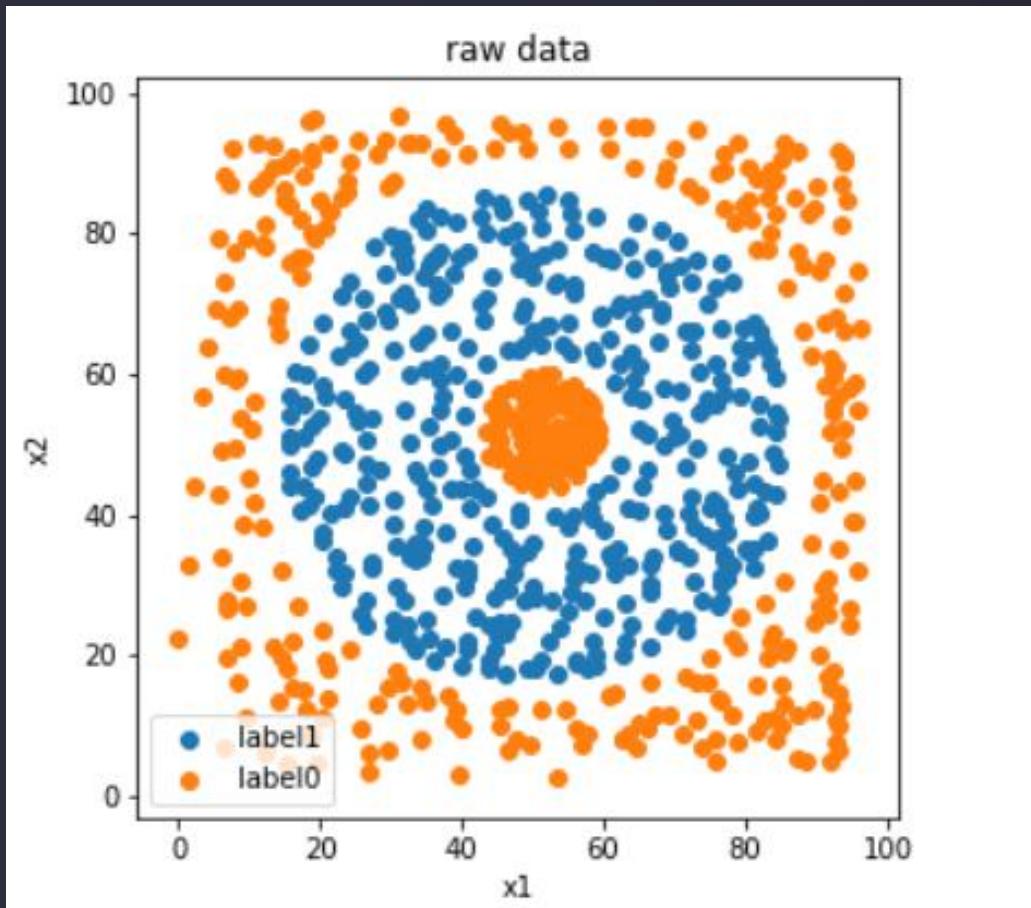
赵辛

Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

任务一：MLP快速搭建非线性二分类模型

基于task1_data数据，建立mlp模型，实现非线性边界二分类。



- 1、数据分离:`test_size=0.2,`
`random_state=0;`
- 2、建模并训练模型（迭代1000次），计算训练集、测试集准确率；
- 3、可视化预测结果
- 4、继续迭代6000次，重复步骤2-3
- 5、迭代1-10000次（500为间隔），查看迭代过程中的变化(可视化结果、准确率)

模型结构：一层隐藏层，25个神经元，激活函数：
`sigmoid`



Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

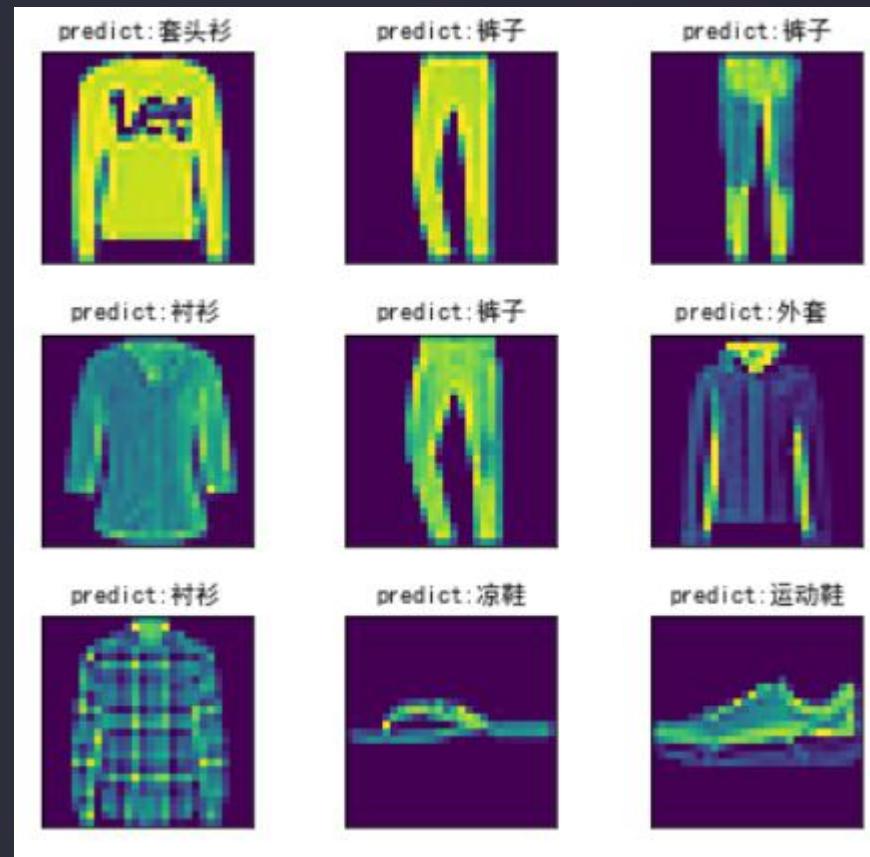
赵辛

Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

任务二：Fashion_mnist服饰分类

基于fashion_mnist数据集，建立mlp模型，实现服饰图片十分类



- 1、实现图像数据加载、可视化
- 2、进行数据预处理：维度转化，归一化、输出结果格式转化
- 3、建立mlp模型，进行模型训练与预测，计算模型在训练、测试数据集的准确率
- 4、选取一个测试样本，预测其类别
- 5、选取测试集前10个样本，分别预测其类别

模型结构：两层隐藏层（激活函数：relu），分别有392、196个神经元；输出层10类，激活函数softmax



Python3人工智能入门+实战提升：深度学习

Chapter 2 深度学习之多层感知器

赵辛

Chapter 2 深度学习之多层感知器

-
- 1 --逻辑回归实现非线性边界分类
 - 2 --逻辑回归到多层感知器MLP
 - 3 --MLP实现非线性分类
 - 4 --MLP实现多分类
 - 5 --激活函数
 - 6 --MLP损失函数与反向传播算法
 - 7 --实战准备
 - 8 --实战(一)MLP实现非线性边界数据分类
 - 9 --实战(二)Fashion_mnist服饰预测

| 玩转谷歌Playground

<https://playground.tensorflow.org/>



Python3人工智能入门+实战提升：深度学习

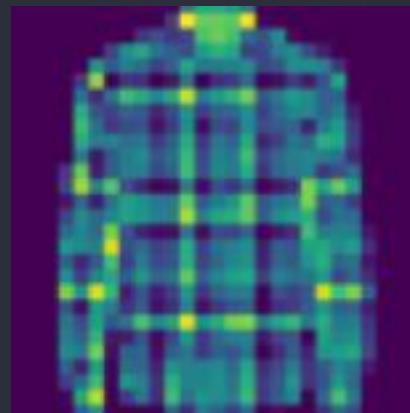
Chapter 3 卷积神经网络CNN

赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

逻辑回归实现图像分类



Size: 28*28

输入图片 (28 X 28 像素)



28行

[[160 164 169 ... 185 184 183]
[126 131 136 ... 184 183 182]
[127 132 137 ... 183 182 181]
...
[198 193 178 ... 206 195 174]
[176 183 175 ... 188 144 125]
[190. 190 157 ... 200 145 144]]

28列

如果使用逻辑回归算法完成该分类任务，对于有 $28 \times 28 = 784$ 个特征的样本，其生成的三次多项式特征项总数高达：

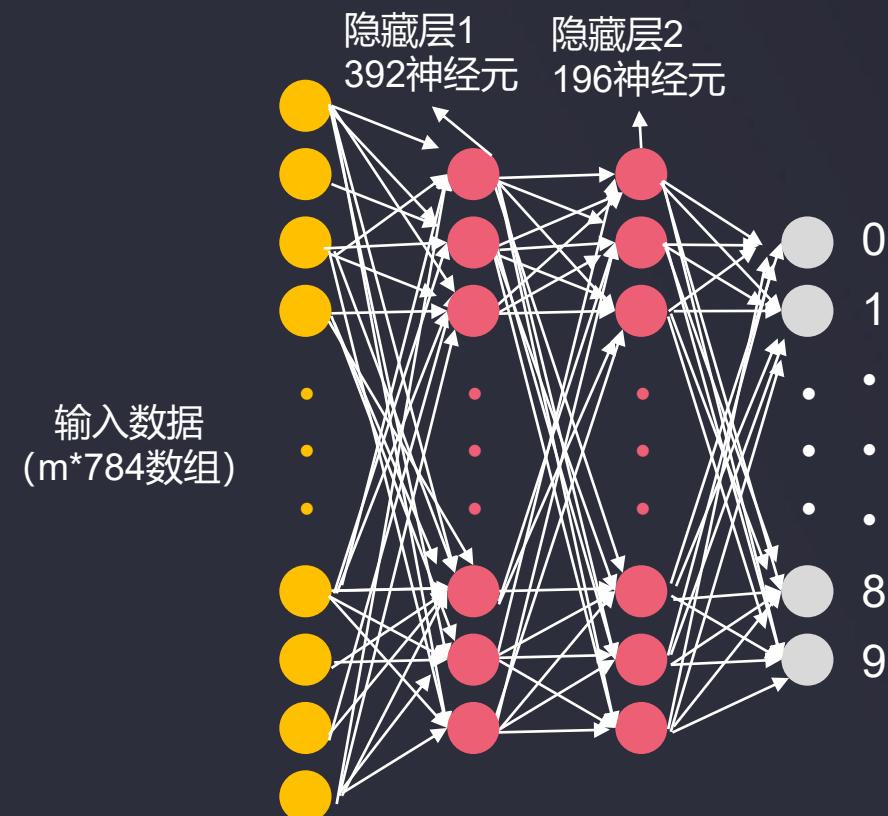
80,931,145

```
from sklearn.preprocessing import PolynomialFeatures  
poly = PolynomialFeatures(degree=3)  
fig_poly = poly.fit_transform(fig)  
print(fig.shape)  
print(fig_poly.shape)
```

(1, 784)

(1, 80931145)

神经网络实现图像分类



Fashion_mnist识别mlp模型参数概览

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 392)	307720
dense_2 (Dense)	(None, 196)	77028
dense_3 (Dense)	(None, 10)	1970
Total params: 386,718 Trainable params: 386,718 Non-trainable params: 0		

mlp神经网络模型 (两层隐藏层, 分别有392、196神经元)
待训练参数数量:

386,718

$$\frac{386718}{80931145} = 0.0047$$

神经网络实现图像分类



输入图片 (400 X 500 像素)

如果图片尺寸更大、细节信息更多、类别增加，
怎么办？！

400 X 500 像素图片识别mlp模型参数概览

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 392)	78400392
dense_5 (Dense)	(None, 196)	77028
dense_6 (Dense)	(None, 10)	1970

Total params: 78,479,390

Trainable params: 78,479,390

Non-trainable params: 0

办法：先提取出图像中的关键信息(轮廓)，
再建立mlp模型。

| 图像卷积运算

图像的卷积（convolution）运算，即通过对图像矩阵与滤波器矩阵进行对应相乘再求和运算，可实现图像中特定轮廓特征的快速搜索。

A与B的卷积通常表示为：

$$A * B \text{ 或 } convolution(A, B)$$

图像卷积运算

图像的卷积 (convolution) 运算，即通过对图像矩阵与过滤器矩阵进行对应相乘再求和运算，可实现图像中特定轮廓特征的快速定位

A 为图像矩阵: $A = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}$, F 为滤波器矩阵: $F = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$, 对其进行卷积运算:

图像卷积运算

图像的卷积 (convolution) 运算，即通过对图像矩阵与过滤器矩阵进行对应相乘再求和运算，可实现图像中特定轮廓特征的快速定位

A 为图像矩阵: $A = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}$, F 为滤波器矩阵: $F = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$, 对其进行卷积运算:

$$A * F = \begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \begin{bmatrix} X_{11}w_{11} + X_{12}w_{12} & X_{12}w_{11} + X_{13}w_{12} \\ +X_{21}w_{21} + X_{22}w_{22} & +X_{22}w_{21} + X_{23}w_{22} \\ X_{21}w_{11} + X_{22}w_{12} & X_{22}w_{11} + X_{23}w_{12} \\ +X_{31}w_{21} + X_{32}w_{22} & +X_{32}w_{21} + X_{33}w_{22} \end{bmatrix}$$

| 图像卷积运算：寻找竖向轮廓

50	50	50	0	0	0
50	50	50	0	0	0
50	50	50	0	0	0
50	50	50	0	0	0
50	50	50	0	0	0
50	50	50	0	0	0

* 竖向轮廓过滤器 =

1	0	-1
1	0	-1
1	0	-1

0	150	150	0
0	150	150	0
0	150	150	0
0	150	150	0

包含竖向轮廓的区域非常亮（灰度值高）

| 图像卷积运算：寻找横向轮廓

50	50	50	50	50	50
50	50	50	50	50	50
50	50	50	50	50	50
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

* 横向轮廓过滤器 =

1	1	1
0	0	0
-1	-1	-1

0	0	0	0
150	150	150	150
150	150	150	150
0	0	0	0

包含横向轮廓的区域非常亮（灰度值高）

将图片与轮廓滤波器进行卷积运算，
可快速定位特定轮廓特征

卷积神经网络的核心：寻找合适的轮廓过滤器

计算机根据训练图片即及其类别，自动寻找合适的轮廓过滤器，再使用该过滤器去寻找图像轮廓用于判断新图片所属类别

轮廓过滤器

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

最小化输出损失函数的过程，也是寻找合适
 W 的过程，即寻找合适的过滤器

竖向轮廓过滤器

1	0	-1
1	0	-1
1	0	-1

横向轮廓过滤器

1	1	1
0	0	0
-1	-1	-1

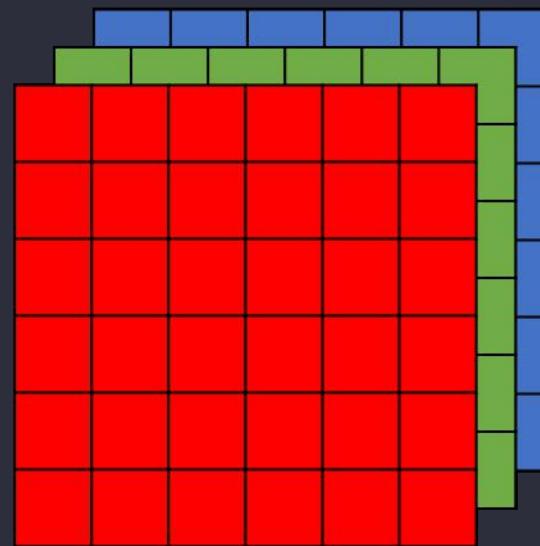
sobel过滤器

-1	0	1
-2	0	2
-1	0	1

一张复杂的图像包含很多不同的轮廓，
因此过滤器数量也不止一个

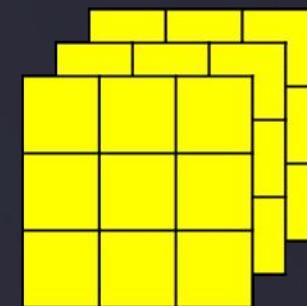
彩色图的卷积运算

RGB图像的卷积：对R\G\B三个通道分别求卷积再相加



6*6*3RGB图矩阵

*

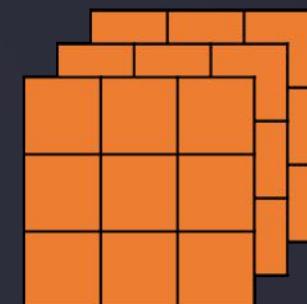


3*3*3过滤器1

=

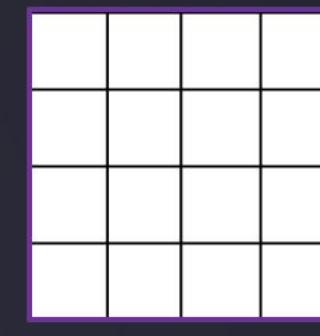


4*4矩阵

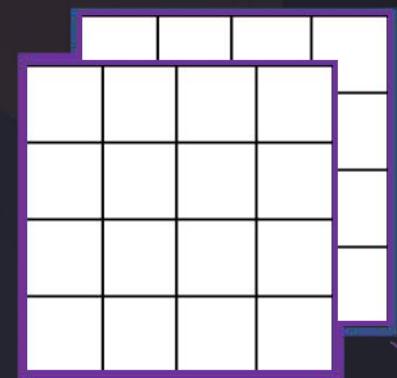


3*3*3过滤器2

=



4*4矩阵



4*4*2矩阵

红绿蓝三通道图片，经过一个过滤器的卷积运算后，变成一个通道！

知识巩固

问题：计算以下图像矩阵经过卷积运算后，得到的新矩阵

60	60	0	0	0	0
60	60	0	0	0	0
60	60	0	0	0	0
60	60	60	60	0	0
60	60	60	60	0	0
60	60	60	60	0	0

*

横向轮廓过滤器

1	1	1
0	0	0
-1	-1	-1



Python3人工智能入门+实战提升：深度学习

Chapter 3 卷积神经网络CNN

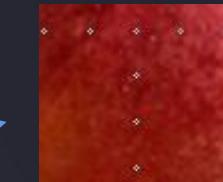
赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(一) VGG16+mlp高准确率识别猫狗

| 现实问题思考

一张图片中很多信息是不重要甚至重复多余的，使用所有像素点数据会增加运算量、可能导致过拟合、降低模型的容错性



思考：能否采用一定方法对信息进行有效的压缩？

池化 (Pooling)

池化：也称为欠采样或下采样，指按照一定的规则对图像矩阵进行处理，实现信息压缩与数据降维

作用：特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性

最大法池化 (Max-pooling)

20	5	2	10
3	10	7	1
8	1	6	9
4	7	3	5

pooling_size: (2,2)
stride: (2,2)

The diagram illustrates the Max-pooling process. A 4x4 input matrix is shown on the left, with values: Row 1: 20, 5, 2, 10; Row 2: 3, 10, 7, 1; Row 3: 8, 1, 6, 9; Row 4: 4, 7, 3, 5. A 2x2 pooling window is applied with a stride of 2. The output matrix on the right has values: Top-left: 20 (max of 20, 5), Top-right: 10 (max of 2, 10), Bottom-left: 8 (max of 8, 1), Bottom-right: 9 (max of 6, 9).

20	10
8	9

Stride为窗口滑动步长，用于池化、卷积的计算中。

池化 (Pooling)

池化：也称为欠采样或下采样，指按照一定的规则对图像矩阵进行处理，实现信息压缩与数据降维

作用：特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性

平均法池化 (Avg-pooling)

20	5	2	10
3	10	7	1
8	1	6	9
4	7	3	5

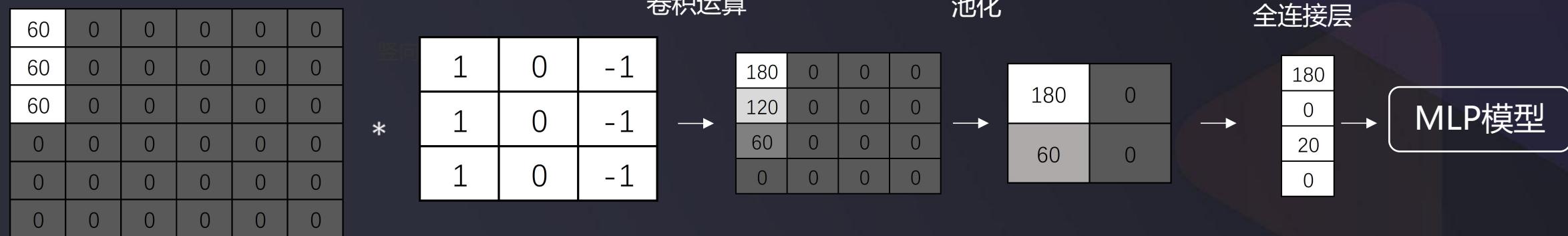
pooling_size: (2,2)
stride: (2,2)

9.5	5
5	5.75

Stride为窗口滑动步长，用于池化、卷积的计算中。

卷积神经网络(Convolutional Neural Network)

把卷积、池化、mlp先后连接在一起，组成一个能够高效提取图像重要信息的神经网络。



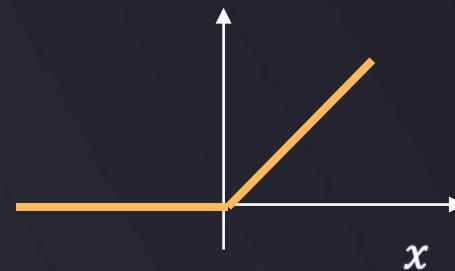
卷积之后、池化之前会增加激活函数以筛选保留重要信息

作用：

- 1、使部分神经元为0，防止过拟合
- 2、助于模型的求解

通常选用Relu函数：

$$f(x) = \max(x, 0)$$



| 卷积神经网络两大特点

- ① 参数共享 (parameter sharing): 同一个特征过滤器可用于整张图片
- ② 稀疏连接 (sparsity of connections): 生成的特征图片每个节点只与原图片中特定节点连接

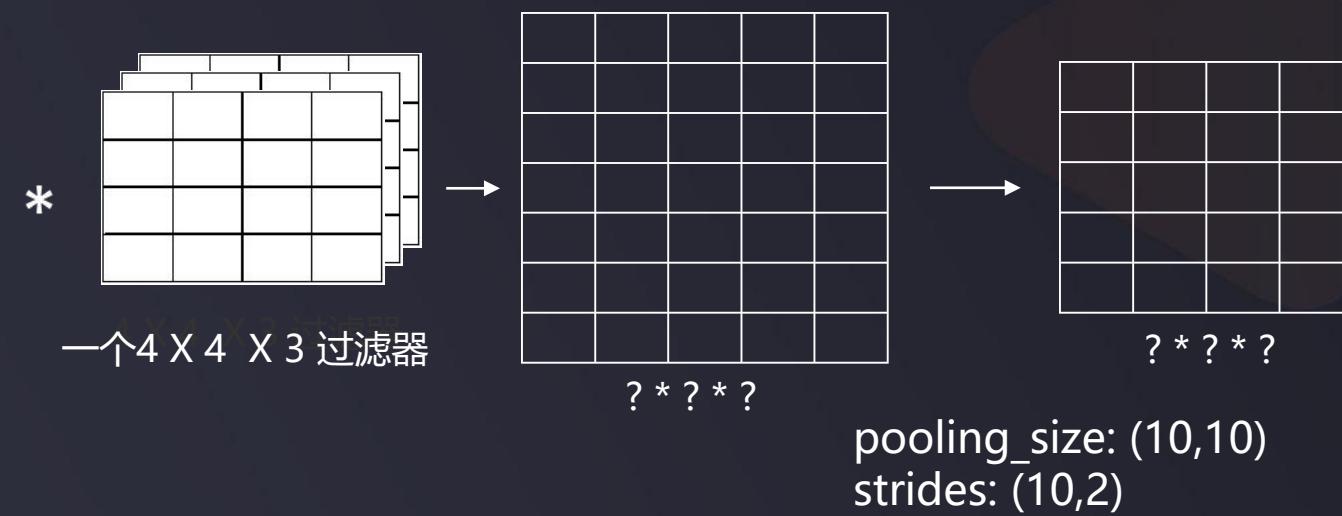
$$\begin{matrix} 60 & 60 & 0 & 0 & 0 & 0 \\ 60 & 60 & 0 & 0 & 0 & 0 \\ 60 & 60 & 0 & 0 & 0 & 0 \\ 60 & 60 & 60 & 60 & 0 & 0 \\ 60 & 60 & 60 & 60 & 0 & 0 \\ 60 & 60 & 60 & 60 & 0 & 0 \end{matrix} * \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} = \begin{matrix} 180 & 180 & 0 & 0 \\ 120 & 120 & 60 & 60 \\ 60 & 60 & 120 & 120 \\ 0 & 0 & 180 & 180 \end{matrix}$$

知识巩固

问题：以下有一张 400×500 像素的RGB图，计算经过卷积、池化处理以后，得到新矩阵的维度。



输入图片 (400×500 像素)





Python3人工智能入门+实战提升：深度学习

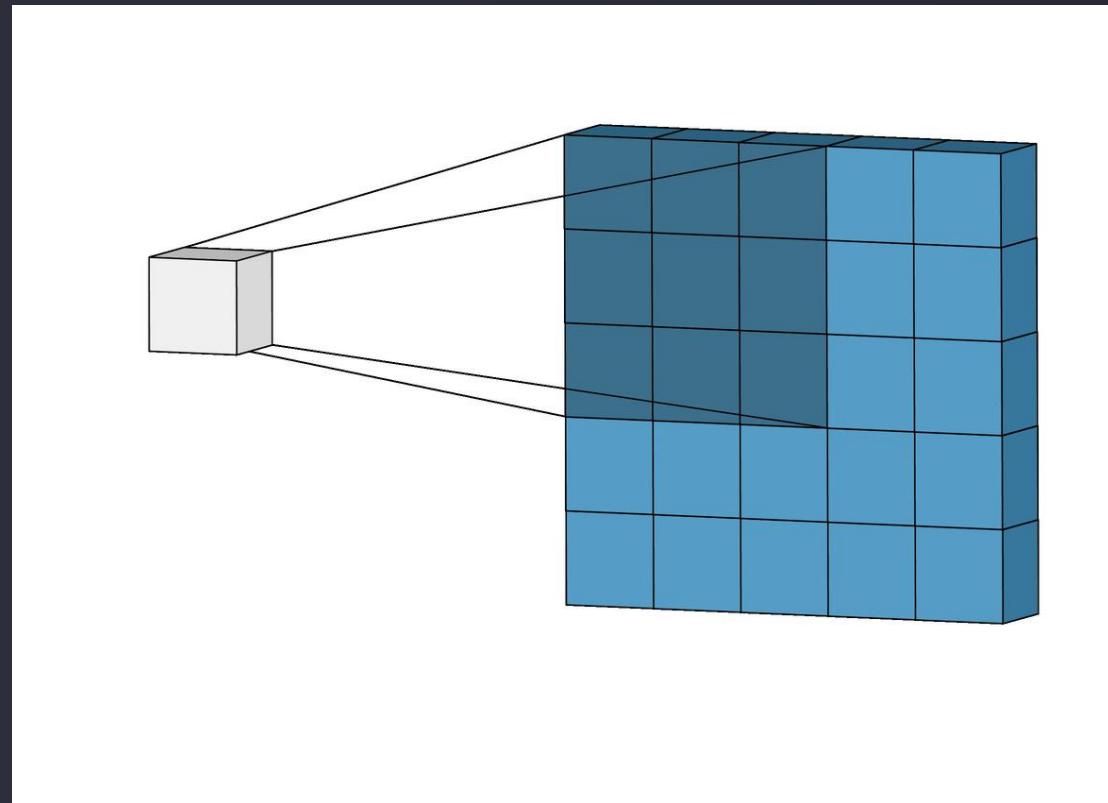
Chapter 3 卷积神经网络CNN

赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

现实问题思考



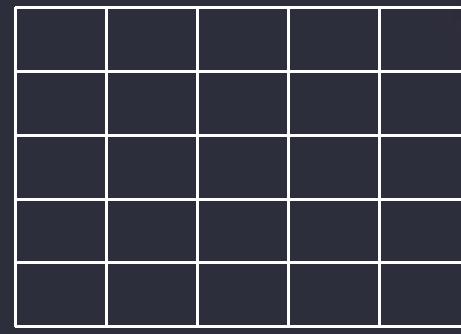
每个像素点被使用的次数：

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

周边位置的像素使用次数将明显少于
中间位置！

| 图像卷积的两个常见问题

- ① 边缘信息使用少，容易被忽略
- ② 图像被压缩，信息丢失



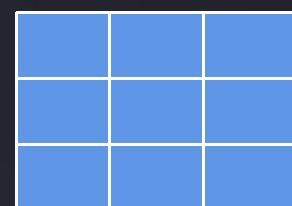
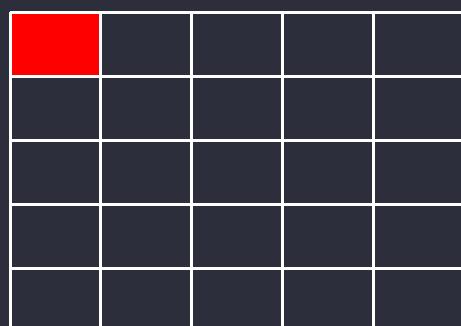
5*5输入

$$\begin{matrix} * & \begin{matrix} & & \\ & & \\ & & \end{matrix} & \rightarrow & \begin{matrix} & & \\ & & \end{matrix} \end{matrix}$$

The diagram shows a 5x5 input grid multiplied by a 3x3 filter kernel, resulting in a 3x3 output grid. The asterisk (*) indicates convolution, and the arrow indicates the result.

