




第三章 人工智能开发入门

主讲：陈建海

浙江大学计算机科学与技术学院

2024年9月





提纲

- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

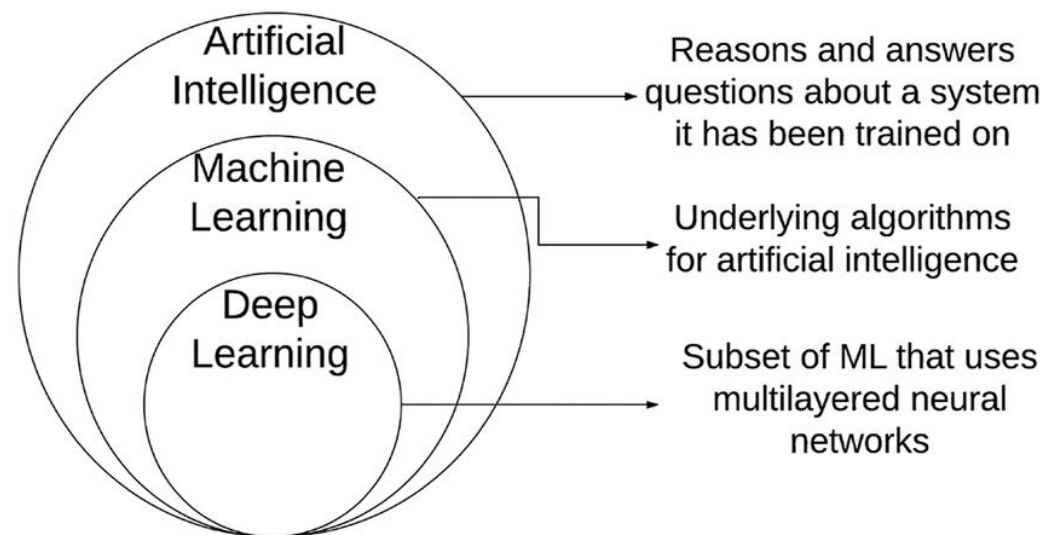
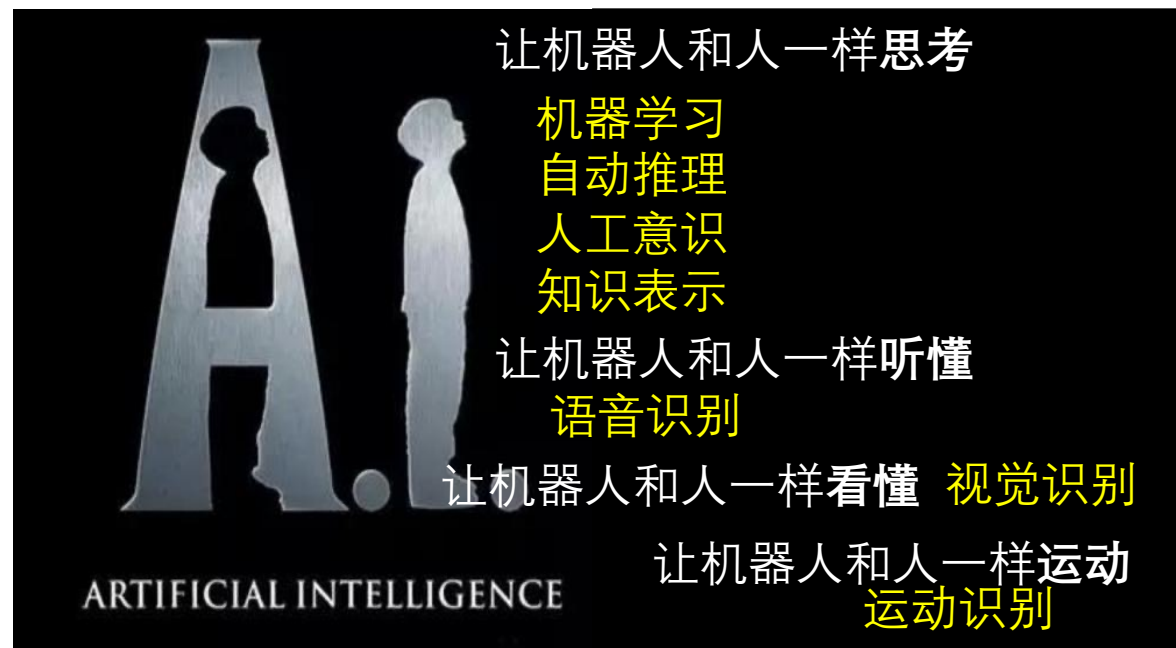


提纲

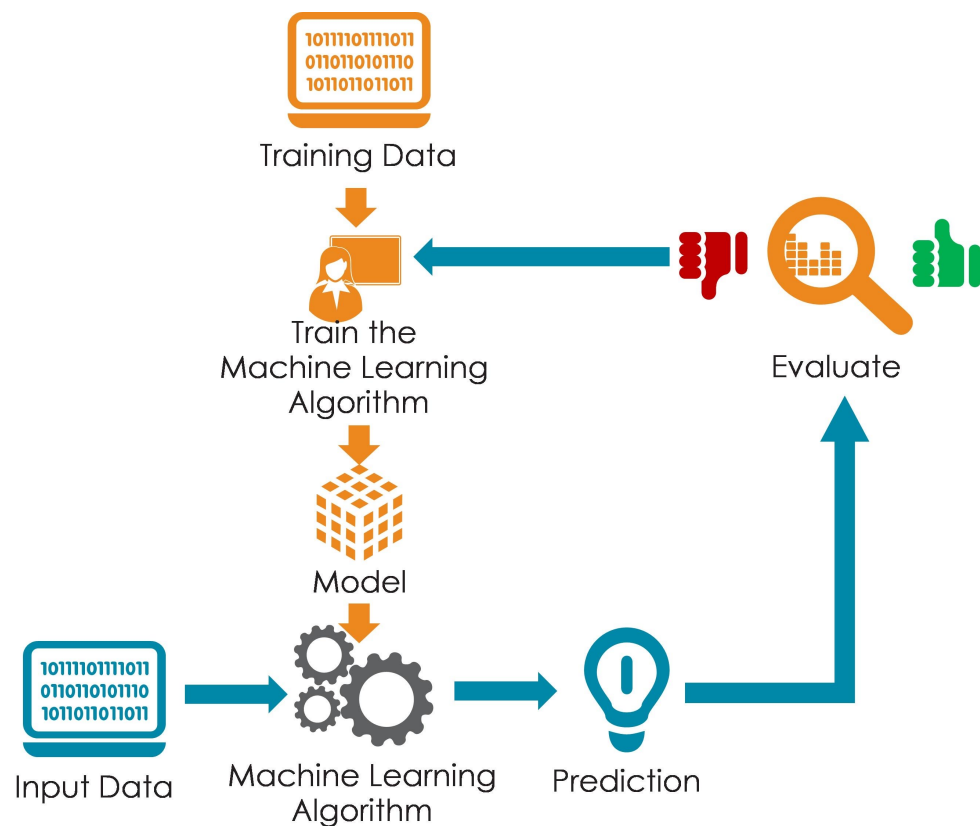
- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

1 AI基础回顾-什么是AI

- **人工智能(Artificial Intelligence)**, 英文缩写为**AI**。
 - 研究、开发用于**模拟、延伸和扩展**人的智能的理论、方法、技术及应用系统的一门新的技术科学。
 - 企图了解**智能**的本质, 并**生产出**一种新的能以**人类智能相似的方式**做出反应的**智能机器**
 - 研究包括**机器人、语音识别、图像识别、自然语言处理和专家系统**等。
 - **AI**是对人的**意识、思维的信息过程的模拟**, 不是人的智能, 但能像人那样**思考、也可能超过人的智能**。

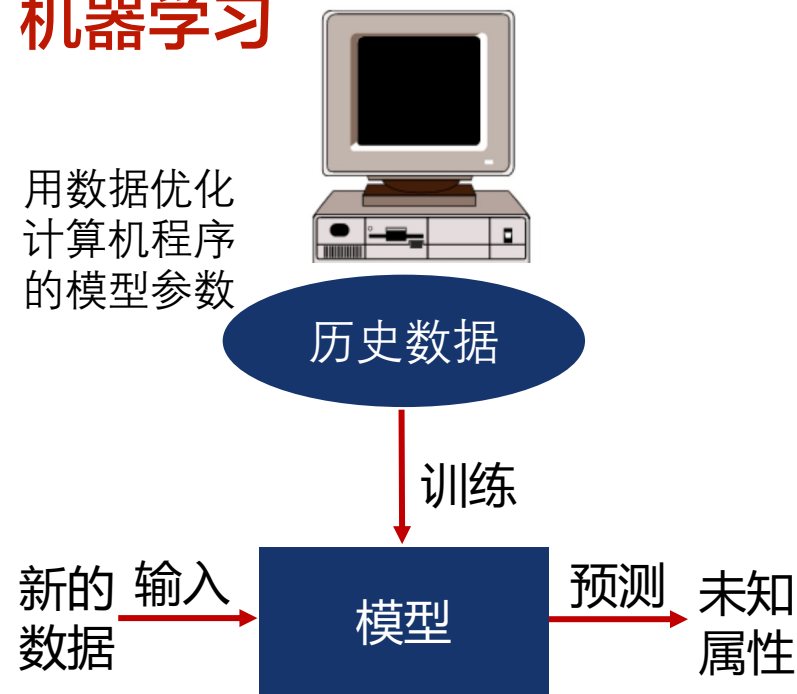


- 机器学习是AI的核心
- 采用AI解决任何一个任务往往都需要的步骤：
 - 定义模型
 - 训练模型
 - 测试模型



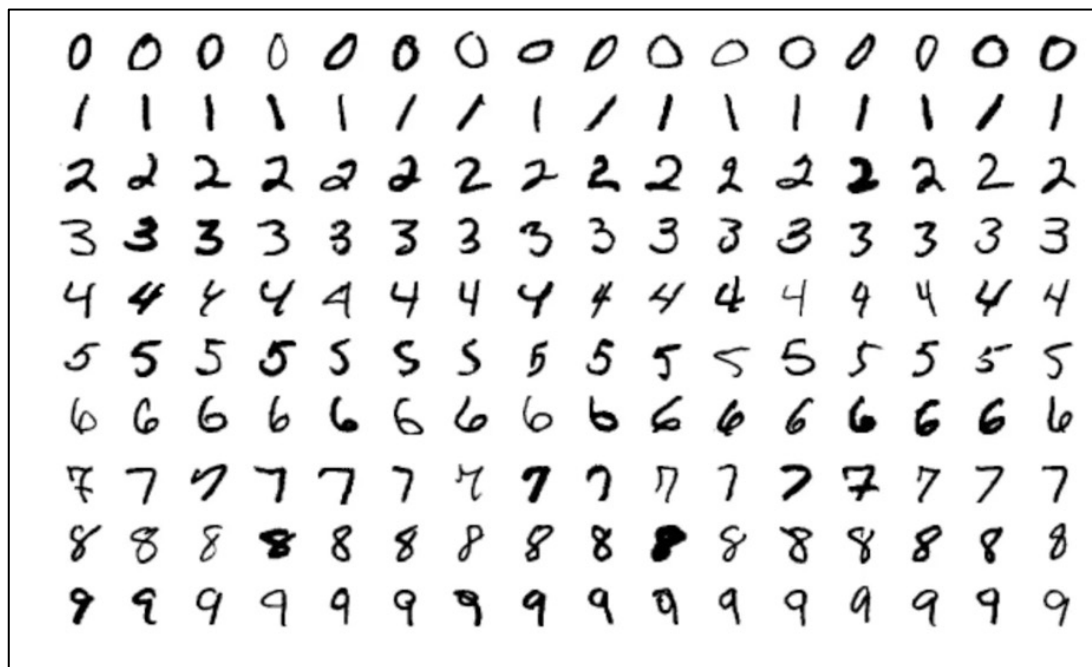
机器学习

用数据优化
计算机程序
的模型参数



1 MNIST案例—手写数字识别

□假如我们在白纸上用黑色的笔写上一个数字（0-9），再用手机拍一张照片。我们希望让机器能够认知写了什么字。这就是典型的人工智能问题，机器如何识别一个手写体的数字。



1 机器识别概述

- 定义：机器识别是指计算机系统通过分析和理解数据来识别模式、对象或信息的过程。常见的机器识别包括**图像识别**、**语音识别**和**文本识别**。
- 重要性：机器识别技术在许多领域有着广泛应用，如**自动驾驶汽车**、**虚拟助手**、**医疗诊断**和**安全监控**等。
- 应用场景：展示一些典型应用场景的图片或视频，例如自动驾驶汽车识别交通标志、智能音响识别语音命令。

1 机器识别的工作原理

□基本原理：

机器识别依赖于数据输入、特征提取、模型训练和预测四个主要步骤。

数据输入通过传感器或其他数据采集设备获取，特征提取用于从原始数据中提取有用信息，模型训练是基于特征数据进行学习，预测则是模型对新数据进行分类或识别。

□技术基础：

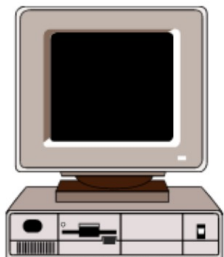
机器学习：基于数据进行模式识别和预测。

深度学习：通过神经网络模拟人脑的处理方式进行复杂模式识别。

1 机器学习&深度学习

机器学习

用数据优化
计算机程序
的模型参数



历史数据

训练

模型

新的输入
数据

预测 未知
属性

深度学习

通过经验自
动改进的计
算机算法



经验

归纳

规律

新的输入
问题

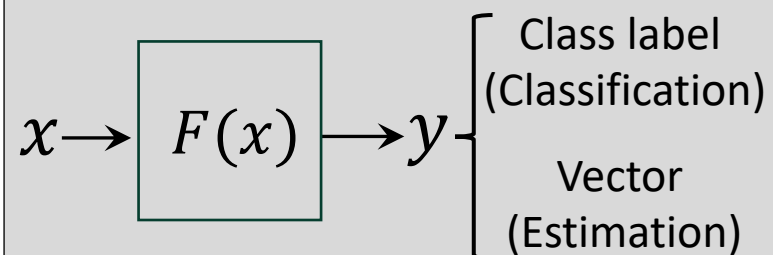
预测 未来

机器学习的基本定理

模型的出错率 $\propto \frac{\text{模型的复杂程度}}{\text{样本的大小}}$

推论：

模型复杂 - \gg 大样本
样本小 - \gg 简化模型



- 深度学习是使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。
- 深度学习是一种基于对数据进行表征学习的方法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

