学习推荐

- 华为培训与认证官方网站
  - http://learning.huawei.com/cn/
- 华为在线学习
  - http://support.huawei.com/learning/elearning?lang=zh
- 华为职业认证
  - http://support.huawei.com/learning/NavigationAction!createNavi?navId=_31&lang=zh
- 查找培训入口
  - http://support.huawei.com/learning/NavigationAction!createNavi?navId=_trainingsearch&lang=zh

更多信息

- 华为培训APP

AI概览

www.huawei.com

HUAWEI

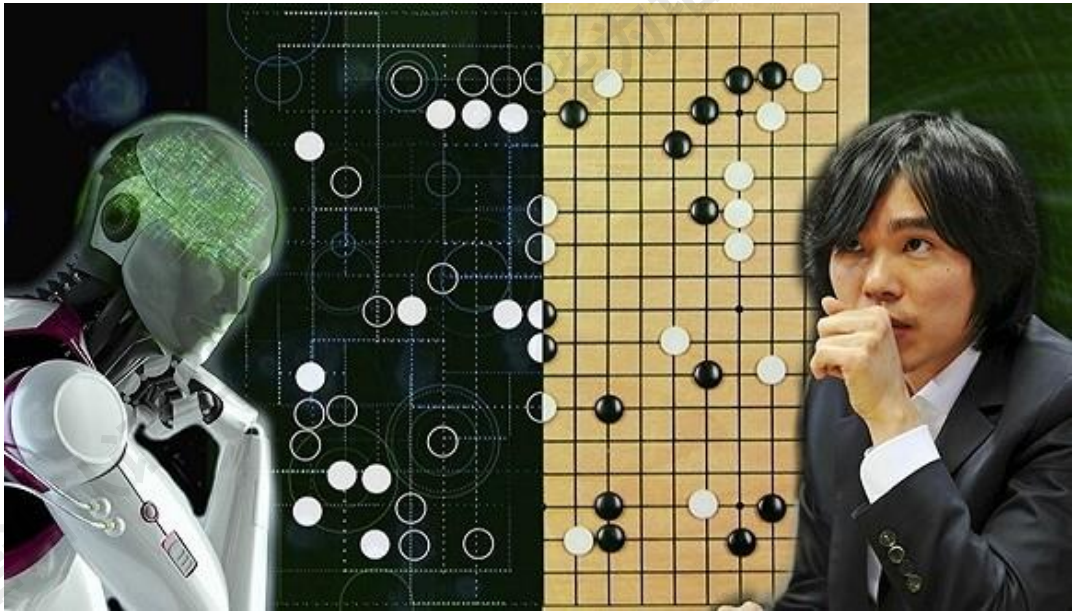
目标

- 学完本课程后，您将能够：
  - 了解人工智能的发展简史
  - 掌握人工智能包含的技术与相关概念
  - 了解人工智能时代的公平与正义
  - 了解人工智能时代的人机关系与治理

目录

人工智能的再度崛起

- 自2016年3月，AlphaGo以4:1的总比分大胜韩国职业围棋九段李世石，引爆了人工智能，进入了公众的视野，此后的发展就一发不可收拾。



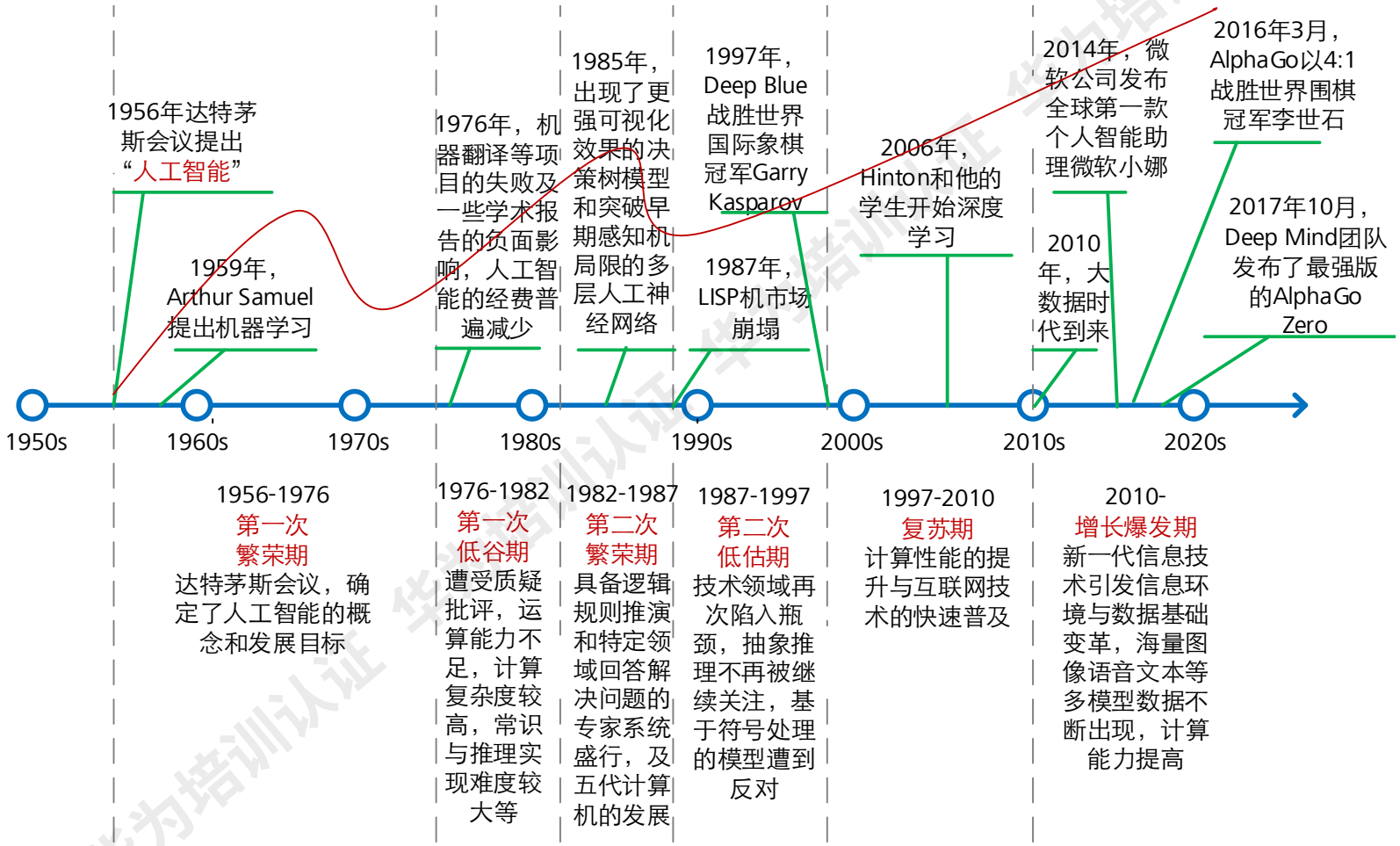
人工智能：正名于达特茅斯会议

- 1956年8月，在美国达特茅斯学院中，约翰·麦卡锡（John McCarthy，LISP语言创始人）、马文·闵斯基（Marvin Minsky，人工智能与认知学专家）、克劳德·香农（Claude Shannon，信息论的创始人）、艾伦·纽厄尔（Allen Newell，计算机科学家）、赫伯特·西蒙（Herbert Simon，诺贝尔经济学奖得主）等科学家聚在一起，讨论着一个完全不食人间烟火的主题：用机器来模仿人类学习以及其他方面的智能。
- 会议足足开了两个月的时间，虽然大家没有达成普遍的共识，但是却为会议讨论的内容起了一个名字：人工智能。因此，**1956年也就成为了人工智能元年。**

五十年后达特茅斯重逢

- 2006年，达特茅斯会议五十年后，当事人重聚。

人工智能发展简史

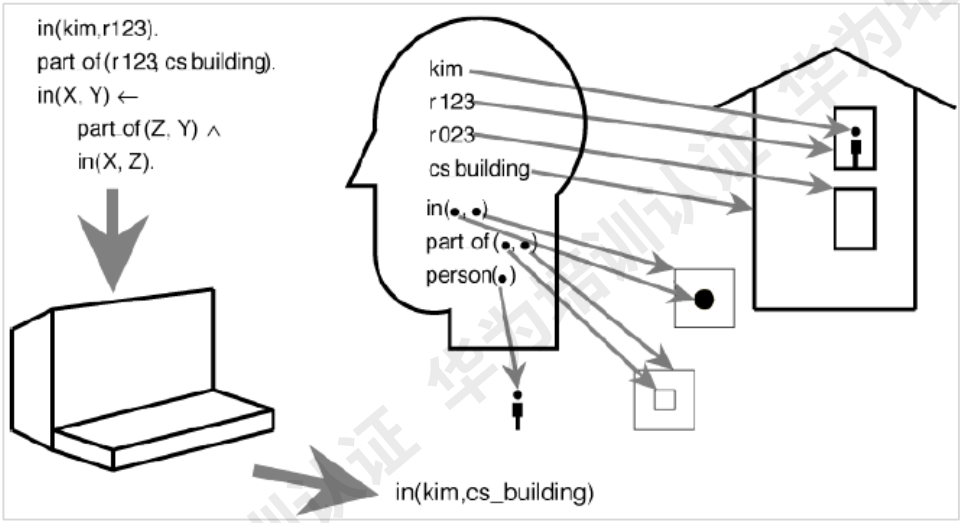


符号主义 (1)

- 符号主义（逻辑主义、心理学派、计算机学派）
  - 原理：物理符号系统假设和有限合理性原理。
  - 起源：源于数理逻辑。
  - 基本思想：
    - 认为人的认知基元是符号，认知过程即符号操作过程。
    - 认为人是一个物理符号系统，计算机也是一个物理符号系统，因此，能用计算机来模拟人的智能行为。
    - 认为知识是信息的一种形式，是构成智能的基础。人工智能的核心问题是知识表示、知识推理。
  - 学派代表：纽厄尔、西蒙、尼尔逊等。

符号主义 (2)

符号主义



代表人物

John McCarthy (1927-2011)　Allen Newell (1927-1992)　Herbert Simon (1916-2001)　Edward Feigenbaum (1936-)

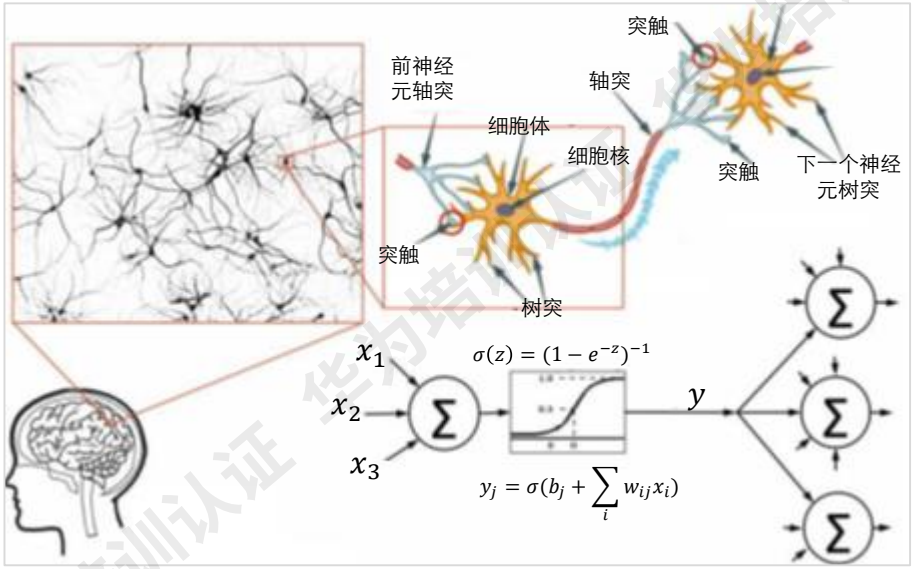
HUAWEI

连接主义 (1)

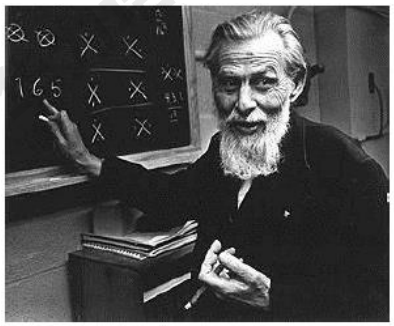
- 连接主义
  - 原理：神经网络及神经网络间的连接机制与学习算法。
  - 起源：源于仿生学，特别是人脑模型的研究。
  - 基本理论：
    - 认为思维基本是神经元，而不是符号处理过程。
    - 认为人脑不同于电脑，并提出连接主义的大脑工作模式，用于取代符号操作的电脑工作模式。
  - 学派代表：卡洛克、皮茨、Hopfield、鲁梅尔哈特等。

连接主义 (2)

连接主义

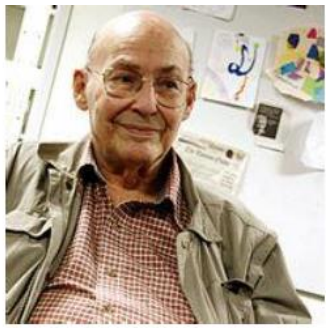


代表人物



Warren S. McCulloch
(1898-1969)

Walter H. Pitts
(1923-1969)



Marvin Minsky
(1927-2016)

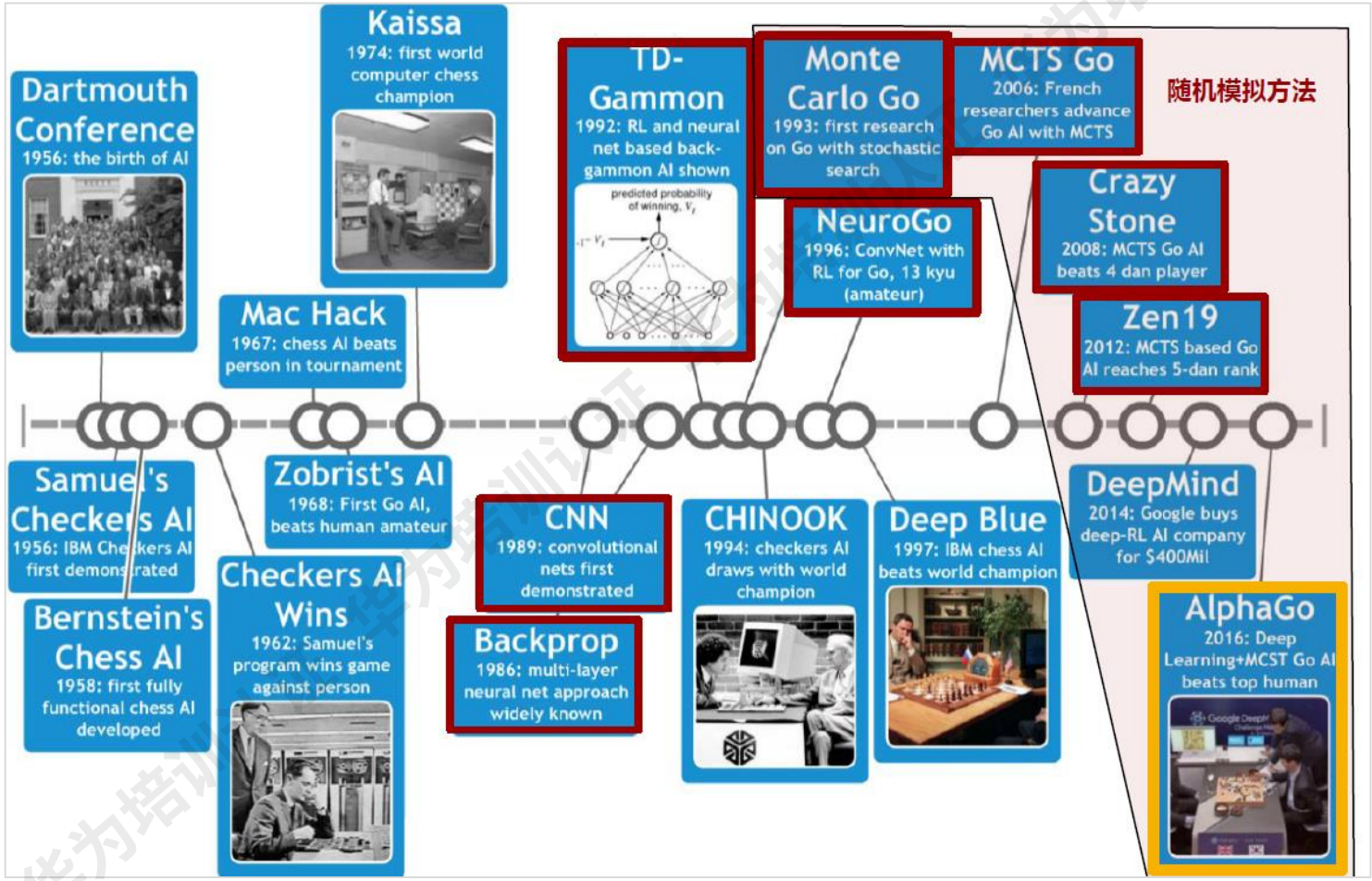
行为主义

- 行为主义（进化主义、控制论学派）
  - 原理：控制论及感知-动作型控制系统。
  - 起源：源于控制论。
  - 基本理论：
    - 认为智能取决于感知和行动，提出智能行为的“感知-动作”模式。
    - 认为智能不需要知识、不需要表示、不需要推理；人工智能可以像人类智能一样逐步进化；智能行为只能在现实世界中与周围环境交互作用而表现出来。

三大人工智能学派的优势和劣势分析

人工智能三大学派	知识表达	黑箱	特征学习	可解释性	是否需要大样本	计算复杂性	组合爆炸	环境互动	过拟合问题
符号主义（逻辑主义）	强	否	无	强	否	高	多	否	无
连接主义（仿生学派）	弱	是	有	弱	是	高	少	否	有
行为主义（决策控制）	强	否	无	强	否	一般	一般	是	无

人工智能的棋类游戏简史：三大主义走向融合

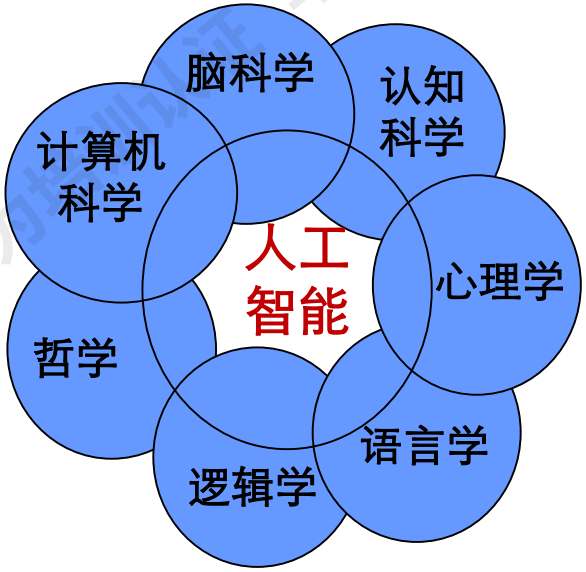


HUAWEI

目录

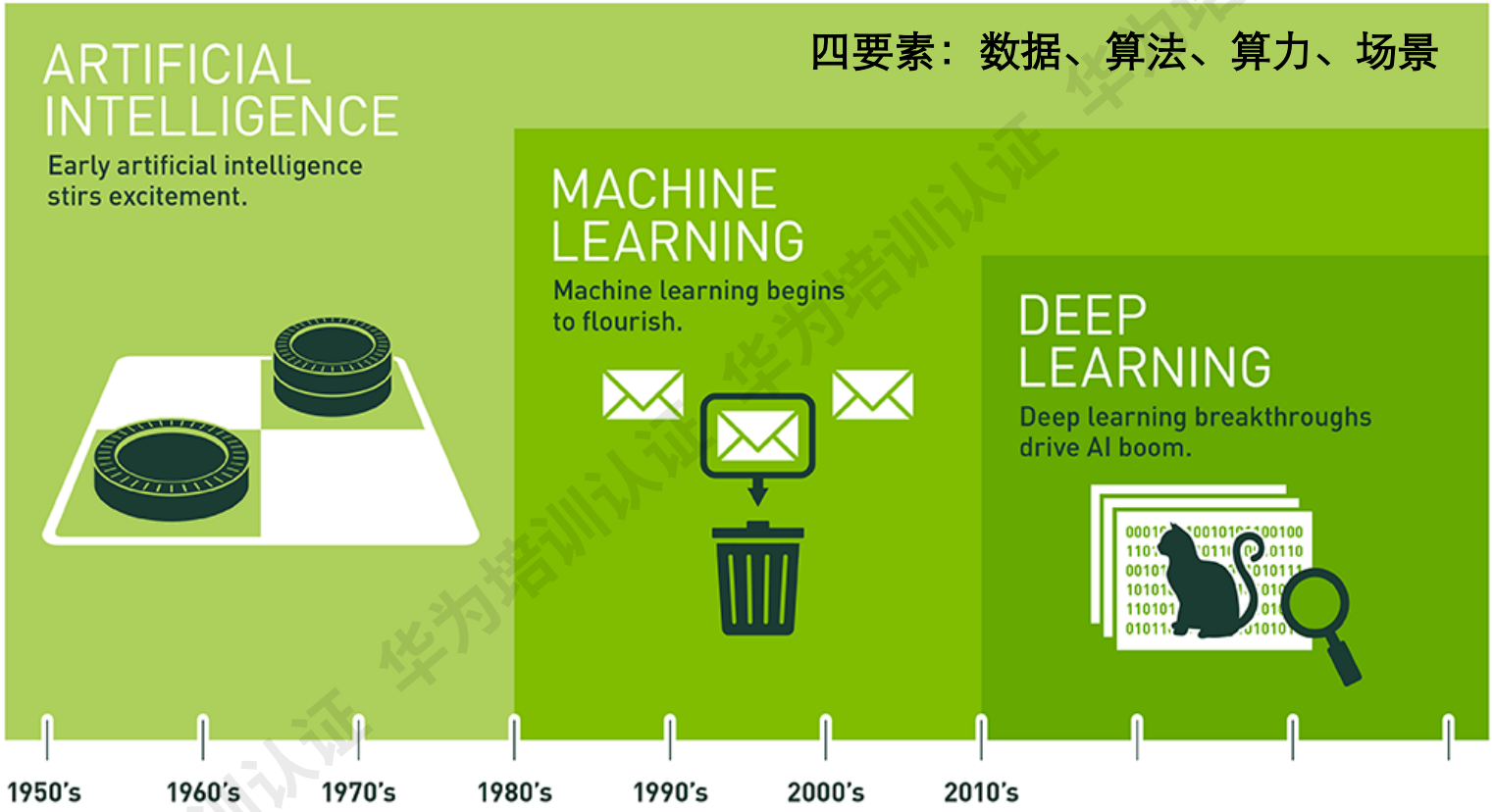
什么是人工智能

- **人工智能（Artificial Intelligence）：**它是研究、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。1956年由约翰.麦卡锡首次提出，当时的定义为“制造智能机器的科学与工程”。人工智能目的就是让机器能够像人一样思考，让机器拥有智能。时至今日，人工智能的内涵已经大大扩展，是一门交叉学科。