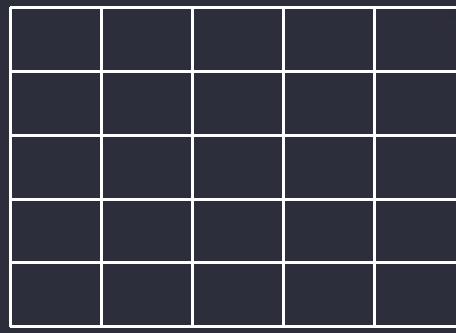
3*3过滤器

3*3

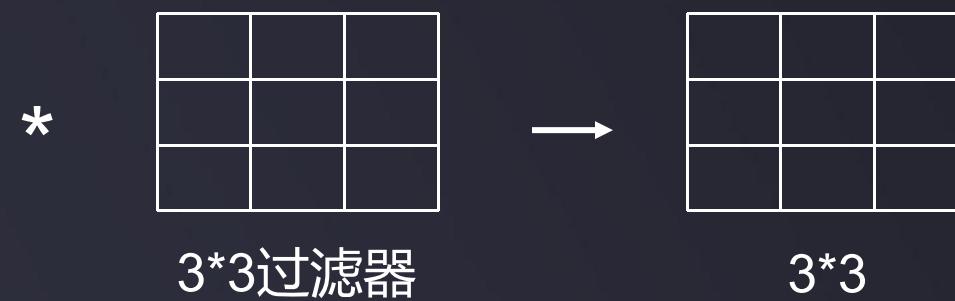


| 图像卷积的两个常见问题

- ① 边缘信息使用少，容易被忽略
- ② 图像被压缩，信息丢失



5*5输入

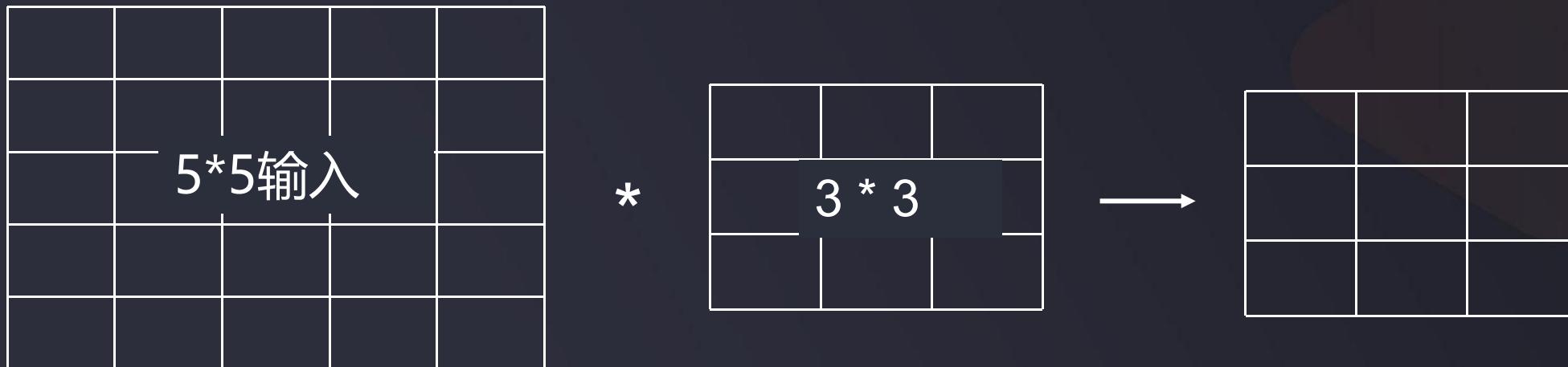


n*n输入



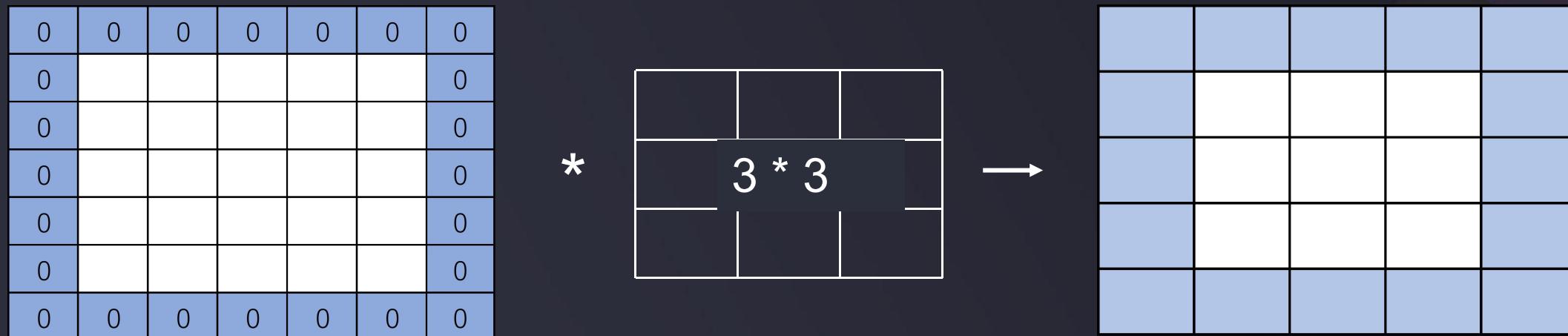
| 图像填充 (padding)

通过在图像周边添加新的像素，使边缘位置图像信息得到更多的利用



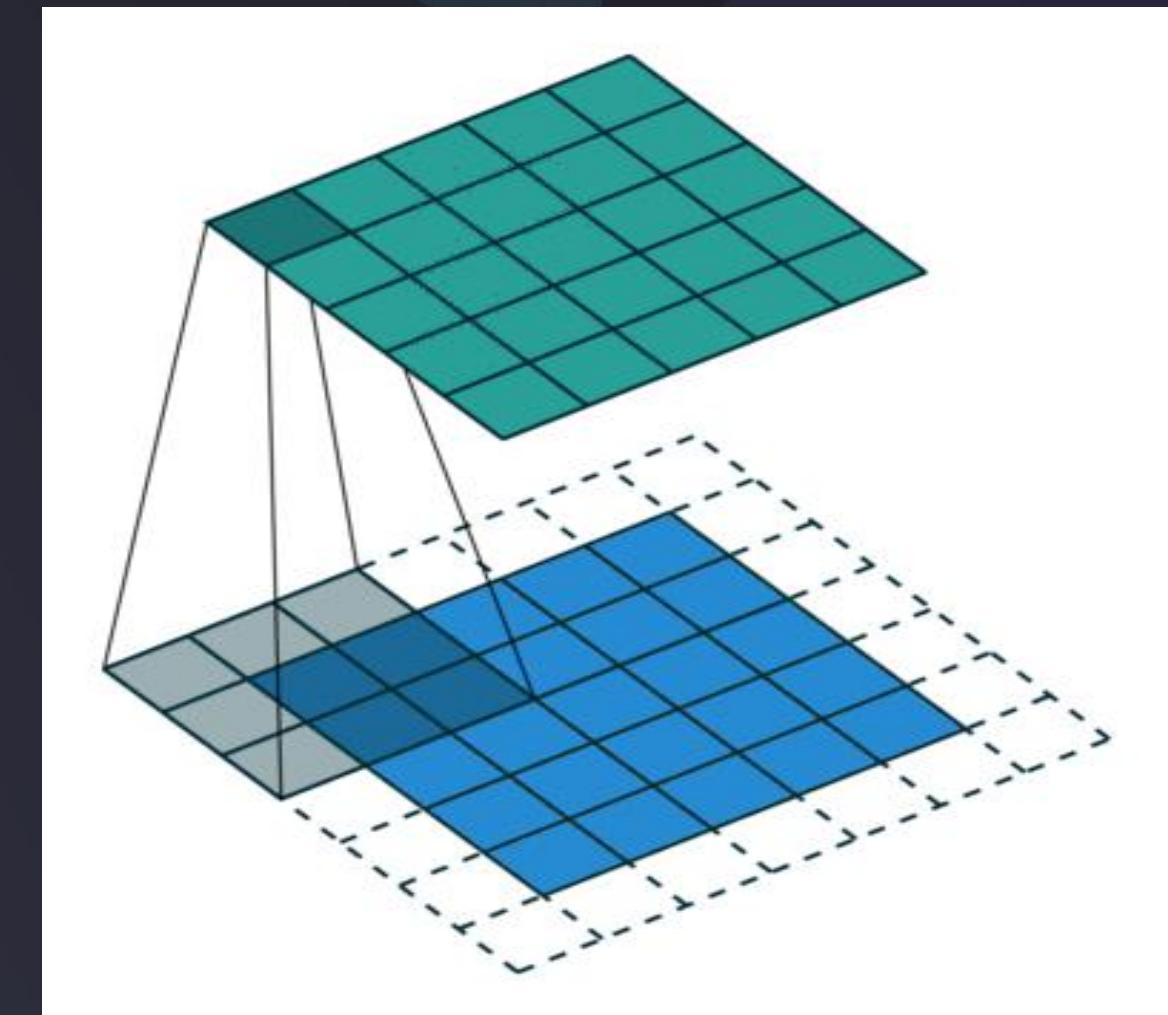
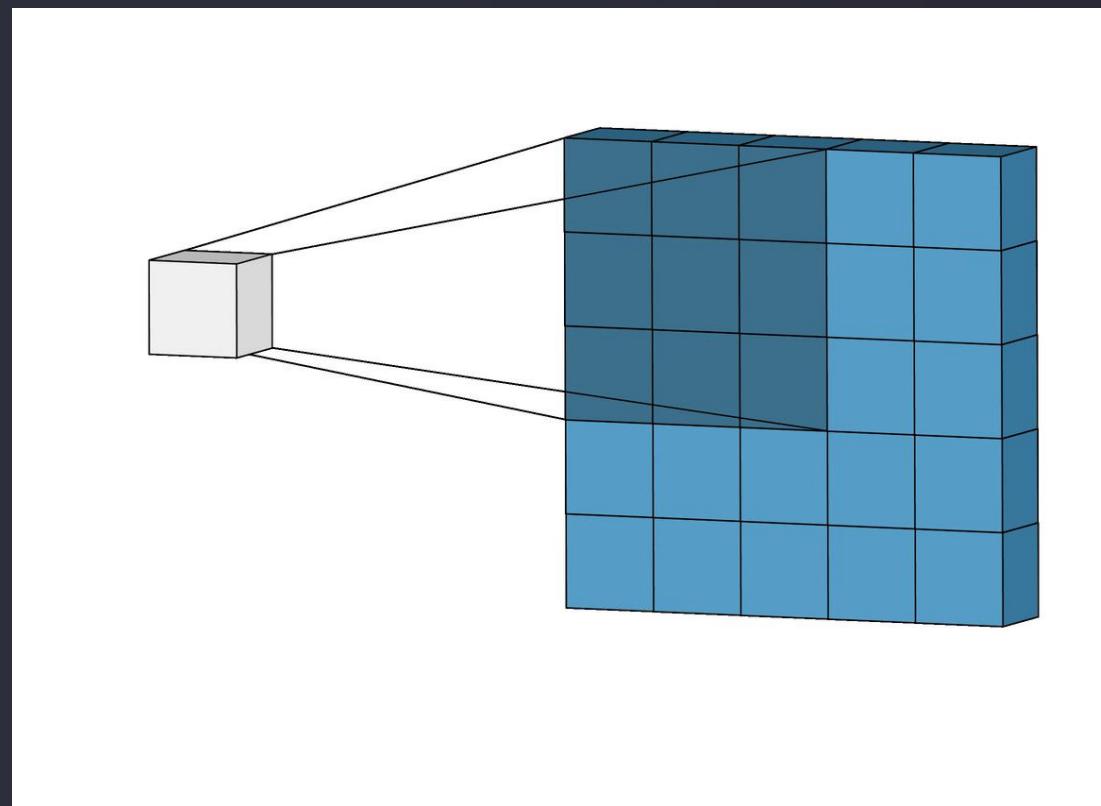
| 图像填充 (padding)

通过在图像周边添加新的像素，使边缘位置图像信息得到更多的利用



通过padding增加像素的数量，由过滤器尺寸与stride决定

| 图像填充后的卷积运算





Python3人工智能入门+实战提升：深度学习

Chapter 3 卷积神经网络CNN

赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

卷积神经网络(Convolutional Neural Network)

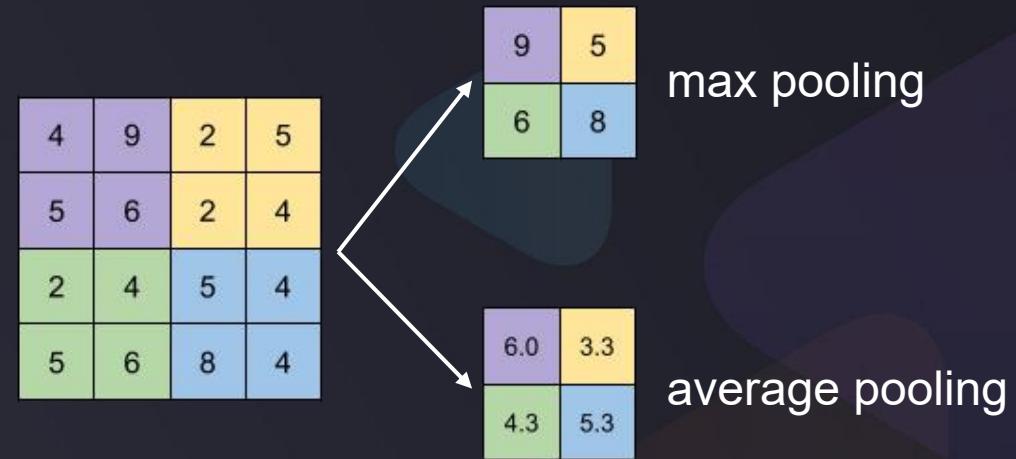
把卷积、池化、mlp先后连接在一起，组成一个能够高效提取图像重要信息的神经网络。



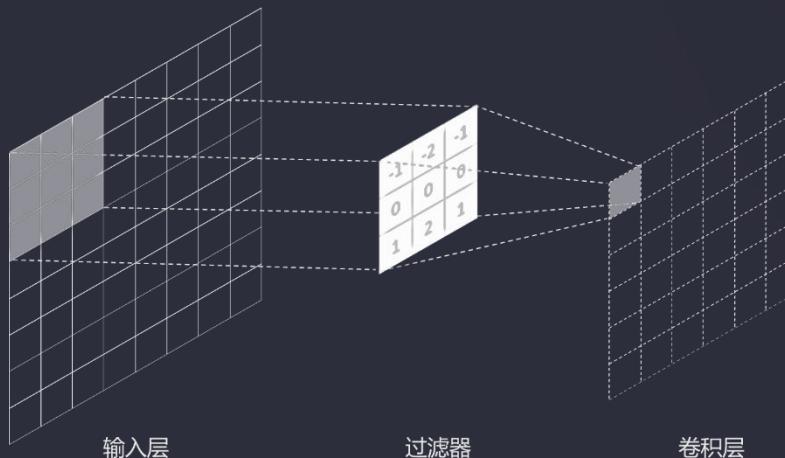
图像填充

0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

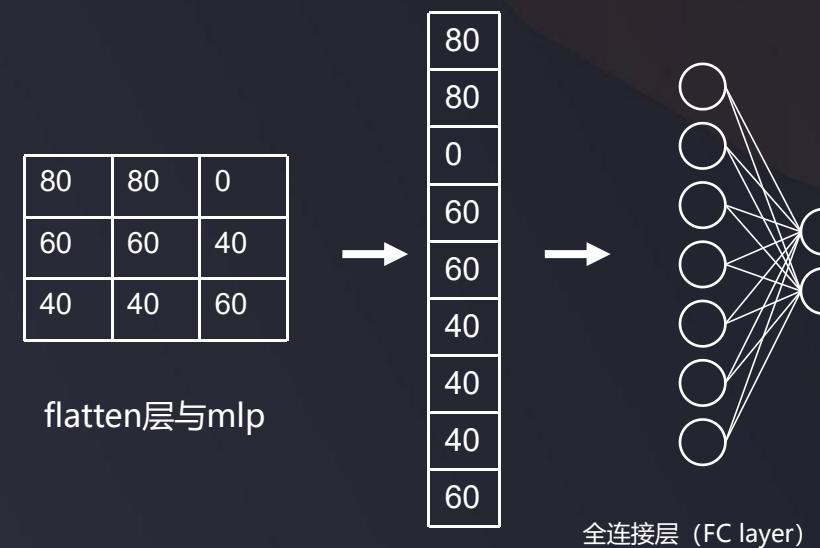
池化运算



卷积运算



多层感知器结构+预测输出



如何组合，能够构建出表现更好的结构？

| 站在巨人的肩膀上成长

卷积

filter尺寸

池化

哪种池化方法

填充

是否要填充

如何填充

尝试不同的组合、训练模型、对比结果

学习与借鉴别人训练好的模型！

| 站在巨人的肩膀上成长：经典的CNN模型

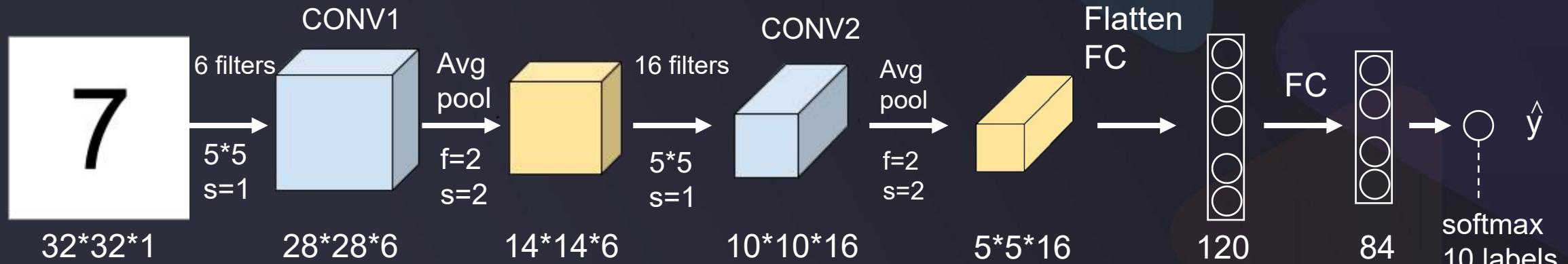
如何做：

- ① 参考经典的CNN结构，将其核心思想运用到新模型设计
- ② 使用经典的CNN模型结构提取图像重要轮廓，再建立MLP模型

经典的CNN模型：

- ① LeNet-5 ② AlexNet ③ VGG

|经典的CNN模型： LeNet-5



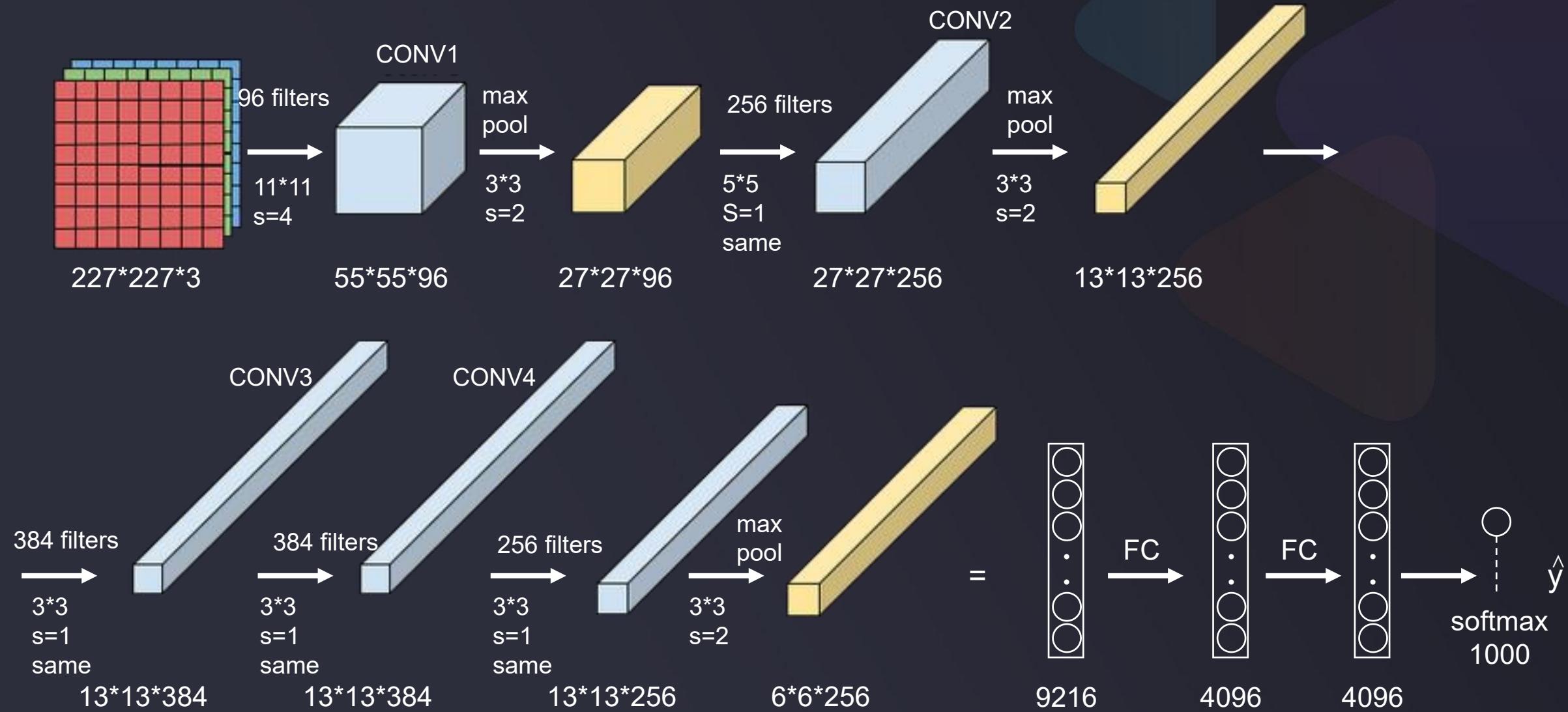
输入图像： 32×32 灰度图，1个通道 (channel)

训练参数： 约60,000个

特点：

- ① filter均为 $5 \times 5(s=1)$ 、池化为avg pool ($f=2, s=2$)，卷积与池化先后成对使用
- ② 随着网络越深，图像的高度和宽度在缩小，通道数在增加

|经典的CNN模型：AlexNet



|经典的CNN模型： AlexNet

输入图像： 227 X 227 X 3 RGB图， 3个通道

训练参数： 约60,000,000个

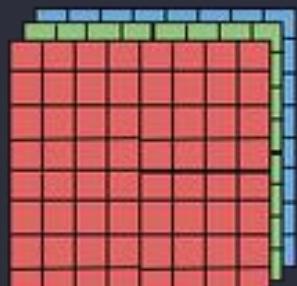
特点：

- ① 引入图像填充，采用max pool
- ② 更为复杂的结构，能够提取出更丰富的特征
- ③ 适用于识别较为复杂的彩色图，可识别1000种类别

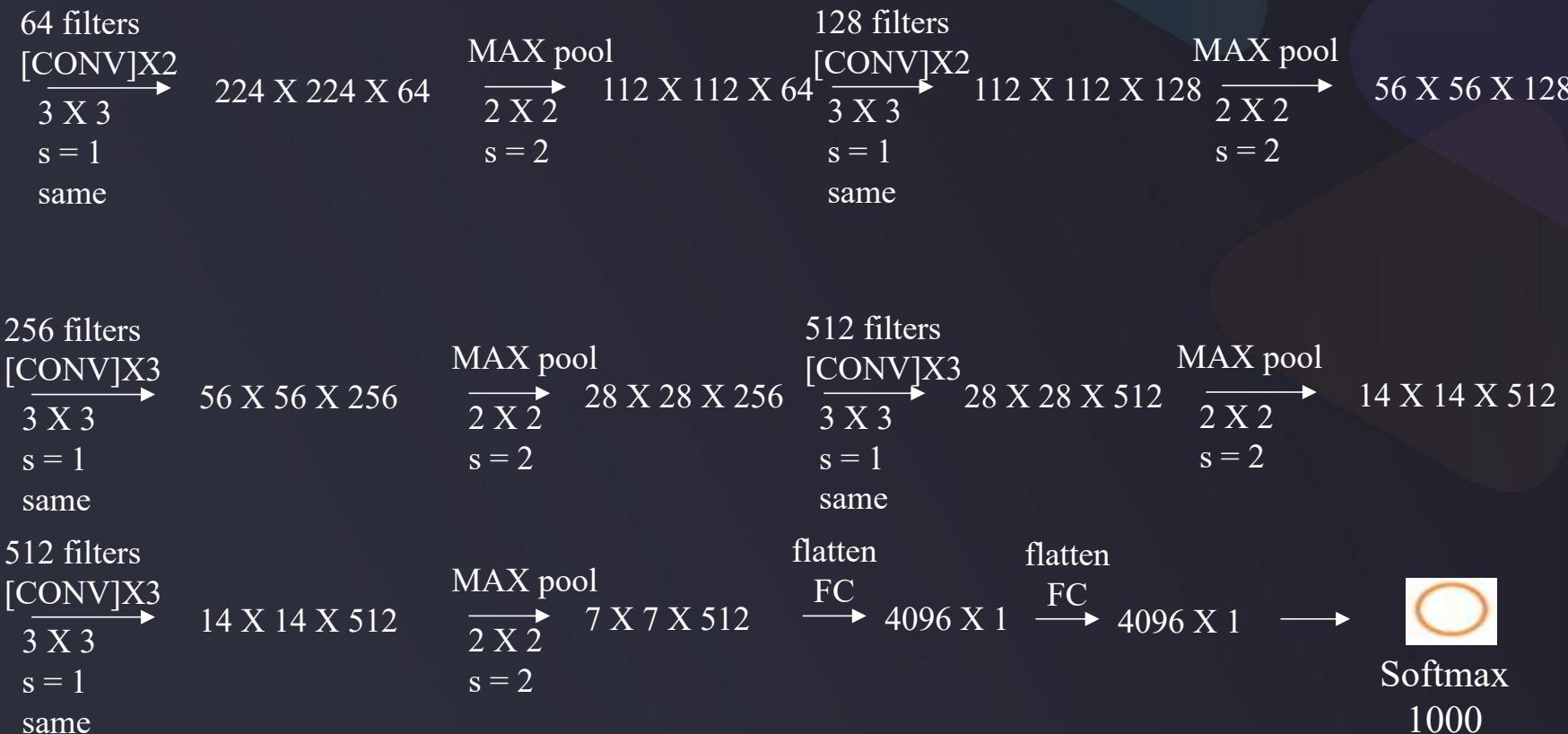
在计算机视觉应用中，深度学习可以很好的完成任务！

AlexNet 论文：<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

|经典的CNN模型： VGG-16



224 X 224 X 3



|经典的CNN模型： VGG-16

输入图像： 227 X 227 X 3 RGB图， 3个通道

训练参数： 约138,000,000个

特点：

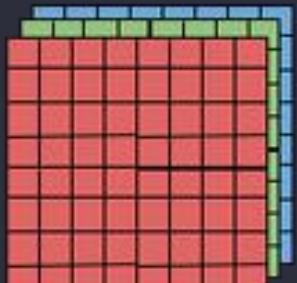
- 1、filter均为 3×3 , 步长为 1, 卷积前均使用了图像填充padding (same convolution) ;
- 2、池化均为max pool, 窗口尺寸均为 2×2 , 步长都是 2;
- 3、使用更多的filter提取轮廓信息, 进一步提高识别准确率

|经典CNN+自定义MLP

如何做：

- ① 参考经典的CNN结构，将其核心思想运用到新模型设计
 - ② 使用经典的CNN模型结构提取图像重要轮廓，再建立MLP模型
-
- ① 加载经典的CNN模型，剥除其FC层，用于提取图像的重要轮廓信息
 - ② 把经过模型处理后的数据作为输入，分类结果为输出，建立mlp模型
 - ③ 模型训练，寻找图片不同类别对应的关键信息

| VGG16+自定义MLP



224 X 224 X 3

64 filters
[CONV]X2
3 X 3
s = 1
same

MAX pool
 $\frac{2}{2} \times \frac{2}{2}$
s = 2

128 filters
[CONV]X2
3 X 3
s = 1
same

MAX pool
 $\frac{2}{2} \times \frac{2}{2}$
s = 2

256 filters
[CONV]X3
3 X 3
s = 1
same

MAX pool
 $\frac{2}{2} \times \frac{2}{2}$
s = 2

512 filters
[CONV]X3
3 X 3
s = 1
same

MAX pool
 $\frac{2}{2} \times \frac{2}{2}$
s = 2

512 filters
[CONV]X3
3 X 3
s = 1
same

MAX pool
 $\frac{2}{2} \times \frac{2}{2}$
s = 2

flatten

FC \rightarrow 4096 X 1

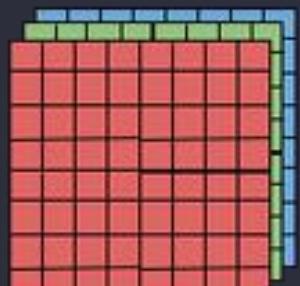
flatten

FC \rightarrow 4096 X 1 \longrightarrow

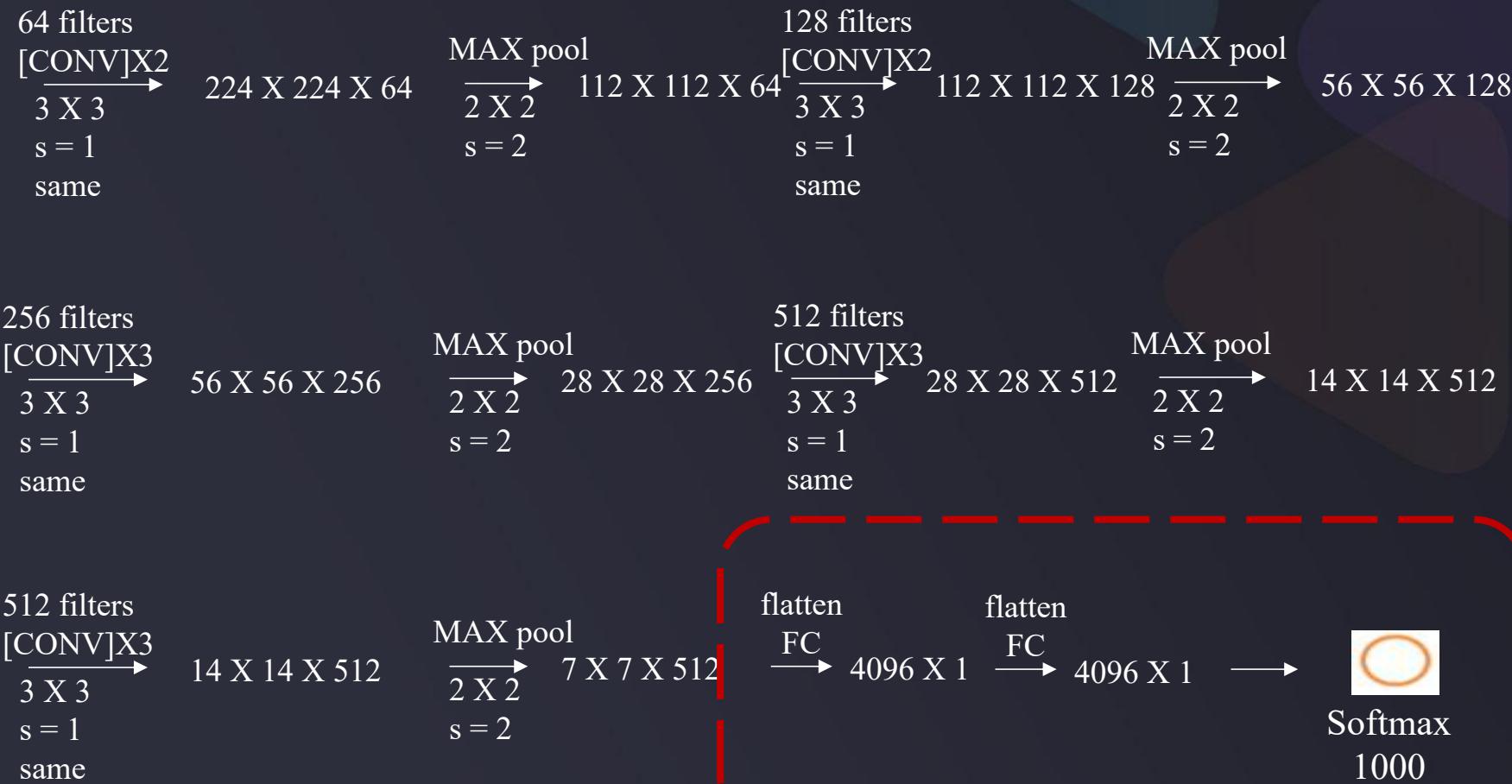
Softmax

1000

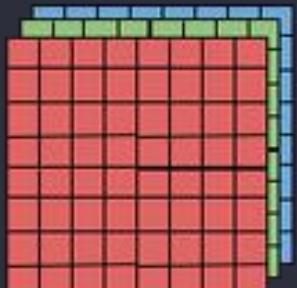
| VGG16+自定义MLP



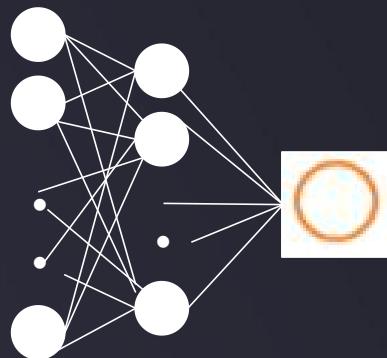
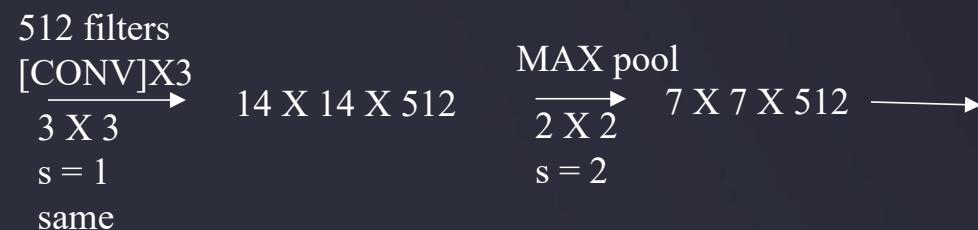
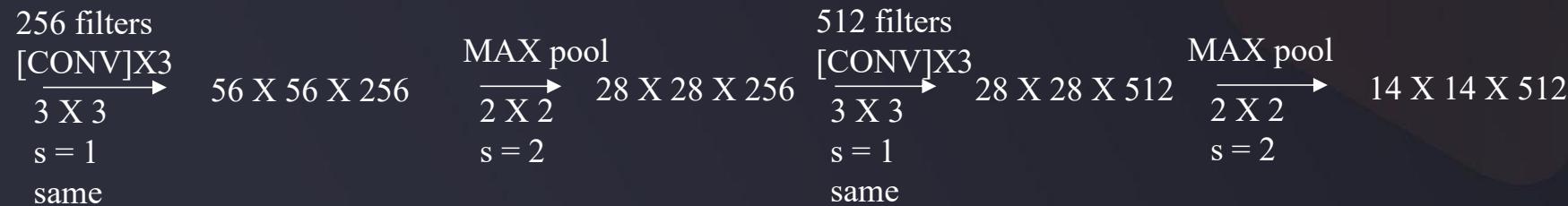
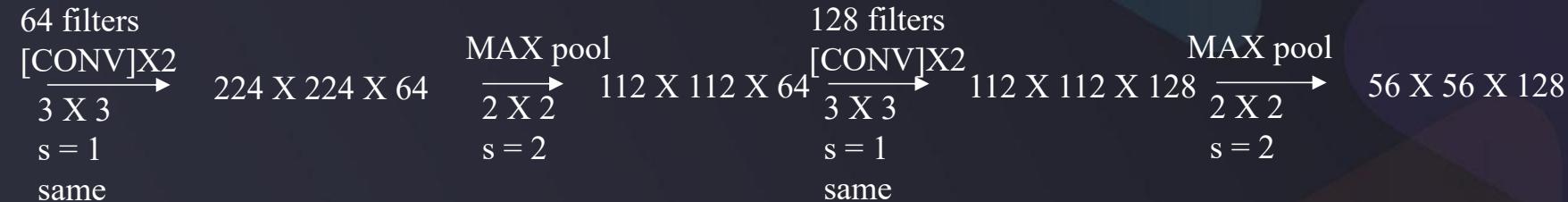
224 X 224 X 3