1 人工智能开源框架

□人工智能框架是AI算法模型设计、训练于验证的一套标准接口、特性库与工具包，集成了算法的封装、数据调用以及计算资源的使用。

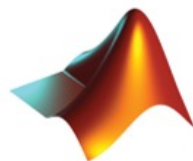
□开源人工智能框架为开发人员创建**人工智能应用程序**提供了便捷的工具。

□这些框架**免费提供**，鼓励合作并通过**社区贡献促进创新**。

Caffe

Caffe2

Chainer



MATLAB



mxnet

PaddlePaddle

PyTorch

TensorFlow

torch

Wolfram Language

1 AI开源工具框架概述

□人工智能开源工具和框架，目的是帮助开发者快速搭建和部署AI模型

1. 编程语言：python
2. 开发环境：python IDLE, PyCharm, Jupyter notebook

◆ Scikit-learn

用于机器学习的Python库，提供各种常用的机器学习算法和工具。

◆ PyTorch

由Facebook开发的深度学习框架，提供动态计算图和易用的API  PyTorch

◆ TensorFlow

由Google开发的深度学习框架，支持灵活的模型构建和训练

◆ CNTK

由MS微软开发的CNTK开源工具包，商业级分布式深度学习

◆ Caffe

由 Berkeley Vision and Learning Center 开发的深度学习框架，适用于计算机视觉任务

◆ OpenCV

用于计算机视觉的开源库，提供各种图像处理和计算机视觉算法

◆ MXNet

Apache的支持分布式训练的深度学习框架，提供灵活的API和高性能计算

◆ Theano

用于深度学习的数学表达式库，支持定义、优化和评估数学表达式

◆ Keras

高层神经网络API，支持多种深度学习框架，如 TensorFlow 和 Theano

常用人工智能开源工具框架



提纲

- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

2 Python概述

- Python是一种面向对象的解释型计算机程序设计语言，由荷兰人Guido van Rossum于1989年发明，第一个公开发行人版发行于1991年。
- Python的设计哲学是“优雅”、“明确”、“简单”。
- Python是自由软件之一，免费、开源。
- Python已经被移植到许多平台上，包括Unix/Linux、Windows、Mac OS。



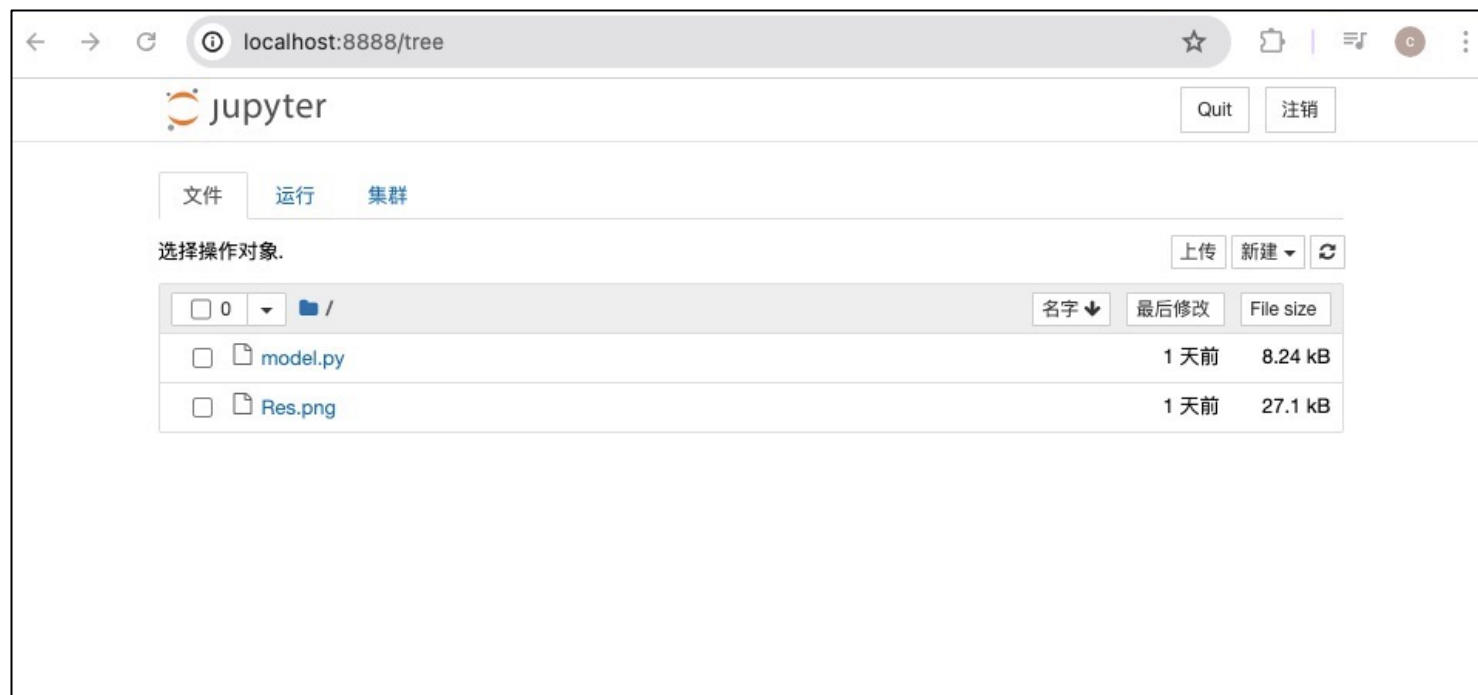
2 学习python工具——Jupyter notebook

□什么是 Jupyter Notebook?

- Jupyter Notebook 是一个开源的 web 应用程序，可以创建和共享包含代码、方程式、可视化和文本的文档，广泛用于数据分析、机器学习、数据可视化和教育。
- 执行python命令所见即所得

□Jupyter使用

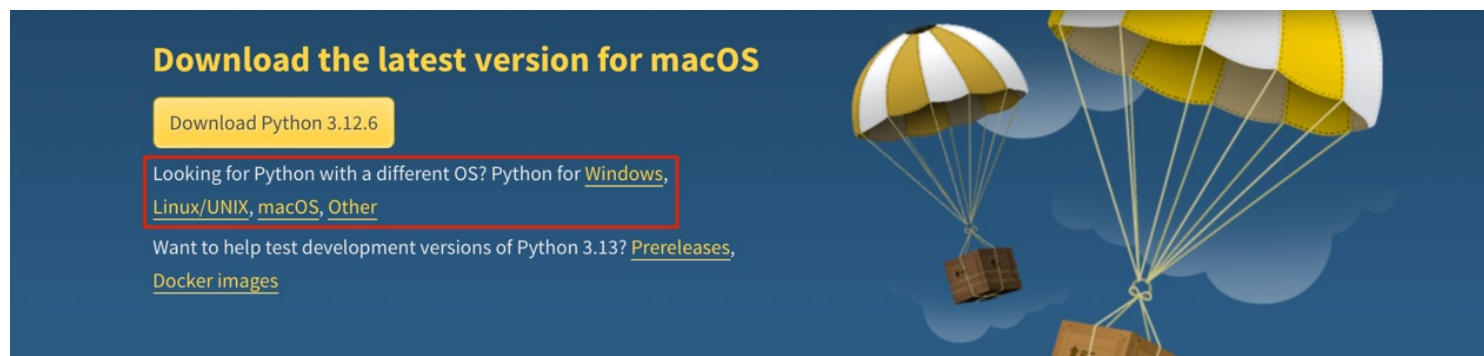
- 安装命令：
pip3 install notebook
- 启动命令：
jupyter notebook
- 退出：直接关闭窗口



2 Python IDLE开发环境

□ Python 官网 : <https://www.python.org/downloads>

□ 选择操作系统



□ 选择Python版本

- [Python 3.12.6 - Sept. 6, 2024](#)
Note that Python 3.12.6 *cannot* be used on Windows 7 or earlier.
 - [Download Windows installer \(64-bit\)](#)
 - [Download Windows installer \(32-bit\)](#)
 - [Download Windows installer \(ARM64\)](#)
 - [Download Windows embeddable package \(64-bit\)](#)
 - [Download Windows embeddable package \(32-bit\)](#)
 - [Download Windows embeddable package \(ARM64\)](#)
- [Python 3.9.20 - Sept. 6, 2024](#)
Note that Python 3.9.20 *cannot* be used on Windows 7 or earlier.
 - No files for this release.

2 运行程序的方式

□ 方式一：交互式解释器执行

```
C:\Users\cy>python
Python 3.12.4 | packaged by Anaconda, Inc. |
Type "help", "copyright", "credits" or "licen
>>> 3+5*6
33
>>> print("hello world")
hello world
```

□ 方式二：命令行环境运行

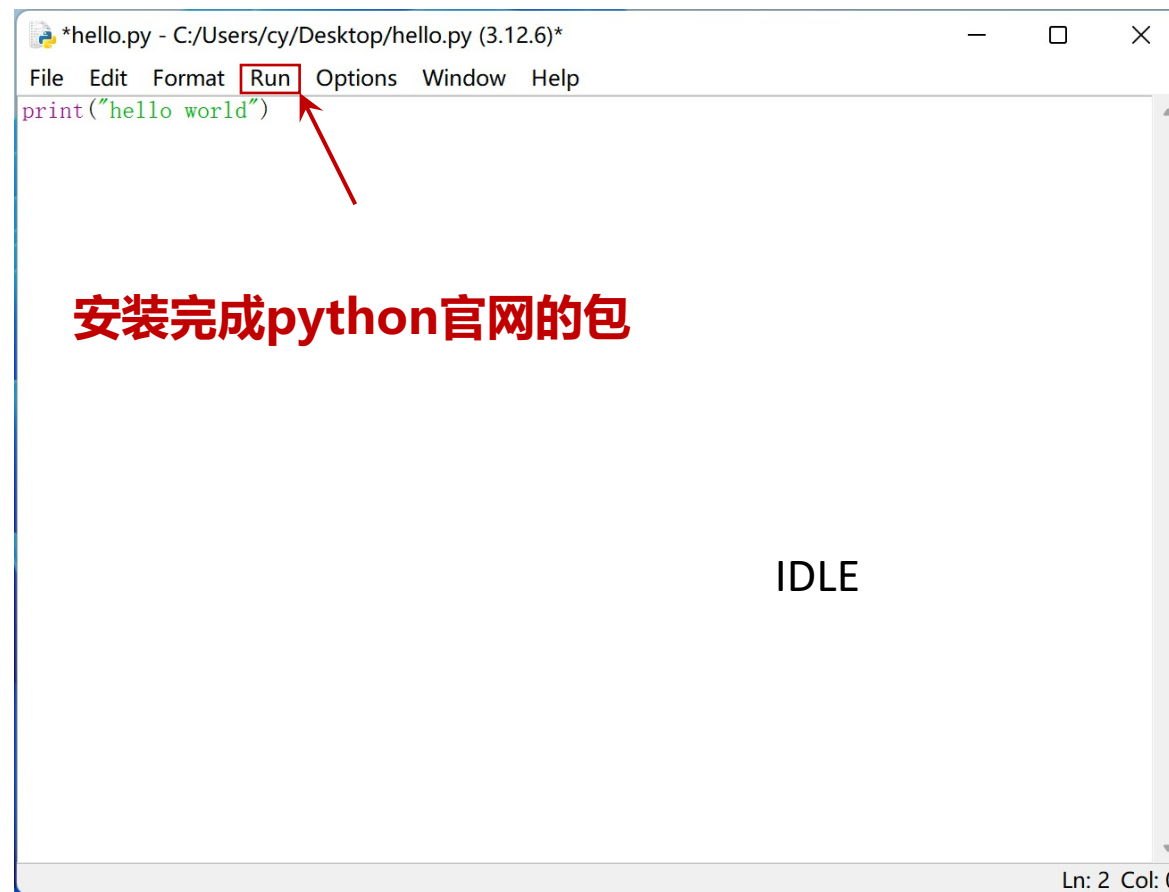
```
C:\Users\cy>python hello.py
hello world
```

hello.py是python程序

hello.py:

```
print("hello world")
```

□ 方式三：File | New, Save, Run



2 Python基础语法

❑ 菜鸟教程：<https://www.runoob.com/python/python-basic-syntax.html>

❑ 基础语法

- ❑ python变量

- ❑ python运算符

- ❑ python语句：条件语句、循环语句（While、for）、break、continue语句

- ❑ python列表（数组）

- ❑ python字符串

- ❑ python函数

❑ 高级编程

- ❑ 面向对象

- ❑ 正则表达式

- ❑ ...