人工智能的层次结构

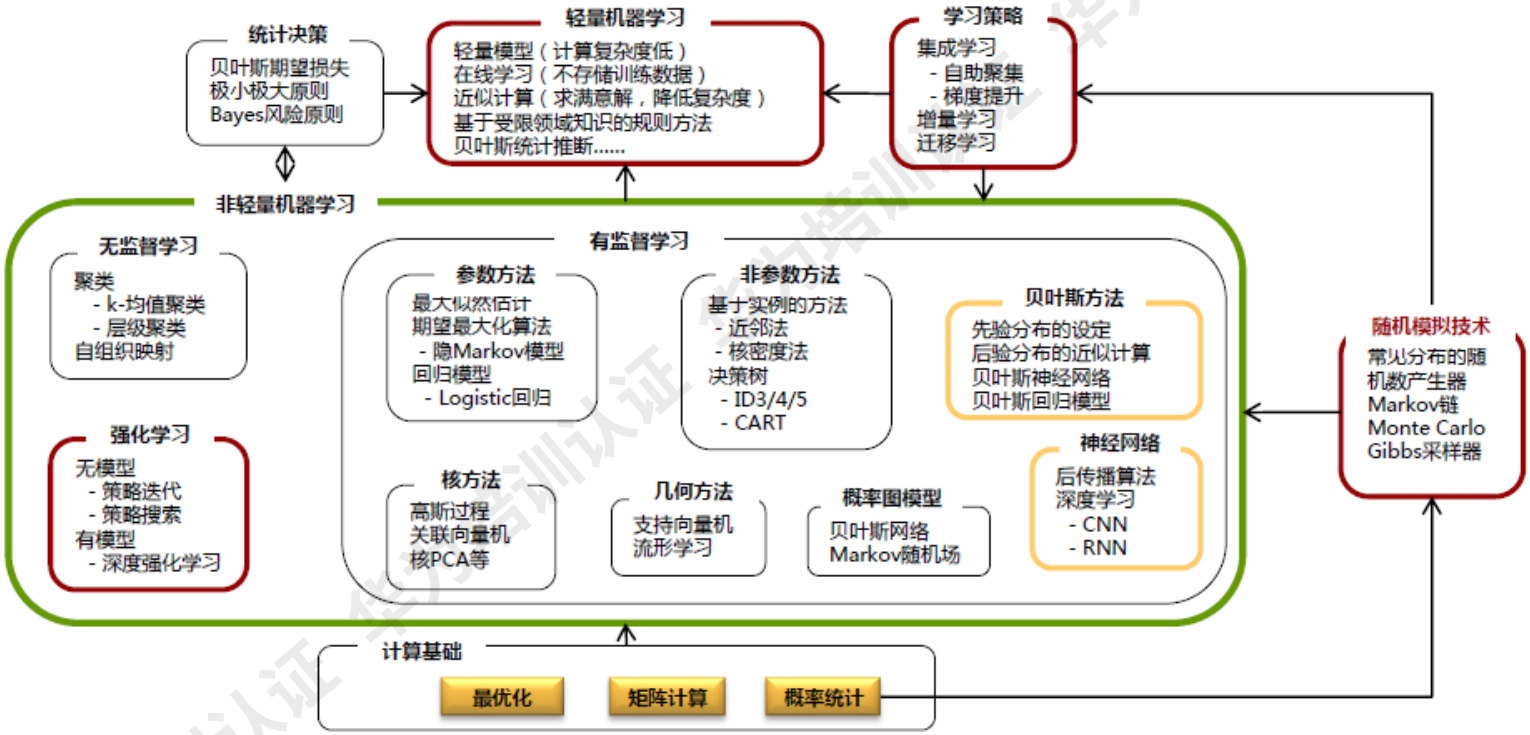
AI、机器学习、深度学习的关系



AI、机器学习、深度学习的关系

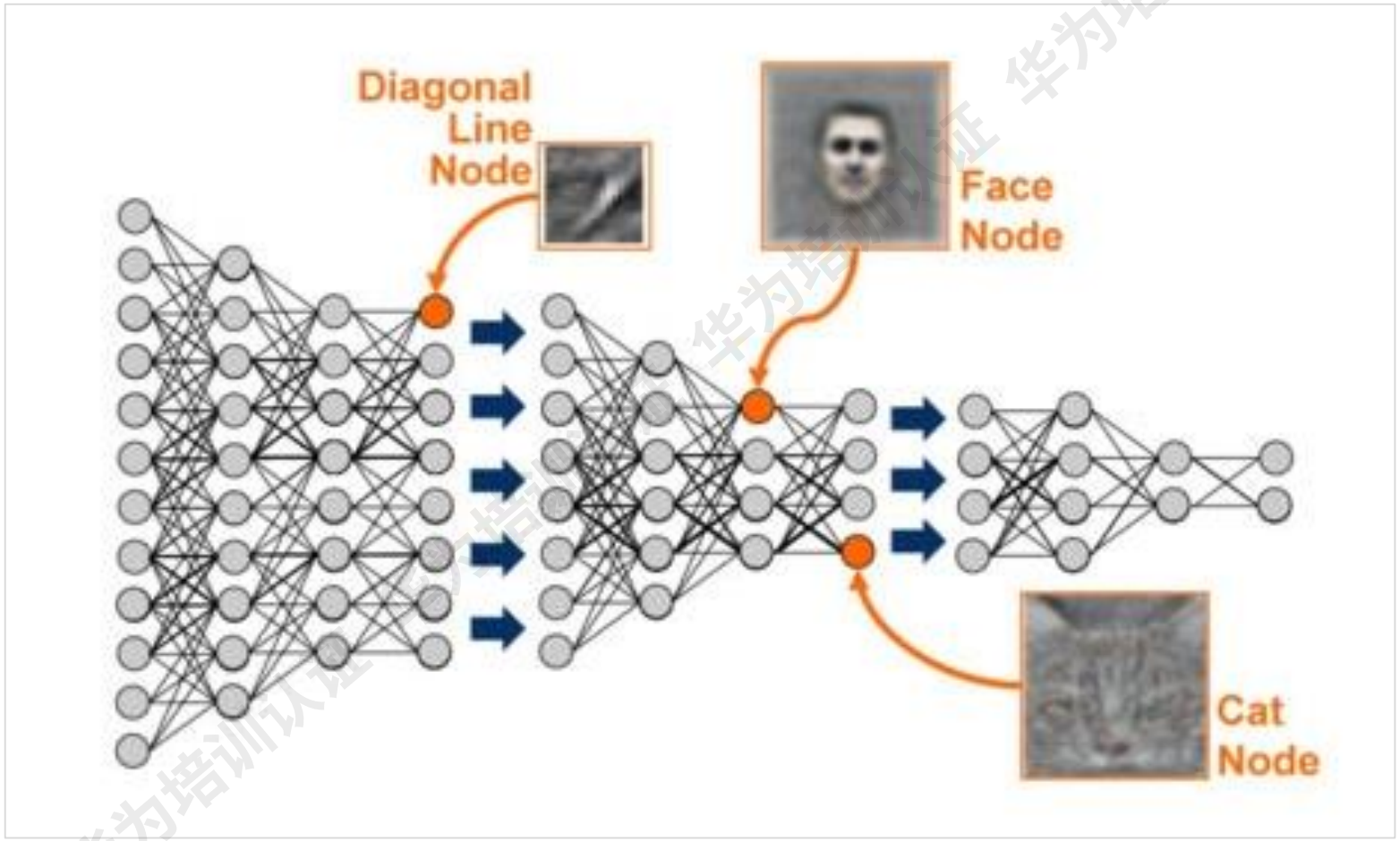
- 人工智能：是研究、开发用于模拟、延伸和扩展人的智能的理论、方法及应用系统的一门新的技术科学。
- 机器学习：专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。是人工智能的核心研究领域之一，任何一个没有学习能力的系统都很难被认为是一个真正的智能系统。
- 深度学习：源于人工神经网络的研究，含多隐层的多层感知器就是一种深度学习结构。深度学习是机器学习研究中的一个新的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，例如图像，声音和文本。

机器学习关键技术



HUAWEI

深度学习



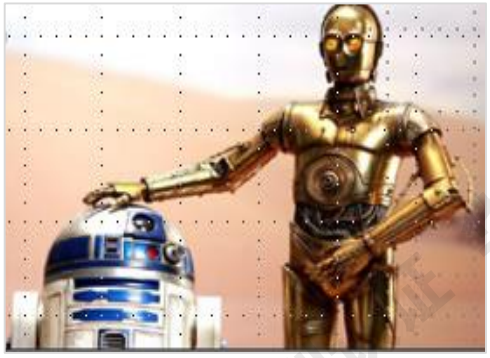
目录

AI应用场景

无人驾驶

智能家居

虚拟现实



智能机器人

智能投顾

智能医疗

HUAWEI

语音处理

- 语音处理主要是自动且准确的转录人类的语音。一个完整的语音处理系统，包括前端的信号处理、中间的语音语义识别和对话管理以及后期的语音合成。
  - 前端处理：说话人声检测，回声消除，唤醒词识别，麦克风阵列处理，语音增强等。
  - 语音识别：特征提取，模型自适应，声学模型，语言模型，动态解码等。
  - 语义识别和对话管理：更多属于自然语言处理的范畴。
  - 语音合成：文本分析、语言学分析、音长估算、发音参数估计等。
- 应用：包括医疗听写、语音书写、电脑系统声控、电话客服等。
- 未来：真正做到像正常人类一样，与他人流畅沟通，自由交流，还有待时日。

计算机视觉

- 计算机视觉指计算机从图像中识别出物体、场景和活动的能力，包含图像处理、识别检测、分析理解等技术。
  - 图像处理：去噪声、去模糊、超分辨率处理、滤镜处理等。
  - 图像识别：过程包括图像预处理、图像分割、特征提取、判断匹配，可以用来处理分类、定位、检测、分割问题等。
  - 图像理解：本质是图像与文本间的交互，可用来执行基于文本的图像搜索、图像描述生成、图像问答等。
- 应用：
  - 医疗成像分析被用来提高疾病的预测、诊断和治疗。
  - 在安防及监控领域被用来指认嫌疑人。
  - 在购物方面，消费者现在可以用智能手机拍摄下产品以获得更多信息。
- 未来：计算机视觉有望进入自主理解、分析决策的高级阶段，真正赋予机器“看”的能力，在无人车、智能家居等场景发挥更大的价值。

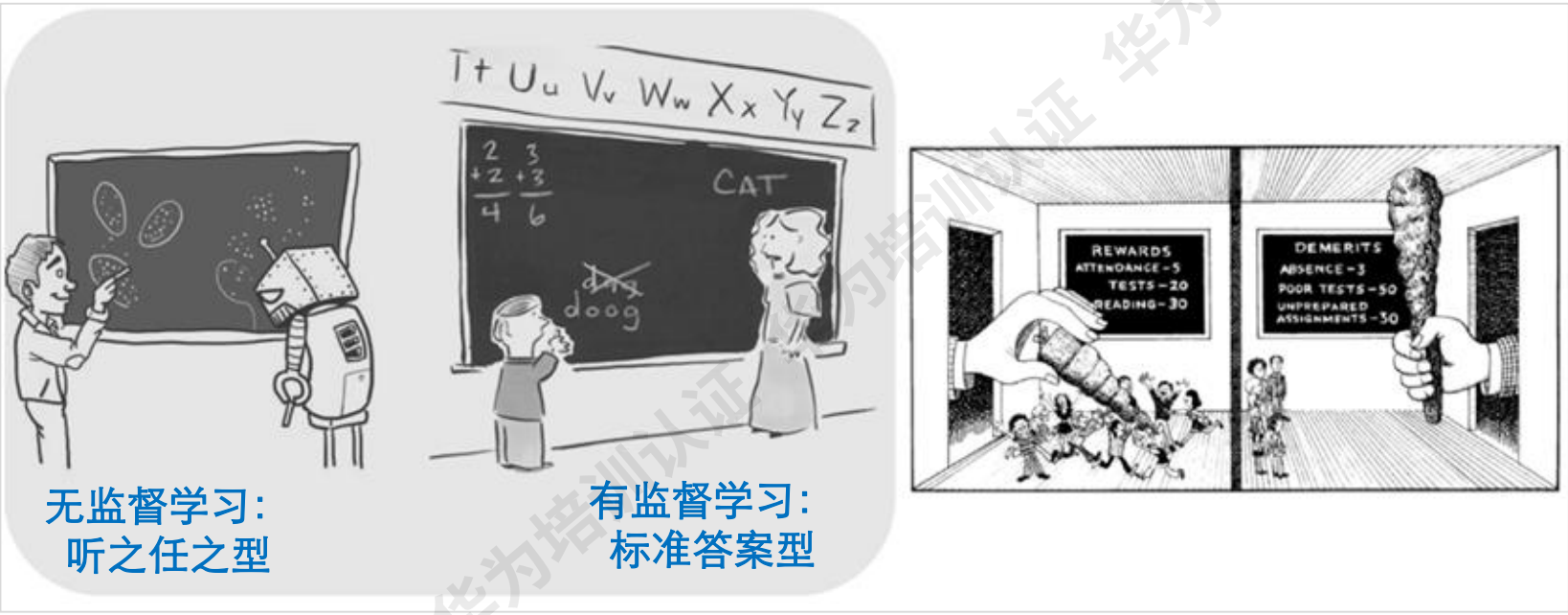
自然语言处理

- 自然语言处理的几个核心环节：知识的获取与表达、自然语言理解、自然语言生成等，也相应出现了知识图谱、对话管理、机器翻译等研究方向。
  - 知识图谱：基于语义层面对知识进行组织后得到的结构化结果。
  - 对话管理：包含闲聊、问答、任务驱动型对话。
  - 机器翻译：由传统的PBMT方法到Google的GNMT，流畅度与正确率大幅提升。
- 应用：搜索引擎、对话机器人、机器翻译、甚至高考机器人、办公智能秘书。

机器学习 (1)

- 机器学习：是研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。机器学习是AI的核心，是使计算机具有智能的根本途径。
- 研究方向：
  - 在垂直领域的广泛应用。如金融、律政、医疗等领域。
  - 从解决简单的凸优化问题到解决非凸优化问题。
  - 从监督学习向非监督学习和强化学习的演进。
- 未来：强化学习、迁移学习。

机器学习 (2)



无监督学习：
听之任之型

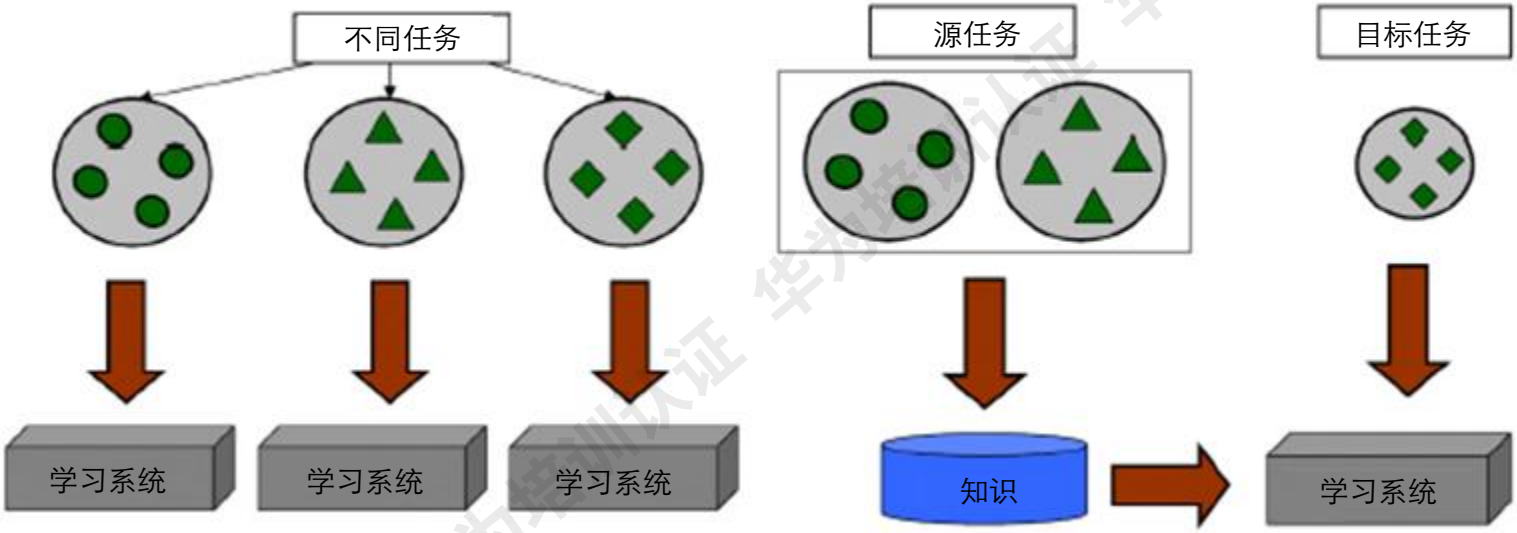
有监督学习：
标准答案型

传统机器学习

强化学习：奖励惩罚型

HUAWEI

机器学习 (3)



(a) 传统机器学习　　(b) 迁移学习

人工智能还处于初级阶段

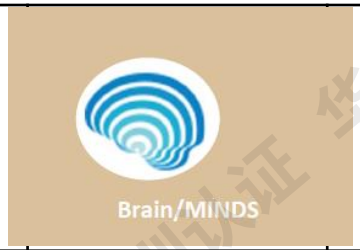
- 理性地看待人工智能的现状：
  - 人工智能还非常初级，目前适合“已知环境、目标明确、行动可预测”的场景。深度学习在图像识别、语音识别、翻译等领域，人工智能基本具备人的识别能力，甚至超越了人类，基于这些能力应用到了很多场景，如医疗、公共安全等。但在推理、认知等方面仍十分欠缺。
  - 人工智能不是要等到超越人的智慧才进入使用，而是只要在某个方面比人做得好就可以进入使用。

人工智能未来将走向融合

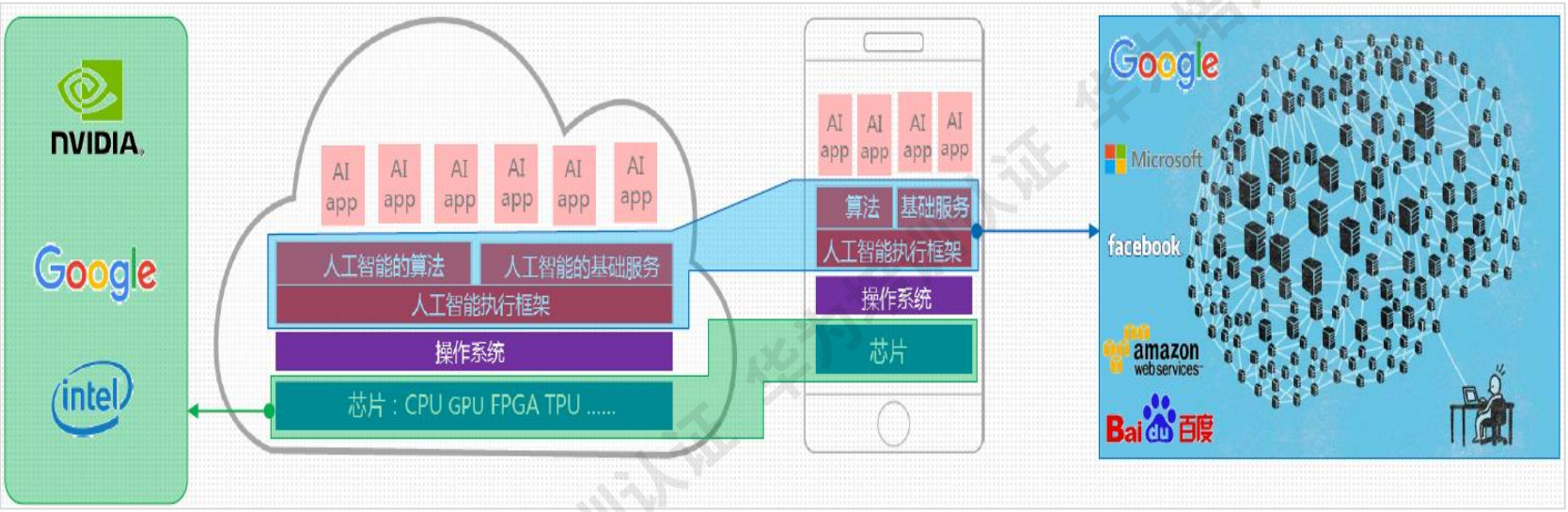
- 人工智能的发展趋势是走向融合：传统机器学习+深度学习+强化学习+知识推理+智能决策。未来五年，AI进入井喷期（Stanford教授等的观点）：
  - 从目前的supervised learning会逐步到Flexible learning。
  - 计算机视觉会在工业界得到普及，如教育、医疗、交通、公共安全等。
  - 机器人会在10年内产业化。

目录

全球类脑研究情况

			
美国	欧盟	日本	中国
▪ **Brain Initiative项目：** 以探索人类大脑工作机制为主（2013年启动，45亿美元） ▪ **SyNAPSE项目：** 开发大规模电子神经形态计算机原型（2008年~2016年）	▪ **Human Brain Project：** 侧重未来信息、通讯技术，未来医疗（2013年启动，10亿欧元）	▪ **Brain/Minds：** 以狨猴为模型研究各种脑功能和脑疾病的机理（2014年启动，2.7亿美元）	▪ **中国脑计划：** “一体两翼”的研究格局：脑认知原理的基础研究为一体，脑重大疾病和类脑人工智能的研究为两翼（国家层面待启动100亿RMB、地方层面已立项）
SyNAPSE项目： ▪ IBM主导的超低功耗新型计算系统的应用（TrueNorth芯片、系统架构设计、算法实现） ▪ HRL主导研发具有认知、学习、推理能力的计算系统，强调自主学习能力（基于忆阻器的类脑芯片研究）	**Human Brain Project：** ▪ 未来神经科学、未来医学、未来计算 ▪ 人脑战略性数据，认知行为架构、理论型神经科学、神经信息学、大脑模拟仿真、高性能计算平台、医学信息学、神经形态计算平台、神经机器人平台、模拟应用	**Brain/Minds：** ▪ 用功能MRI等技术对大脑功能进行定位 ▪ 收集和分析患者大脑成像等相关研究信息	**地方脑计划** ▪ 中科院成立类脑智能研究中心和脑神经计算研究组：从算法模型与信息处理到脑的模拟（寒武纪系列神经网络加速器） ▪ 清华大学成立类脑计算研究中心：系统设计、仿真建模、硬件材料的脑体系工程（Tianji类脑芯片）

人工智能正在改变产业格局



- 人工智能可能会引起“芯片架构的改变，进而引起产业格局的改变”，英伟达、Google、Intel等在争夺未来的主导权。
- 人工智能的软件技术平台就是“下一个操作系统”，技术平台将是大玩家的竞争，垂直行业的AI应用或服务可能都基于几大平台的云服务提供。
- 争夺“未来世界的数字大脑”成为各大信息公司的战略愿景，云计算、大数据、人工智能是三位一体的云服务。

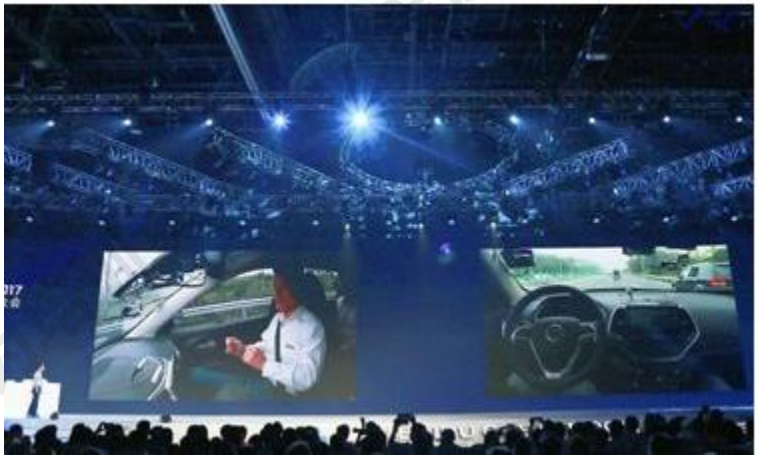
目录

AI怎样负责？

- 美国当地时间周日晚22点（2018年3月19日），亚利桑那州坦佩市郊区，一辆Uber自动驾驶测试车在由南向北行进过程中，撞上推着自行车在人行道之外横穿马路的49岁女子伊莱·赫兹柏格，该女子送医后抢救无效宣告死亡。这是全球首例无人驾驶汽车致死事故。

中国的无人驾驶

2017.12.2 4台阿尔法巴在深圳首发试运行



李彦宏乘坐百度研发无人驾驶汽车

- 梅花天使创投的创始合伙人吴世春表示：“无人驾驶必然是社会发展的大趋势，但必须有标准进行约束。Uber无人车撞死行人很让人震惊和心痛，不过据统计，全世界每年因道路交通事故死亡的人数多达100多万。因此，每一项新技术要走向社会一定得先通过不断循序渐进的实验和测试，以确保安全性。就像飞机要取得适航证才能投入商业运营，无人驾驶系统也应该有官方或者权威的第三方提供评测、颁发许可证后才能上路行驶，而不是由公司自行决定是否可以投入市场。包括机器人、人工智能等，所有的新科技都如是。”

自动驾驶领域的立法尝试

- 2013年美国道路交通安全管理局发布《自动驾驶汽车的基本政策》，对自动驾驶汽车测试事故的责任承担做了规定。
- 2016年8月，联合国科教文组织与世界科学知识与技术伦理委员会在《关于机器人伦理的初步草案报告》中对机器人的责任进行了探讨，提出：采取责任分担的解决路径，让所有参与到机器人的发明、授权和分配过程中的人来分担责任。
- 2017年5月12日，德国通过由运输部提议的《修订案》，在驾驶时系统不可完全取代驾驶人，驾驶人应留在汽车驾驶位上，并能够随时接管车辆的控制；尽管（自动化驾驶）有电脑的补充，但最终责任原则上应主要落在驾驶人身上。
- 目前的法律框架下，机器人自身不对因其行为或者疏忽而给第三方造成的损害承担责任。

隐私怎样保护？

- 德州大学的两名研究人员，成功将Netflix放出的“经过匿名处理的”上亿条用户电影评分数据与具体用户对应了起来，Netflix被迫取消影片推荐算法比赛。

立法保护与技术应用

- 立法保护：
  - 自1973年第一部个人数据保护法《瑞典数据法》，至2016年12月，全球已有110多个国家和地区制定了专门的个人信息保护法。
  - 2012年中国全国人大常委会通过《关于加强网络信息保护的规定》。2016年，通过《网络安全法》。
- 技术应用：
  - 匿名化处理技术：将个人数据移除可识别个人信息的部分，并且通过这一方法，数据主体不会再被识别。

算法默认是否公平？

- 算法决策在很多时候其实就是一种预测，用过去的数据预测未来的趋势。算法模型和数据输入决定着预测的结果。因此这两个要素也就成为了算法歧视的主要来源。比如：
  - 谷歌的图片软件曾错将黑人的图片标记为“大猩猩”。
  - 2016年3月23日，微软的智能聊天机器人Tay一上线就被“教坏”了，成为了一个集反犹太人、性别歧视、种族歧视于一身的“不良少女”，于是，上线不到一天，Tay被紧急下线。

很多未解决的问题

- 人工智能创作物受版权法保护吗?
- 谁来赋权于机器人?
- 赋予机器人哪些权利?
- …

目录

三代机器人

- 第一代机器人：示教再现型机器人。第一代机器人可以重复的根据人当时示教的结果，再现出这种动作，该类机器人的特点是它对外界的环境没有感知。
- 第二代机器人：带感觉的机器人。这种带感觉的机器人是类似人在某种功能的感觉。比如：力觉、触觉、听觉，来判断力的大小和滑动的情况。
- 第三代机器人：智能机器人。理想中所追求的最高阶段，智能机器人，只要告诉他做什么，它就能完成，目前的发展还是相对的只是在局部的概念和含义。