VGG16+自定义MLP



224 X 224 X 3



根据需求设计mlp结构

知识巩固

问题：设计一个最少10层的CNN结构，结合卷积层、池化层、padding，计算每层的尺寸。

卷积

filter尺寸

池化

Filter数量

填充

哪种池化方法

Stride如何设置

是否要填充

如何填充



Python3人工智能入门+实战提升：深度学习

Chapter 3 卷积神经网络CNN

赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

任务一：CNN识别猫狗

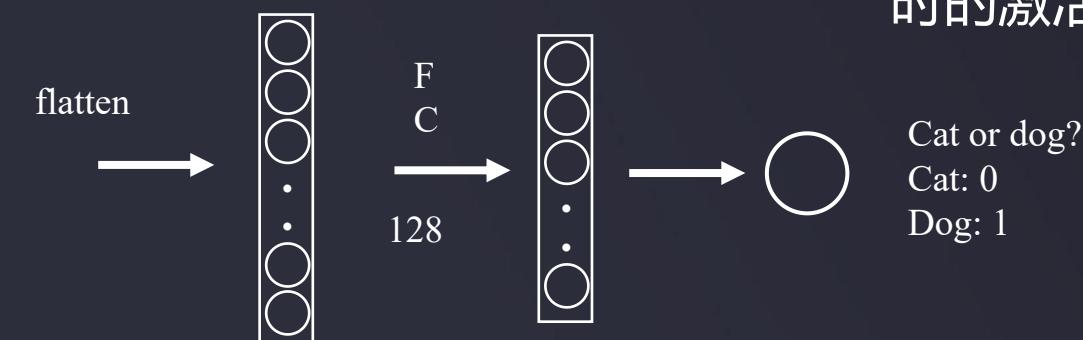
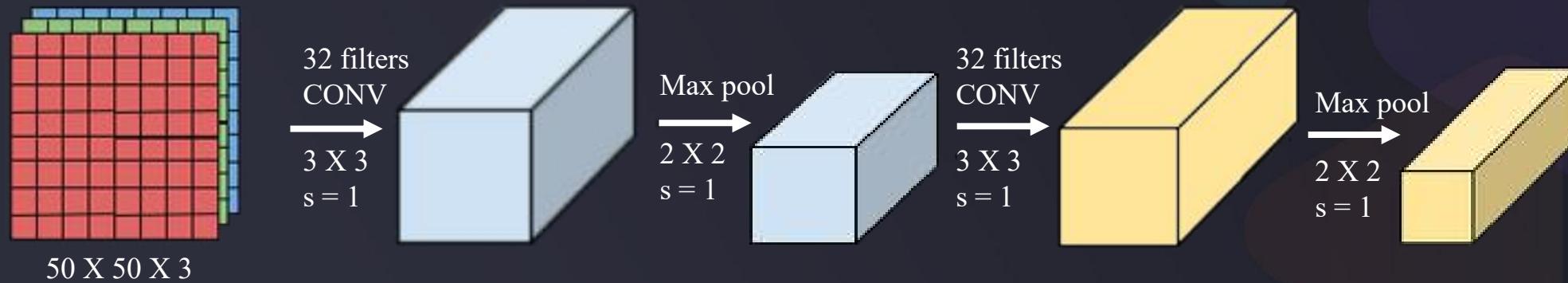
基于task1_data数据，建立cnn模型，实现猫狗识别。



- 1、通过ImageDataGenerator模块，实现本地图片批量加载；
- 2、查看数据基本结构，可视化加载后的样本图片；
- 3、建模并训练模型（迭代20次），计算训练集、测试集准确率；
- 4、对提供的1-9猫/狗图片，进行预测

模型结构：后一页PPT

CNN识别猫狗-CNN结构



备注：卷积层采用relu作为输出时的激活函数

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

图片加载：

```
from keras.preprocessing.image import ImageDataGenerator
```

```
# 图像增强/预处理配置 (数值归一化、缩放、旋转、平移等)
train_datagen = ImageDataGenerator(rescale = 1./255)
```

#加载图像：

```
training_set =
train_datagen.flow_from_directory('./Dataset/training_set',
target_size = (50, 50),
batch_size = 32,
class_mode = 'binary')
```

```
Found 8000 images belonging to 2 classes.
```

参考资料：<https://keras.io/preprocessing/image/>

CNN识别猫狗

数据加载及展示

```
#查看数据类型  
print(type(training_set))
```

```
<class 'keras.preprocessing.image.DirectoryIterator'>
```

数据预处理

```
#每个批次样本的数量  
training_set.batch_size
```

32

模型建立及训练

```
#加载样本的名称  
training_set.filenames
```

```
Out[68]: ['cats\\cats_1.jpg',  
          'cats\\cats_10.jpg',  
          'cats\\cats_100.jpg',  
          'cats\\cats_1000.jpg',  
          'cats\\cats_1001.jpg',  
          'cats\\cats_1002.jpg',  
          'cats\\cats_1003.jpg',  
          'cats\\cats_1004.jpg',  
          'cats\\cats_1005.jpg',  
          'cats\\cats_1006.jpg',  
          'cats\\cats_1007.jpg',  
          'cats\\cats_1008.jpg',  
          'cats\\cats_1009.jpg',  
          'cats\\cats_1010.jpg',  
          'cats\\cats_1011.jpg',  
          'cats\\cats_1012.jpg',  
          'cats\\cats_1013.jpg',  
          'cats\\cats_1014.jpg',  
          'cats\\cats_1015.jpg',  
          'cats\\cats_1016.jpg',  
          'cats\\cats_1017.jpg',  
          'cats\\cats_1018.jpg',  
          'cats\\cats_1019.jpg',  
          'cats\\cats_1020.jpg',  
          'cats\\cats_1021.jpg',  
          'cats\\cats_1022.jpg',  
          'cats\\cats_1023.jpg',  
          'cats\\cats_1024.jpg',  
          'cats\\cats_1025.jpg',  
          'cats\\cats_1026.jpg',  
          'cats\\cats_1027.jpg',  
          'cats\\cats_1028.jpg',  
          'cats\\cats_1029.jpg',  
          'cats\\cats_1030.jpg',  
          'cats\\cats_1031.jpg',  
          'cats\\cats_1032.jpg',  
          'cats\\cats_1033.jpg',  
          'cats\\cats_1034.jpg',  
          'cats\\cats_1035.jpg',  
          'cats\\cats_1036.jpg',  
          'cats\\cats_1037.jpg',  
          'cats\\cats_1038.jpg',  
          'cats\\cats_1039.jpg',  
          'cats\\cats_1040.jpg',  
          'cats\\cats_1041.jpg',  
          'cats\\cats_1042.jpg',  
          'cats\\cats_1043.jpg',  
          'cats\\cats_1044.jpg',  
          'cats\\cats_1045.jpg',  
          'cats\\cats_1046.jpg',  
          'cats\\cats_1047.jpg',  
          'cats\\cats_1048.jpg',  
          'cats\\cats_1049.jpg',  
          'cats\\cats_1050.jpg',  
          'cats\\cats_1051.jpg',  
          'cats\\cats_1052.jpg',  
          'cats\\cats_1053.jpg',  
          'cats\\cats_1054.jpg',  
          'cats\\cats_1055.jpg',  
          'cats\\cats_1056.jpg',  
          'cats\\cats_1057.jpg',  
          'cats\\cats_1058.jpg',  
          'cats\\cats_1059.jpg',  
          'cats\\cats_1060.jpg',  
          'cats\\cats_1061.jpg',  
          'cats\\cats_1062.jpg',  
          'cats\\cats_1063.jpg',  
          'cats\\cats_1064.jpg',  
          'cats\\cats_1065.jpg',  
          'cats\\cats_1066.jpg',  
          'cats\\cats_1067.jpg',  
          'cats\\cats_1068.jpg',  
          'cats\\cats_1069.jpg',  
          'cats\\cats_1070.jpg',  
          'cats\\cats_1071.jpg',  
          'cats\\cats_1072.jpg',  
          'cats\\cats_1073.jpg',  
          'cats\\cats_1074.jpg',  
          'cats\\cats_1075.jpg',  
          'cats\\cats_1076.jpg',  
          'cats\\cats_1077.jpg',  
          'cats\\cats_1078.jpg',  
          'cats\\cats_1079.jpg',  
          'cats\\cats_1080.jpg',  
          'cats\\cats_1081.jpg',  
          'cats\\cats_1082.jpg',  
          'cats\\cats_1083.jpg',  
          'cats\\cats_1084.jpg',  
          'cats\\cats_1085.jpg',  
          'cats\\cats_1086.jpg',  
          'cats\\cats_1087.jpg',  
          'cats\\cats_1088.jpg',  
          'cats\\cats_1089.jpg',  
          'cats\\cats_1090.jpg',  
          'cats\\cats_1091.jpg',  
          'cats\\cats_1092.jpg',  
          'cats\\cats_1093.jpg',  
          'cats\\cats_1094.jpg',  
          'cats\\cats_1095.jpg',  
          'cats\\cats_1096.jpg',  
          'cats\\cats_1097.jpg',  
          'cats\\cats_1098.jpg',  
          'cats\\cats_1099.jpg']
```

模型预测

结果展示及表现评估

CNN识别猫狗

数据加载及展示

```
#确认输入数据标签  
training_set.class_indices
```

```
{'cats': 0, 'dogs': 1}
```

数据预处理

```
#加载后按批次存放的每个样本对应的原始数据序号  
training_set.index_array
```

```
array([4917, 1369, 681, ..., 7673, 1603, 297])
```

模型建立及训练

```
#加载后第一个批次数据集第一个样本的名称  
training_set.filenames[4917]
```

```
'dogs\\dogs_1824.jpg'
```

模型预测

结果展示及表现评估

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

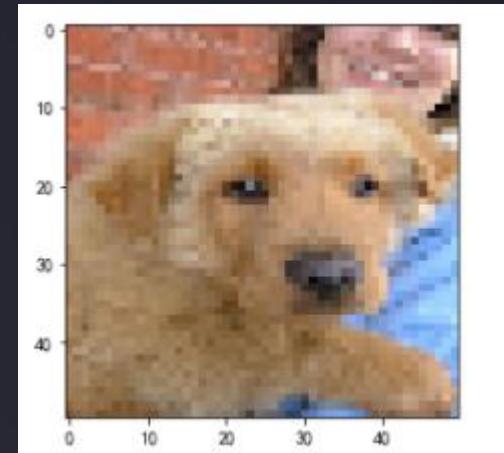
```
#training_set为tuple类型，第一个[]确定第几批次  
#第二个[]确定输入的图像数据x或者是结果标签y  
#第三个[]检索对应批次第几个样本的数据  
training_set[0][1]
```

```
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 1.,  
       0., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0., 1.],  
      dtype=float32)
```

```
#第一批次所有样本的图像数据维度确认  
training_set[0][0].shape
```

(32, 50, 50, 3)

```
#可视化第一批次第一个样本  
fig1 = plt.figure()  
plt.imshow(training_set[0][0][:,:,:])
```



CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

建立CNN模型：

```
from keras.models import Sequential  
from keras.layers import Conv2D, MaxPooling2D,  
Flatten, Dense  
  
model = Sequential()  
# 卷积层  
model.add(Conv2D(32, (3, 3), input_shape = (50, 50,  
3), activation = 'relu'))  
# 池化层  
model.add(MaxPooling2D(pool_size = (2, 2)))
```

```
# 第二个卷积、池化层  
model.add(Conv2D(32, (3, 3), activation = 'relu'))  
model.add(MaxPooling2D(pool_size = (2, 2)))  
# Flattening层  
model.add(Flatten())  
# 全连接层  
model.add(Dense(units = 128, activation = 'relu'))  
model.add(Dense(units = 1, activation = 'sigmoid'))
```

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#模型求解参数配置

```
model.compile(optimizer='adam', loss='binary_crossentropy',  
metrics=['accuracy'])  
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 22, 22, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 11, 11, 32)	0
flatten_1 (Flatten)	(None, 3872)	0
dense_1 (Dense)	(None, 128)	495744
dense_2 (Dense)	(None, 1)	129
Total params:	506,017	
Trainable params:	506,017	
Non-trainable params:	0	

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#模型训练

```
model.fit_generator(training_set,epochs=20)
```

```
Epoch 1/20  
250/250 [=====] - 72s 287ms/step - loss: 0.6766 - accuracy: 0.5790  
Epoch 2/20  
250/250 [=====] - 40s 159ms/step - loss: 0.5762 - accuracy: 0.7014  
Epoch 3/20  
250/250 [=====] - 39s 155ms/step - loss: 0.5188 - accuracy: 0.7435  
Epoch 4/20  
250/250 [=====] - 41s 163ms/step - loss: 0.4756 - accuracy: 0.7664  
Epoch 5/20  
250/250 [=====] - 40s 160ms/step - loss: 0.4424 - accuracy: 0.7955  
Epoch 6/20  
250/250 [=====] - 39s 157ms/step - loss: 0.4086 - accuracy: 0.8125  
Epoch 7/20  
250/250 [=====] - 45s 178ms/step - loss: 0.3729 - accuracy: 0.8316  
Epoch 8/20  
250/250 [=====] - 38s 154ms/step - loss: 0.3285 - accuracy: 0.8569  
Epoch 9/20  
250/250 [=====] - 39s 155ms/step - loss: 0.2826 - accuracy: 0.8815  
Epoch 10/20  
250/250 [=====] - 42s 167ms/step - loss: 0.2529 - accuracy: 0.8896  
Epoch 11/20
```

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测及表现评估

结果展示

```
#训练数据集整体样本预测准确率  
accuracy_train = model.evaluate_generator(training_set)  
print(accuracy_train)
```

```
[0.008329642936587334, 0.9991250038146973]
```

```
#测试数据集预测准确率  
test_set =  
train_datagen.flow_from_directory('./dataset/test_set',target_size=(50,50),batch_size=32,class_mode='binary')
```

```
accuracy_test = model.evaluate_generator(test_set)  
print(accuracy_test)
```

```
Found 2000 images belonging to 2 classes.  
[1.2355401515960693, 0.7534999847412109]
```

CNN识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

```
#单张图片加载与结果预测
from keras.preprocessing.image import load_img,
img_to_array
pic_dog = '1.png'
pic_dog = load_img(pic_dog,target_size=(50,50))
pic_dog = img_to_array(pic_dog)
pic_dog = pic_dog/255
pic_dog = pic_dog.reshape(1,50,50,3)
result = model.predict_classes(pic_dog)
print('dog' if result==1 else 'cat')
fig10 = plt.figure()
plt.imshow(pic_dog[0])
```



CNN识别猫狗

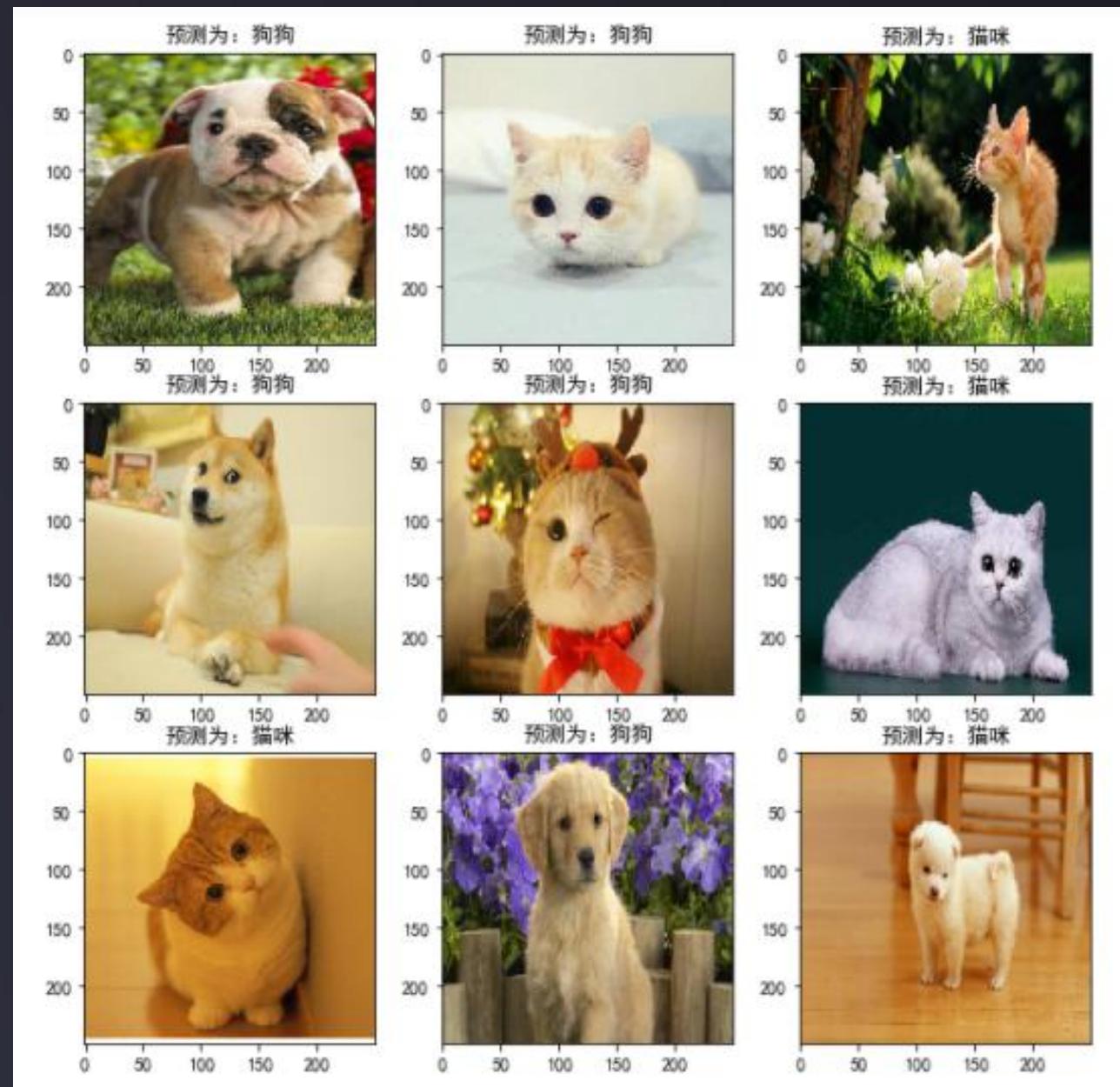
数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示



任务二：VGG16+mlp高准确率识别猫狗

基于task2_data，利用VGG16结构，提高猫狗识别的准确率



- 1、对单张图片，利用VGG16提取图像特征
- 2、对所有图片，利用VGG16进行特征提取，并把数据分为训练数据、测试数据两部分
- 3、对提取特征后的数据建立mlp模型，进行模型训练，计算模型在训练、测试数据集的准确率
- 4、对提供的1-9猫/狗图片，进行预测，将结果与任务一的结果进行对比

备注：

- 1、数据分离参数：`test_size=0.2, random_state=0`
- 2、mlp模型只有一个隐藏层（10个神经元）、激活函数relu

VGG16+mlp高准确率识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

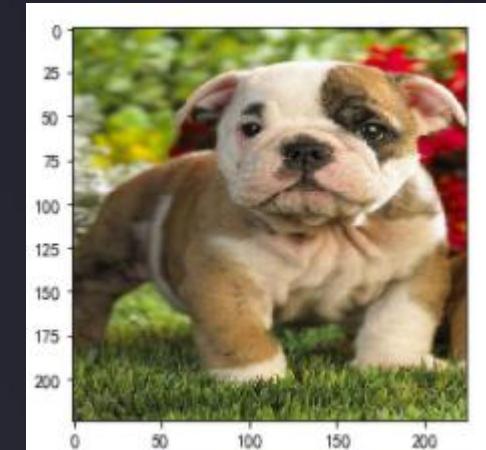
#单张图片加载

```
from keras.preprocessing.image import  
load_img,img_to_array  
img_path = '1.png'  
img = load_img(img_path,target_size=(224,224))  
print(type(img))
```

<class 'PIL.Image.Image'>

#图像可视化

```
from matplotlib import pyplot as plt  
fig1 = plt.figure()  
plt.imshow(img)
```



VGG16+mlp高准确率识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#将图片数据转化为数组格式类型  
img = img_to_array(img)  
print(type(img))  
print(img.shape)
```

```
<class 'numpy.ndarray'>  
(224, 224, 3)
```

```
#维度转化  
from keras.applications.vgg16 import preprocess_input  
import numpy as np  
x = np.expand_dims(img, axis=0)  
x = preprocess_input(x)  
print(x.shape)
```

```
(1, 224, 224, 3)
```

VGG16+mlp高准确率识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#特征提取

```
from keras.applications.vgg16 import VGG16  
#载入VGG16结构（去除全连接与输出层）：  
model_vgg = VGG16(weights='imagenet',include_top=False)  
features = model_vgg.predict(x)  
print(features.shape)
```

(1, 7, 7, 512)

#flatten展开

```
features = features.reshape(1,7*7*512)  
print(features.shape)
```

(1, 25088)

批量提取图片特征



```
In [6]: #load image and preprocess it with vgg16 structure
#--by flare
from keras.preprocessing.image import img_to_array,load_img
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
import numpy as np

model_vgg = VGG16(weights='imagenet', include_top=False)
#define a method to load and preprocess the image
def modelProcess(img_path,model):
    img = load_img(img_path, target_size=(224, 224))
    img = img_to_array(img)
    x = np.expand_dims(img,axis=0)
    x = preprocess_input(x)
    x_vgg = model.predict(x)
    x_vgg = x_vgg.reshape(1,25088)
    return x_vgg
#list file names of the training datasets
import os
folder = "dataset/data_vgg/cats"
dirs = os.listdir(folder)
#generate path for the images
img_path = []
for i in dirs:
    if os.path.splitext(i)[1] == ".jpg":
        img_path.append(i)
img_path = [folder+"//"+i for i in img_path]

#preprocess multiple images
features1 = np.zeros([len(img_path),25088])
for i in range(len(img_path)):
    feature_i = modelProcess(img_path[i],model_vgg)
    print('processed:',img_path[i])
    features1[i] = feature_i
```

VGG16+mlp高准确率识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#建立mlp模型

```
from keras.models import Sequential  
from keras.layers import Dense  
model = Sequential()  
model.add(Dense(units=10,activation='relu',input_dim=25088))  
model.add(Dense(units=1,activation='sigmoid'))  
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
dense_3 (Dense)	(None, 10)	250890
dense_4 (Dense)	(None, 1)	11
<hr/>		
Total params: 250,901		
Trainable params: 250,901		
Non-trainable params: 0		

VGG16+mlp高准确率识别猫狗

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#参数配置与训练

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])  
#train the model  
model.fit(X_train,y_train,epochs=50)
```

```
Epoch 1/50  
480/480 [=====] - 0s 660us/step - loss: 2.7832 - accuracy: 0.5667  
Epoch 2/50  
480/480 [=====] - 0s 429us/step - loss: 0.5504 - accuracy: 0.7146  
Epoch 3/50  
480/480 [=====] - 0s 423us/step - loss: 0.4494 - accuracy: 0.9208  
Epoch 4/50  
480/480 [=====] - 0s 435us/step - loss: 0.3626 - accuracy: 0.9729  
Epoch 5/50  
480/480 [=====] - 0s 433us/step - loss: 0.3473 - accuracy: 0.9792  
Epoch 6/50  
480/480 [=====] - 0s 431us/step - loss: 0.3417 - accuracy: 0.9854  
Epoch 7/50  
480/480 [=====] - 0s 431us/step - loss: 0.3292 - accuracy: 0.9917  
Epoch 8/50  
480/480 [=====] - 0s 435us/step - loss: 0.3255 - accuracy: 0.9896
```

VGG16+mlp高准确率识别猫狗

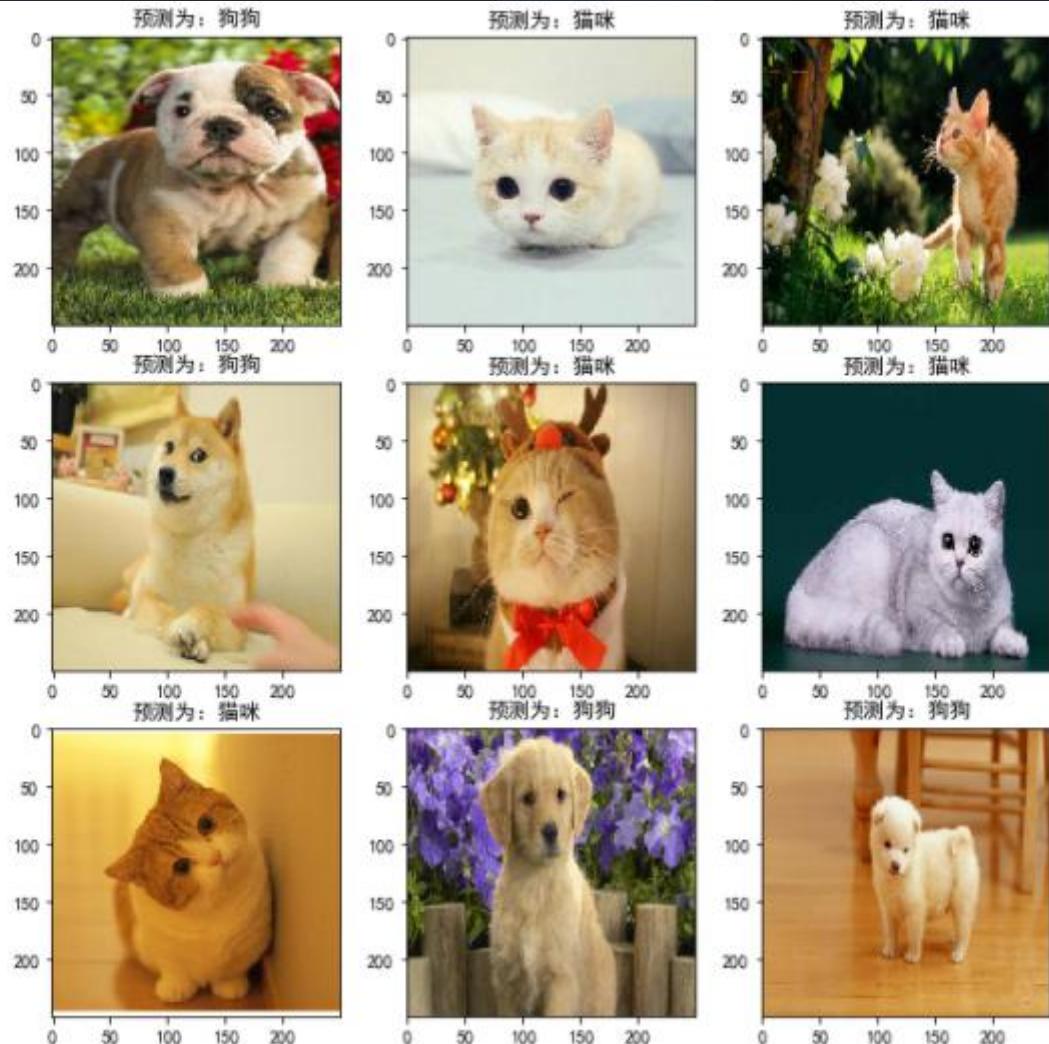
数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示





Python3人工智能入门+实战提升：深度学习

Chapter 5 聚类

赵辛



Python3人工智能入门+实战提升：深度学习

Chapter 3 卷积神经网络CNN

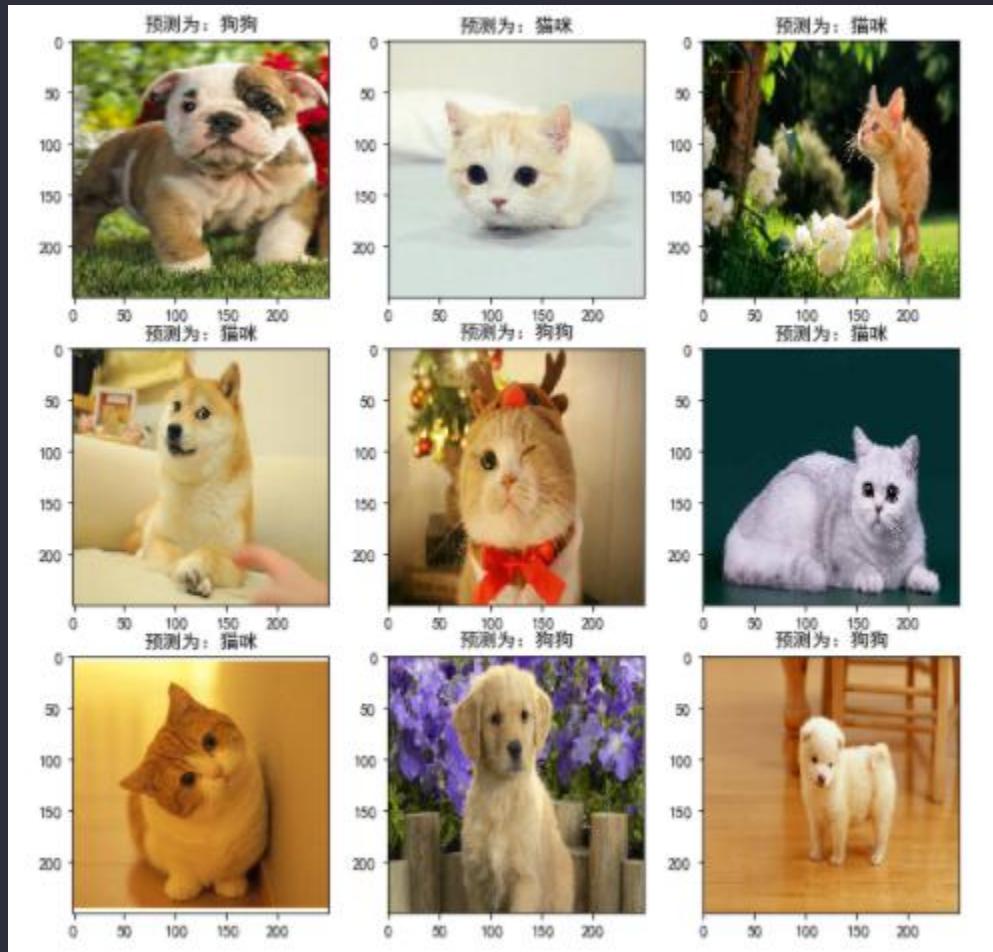
赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

任务一：CNN识别猫狗

基于task1_data数据，建立cnn模型，实现猫狗识别。



- 1、通过ImageDataGenerator模块，实现本地图片批量加载；
- 2、查看数据基本结构，可视化加载后的样本图片；
- 3、建模并训练模型（迭代20次），计算训练集、测试集准确率；
- 4、对提供的1-9猫/狗图片，进行预测

模型结构：后一页PPT



Python3人工智能入门+实战提升：深度学习

Chapter 3 卷积神经网络CNN

赵辛

Chapter 3 卷积神经网络CNN

-
- 1 --图像卷积运算
 - 2 --池化
 - 3 --图像填充
 - 4 --经典的CNN模型
 - 5 --实战准备
 - 6 --实战(一) CNN识别猫狗
 - 7 --实战(二) VGG16+mlp高准确率识别猫狗

任务二：VGG16+mlp高准确率识别猫狗

基于task2_data，利用VGG16结构，提高猫狗识别的准确率



- 1、对单张图片，利用VGG16提取图像特征
 - 2、对所有图片，利用VGG16进行特征提取，并把数据分为训练数据、测试数据两部分
 - 3、对提取特征后的数据建立mlp模型，进行模型训练，计算模型在训练、测试数据集的准确率
 - 4、对提供的1-9猫/狗图片，进行预测，将结果与任务一的结果进行对比
- 备注：
- 1、数据分离参数：`test_size=0.2, random_state=0`
 - 2、mlp模型只有一个隐藏层（10个神经元）、激活函数relu



Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构：LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一）RNN预测股价
 - 6 --实战（二）LSTM文本生成

现实问题思考

人类感知世界最重要的五种方式：视觉、听觉、触觉、嗅觉、味觉

听觉：

- ① 听到声音
- ② 交流词句的顺序代表不同的含义

下课没

VS

没下课

数据以不同的序列展示将代表不同的信息：序列模型

|序列模型 (Sequence Model)

任务：识别所说的话

在进行语音识别时，给定了一个输入音频片段 X，并要求输出对应的文字记录 Y。

输入音频片段



转化出的文字：

好的，那我们现在开始上课，那我们可以
看到在我讲的过程中，他的这个转化效果
是非常好的。

基于语音内容及其前后信息进行
文字预测

|序列模型 (Sequence Model)

任务： 物体位置预测

$T = 5\text{s}$ 的时候，车在什么位置？



通过目标前一个时刻所处的位置、速度和方向，
可以预测其下一个时刻的位置

| 序列模型

定义：输入或者输出中包含有序列数据的模型叫做序列模型。

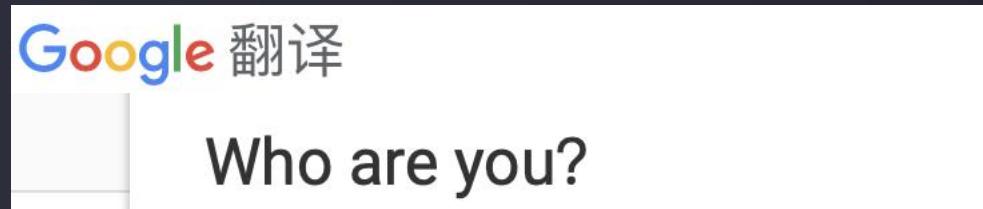
两大特点：

1. 输入（输出）元素之间是具有顺序关系。不同的顺序，得到的结果应该是不同的，比如“不睡觉”和“睡觉不”这两个短语的意思是不同的
2. 输入输出不定长。比如聊天机器人，聊天之间的对话长度都是不定的。

序列中包含信息

场景应用

机器翻译



你是谁?