2 Python函数

- 组织好的，可重复使用的，用来实现单一，或相关联功能的代码段
- 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号()。

函数定义

```
def functionname( parameters ):  
    "函数_文档字符串"  
    function_suite  
    return [expression]
```

示例

```
# 定义函数  
def printme( str ):  
    "打印传入的字符串到标准显示设备上"  
    print (str)  
    return  
  
# 调用函数  
printme("我要调用用户自定义函数!")  
printme("再次调用同一函数")
```

2 Python高级语法

□**类(Class)**: 用来描述具有相同的属性和方法的对象的集合。定义了该集合中每个对象所共有的属性和方法。对象是类的实例。

□**方法**: 类中定义的函数。

□**对象**: 通过类定义的数据结构实例。对象包括两个数据成员（类变量和实例变量）和方法。

□**类变量**: 类变量在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。

□**数据成员**: 类变量或者实例变量, 用于处理类及其实例对象的相关的数据。

□**实例化**: 创建一个类的实例, 类的具体对象。

□**方法重写**: 如果从父类继承的方法不能满足子类的需求, 可以对其进行改写, 这个过程叫方法的覆盖 (override), 也称为方法的重写。

□**局部变量**: 定义在方法中的变量, 只作用于当前实例的类。

□**实例变量**: 在类的声明中, 属性是用变量来表示的。这种变量就称为实例变量, 是在类声明的内部但是在类的其他成员方法之外声明的。

□**继承**: 即一个派生类 (derived class) 继承基类 (base class) 的字段和方法。继承也允许把一个派生类的对象作为一个基类对象对待。例如, 有这样一个设计: 一个Dog类型的对象派生自Animal类, 这是模拟"是一个 (is-a)"关系 (例图, Dog是一个Animal)。

2 Python模块与包

□模块：包含Python代码的文件

□包：包含多个模块的文件夹

□导入模块的方法

- import 模块名

- from 模块名 import 功能名

- from 模块名 import *

- import 模块名 as 别名

- from 模块名 import 功能名 as 别名

□导入包的方法

- import package # 导入整个包

- import package.module1 # 导入包中的模块

- from package.module1 import function # 从包中的模块导入特定功能

```
>>> import math
>>> print(math.sqrt(16))
4.0
```

```
>>> from math import sqrt
>>> print(sqrt(16))
4.0
```

```
>>> from math import *
>>> print(sqrt(16))
4.0
>>> print(exp(2))
7.38905609893065
```

```
>>> import math as m
>>> print(m.sqrt(16))
4.0
```

```
>>> from math import sqrt as m_sqrt
>>> print(m_sqrt(16))
4.0
```

2 Python高级语法-class举例

- 使用 class 语句来创建一个新类，class 之后为类的名称并以冒号结尾。

```
class ClassName:
    '类的帮助信息' #类文档字符串
    class_suite #类体
```

举例：1、员工信息类的定义创建

```
class Employee:
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print("Total StuCount %d", Employee.empCount)

    def displayEmployee(self):
        print("Name : ", self.name, ", Salary: ", self.salary)
```

2、实例化和调用类的函数

```
emp1=Employee("zzz1",2000)
emp1.displayCount()
emp1.displayEmployee()
emp2=Employee("zzz2",3000)
emp2.displayCount()
emp2.displayEmployee()
```

3、运行结果

```
Total StuCount %d 1
Name :   zzz1 , Salary:   2000
Total StuCount %d 2
Name :   zzz2 , Salary:   3000
[Finished in 568ms]
```

2 Python异常处理

□通过 try 和 except 语句来实现，捕获和处理在程序执行过程中可能出现的错误，从而避免程序的崩溃。

基本语法

```
try:
    # 可能会引发异常的代码
    pass
except SomeException as e:
    # 异常处理代码
    pass
```