智能机器人的分类

- 人工智能研究在国际上至今尚无统一的定义，目前普遍将智能机器分为四类：
  - “像人一样思考”：弱人工智能领域，如Watson、AlphaGo。
  - “像人一样行动”：弱人工智能领域，如人形机器人、iRobot、波士顿动力公司的Atlas。
  - “理性地思考”：强人工智能，尚无法达到，瓶颈在脑科学。
  - “理性的行动”：强人工智能。

机器人三定律

- 1942年美国科幻巨匠阿西莫夫提出“机器人三定律”
  - 定律1：机器人不得伤害人类，或者目睹人类将遭受危险而袖手不管。
  - 定律2：机器人必须服从人给予它的命令，除非违背第一定律。
  - 定律3：机器人必须保护自己，除非违背第一、第二定律。

人机关系的设想与AI治理

- **人机关系的设想：**
  - 担忧机器威胁人类，通过控制AI实现人机共存。
  - AI成为人类意识的代理者，人类通过AI延伸自我。
  - 未来“虚拟的真实存在”或将成真。
- **AI治理：**
  - 治理应当建立在技术与产业革新的基础之上。
  - 适度性监管，保持权利的谦逊。
  - 不要陷入泛安全化误区。
  - 以促进发展和创新为目的。
  - 鼓励多元主体参与的多层次治理模式。

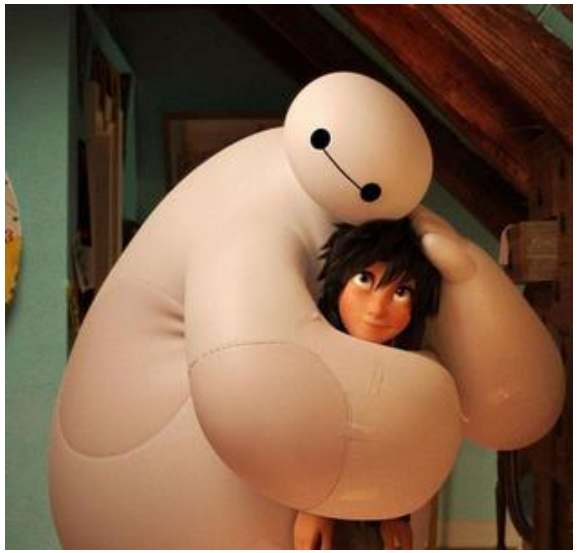
目录

机器人同事

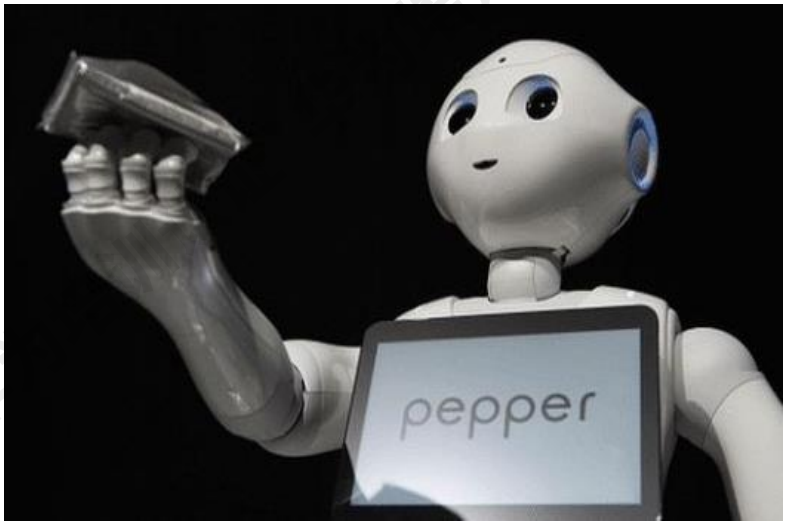
ASIMO

服务员

灵魂伴侣



大白



日本情感机器人Pepper

Siri

HUAWEI

人工智能的机会和挑战，创造新市场和改变“价值分配”……

机会：从“效率”到“智慧”，人工智能将创造比今天IT市场更大的市场（2万亿美元）这是整个信息产业正在争夺的新机会。

挑战：在整个产业链中，掌握智能的人，将具备更大的话语权，从价值链中获得更多价值；这就是GE等传统企业纷纷成立“数字部门”背后原因。

吓尿指数

HUAWEI

本章总结

- 本章首先介绍了人工智能的过去、现在、未来，描述了人工智能所包含的技术与发展全貌，进而阐述了人工智能时代的思考与问题，如公平正义、人机关系、AI治理等。

思考题

1. AI的英文缩写是？（　）

   A. Automatic Intelligence

   B. Artificial Intelligence

   C. Automatic Information

   D. Artificial Information

2. 神经网络研究属于下列哪个学派？（　）

   A. 符号主义

   B. 连接主义

   C. 行为主义

   D. 都不是

思考题

3. 1997年5月，著名的“人机大战”最终计算机以3.5比2.5的总比分将世界国际象棋棋王卡斯帕罗夫击败，这台计算机被称为？（　）

A .深蓝

B. 深绿

C. 深思

D. 蓝天

4. 人工智能的含义最早由一位科学家于1950年提出，并且同时提出一个机器智能的测试模型，请问这个科学家是？（　）

A .明斯基

B. 扎德

C. 图灵

D. 冯.诺依曼

学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

Python编程基础

www.huawei.com

HUAWEI

前言

- Python 是一门简单易学且功能强大的编程语言。
- Python 拥有高效的高级数据结构，并且能够简单且快速地进行面向对象编程。
- Python 优雅的语法和动态类型，再结合它的解释性，使其在大多数平台上成为编写脚本或开发应用程序的理想语言。

目标

- 学完本课程后，您将能够：
  - 了解Python编译环境与安装过程。
  - 掌握Python数据结构、数据类型、条件语句、循环语句、函数、模块等等。
  - 能够简单地应用到实际场景中。

目录

Python的历史

- Python是自由软件的丰硕成果之一。
- Python是纯粹的自由软件， 源代码和解释器都遵循 GPL(GNU General Public License)协议。

创始人	Guido van Rossum
时间地点	1989年圣诞期间在阿姆斯特丹创造。
名字来源	大蟒蛇飞行马戏团的爱好者。
渊源	从ABC发展而来主要受Modula-3的影响，结合了Unix shell和C的习惯。

Python的起源

- Guido van Rossum：
  - 数学硕士
  - 计算机硕士
- Python哲学：
  - Python是工程，不是艺术。
  - 解决一种问题只有一个办法。
  - 简单优于繁复，明确优于晦涩。

什么是Python

- Python 是一种编程语言。
- Python 是一种通用的高级编程语言。
- Python 能用于多种领域的程序开发：
  - 数据科学
  - 编写系统工具
  - 开发图形界面的应用
  - 写基于网络的软件
  - 与数据库交互

Python与其它语言的区别 (1)

- Python & C：
  - python是动态编译语言，C是静态编译语言。
  - C中内存管理是由开发者管理，Python中内存问题由Python解释器负责。
  - Python有很多第三方库。C语言中对于混杂数组（Python中的列表）和哈希表（python中得字典）还没有相应的标准库。
  - Python不能用来写内核。C可以。
  - 借助Python语言提供的API，使用C或者C++来对Python进行功能性扩展。
- Python & SHELL：
  - Python语法简单，可移植性好。
  - Shell代码写出来的脚本较长。
  - Python可以重用代码，提倡简洁的代码设计，高级的数据库结构和模块化组件。

Python与其它语言的区别 (2)

- Python & Java：
  - Python是动态编译语言，Java是静态编辑语言。
  - Python支持面向对象和函数编程方式。Java支持面向对象。
  - Python比Java要简单，非常适合构造快速原型。
  - Python和Java适合多名程序员以渐进方式协同开发大型项目。

Python的开发环境

- VIM - 主要在linux下使用。
- IDLE - 集成开发环境。
- Sublime Text - 轻量级的编辑工具。
- Eclipse - 收费的。
- Eric4 - 基于PyQT4，功能强大。
- Boa - 类似于delphi的IDE(wxPython)。
- WingIDE - 共享软件 。
- 其它编辑器： notepad++，editplus ……

Python的优点 (1)

- 简单 - Python是一种代表简单主义思想的语言。
- 易学 - 关键字少,结构简单,语法清晰。
- 开源 - Python是FLOSS（自由/开放源码软件）之一。
- 可解释性 - Python语言写的程序不需要编译成二进制代码，可以直接从源代码运行程序。
- 可移植性 - 由于它的开源本质，Python已经被移植在许多平台上。
  - 这些平台包括:Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE甚至还有Symbian和Google基于linux开发的android平台。
- 可扩展性 - 如果需要一段关键代码运行得更快或者希望某些算法不公开，可以部分程序用C或C++编写，然后在Python程序中使用它们。

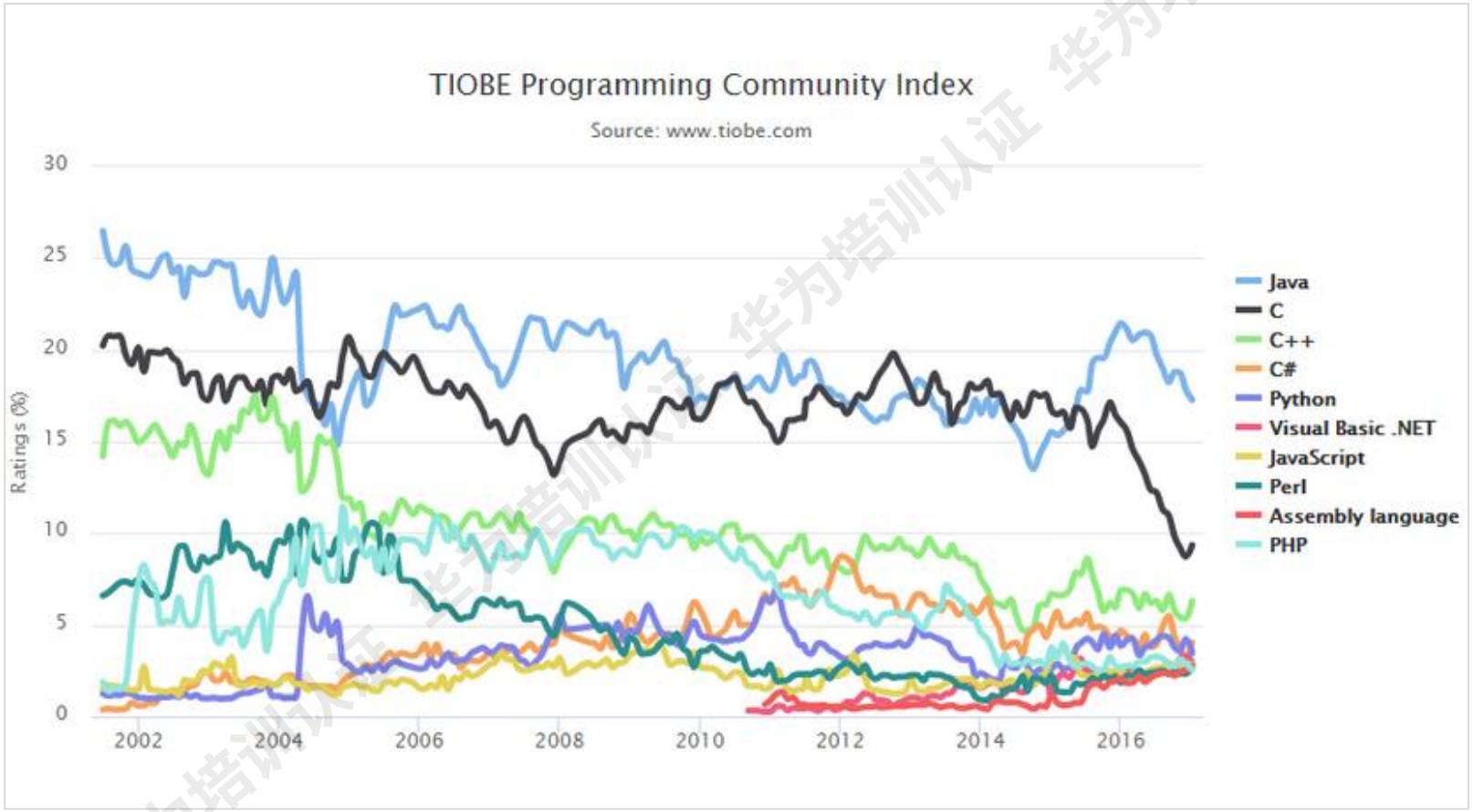
Python的优点 (2)

- 高级语言 - 当你用Python语言编写程序的时候，你无需考虑诸如如何管理你的程序使用的内存一类的底层细节。
- 可嵌入性 - 可以把Python嵌入C/C++程序，从而向程序用户提供脚本功能。
- 面向对象 - Python既支持面向过程的编程也支持面向对象的编程。在“面向过程”的语言中，程序是由过程或仅仅是可重用代码的函数构建起来的。在“面向对象”的语言中，程序由数据和功能组合而成的对象构建而来。
- 丰富的库 - Python标准库确实很庞大。它可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、GUI（图形用户界面）、Tk和其他与系统有关的操作。

Python的语法特点

- 动态语言特性，可在运行时改变对象本身(属性和方法等)。
- Python使用缩进，而不是一对花括号{}来划分语句块。
- 多个语句在一行使用“;”分隔。
- 注释符是#，注释多行使用doc string(‘‘‘...’’’)。
- 变量无需类型定义。
- 可进行函数式编程（FP）。

流行度排名



HUAWEI

Python2与python3的区别（1）

- Python 3不能向后兼容 Python 2，这就需要人们决定该使用哪个版本的语言。
- 许多封装库只适用于 Python 2，但是由于 Python 3 背后的开发团队重申了终止对 Python 2 的支持，促使更多的库被移植到 Python 3 上来。
- 从对 Python 3 提供支持的Python包的数量来看出，Python 3 已得到越来越多的采用。

Python2与python3的区别 (2)

- print函数
- Unicode
- 除法运算
- 异常
- Xrange
- 数据类型
- 不等运算符

Python的安装 (1)

- Linux用户：
  - 下载Python包，并安装。

```
$tar -zxf python3.6.4.tar.gz
$cd Python3.6.4
$./configure
$make && make install
```

  - 建立软连接。

```
$mv /usr/bin/python  /usr/bin/python.bak
$ln -s /usr/local/bin/python3.6.4
/usr/bin/python
```

  - 检查。

```
 $python -V
```

Python的安装 (2)

- Windows用户：
  - 下载 Python官方发布的 Python 安装程序。
  - 选择最新的 Python Windows 安装程序，下载 .exe 安装文件。
  - 双击安装程序 Python-3.x.exe。
  - 增加环境变量：右键“我的电脑”>“属性”>“高级”>“环境变量”，在path里输入你的python安装位置。
  - 测试是否安装成功：开始>程序>python 3.x>启动 Python command line，然后输入：print("Hello World“)，如果输出"Hello World"，那就表明安装成功了。

Python的启动

- Linux启动python：

```
[root@yaokaiqiangzjhw ~]# python3
Python 3.6.4 (default, Apr  9 2018, 13:51:42)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

- Windows启动python：

```
C:\Users\ywx516714>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

Python的程序执行

- 命令行模式：
  - Linux：
    - 在linux命令行输入Python命令。
  - Window ：
    - 在dos提示符下输入Python命令。
- 脚本模式：
  - 将Python语句存入脚本文件，在命令行中执行它。

```
输入：python hello.py
输出：hello world !
```

Vim

- Vim是一款轻量级的IDE。
- 如果不安装过多插件或者插件性能没有问题的话，使用Vim开发对硬件的要求是比较低的。
- Vim能在本地和远程服务器上获得一致的编程体验。
- Vim拥有“所思即所得的编辑速度”。

目录

列表 (list)

- 元素英文list。
- list是一种有序的集合，可以随时添加和删除其中的元素。
- list里面的元素类型可以不同，可以通过索引来访问list中的每一个元素，正向索引第一位从0开始，反向索引第一位是从-1开始。

列表的常用操作

- 访问
- 更新（append，insert）
- 删除元素（del）
- 列表脚本操作符（+/*）
- 列表截取

Python列表函数

- cmp(list1, list2),
- len(list)
- max(list)
- min(list)
- list(seq)

Python列表方法

- list.append(obj)
- list.count(obj)
- list.extend(seq)
- list.index(obj)
- list.insert(index, obj)
- list.pop(obj=list[-1])
- list.remove(obj)
- list.sort([func])
- list.reverse()

元组 (tuple)

- 元组英文：tuple，用()定义。
- 与list类似，tuple一旦初始化就不能修改，tuple在定义时，元素就必须确定下来。
- tuple没有append()、insert()方法，也不能赋值成另外的元素，其他获取方法和list是一样的。
- 因为tuple不可变，所以代码更安全，如有可能尽量使用tuple代替list。
- tuple创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。

元组的常用操作

- 访问
- 修改（元组运算）
- 删除（del元组）
- 元组运算符（+，*）
- 元组索引和截取
- 无关闭分隔符

元组内置函数

- cmp(tuple1, tuple2)
- len(tuple)
- max(tuple)
- min(tuple)
- tuple(seq)

目录

字符串定义

- Python中的字符串是零个或多个的字符所组成的序列，字符串是Python内建的几种序列之一；
- 在Python中字符串是不可变的，即我们在C、C++语言中常说的字符串常量。
- Python的字符串表示有单引号，双引号和三引号，还有转义字符、原始字符串等。
  - name='JohnSmith‘
  - name="Alice“
  - name="""Bob"""

字符串格式化 (1)

- Python 支持格式化字符串的输出 。尽管这样可能会用到非常复杂的表达式，但最基本的用法是将一个值插入到一个有字符串格式符 %s 的字符串中。
- Python中的字符串格式化是通过字符串格式化操作符（百分号%）来实现的，其字符串转换类型表及其格式化操作符辅助指令如后续表格所示。

输入： print(“My name is %s and age is %d !” %('AI', 63))

输出： My name is AI and age is 63 !

字符串格式化 (2)

- 字符串格式转换类型：

格式	格式符描述
%c	字符及其ASCII码。
%s	字符串。
%d	有符号整数(十进制)。
%u	无符号整数(十进制)。
%o	无符号整数(八进制)。
%x	无符号整数(十六进制)。
%X	无符号整数(十六进制大写字符)。
%e	浮点数字(科学计数法)。
%E	浮点数字(科学计数法，用E代替e)。
%f	浮点数字(用小数点符号)。
%g	浮点数字(根据值的大小采用%e或%f)。
%G	浮点数字(类似于%g)。
%p	指针(用十六进制打印值的内存地址)。

字符串格式化 (3)

- 格式化操作符辅助指令：

符号	作用
*	定义宽度或者小数点精度。
-	用作左对齐。
+	在正数前面显示加号（+）。
<sp>	在正数前面显示空格。
#	在八进制数前面显示零（0），在十六进制前面显示0x或者0X（取决于用的是x还是X）
0	显示的数字前面填充0而不是默认的空格。
%	%%输出一个单一的%。
(var)	映射变量（字典参数）。
m.n	m是显示的最小总宽度，n是小数点后的位数。

字符串操作符

- Python中没有专门的Char类型，一个字符就是长度为1的字符串，同时Python的字符串是不可改变的，并且Python字符串后是没有’\0’结尾的。Python中字符串是一个字符的序列，在内存的存放如下表所示：

	P	y	t	h	o	n
下标	0	1	2	3	4	5
下标	-6	-5	-4	-3	-2	-1

- 在Python中字符串下标从0开始，-1表示序列中最后一个值的下标，1表示第二个字符的下标，-2表示倒数第二字符的下标，以此类推。

字符串方法

- 提到函数之前，我们要先介绍一下Python最有用的内置函数dir。如果你想知道Python的字符串到底有哪些属性和方法，可以使用dir函数查看。

```
>>> dir('')
['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
'__eq__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
'__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__',
'__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__str__', 'capitalize', 'center', 'count', 'decode', 'encode',
'endswith', 'expandtabs','find', 'index', 'isalnum', 'isalpha',
'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust',
'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith',
'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

string模块

- Python中的字符串除了前面提到的一些属性、方法外，还有一个string模块。string模块到底有哪些属性和方法，可以使用下面的dir内建函数来查看：

```
>>> import string
>>> dir(string)
['Template', '_TemplateMetaclass', '__builtins__', '__doc__', '__file__',
'__name__', '_float', '_idmap', '_idmapL', '_int', '_long', '_multimap',
'_re', 'ascii_letters', 'ascii_lowercase', 'ascii_uppercase', 'atof',
'atof_error', 'atoi', 'atoi_error', 'atol', 'atol_error', 'capitalize',
'capwords', 'center', 'count', 'digits', 'expandtabs', 'find',
'hexdigits', 'index', 'index_error', 'join', 'joinfields', 'letters',
'ljust', 'lower', 'lowercase', 'lstrip', 'maketrans', 'octdigits',
'printable', 'punctuation',
replace', 'rfind', 'rindex', 'rjust', 'rsplit', 'rstrip', 'split', 'spli
tfields', 'strip', 'swapcase', 'translate', 'upper', 'uppercase', 'white
space', 'zfill']
>>>
```

目录

字典

- 字典又名dictionary。
- 字典是另一种可变容器模型，且可存储任意类型对象。
- 字典的每个键值 key=>value 对用冒号“:”分割，每个键值对之间用逗号“,”分割，整个字典包括在花括号“{}”中 。
- 键一般是唯一的，且键的类型不可变，如果键重复则最后的一个键值对会替换前面的，值不需要唯一，可以取任何数据类型。
- 格式如下：
  - d = {key1 : value1, key2 : value2 }

Python字典操作

- 访问
- 修改
- 删除

字典内建函数

函数名	函数说明
cmp(dict1,dict2)	比较两个字典元素。
len(dict)	计算字典元素个数，即键的总数。
str(dict)	输出字典可打印的字符串表示。
type(variable)	返回输入的变量类型，如果变量是字典就返回字典类型。

字典内建方法

- 内建方法：

方法名	方法说明
has_key(x)	如果字典中有键x，则返回真。
keys()	返回字典中键的列表。
values()	返回字典中值的列表。
items()	返回tuples的列表。每个tuple由字典的键和相应值组成。
clear()	删除字典的所有条目。
copy()	返回字典高层结构的一个拷贝，但不复制嵌入结构而只复制对那些结构的引用。
update(x)	用字典x中的键值对更新字典内容。
get(x[,y])	返回键x，若未找到该键返回none，若提供y，则未找到x时返回y。
pop()	删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。
popitem()	随机返回并删除字典中的一对键和值。

目录

if语句

- Python 支持三种不同的控制结构：if，for和while，不支持C语言中的switch语句。
- Python 编程中 if 语句用于控制程序的执行，基本形式为：

```
if 判断条件1:
    执行语句1……
elif 判断条件2:
    执行语句2……
elif 判断条件3:
    执行语句3……
else:
    执行语句4……
```

While语句

- Python 语言中 while 语句用于执行循环程序，在某条件下，循环执行某段程序，以处理需要重复处理的相同任务。
- 当while语句的条件永远不会为布尔假时，循环将永远不会结束，形成无限循环，也称死循环。可以在循环体内使用break语句强制结束死循环。
- While语句的用法：

```
count = 0

while (count < 9):
    print("The count is:", count)
    count = count + 1

print("Good bye!")
```

for语句

- Python 语言for循环可以遍历任何序列的项目，如一个列表，一个字典或者一个字符串。
- for语句与传统的for语句不太一样，它接受可迭代对象（例如序列或迭代器）作为其参数，每次迭代其中的一个元素。

```
for num in nums:
    if num == 1:
        print(num+”---”)
    elif num == 2:
        print(num+”///”)
    else:
        print('break not triggered')
```

循环嵌套

- Python 语言允许在一个循环体里面嵌入另一个循环体。
- Python for 循环嵌套语法：

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

- Python while 循环嵌套语法：

```
while expression:
    while expression:
        statement(s)
    statement(s)
```

Break与continue

- break 结束整个循环，如果触发了break，则循环的else不会执行。
- continue 结束循环的当次迭代，开始下一轮迭代。
- 如果您使用嵌套循环，break语句将停止执行最深层的循环，并开始执行下一行代码。
- continue 语句用来告诉Python跳过当前循环的剩余语句，然后继续进行下一轮循环。
- break 和 continue 语句都可用在while和for循环中。

目录

Python函数

- 函数是组织好的，可重复使用的，用来实现单一或相关联功能的代码段。
- 函数能提高应用的模块性，和代码的重复利用率。
- Python提供了许多内建函数，比如print()。也可以自己创建函数，这被叫做用户自定义函数。

定义一个函数

- 自定义一个函数，规则如下：
  - 函数代码块以 def 关键词开头，后接函数名称和圆括号()。
  - 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
  - 函数的第一行语句可以选择性地使用文档字符串，用于存放函数说明。
  - 函数内容以冒号起始，并且缩进。
  - Return[表达式]结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 None。

函数的调用

- 定义一个函数只给了函数一个名称，指定了函数里包含的参数，和代码块结构。
- 这个函数的基本结构完成以后，你可以通过另一个函数调用执行，也可以直接从Python提示符执行。

```
# 定义函数
def test(str):
    "打印任何传入的字符串"
    return str;