语音识别



Hi Siri, 讲个笑话

行为预测



物体在某时刻的位置

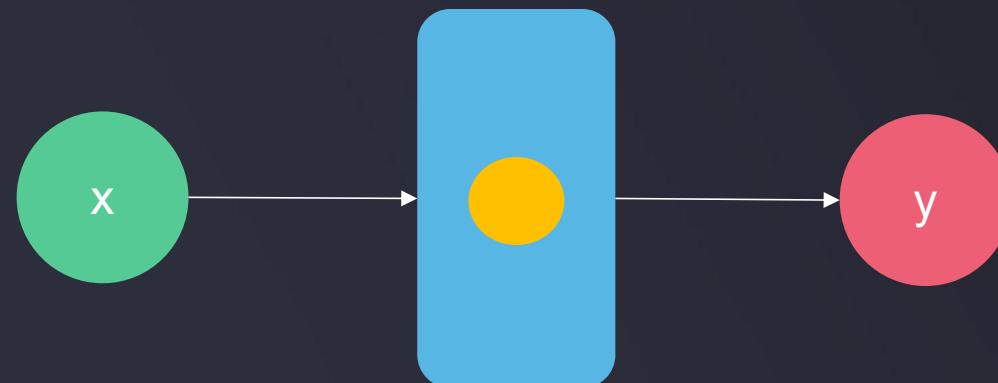
股价预测



涨? 跌?

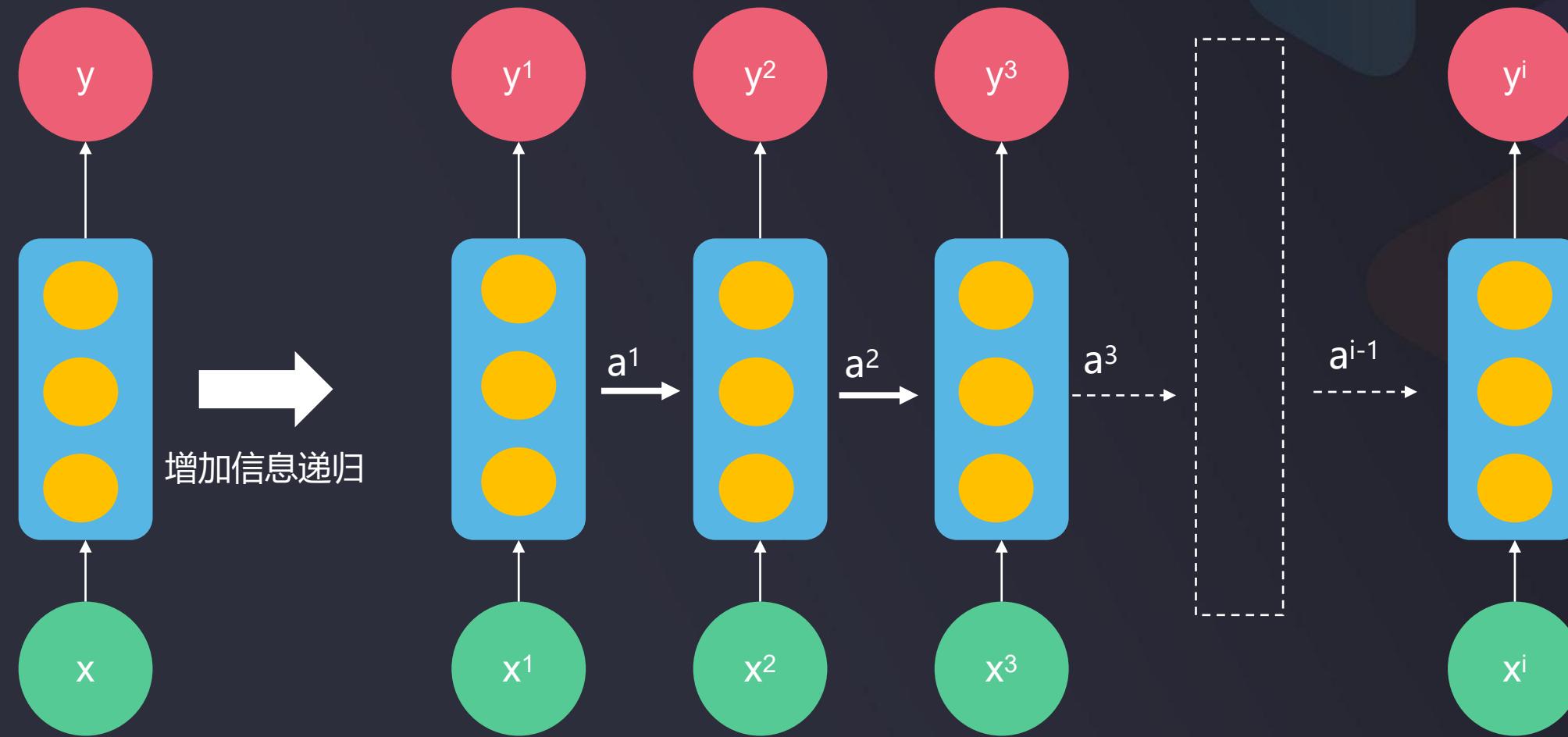
| 循环神经网络 (RNN)

一类以序列 (sequence) 数据为输入，在序列的演进方向进行递归 (recursion) 的神经网络。



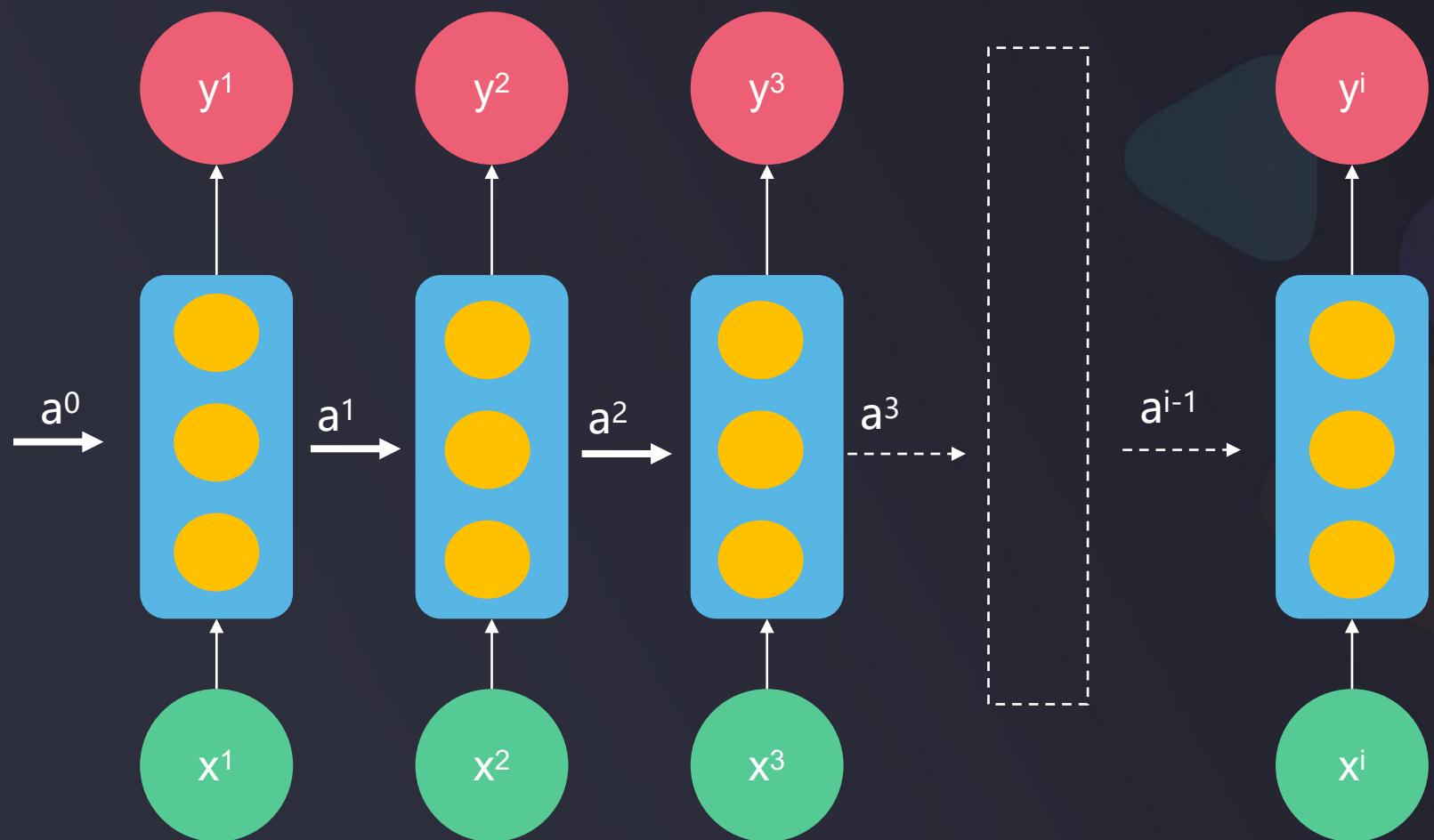
mlp简化结构

| 循环神经网络 (RNN)



mlp简化结构

前部序列的信息经处理后，作为输入信息传递到后部序列



知识巩固

如果建立一个简单的RNN模型，输入是一维的数字序列，RNN有一个隐藏层，包含3个神经元，输出也为一维的数字序列，建立这样的模型，需要训练的参数有多少？

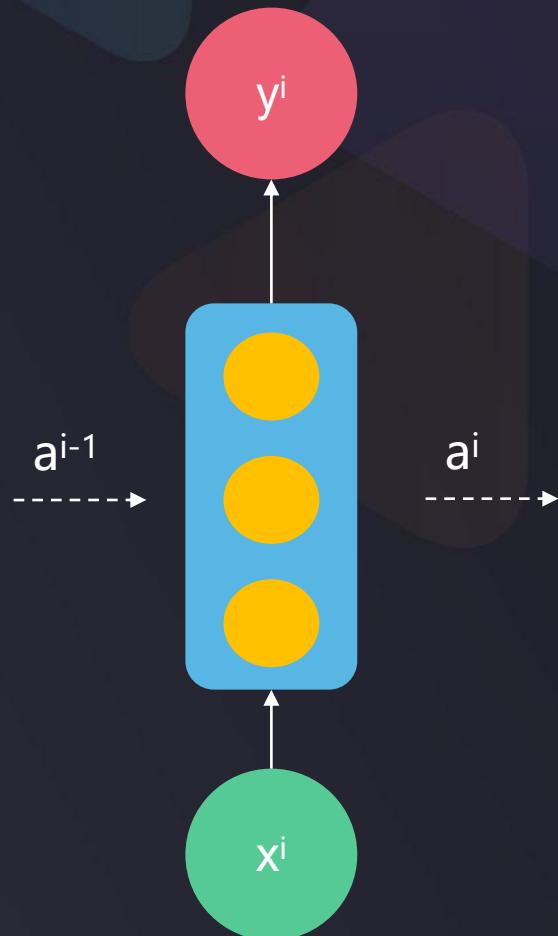
提示：训练参数其实就是 w 、 b ，思考其具体元素个数。

$$a^1 = g_a(w_{aa}a^0 + w_{ax}x^1 + b_a)$$

$$y^1 = g_y(w_{ya}a^0 + b_y)$$

$$a^i = g_a(w_{aa}a^{i-1} + w_{ax}x^i + b_a)$$

$$y^i = g_y(w_{ya}a^{i-1} + b_y)$$





Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构：LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一）RNN预测股价
 - 6 --实战（二）LSTM文本生成

| 现实案例思考

自动寻找语句中的人名：

- ① Teacher of this course is Flare Zhao.
- ② I can see the flare of the fire.

Teachers of this course are Flare Zhao and Charlie Chu.

RNN处理字符串：转化字典

词汇数值化：建立一个词汇-数值的字典，然后把输入词汇转化为数值数据

Teachers of this course are Flare Zhao and Charlie Chu.

$$\left\{ \begin{array}{l} a : 1 \\ am : 2 \\ \dots : \dots \\ jack : 100 \\ \dots : \dots \\ flare : 1000 \\ teachers : 1001 \\ course : 1002 \\ \dots : \dots \end{array} \right\}$$

一部每页只有一个单词的字典，页码就是单词对应的数值

RNN处理字符串：one-hot向量格式

不同字符之间不存在定量关系，因此需要通过one-hot向量格式的数据来表达字符信息

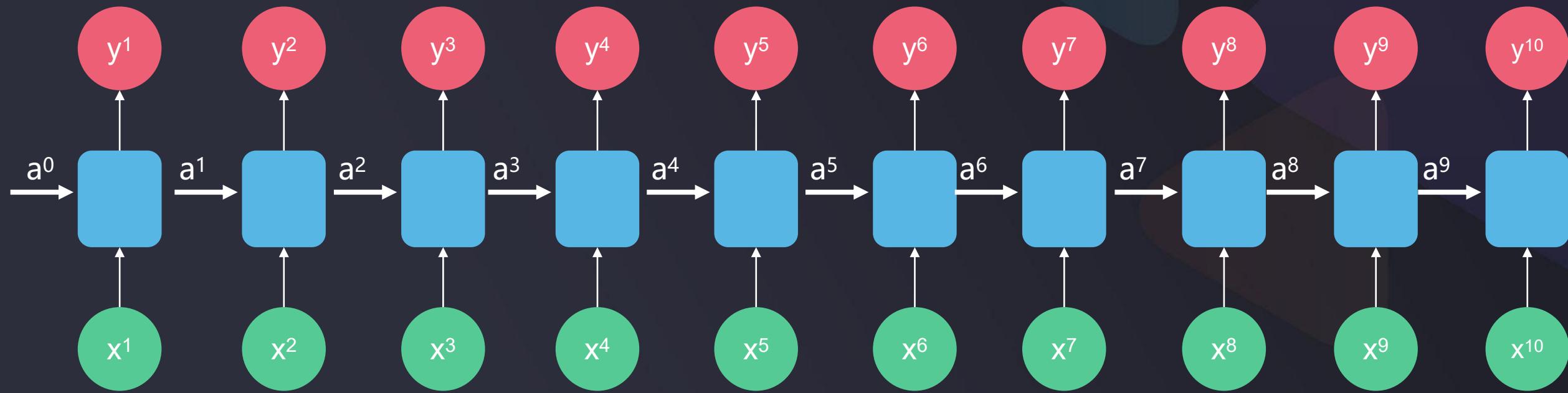
<i>a</i>	:	1
<i>of</i>	:	2
...	:	...
<i>jack</i>	:	100
...	:	...
<i>flare</i>	:	1000
<i>teachers</i>	:	1001
<i>course</i>	:	1002
...	:	...

字符*a* 与 *jack*没有量化关系，但数字1和100则有

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

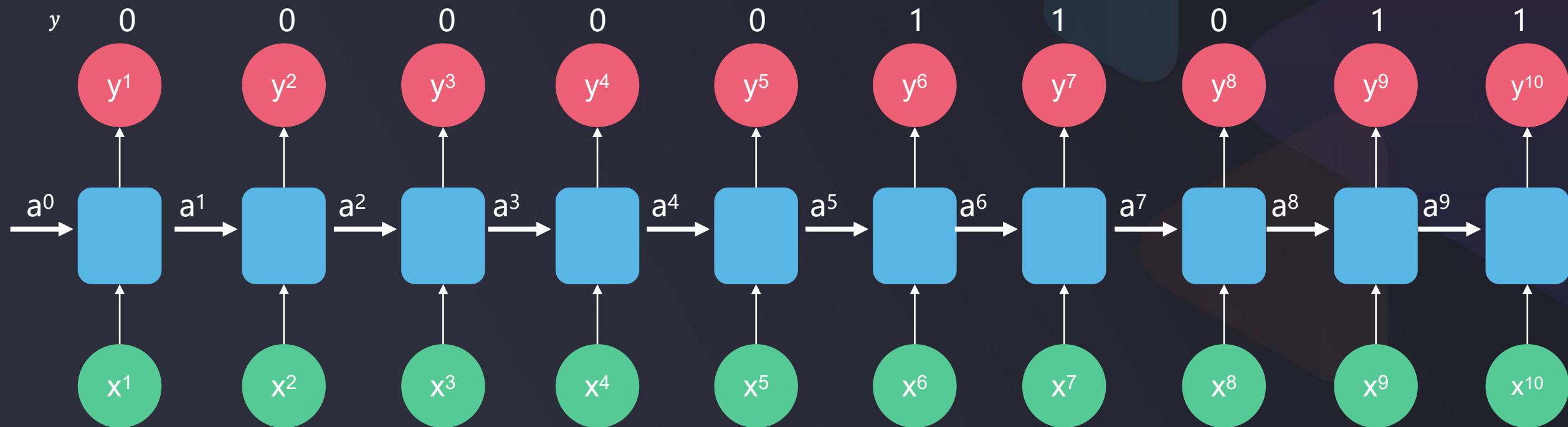
a: 1

RNN字符预测



Teachers of this course are Flare Zhao and Charlie Chu.

RNN字符预测



Teachers of this course are Flare Zhao and Charlie Chu.

目标：最小化总损失， $Min(Loss) = Min(\sum loss(i))$

RNN处理字符串：转化字典的另外一种方式

对每个字母建立字典

Teachers of this course are Flare Zhao and Charlie Chu.

$$\left\{ \begin{array}{l} a : 1 \\ am : 2 \\ \dots \\ jack : 100 \\ \dots : \dots \\ rose : 1000 \\ have : 1001 \\ course : 1002 \\ \dots : \dots \end{array} \right\}$$

$$\left\{ \begin{array}{l} a : 1 \\ b : 2 \\ c : 3 \\ \dots : \dots \\ z : 26 \\ A : 27 \\ \dots : \dots \\ 1 : 53 \\ \dots : \dots \end{array} \right\}$$

字典更小，但需要
更复杂的模型结构
去提取重要信息

知识巩固

问题：在字符串处理的任务中，输入给模型的数据是以下的哪种，为什么要使用这种数据格式？

- A、数值，如1359...
- B、普通字符，如acef....
- C、字符对应的one-hot编码数据，如

$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \dots \end{Bmatrix}, \begin{Bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \dots \end{Bmatrix}, \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \dots \end{Bmatrix}, \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \dots \end{Bmatrix}, \dots$$



Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

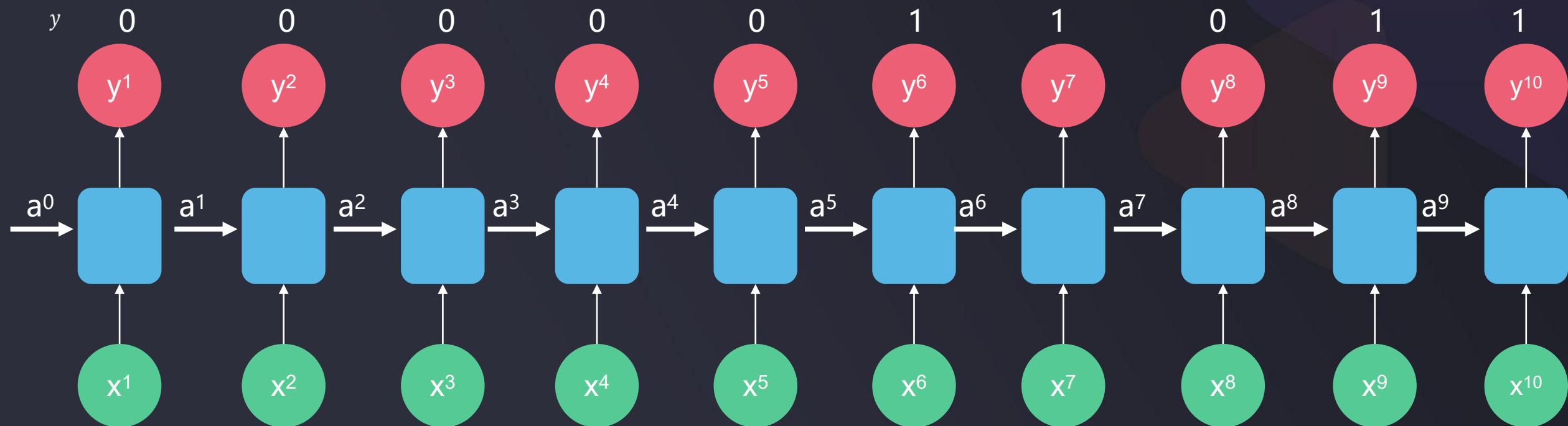
赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构: LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一） RNN预测股价
 - 6 --实战（二） LSTM文本生成

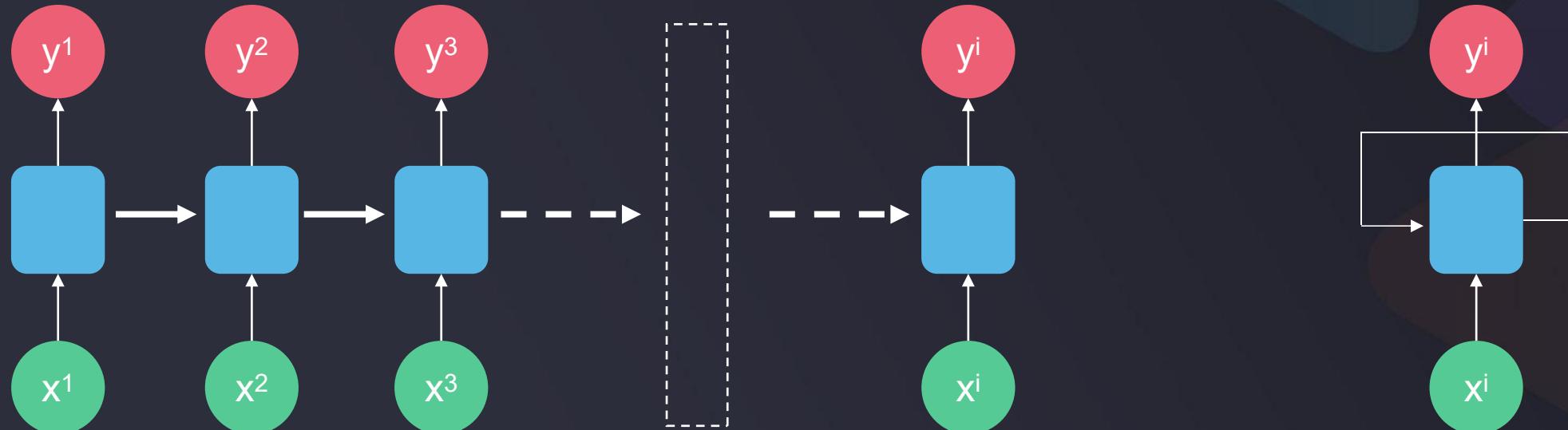
现实问题思考

通过输入字符，判断该字符是否为人名



Teachers of this course are Flare Zhao and Charlie Chu.

| 基本的RNN结构



RNN符号

输入: $x^1, x^2, x^3, \dots, x^i$

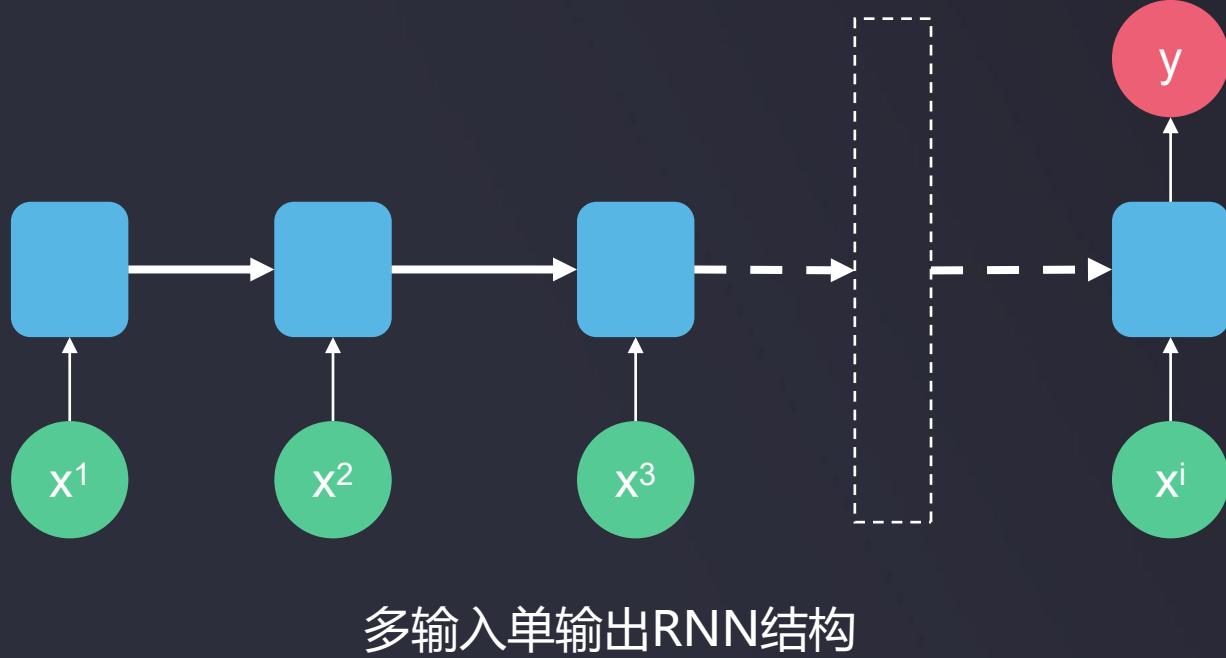
输出: $y^1, y^2, y^3, \dots, y^i$

特点: 一个输入样本本对应一个输出结果

应用: 特定信息查找

多输入多输出且输入输出样本数相同的RNN结构

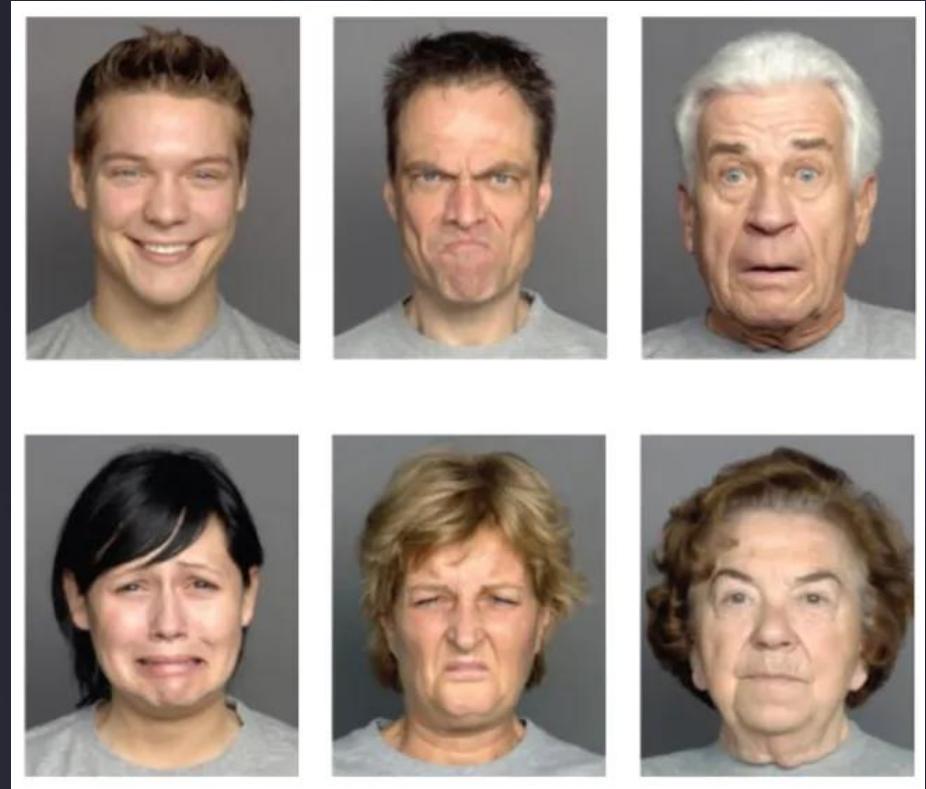
| 基本的RNN结构：多对一



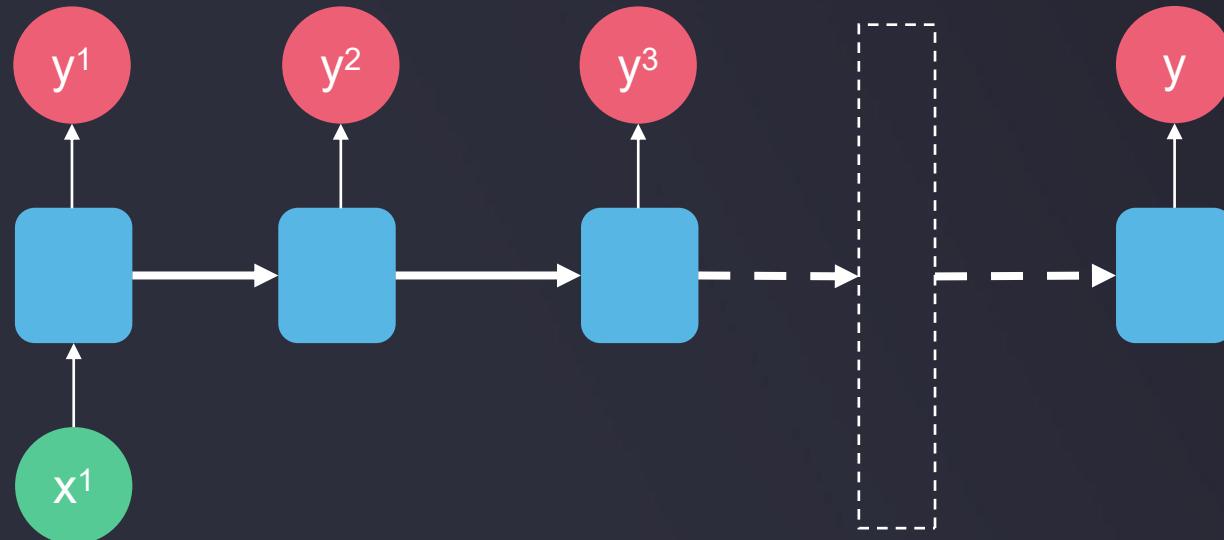
输入: $x^1, x^2, x^3, \dots x^i$ 输出: y

应用: 情感识别

举例: I am angry about this thing.
判断: negative



| 基本的RNN结构：一对多



单输入多输出RNN结构

输入： x

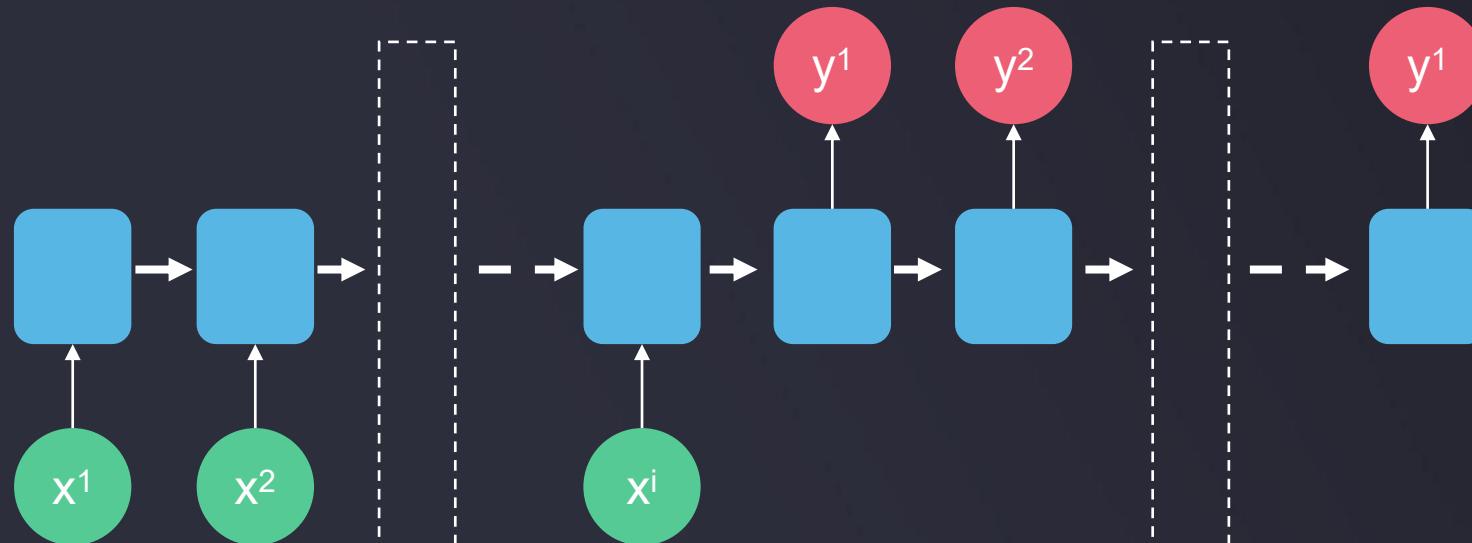
输出： $y^1, y^2, y^3, \dots y^i$

应用：图像字幕、音乐生成、文章生成

应用：序列数据生成器



| 基本的RNN结构：多对多



输入: $x^1, x^2, x^3, \dots x^i$

输出: $y^1, y^2, y^3, \dots y^j$

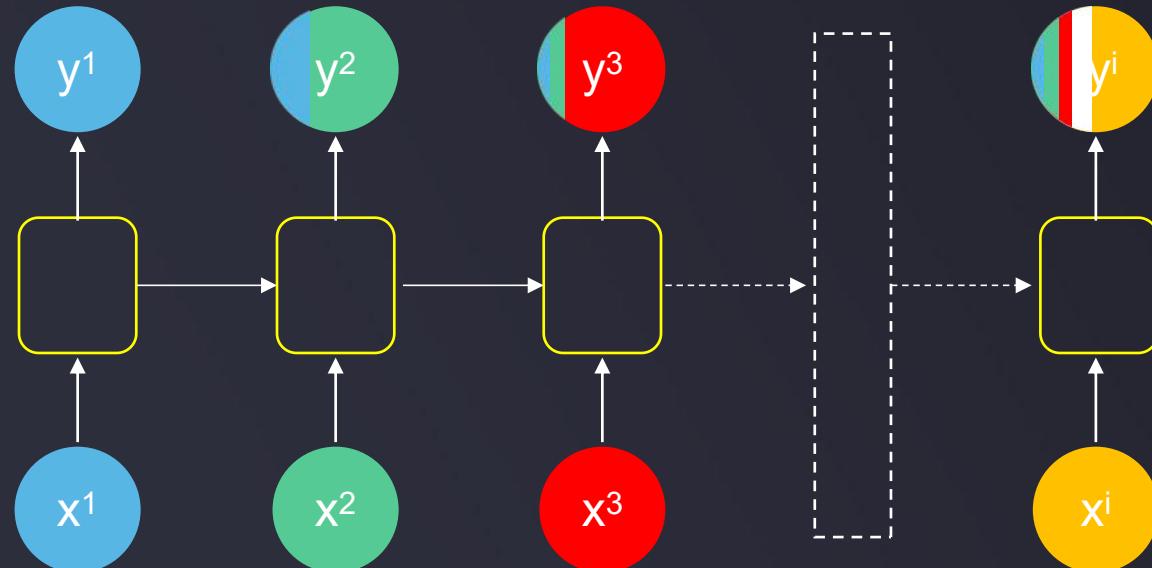
应用: 机器翻译

举例: Do you want to sing with me ?