示例

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
```




提纲

- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

3 什么是 Anaconda?

□ Anaconda 是一个开源的 Python 和 R 数据科学平台，旨在简化包管理和环境管理，提供对数据科学和机器学习工具的支持，包含 Conda 包管理器和环境管理等。

核心组件：

- **Conda**：包管理和环境管理工具
- **Anaconda Navigator**：图形用户界面用于管理环境和包
- **Jupyter Notebook**：交互式计算环境
- **Spyder**：Python IDE 适用于数据分析

.....



3 Anaconda安装指南

□官方网站：<https://www.anaconda.com/download/>


□选择操作系统及对应python版本

Download Now




For installation assistance, refer to [Troubleshooting](#).

Download Distribution by choosing the proper installer for your machine.

Download for Mac



Anaconda Installers

 Windows Python 3.12 ↓ 64-Bit Graphical Installer (912.3M)	 Mac Python 3.12 ↓ 64-Bit (Apple silicon) Graphical Installer (704.7M) ↓ 64-Bit (Apple silicon) Command	 Linux Python 3.12 ↓ 64-Bit (x86) Installer (1007.9M) ↓ 64-Bit (AWS Graviton2 / ARM64) Installer (800.6M)
------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3 Conda – 包和环境管理器

- 包管理：安装、更新和删除软件包
- 环境管理：创建和管理虚拟环境，支持多个项目使用不同的依赖
- 常用命令：
 - 创建环境：conda create -n myenv python=3.12.0
 - 激活环境：conda activate myenv
 - 退出环境：conda deactivate
 - 安装包：conda install numpy

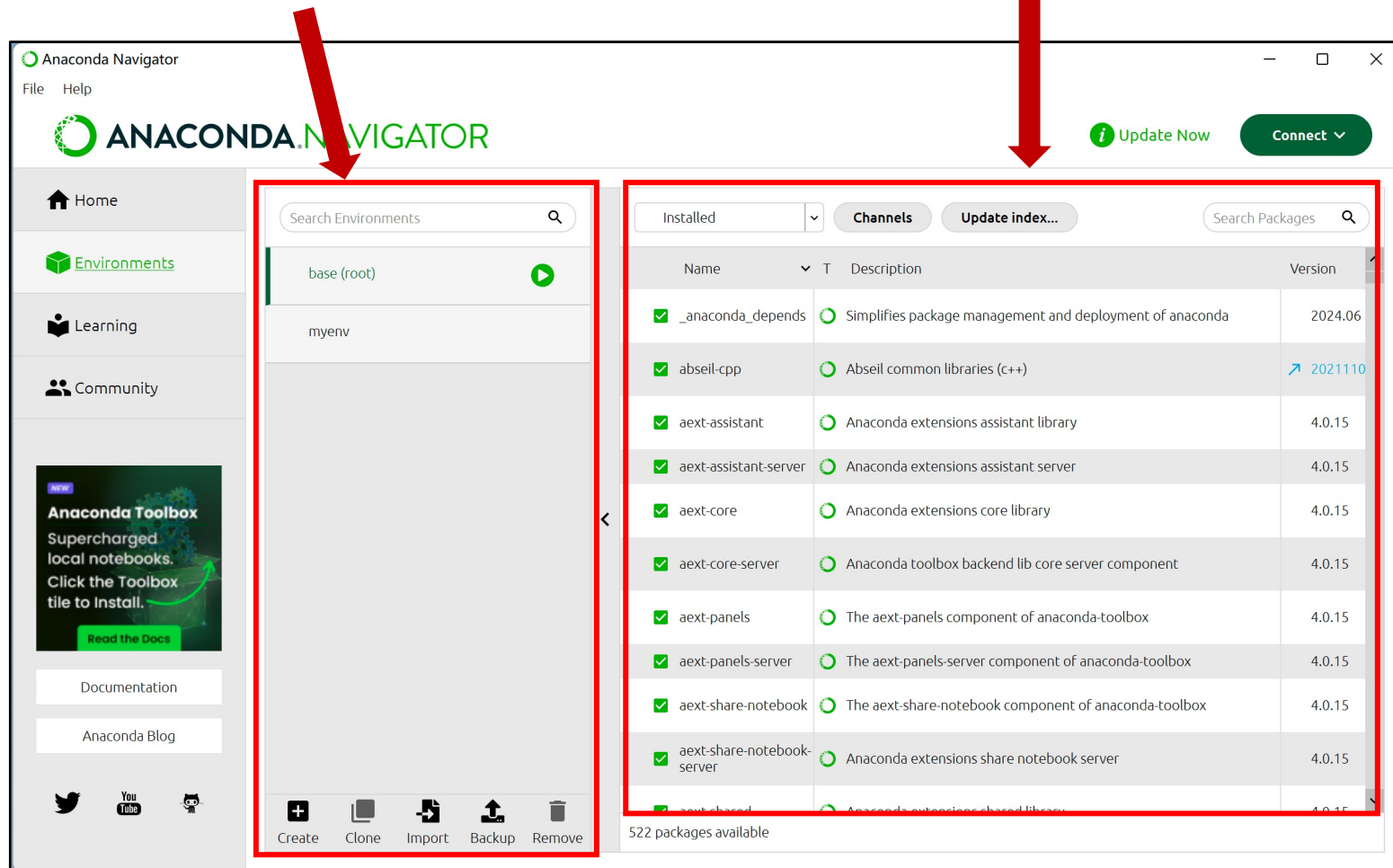
```
C:\Users\cy>conda create --name myenv
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

```
C:\Users\cy>conda activate myenv
(myenv) C:\Users\cy>
```

```
(myenv) C:\Users\cy>conda install numpy
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

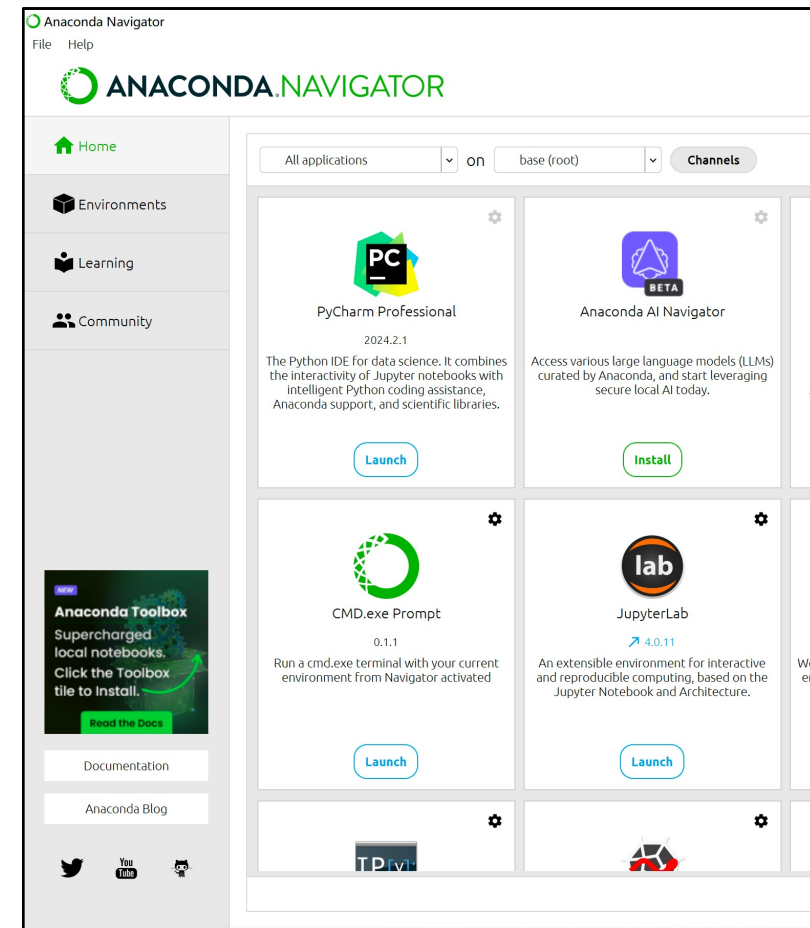
3 Anaconda Navigator – 图形用户界面

□ 虚拟环境管理



□ 包管理

□ 安装和启动应用



3 什么是 PyCharm?

- PyCharm 是由 JetBrains 开发的一款强大的集成开发环境（IDE），专为 Python 开发而设计。
- 目标：提供全面的开发工具和支持，提升 Python 开发的效率和质量。

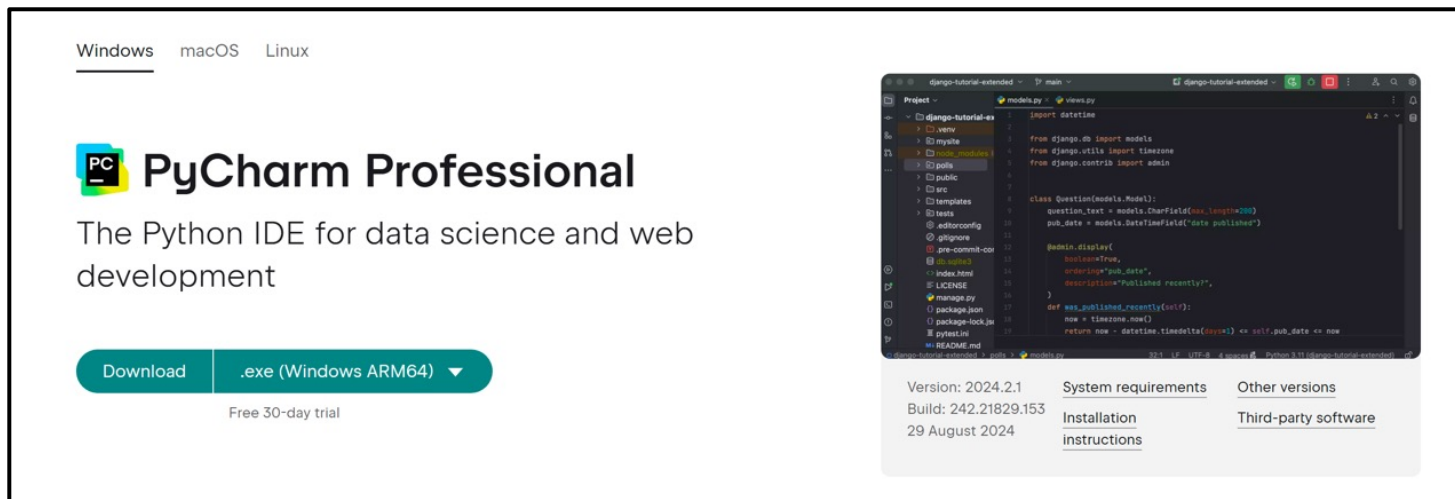


核心功能：

- **智能代码编辑**：自动补全、实时错误检查和代码重构
- **调试工具**：强大的调试功能，支持断点、变量监视等
- **测试支持**：支持单元测试、集成测试和测试覆盖率分析
- **版本控制**：集成 Git、SVN 等版本控制系统
- **环境管理**：支持虚拟环境和 Conda 环境

3 PyCharm的安装

- ❑ 官方网站：<https://www.jetbrains.com/pycharm/download>
- ❑ 选择对应的操作系统并下载



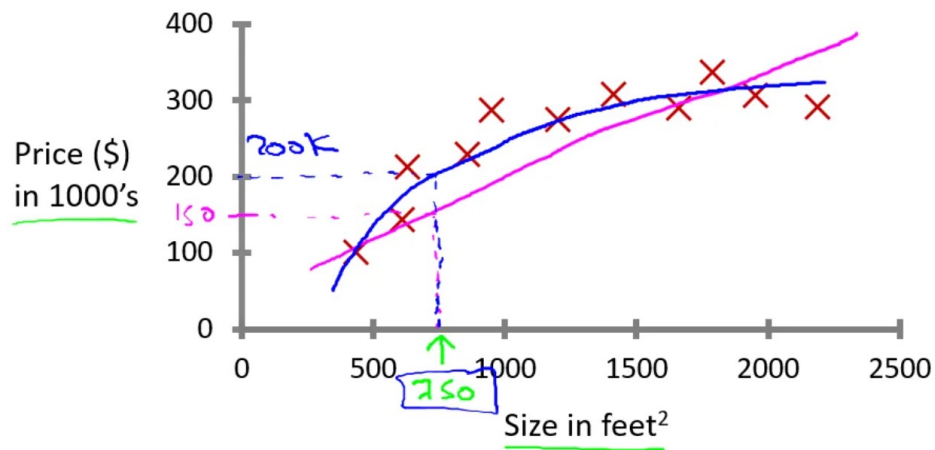


提纲

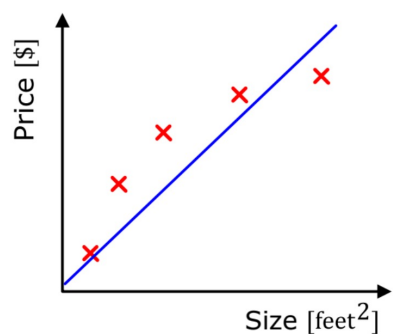
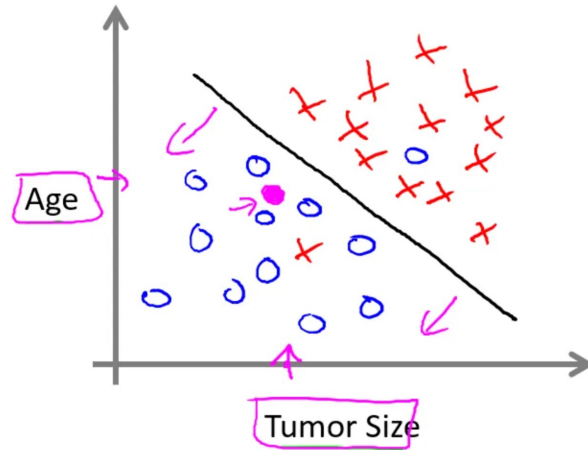
- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

4 深度学习算法简介：数学基础

□ 经过算法预测的结果是一个连续的值，我们称这样的问题为回归问题。

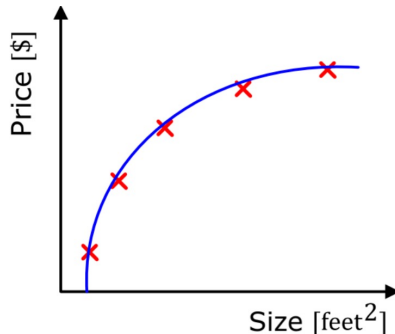


□ 算法能够学会如何将数据分类到不同的类里，我们称这样的问题为分类问题。

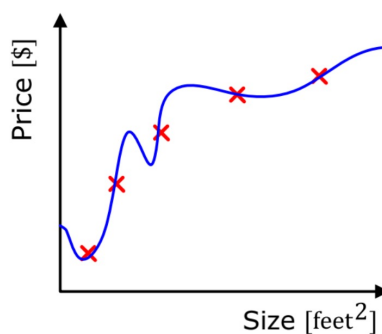


Underfit

High bias



OK



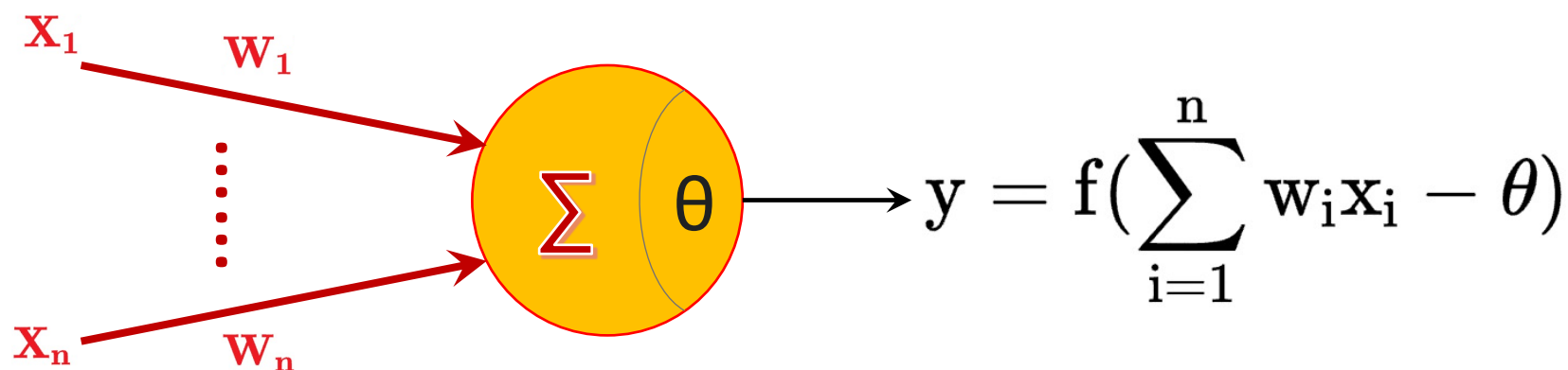
Overfit

High variance

第一个模型是一个线性模型，低度拟合，不能很好地适应训练集；第三个模型是一个四次多项式的模型，过度拟合，虽然能非常好地适应我们的训练集，但在新输入变量进行预测时可能会效果不好；中间的模型似乎最合适。

4 深度学习算法简介：感知器

- 感知器是一个由线性阈值元件组成的单层（或多层）神经元的神经网络
- 当输入的加权和大于或等于阈值时，输出为1，否则为0
- 模型假定神经元中间的耦合程度（即加权系数 W ）可变，这样，该模型可以学习



- 当感知器用于两类模型的分类时，相当于在高维样本空间中，用一个超平面将两类样本分开
- 神经网络的学习过程就是神经网络参数的设定过程
- 一个神经网络结构确定之后，需要对一系列参数（权重、阈值等）进行有效的设定。这个过程叫做学习或训练过程，此时的方法叫学习算法

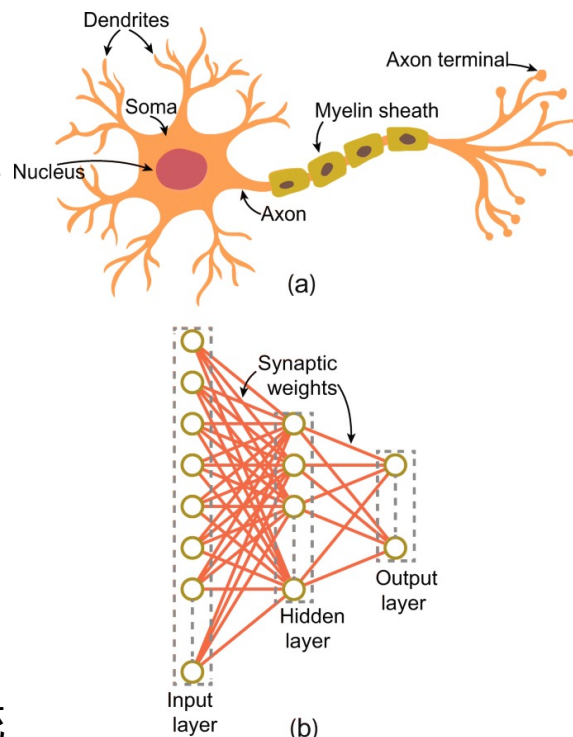
4 深度学习的训练方法

□ 监督学习

通过已有的**训练样本（即已知数据以及其对应的输出）**训练得到一个最优模型（这个模型属于某个函数的集合，最优则表示在某个评价准则下是最佳的），再利用这个模型将所有的输入映射为相应的输出，对输出进行简单的判断从而实现分类的目的，从而也就具有了对未知数据进行分类的能力

□ 强化学习

类似人类与环境交互的方式，智能系统从环境到行为映射的学习，以使奖励信号函数值最大。**环境对产生动作的好坏通过奖励信号作评价**，而不是告诉强化学习系统如何去产生正确的动作。**强化学习不能立即得到标记，而只能得到一个反馈**，因此可以说强化学习是一种具有“延迟标记信息”的监督学习
典型案例：AlphaGo



各种神经网络类型

- Logistic
- RBM
- Auto Encoder
- Sparse Coding
- **Convolutional (卷积)**

2006年，Geoffrey Hinton在《科学》上发表论文提出深度学习主要观点：

- 多隐层的人工神经网络具有优异的特征学习能力，学习得到的特征对数据有更本质的刻画，从而有利于可视化或分类
- 深度神经网络在训练上的难度，可以通过“逐层初始化”（layer-wise pre-training）来有效克服，逐层初始化可通过无监督学习实现的