# 调用函数
test("我要调用用户自定义函数!");
test("再次调用同一函数");
```

参数传递

- 在 python 中，类型属于对象，变量是没有类型的。

```
a = [1,2,3]
a = "Huawei"
```

- 以上代码中，[1,2,3] 是 List 类型，“Huawei” 是 String 类型，而变量 a 是没有类型，它仅仅是一个对象的引用（一个指针），可以是 List 类型对象，也可以指向 String 类型对象。

参数类型

- 以下是调用python函数时可使用的正式参数类型：
  - 必备参数：必备参数须以正确的顺序传入函数，调用时的数量必须和声明时的一样。
  - 关键字参数：关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值。
  - 默认参数：调用函数时，缺省参数的值如果没有传入，则被认为是默认值。
  - 不定长参数：你可能需要一个函数能处理比当初声明时更多的参数。这些参数叫做不定长参数，声明时不会命名。

匿名函数

- python 使用 lambda 来创建匿名函数：
  - lambda只是一个表达式，函数体比def简单很多。
  - lambda的主体是一个表达式，而不是一个代码块。仅仅能在lambda表达式中封装有限的逻辑进去。
  - lambda函数拥有自己的命名空间，且不能访问自有参数列表之外或全局命名空间里的参数。
  - 虽然lambda函数看起来只能写一行，却不等同于C或C++的内联函数，后者的目的是调用小函数时不占用栈内存从而增加运行效率。

全局变量和局部变量

- 定义在函数内部的变量拥有一个局部作用域，叫做局部变量；定义在函数外的拥有全局作用域，叫做全局变量。
- 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。
- 调用函数时，所有在函数内声明的变量名称都将被加入到作用域中。

操作系统相关的调用和函数 - sys

- 系统相关的信息模块 sys。
- sys.argv 实现从程序外部向程序传递参数。
- sys.stdout sys.stdin sys.stderr 分别表示标准输入输出,错误输出的文件对象。
- sys.stdin.readline() 从标准输入读一行 sys.stdout.write("a") 屏幕输出a。
- sys.exit() 退出程序。
- sys.modules 是一个字典，用于保存所有导入的module。
- sys.platform 得到运行的操作系统环境。
- sys.path 是一个list,指明所有查找module的路径。

操作系统相关的调用和函数 - os

- os.environ 包含环境变量的映射关系；os.environ[“HOME”] 可以得到环境变量HOME的值。
- os.chdir(dir) 改变当前工作目录；os.chdir(‘d:\\outlook’)。
- os.getcwd() 得到当前目录。
- os.getegid() 得到有效组id；os.getgid() 得到有效组id。
- os.getuid() 得到用户id；os.geteuid() 得到有效用户id。
- os.setegid() os.setegid() os.seteuid() os.setuid()。
- os.getgruops() 得到用户组名称列表。
- os.getlogin() 得到用户登录名称。
- os.getenv() 得到环境变量。
- os.putenv() 设置环境变量。
- os.umask() 设置当前权限掩码，同时返回先前的权限掩码。在unix，Windows中有效。
- os.system(cmd) 利用系统调用，运行cmd命令。

目录

面向对象编程

- 面向对象编程 - Object Oriented Programming，简称OOP，是一种程序设计思想。OOP把对象作为程序的基本单元，一个对象包含了**数据**和**操作数据的函数**。
- 面向过程的程序设计把计算机程序视为一系列的命令集合，即一组函数的顺序执行。为了简化程序设计，面向过程把函数继续切分为子函数，即把大块函数通过切割成小块函数来降低系统的复杂度。
- 面向对象的程序设计把计算机程序视为一组对象的集合，而每个对象都可以接收其他对象发过来的消息，并处理这些消息，计算机程序的执行就是一系列消息在各个对象之间传递。
- 在Python中，所有数据类型都可以视为对象，当然也可以自定义对象。自定义的对象数据类型就是面向对象中的类（Class）的概念。

面向对象的设计思想

- 面向对象的设计思想是从自然界中来的，因为在自然界中，类（Class）和实例（Instance）的概念是很自然的。
- Class是一种抽象概念，比如我们定义的Class——Student，是指学生这个概念，而实例（Instance）则是一个个具体的Student，比如，Bart Simpson和Lisa Simpson是两个具体的Student。所以，面向对象的设计思想是抽象出Class，根据Class创建Instance。
- 面向对象的抽象程度又比函数要高，因为一个Class既包含数据，又包含操作数据的方法。

面向对象设计与面向对象编程的关系

- 面向对象设计（OOD）不会特别要求面向对象编程语言。事实上，OOD可以由纯结构化语言来实现，比如C，但如果想要构造具备对象性质和特点的数据类型，就需要在程序上作更多的努力。当一门语言内建OO特性，OO编程开发就会更加方便高效。
- 另一方面，一门面向对象的语言不一定会强制你写OO方面的程序。例如C++可以被认为“更好的 C”；而 Java，则要求万物皆类，此外还规定，一个源文件对应一个类定义。然而，在Python 中，类和OOP都不是日常编程所必需的。尽管它从一开始设计就是面向对象的，并且结构上支持 OOP，但 Python没有限定或要求你在你的应用中写OO的代码。
- OOP是一门强大的工具，不管你是准备进入，学习，过渡，或是转向，OOP都可以任意支配。

Python OOP常用术语

- 抽象/实现
- 封装/接口
- 合成
- 派生/继承/继承结构
- 泛化/特化
- 多态
- 自省/反射

类

- 类是一种数据结构，我们可以用它来定义对象，后者把数据值和行为特性融合在一起。类是现实世界的抽象的实体以编程形式出现。实例是这些对象的具体化。可以类比一下，类是蓝图或者模型，用来产生真实的物体（实例）。
- 在 Python 中，类声明与函数声明很相似，头一行用一个相应的关键字，接下来是一个作为它的 定义的代码体，如下所示：

```
def functionName(args):
    'function documentation string' #函数文档字符串
    function_suite #函数体。

class ClassName(object):
    'Click class documentation string'#类文档字符串
    class_suite#类体。
```

继承

- 继承是一种创建类的方法，在python中，一个类可以继承来自一个或多个父类。原始类称为基类或超类。
- 假如已经有几个类，而类与类之间有共同的变量属性和函数属性，那就可以把这几个变量属性和函数属性提取出来作为基类的属性。而特殊的变量属性和函数属性，则在本类中定义，这样只需要继承这个基类，就可以访问基类的变量属性和函数属性。可以提高代码的可扩展性。
- **抽象**即提取类似的部分。
- 基类就是抽象多个类共同的属性得到的一个类。

组合与派生

- 一个类被定义后，目标就是要把它当成一个模块来使用，并把这些对象嵌入到你的代码中去， 同其它数据类型及逻辑执行流混合使用。有两种方法可以在你的代码中利用类。
- 第一种是**组合** (composition)。就是让不同的类混合并加入到其它类中，来增加功能和代码重用性。你可以在一个大点的类中创建你自己的类的实例，实现一些其它属性和方法来增强对原来的类对象。
- 另一种是**派生**。派生就是子类在继承父类的基础上衍生出新的属性。子类中独有的父类中没有的，或子类定义与父类重名的对象。子类也叫派生类。
- 当类之间有显著的不同时，组合表现得很好；但当你设计“相同的类但有一些不同的功能”时，派生就是一个更加合理的选择了。

子类

- OOP的更强大方面之一是能够使用一个已经定义好的类，扩展它或者对其进行修改，而不会影响系统中使用现存类的其它代码片段。OOD允许类特征在子孙类或子类中进行继承。这些子类从基类（或 称祖先类，超类）继承它们的核心属性。而且，这些派生可能会扩展到多代。
- 在一个层次的派生关系中的相关类（或者是在类树图中垂直相邻）是父类和子类关系。从同一个父类派生出来的这些类（或者是在类树图中水平相邻）是同胞关系。父类和所有高层类都被认为是祖先。

私有化

- 默认情况下，属性在 Python 中都是“public”，类所在模块和导入了类所在模块的其他模块都可以访问到。很多OO语言给数据加上一些可见性，只提供访问函数来访问其值。
- 大多数 OO 语言提供“访问控制符”来限定成员函数的访问。
- 双下划线(__)
  - Python 为类元素（属性和方法）的私有性提供初步的形式。由双下划线开始的属性在运行时被 “混淆”，所以直接访问是不允许的。实际上，会在名字前面加上下划线和类名。
- 单下划线(_)
  - 简单的模块级私有化只需要在属性名前使用一个单下划线字符。 这就防止模块的属性用 “from mymodule import *”来加载。这是严格基于作用域的，所以这同样适合于函数。

目录

获取当前日期和时间

- 我们先看如何获取当前日期和时间：

```
>>> from datetime import datetime
>>> now = datetime.now() # 获取当前datetime
>>> print(now)
2015-05-18 16:28:07.198690
>>> print(type(now))
<class 'datetime.datetime'>
```

- 注意到datetime是模块，datetime模块还包含一个datetime类，通过from datetime import datetime导入的才是datetime这个类。如果仅导入import datetime，则必须引用全名datetime.datetime。datetime.now()返回当前日期和时间，其类型是datetime。

datetime转换为timestamp

- 在计算机中，时间实际上是用数字表示的。
- 我们把1970年1月1日 00:00:00 UTC+00:00时区的时刻称为**epoch time**，记为0（1970年以前的时间timestamp为负数），当前时间就是相对于epoch time的秒数，称为**timestamp**。你可以认为：timestamp = 0 = 1970-1-1 00:00:00 UTC+0:00。对应的北京时间是：timestamp = 0 = 1970-1-1 08:00:00 UTC+8:00。可见timestamp的值与时区毫无关系，因为timestamp一旦确定，其**UTC**时间就确定了，转换到任意时区的时间也是完全确定的，这就是为什么计算机存储的当前时间是以timestamp表示的，因为全球各地的计算机在任意时刻的timestamp都是完全相同的。把一个datetime类型转换为timestamp只需要简单调用timestamp()方法：

```
>>> from datetime import datetime
>>> dt = datetime(2015,4,19,12,20)   #用指定日期时间创建
>>> dt.timestamp()      #把datetime转换为timestamp
1429417200.0
```

- 注意Python的timestamp是一个浮点数。如果有小数位，小数位表示毫秒数。
- 某些编程语言（如Java和JavaScript）的timestamp使用整数表示毫秒数，这种情况下只需要把timestamp除以1000就得到Python的浮点表示方法。

timestamp转换为datetime

- 要把timestamp转换为datetime，使用datetime提供的fromtimestamp()方法：

```
>>> from datetime import datetime
>>> t = 1429417200.0
>>> print(datetime.fromtimestamp(t))
2015-04-19 12:20:00
```

- 注意到timestamp是一个浮点数，它没有时区的概念，而datetime是有时区的。本地时间是指当前操作系统设定的时区。例如北京时区是东8区，则本地时间：2015-04-19 12:20:00。即UTC+8:00时区的时间：2015-04-19 12:20:00 UTC+8:00。而此刻的格林威治标准时间与北京时间差了8小时，也就是UTC+0:00时区的时间应该是：2015-04-19 04:20:00 UTC+0:00。timestamp也可以直接被转换到UTC标准时区时间：

```
>>> from datetime import datetime
>>> t = 1429417200.0
>>> print(datetime.fromtimestamp(t)) # 本地时间
2015-04-19 12:20:00
>>> print(datetime.utcfromtimestamp(t)) # UTC时间
2015-04-19 04:20:00。
```

本地时间转换为UTC时间

- 本地时间是指系统设定时区的时间，例如北京时间是UTC+8:00时区的时间，而UTC时间指UTC+0:00时区的时间。一个datetime类型有一个时区属性tzinfo，但是默认为None，所以无法区分这个datetime到底是哪个时区，除非强行给datetime设置一个时区。

```
>>> from datetime import datetime, timedelta, timezone
>>> tz_utc_8 = timezone(timedelta(hours=8)) #创建时UTC+8:00
>>> now = datetime.now()
>>> now
datetime.datetime(2015, 5, 18, 17, 2, 10, 871012)

>>> dt = now.replace(tzinfo=tz_utc_8)      #强制设置为UTC+8:00
>>> dt
datetime.datetime(2015, 5, 18, 17, 2, 10, 871012,
tzinfo=datetime.timezone(datetime.timedelta(0, 28800)))
```

目录

正则表达式介绍（1）

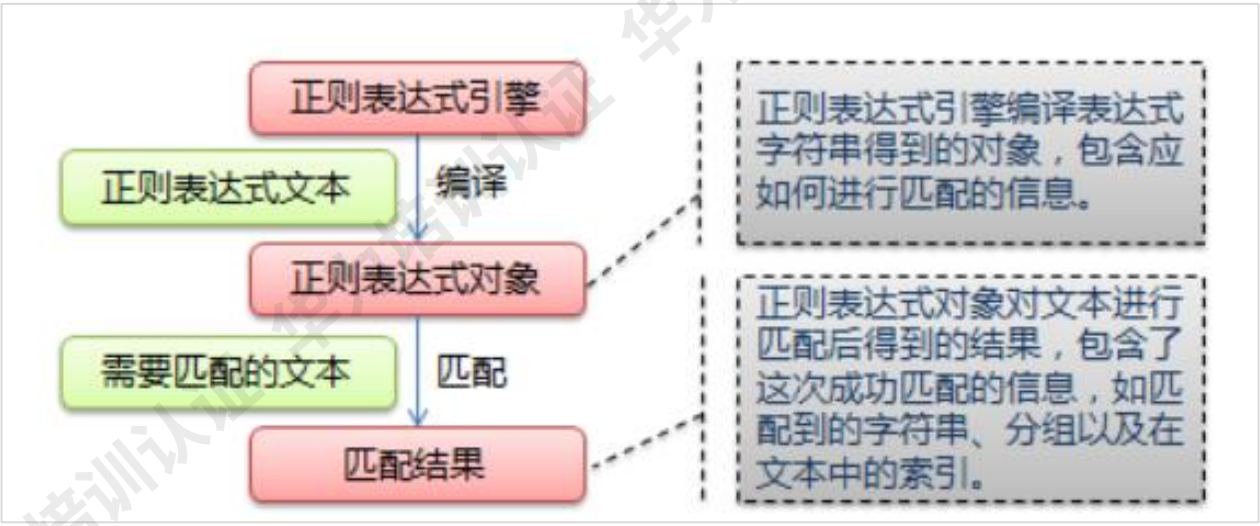
- 正则表达式是由一些字符和特殊符号组成的字符串，他们描述了模式的重复或者多个字符，于是正则表达式能够按照某种模式匹配一系列有相似特征的字符串。
- 正则表达式为高级的文本模式匹配、抽取、与/或文本形式的搜索和替换功能提供了基础。
- Python通过标准库中的re模块来支持正则表达式。

正则表达式介绍（2）

- 正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。
- re 模块使 Python 语言拥有全部的正则表达式功能。
- re 模块也提供了与这些方法功能完全一致的函数，这些函数使用一个模式字符串做为它们的第一个参数。
- compile 函数根据一个模式字符串和可选的标志参数生成一个正则表达式对象，该对象拥有一系列方法用于正则表达式匹配和替换。
- match 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回none。
- search 扫描整个字符串并返回第一个成功的匹配。

正则表达式的匹配流程

- 正则表达式的大致匹配过程是：依次拿出表达式和文本中的字符比较，如果每一个字符都能匹配，则匹配成功；一旦有匹配不成功的字符则匹配失败。



re模块

- Python通过re模块提供对正则表达式的支持。
- 使用re的一般步骤是先将正则表达式的字符串形式编译为Pattern实例，然后使用Pattern实例处理文本并获得匹配结果（一个Match实例），最后使用Match实例获得信息，进行其他的操作。

```
import re

# 将正则表达式编译成Pattern对象
pattern = re.compile(r'hello')

# 使用Pattern匹配文本，获得匹配结果，无法匹配时将返回None
match = pattern.match('hello world!')

if match:
    # 使用Match获得分组信息
    print(match.group())

# 输出
hello
```

re模块函数和正则表达式对象方法

函数/方法	描述	示例	res.group()/res
compile(pattern,flag=0)	使用任何可选的标记来编译正则表达式模式，返回正则对象	`res = re.compile(".*")` `print res.search("abcd").group()`	abcd
match(pattern,string,flag=0)	从字符串开头匹配	`res = re.match(".*","abcdxxxx")`	abcd
search(pattern,string,flag=0)	从字符串任何地方开始匹配	`res = re.search(".*","xxxabcdxx")`	abcd
findall(pattern,string,flag=0)	查找字符串中所有出现的正则表达式模式，返回列表	`res = re.findall("a", "abdadafaf")`	['a','a','a','a']
finditer(pattern,string,flag=0)	查找字符串中所有出现的正则表达式模式，返回迭代器	res = re.finditer("a", "abdadafaf") print res.next().group()	a
split(pattern,string,max=0)	根据正则模式把字符串分割为列表	`re.split(",","li,yang,zhao")`	['li','yang','zhao']
sub(pattern,repl,string,count=0)	使用repl替换正则表达式出现在string中的位置	`res = re.sub(",","-","l,y,z")`	l-y-z

常用的匹配对象方法

函数/方法	描述	示例	结果
group(num=0)	返回整个匹配对象，或者编号为num的特定子组	`Print(re.match(".*","abcdxxxx").group())`	abcdxxxx
groups(default=None)	返回一个包含所有子组的元组	`Print(re.search("(\w\w\w)-(\d\d\d)","abc-123").groups())`	('abc', '123')
groupdict(default=None)	返回一个包含所有匹配的命名子组的字典，子组名称为键值	`res = re.search("(?P<lamb>\w\w\w)-(?P<num>\d\d\d)","abc-123")` `Print(str(res.groupdict()))`	{'lamb':'abc', 'num': '123'}
re.i,re.IGNORECASE	不区分大小写	`res =re.search("abc","aBcxx",re.I)` `Print(res.group())`	aBc
re.L,re.LOCAL	根据所使用的本地语言环境，通过\w, \W,\b,\B,\s,\S实现匹配	`res = re.search("\w\w\w","aBcxx",re.L)` `Print(res.group())`	aBc
re.M,re.MULTILINE	^和$分别匹配目标字符串的起始和结尾，而不是严格匹配字符串本身的起始和结尾	`res = re.search("^aB","aBcxx",re.M)` `Print(res.group())`	aB

compile

- 这个方法是Pattern类的工厂方法，用于将字符串形式的正则表达式编译为Pattern对象。 第二个参数flag是匹配模式，取值可以使用按位或运算符‘|’表示同时生效，比如re.I | re.M。另外，你也可以在regex字符串中指定模式，比如：re.compile('pattern', re.I | re.M)与re.compile('(?im)pattern')是等价的。
- 可选值有：
  - re.I(re.IGNORECASE): 忽略大小写。
  - M(MULTILINE): 多行模式，改变‘^’和‘$’的行为。
  - S(DOTALL): 点任意匹配模式，改变‘.’的行为。
  - L(LOCALE): 使预定字符类 \w \W \b \B \s \S 取决于当前区域设定。
  - U(UNICODE): 使预定字符类 \w \W \b \B \s \S \d \D 取决于unicode定义的字符属性。
  - X(VERBOSE): 详细模式。这个模式下正则表达式可以是多行，忽略空白字符，并可以加入注释。

Pattern

- Pattern对象是一个编译好的正则表达式，通过Pattern提供的一系列方法可以对文本进行匹配查找。
- Pattern不能直接实例化，必须使用re.compile()进行构造。
- Pattern提供了几个可读属性用于获取表达式的相关信息：
  - pattern: 编译时用的表达式字符串。
  - flags: 编译时用的匹配模式。数字形式。
  - groups: 表达式中分组的数量。
  - groupindex: 以表达式中有别名的组的别名为键、以该组对应的编号为值的字典，没有别名的组不包含在内。

match (1)

- Match对象是一次匹配的结果，包含了很多关于此次匹配的信息，可以使用Match提供的可读属性或方法来获取这些信息。
- match属性：
  - string: 匹配时使用的文本。
  - re: 匹配时使用的Pattern对象。
  - pos: 文本中正则表达式开始搜索的索引。值与Pattern.match()和Pattern.seach()方法的同名参数相同。
  - endpos: 文本中正则表达式结束搜索的索引。值与Pattern.match()和Pattern.seach()方法的同名参数相同。
  - lastindex: 最后一个被捕获的分组在文本中的索引。如果没有被捕获的分组，将为None。
  - lastgroup: 最后一个被捕获的分组的别名。如果这个分组没有别名或者没有被捕获的分组，将为None。

match (2)

- match方法：
  - **group([group1, ...]):** 获得一个或多个分组截获的字符串；指定多个参数时将以元组形式返回。group1可以使用编号也可以使用别名；编号0代表整个匹配的子串；不填写参数时，返回group(0)；没有截获字符串的组返回None；截获了多次的组返回最后一次截获的子串。
  - **groups([default]):** 以元组形式返回全部分组截获的字符串。相当于调用group(1,2,…last)。default表示没有截获字符串的组以这个值替代，默认为None。
  - **groupdict([default]):** 返回以有别名的组的别名为键、以该组截获的子串为值的字典，没有别名的组不包含在内。default含义同上。
  - **start([group]):** 返回指定的组截获的子串在string中的起始索引（子串第一个字符的索引）。group默认值为0。
  - **end([group]):** 返回指定的组截获的子串在string中的结束索引（子串最后一个字符的索引+1）。group默认值为0。
  - **span([group]):** 返回(start(group), end(group))。
  - **expand(template):** 将匹配到的分组代入template中然后返回。template中可以使用\id或\g<id>、\g<name>引用分组，但不能使用编号0。\id与\g<id>是等价的；但\10将被认为是第10个分组，如果你想表达\1之后是字符'0'，只能使用\g<1>0。

特殊符号和字符 - 符号 (1)

表示法	描述	匹配的表达式	res.group()
literal	匹配文本字符串的字面值literal	res=re.search(“foo”,”xxxfooxxx”)	foo
re1|re 2	匹配正则表达式re1或者re 2	res=re.search(“foo|bar”,”xxxfooxxx”)	foo
.	匹配任何字符（除\n之外）	res=re.search(“b.b”,”xxxbobxxx”)	bob
^	匹配字符串起始部分	res=re.search("^b.b", "bobx xx")	bob
$	匹配字符串结尾部分	res=re.search("b.b$","xx xbob")	bob
*	匹配0次或者多次前面出现的正则表达式(从字符串开头匹配)	res= re.search("bob*","bobbo") res1= re.search(".*","bobboddd")	Bobb bobboddd
+	匹配1次或者多次前面出现的正则表达式	res= re.search("bob+","xxxxbobbbbbob")	bobbbb

特殊符号和字符 - 符号 (2)

表示法	描述	匹配的表达式	res.group()
?	匹配0次或者1次前面出现的正则表达式	`res=re.search("bob?","bobbod")`	bob
{N}	匹配N次前面出现的正则表达式	`res=re.search("bob{2}","bobbbbod")`	bobb
{M,N}	匹配M到N次前面出现的正则表达式	`res=re.search("bob{2,3}","bobbbbod")`	bobbb
[…]	匹配来自字符集的任意单一字符	`res= re.search("[b,o,b]","xxbobxx")`	b
[..X-Y..]	匹配x~y范围中任意单一字符	`res= re.search("[a-z]","xxbobxx")`	x
[^…]	不匹配字符串中任何一个字符，包括某一范围的字符	`res= re.search("[^a-z]","xx214bobxx")` `res1=re.search("[^2,x,1]","xx214bobxx")`	2 4
(*|+|{})?	用于匹配频繁出现或者重复出现的符号的非贪婪版本	`res= re.search(".[+|?]?[1-9]","ds4b")`	s4

特殊符号和字符 - 特殊字符

表示法	描述	匹配的表达式	res.group()
\d	任何十进制数字（\D不匹配任何十进制数字）	res=re.search("xx\dxx","oxx4xxo")	xx4xx
\w	匹配任何字母或者数字的字符（\W为不匹配）	res=re.search("xx\w\wxx","oxxa4xxo")	xxa4xx
\s	匹配任何空格字符（\S为不匹配）	res=re.search("xx\sxx","oxx xxo")	xx xx
\b	配任何单词边界（\B相反）	res=re.search(r"\bthe","xxx the xxx")	the
\N	匹配以保存的子组N		
\c	逐字匹配任何特殊字符c	res=re.search("*","x*x")	*
\A(\Z)	匹配字符串的起始（结束）	res=re.search("\ADear","Dear Mr.Li")	Dear

目录

Python文件操作

- 文件操作对编程语言的重要性很高，如果数据不能持久读取，保存，使用等，信息技术也就失去了意义。
- 常见的文件操作类型包括对文件的打开和关闭，对文件的读写操作，文件备份等等。

文件操作（1）

- 打开文件：
  - f.open(‘文件名’,‘访问模式’)
  - 常用的访问模式：

访问模式	说明
r	以只读方式打开文件。
w	打开一个文件只用于写入。
a	打开一个文件用于追加。
rb	以二进制格式打开一个文件用于只读。
wb	以二进制格式打开一个文件只用于写入。
ab	以二进制格式打开一个文件用于追加。
r+	打开一个文件用于读写。
w+	打开一个文件用于读写。
a+	打开一个文件用于读写。
rb+	以二进制格式打开一个文件用于读写。
wb+	以二进制格式打开一个文件用于读写。
ab+	以二进制格式打开一个文件用于追加。

文件操作（2）

- 写数据：