翻译: 你想和我一起唱歌吗?

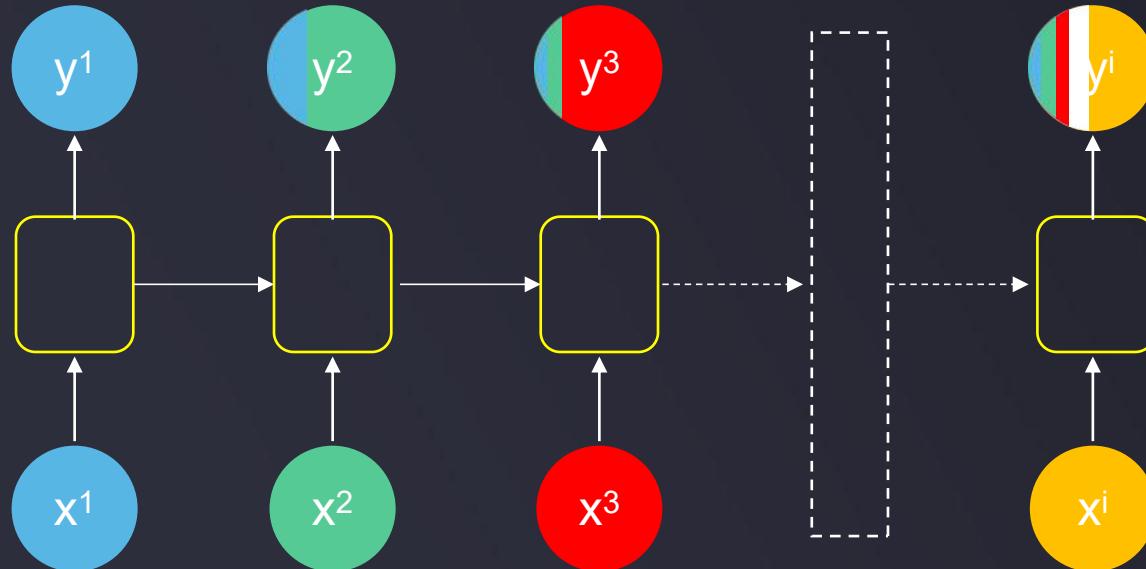


RNN结构中的信息丢失问题



- ① 前部序列信息在传递到后部的同时，信息权重下降，导致重要信息丢失
- ② 反向传播算法进行模型求解时，梯度消失

RNN结构中的信息丢失问题

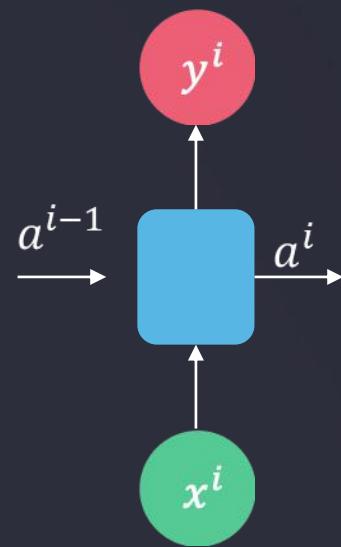


The teacher , who teaches artificial intelligence, ____ very nice.

The teacher , who teaches artificial intelligence, ____ very nice.

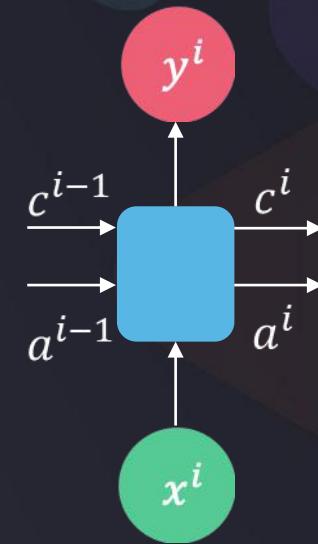
需要寻找一个保留重要
信息决策权重的方法

长短期记忆网络 (LSTM)



普通RNN单元

通过 a^i 传递前部序列信息，距离越远信息丢失越多

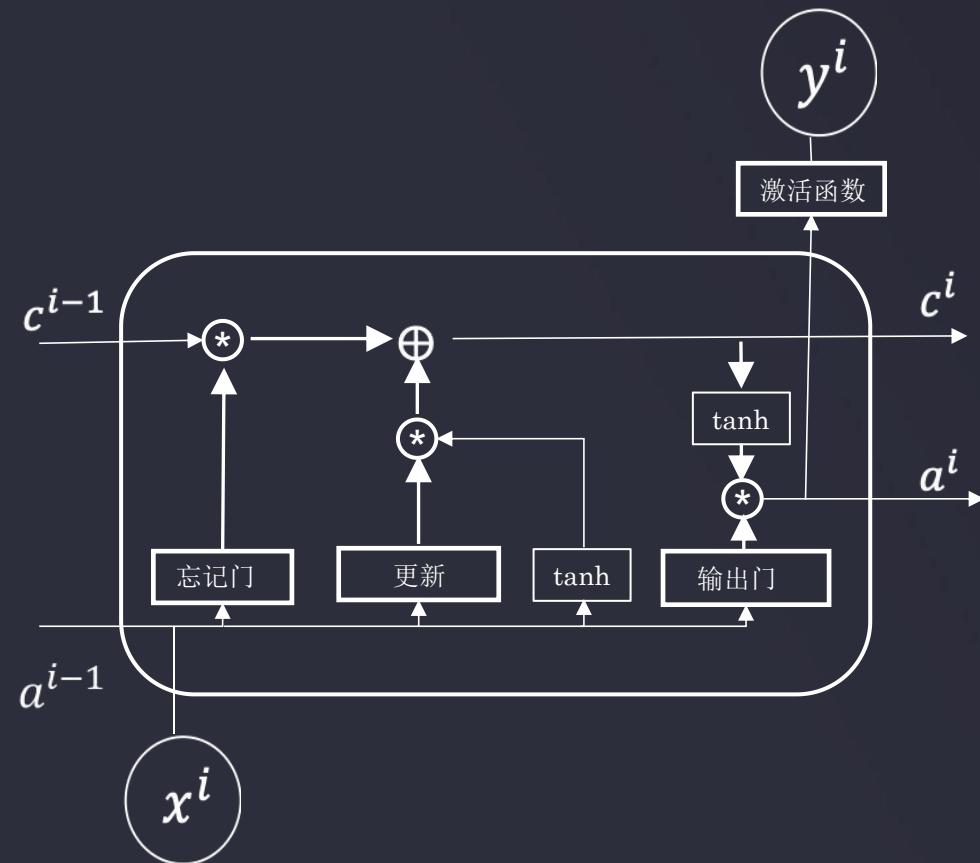


LSTM单元

增加记忆细胞 c^i ，可以传递前部远处重要信息

a^i 实现了序列信息传递，记忆细胞 c^i 则保证了重要信息不易丢失，提高预测准确性

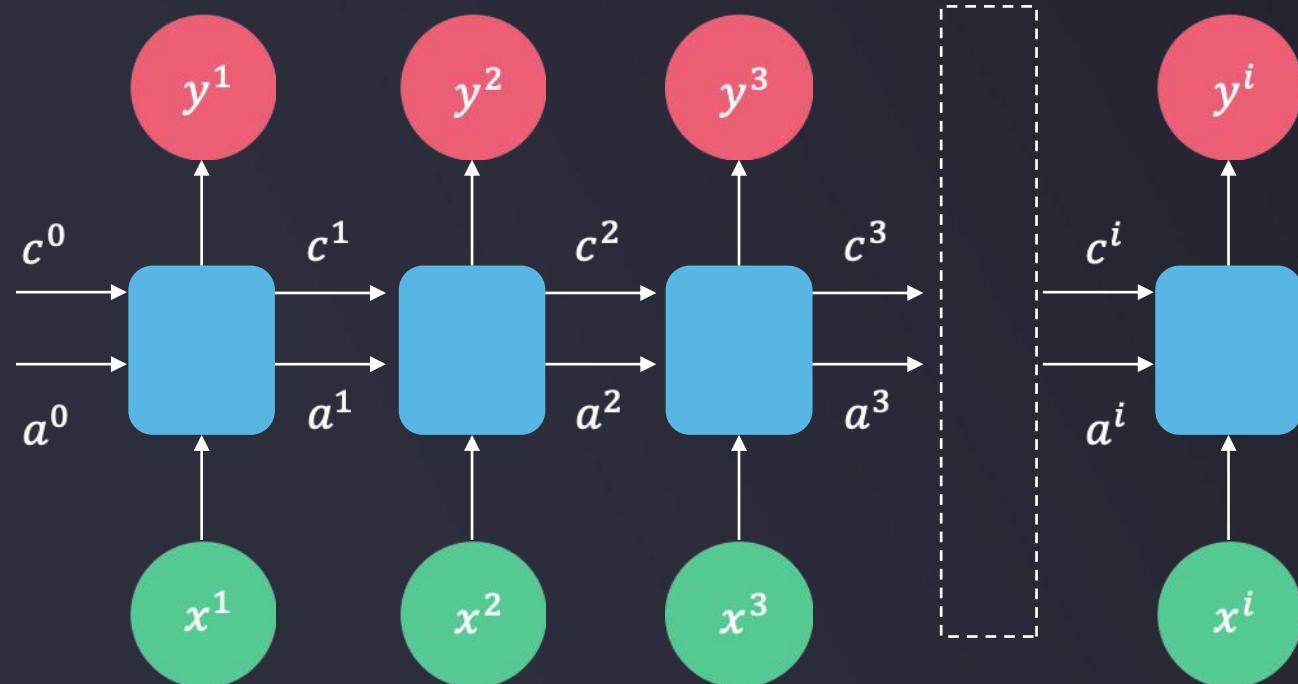
长短期记忆网络 (LSTM)



- ① 忘记门：选择性丢弃 a^{i-1} 与 x^i 中不重要的信息
- ② 更新门：确定给记忆细胞添加哪些信息
- ③ 输出门：筛选需要输出的信息

参考链接：https://blog.csdn.net/dfly_zx/article/details/108670509

长短期记忆网络 (LSTM)

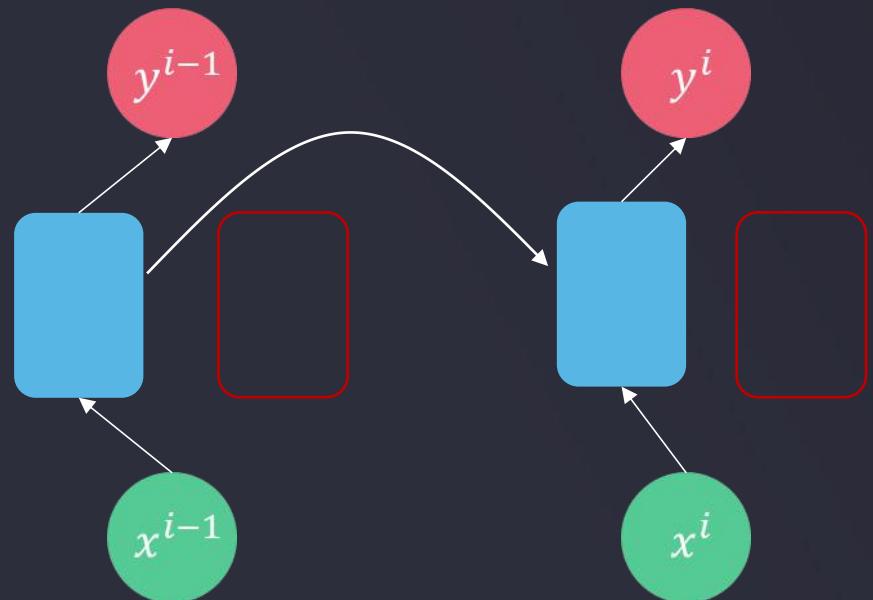


- ① 即使网络结构很深（很多层），前部的重要性息也能在后续预测中保留；
- ② 减少了普通RNN在求解过程中的梯度消失问题

双向循环神经网络 (BRNN)

判断flare是否为人名：

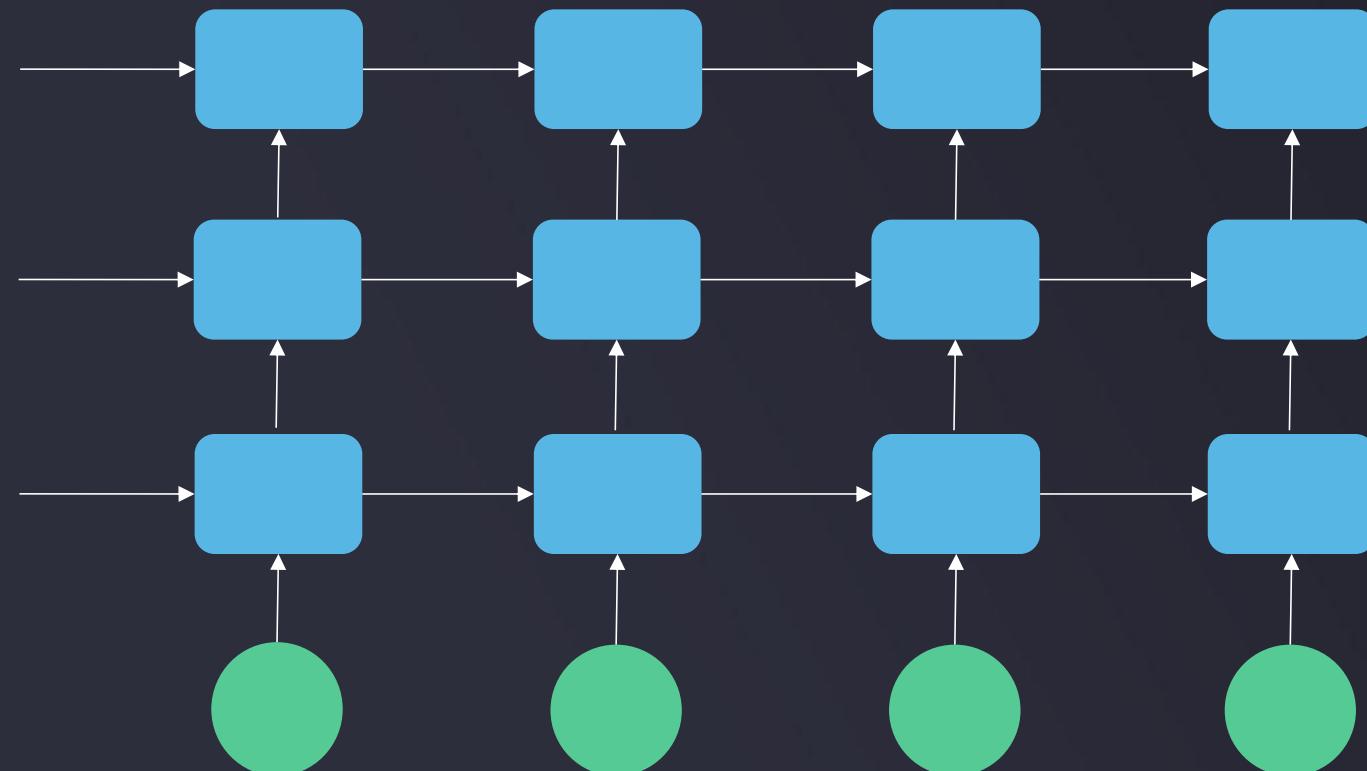
- 1、Teachers of this course are Flare Zhao and Charlie Chu.
- 2、I can see the flare of the fire.



后部序列信息也用于前部信息的预测

| 深层循环神经网络 (DRNN)

对于更难提取的复杂信息，可以把单层RNN叠起来或和mlp结构结合使用



| 知识巩固

对课程中学习的RNN结构进行总结规律，对比不同结构的特点和应用场景



Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

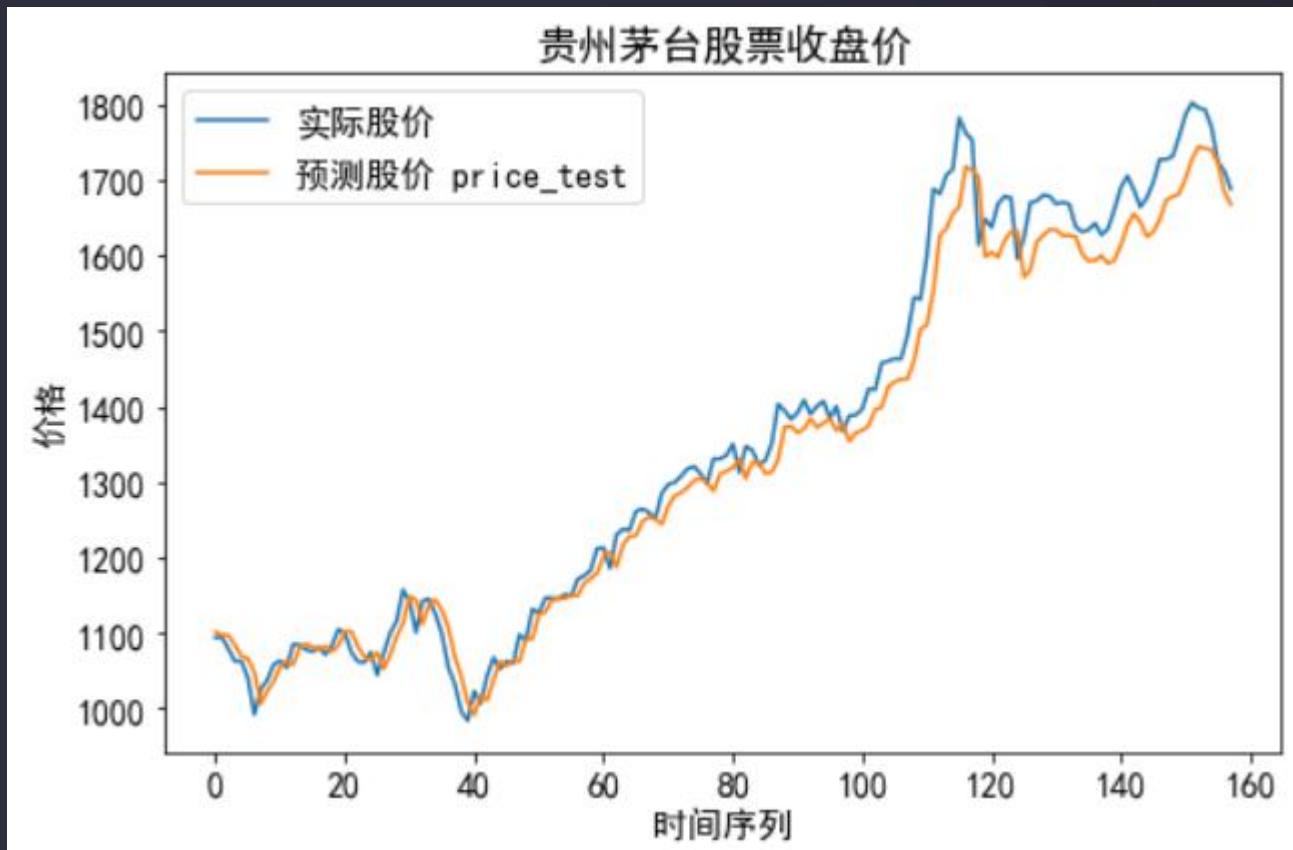
赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构：LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一）RNN预测股价
 - 6 --实战（二）LSTM文本生成

任务一：RNN预测股价

基于task1_data数据，建立rnn模型，预测贵州茅台次日股价。



- 1、完成基本的数据加载、可视化工作；
 - 2、数据预处理：将数据转化为符合RNN模型输入要求的数据；
 - 3、建立RNN模型并训练模型，计算训练集、测试集模型预测r2分数；
 - 4、可视化预测表现；
 - 5、将测试数据预测结果保存到本地csv文件
- 模型结构：单层RNN，5个神经元
 - 每次使用前10个数据预测第11个数据

RNN股价预测-输入数据格式

`Input_shape = (samples, time_steps, features)`

`samples` : 样本数量 (可不填写)

`time_steps` : 序列长度, 即用多少个连续样本预测一个输出

`features` : 每个样本的特征数

举例：假设股票数据样本有1000个，每次用10条数据预测第11条，股票数据为单维度数值，则输入数据的shape为 (1000, 10, 1)

RNN 股价预测

数据加载及展示

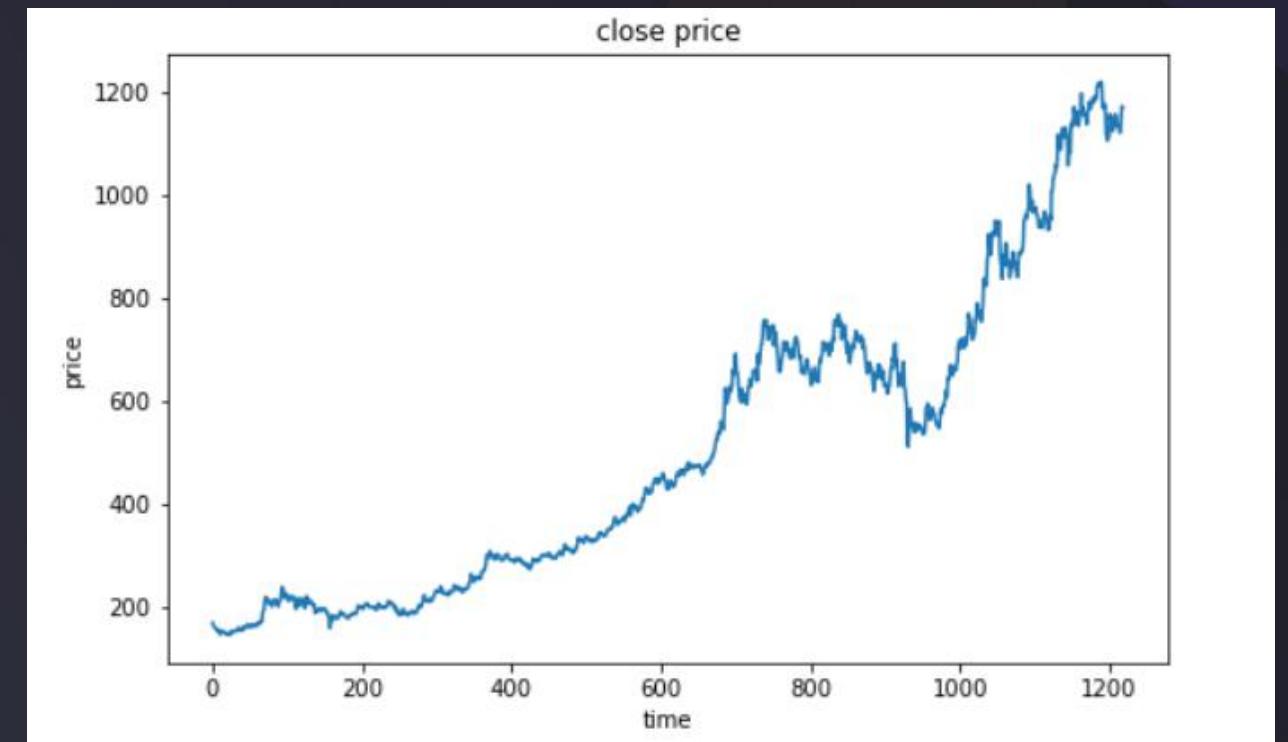
数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#可视化  
%matplotlib inline  
from matplotlib import pyplot as plt  
fig1 = plt.figure(figsize=(8,5))  
plt.plot(price)  
plt.title('close price')  
plt.xlabel('time')  
plt.ylabel('price')  
plt.show()
```



RNN股价预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#提取出符合要求的序列数据
def extract_data(data,time_step):
    X = []
    y = []
    #0,1,2,3...10:11个样本; time_step=10;0,1..9;1,2..10两组 (两组
    #样本)
    for i in range(len(data)-time_step):
        X.append([a for a in data[i:i+time_step]])
        y.append(data[i+time_step])
    X = np.array(X)
    X = X.reshape(X.shape[0],X.shape[1],1)
    return X, y
```

RNN 股价预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#数据提取  
time_step = 10  
X,y = extract_data(price_norm,time_step)  
print(X[0:2,:,:])  
print(y)  
print(X.shape,len(y))
```

```
[[0.13738403]  
 [0.13420248]  
 [0.13088522]  
 [0.13008473]  
 [0.12910113]  
 [0.12646899]  
 [0.12541758]  
 [0.12426432]  
 [0.12622479]  
 [0.12625194]]
```

```
[[0.13420248]  
 [0.13088522]  
 [0.13008473]  
 [0.12910113]  
 [0.12646899]  
 [0.12541758]  
 [0.12426432]  
 [0.12622479]  
 [0.12625194]  
 [0.11904087]]
```

```
[0.11904087]]]  
[0.11904087351116396, 0.12080462726854004, 0.12370802720751509, 0.122941484  
0455, 0.12400650987415923, 0.12218166557100021, 0.12039076957113536, 0.1194  
981364104, 0.11859314951119775, 0.12100134702603434, 0.1202415282378133, 0.  
390817787567, 0.12502405696492494, 0.12406079435913682, 0.12424398399575304  
1227062599957868, 0.12425247006492756, 0.12261207025917050, 0.12470247000480
```

(1209, 10, 1) 1209

RNN股价预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
from keras.models import Sequential  
from keras.layers import Dense, SimpleRNN  
model = Sequential()  
  
#add RNN layer  
model.add(SimpleRNN(units=5, input_shape=(10,1),activation='relu'))  
  
#add output layer  
model.add(Dense(units=1,activation='linear'))  
#configure the model  
model.compile(optimizer='adam',loss='mean_squared_error')  
model.summary()
```

```
Model: "sequential_3"  
=====  
Layer (type)          Output Shape         Param #  
=====  
simple_rnn_3 (SimpleRNN)    (None, 5)           35  
=====  
dense_3 (Dense)        (None, 1)            6  
=====  
Total params: 41  
Trainable params: 41  
Non-trainable params: 0
```

RNN股价预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#模型训练

```
model.fit(X,y,batch_size=30,epochs=200)
```

```
Epoch 1/200  
1209/1209 [=====] - 0s 390us/step - loss: 0.1352  
Epoch 2/200  
1209/1209 [=====] - 0s 154us/step - loss: 0.0421  
Epoch 3/200  
1209/1209 [=====] - 0s 169us/step - loss: 0.0110  
Epoch 4/200  
1209/1209 [=====] - 0s 171us/step - loss: 0.0077  
Epoch 5/200  
1209/1209 [=====] - 0s 174us/step - loss: 0.0066  
Epoch 6/200  
1209/1209 [=====] - 0s 177us/step - loss: 0.0055  
Epoch 7/200  
1209/1209 [=====] - 0s 169us/step - loss: 0.0044
```

RNN 股价预测

数据加载及展示

```
#数据归一化处理  
price_norm = price/max(price)  
print(price_norm)
```

```
0      0.137384  
1      0.134202  
2      0.130885  
3      0.130085
```

数据预处理

模型建立及训练

```
#数据还原  
y_train_predict = model.predict(X)*max(price)  
y_train = [i*max(price) for i in y]  
print(y_train_predict,y_train)
```

模型预测

```
[[ 153.78616 ]  
[ 147.06294 ]  
[ 147.19582 ]  
...  
[ 1124.4689 ]  
[ 1145.0363 ]  
[ 1165.809 ] ] [ 145.1705, 147.3214, 150.8621  
7247, 146.5768, 144.6245, 147.5613, 146.6347  
631, 150.9697, 151.648, 150.7463, 152.1858,
```