- 在著名的ImageNet问题上将错误率从26%降低到了15%，并且输入没有用到其他任何人工特征，仅仅是图像的像素

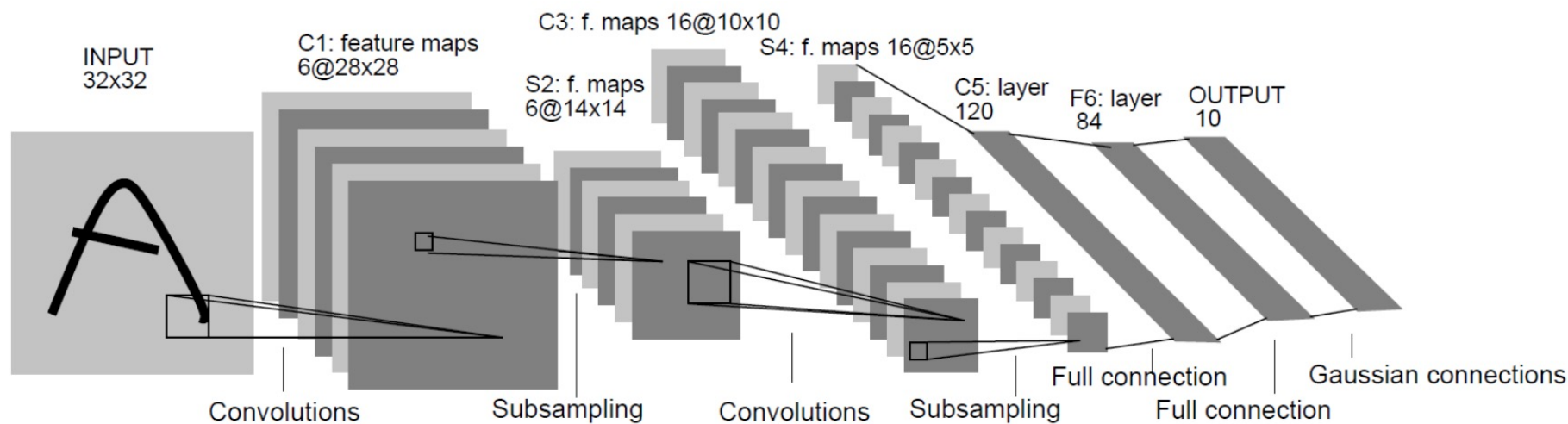
□ 迁移学习

将从拥有大数据的源领域上学习到的东西应用到仅有小数据的目标领域上去，实现个性化迁移，即举一反三、触类旁通。

典型案例：斯坦福学者使用卫星图像获取的灯光信息来分析非洲大陆的贫穷情况

4

深度学习的神经网络训练方法



输入图片大小： 32×32

卷积窗大小： 5×5

卷积窗种类： 6

输出特征图数量： 6

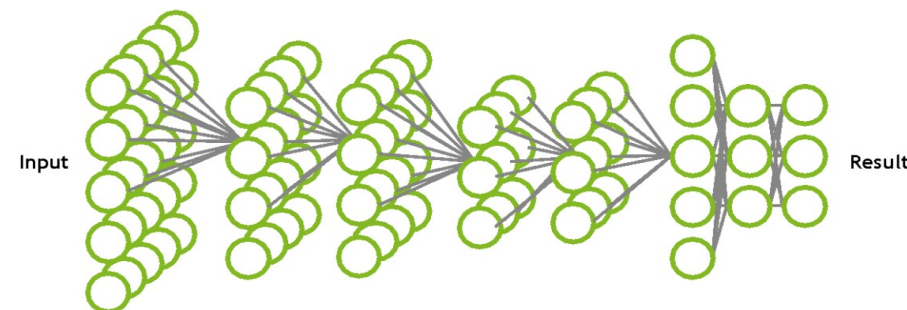
输出特征图大小： 28×28 ($32 - 5 + 1$)

神经元数量： 4704 [$(28 \times 28) \times 6$]

连接数： 122304 [$(5 \times 5 + 1) \times 6 \times$

(28×28)]

可训练参数： 156 [$(5 \times 5 + 1) \times 6$]

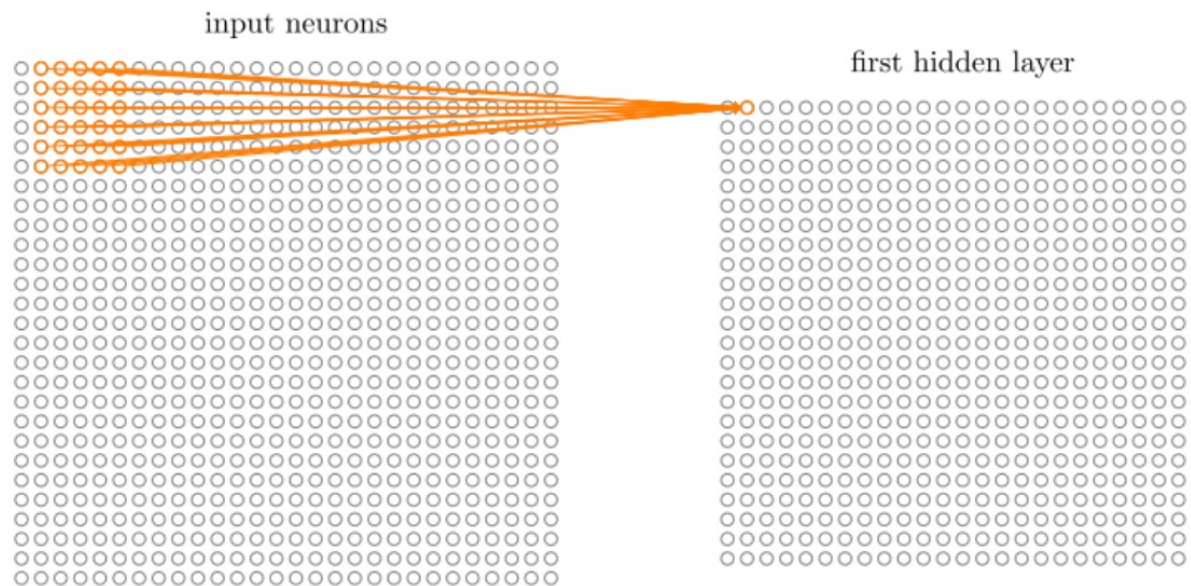
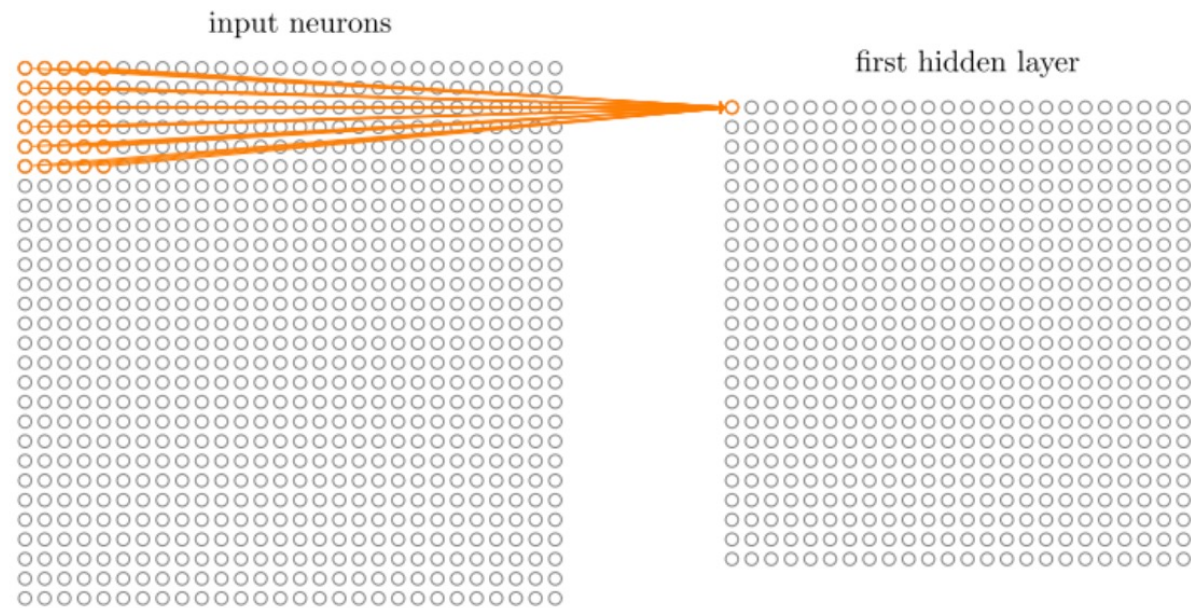
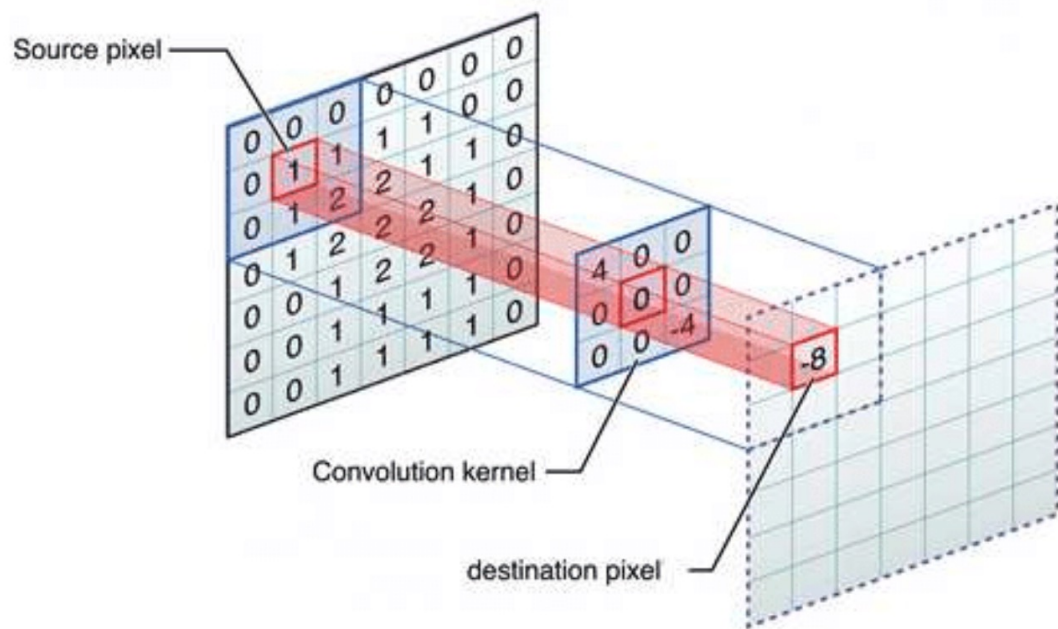


LeNet-5：卷积神经网络手写数字识别的应用

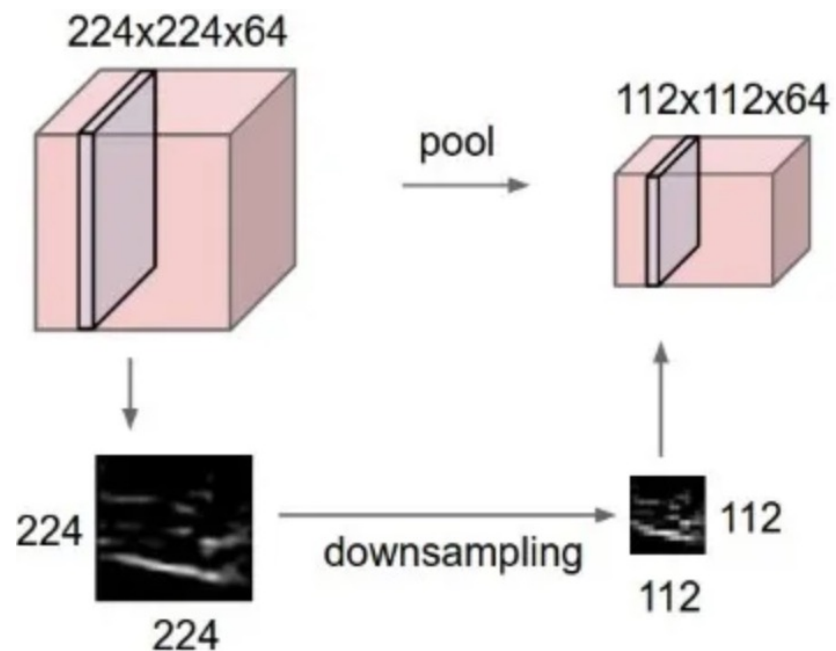
- 卷积过程包括：用一个可训练的滤波器 f_x 去卷积一个输入的图像，然后加一个偏置 b_x ，得到卷积层 C_x 。
- 子采样过程包括：每邻域四个像素求和变为一个像素，加权再增加偏置，通过一个激活函数，产生一个缩小四倍的特征映射图

4

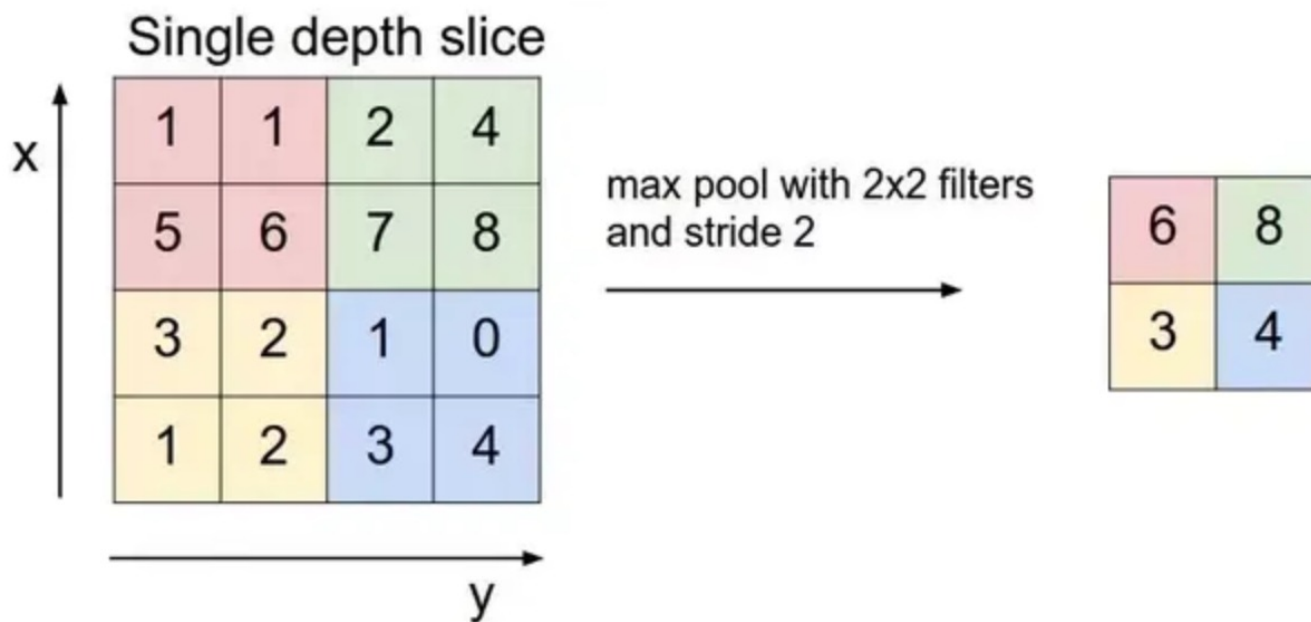
局部感受—卷积



4 池化

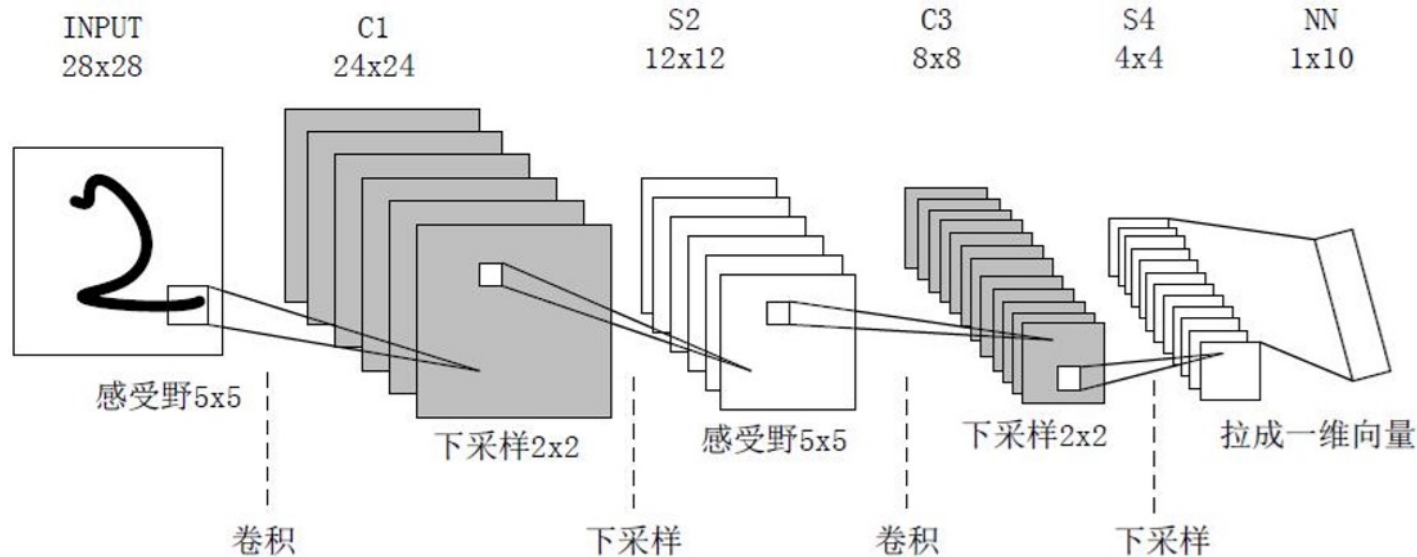
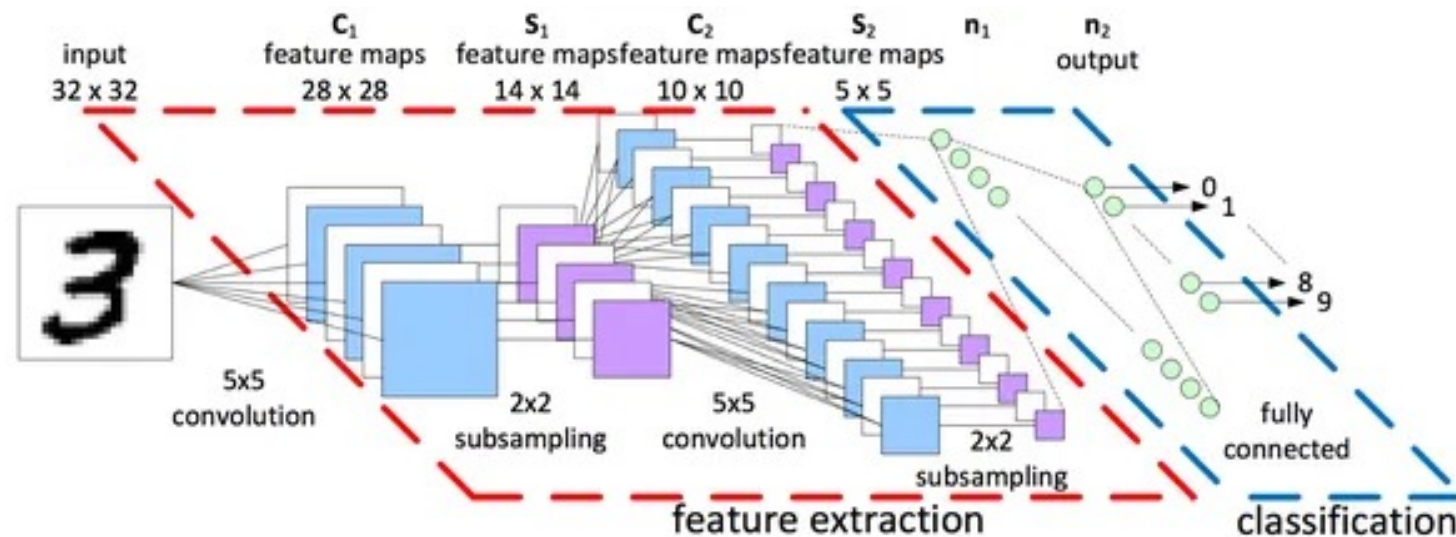


- 原理：根据图像局部相关的原理，图像某个邻域内只需要一个像素点就能表达整个区域的信息
- 常见的方法：
 - 最大值池化 (max-pooling)
 - L2池化 (L2 pooling)
 - 均值池化 (Mean Pooling)



4 LeNet-5

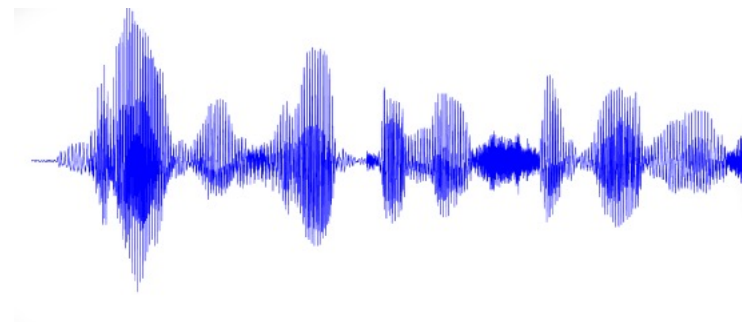
- Yann Lecun, 1989年用美国邮政系统提供的近万个手写数字的样本来训练神经网络系统，在独立的测试样本中，错误率只有5%



- 进一步运用CNN，开发出LeNet-5用于读取银行支票上的手写数字，这个支票识别系统在九十年代末占据了美国接近20%的市场

4 深度学习的成功应用

- AlphaGo战胜李世乭
- 图像识别全面超越人类
- 语音识别接近人类
 - 将声学模型中混合高斯模型替换为DNN模型
 - 获得30% + 相对提升
- Tesla Autopilot投入商用
- Google Translate投入商用
 - 它把原文例如中文词先翻成一个词向量，变成一个数字向量。
 - 它对这个词向量再编辑，变成一个语义表示的方式。
 - 再把它翻译成它的目标语言，例如英文。



4 TensorFlow与PyTorch

- TensorFlow作为一个开源的机器学习框架，由Google公司开发并在2015年发布。
- 它以其灵活性、高效性和易用性而闻名，被广泛应用于学术界和工业界。
- TensorFlow支持分布式计算，可以在多种平台上运行，如PC、服务器、手机等。
- 同时，TensorFlow提供了丰富的API，可以快速构建和训练各种机器学习模型。

- PyTorch算是相当简洁优雅且高效快速的框架，设计追求最少的封装，尽量避免重复造轮子
- 算是所有的框架中面向对象设计的最优雅的一个，设计最符合人们的思维，它让用户尽可能地专注于实现自己的想法
- 与Google的Tensorflow类似，FAIR的支持足以确保PyTorch获得持续的开发更新
- 不错的文档（相比FB的其他项目，PyTorch的文档简直完善，参考Thrift），PyTorch作者亲自维护的论坛 供用户交流和求教问题
- 入门简单

4 PyTorch是什么

□PyTorch是什么

- Facebook于2017年发布的**机器学习框架**，**基于Python的科学计算包**，服务于以下两种场景：**使用GPU的强大计算能力**，提供最大的灵活性和高速的**深度学习研究平台**
- 以其灵活性和直观的方式而受到广大开发者的青睐，支持**动态图**，使得**调试和开发过程**更加直观和高效。
- 提供丰富的**预训练模型和工具**，方便开发者进行**迁移学习和快速原型设计**。

□PyTorch的由来

- Torch**是一个与**Numpy**类似的**张量 (Tensor)** 操作库，与Numpy不同的是**Torch对GPU支持的很好**，**Lua**是Torch的上层包装。

注：LUA虽然快，但是太小众了，所以才会有PyTorch的出现。

- PyTorch和Torch使用包含所有相同性能的C库**：TH, THC, THNN, THCUNN，但使用了不同的上层包装语言。主要由Facebook的人工智能研究小组开发。Uber的"Pyro"也是使用的这个库。

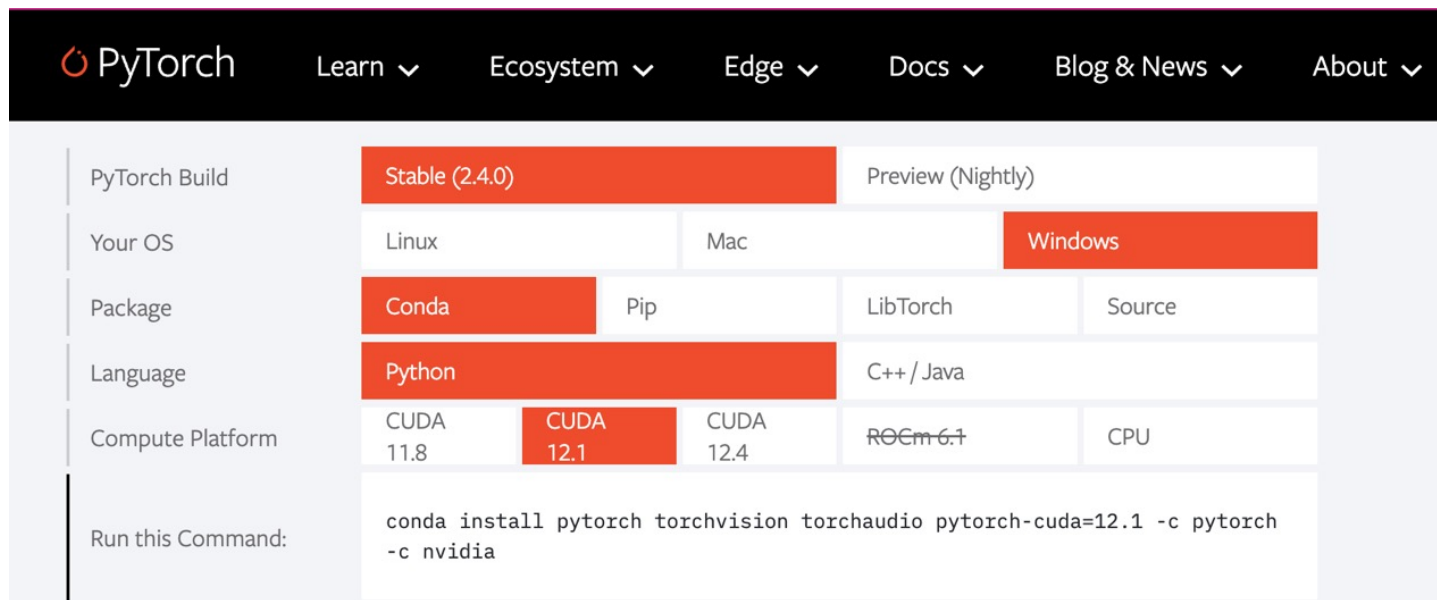
4 PyTorch是什么

□ PyTorch环境搭建

□ **PyTorch的安装十分简单**，根据**PyTorch官网**(<https://pytorch.org/>)，对系统选择和安装方式等灵活选择即可。

□ 在1.2版本以后，pytorch只支持cuda 9.2以上了，所以需要对cuda进行升级

□ 目前测试大部分显卡都可以用，包括笔记本的MX250也是可以顺利升级到cuda 10.1。



The screenshot shows the PyTorch website's installation guide. The navigation bar includes links for Learn, Ecosystem, Edge, Docs, Blog & News, and About. The main content area is a grid of options for installing PyTorch. The selected options are: Stable (2.4.0) for the build, Windows for the OS, Conda for the package, Python for the language, and CUDA 12.1 for the compute platform. The resulting command to run is: `conda install pytorch torchvision torchaudio pytorch-cuda=12.1 -c pytorch -c nvidia`.

PyTorch Build	Stable (2.4.0)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	ROCm 6.1
Run this Command:	<code>conda install pytorch torchvision torchaudio pytorch-cuda=12.1 -c pytorch -c nvidia</code>			

4 PyTorch

□大多数深度学习框架的核心都是围绕Tensor与Autograd。

□Tensor为PyTorch的核心数据结构，类似于多维数组，用于存储和操作模型的输入输出以及模型参数。