```
f = open(“name.txt”,“w”)
f.write(“libai”)
f.close()
```

- 读数据：

```
f = open(“name.txt”,“r”)

lines = f.realines()
for line in lines:
    print(line)
```

- 关闭文件：

```
f.close()
```

批量修改文件名

- 获取目标文件夹
  - dirName = input(“请输入指定文件夹：”)。
- 获取文件夹中的文件名
  - fileNames = dirName.listdir(dirName)。
  - os.chdir(dirName)。
- 重命名
  - for name in fileNames。
  - os.rename(name,“张三”+name)。

其他文件操作函数

- 读写文件：
  - f.write(“a”) f.write(str) 写一字符串 f.writeline() f.readlines() 与下read类同。
  - f.read()全读出来 f.read(size) 表示从文件中读取size个字符。
  - f.readline() 读一行,到文件结尾,返回空串. f.readlines() 读取全部，返回一个list. list每个元素表示一行，包含“\n”\。
  - f.tell()返回当前文件读取位置。
  - f.seek(off, where) 定位文件读写位置. off表示偏移量，正数向文件尾移动，负数表示向开头移动。
  - where为0表示从开始算起,1表示从当前位置算,2表示从结尾算。
  - f.flush()刷新缓存。

思考题

1.Python是完全面向对象的语言。下列选项属于Python对象的是?（）

A: 函数

B: 模块

C: 数字

D: 字符串

2.不属于对Python文件对象操作的步骤是? （）

A: open

B: delete

C: read

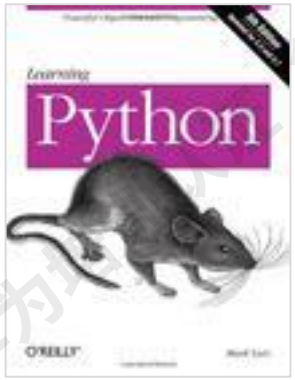
D: write

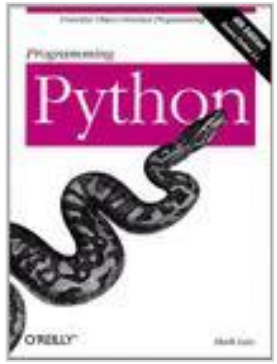
本章总结

- 本课程重点介绍了Pyhon语言和基础语法，如Python编译环境介绍与安装数字表达式、变量、语句、字符串、获取用户的输入、函数、模块等常用操作。

更多信息

- 官方网站
  - www.python.org
- 案头书
  - 《Learning Python》
  - 《Python标准库》
  - 《Programming Python》





学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

数学基础

www.huawei.com

HUAWEI

目标

- 学完本课程后，您将能够：
  - 掌握线性代数的基础知识及应用
  - 掌握概率与信息论的基础知识及应用
  - 掌握数值计算的作用以及优化问题的分类与解决方法

目录

线性代数

- **线性代数**是代数学的一个分支，主要处理线性问题。**线性关系**是指数学对象之间的关系是以一次形式来表达的。线性代数需要解决的第一个问题就是求解**线性方程组**。
- **行列式**和**矩阵**为处理线性问题提供了有力的工具，也推动了线性代数的发展。**向量**概念的引入，形成了向量空间的概念，而线性问题都可以用向量空间的观点加以讨论。因此向量空间及其线性变换，以及与此相联系的矩阵理论，构成了线性代数的中心内容。
- 它的特点是研究的变量数量较多，关系复杂，方法上既有严谨的逻辑推证、又有巧妙的归纳综合、也有繁琐和技巧性很强的数字计算。

引入案例 (1)

- 为了避免肥胖，提升员工健康状况，2018年初大数据部门组织月度跑步活动。规则如下：部门为参与者在月初定制月度目标，对完成目标者进行奖励，对未完成者进行惩罚，奖惩金额为：

$$w_i = (s_i - d_i)x_i = h_i x_i,$$

其中w_i为第i月总奖惩金额，s_i为总公里数，d_i为月度目标，h_i为实际距离与月度目标的差，x_i为每月对每公里的奖惩金额。活动影响良好，同时云部门也开展起来。以下数据为第一季度部分参与员工每月与月度目标差以及第一季度的总奖励值：

月份 姓名	h_1	h_2	h_3	w
小陈	10	8	12	20
小刘	4	4	2	8
小姚	2	-4	-2	-5

表1 大数据部门

月份 姓名	h_1	h_2	h_3	w
小李	2	4	5	10
小黄	4	2	2	6
小傅	-2	2	2	3

表2 云部门

引入案例 (2)

- 根据上述案例，是否可以求出大数据部门每月制定的每公里的奖励金额x_i？根据数据可列出以下方程组：

$$\begin{cases} 10x_1 + 8x_2 + 12x_3 = 20 \\ 4x_1 + 4x_2 + 2x_3 = 8 \\ 2x_1 - 4x_2 - 2x_3 = -5 \end{cases} \quad (1.1)$$

那么问题转化为，若能求得此方程组的解，即可求得部门每月制定的每公里的奖励金额。

标量、向量、矩阵

- **向量**：一个向量是一列数。这些数是有序排列的。通过次序中的索引，我们可以确定每个单独的数。例如：

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- **矩阵**：由$m \times n$个数$a_{ij}(i = 1，2，\cdots，m；j = 1，2，\cdots，n)$排成$m$行$n$列的数表：

$$\begin{matrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{matrix}$$

称为m行n列矩阵，简称$m \times n$矩阵。记作：

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

简记为$A = A_{m \times n} = (a_{ij})_{m \times n} = (a_{ij})$.特殊的，行数与列数都等于$n$的矩阵称为$n$阶矩阵或$n$阶方阵。

行列式

- **行列式**，记作$\det(A)$，是一个将方阵A映射到实数的函数。

$$\det(A)=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{vmatrix}.$$

- 行列式的意义：
  - 行列式等于矩阵特征值的乘积。
  - 行列式的绝对值可以用来衡量矩阵参与矩阵乘法后空间扩大或缩小了多少。如果行列式是0，那么空间至少沿着某一维完全收缩了，使其失去了所有的体积；如果行列式是1，那么这个转换保持空间体积不变。

矩阵的运算

- **矩阵加法**：设$A = (a_{ij})_{s\times n}$，$B = (b_{ij})_{s\times n}$都是数域K上的$s \times n$矩阵，矩阵的和定义为$C = A + B = (a_{ij} + b_{ij})_{s\times n}$.

  注：只有矩阵A、B的行列数一样，两矩阵才可以相加。

- **标量和矩阵乘法**：设$A = (a_{ij})_{s\times n}$，$k \in K$，k与矩阵A的乘积定义为$kA = (ka_{ij})_{s\times n}$.标量与矩阵相加同理。

- **矩阵乘法**：若矩阵$A = (a_{ij})_{s\times n}$，$B = (b_{ij})_{n\times p}$，则

$$C = AB = (c_{ij})_{s\times p},$$

  其中$C_{i,j} = \sum_k A_{i,k}B_{k,j}$.

  注：矩阵A的列数必须和矩阵B的行数相等，AB才有意义。

矩阵的转置

- **转置矩阵**：把矩阵$A=(a_{ij})_{s\times n}$的行与列相互交换产生的矩阵称为A的转置，记作A'或A^T。
- 例如：

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \Rightarrow A^T=\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix}.$$

- 转置矩阵的性质：
  - $(A^T)^T=A$
  - $(A+B)^T=A^T+B^T$
  - $(\lambda A)^T=\lambda A^T$
  - $(AB)^T=B^TA^T$

迹运算

- **迹运算**返回的是矩阵对角元素的和：

$$Tr(A) = \sum_i A_{i,i}.$$

- 迹运算的性质：
  - $Tr(A) = Tr(A^T)$
  - $Tr(a) = a$
  - $Tr(ABC) = Tr(CAB) = Tr(BCA)$

案例计算

- 根据引入案例，可得：
  - 第一季度大数据部门与云部门跑步情况可表示如下：

$$A=\begin{bmatrix}10 & 8 & 12\\ 4 & 4 & 2\\ 2 & -4 & -2\end{bmatrix},\quad B=\begin{bmatrix}2 & 4 & 5\\ 4 & 2 & 2\\ -2 & 2 & 2\end{bmatrix}.$$

  - 对上述矩阵做以下矩阵运算：

$$C_1=A+B=\begin{bmatrix}12 & 12 & 17\\ 8 & 6 & 4\\ 0 & -2 & 0\end{bmatrix},\quad C_2=2A=\begin{bmatrix}20 & 16 & 24\\ 8 & 8 & 4\\ 4 & -8 & -4\end{bmatrix},\quad C_3=AB=\begin{bmatrix}28 & 80 & 90\\ 20 & 28 & 32\\ -8 & -4 & -2\end{bmatrix}.$$

  - 根据矩阵相乘规则，方程组（1.1）可以用矩阵表示为：

$$\begin{cases}10x_1+8x_2+12x_3=20\\ 4x_1+4x_2+2x_3=8\\ 2x_1-4x_2-2x_3=-5\end{cases}\Rightarrow\begin{bmatrix}10 & 8 & 12\\ 4 & 4 & 2\\ 2 & -4 & -2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}20\\ 8\\ -5\end{bmatrix}\Rightarrow Ax=C.$$

目录

单位矩阵、逆矩阵

- **单位矩阵：**所有沿主对角线的元素都是1，而其他位置的所有元素都是0的矩阵。任意矩阵与单位矩阵相乘，都不会改变。

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

- **逆矩阵**：**方阵**A的逆矩阵记作A^{-1}，其满足$A^{-1}A = I_n$.

对角矩阵

- **对角矩阵**：主对角线之外的元素皆为0的矩阵。常写为$diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$.

$$D=\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \vdots & \lambda_n\end{bmatrix}.$$

- 对角矩阵的性质：
  - 对角矩阵的和、差、积、方幂为主对角线上元素的和、差、积、方幂。
  - 其逆矩阵为：

$$D^{-1}=\begin{bmatrix}\lambda_1^{-1} & 0 & \cdots & 0\\ 0 & \lambda_2^{-1} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \vdots & \lambda_n^{-1}\end{bmatrix}.$$

对称矩阵

- **对称矩阵：**设方阵$A=(a_{ij})_{n\times n}$，满足$A^T=A$，即$a_{ij}=a_{ji}$，则称A为对称矩阵。

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix}.$$

- **正交矩阵：**设方阵$A=(a_{ij})_{n\times n}$，满足$AA^T=A^TA=I_n$，则称A为正交矩阵。即$A^{-1}=A^T$.

目录

特征分解 (1)

- **特征分解**是使用最广的矩阵分解之一，即我们通过将矩阵分解成一组**特征向量和特征值**乘积的方法来发现矩阵表示成数组元素时不明显的函数性质。
- 设A是数域K上的n级矩阵，如果K^n中有非零列向量α使得

$$A\alpha = \lambda\alpha，且\lambda \epsilon K，$$

则称λ是A的一个**特征值**，称α是A的属于特征值λ的一个**特征向量**。

- 例如：

$$A=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}, \quad \alpha=\begin{bmatrix}1\\ 1\end{bmatrix}, \quad 由于A\alpha=\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix}=2\begin{pmatrix}1\\ 1\end{pmatrix}=2\alpha.$$

因此，2是A的一个特征值，α是A的属于特征值2的一个特征向量。

特征分解 (2)

- 怎样求矩阵A的特征值与特征向量：

$$
\begin{aligned}
& A\alpha = \lambda\alpha \\
\Leftrightarrow \quad & A\alpha - \lambda\alpha = 0 \\
\Leftrightarrow \quad & (A - \lambda I)\alpha = 0 \\
\overset{\alpha \neq 0}{\Leftrightarrow} \quad & |A - \lambda I| = 0 \\
\Leftrightarrow \quad & \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0.
\end{aligned}
$$

其中，$|A - \lambda I| = 0$称为矩阵A的特征方程，λ为特征方程的解，即特征根，将特征根λ代入$A\alpha = \lambda\alpha$即可求得特征向量α。

特征分解 (3)

- 例：求$A=\begin{bmatrix}3 & -1\\ -1 & 3\end{bmatrix}$的特征值和特征向量。

解：A的特征多项式为：$\begin{vmatrix}3-\lambda & -1\\ -1 & 3-\lambda\end{vmatrix}=(3-\lambda)^2-1=(4-\lambda)(2-\lambda)$，所以A的特征值为$\lambda_1=2$，$\lambda_2=4$.

当$\lambda_1=2$时，对应的特征向量满足$\begin{bmatrix}3-2 & -1\\ -1 & 3-2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}$，解得$x_1=x_2$，所以对应的特征向量为$p_1=\begin{bmatrix}1\\ 1\end{bmatrix}$.故$\lambda_1=2$时的特征向量为$kp_1(k\neq 0)$.

当$\lambda_2=4$时，同理解得$x_1=-x_2$，所以对应的特征向量为$p_2=\begin{bmatrix}-1\\ 1\end{bmatrix}$.故$\lambda_2=4$时的特征向量为$kp_2(k\neq 0)$.

特征分解 (4)

- 设$\boldsymbol{A}$有n个线性无关的特征向量α_1，α_2，$\cdots$，α_n，相对应的特征值为λ_1，λ_2，$\cdots$，λ_n，则$\boldsymbol{A}$的**特征分解**为：

$$A = Pdiag(\lambda)P^{-1},$$

其中$P = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$，$\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$.

- **矩阵的正定性**：
  - 所有特征值都是正数的矩阵称为正定。
  - 所有特征值都是非负数的矩阵称为半正定。
  - 所有特征值都是负数的矩阵称为负定。
  - 所有特征值都是非正数的矩阵称为半负定。

奇异值分解

- **奇异值分解**：将矩阵分解为奇异向量和奇异值。可以将矩阵$A = (a_{ij})_{m \times n}$分解为三个矩阵的乘积：

$$A = UDV^T,$$

其中$U = (b_{ij})_{m \times m}$，$D = (c_{ij})_{m \times n}$，$V^T = (d_{ij})_{n \times n}$.矩阵$U$和$V$都为正交矩阵，矩阵$U$的列向量称为左奇异向量，矩阵$V$的列向量称为右奇异向量，$D$为对角矩阵（不一定为方阵），$D$对角线上的元素称为矩阵$A$的奇异值。

Moore - Penrose伪逆

- **Moore-Penrose伪逆**使得在求解线性方程$Ax = y(A = (a_{ij})_{m\times n},\ m \neq n)$的问题上取得了进展。矩阵A的伪逆定义为：

$$A^{+} = \lim_{\alpha \searrow 0}(A^{T}A + \alpha I)^{-1}A^{T},$$

计算伪逆的算法实际基于以下公式

$$A^{+} = VD^{+}U^{T},$$

其中U、D、V是矩阵A奇异值分解后得到的矩阵。对角矩阵D的伪逆D^{+}是其非零元素取倒数之后再转置得到的。

实例：主成分分析 (1)

- **主成分分析（Principal Component Analysis，PCA）**：是一种统计方法。通过**正交变换**将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。
- 基本原理：假设有n个对象，每一个对象都有x_1， x_2，…，x_p个要素构成，它们所对应的要素数据用下表给出：

要素 / 对象	x_1	x_2	…	x_j	…	x_p
1	x_{11}	x_{12}	…	x_{1j}	…	x_{1p}
2	x_{21}	x_{22}	…	x_{2j}	…	x_{2p}
…	…	…	…	…	…	…
i	x_{i1}	x_{i2}	…	x_{ij}	…	x_{ip}
…	…	…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{nj}	…	x_{np}

实例：主成分分析 (2)

- 原变量为x_1，x_2，…，x_p，降维处理后，设它们的综合指标，即新变量为z_1，z_2，…，$z_m(m \le p)$，z_1，z_2，…，z_m分别称为x_1，x_2，…，x_p的第1，2，…，m主成分。可表示为：

$$\begin{cases} z_1 = l_{11}x_1 + l_{12}x_2 + \ldots + l_{1p}x_p \\ z_2 = l_{21}x_1 + l_{22}x_2 + \ldots + l_{2p}x_p \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ z_m = l_{m1}x_1 + l_{m2}x_2 + \ldots + l_{mp}x_p \end{cases}.$$

- 要想求得m个主成分，步骤如下：
  - 系数l_{ij}满足如下规则：z_i与$z_j(i \ne j; i, j = 1, 2, \ldots, m)$相互无关。$z_1$是$x_1$，$x_2$，…，$x_p$的一切线性组合中方差最大者；$z_2$是与$z_1$不相关的$x_1$，$x_2$，…，$x_p$的所有线性组合中方差最大者；以此类推；$z_m$是与$z_1$，$z_2$，…，$z_{m-1}$都不相关的$x_1$，$x_2$，…，$x_p$的所有线性组合中方差最大者。
  - 依据上述规则可知，l_{ij}分别是x_1，x_2，…，x_p的**相关系数矩阵**的m个较大**特征值**所对应的**特征向量**。