结果展示及表现评估

RNN股价预测

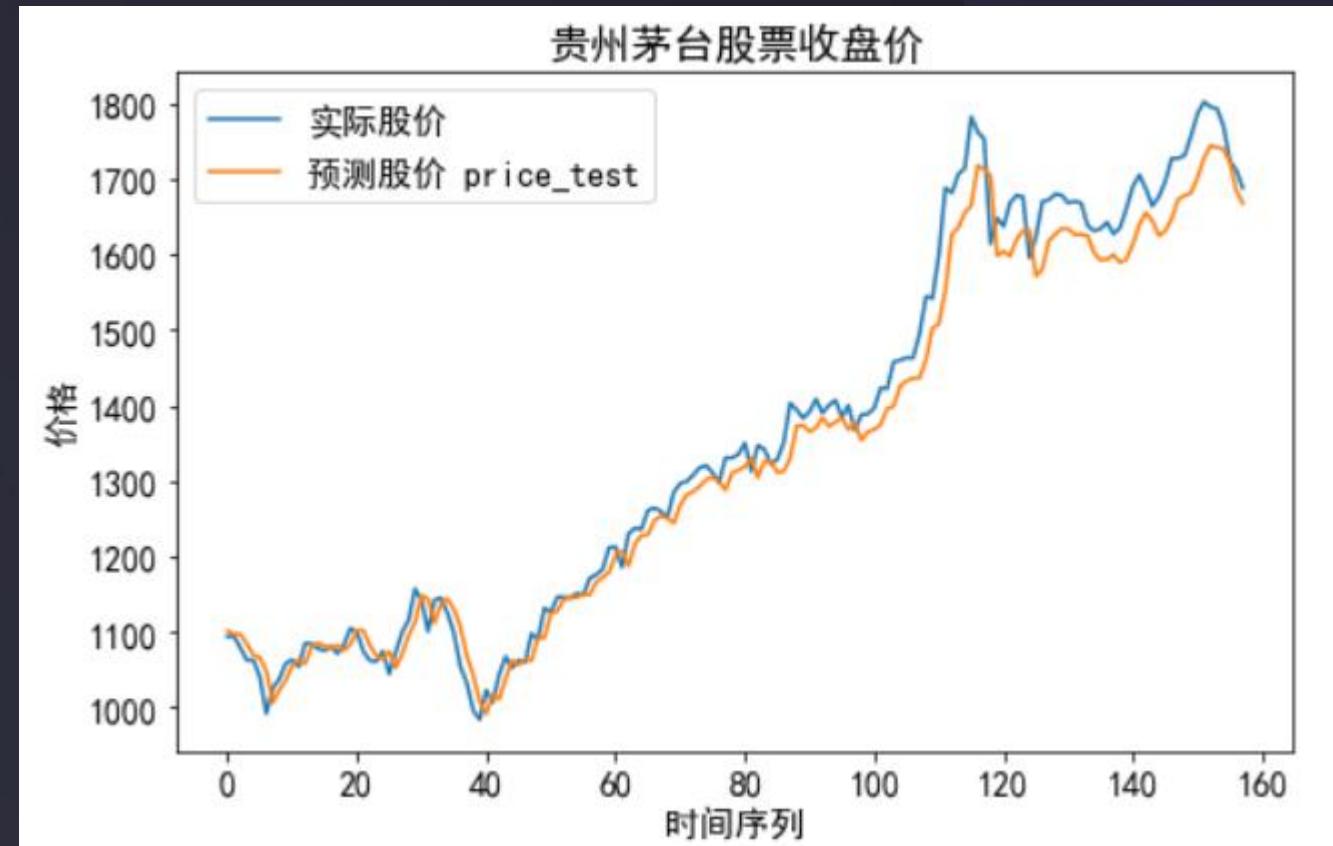
数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示



任务二：LSTM文本生成

基于task2_data，建立LSTM模型，生成文本。

```
Artificial intelligence (AI), --predict next letter is--- s  
rtificial intelligence (AI), s --predict next letter is--- o  
tificial intelligence (AI), so --predict next letter is--- m  
ificial intelligence (AI), som --predict next letter is--- e  
ficial intelligence (AI), some --predict next letter is--- t  
icial intelligence (AI), somet --predict next letter is--- i  
cial intelligence (AI), someti --predict next letter is--- m  
ial intelligence (AI), sometim --predict next letter is--- e  
al intelligence (AI), sometime --predict next letter is--- s  
l intelligence (AI), sometimes --predict next letter is---  
intelligence (AI), sometimes --predict next letter is--- c  
intelligence (AI), sometimes c --predict next letter is--- a  
ntelligence (AI), sometimes ca --predict next letter is--- l  
telligence (AI), sometimes cal --predict next letter is--- l  
elligence (AI), sometimes call --predict next letter is--- e  
lligence (AI), sometimes calle --predict next letter is--- d
```

- 1、加载本地文本数据，生成字典
- 2、数据预处理：将数据转化为符合LSTM模型输入要求的数据，确认数据结构；
- 3、建立LSTM模型，进行模型训练，计算模型在训练、测试数据集的准确率
- 4、预测“ flare is a teacher in ai industry. He obtained his phd in Australia.” 的后续字符
 - 模型结构：单层LSTM，30神经元
 - 每次使用前30个字符预测第31个字符

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#文本数据加载  
data = open('task2_data').read()  
#移除换行符  
data = data.replace("\n","").replace("\r","")  
print(data)
```

Artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and animals. The field of study is the design and creation of intelligent agents: any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Artificial intelligence, is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and animals. The field of study is the design and creation of intelligent agents: any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#字符去重处理  
letters = list(set(data))  
print(letters)  
features = len(letters)  
print(features)
```

```
['d', 'l', 'e', 'g', 'h', 'z', ' ', 'k', 'I',  
'f', 'o', '.', 'r', ' ', 'y', ':', 'p', 'A',  
'b', 'm', 'c', '(', ')', 's', 'n', ',',  
'x', 'a', 'v', 't', 'i', 'L']
```

33

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#建立字典  
#int to char  
int_to_char = {a:b for a,b in enumerate(letters)}  
print(int_to_char)  
#char to int  
char_to_int = {b:a for a,b in enumerate(letters)}  
print(char_to_int)
```

```
{0: 'd', 1: 'l', 2: 'e', 3: 'g',  
6: "'", 7: 'k', 8: 'I', 9: 'f',  
'.', 12: 'r', 13: ' ', 14: 'y',  
'p', 17: 'A', 18: 'b', 19: 'm',
```

```
{'d': 0, 'l': 1, 'e': 2, 'g': 3, 'h': 4,  
':': 6, 'k': 7, 'I': 8, 'f': 9, 'o': 10,  
'r': 12, ' ': 13, 'y': 14, '': 15,  
'A': 17, 'b': 18, 'm': 19, 'c': 20,
```

字符串预处理



LSTM字符数据预
处理.ipynb

```
from keras.utils import to_categorical
#滑动窗口提取数据(长字符串切片一次提取出来)
def extract_data(data, slide):
    x = []
    y = []
    for i in range(len(data) - slide):
        x.append([a for a in data[i:i+slide]])
        y.append(data[i+slide])
    return x,y

#字符到数字的批量转化(根据字典实现字符到数字的转化)
def char_to_int_Data(x,y, char_to_int):
    x_to_int = []
    y_to_int = []
    for i in range(len(x)):
        x_to_int.append([char_to_int[char] for char in x[i]])
        y_to_int.append([char_to_int[char] for char in y[i]])
    return x_to_int, y_to_int

#实现输入字符文本的批量处理, 输入整个字符串、滑动窗口大小、转化字典
def data_preprocessing(data, slide, num_letters, char_to_int):
```

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#完成字符串预处理

```
X, y = data_preprocessing(data,time_step,features,char_to_int)
```

data: 待处理的字符串

time step : 序列的长度

features : 样本的特征数

char to int : 字符转数字的字典

X: 转化为one-hot格式的数组

y : 转化为字符对应数值的列表

#确认数据维度

```
print(X[0])
```

```
print(X.shape)
```

```
print(len(y))
```

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

(11160, 30, 33)
11160

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#输出结果格式转化  
y_train_category = to_categorical(y_train,features)  
print(y_train_category)  
print(y_train_category.shape)
```

```
[[0. 0. 0. ... 0. 1. 0.]  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 0. 1. ... 0. 0. 0.]  
 ...  
 [0. 0. 0. ... 0. 0. 0.]  
 [0. 1. 0. ... 0. 0. 0.]  
 [0. 0. 0. ... 0. 1. 0.]]  
(10044, 33)
```

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#建立LSTM模型

```
from keras.models import Sequential  
from keras.layers import Dense,LSTM
```

```
model = Sequential()
```

```
model.add(LSTM(units=30,input_shape=(X_train.shape[1],X_train.shape[2]),activation='relu'))
```

```
model.add(Dense(units=features,activation='softmax'))  
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 30)	7680

dense_3 (Dense)	(None, 33)	1023
-----------------	------------	------

Total params: 8,703

Trainable params: 8,703

Non-trainable params: 0

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#参数配置与训练

```
model.compile(optimizer='adam',loss='categorical_crossentropy',  
metrics=['accuracy'])  
model.fit(X_train,y_train_category,batch_size=1000,epochs=50)
```

```
Epoch 1/50  
10044/10044 [=====] - 3s 275us/step - loss: 3.4926 - accuracy: 0.0975  
Epoch 2/50  
10044/10044 [=====] - 2s 226us/step - loss: 3.4517 - accuracy: 0.1186  
Epoch 3/50  
10044/10044 [=====] - 2s 206us/step - loss: 3.3782 - accuracy: 0.1186  
Epoch 4/50  
10044/10044 [=====] - 2s 206us/step - loss: 3.1972 - accuracy: 0.1186  
Epoch 5/50  
10044/10044 [=====] - 2s 198us/step - loss: 3.0740 - accuracy: 0.1191  
Epoch 6/50  
10044/10044 [=====] - 2s 232us/step - loss: 2.9873 - accuracy: 0.1509  
Epoch 7/50  
10044/10044 [=====] - 2s 224us/step - loss: 2.9343 - accuracy: 0.1525
```

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

#新字符预测

```
y_train_predict = model.predict_classes(X_train)  
print(y_train_predict)
```

```
[31 18 2 ... 13 1 31]
```

#预测结果转化为字符

```
y_train_predict_char = [int_to_char[i] for i in  
y_train_predict]  
print(y_train_predict_char)
```

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

#新字符串预测

```
new_letters = 'Artificial intelligence (AI), sometimes called machine  
intelligence, is intelligence demonstrated by machines'
```

```
X_new, y_new =
```

```
data_preprocessing(new_letters,time_step,features,char_to_int)
```

```
y_new_predict = model.predict_classes(X_new)
```

```
print(y_new_predict)
```

```
[24 10 19 2 30 31 19 2 24 13 20 28 1 1 2 0 13 19 28 20 4 31 25 2  
13 31 25 30 2 1 1 31 3 2 25 20 2 26 13 31 24 13 31 25 30 2 1 1  
31 3 2 25 20 2 13 0 2 19 10 25 24 30 12 28 30 2 0 13 18 14 13 19  
28 20 4 31 25 2 2]
```

#结果转化为字符

```
y_new_predict_char = [int_to_char[i] for i in y_new_predict]
```

```
print(y_new_predict_char)
```

```
['s', 'o', 'm', 'e', 't', 'i', 'm', 'e', 's', ' ', 'c', 'a', 'l', 'l', 'e',  
'h', 'i', 'n', 'e', ' ', 'i', 'n', 't', 'e', 'l', 'l', 'i', 'g', 'e', 'n',  
's', ' ', 'i', 'n', 't', 'e', 'l', 'l', 'i', 'g', 'e', 'n', 'c', 'e', ' ',  
's', 't', 'r', 'a', 't', 'e', 'd', ' ', 'b', 'y', ' ', 'm', 'a', 'c', 'h', '']
```

LSTM文本生成

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示

Artificial intelligence (AI), --predict next letter is--- s
rtificial intelligence (AI), s --predict next letter is--- o
tificial intelligence (AI), so --predict next letter is--- m
ifificial intelligence (AI), som --predict next letter is--- e
ficial intelligence (AI), some --predict next letter is--- t
icial intelligence (AI), somet --predict next letter is--- i
cial intelligence (AI), someti --predict next letter is--- m
ial intelligence (AI), sometim --predict next letter is--- e
al intelligence (AI), sometime --predict next letter is--- s
l intelligence (AI), sometimes --predict next letter is---
intelligence (AI), sometimes --predict next letter is--- c
intelligence (AI), sometimes c --predict next letter is--- a
ntelligence (AI), sometimes ca --predict next letter is--- l
telligence (AI), sometimes cal --predict next letter is--- l
elligence (AI), sometimes call --predict next letter is--- e
lligence (AI), sometimes calle --predict next letter is--- d
ligence (AI), sometimes called --predict next letter is---
igence (AI), sometimes called --predict next letter is--- m
gence (AI), sometimes called m --predict next letter is--- a



Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

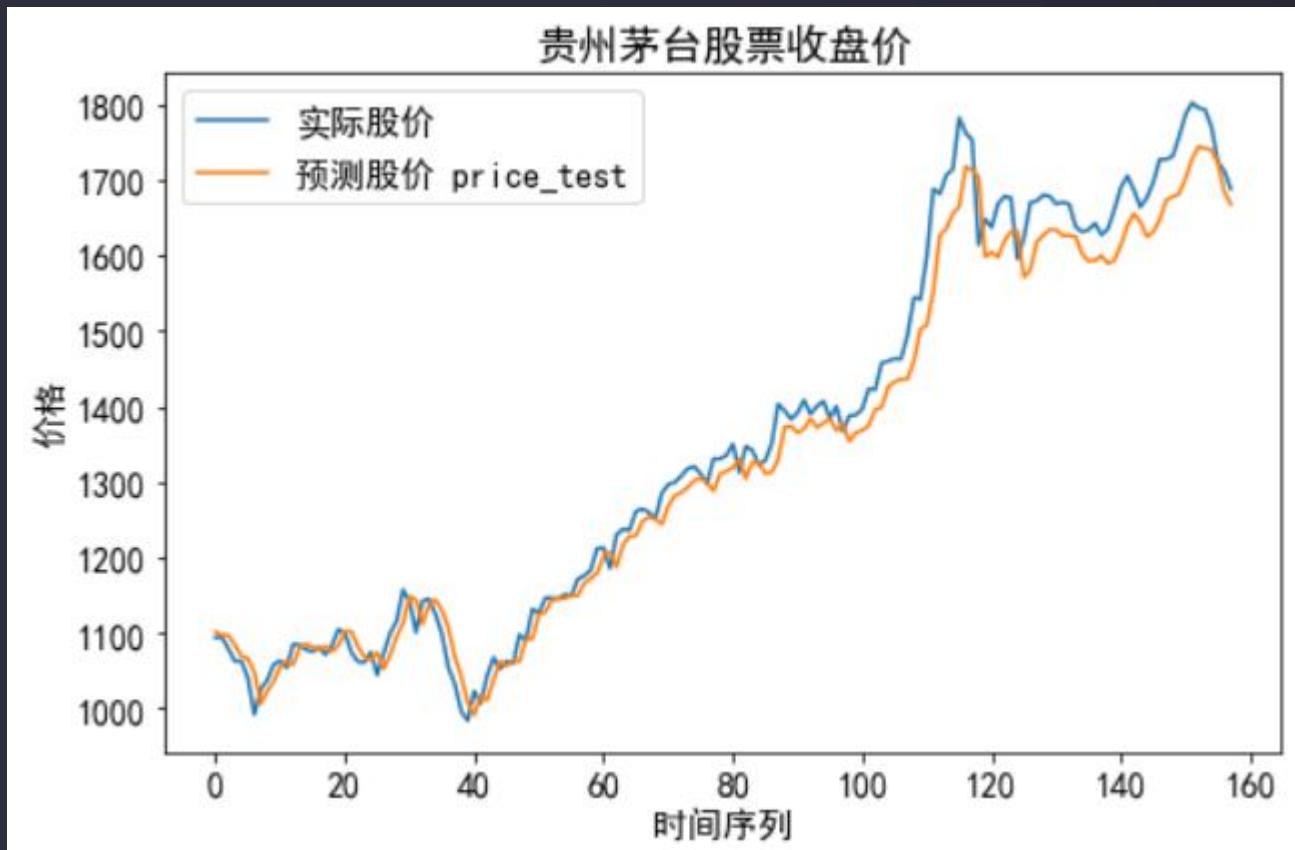
赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构：LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一） RNN预测股价
 - 6 --实战（二） LSTM文本生成

任务一：RNN预测股价

基于task1_data数据，建立rnn模型，预测贵州茅台次日股价。



- 1、完成基本的数据加载、可视化工作；
 - 2、数据预处理：将数据转化为符合RNN模型输入要求的数据；
 - 3、建立RNN模型并训练模型，计算训练集、测试集模型预测r2分数；
 - 4、可视化预测表现；
 - 5、将测试数据预测结果保存到本地csv文件
- 模型结构：单层RNN，5个神经元
 - 每次使用前10个数据预测第11个数据



Python3人工智能入门+实战提升：深度学习

Chapter 4 深度学习之循环神经网络

赵辛

Chapter 4 深度学习之循环神经网络

-
- 1 --序列模型之循环神经网络RNN
 - 2 --rnn处理字符串
 - 3 --多样的RNN结构：LSTM、BRNN、DRNN
 - 4 --实战准备
 - 5 --实战（一）RNN预测股价
 - 6 --实战（二）LSTM文本生成

任务二：LSTM文本生成

基于task2_data，建立LSTM模型，生成文本。