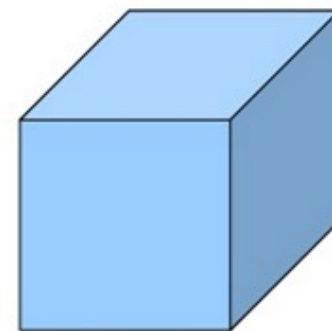
□Autograd是PyTorch中的一个核心特性，用于自动求导，极大地简化了构建和训练神经网络的过程。



1d-tensor



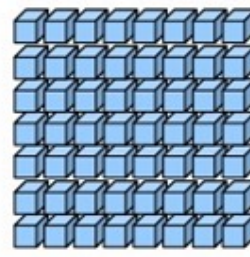
2d-tensor



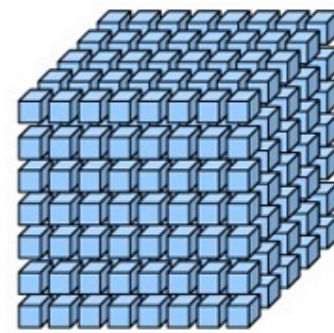
3d-tensor



4d-tensor



5d-tensor



6d-tensor

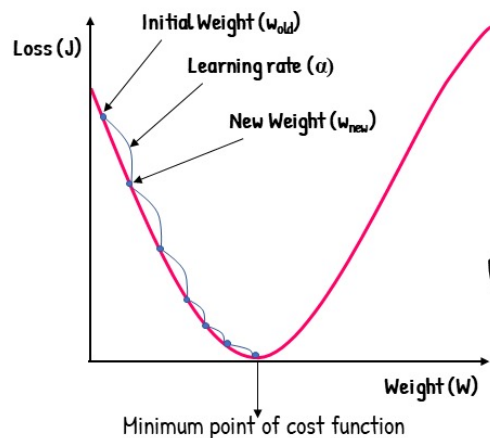
4 PyTorch

□ 梯度下降法是一种优化算法，用于训练机器学习模型和神经网络。

□ 训练数据可以帮助模型不断学习，梯度下降算法中的**成本函数**就像是晴雨表，通过每次**参数更新的迭代**来**衡量模型的准确度**。

□ 该模型持续调整其参数，直至**该函数接近于或等于零**，以使产生的误差尽可能最小。

Gradient Descent



$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\delta J}{\delta w}$$

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^m [h_{\Theta}(x_i) - y_i]^2$$

↑ ↑
Predicted Value True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

↑
Learning Rate

Now,

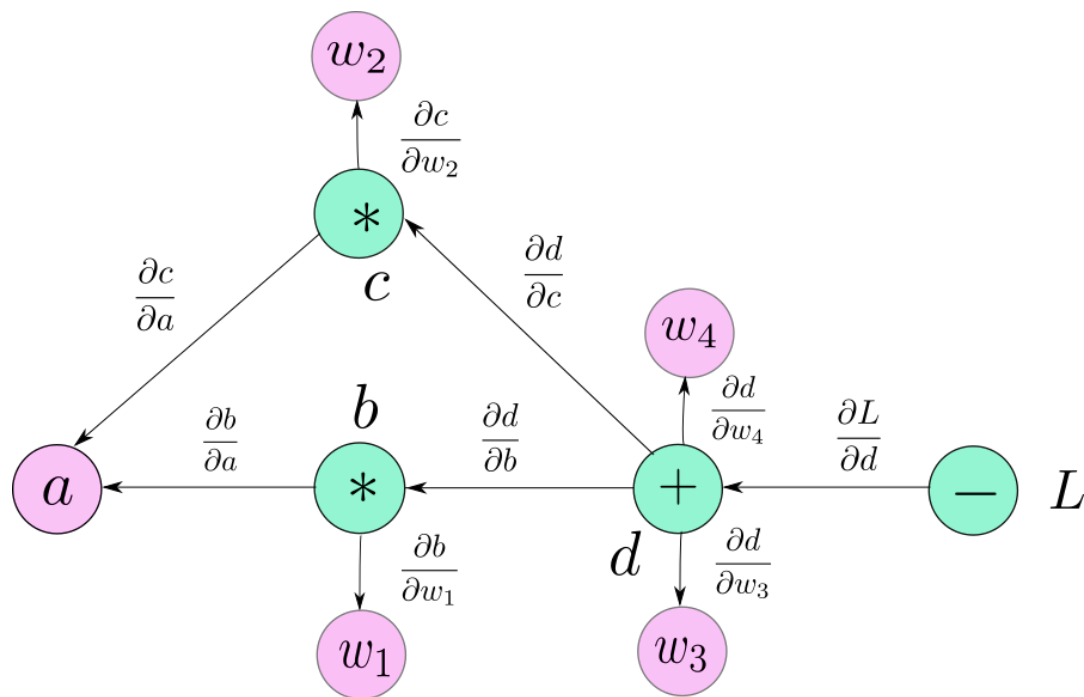
$$\begin{aligned} \frac{\partial}{\partial \Theta} J_{\Theta} &= \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^m [h_{\Theta}(x_i) - y]^2 \\ &= \frac{1}{m} \sum_{i=1}^m (h_{\Theta}(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y) \\ &= \frac{1}{m} (h_{\Theta}(x_i) - y) x_i \end{aligned}$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^m [(h_{\Theta}(x_i) - y) x_i]$$

4 PyTorch– Autograd

□ **Autograd**：深度学习模型的训练依靠梯度下降法，而Autograd计算图是PyTorch实现的**自动**求导（面对一个复杂的深度学习模型采用手动求导是不现实的）算法，其本质为一个有向无环图，即可在前行传播的过程中自动求导（利用链式法则），可在后向传播的过程中自动求导。



$$b = \omega_1 * a$$

$$c = \omega_2 * a$$

$$d = \omega_3 + \omega_4 + c + b$$

$$L = d$$

$$\frac{\partial L}{\partial d} = 1$$

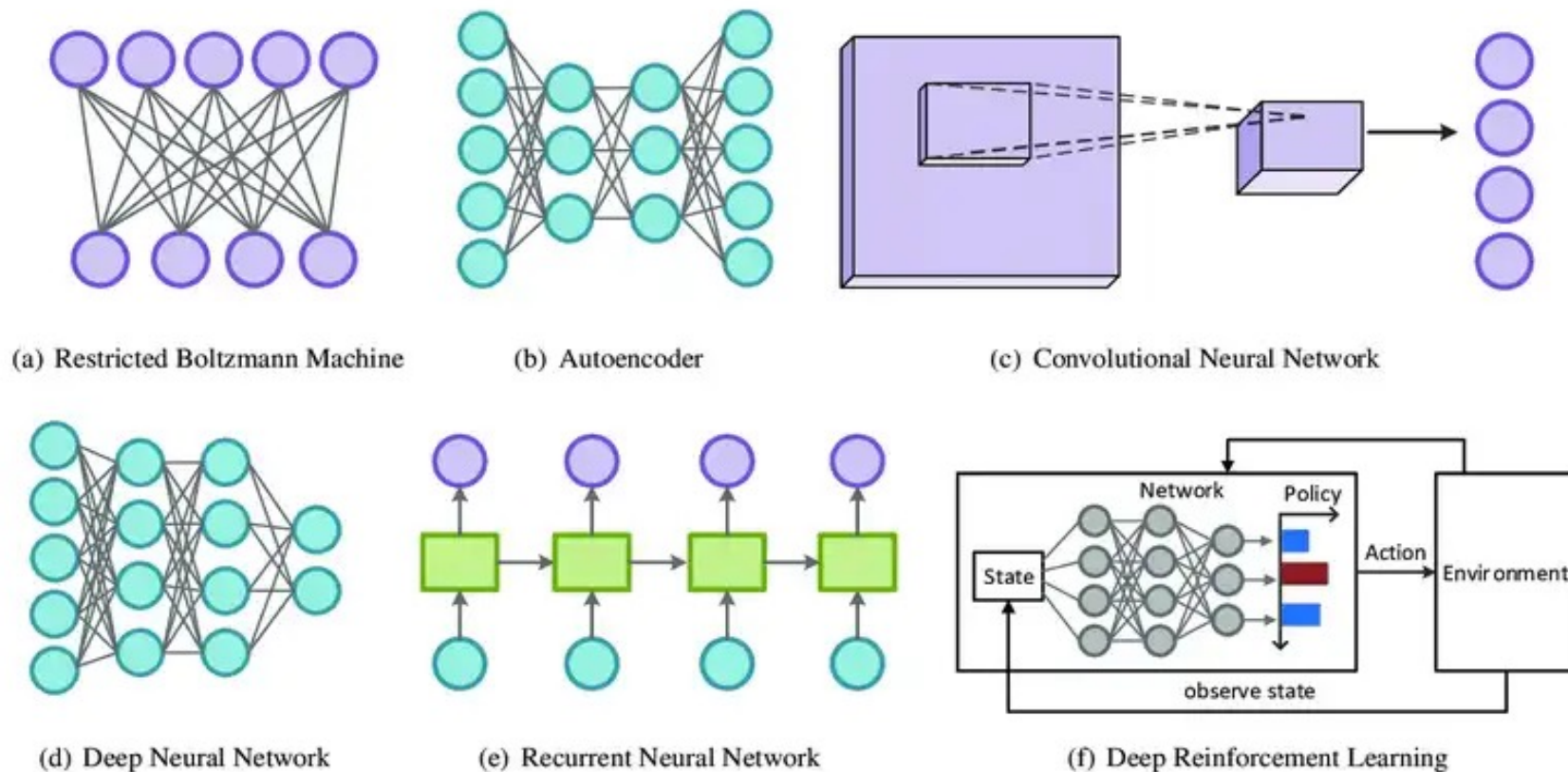
$$\frac{\partial d}{\partial \omega_3} = \frac{\partial d}{\partial \omega_4} = \frac{\partial d}{\partial c} = \frac{\partial d}{\partial b} = 1$$

$$\frac{\partial b}{\partial \omega_1} = a \quad \frac{\partial b}{\partial a} = \omega_1$$

$$\frac{\partial c}{\partial \omega_2} = a \quad \frac{\partial c}{\partial a} = \omega_2$$

4 PyTorch-深度学习

深度学习发展至今已产生诸如卷积神经网络、循环神经网络、生成对抗网络与Transformer等许多模型。

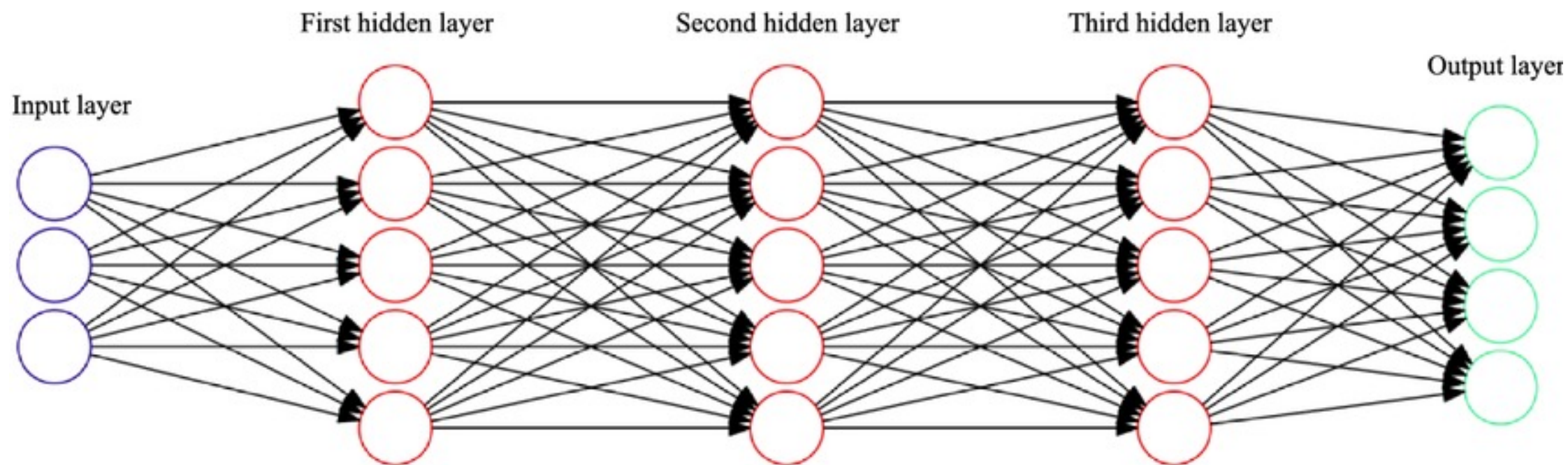


其本质都是输入在通过由参数描述的模型后得到输出，输出与实际输出做运算构成损失函数，再对损失函数运用梯度下降法得到对模型中每个参数的更新量，参数更新后再开始新一轮的训练，直到损失值下降到一定程度。

4 PyTorch

■全连接网络基本概念

- 全连接神经网络是一种多层感知器，由输入层、隐藏层和输出层组成。
- 其中，输入层负责接收外部输入的特征向量，隐藏层通过非线性变换对输入进行压缩或抽象，输出层则产生最终的输出结果。
- 在全连接神经网络中，每个神经元都与上一层的所有神经元相连，因此其网络结构可以看作是一个非常深的卷积神经网络。



4 PyTorch—如何学习

□ 官方文档永远是最好的资料

<https://pytorch.org/docs/stable/index.html>

Community [+]
Developer Notes [+]
Language Bindings [+]
Python API [-]
torch
torch.nn
torch.nn.functional
torch.Tensor
Tensor Attributes
Tensor Views
torch.amp
torch.autograd
torch.library
torch.cpu
torch.cuda
Understanding CUDA Memory Usage
Generating a Snapshot
Using the visualizer
Snapshot API Reference
torch.mps
torch.xpu
Meta device
torch.backends
torch.export
torch.distributed
torch.distributed.algorithms.join

PyTorch documentation

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.

Features described in this documentation are classified by release status:

Stable: These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).

Beta: These features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification. We are not, however, committing to backwards compatibility.

Prototype: These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.

Community

- [PyTorch Governance | Build + CI](#)
- [PyTorch Contribution Guide](#)
- [PyTorch Design Philosophy](#)
- [PyTorch Governance | Mechanics](#)
- [PyTorch Governance | Maintainers](#)

PyTorch documentation
Indices and tables

Conv2d