  - 当前i个主成分**累计贡献率**达到85%-90%，就取前i个主成分作为新变量。

实例：主成分分析 (3)

- 相关系数矩阵及相关系数：

$$R=\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ r_{21} & r_{22} & \cdots & r_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ r_{p1} & r_{p2} & \cdots & r_{pp} \end{bmatrix}, \qquad r_{ij}=\frac{\sum_{k=1}^{p}(x_{ki}-\overline{x_i})(x_{kj}-\overline{x_j})}{\sqrt{\sum_{k=1}^{p}(x_{ki}-\overline{x_i})^2\sum_{k=1}^{p}(x_{kj}-\overline{x_j})^2}}.$$

- 主成分贡献率及累计贡献率：

$$Q_i=\frac{\lambda_i}{\sum_{k=1}^{p}\lambda_k}(i=1,2,\cdots,p), \qquad Q=\frac{\sum_{k=1}^{i}\lambda_k}{\sum_{k=1}^{p}\lambda_k}(i=1,2,\cdots,p).$$

目录

1. 线性代数
2. **概率与信息论**
   - 概率论基本概念
   - 随机变量及其分布函数
   - 随机变量的数字特征
   - 信息论
3. 数值计算

为什么要使用概率？

- 概率论使我们能够提出不确定的声明以及在不确定性存在的情况下进行推理，而信息论使我们能够量化概率分布中的不确定性总量。
- 不确定性有3种可能的来源：
  - 被建模系统内在的随机性。
  - 不完全观测。
  - 不完全建模。

随机试验

- 满足以下三个特点的试验称为**随机试验**:
  - 可以在相同的条件下重复进行。
  - 每次试验的可能结果不止一个，并且能事先明确试验的所有可能结果。
  - 进行一次试验之前不能确定哪一个结果会出现。
- 举例:
  - E_1:抛两枚硬币，出现正面H、反面T的情况。
  - E_2:抛一枚骰子，观察可能出现的点数情况。

样本点、样本空间、随机事件

- **样本点（sample point）**：随机试验的每一个可能的结果称为样本点，用e表示。
- **样本空间（sample space）**：随机试验E的所有可能结果组成的集合，记作S，即$S = \{e_1, e_2, \ldots, e_n\}$.
- **随机事件（random variables events）**：样本空间S的任一子集A。属于事件A的样本点出现，则称事件A发生。特别的，仅含一个样本点的随机事件，称为**基本事件**。
- 举例：

随机试验E_2:抛一枚骰子，观察可能出现的点数情况。

样本空间为：$S = \{1, 2, 3, 4, 5, 6\}$.

样本点为：$e_i = 1, 2, 3, 4, 5, 6$.

随机事件A_1：“骰子出现的点数为5”，即$A_1 = \{x | x = 5\}$.

频率与概率

- **频率**：在相同的条件下，进行n次试验，在这n次试验中，事件A发生的次数n_A称为事件A发生的频数。比值$\frac{n_A}{n}$称为事件A发生的概率，并记成$f_n(A)$.
- **概率**：设E是随机试验，S是其样本空间。对于E的每一事件A赋予一个实数，记为$P(A)$，称为事件A的概率，如果集合函数$P(*)$，满足下列条件：
  - 非负性：对于每一个事件A，有$0 \leq P(A) \leq 1$.
  - 规范性：对于必然事件S，有$P(S) = 1$.
  - 可列可加性：设A_1，A_2，...是两两互不相容的事件，即对于$A_i A_j = \emptyset$，$i \neq j$，i，$j = 1$，2，$\cdots$，有$P(A_1 \cup A_2 \cup \cdots) = P(A_1) + P(A_2) + \cdots$.

随机变量

- **随机变量（random variable）**：表示随机试验各种结果的实值单值函数。
- 举例1：随机试验E_4:抛两枚骰子，观察可能出现的点数的和。试验的样本空间是$S=\{e\}=\{(i，j)|i，j=1，2，3，4，5，6\}$，$i$，$j$分别是第1次，第2次出现的点数，以$X$记为两球号码之和，则$X$ 是一个随机变量。

$$X=X(e)=X(i，j)=i+j，i，j=1，2，\cdots，6.$$

- 举例2：随机试验E_1：抛两枚硬币，出现正面H、反面T的情况。试验的样本空间是$S=\{HH，HT，TH，TT\}$，以Y记为两次投掷硬币得到反面T的总数，则Y是一个随机变量。

$$Y=Y(e)=\begin{cases}0,e=HH,\\1,e=HT,TH,\\2,e=TT.\end{cases}$$

目录

HUAWEI

离散型随机变量与分布律

- **离散型随机变量**：随机变量的全部可能取到的值是有限个或可列无限多个。如记录某监控卡口在1分钟内通过的的车辆数目。
- **分布律**：设离散型随机变量X的所有可能取值为$x_k(k=1，2，\cdots)$，X取各个可能值的概率，即事件$\{X=x_k\}$的概率，为

$$P\{X=x_k\}=p_k，k=1，2，\cdots.$$

由概率的定义，p_k满足如下两个条件：

（1）$p_k \geq 0$，$k=1$，2，$\cdots$.

（2）$\sum_{k=1}^{\infty} p_k = 1.$

分布律也可以用表格的形式来表示：

X	x_1	x_2	$\cdots$	x_n	$\cdots$
p_k	p_1	p_2	$\cdots$	p_n	$\cdots$

特殊分布 - 伯努利分布

- **伯努利分布（0-1分布，两点分布，a-b分布）：** 设随机变量X只可能取0与1两个值，它的分布律是:

$$P\{X=k\}=p^k(1-p)^{1-k},\ k=0,\ 1\ (0<p<1),$$

则称X服从以p为参数的伯努利分布。

- 伯努利分布的分布律也可以写成:

X	0	1
p_k	$1-p$	p

其中，$E(X)=p$，$Var(X)=p(1-p)$.

特殊分布 - 二项分布

- **n次独立重复试验：**将实验E重复进行n次，若各次试验的结果互不影响，则称这n次实验是相互独立的。
- 满足如下条件的试验称为**n重伯努利试验**：
  - 每次实验都在相同的条件下重复进行。
  - 每次试验只有两个可能的结果：A及$\bar{A}$且$P(A)=p$.
  - 每次试验的结果相互独立。

若用X表示n重伯努利试验中事件A发生的次数，则n次试验中事件A发生k次的概率为：

$$P(X=k)=C_n^k p^k(1-p)^{n-k},\ k=0,\ 1,\ 2,\ \cdots n,$$

此时称X服从参数为n，p的**二项分布**，记为$X\sim B(n,\ p)$.其中$E(X)=np$，$Var(x)=np(1-p)$.

特殊分布 - 泊松分布

- **泊松（Poisson）定理：**设$\lambda > 0$为一常数，n是任意正整数。设$np = \lambda$，则对任一固定的非负整数k，有

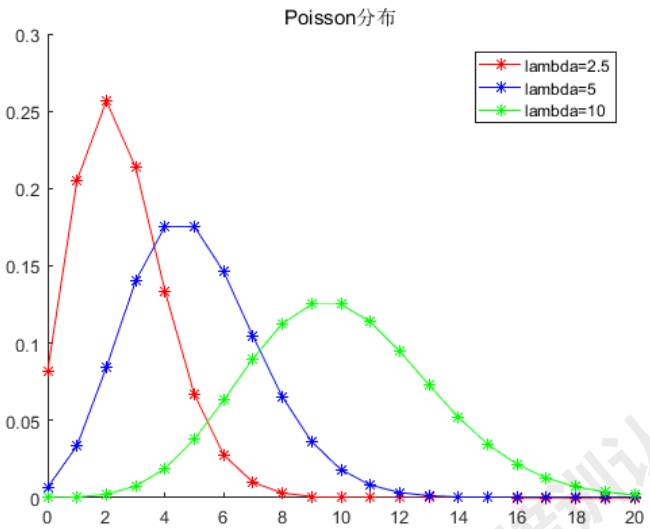
$$\lim_{n\to\infty} C_n^k p^k (1-p)^{n-k} \approx \frac{\lambda^k e^{-\lambda}}{k!}.$$

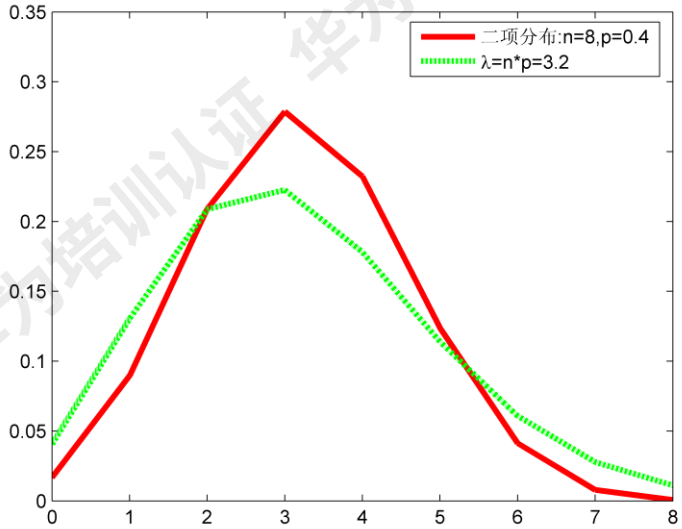
- **泊松分布：**若随机变量所有可能的取值为0，1，2，…，而取每个值的概率为：

$$P\{X = k\} = \frac{\lambda^k e^{-\lambda}}{k!}，\ k = 0，1，2，\cdots,$$

则称X服从参数为λ的泊松分布，记为：$X \sim P(\lambda)$.其中，$E(X) = \lambda，\ D(X) = \lambda.$

泊松分布与二项分布的关系





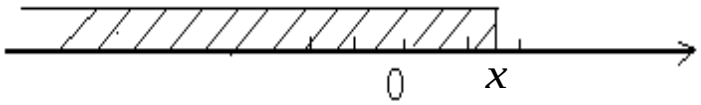
- 泊松分布与二项分布的数学模型都是伯努利概型，泊松分布式二项分布当n很大p很小时的近似计算。

分布函数

- **分布函数**：设X是一个随机变量，x是任意实数，函数$F(x)$称为X的分布函数。

$$F(x) = P\{X \le x\}, \quad -\infty < x < \infty.$$

  - 分布函数$F(x)$具有以下的基本性质：
    - $F(x)$是一个不减函数。
    - $0 \le F(x) \le 1$，且$F(-\infty) = \lim\limits_{x \to -\infty} F(x) = 0$，$F(\infty) = \lim\limits_{x \to \infty} F(x) = 1$.
    - $F(x+0) = F(x)$，即$F(x)$是右连续的。
  - 分布函数$F(x)$的意义：如果将X看成是数轴上的随机点的坐标，那么分布函数$F(x)$在x处的函数值就表示X落在区间$(-\infty, x]$上的概率。



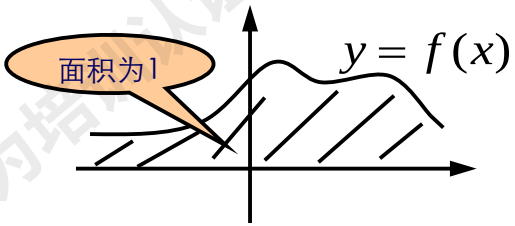
连续型随机变量与概率密度函数

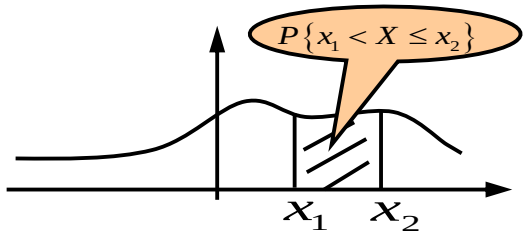
- 如果对于随机变量X的分布函数$F(x)$，存在非负函数$f(x)$，使对于任意实数x有

$$F(x)=\int_{-\infty}^{x}f(t)dt,$$

则称X为**连续型随机变量**，其中函数$f(x)$称为X的**概率密度函数**，简称概率密度。

- 概率密度$f(x)$具有以下性质：
  - $f(x)\geq 0.$
  - $\int_{-\infty}^{+\infty}f(x)dx=1.$
  - 对于任意实数x_1，$x_2(x_1<x_2)$，$P\{x_1<X\leq x_2\}=F(x_2)-F(x_1)=\int_{x_1}^{x_2}f(x)dx.$
  - 若$f(x)$在点x处连续，则有$F'(x)=f(x).$
  - 连续型随机变量X取任一实数的概率值为0，即$P(X=a)=0.$



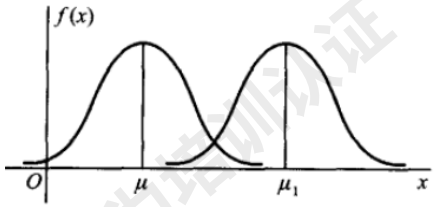


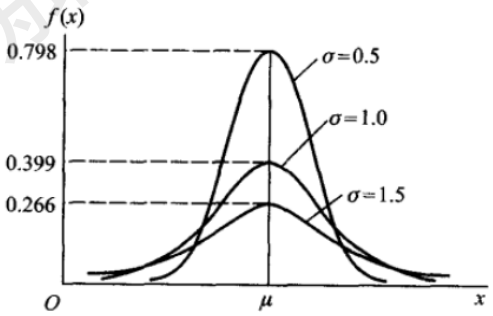
特殊分布 - 正态分布

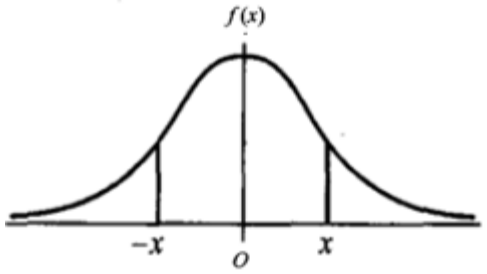
- 若连续型随机变量X的概率密度函数为

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}},\quad -\infty<x<\infty,$$

其中μ，$\sigma(\sigma>0)$为常数，则称X服从参数为μ，σ的正态分布或高斯分布，记为$X\sim N(\mu,\ \sigma^2)$.特别当$\mu=0$，$\sigma=1$时称随机变量X服从标准正态分布，记为$X\sim N(0,\ 1)$.





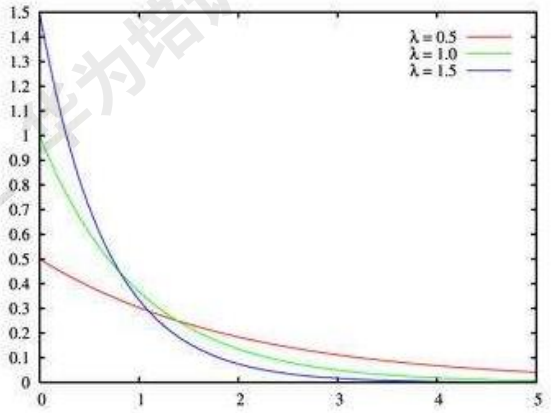


特殊分布 - 指数分布

- 若连续型随机变量X的概率密度为

$$f(x)=\begin{cases}\lambda e^{-\lambda x}, x>0\\ 0 \quad\quad ,otherwise\end{cases}$$

其中$\lambda>0$为常数，表示随机事件发生一次的时间，则称X服从参数为λ的指数分布，记为$X\sim E(\lambda).E(X)=\frac{1}{\lambda}$，$Var(X)=\frac{1}{\lambda^2}$.

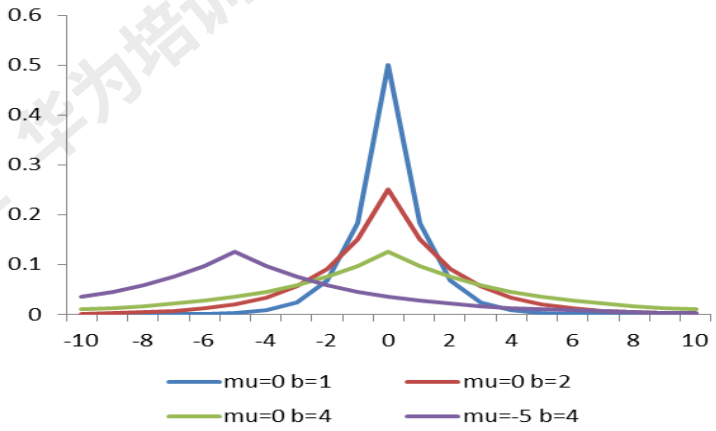


特殊分布 - Laplace分布

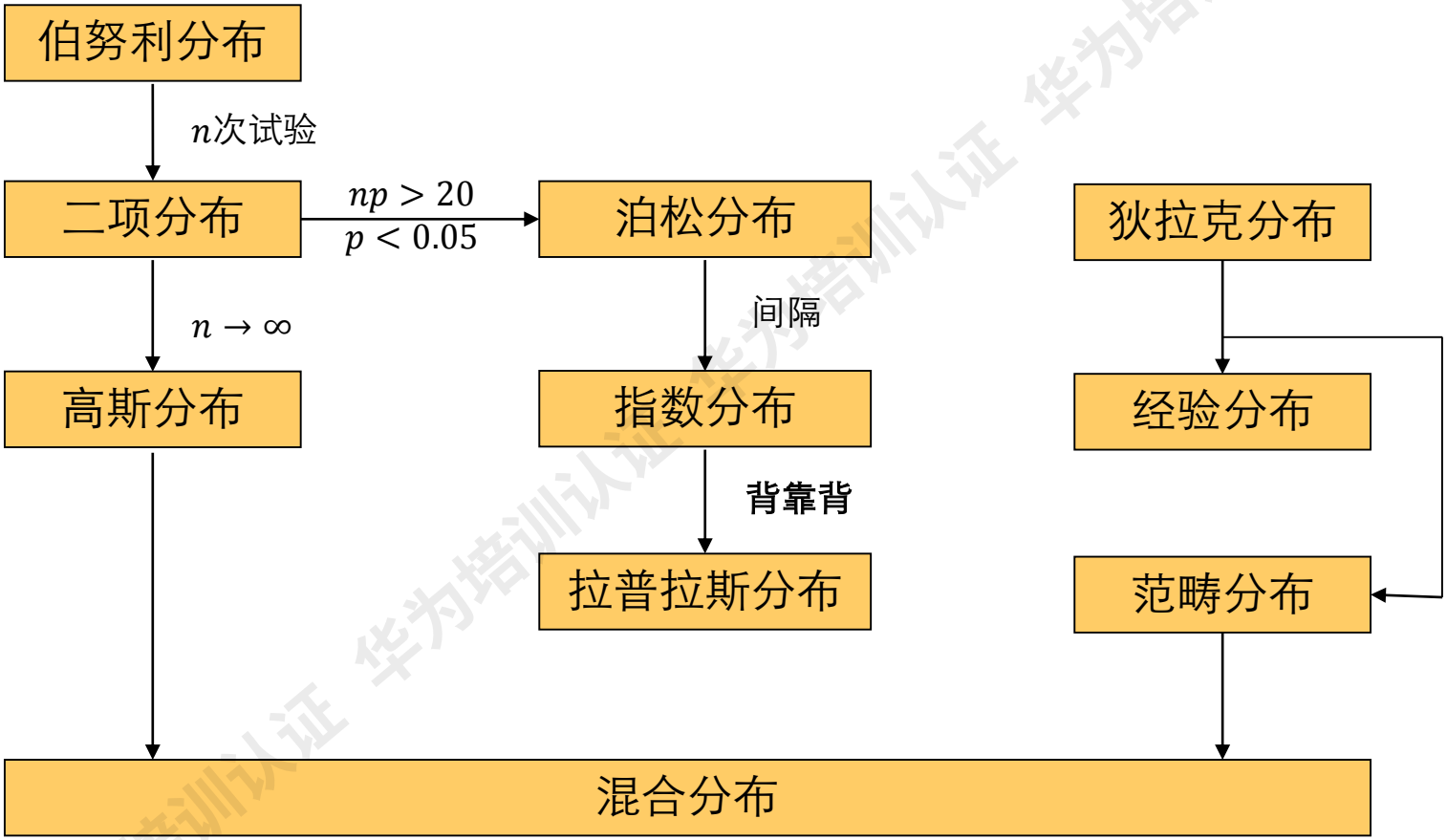
- 若连续型随机变量X的概率密度为

$$Laplace(x;\mu,\ b)=\frac{1}{2b}e^{-\frac{|x-\mu|}{b}},$$

其中μ是位置参数，b是尺度参数。则称X服从Laplace分布，记为$X\sim Laplace(x;\mu,\ b).E(X)=\mu,\ Var(X)=2b^2.$



概率分布小结



HUAWEI

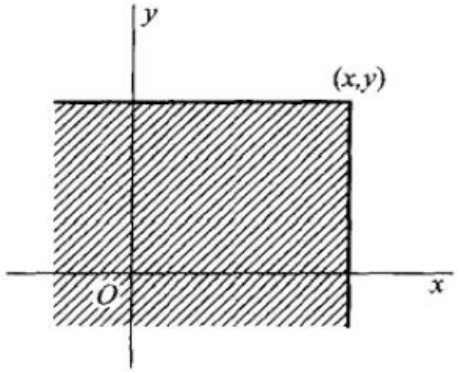
二维随机变量与联合分布函数

- **二维随机变量**：设E是一个随机试验，它的样本空间是$S=\{e\}$，设$X=X(e)$和$Y=Y(e)$是定义在S上的随机变量，由它们构成的一个向量$(X，Y)$，叫做**二维随机变量**。
- **二维随机变量的分布函数**：设$(X，Y)$是二维随机变量，对于任意实数x，y，二元函数：

$$F(x，y)=P\{(X\le x)\cap(Y\le y)\}=P\{X\le x，Y\le y\}$$

称为二维随机变量$(X，Y)$的分布函数，或称为随机变量X和Y的联合分布函数。

- **联合分布函数的意义**：如果将$(X，Y)$看成是平面上随机点的坐标，那么分布函数$F(x，y)$在$(x，y)$处的函数就是随机点$(X，Y)$落在以点$(x，y)$为顶点而位于该点左下方的无穷矩形域内的概率。



二维离散型随机变量与联合分布律

- 二维离散型随机变量：离散型随机变量$(X，Y)$全部可能取到的值是有限对或可列无限多对。
- X和Y的联合分布律：

Y \ X	x_1	x_2	…	x_i	…
y_1	p_{11}	p_{12}	…	p_{1i}	…
y_2	p_{21}	p_{22}	…	p_{2i}	…
⋮	⋮	⋮		⋮	
y_j	p_{j1}	p_{j2}	…	p_{ji}	…
⋮	⋮	⋮		⋮	

二维连续变量与联合概率密度

- 二维随机变量$(X，Y)$的分布函数$F(x，y)$，如果存在非负的函数$f(x，y)$使对于任意$x，y$有

$$F(x，y)=\int_{-\infty}^{y}\int_{-\infty}^{x}f(u，v)dudv，$$

则称$(X，Y)$是连续型的二维随机变量，函数$f(x，y)$称为二维随机变量X和Y的联合概率密度。

边缘分布

- **边缘分布函数**：二维随机变量$(X，Y)$作为一个整体，具有分布函数$F(x，y)$。而X和Y都是随机变量，各自也有分布函数，将它们分别记为$F_X(x)$，$F_Y(x)$依次称为二维随机变量$(X，Y)$关于X和关于Y的边缘分布函数。其中$F_X(x)=P\{X\le x\}=P\{X\le x，Y\le\infty\}=F(x，\infty).$
  - 对于离散型随机变量：
    - 边缘分布函数：$F_X(x)=\sum_{x_i\le x}\sum_{j=1}^{\infty}p_{ij}$.
    - 边缘密度函数：$p_{i.}=\sum_{j=1}^{\infty}p_{ij}$，$j=1，2，\cdots.$
  - 对于连续型随机变量：
    - 边缘分布函数：$F_X(x)=F(x，\infty)=\int_{-\infty}^{x}[\int_{-\infty}^{+\infty}f(x，y)dy]dx.$
    - 边缘概率密度：$f_X(x)=\int_{-\infty}^{+\infty}f(x，y)dy.$

条件概率、贝叶斯公式

- 在很多情况下，我们感兴趣的是某个事件在给定其他事件发生时出现的概率，这种概率叫做**条件概率**：

$$P(Y|X) = \frac{P(YX)}{P(X)}$$

- 我们经常需要在已知$P(Y|X)$的情况下计算$P(X|Y)$，此时若还知道$P(X)$，我们可以用**贝叶斯公式**来计算：

$$P(X|Y) = \frac{P(XY)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

- 假设X是由相互独立的事件组成的概率空间$\{X_1, X_2, \ldots, X_n\}$，则$P(Y)$可以用**全概率公式**展开：$P(Y) = P(Y|X_1)P(X_1) + P(Y|X_2)P(X_2) + \cdots + P(Y|X_n)P(X_n)$，此时**贝叶斯公式**可表示为：

$$P(X_i|Y) = \frac{P(Y|X_i)P(X_i)}{\sum_{i=1}^{n} P(Y|X_i)P(X_i)}$$

- 条件概率的链式法则：

$$P(X_1, X_2, \ldots, X_n) = P(X_1)\prod_{i=2}^{n} P(X_i|X_1, \ldots, X_{i-1})$$

独立性和条件独立

- 两个随机变量X和Y，若对于所有x，y有

$$P(X = x,\ Y = y) = P(X = x)P(Y = y),$$

则称随机变量X和Y是**相互独立的**，记作$X \perp Y$。

- 如果关于X和Y的条件概率对于Z的每一个值有

$$P(X = x,\ Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z),$$

则称随机变量X和Y在给定随机变量Z时是**条件独立的**，记作$X \perp Y|Z$。

贝叶斯规则举例

- 王某去医院作验血化验，其结果居然为阳性，检查他可能患上了X疾病。据网上资料显示，在得病的人中做实验，有1%的人是假阳性，99%的人是真阳性。而在未得病的人中做实验，有1%的人是假阴性，99%的人是真阴性。于是，王某想，既然只有1%的假阳性率，99%都是真阳性，那我已被感染X病的概率便应该是99%。可是，医生却告诉他，他被感染的概率只有0.09左右。

$X = 1$患病，$X = 0$未患病，$y = 1$检查阳性，$y = 0$检查阴性：

$$P(X = 1|y = 1) = \frac{P(X = 1)P(y = 1|X = 1)}{P(y = 1|X = 1)P(X = 1) + P(y = 1|X = 0)P(X = 0)}$$

$$= \frac{P(X = 1) \times 0.99}{0.99 \times P(X = 1) + 0.01 \times (1 - P(X = 1))}$$

若$P(X = 1) = 0.001$，$P(X = 1，\ y = 1) = 0.09$.

目录

期望、方差

- **数学期望（或均值，亦简称期望）**:是试验中每次可能结果的概率乘以其结果的总和，是最基本的数学特征之一。它反映随机变量平均取值的大小。
  - 对于离散型随机变量：$E(X)=\sum_{k=1}^{\infty}x_k\,p_k$，$k=1$，$2$，$\cdots$.
  - 对于连续型随机变量：$E(X)=\int_{-\infty}^{\infty}xf(x)dx$.
- **方差：**是在概率论和统计方差衡量随机变量或一组数据时离散程度的度量。概率论中方差用来度量随机变量和其数学期望之间的偏离程度。

$$D(X)=Var(X)=E\{[X-E(X)]^2\}$$

另外，$\sqrt{D(X)}$，记为$\sigma(X)$，称为标准差或均方差。$X^*=\frac{X-E(X)}{\sigma(X)}$称为$X$的标准化变量。

协方差、相关系数、协方差矩阵

- **协方差**：在某种意义上给出了两个变量线性相关性的强度以及这些变量的尺度。

$$Cov(X，Y) = E\big(X - E(X)\big)E\big(Y - E(Y)\big).$$

- **相关系数**又叫线性相关系数，用来度量两个变量间的线性关系。

$$\rho_{XY} = \frac{Cov(X，Y)}{\sqrt{D(X)}\sqrt{D(Y)}}.$$

- 随机变量$(X_1，X_2)$的**协方差矩阵**：

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

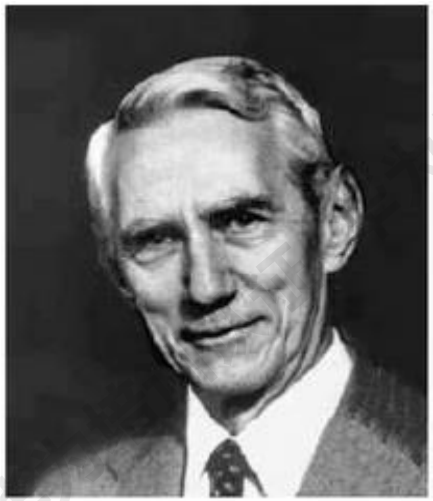
其中，$c_{ij} = Cov\big(X_i，X_j\big) = E\{[X_i - E(X_i)]\big[X_j - E\big(X_j\big)\big]\}$，$i，j = 1，2，\cdots，n.$