```
Artificial intelligence (AI), --predict next letter is--- s  
rtificial intelligence (AI), s --predict next letter is--- o  
tificial intelligence (AI), so --predict next letter is--- m  
ificial intelligence (AI), som --predict next letter is--- e  
ficial intelligence (AI), some --predict next letter is--- t  
icial intelligence (AI), somet --predict next letter is--- i  
cial intelligence (AI), someti --predict next letter is--- m  
ial intelligence (AI), sometim --predict next letter is--- e  
al intelligence (AI), sometime --predict next letter is--- s  
l intelligence (AI), sometimes --predict next letter is---  
intelligence (AI), sometimes --predict next letter is--- c  
intelligence (AI), sometimes c --predict next letter is--- a  
ntelligence (AI), sometimes ca --predict next letter is--- l  
telligence (AI), sometimes cal --predict next letter is--- l  
elligence (AI), sometimes call --predict next letter is--- e  
lligence (AI), sometimes calle --predict next letter is--- d
```

- 1、加载本地文本数据，生成字典
- 2、数据预处理：将数据转化为符合LSTM模型输入要求的数据，确认数据结构；
- 3、建立LSTM模型，进行模型训练，计算模型在训练、测试数据集的准确率
- 4、预测“ flare is a teacher in ai industry. He obtained his phd in Australia.” 的后续字符
 - 模型结构：单层LSTM，30神经元
 - 每次使用前30个字符预测第31个字符



Python3人工智能入门+实战提升：深度学习

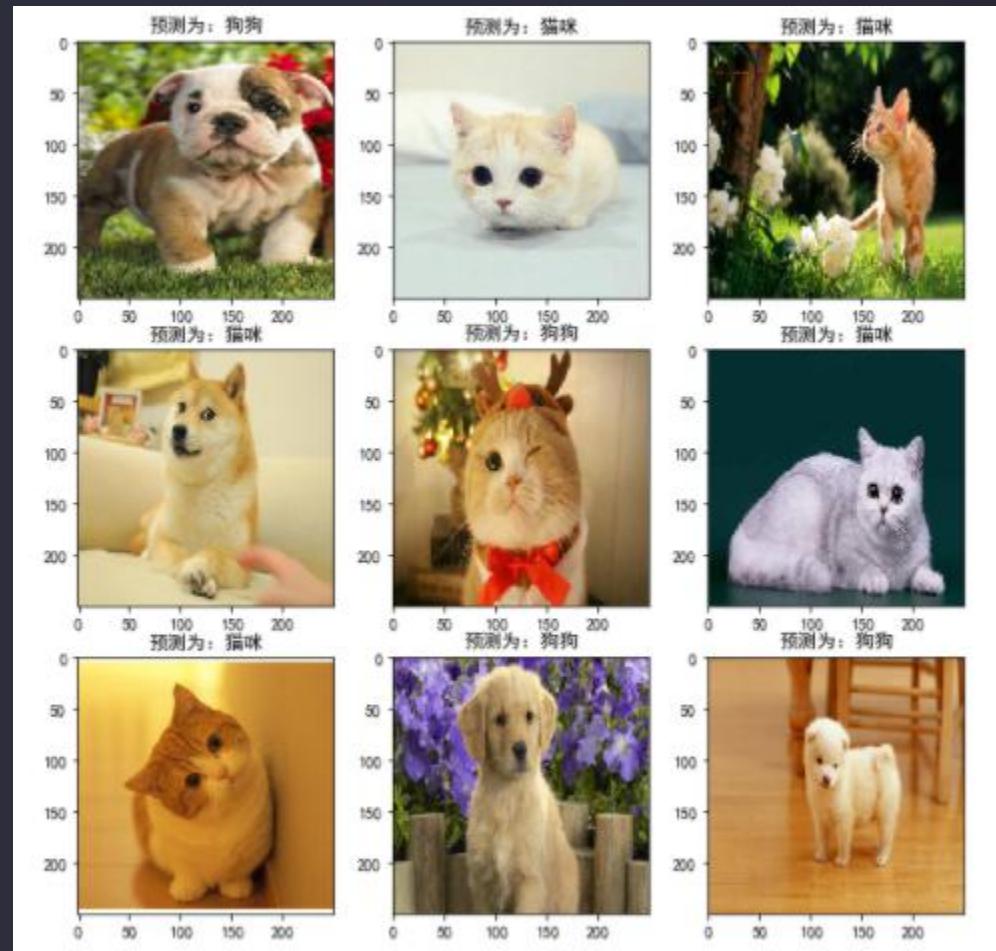
Chapter 5 迁移学习与混合模型

赵辛

Chapter 5 迁移学习与混合模型

-
- 1 --迁移学习与在线学习
 - 2 --混合模型
 - 3 --实战准备
 - 4 --实战（一）mlp模型迁移数据预测
 - 5 --实战（二）少样本区分奇特草莓

|第三章实战猫狗识别



基于图片数据，建立模型，识别猫狗

两种方法：

- ① 建立新的CNN模型
- ② VGG16+MLP结构，建立模型

|第三章实战猫狗识别

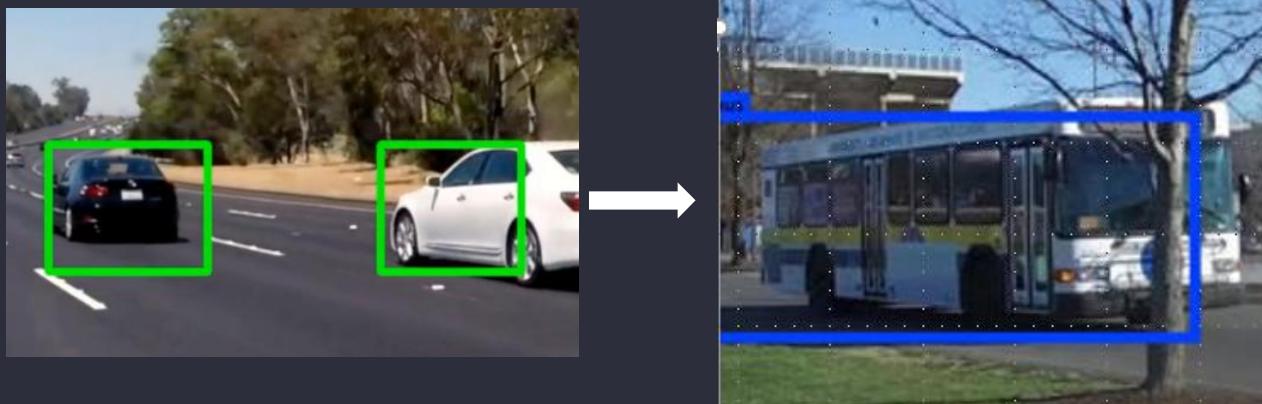
模型	全新CNN结构	VGG16+MLP
训练参数	506,017	250,901
训练时间	700s	20s
训练集准确率	99%	99.70%
测试集准确率	75.30%	95%
网络图片准确率	78%	100%



迁移学习

迁移学习 (transfer learning) 是一种机器学习方法，就是把为任务 A 开发的模型作为初始点，重新使用在任务 B 模型开发的过程中。

目的：运用已有的知识帮助发现新信息中的规律，并将其应用到解决不同但相似领域的问题。



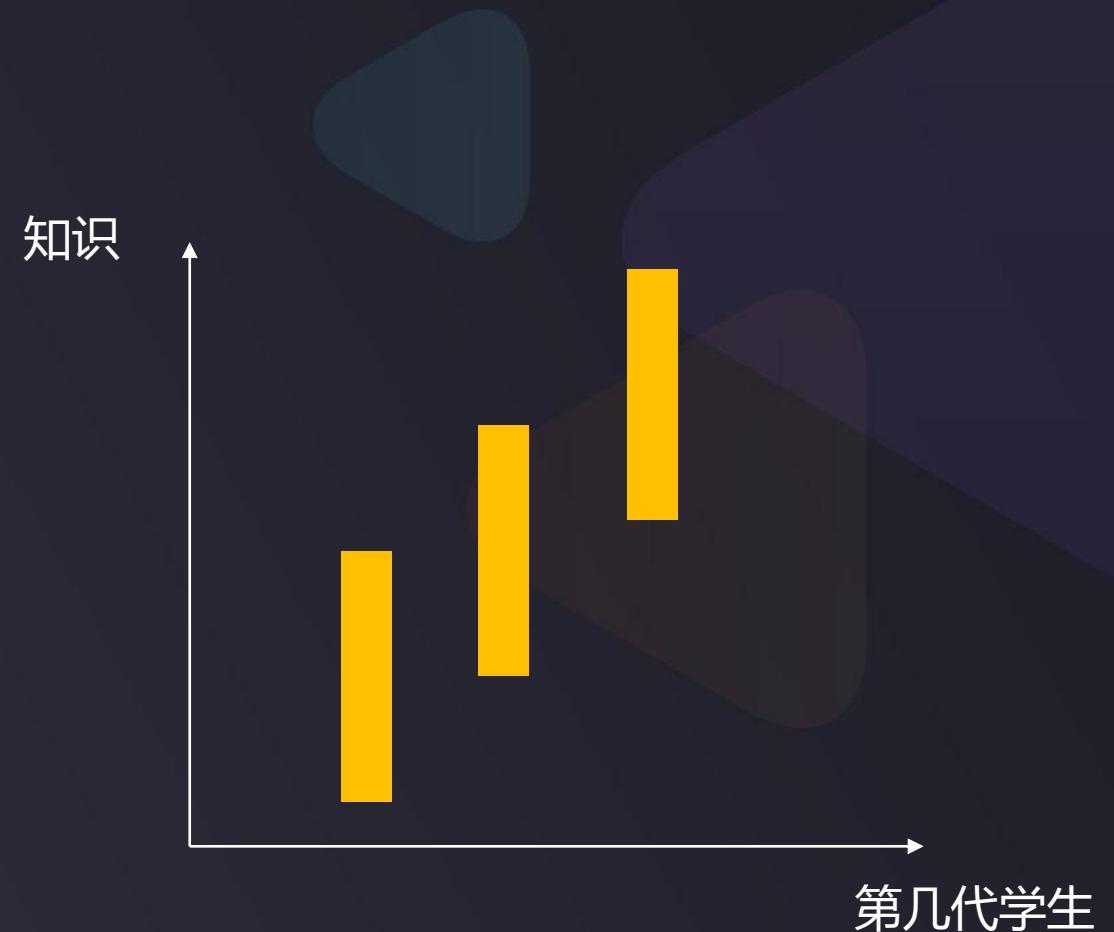
轿车识别模型应用到公交车识别

美式英
语翻译 → 英式英
语翻译

美式英语翻译系统模型应用于英式英语翻译系统

迁移学习

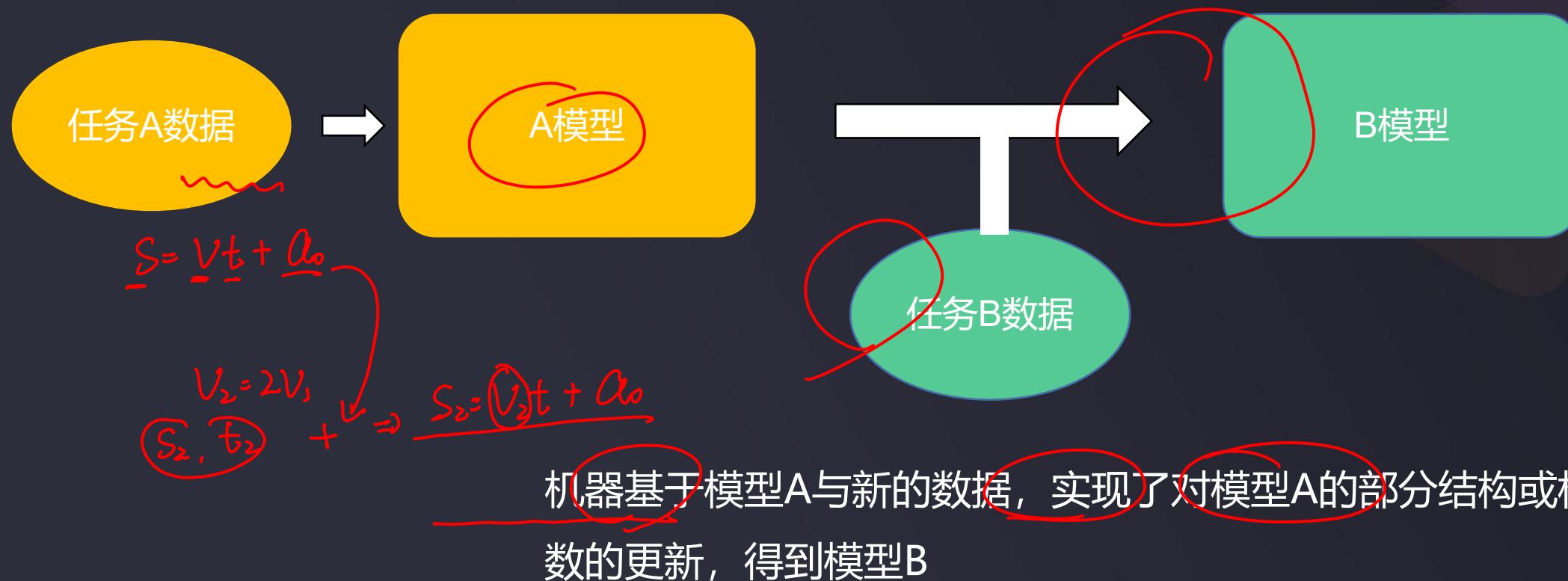
老师传授知识和技能的过程像迁移学习



- ① 第一代老师根据经验总结规律，教授给学生；
- ② 学生成为第二代老师，在前人的基础上发现新的规律，教授给下一代学生；
- ③ 随着时间推移，对的信息流传下来，复杂的规律被找到

迁移学习

通过任务A的数据，模型A学习了核心的模型结构、权重系数(‘weights’)



迁移学习的三种实现方法

① 特征提取

使用模型A，对原数据进行特征信息提取（通常会移除模型A的预测输出层）

② 结构引用

直接使用模型A的结构，利用新数据对其重新重新/二次训练，实现权重系数更新

③ 部分训练

直接使用模型A的结构，利用新数据重新/二次训练，更新其指定的部分权重系数

迁移学习的三种实现方法



没有固定的方法，可根据情况灵活运用

迁移学习的价值

对样本数据量需求相对较小

时间

数据

表现

待更新参数少，训练速度快

通常会比很多全新的模型有更好的表现

模型	全新CNN结构	VGG16+MLP
训练参数	506,017	250,901
训练时间	700s	20s
训练集准确率	99%	99.70%
测试集准确率	75.30%	95%
网络图片准确率	78%	100%

|现实问题思考：实时信息的价值



同一个商品

- 相同用户在不同时期的付费意愿是不同的
- 不同用户在相同时期的付费意愿也是不同的

一个根据实时用户付费意愿、动态更新的
模型，可以帮助确定更合理的价格！

|在线学习 (Online learning)

在线学习，即将新数据输入给已经训练好的模型，实现模型更新，发现新的价值信息。

价值：对于新数据，不需要对全数据集进行再次训练，但能高效地发现新规律

要求：需要有连续的数据流输入模型

特点：模型结构保持，权重系数更新

数学公式：

$$\left\{ \begin{array}{l} temp_{\theta_j} = \theta_j - \alpha \times (y_{predict} - y) \times x_j \\ \theta_j = temp_{\theta_j} \end{array} \right\}$$

x, y 代表了输入的数据流， θ_j 为模型中对应的权重系数

| 在线学习案例

任务：针对物流公司快递价格，在不同时期，如何进行定价

- ① 考虑距离、运输量、天气、、同时期其他公司价格、历史参考价格等等，建立一个专家模型
- ② 根据客户的实时下单情况，预测客户对不同价格的购买意愿，根据意愿度调整价格



输入数据：地点信息、物品信息、报价

输出结果：客户是否下单

学习链接：<https://medium.com/value-stream-design/online-machine-learning-515556ff72c5>

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

$$g(x) = \theta_0 + \theta_1 x_1 + \dots$$

| 知识巩固

问题：思考迁移学习、在线学习、机器学习、深度学习的关系



Python3人工智能入门+实战提升：深度学习

Chapter 5 迁移学习与混合模型

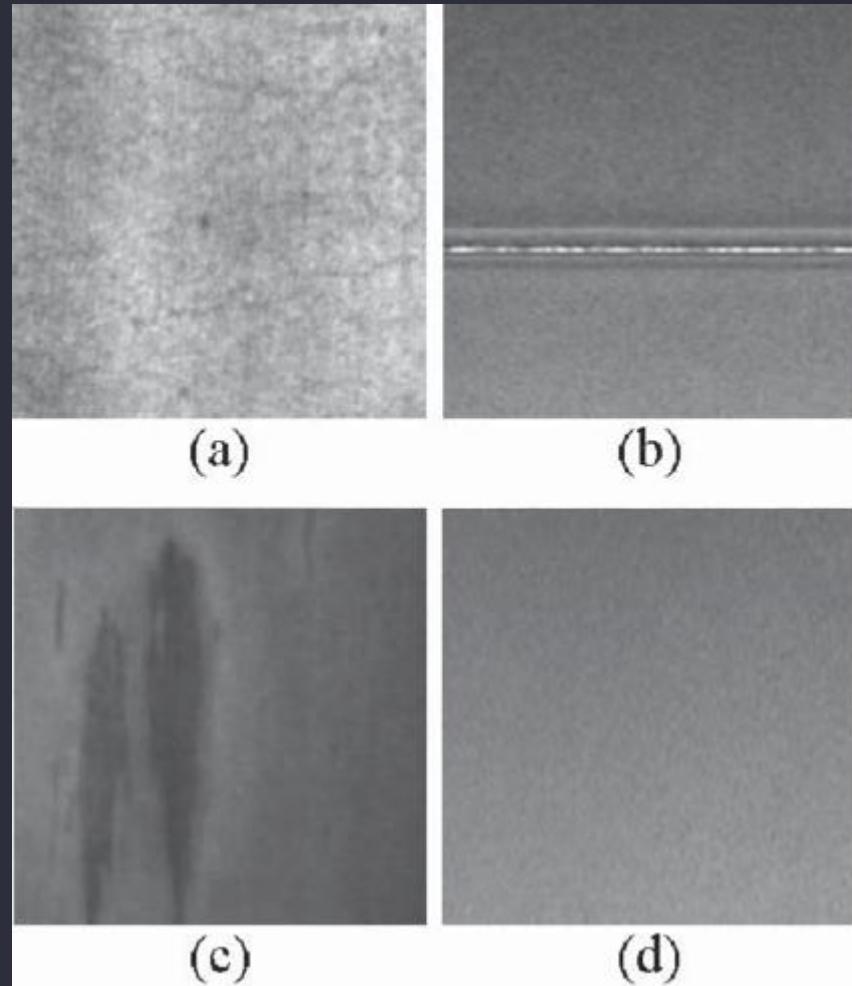
赵辛

Chapter 5 迁移学习与混合模型

-
- 1 --迁移学习与在线学习
 - 2 --混合模型
 - 3 --实战准备
 - 4 --实战（一）mlp模型迁移数据预测
 - 5 --实战（二）少样本区分奇特草莓

现实问题思考

你接到一个产品瑕疵检测的任务，通过图片自动判断产品表面是否有瑕疵



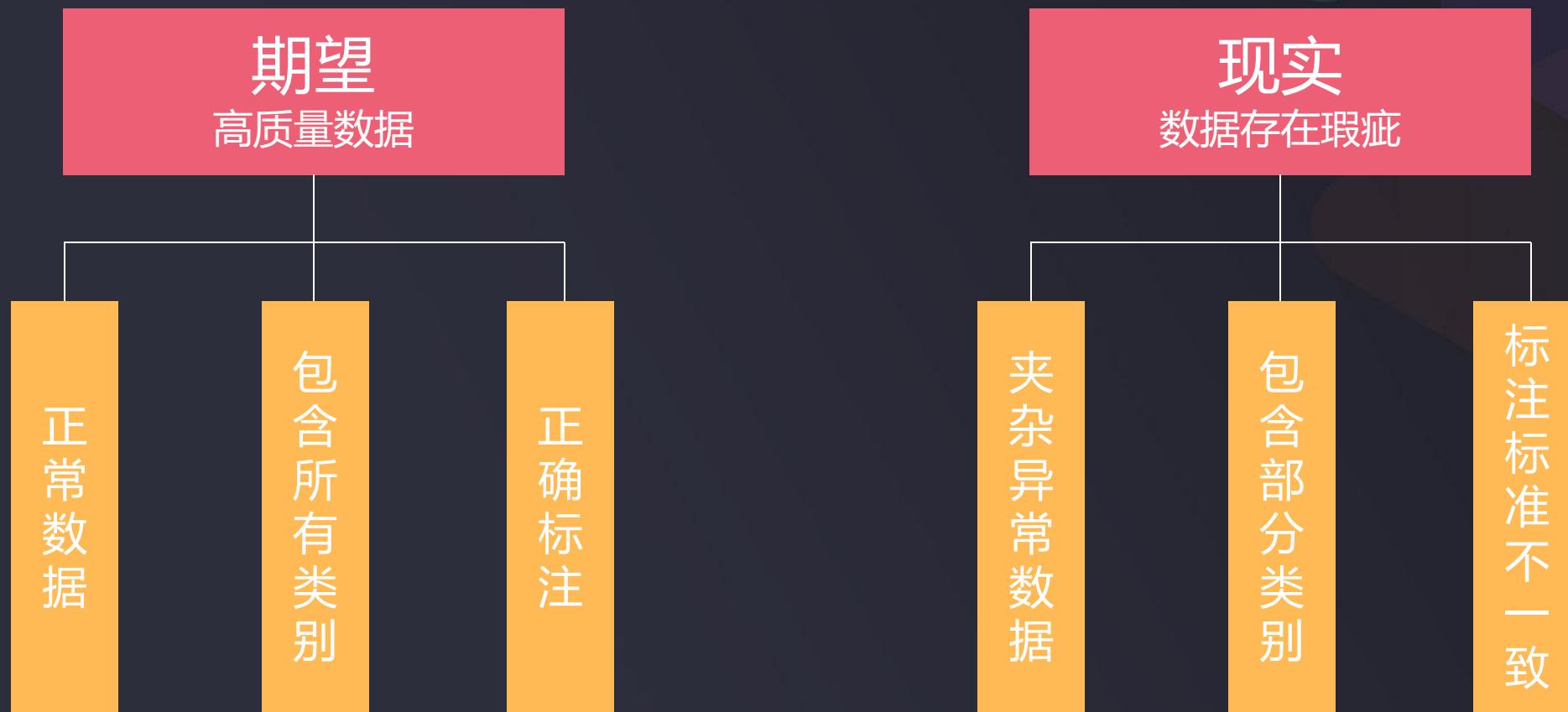
期望
高质量数据

正常数据

包含所有类别

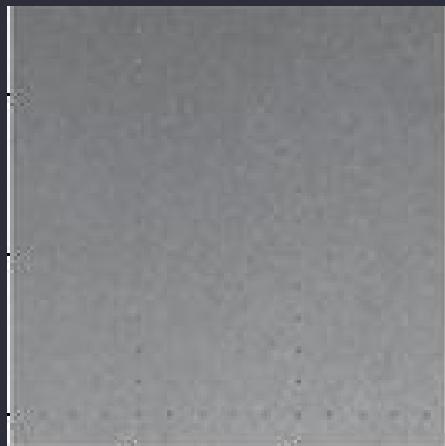
正确标注

| 现实数据会有很多问题



缺陷检测案例：良品都一个样，但次品的缺陷有很多

良品表面图

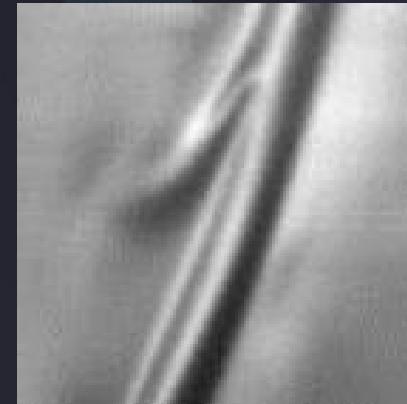


无法穷尽所有瑕疵

金属表面缺陷检测



(a)凹痕



(b)折痕



(c)刮伤



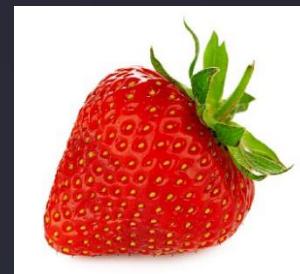
(d)油污

|奇特草莓检测

普通草莓



奇特草莓



样本总数40个，其中有12个有正确标签（都是普通草莓），其他样本均无标签，在不对剩下样本打标签的情况下建立模型实现奇特草莓检测。

|奇特草莓检测

样本总数40个，其中有12个有正确标签（都是普通草莓），其他样本均无标签，在不对剩下样本打标签的情况下建立模型实现奇特草莓检测。

特点：

- ① 负样本类别无法穷尽，但正样本相似度较高
- ② 存在带有标签的样本，但占总样本的比例不高
- ③ 总样本数较少

思路：

- ① 借助无监督学习寻找正样本重要属性，区分正负样本
- ② 充分利用标签样本，对无监督模型进行监督优化
- ③ 尝试数据增强，增加样本数

监督学习 + 无监督学习

半监督学习

半监督学习（Semi-Supervised Learning）是监督学习与无监督学习相结合的一种学习方法。半监督学习使用大量的未标记数据，以及同时使用标记数据，来进行模式识别工作。

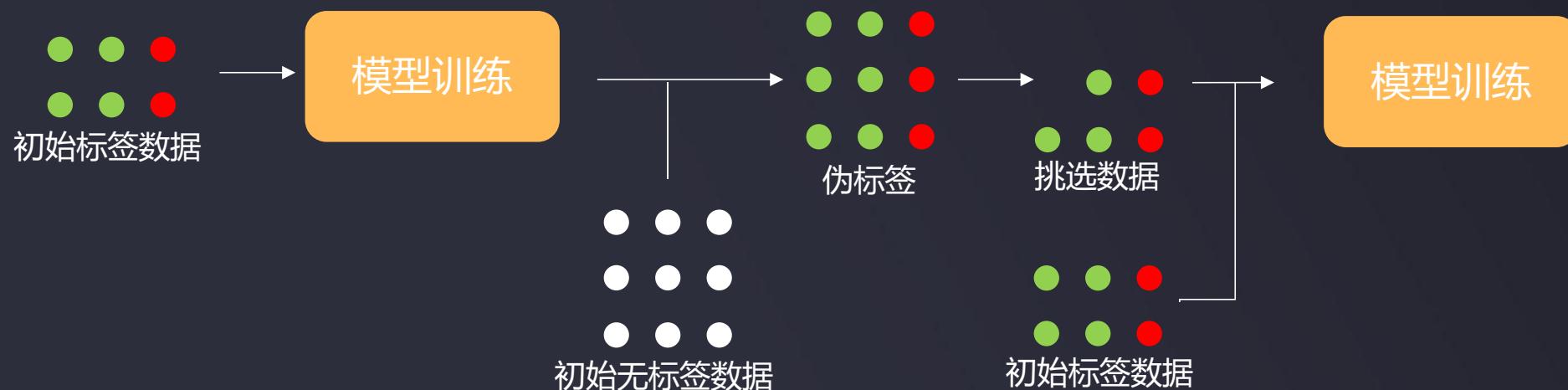
意义：在标记样本有限的情况下，尽可能识别出同类的共同特性，同时将大量无标签样本利用起来，发挥它们的价值

监督学习 + 无监督学习

寻找所有样本中的潜在规律，利用标签信息提高预测准确率

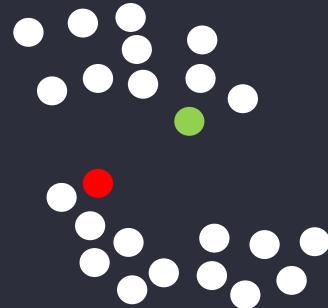
|半监督学习方法之一：伪标签学习

伪标签学习 (pseudo-labelling)：用有标签数据训练一个分类器，然后用这个分类器对无标签数据进行分类，这样就会产生伪标签 (pseudo label) 或软标签 (soft label)，挑选你认为分类正确的无标签样本（预先设定一个挑选准则），把选出来的无标签样本用来训练分类器。



伪标签学习

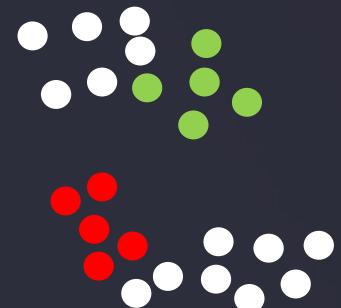
开始只有两个标签数据（红、绿点），白色为无标签数据



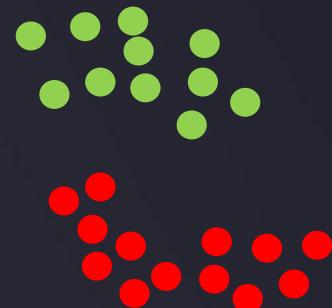
(a) 原始数据分布



(c) 继续挑选



(b) 挑选部分伪标签



(d) 分类完成

伪标签学习的局限性

- ① 标签数据可能都是同一个类别，无法训练出有效的分类器
- ② 可能无法寻找到合适的筛选准则

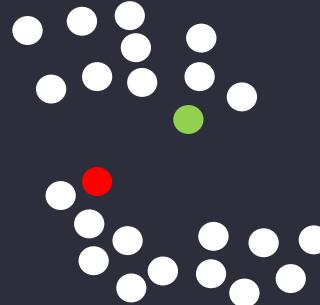
半监督的核心：尽可能将数据中的正确信息利用起来

- ① 相似样本的共同特性
- ② 标签样本的监督信息

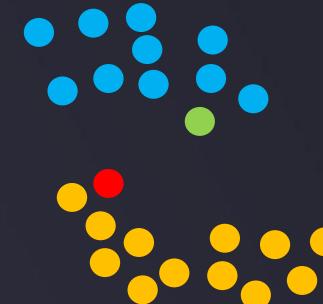
并没有特定的训练方式，多尝试
实现监督学习+无监督学习的灵活运用！

半监督学习：聚类分析+结果矫正

利用有标签的部分数据，对聚类模型结果进行结果矫正



(a) 原始数据分布



(b) 无监督聚类



(c) 正确标签矫正结果



并没有特定的训练方式，多尝试，实现监督学习+无监督学习的灵活运用！

半监督学习

并没有特定的训练方式，多尝试，实现监督学习+无监督学习的灵活运用！

有标签数据提取特征的半监督学习：

- ① 用有标签数据训练网络
- ② 通过训练好的模型的提取无标签数据特征，帮助建立模型完成任务

有标签数据并不局限于自己收集到的数据！

还可以利用别人使用的相似的数据或者模型，比如：利用VGG16提取图像特征！

|奇特草莓检测

普通草莓



监督学习

+ 无监督学习

奇特草莓



CNN图像特征提取

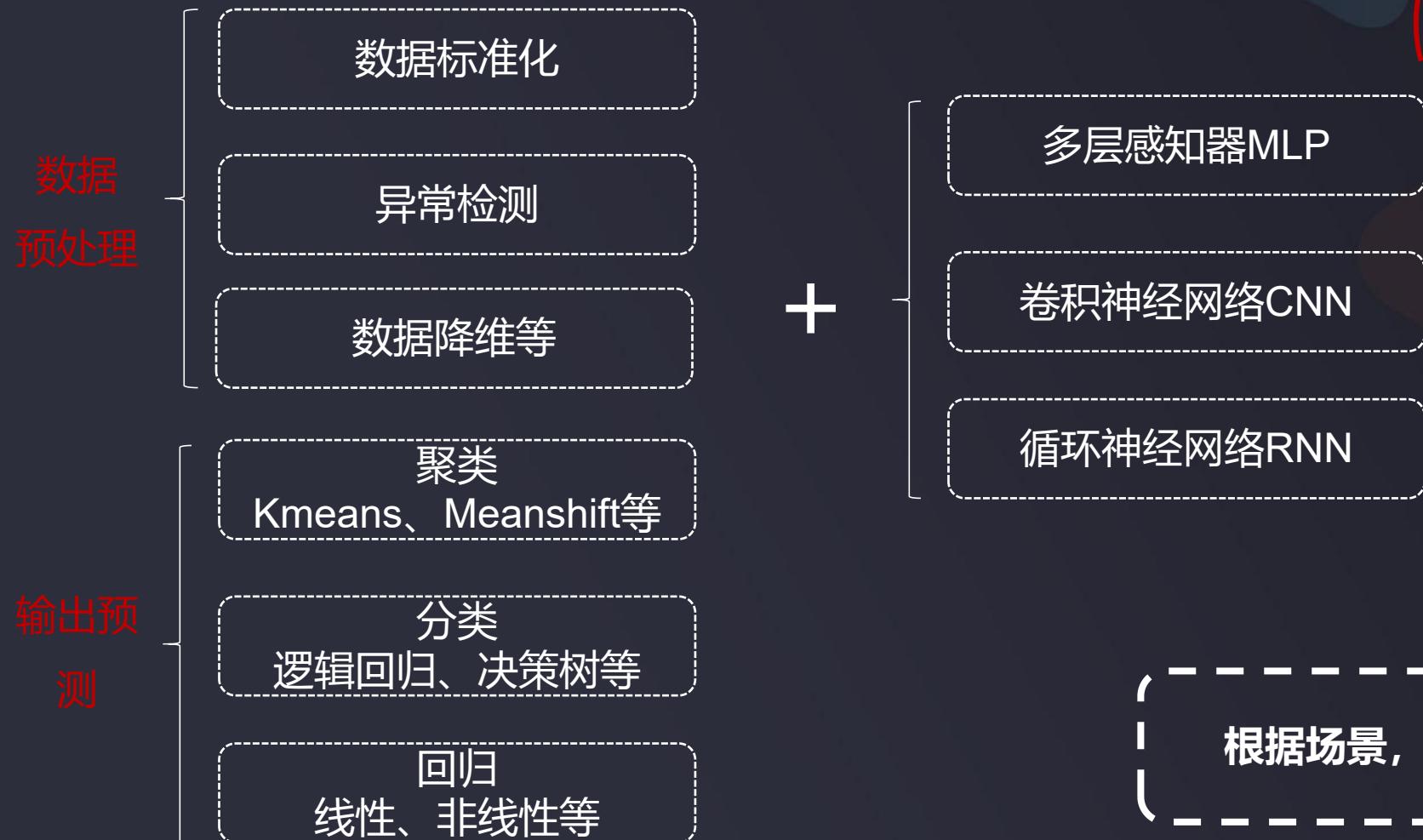
+ 聚类分析

深度学习

+ 机器学习

通过多种学习方法、不同技术的结合，有效地完成复杂现实任务！

机器学习 + 深度学习



复杂特
征提取

机器学习 + 深度学习

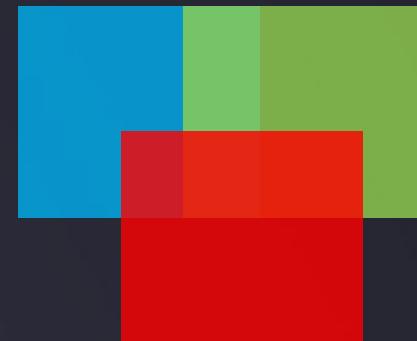
根据场景，灵活组合！

监督
学习

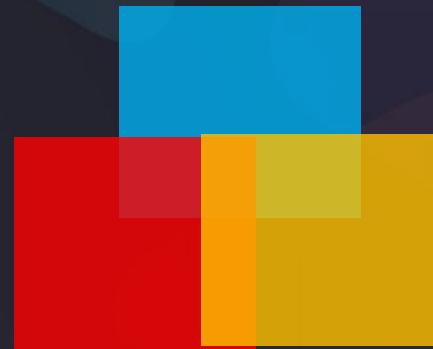
无监督
学习

机器
学习

深度
学习



(a)半监督学习



(b)机器+深度学习



(c)监督+无监督、机器+深度学习

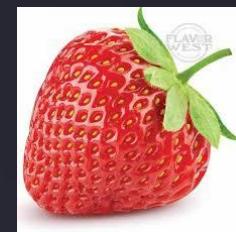
知识巩固

问题：结合课程知识，思考奇特草莓检测案例的完成方法

普通草莓



奇特草莓



样本总数40个，其中有12个有正确标签（都是普通草莓），其他样本均无标签，在不对剩下样本打标签的情况下建立模型实现奇特草莓检测。

特点：样本少、部分标注（且都是正样本）、负样本类别不可穷尽、图像数据



Python3人工智能入门+实战提升：深度学习

Chapter 5 迁移学习与混合模型

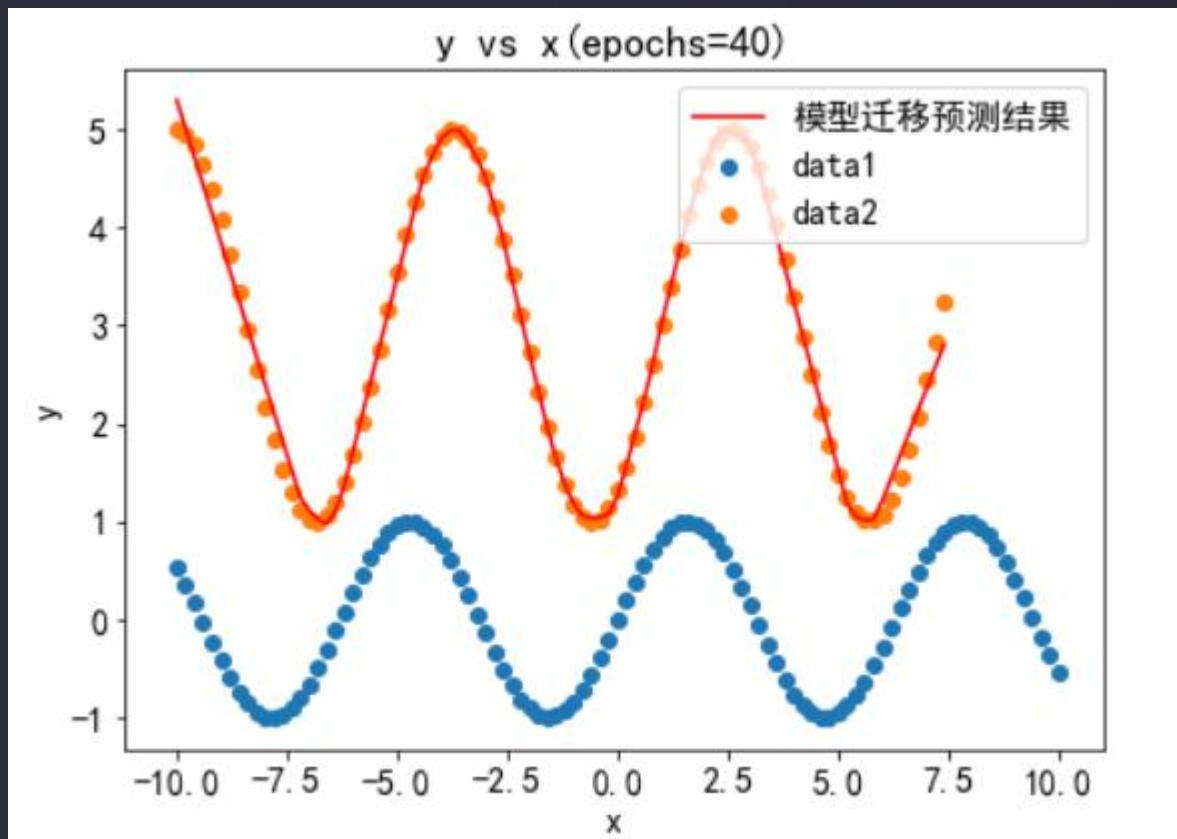
赵辛

Chapter 5 迁移学习与混合模型

-
- 1 --迁移学习与在线学习
 - 2 --混合模型
 - 3 --**实战准备**
 - 4 --实战（一）mlp模型迁移数据预测
 - 5 --实战（二）少样本区分奇特草莓

任务一：mlp模型迁移数据预测

基于task1_data1、task1_data2数据，建立mlp模型，并实现模型迁移学习



- 1、基于data1完成基本的数据加载、可视化工作；
- 2、建立mlp模型，epochs分别为500、1000、1500、2500次，可视化数据预测结果
- 3、模型存储为model1及加载为model2
- 4、基于data2，对model2进行迁移训练，
epoch=20、100，可视化模型对数据的预测结果
 - 模型结构：两个隐藏层，每层50神经元（激活函数relu），输出层激活函数linear

拓展任务：逐步增加epochs，生成预测结果gif，查看迁移学习过程

mlp模型迁移数据预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
from keras.models import Sequential  
from keras.layers import Dense  
model1 = Sequential()  
model1.add(Dense(units=50, input_dim = 1, activation='relu'))  
model1.add(Dense(units=50,activation='relu'))  
model1.add(Dense(units=1, activation = 'linear'))  
model1.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 50)	100
dense_2 (Dense)	(None, 50)	2550
dense_3 (Dense)	(None, 1)	51
<hr/>		
Total params: 2,701		
Trainable params: 2,701		
Non-trainable params: 0		

mlp模型迁移数据预测

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

#参数配置与模型训练

```
model1.compile(optimizer='adam',loss='mean_squared_error')  
model1.fit(X,y,epochs=500)
```

```
Epoch 1/500  
101/101 [=====] - 1s 6ms/step - loss: 2.6715  
Epoch 2/500  
101/101 [=====] - 0s 129us/step - loss: 1.6110  
Epoch 3/500  
101/101 [=====] - 0s 168us/step - loss: 1.1507  
Epoch 4/500  
101/101 [=====] - 0s 247us/step - loss: 0.8394  
Epoch 5/500  
101/101 [=====] - 0s 158us/step - loss: 0.5785  
Epoch 6/500  
101/101 [=====] - 0s 228us/step - loss: 0.4828  
Epoch 7/500  
101/101 [=====] - 0s 139us/step - loss: 0.4872  
Epoch 8/500  
101/101 [=====] - 0s 178us/step - loss: 0.4862
```

mlp模型迁移数据预测

数据加载及展示

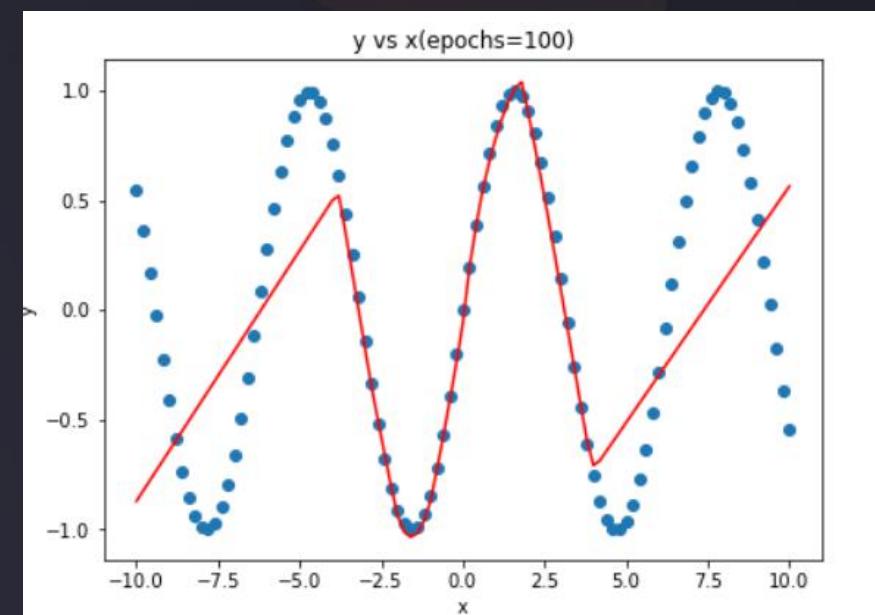
数据预处理

模型建立及训练

模型预测

结果展示及表现评估

```
#结果预测与可视化  
y_predict = model1.predict(X)  
fig2 = plt.figure(figsize=(7,5))  
plt.scatter(X,y)  
plt.plot(X,y_predict,'r')  
plt.title('y vs x(epochs=100)')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.show()
```



mlp模型迁移数据预测

模型保存到本地：

```
from sklearn.externals import joblib  
joblib.dump(model, "model1.m")
```

或者直接引入joblib包（先通过pip install joblib安装）：import joblib

['model1.m']

加载本地模型：