```
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None) [SOURCE]
```

Linear

```
CLASS torch.nn.Linear(in_features, out_features, bias=True, device=None, dtype=None) [SOURCE]
```

Applies a linear transformation to the incoming data: $y = xA^T + b$.

Dropout

```
CLASS torch.nn.Dropout(p=0.5, inplace=False) [SOURCE]
```

During training, randomly zeroes some of the elements of the input tensor with probability p .



提纲

- 1 机器识别**
- 2 Python入门**
- 3 集成开发环境**
- 4 主流深度学习框架**
- 5 AI算法库和开发工具包**

5 AI算法库—数据处理和分析—Pandas

- ❑ Pandas 是一个开源数据分析库，广泛用于数据科学、数据分析和机器学习领域。
- ❑ 基于 NumPy 构建，主要用于 Python 语言，充分利用了 NumPy 的高性能计算能力。
- ❑ 提供高效、灵活的数据结构，允许用户轻松地处理和分析数据，其核心数据结构主要有：
 - ❑ Series: 一维数组，带有标签
 - ❑ DataFrame: 二维表格，类似于 Excel 表格
 - ❑ Panel (较少使用): 三维数组（通常已被弃用）

□导入Pandas库

```
import pandas as pd
```

□创建 DataFrame

```
data = {
    '姓名': ['Alice', 'Bob', 'Charlie', 'Davis'],
    '年龄': [25, 30, 35, 45],
    '城市': ['北京', '上海', '广州', '杭州']
}
```

```
df = pd.DataFrame(data)
print(df)
```

	姓名	年龄	城市
0	Alice	25	北京
1	Bob	30	上海
2	Charlie	35	广州
3	Davis	45	杭州

□读取 CSV 文件

```
df = pd.read_csv('test.csv')
print(df.head()) # 打印前五
```

	Id	Dates	DayOfWeek	PdDistrict	Address \
0	0	2015-05-10 23:59:00	Sunday	BAYVIEW	2000 Block of THOMAS AV
1	1	2015-05-10 23:51:00	Sunday	BAYVIEW	3RD ST / REVERE AV
2	2	2015-05-10 23:50:00	Sunday	NORTHERN	2000 Block of GOUGH ST
3	3	2015-05-10 23:45:00	Sunday	INGLESIDE	4700 Block of MISSION ST
4	4	2015-05-10 23:45:00	Sunday	INGLESIDE	4700 Block of MISSION ST

	X	Y
0	-122.399588	37.735051
1	-122.391523	37.732432
2	-122.426002	37.792212
3	-122.437394	37.721412
4	-122.437394	37.721412

□数据选择

```
# 选择一列
print(df['姓名'])
```

0	Alice
1	Bob
2	Charlie

Name: 姓名, dtype: object

```
# 选择特定行
print(df.iloc[0]) # 第一行
```

姓名	Alice
年龄	25
城市	北京

Name: 0, dtype: object

5 AI算法库-数据处理和分析-Pandas

□添加新列

```
df['收入'] = [5000, 6000, 2000, 8000]
print(df)
```

	姓名	年龄	城市	收入
0	Alice	25	北京	5000
1	Bob	30	上海	6000
2	Charlie	35	广州	2000
3	Davis	45	杭州	8000

□数据排序

```
sorted_df = df.sort_values(by='收入', ascending=False)
print(sorted_df)
```

	姓名	年龄	城市	收入
3	Davis	45	杭州	8000
1	Bob	30	上海	6000
0	Alice	25	北京	5000
2	Charlie	35	广州	2000

□处理缺失值

```
df['收入'] = df['收入'].fillna(df['收入'].mean()) # 用均值填充缺失值
print(df)
```

□保存 DataFrame

```
df.to_csv('output.csv', index=False)
```

□数据过滤

```
# 年龄大于30的行
filtered_df = df[df['年龄'] > 30]
print(filtered_df)
```

	姓名	年龄	城市	收入
2	Charlie	35	广州	2000
3	Davis	45	杭州	8000

5 AI算法库-数值计算和科学计算-NumPy

- NumPy是Python的一个科学计算库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。
- NumPy 为开放源代码并且由许多协作者共同维护开发。
- NumPy 是运行速度非常快的数学库，主要用于数组计算，包含功能：
 - 一个强大的N维数组对象 ndarray
 - 广播功能函数
 - 整合 C/C++/Fortran 代码的工具
 - 线性代数、傅里叶变换、随机数生成

□导入NumPy库

```
import numpy as np
```

□创建 NumPy 数组

```
# 从列表创建一维数组
array_1d = np.array([1, 2, 3, 4, 5])
print(array_1d)