目录

HUAWEI

信息论

- **信息论**是应用数学的一个分支，主要研究的是对一个信号包含信息的多少进行量化。信息论创立的标志是1948年香农发表论文“A Mathematical Theory of Communication”。在这篇文章中香农创造性的采用概率论的方法来研究通信中的问题，并且对信息给予了科学的定量描述，第一次提出了**信息熵**的概念。



信息量

- 信息论的基本想法是一个不太可能的事情居然发生了，要比一个非常可能的事件发生，能提供更多的信息。消息说：“今天早上太阳升起”，信息量太少以至于没必要发送；但一条消息说：“今天早上有日食”，信息量就很丰富。那么定义一个事件$X = x$的**自信息**为$I(x)$应满足以下条件：
  - $f(p)$应是概率p的严格单调递减函数，即当$p_1 > p_2$，$f(p_1) < f(p_2)$；
  - 当$p = 1$时，$f(p) = 0$.
  - 当$p = 0$时，$f(p) = \infty$.
  - 两个独立事件的联合信息量应等于它们分别的信息量之和。

  因此若一个消息出现的概率为p，则这一消息所含的**信息量**为：

$$I(x) = -\log_2 p.$$

  例：抛一枚均匀硬币，出现正面与反面的信息量为：$I(\text{正}) = I(\text{反}) = 1bit$.

信息熵

- 信源含有的信息量是信源发出的所有可能消息的平均不确定性。信息论创始人香农把信源所含有的信息量称为**信息熵(Entropy)**，是指数据分区D所含信息量的**统计平均值**。对D中有m个元组分类的信息熵计算如下：

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i).$$

其中，p_i是D中任意一个元组属于类C_i的非零概率，$p_i = \frac{|C_{i,D}|}{|D|}$.

- 例如：抛一枚均匀硬币的信息熵是多少?

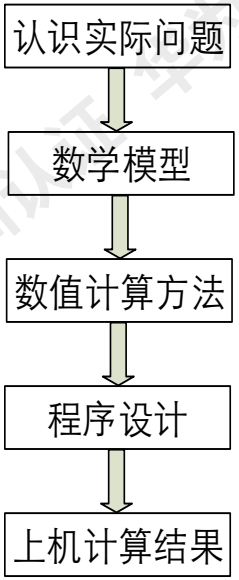
$$Info(D) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1bit.$$

目录

HUAWEI

数值计算

- **数值计算:**指有效使用数字计算机求数学问题近似解的方法与过程，以及由相关理论构成的学科。其中用计算机解决实际问题的过程如下:



上溢和下溢

- **下溢**：当接近零的数被四舍五入为零时发生下溢。许多函数在其参数为零而不是一个很小的正数时才会表现出质的不同。
- **上溢**：当大量级的数被近似为∞或−∞时发生上溢。进一步的运算通常会导致这些无限值变为非数字。
- **大数吃小数**：当$a \gg b$时，$a + b = a$，发生数值异常。
- ***Softmax*函数**可以对上溢和下溢进行**数值稳定**：

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}.$$

病态条件数

- **病态条件数**：条件数指的是函数相对于输入的微小变化而变化的快慢程度。
- 考虑函数$f(x) = A^{-1}x$，当$A \in \mathbb{R}^{n\times n}$具有特征分解时，其条件数为

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|,$$

这是最大和最小特征值的模之比。当该数很大时，矩阵求逆对输入的误差特别敏感。

这种敏感性是矩阵本身的固有特性，而不是矩阵求逆期间舍入误差的结果。即使我们乘以完全正确的矩阵逆，病态条件的矩阵也会放大预先存在的误差。在实践中，该错误将与求逆过程本身的数值误差进一步复合。

目录

HUAWEI

最优化问题

- **最优化问题**：指的是改变x以最小化或最大化某个函数$f(x)$的任务。可以表示为

$$min(max)\, f(x) \qquad \text{目标函数的极小（极大）}$$

$$s.t.\ \ g_i(x) \geq 0, i = 1，\ 2，\ \cdots，\ m， \qquad \text{不等式约束}$$

$$h_j(x) = 0，\ j = 1，\ 2，\ \cdots，\ p， \qquad \text{等式约束}$$

其中$x = (x_1，\ x_2，\cdots，\ x_n)^T \in R^n$，我们将$f(x)$称为目标函数或准则，当对其进行最小化时，也把它称为**代价函数**、**损失函数**或**误差函数**。

最优化问题的分类 (1)

- **约束优化**：是优化问题的分支。有时候，在x的所有可能值下最大化或最小化一个函数$f(x)$不是我们所希望的。相反，我们可能希望在x的某些集合s中找$f(x)$的最大值或最小值。集合s内的点称为**可行点**。
- **无约束条件**，可以写为

$$minf(x),$$

常用方法为Fermat定理。即令$f'(x)$=0，求得临界点。然后验证临界点是否取得极值。

- **等式约束条件**，可以写为

$$minf(x)$$
$$s.t.\ \ h_i(x)=0,\ i=1,\ 2,\ \cdots,\ n.$$

常用方法为拉格朗日乘子法，即引入n个拉格朗日乘子λ，构造拉格朗日函数$L(x,\ \lambda)=f(x)+\sum_{i=1}^{n}\lambda_i h_i(x)$，然后对各个变量求偏导，令其为零，可以求得候选值集合，然后验证求得最优值。

最优化问题的分类 (2)

- **不等式约束条件**，可以写为

$$minf(x)$$
$$s.t.\ \ h_i(x)=0,\ \ i=1,\ \ 2,\ \ \cdots,\ \ n,$$
$$g_j(x)\le 0,\ \ j=1,\ \ 2,\ \ \cdots,\ \ m.$$

通常方法为引入新的变量λ_i和α_j，将所有的等式、不等式约束以及$f(x)$构造成**广义拉格朗日函数**，即

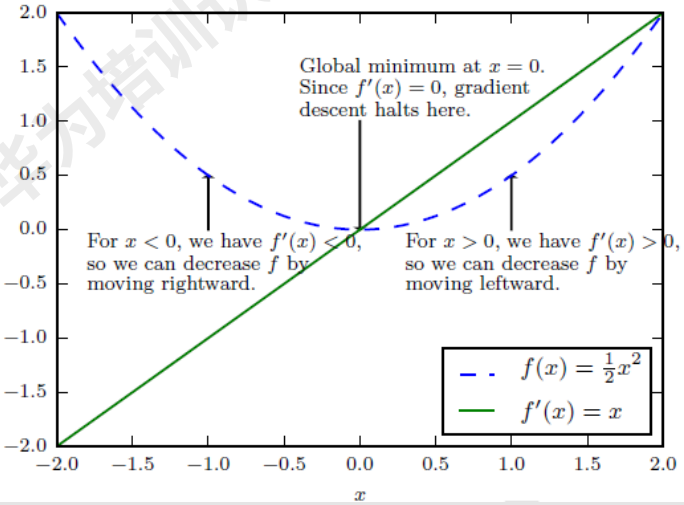
$$L(x,\ \lambda,\ \alpha)=f(x)+\sum_i \lambda_i\, h_i(x)+\sum \alpha_j\, g_j(x),$$

我们可以使用一组简单的性质来描述约束优化问题的最优点，这些性质称为**KKT（Kuhn-Kuhn-Tucker）条件**。具体如下：

- 广义拉格朗日的梯度为0。
- 所有关于x和KKT乘子的约束都满足。
- 不等式约束显示的“互补松弛型”：$\alpha \odot h(x)=0.$

基于梯度的优化方法 (1)

- **梯度下降**：导数告诉我们如何更改x来略微地改善y。例如，我们知道对于足够小的Δx来说，$f(x-\Delta x sign(f'(x)))$是比$f(x)$小的。因此我们可以将$x$往导数的反方向移动一小步来减小$f(x)$，这种技术称为梯度下降。
- 一维函数的极值问题：
  - 函数的局部极值点意味着不能通过移动x减小或增大$f(x)$。
  - $f'(x)=0$的点称为临界点或驻点。
  - 函数的极值点一定是驻点，反之未必。



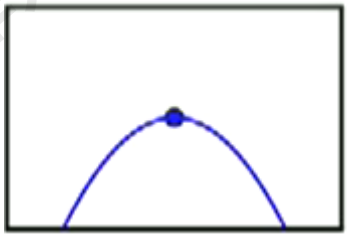
基于梯度的优化方法 (2)

- 凸函数：对于$\lambda \in (0, 1)$，任意x_1，$x_2 \in R$，都有

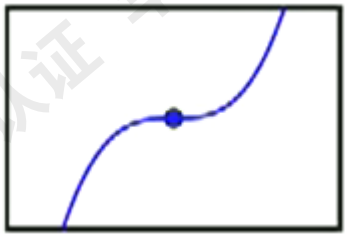
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2),$$

则称$f(x)$是一个凸函数。凸函数的极值点出现在驻点处。

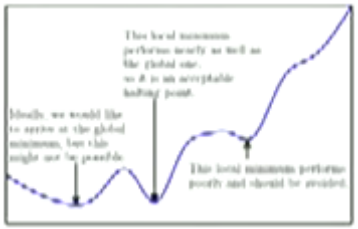
(a)



(b)



(c)



(d)

基于梯度的优化方法 (3)

- 推广至多维函数的情形，用偏导数描述函数相对于各自变量的变化程度。
- **梯度（gradient）**：是相对于一个向量X的导数，符号一般记为$\nabla_x f(x)$，$f(x)$在方向u（单位向量）的方向导数即是$u^T \nabla_x f(x)$.
- 对于一个最小化 $f(x)$ 的任务，我们希望找到其向下变化最快的方向，其中θ为u与梯度$\nabla_x f(x)$的夹角。

$$\min_{u,u^T u=1} u^T \nabla_x f(x)$$

$$= \min_{u,u^T u=1} \|u\|_2 \|\nabla_x f(x)\|_2 \cos\theta$$

可见，$f(x)$数值减小最大的方向就是梯度的负方向。

基于梯度的优化方法 (4)

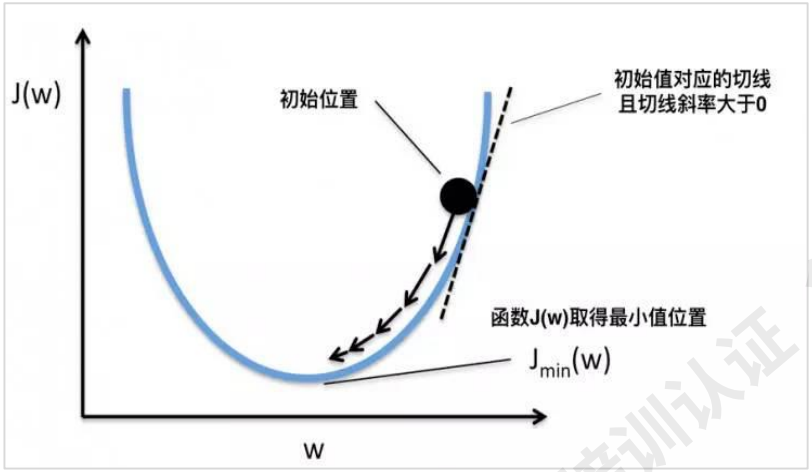
- 正梯度向量指向上坡，负梯度向量指向下坡。我们在负梯度方向上移动可以减小$f(x)$，这被称为**最速下降法**（method of steepest descent）或**梯度下降**（gradient descent）。
- 在梯度下降法下，更新点被建议为：

$$x' = x - \varepsilon \nabla_x f(x),$$

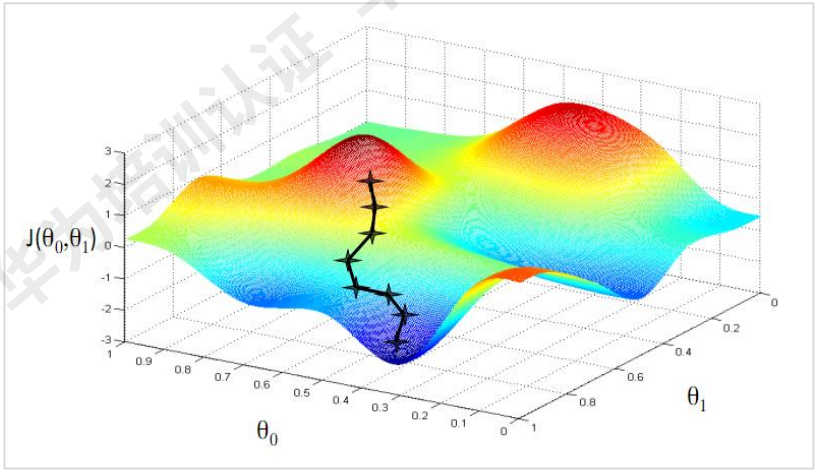
其中 ε 为学习率（learning rate），是一个确定步长的正标量。

- 迭代在梯度为零或趋近于零的时候收敛。

基于梯度的优化方法 (5)



二维空间



多维空间

思考题

1. 随机变量的分布函数、分布律、密度函数有什么联系和区别?

思考题

1. (单选) 三个矩阵A、B、C的行列数分别是3行2列、2行3列、3行3列，则下列哪个运算有意义？（　）

   A. AC

   B. BC

   C. A+B

   D. AB-BC

2. (判断题) 主成分分析（Principal Component Analysis，PCA）是一种统计方法。通过正交变换将一组可能存在相关性的变量转换为一组线性相关的变量，转换后的这组变量叫主成分。（　）

   A. True

   B. False

思考题

3. (单选) X、Y是随机变量，C是常数，下列选项中对数学期望的性质描述不正确的是？（　）

A. E(C)=C

B. E(X+Y)=E(X)+E(Y)

C. E(CX)=CE(X)

D. E(XY)=E(X)E(Y)

4. (判断题) 相关系数又叫线性相关系数，用来度量两个变量间的线性关系，其值为大于0的实数。（　）

A. True

B. False

本章总结

- 本章主要介绍了深度学习基础知识，其中包括线性代数基础知识、概率与信息论基础知识、数值计算基础知识，为后续学习奠定基础。

学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

TensorFlow介绍

www.huawei.com

HUAWEI

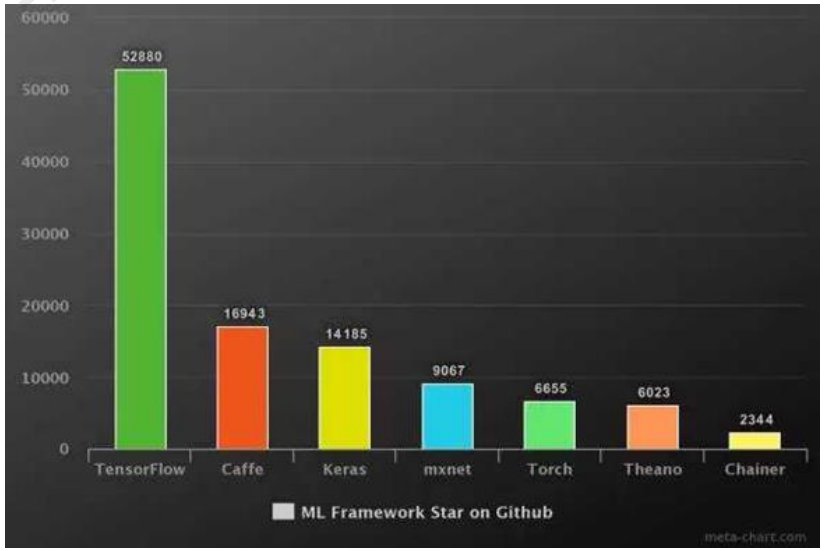
目标

- 学完本课程后，您将能够：
  - 了解TensorFlow是什么及其特点
  - 掌握TensorFlow基础知识及搭建环境的方法
  - 熟悉TensorFlow各个模块
  - 掌握TensorFlow的开发基本步骤
  - 熟悉其他深度学习框架

目录

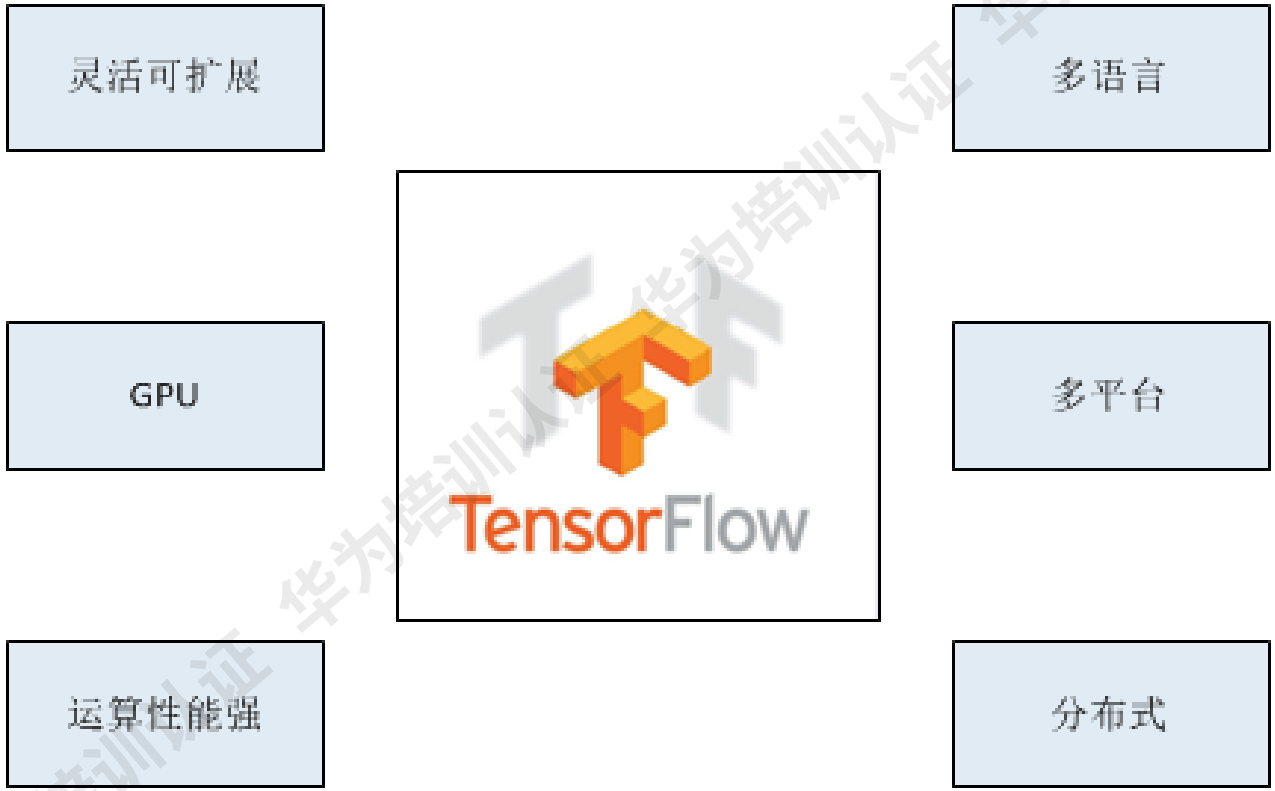
TensorFlow是什么

- TensorFlow是谷歌开源的第二代用于数字计算的软件库。
- TensorFlow计算框架可以很好的支持深度学习的各种算法，但它的应用不局限于深度学习，可以支持多种计算平台，系统稳定性较高。
- 源于TensorFlow的开源性，方便大家维护更新，提高开发效率。



目录

TensorFlow特点



TensorFlow可以做什么 (1)

- 自动驾驶小车
- 生成音乐
- 图像识别
- 语音识别
- 语言模型
- 人体行为识别
- 定理证明
- 拿来玩马里奥赛车…

TensorFlow可以做什么 (2)

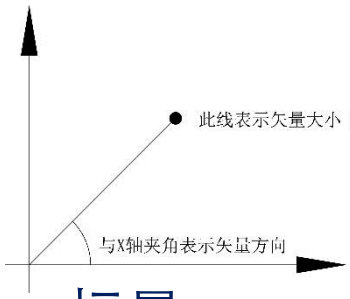
图像风格转换

人脸识别

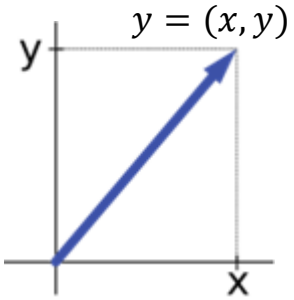
HUAWEI

目录

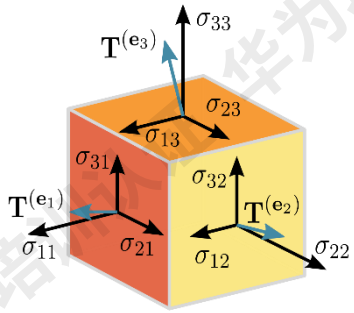
TensorFlow



标量



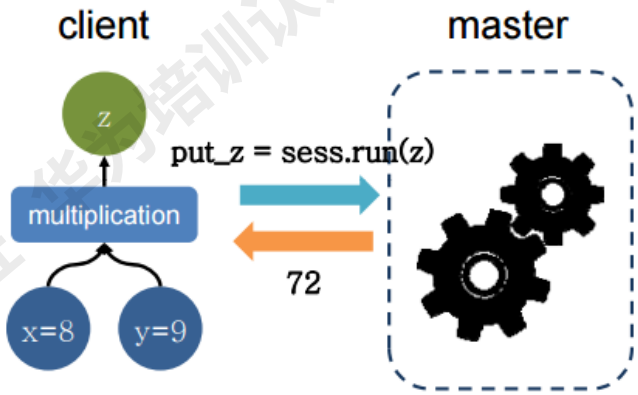
矢量



张量

Tensor

+



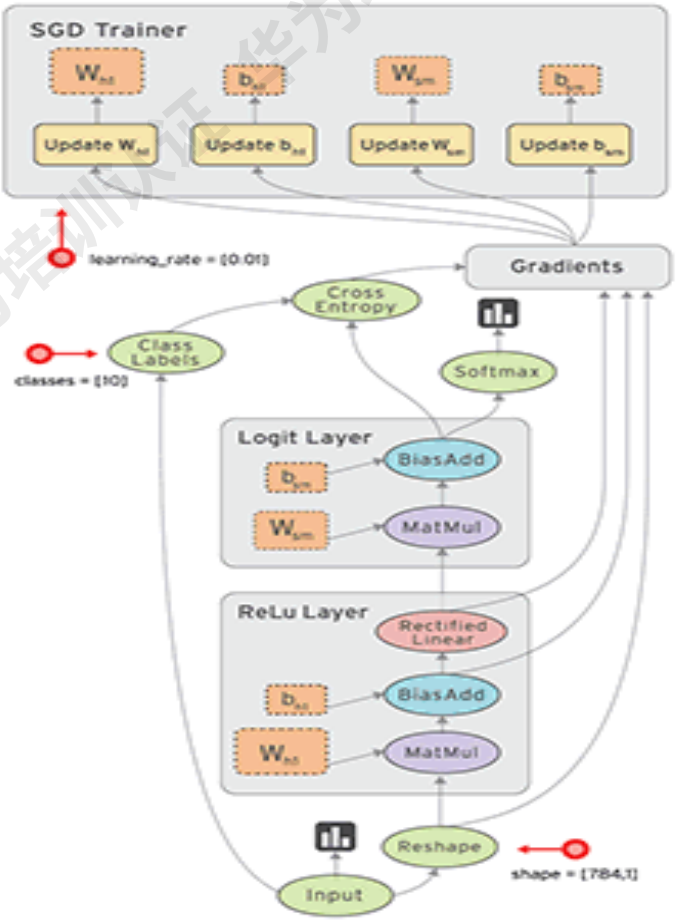
数据流图

Flow

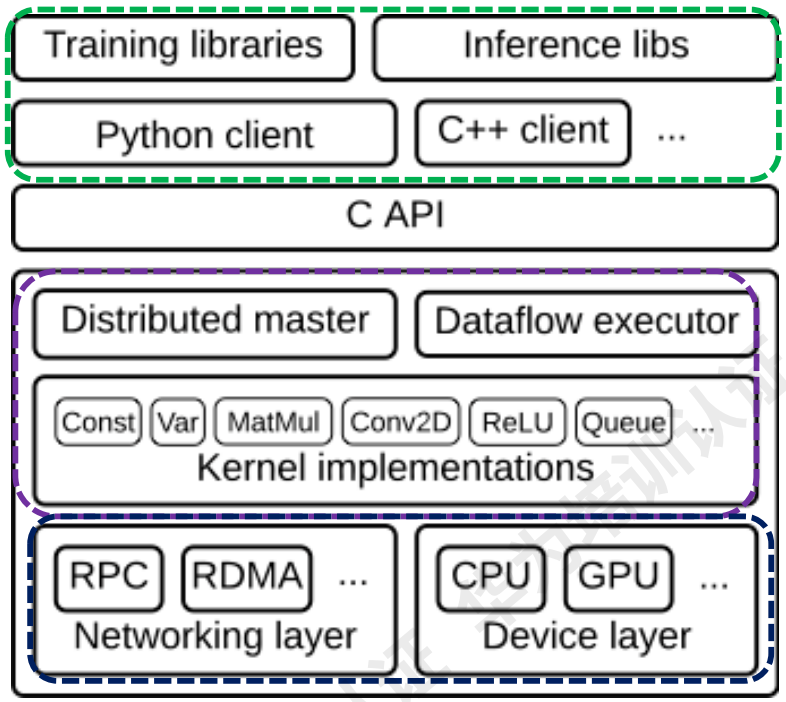
HUAWEI

TensorFlow计算过程

- TensorFlow是一个通过计算图的形式来表述计算的编程系统，是以张量（tensor）在计算图（graph）上流动（flow）的方式实现和执行机器学习/深度学习算法的框架。



TensorFlow架构



- 多种前端语言可供选择，Pyhton，C++，Go等
- 在此基础上，可灵活构建训练和推理库

→ 隔离用户态代码和TF核心代码

- TF核心库使用C++实现，支持几乎所有主流操作系统，Linux，Mac OS X，Windows，Android和iOS
- TF实现了统一的抽象层调用kernelOp，隔离异构代码，同一种Op支持CUDA，OpenCL等多种异构实现
- TF支持多种异构加速平台，如GPU，TPU

→ 设备驱动

逐层分解，高度解耦

TensorFlow基础概念

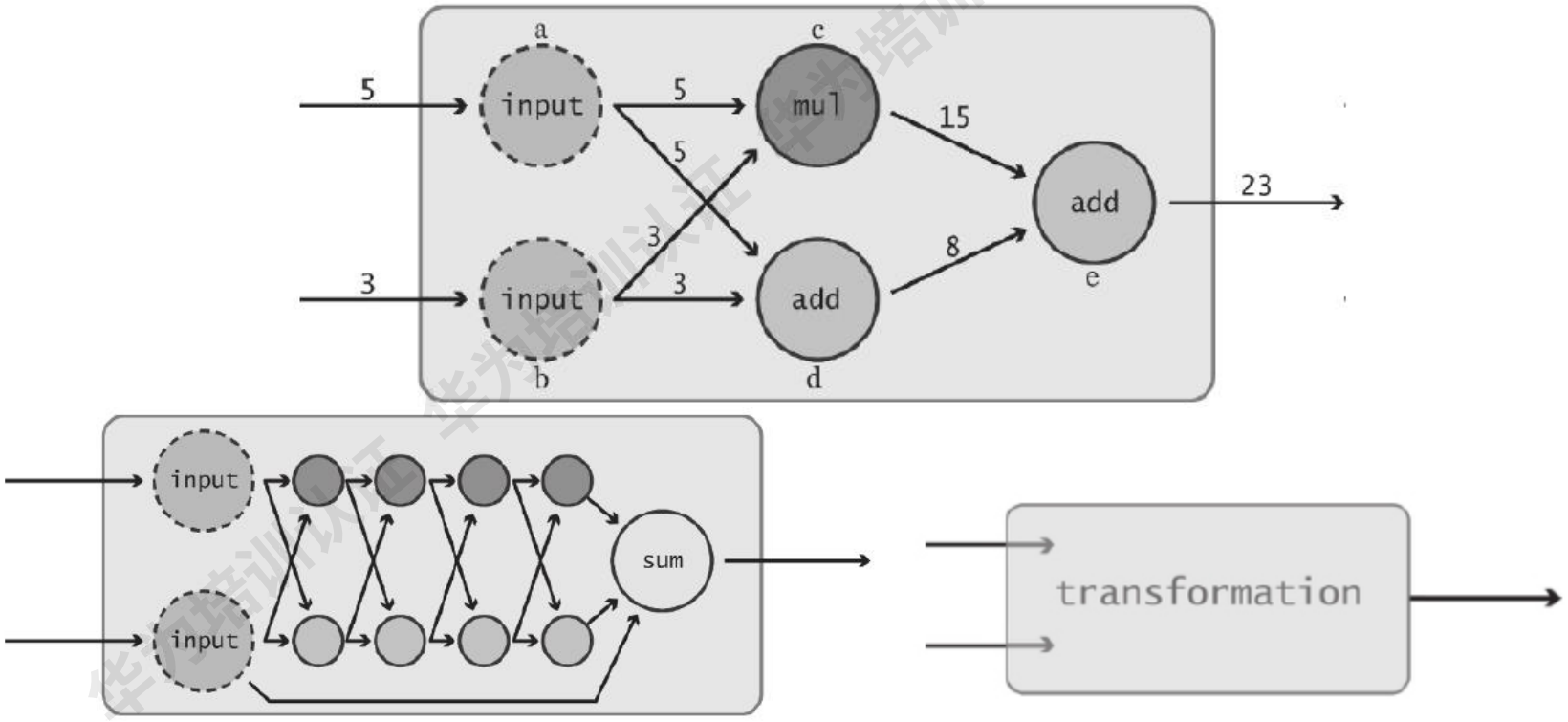
- 张量(tensor)
- 计算图（graph）
  - 节点（node）
  - 边(edge)
- 算子(operation)
- 会话(session)
  - feed
  - fetch
- 变量（variable）

张量 (tensor)

- tensor是数据的载体。tensor可以是任意维度的数据，一维、二维、三维、四维等数据统称为张量。在一个运行的图(graph)中，它是一种流动在节点（node）之间的数据。
- tensor中保存的是张量的数据结构即（name+shape+type)。

计算图 (数据流图)

- node(节点)：对数据的运算或操作，OP。
- edge(边）：数据及其依赖关系。

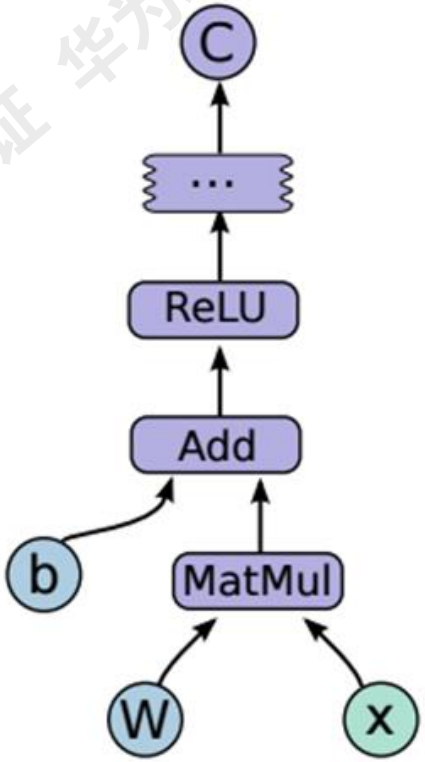


TensorFlow算子

Category	Examples
Element——wise mathematical Operations	Add,Sub,Div,Exp,Log,Greater,Less,Equal,…
Array operations	Concat,Slice,Split,Constant,Rank,Shape,Shuffle,…
Matrix operations	MatMul,MatrixInverse,MatrixDeterminant,…
Stateful operations	Variable,Assign,AssignAdd,…
Neural-net building blocks	SoftMax,Singmoid,ReLU,Convolution2D,MAxPool,…
Checkpointing operations	Save,Restore,…
Queue and synchronization operations	Enqueue,Dequeue,Mutex Acquire,Mutex Release,…
Control flow operations	Merge,Switch,Enter,Leave,NextIteration

会话 (Session)

- Session拥有并管理TensorFlow程序运行时的所有资源。
- 客户端使用会话来和TF系统交互，一般的模式是，建立会话，此时会生成一张空图；在会话中添加节点和边，形成一张图，然后执行。



Feed注入机制

- 把tensor直接连接到图中的一个节点， 临时替换该节点tensor的值。Feed不是在构建图的时候创建，而是在触发图执行的时候去申请，也就是在run或是eval的时候把“feed数据” 以参数的形式来初始化。方法运行结束后，替换的“feed数据”消失，但是图中定义的节点的行为不发生改变。一般与tf.placeholder( )结合使用。

Fetch取回机制

- 取回图中一个操作的结果。取回申请发生在触发执行图的时候，而不是发生在构建图的时候。如果要取回一个或多个节点（node）的 tensor值，可以通过在 Session 对象上调用run( )方法并将待取回节点（node）的列表作为参数来执行图表(graph)。

变量 (Variable)

- 变量维护图执行过程中的状态信息。如在神经网络中，用于标识w、b等系数。变量其实是Python中的Variable对象。
- TensorFlow中变量的初始值可以设置成随机数、常数或者是通过其他变量的初始值计算得到。

目录

TensorFlow模块介绍 (1)

- tf.app:通用入口点脚本。
- tf.nn:支持神经网络。
- tf.bitwise:用于操纵整数的二进制表示的操作。
- tf.compat:与 Python 2和Python 3 具有兼容性的函数。
- tf.contrib:较为复杂，与蒙特卡洛积分，条件随机场。构建层、正则化、初始化、优化、Feature columns有关联。
- tf.estimator（估算器）:用于处理模型的高级工具。
- tf.errors:TensorFlow 错误的异常类型。

TensorFlow模块介绍 (2)

- tf.nn:支持神经网络。
  - 用于构建RNN的模块。最常用到的一个模块，用于构建经典的卷积网络，它下面还包含了 rnn_cell 的子模块，用于构建循环神经网络。
- tf.estimator:
  - 创建一个或多个输入函数。
  - 定义模型的特征列。
  - 实例化一个 Estimator，定义特征列和各类超参数。
  - 在 Estimator 对象上调用一个或多个方法，传入合适的输入函数来作为数据源。

TensorFlow模块介绍 (3)

- tf.layers:网络层封装了变量和作用其上的操作。
  - 比如（全连接层)对输入进行一个加权求和的操作，并且可以作用一个可选择的激活函数。连接的权重和偏置都由网络层对象来管理。
- tf.contrib:该模块提供计算流式指标的函数。
  - 这个模块最常用到的是它的 slim 子模块，所有的易于变动的，或者说实验性质的功能就放在这个模块里面，有着及其丰富的功能子模块。
  - 以动态价值计算的指标 tensors。每个指标声明返回一个“value_tensor”，一个返回指标当前值的幂等运算，还有一个“update_op”，这是一个从当前tensors 测量值累加信息的操作，并返回 “ value_tensor”。

目录

Windows搭建TensorFlow环境 (1)

- 操作系统：Windows/Mac/Linux
- Python 3 https://www.python.org/downloads/release/python-350/
- Anaconda 3（适配Python 3的版本）自带pip软件
- 安装TensorFlow
  - 在线安装nipghtly包（pip install tf-nightly）
  - 安装纯净版TensorFlow（pip install tensorflow）
  - 离线安装

Windows搭建TensorFlow环境 (2)

- 安装GPU版本：
  - 安装CUDA软件包（与TensorFlow版本对应）
  - 安装cuDNN库（与TensorFlow版本对应）
  - 测试显卡
  - 使用nvidia-smi查看显卡信息
  - 查看CUDA版本
  - Linux与Mac上的安装教程可以查阅网上的教程

Mac搭建TensorFlow环境

- 在 OS X 系统上，我们推荐先安装 homebrew，然后执行 brew install python，以便能够使用 homebrew 中的 Python 安装 TensorFlow。另外一种推荐的方式是在 virtualenv 中安装 TensorFlow。