```
model2 = joblib.load("model1.m")
```

迁移学习：

```
model.fit(x2,y2,epochs=20)
```

```
Epoch 1/20  
88/88 [=====] - 0s 1ms/step - loss: 7.6069  
Epoch 2/20  
88/88 [=====] - 0s 79us/step - loss: 2.1651  
Epoch 3/20  
88/88 [=====] - 0s 102us/step - loss: 2.2388  
Epoch 4/20  
88/88 [=====] - 0s 91us/step - loss: 2.1126  
Epoch 5/20  
88/88 [=====] - 0s 102us/step - loss: 1.2270  
Epoch 6/20
```

mlp模型迁移数据预测

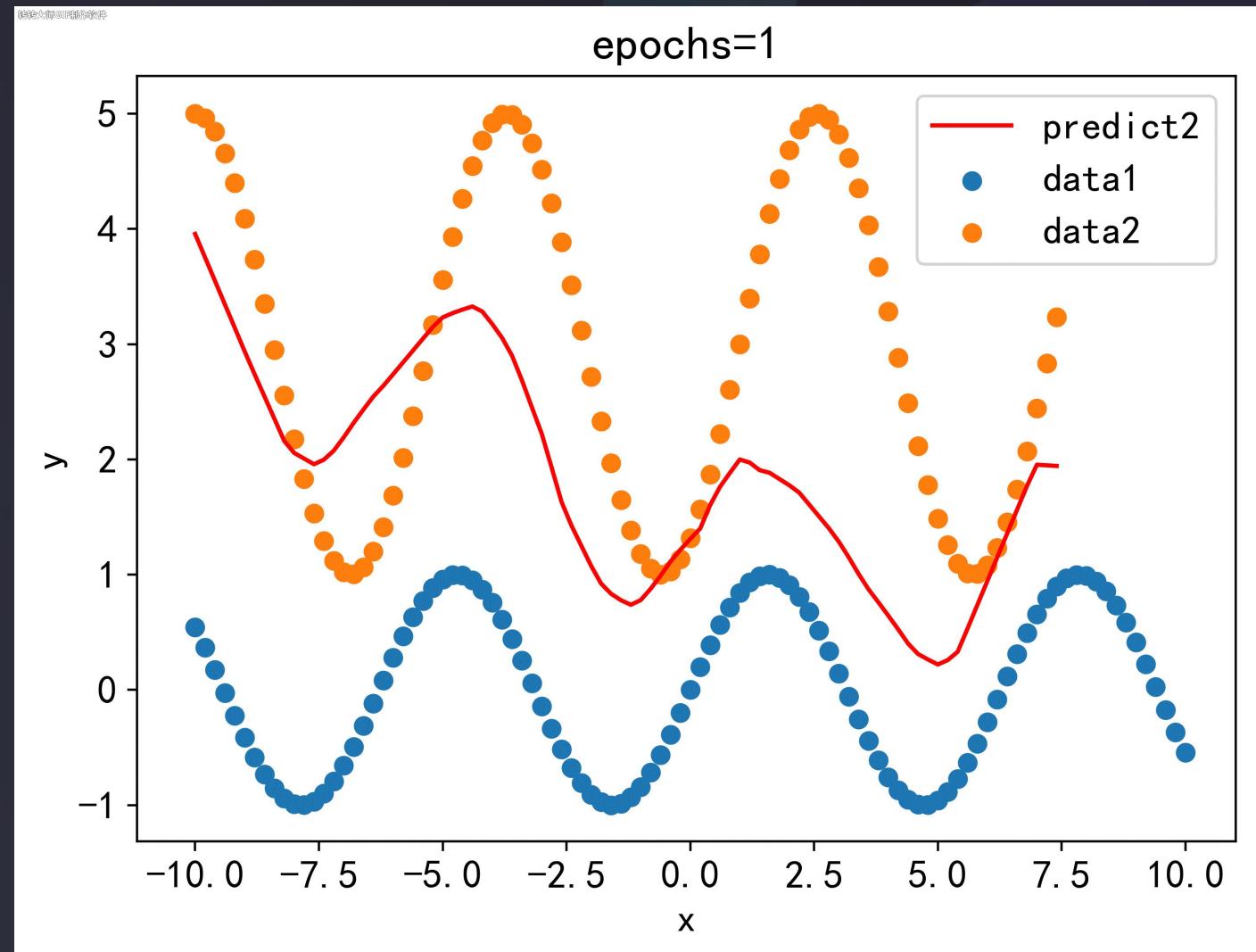
数据加载及展示

数据预处理

模型建立及训练

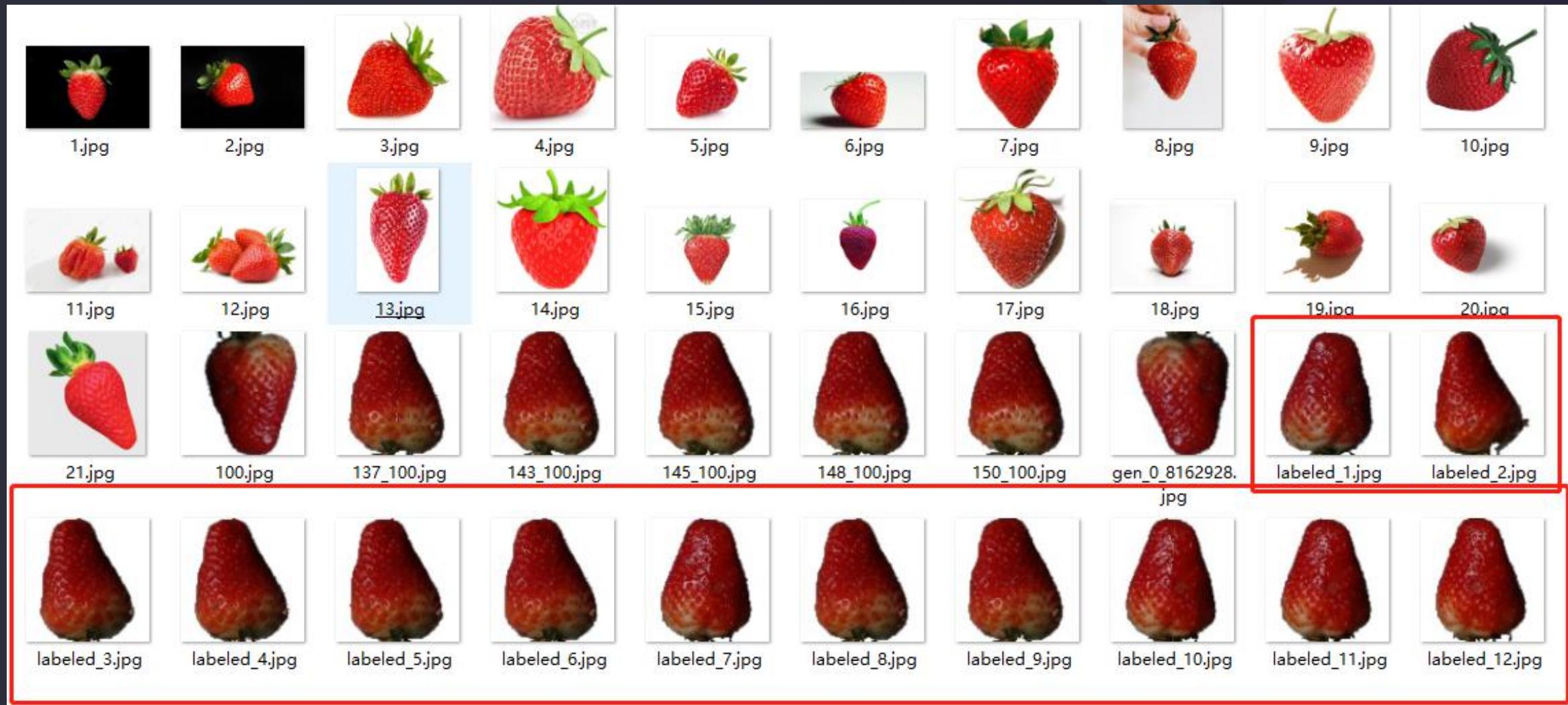
模型预测

结果展示及表现评估



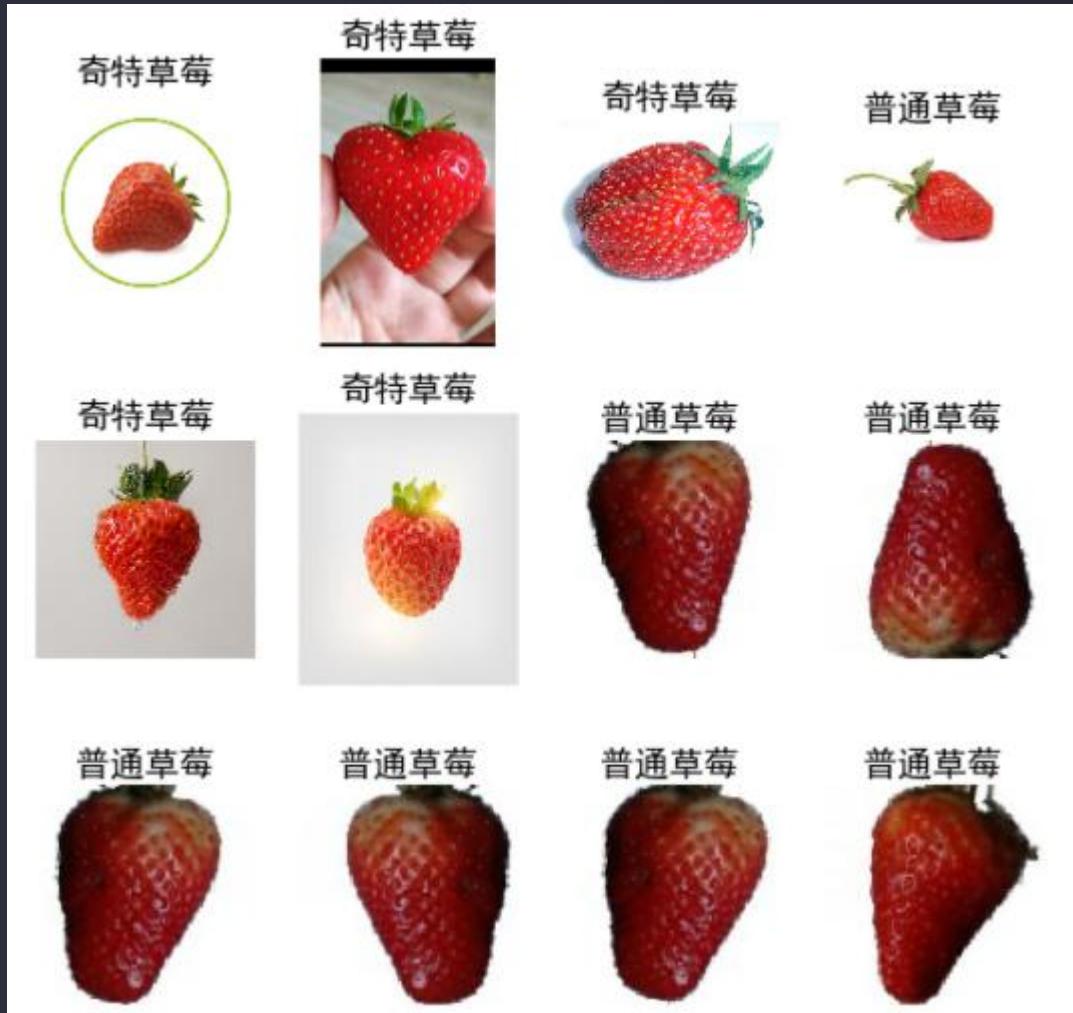
任务二：少样本区分奇特草莓

40张图片，12张打标签（普通草莓），其他均无标签



任务二：少样本区分奇特草莓

基于task2_data//all的40张图片，建立模型有效的区分奇特草莓与普通草莓



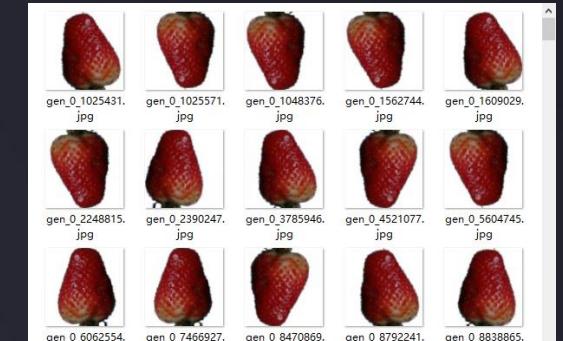
- 1、数据增强：利用12张普通草莓图片生成更多样本
- 2、加载、可视化单张图片，使用VGG16提取特征
- 3、实现对所有样本图片的特征提取
- 4、建立Kmeans模型，完成图像数据聚类
- 5、设定样本更多的类别为普通草莓，完成结果矫正
- 6、使用测试图片test_data，验证模型表现
- 7、建立Meanshift模型，预测类别并进行结果矫正
- 8、引入PCA技术降低噪声数据影响，提升模型表现

| 数据增强：生成更多图片

```
from keras.preprocessing.image import ImageDataGenerator  
path = 'origin_data' # 待增强图片的路径  
dst_path = 'gen_data' # 图片增强后的存储路径  
# 创建实例、配置图片增强参数  
datagen = ImageDataGenerator(rotation_range=10,width_shift_range=0.1,  
height_shift_range=0.02, horizontal_flip=True,vertical_flip=True)  
gen = datagen.flow_from_directory(path,target_size=(224, 224),  
batch_size=2,save_to_dir=dst_path,  
save_prefix='gen',save_format='jpg')
```

```
for i in range(100):  
    gen.next()
```

Found 12 images belonging to 1 classes.



少样本区分奇特草莓

数据加载及展示

数据预处理

模型建立及训练

模型预测

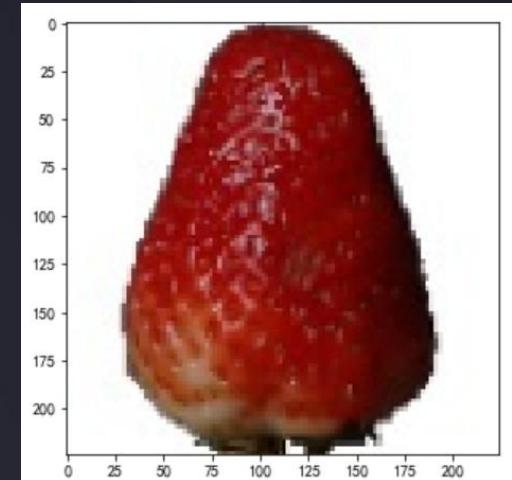
结果展示及表现评估

单张图片载入：

```
from keras.preprocessing.image import  
load_img,img_to_array  
img_path = '1.jpg'  
img = load_img(img_path,target_size=(224,224))
```

单张图片可视化：

```
from matplotlib import pyplot as plt  
fig = plt.figure(figsize=(5,5))  
plt.imshow(img)
```



```
#格式转化  
img = img_to_array(img)  
print(type(img),img.shape)
```

```
<class 'numpy.ndarray'> (224, 224, 3)
```

```
#模型加载、图像矩阵预处理  
from keras.applications.vgg16 import VGG16  
from keras.applications.vgg16 import preprocess_input  
import numpy as np  
model_vgg =  
VGG16(weights='imagenet',include_top=False)  
X = np.expand_dims(img,axis=0)  
X = preprocess_input(X)  
print(X.shape)
```

```
(1, 224, 224, 3)
```

```
#特征提取  
features = model_vgg.predict(X)  
print(features.shape)
```

```
(1, 7, 7, 512)
```

```
#flatten展开  
features = features.reshape(1,7*7*512)  
print(features.shape)
```

```
(1, 25088)
```

|少样本区分奇特草莓

```
#批量图片路径加载
import os
folder = 'task2_data//training_data'
dirs = os.listdir(folder) #获取文件夹下所有文件名称
#名称合并
img_path = []
for i in dirs:
    if os.path.splitext(i)[1] == '.jpg':
        img_path.append(i)
img_path = [folder + "//" + i for i in img_path]
print(img_path)
```

```
['task2_data//training_data//1.jpg', 'ta
k2_data//training_data//100.jpg', 'task2
data//training_data//12.jpg', 'task2_dat
a//training_data//137_100.jpg', 'task2_d
ta//training_data//143_100.jpg', 'task2_
sk2_data//training_data//148_100.jpg',
ask2_data//training_data//150_100.jpg',
'task2_data//training_data//17.jpg', 'ta
k2_data//training_data//19.jpg', 'task2_
ta//training_data//20.jpg', 'task2_data/
```

少样本区分奇特草莓

#定义一个提取图片特征的方法

```
def modelProcess(img_path,model):
    img = load_img(img_path,target_size = (224,224))
    img = img_to_array(img)
    X = np.expand_dims(img,axis=0)
    X = preprocess_input(X)
    X_VGG = model.predict(X)
    X_VGG = X_VGG.reshape(1,7*7*512)
    return X_VGG
```

#图像批量处理

```
features_train = np.zeros([len(img_path),7*7*512])
for i in range(len(img_path)):
    feature_i = modelProcess(img_path[i],model_vgg)
    print('preprocessed:',img_path[i])
    features_train[i] = feature_i
```

```
preprocessed: task2_data//training_data//1.jpg
preprocessed: task2_data//training_data//10.jpg
preprocessed: task2_data//training_data//100.jpg
preprocessed: task2_data//training_data//11.jpg
preprocessed: task2_data//training_data//12.jpg
preprocessed: task2_data//training_data//13.jpg
preprocessed: task2_data//training_data//137_100.jpg
preprocessed: task2_data//training_data//14.jpg
preprocessed: task2_data//training_data//143_100.jpg
preprocessed: task2_data//training_data//145_100.jpg
preprocessed: task2_data//training_data//148_100.jpg
```

少样本区分奇特草莓

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

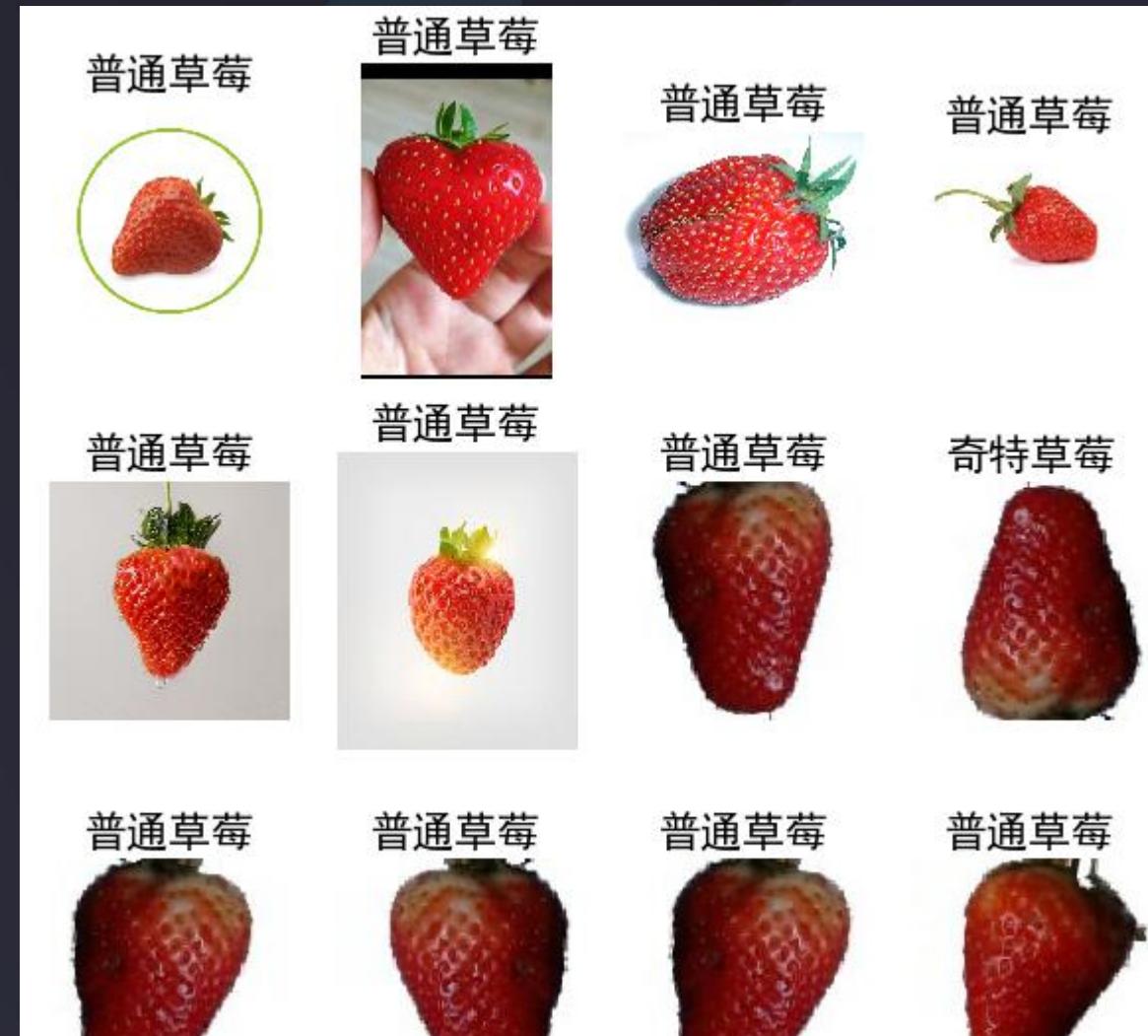
#建立KMeans聚类模型

```
from sklearn.cluster import KMeans  
cnn_kmeans = KMeans(n_clusters=2,max_iter=3000)  
cnn_kmeans.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=3000,  
n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',  
random_state=None, tol=0.0001, verbose=0)
```

|少样本区分奇特草莓

```
#可视化测试数据集预测结果  
fig3 = plt.figure(figsize=(10,10))  
  
for i in range(3):  
    for j in range(4):  
        img = load_img(img_path_test[i*4+j])#read the image  
        plt.subplot(3,4,i*4+j+1)  
        plt.title('普通草莓' if  
y_predict_kmeans_test[i*4+j]==normal_strawberry_id else  
'奇特草莓')  
        plt.imshow(img),plt.axis('off')
```

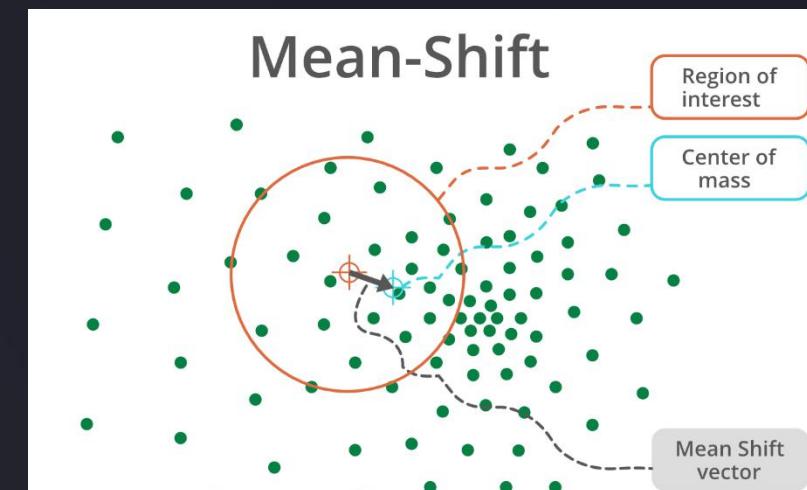


| 均值漂移聚类

均值漂移算法：一种基于密度梯度上升的聚类算法（沿着密度上升方向寻找聚类中心点）

公式：

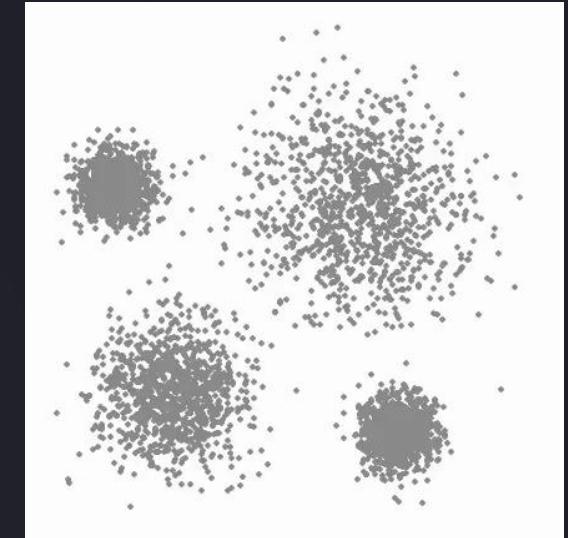
S_h :以 u 为中心点，半径为 h 的高维球区域； k :包含在 S_h 范围内点的个数； x_i :包含在 S_h 范围内的点；
 M^t 为 t 状态下求得的偏移均值； u^t 为 t 状态下的中心



| 均值漂移聚类

算法流程：

- 1、随机选择未分类点作为中心点
- 2、找出离中心点距离在带宽之内的点，记做集合S
- 3、计算从中心点到集合S中每个元素的偏移向量M
- 4、中心点以向量M移动
- 5、重复步骤2-4，直到收敛
- 6、重复1-5直到所有的点都被归类
- 7、分类：根据每个类，对每个点的访问频率，取访问频率最大的那个类，作为当前点集的所属类



少样本区分奇特草莓

数据加载及展示

数据预处理

模型建立及训练

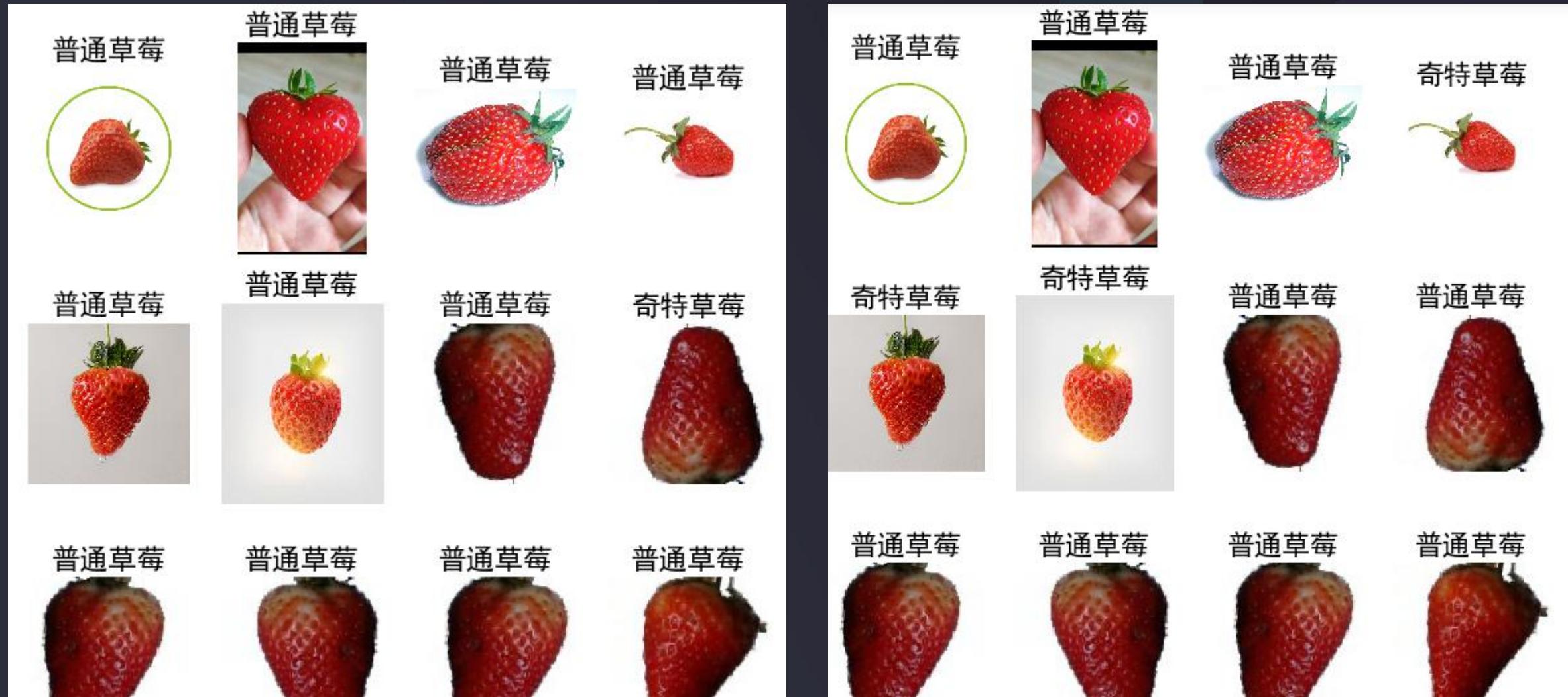
模型预测

结果展示及表现评估

```
#meanshift聚类模型  
from sklearn.cluster import MeanShift, estimate_bandwidth  
#获取区域宽度  
bw = estimate_bandwidth(X,n_samples=150)  
#建立模型并训练  
cnn_ms = MeanShift(bandwidth=bw)  
cnn_ms.fit(X)
```

```
MeanShift(bandwidth=1505.6808591606793, bin_seeding=False, cluster_all=True,  
max_iter=300, min_bin_freq=1, n_jobs=None, seeds=None)
```

|少样本区分奇特草莓



少样本区分奇特草莓

数据加载及展示

数据预处理

模型建立及训练

模型预测

结果展示及表现评估

数据降维（PCA处理）：

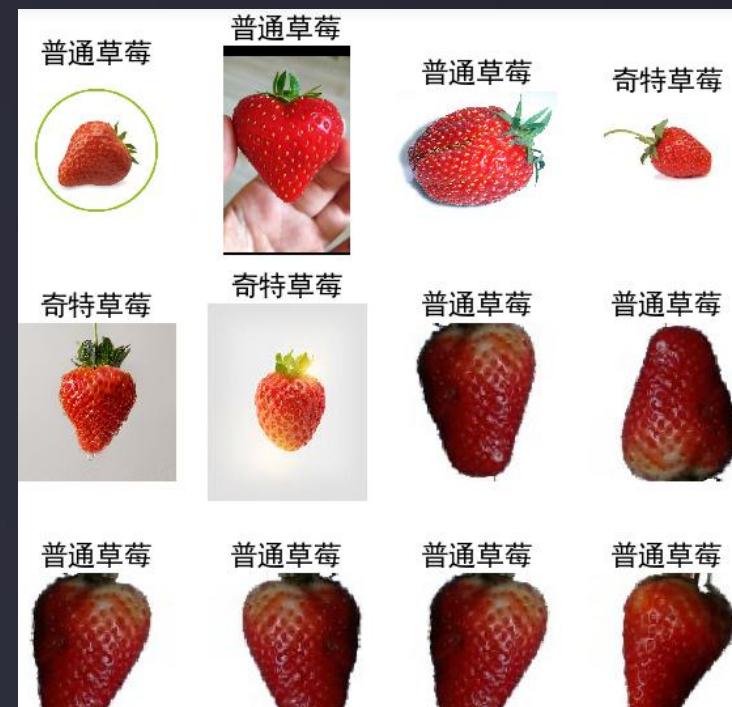
```
from sklearn.preprocessing import StandardScaler  
#数据标准化处理  
stds = StandardScaler()  
X_norm = stds.fit_transform(X)  
#PCA降维  
from sklearn.decomposition import PCA  
pca = PCA(n_components=200)  
X_pca = pca.fit_transform(X_norm)  
#计算主成分方差比例  
var_ratio = pca.explained_variance_ratio  
#查看主成分方差比之和  
print(np.sum(var_ratio))  
#查看将为前后数据维度  
print(X.shape,X_pca.shape)
```

(240, 25088) (240, 200)

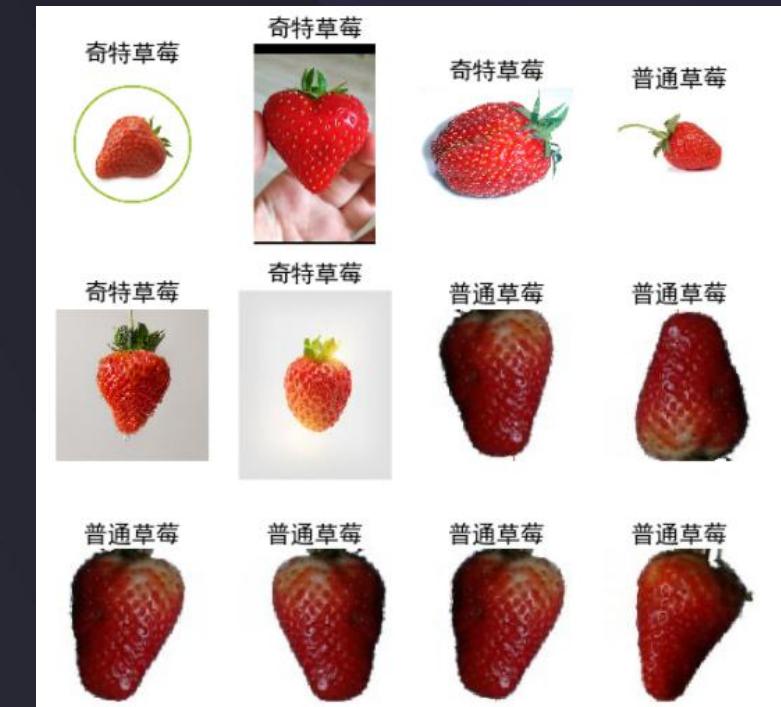
| 少样本区分奇特草莓



12个预测准确5个



12个预测准确9个



12个预测准确11个



Python3人工智能入门+实战提升：深度学习

Chapter 5 迁移学习与混合模型

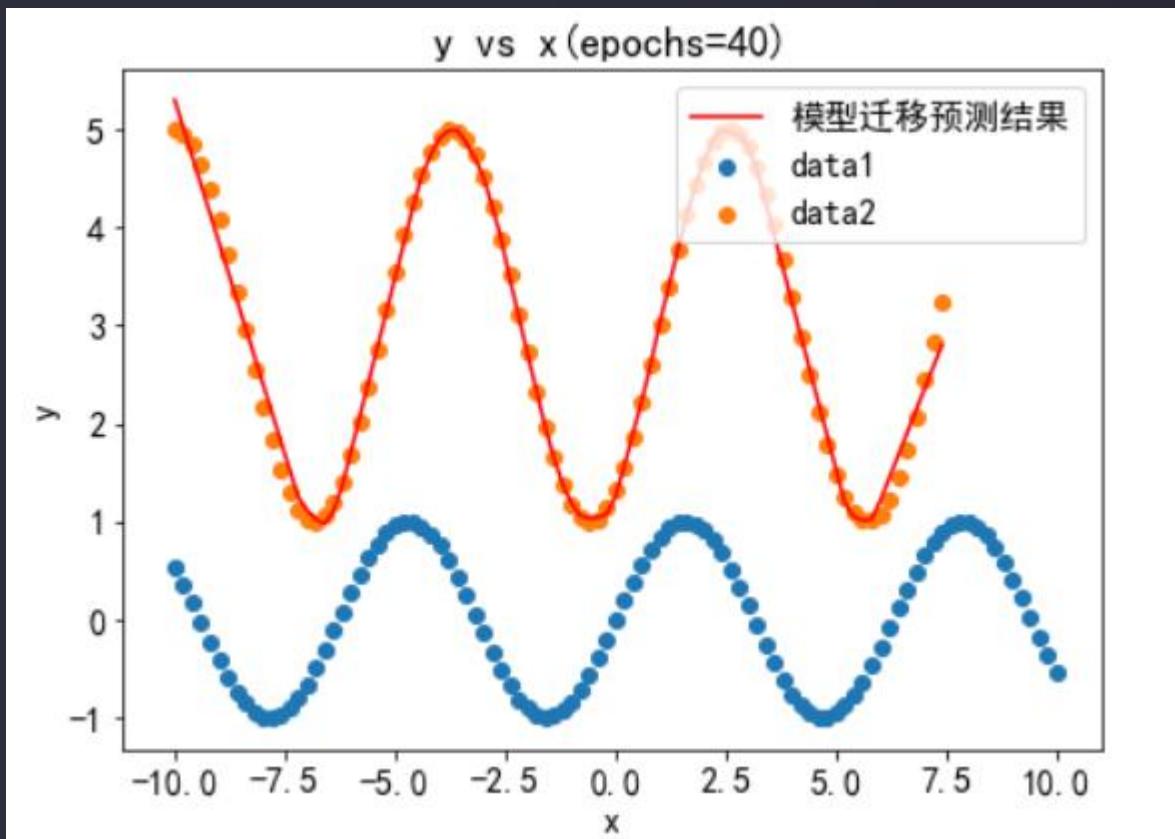
赵辛

Chapter 5 迁移学习与混合模型

-
- 1 --迁移学习与在线学习
 - 2 --混合模型
 - 3 --实战准备
 - 4 --实战（一）mlp模型迁移数据预测
 - 5 --实战（二）少样本区分奇特草莓

任务一：mlp模型迁移数据预测

基于task1_data1、task1_data2数据，建立mlp模型，并实现模型迁移学习



- 1、基于data1完成基本的数据加载、可视化工作；
- 2、建立mlp模型，epochs分别为500、1000、1500、2500次，可视化数据预测结果
- 3、模型存储为model1及加载为model2
- 4、基于data2，对model2进行迁移训练，
epoch=20、100，可视化模型对数据的预测结果
 - 模型结构：两个隐藏层，每层50神经元（激活函数relu），输出层激活函数linear

拓展任务：逐步增加epochs，生成预测结果gif，查看迁移学习过程



Python3人工智能入门+实战提升：深度学习

Chapter 5 迁移学习与混合模型

赵辛

Chapter 5 迁移学习与混合模型

-
- 1 --迁移学习与在线学习
 - 2 --混合模型
 - 3 --实战准备
 - 4 --实战（一）mlp模型迁移数据预测
 - 5 --实战（二）少样本区分奇特草莓

任务二：少样本区分奇特草莓

40张图片，12张打标签（普通草莓），其他均无标签