# 创建二维数组 (矩阵)
array_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(array_2d)

# 创建全零和全一的数组
zeros = np.zeros((2, 3)) # 2行3列
print(zeros)
ones = np.ones((3, 2)) # 3行2列
print(ones)
```

```
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

□数组运算

```
# 数组加法
array_sum = array_1d + 5
print(array_sum)

# 矩阵乘法
matrix_a = np.array([[1, 2], [3, 4]])
matrix_b = np.array([[5, 6], [7, 8]])
matrix_product = np.dot(matrix_a, matrix_b)
print(matrix_product)
```

```
[ 6  7  8  9 10]
[[19 22]
 [43 50]]
```

□数组形状变换

```
# 重新定义数组形状
reshaped_array = np.array([[1, 2, 3], [4, 5, 6]]).reshape(3, 2)
print(reshaped_array)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

5 AI算法库—数据处理和分析—NumPy

□数组切片和索引

```
# 切片
slice_array = array_2d[0, :] # 第一行
print(slice_array)
```

```
[1 2 3]
```

□随机数生成

```
# 生成随机数
random_array = np.random.rand(3, 2) # 3行2列的随机数组
print(random_array)

# 生成正态分布的随机数
normal_array = np.random.normal(loc=0.0, scale=1.0, size=(2, 3))
print(normal_array)
```

```
[[0.82774189 0.24371221]
 [0.11465594 0.39998089]
 [0.93879473 0.26901047]]
[[-0.66758089 1.00197411 1.10016157]
 [-0.96106458 0.96675692 1.11420989]]
```

□基本统计操作

```
# 计算均值
mean_value = np.mean(array_1d)
print("均值:", mean_value)

# 计算方差
variance_value = np.var(array_1d)
print("方差:", variance_value)

# 计算最大值和最小值
max_value = np.max(array_1d)
min_value = np.min(array_1d)
print("最大值:", max_value)
print("最小值:", min_value)
```

```
均值: 3.0
方差: 2.0
最大值: 5
最小值: 1
```

5 可视化工具- Matplotlib库

□ Matplotlib库是一个Python图形可视化库，通过 Matplotlib，我们可以仅编写几行代码，就生成直方图、折线图、散点图等，高效美观的展示我们的数据。

□ 安装：pip3 install matplotlib

□ 导入库

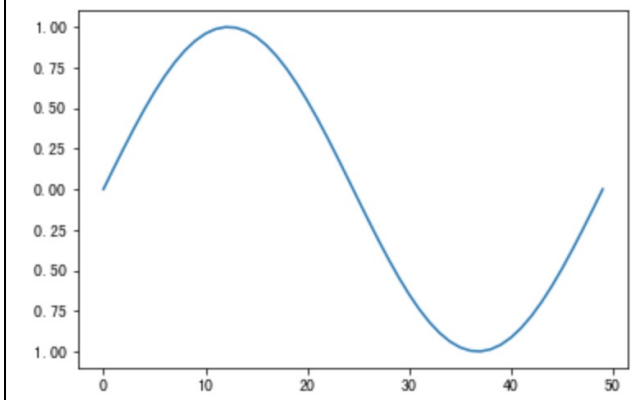
```
# 导入matplotlib和numpy
import matplotlib.pyplot as plt
from numpy import *
%matplotlib inline
```

□ Plot二维图

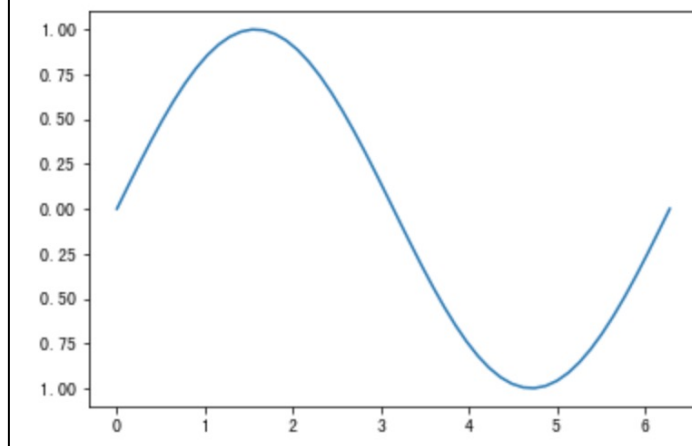
```
plt.plot(y)
plt.plot(x, y)
plt.plot(x, y, format_string)
```

□ 示例

```
# 只给定y值，默认以下标为x轴
x = linspace (0, 2 * pi, 50)
plt. plot(sin (x)) ;
```



```
# 给定x和y值:
plt.plot(x, sin (x));
```



5 机器学习库-scikit-learn

□ Scikit-learn是Python中最受欢迎的机器学习库

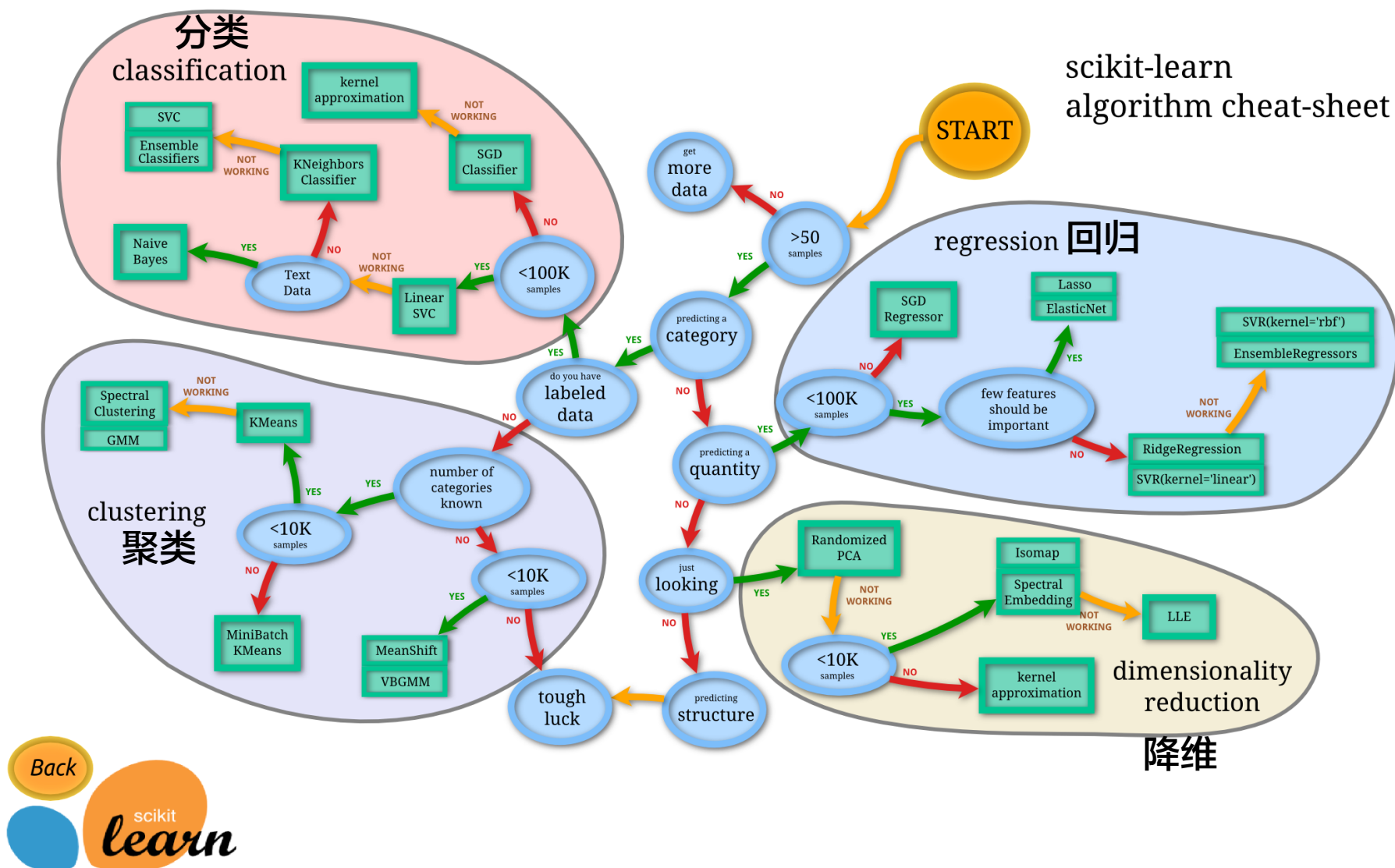
- 提供了丰富多样的机器学习算法，从简单的**线性回归**到复杂的**深度学习模型**。
- 提供了许多实用工具，帮助你进行特征工程、数据预处理、模型选择等等。
- 文档十分完善，有大量的示例代码和教程，可以轻松上手。
- Sklearn使用需要熟悉基本操作，包括如何加载数据，选择模型，训练和评估模型，以及调整模型的参数以获得更好的性能等。

□ Sklearn的下载安装

- 安装好python环境
- 输入 `python -m pip install scikit-learn` 或者
- 输入 `python -m pip install scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple`

5 机器学习库-scikit-learn

由图中，可以看到机器学习 sklearn 库的算法主要有四类：分类，回归，聚类，降维。



5 机器学习库-scikit-learn-数据集

□sklearn.datasets.load_*(())

获取小规模数据集，数据包含在 datasets 里

□sklearn.datasets.fetch_*(data_home=None)

□获取大规模数据集，需要从网络上下载，函数的第一个参数是 data_home，表示数据集下载的目录，默认是 /scikit_learn_data/

□示例-获取鸢尾花数据集

```
from sklearn.datasets import load_iris
# 获取鸢尾花数据集
iris = load_iris()
print("鸢尾花数据集的返回值: \n", iris.keys())
```

鸢尾花数据集的返回值:

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

□sklearn常见数据集

	数据集名称	调用方式	适用算法	数据规模
小数据集	波士顿房价	load_boston()	回归	506*13
小数据集	鸢尾花数据集	load_iris()	分类	150*4
小数据集	糖尿病数据集	load_diabetes()	回归	442*10
大数据集	手写数字数据集	load_digits()	分类	5620*64
大数据集	Olivetti脸部图像数据集	fetch_olivetti_faces()	降维	400*64*64
大数据集	新闻分类数据集	fetch_20newsgroups()	分类	-
大数据集	带标签的人脸数据集	fetch_lfw_people()	分类、降维	-
大数据集	路透社新闻语料数据集	fetch_rcv1()	分类	804414*47236

5 机器学习库-scikit-learn-数据预处理

□通过一些转换函数将特征数据转换成更加适合算法模型的特征数据过程。

□常见的有数据标准化、数据二值化、标签编码、独热编码等。

```
# 导入内建数据集
from sklearn.datasets import load_iris

# 获取鸢尾花数据集
iris = load_iris()

# 获得 ndarray 格式的变量 X 和标签 y
X = iris.data
y = iris.target

# 获得数据维度
n_samples, n_features = iris.data.shape

print(n_samples, n_features)
```

150 4

5 机器学习库-scikit-learn-数据标准化

□数据标准化和归一化是将数据映射到一个小的浮点数范围内，以便模型能快速收敛。标准化有多种方式。

1) min-max标准化 (对象名为MinMaxScaler)，该方法使数据落到[0,1]区间：

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2) Z-score标准化 (对象名为StandardScaler)，该方法使数据满足标准正态分布：

$$x' = \frac{x - \bar{X}}{S}$$

3) 归一化 (对象名为Normalizer，默认为L2归一化)：

$$x' = \frac{x}{\sqrt{\sum_j^m x_j^2}}$$

```
# min-max标准化
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()
sc.fit(X)
results = sc.transform(X)
print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [0.16666667 0.41666667 0.06779661 0.04166667]
```

```
# Z-score标准化
from sklearn.preprocessing import StandardScaler

#将fit和transform组合执行
results = StandardScaler().fit_transform(X)

print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [-1.14301691 -0.13197948 -1.34022653 -1.3154443 ]
```

```
# 归一化
from sklearn.preprocessing import Normalizer

results = Normalizer().fit_transform(X)

print("放缩前:", X[1])
print("放缩后:", results[1])
```

```
放缩前: [4.9 3.  1.4 0.2]
放缩后: [0.82813287 0.50702013 0.23660939 0.03380134]
```

5

- 使用阈值过滤器将数据转化为布尔值，即为二值化。使用Binarizer对象实现数据的二值化。

```
# 二值化, 阈值设置为3
from sklearn.preprocessing import Binarizer

results = Binarizer(threshold=3).fit_transform(X)

print("处理前: ", X[1])
print("处理后: ", results[1])
```

处理前: [4.9 3. 1.4 0.2]

处理后: [1. 0. 0. 0.]

- ❑ 标签编码：使用 LabelEncoder 将不连续的数值或文本变量转化为有序的数值型变量。

```
# 标签编码
from sklearn.preprocessing import LabelEncoder
LabelEncoder().fit_transform(['apple', 'pear', 'orange', 'banana'])

array([0, 3, 2, 1])
```

❑独热编码：对于无序的离散型特征，其数值大小并没有意义，需要对其进行one-hot编码，将其特征的m个可能值转化为m个二值化特征。可以利用OneHotEncoder对象实现。

[illegible]

5 结语

- 人工智能开源框架的崛起不仅推动了技术的快速发展，更在很大程度上促进了全球合作于创新。通过这些开源框架，我们能够更高效地构建智能应用，从而解决现实世界的复杂问题。无论是TensorFlow，PyTorch还是其他优秀的开源项目，他们的存在都让我们看到了技术发展的无限可能。
- 在这个瞬息万变的时代，开源精神为我们提供了无限的机会与可能。我们不仅能够从中学习最先进的技术，还能够通过贡献自己的力量，推动整个行业的进步。正如开源运动核心理念所言：分享与协作使我们更加强大。

总结

1 机器识别

2 Python入门

3 集成开发环境

4 主流深度学习框架

5 AI算法库和开发工具包

Q&A
谢谢各位



- 主 讲 人：陈建海
- 联系方式：13958011808
- <http://person.zju.edu.cn/cjh>