```
# 当前版本只支持 CPU
$ pip install https://storage.googleapis.com/tensorflow/mac/tensorflow-0.5.0-py2-none-any.whl
```

Linux搭建TensorFlow环境

- 在 Linux 下最简单的安装方式, 是使用 pip 安装。

```
# 仅使用 CPU 的版本
$ pip install https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl

# 开启 GPU 支持的版本 (安装该版本的前提是已经安装了 CUDA sdk)
$ pip install https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl
```

TensorFlow主要依赖的工具包 (1)

- Protocol Buffer是TensorFlow依赖的主要工具包之一，它是Google开发的处理结构化数据的工具。
  - ProtoBuf序列化后的数据是二进制流。
  - 预定义数据的格式Schema.proto文件。
  - ProtoBuf数据为XML 10%-30%，解析时间快20-100倍。

```
name: 张三
id: 12345
email: zhangsan@abc.com
```

TensorFlow主要依赖的工具包 (2)

- Bazel（自动化构建工具）
  - 比传统的Makefile，Ant，Bazel速度快，可伸缩性强，灵活性好。
  - 基本概念是项目空间（workspace），简单理解为一个文件夹，包含了编译软件所需的源代码及输出结果的软编译地址。
  - 只支持py_binary，py_library，py_test三种编译方式。

```
-rw-rw-r--  root root 208    BUILD
-rw-rw-r--  root root 48     hello_lib.py
-rw-rw-r--  root root 47     hello_main.py
-rw-rw-r--  root root 0      WORKSPACE
```

运行TensorFlow

- 打开python终端，输入：

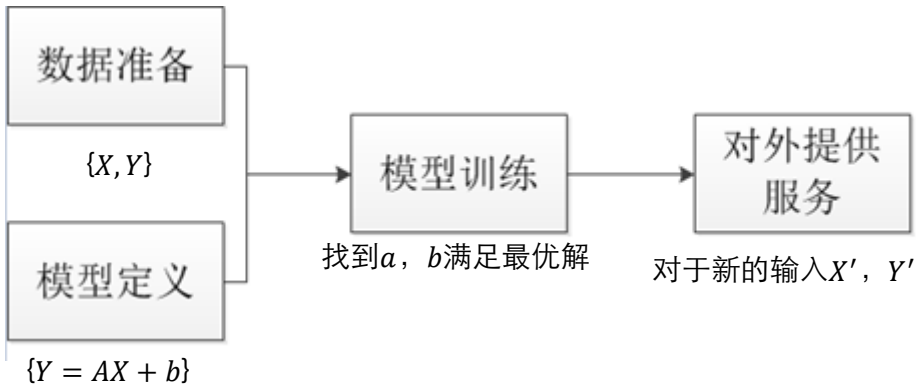
```
$ python

>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print sess.run(hello)
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print sess.run(a+b)
42
>>>
```

目录

TensorFlow开发基本介绍

- 定义输入节点。
- 定义“学习参数”的变量。
- 定义“运算”。
- 优化函数，优化目标。
- 初始化所有变量。
- 模型训练，迭代更新参数到最优解。
- 测试模型。
- 使用模型。

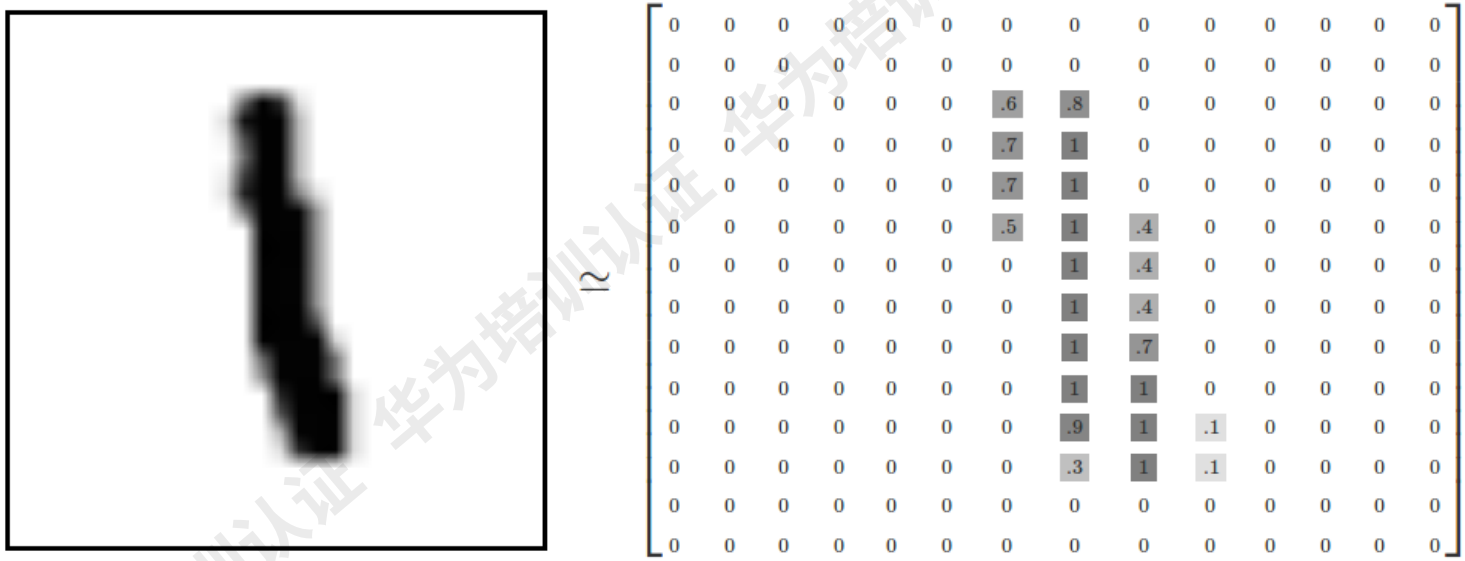


准备数据 (1)

- MNIST是在机器学习领域中的一个经典问题。该问题解决的是把28x28像素的灰度手写数字图片识别为相应的数字，其中数字的范围从0到9。
- 由于MNIST数据集是TensorFlow的示例数据，所以我们不必下载。

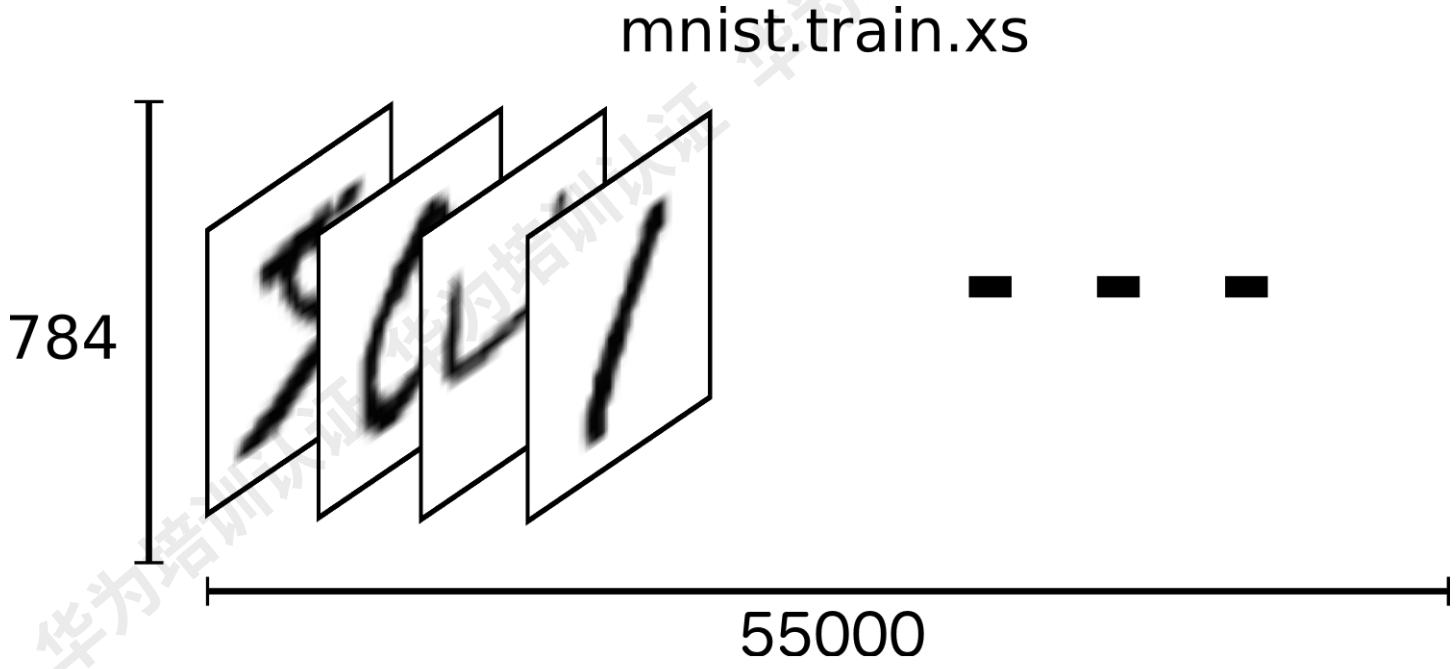
准备数据 (2)

- 一张图片是一个28*28的像素点矩阵，我们可以用一个同大小的二维整数矩阵来表示。



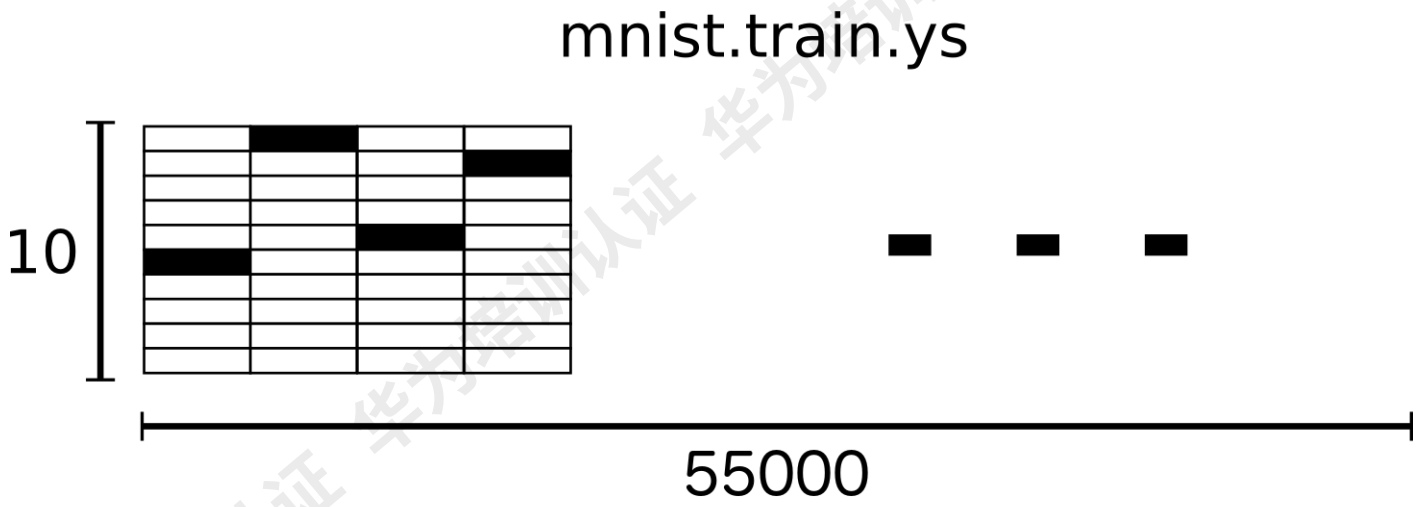
MNIST数据集 (1)

- MNIST的训练数据集可以是一个形状为55000 * 784位的tensor，也就是一个多维数组，第一维表示图片的索引，第二维表示图片中像素的索引（“tensor”中的像素值在0到1之间）。



MNIST数据集 (2)

- mnist.train.labels是一个55000 * 10的二维数组。如下：



定义输入节点

- TensorFlow有以下几种定义输入节点的方法：
  - 通过占位符定义（一般常用方式）
  - 通过字典类型定义（用于输入比较多的情况）
  - 直接定义（较少使用）
- 由于输入的图片是550000*784矩阵。所以创建一个[None,784]的占位符x和一个[None,10]的占位符y，使用feed机制将图片和标签输入。

定义学习参数

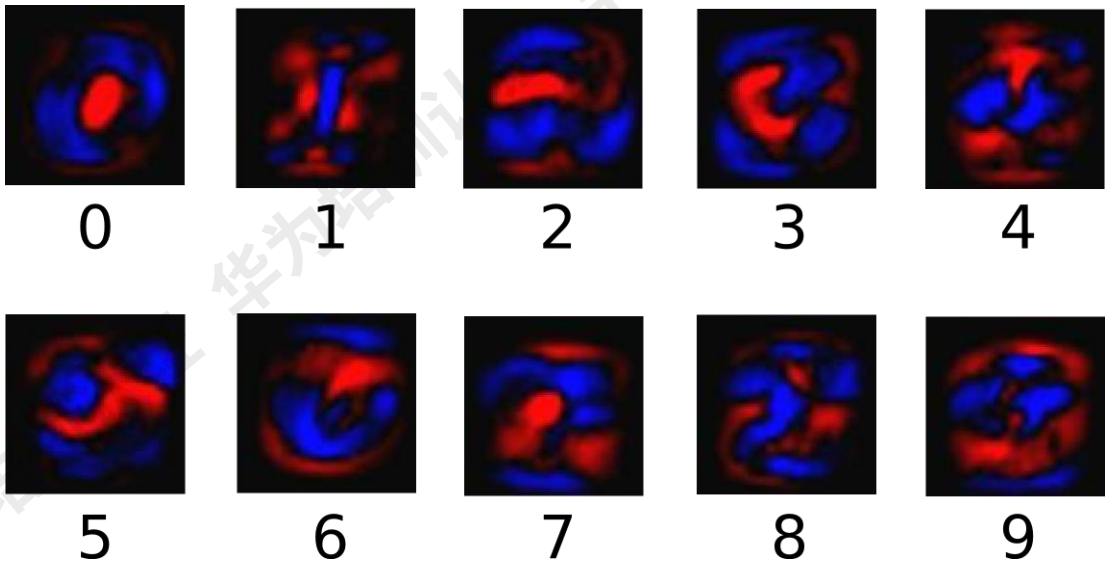
- 定义“学习参数”变量
  - 直接定义
  - 字典定义
- 学习参数包括权重值和偏置量。在TensorFlow中使用Variable定义学习参数。
- 一个Variable代表一个可修改的张量，定义在TensorFlow图中，使用Variable定义的学习参数可用于计算输入值，也可在计算中被修改。

定义“运算”

- 建立模型的核心过程，决定了模型的拟合效果。
- 定义运算的类型：
  - 定义正向传播模型。
  - 定义损失函数（计算输出值与目标值的误差，配合反向传播使用）TensorFlow常用的损失函数为均方误差和交叉熵。
  - 定义反向传播结构。

搭建模型

- softmax模型可以用来给不同的对象分配概率。即使在之后，我们训练更加精细的模型时，最后一步也需要用softmax来分配概率。
- 下面的图片显示了一个模型学习到的图片上每个像素对于特定数字类的权值。红色代表负数权值，蓝色代表正数权值。



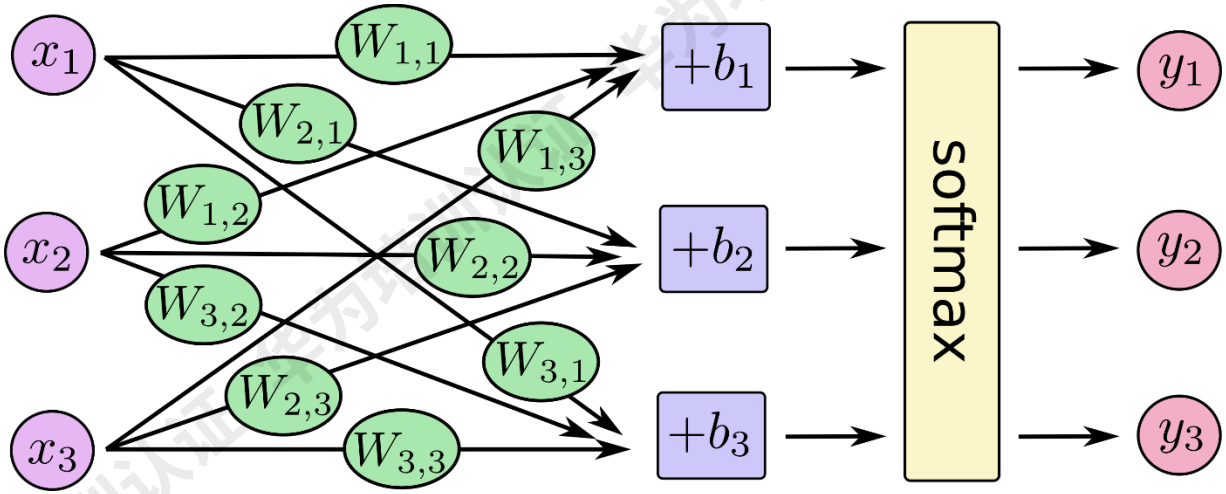
定义模型

- 因为输入往往会带有一些无关的干扰量。所以加入一个额外的偏置量（bias）。因此对于给定的输入图片x它代表的是数字i的证据可以表示为：$evidence_i = \sum_j W_{i,j} x_j + b_i$.
- 其中，W_i代表权重，b_i代表数字i类的偏置量，j代表给定图片 x的像素索引用于像素求和。然后用softmax函数可以把这些证据转换成概率y。

$$y = softmax(evidence)$$

回归模型 (1)

- 对于输入的x_i加权求和，再分别加上一个偏置量，最后再输入到softmax函数中：



回归模型 (2)

- 实际的计算中，我们通常采用矢量计算的方式，如下：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

更进一步，可以写成更加紧凑的方式：$y = softmax(W_x + b)$.

优化函数，优化目标

- 该过程在反向传播中完成，反向传播过程就是沿着正向传播的反方向将误差传递回去，涉及到L1，L2正则化，冲量调节，学习率自适应，Adam随机梯度下降算法。

初始化

- 创建模型，需要创建大量的权重和偏置项。模型中的权重在初始化时应该加入少量的噪声来打破对称性以及避免0梯度。
- 为了不在建立模型的时候反复做初始化操作，我们定义两个函数用于初始化。

```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```

实现模型

- 为了训练我们的模型，我们首先需要定义一个指标来评估这个模型是好的。其实，在机器学习，我们通常定义指标来表示一个模型是坏的，这个指标称为成本（cost）或损失（loss），然后尽量最小化这个指标。成本函数即“交叉熵”（cross-entropy）。

迭代训练模型到最优解

- TensorFlow用梯度下降算法以0.01的学习速率最小化交叉熵。梯度下降算法（gradient descent algorithm）也称为最速下降法，是一个简单的学习过程，TensorFlow只需将每个变量一点点地往使成本不断降低的方向移动，求解极小值，递归性的逼近最小偏差模型。
- 然后开始训练模型，让模型循环训练1000次！

```
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

测试模型

- 利用MNIST里的测试数据测试模型。找出那些预测正确的标签。由于标签向量是由0，1组成，因此最大值1所在的索引位置就是类别标签，为了确定正确预测项的比例，我们可以把布尔值转换成浮点数，然后取平均值。最后，我们计算所学习到的模型在测试数据集上面的正确率。
- tf.argmax(y,1)返回的是模型对于任一输入x预测到的标签值，而tf.argmax(y_,1)代表正确的标签，而用tf.equal来检测预测标签是否与真实标签相匹配。

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```

评估模型

- TensorFlow需要把模型评估加入到计算图中，然后在模型训练完后调用模型评估。
- 模型的评估主要有几个指标：平均准确率、识别的时间、loss下降变化等。
- 在训练模型过程中，模型评估能洞察模型算法，给出提示信息来调试、提高或者改变整个模型。训练模型之后，需要定量评估模型的性能如何。

使用模型 (1)

- 保存模型：
  - 建立一个saver和一个路径，调用save将session中参数保存。
- 使用模型：
  - 与测试模型类似，讲损失值的节点换成输出的节点。
- 读取模型：
  - 读取模型，将图片放到模型里进行预测，将图片及其对应的标签一并显示。

使用模型 (2)

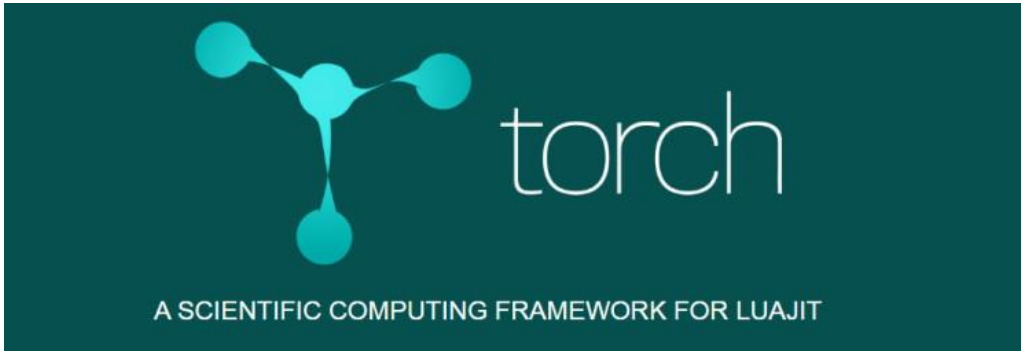
- 读取模型将两张图片放进去预测结果，将两个图片对应的标签显示出来。
- 代码执行过程中，对网络模型的定义不变。重新建立一个session。所有操作在新的session中完成。

```
After 0 training step(s), validation accuracy using average model is 0.103
After 1000 training step(s), validation accuracy using average model is 0.9044
After 2000 training step(s), validation accuracy using average model is 0.9174
After 3000 training step(s), validation accuracy using average model is 0.9258
After 4000 training step(s), validation accuracy using average model is 0.93
After 5000 training step(s), validation accuracy using average model is 0.9346
After 6000 training step(s), validation accuracy using average model is 0.94
After 7000 training step(s), validation accuracy using average model is 0.9422
After 8000 training step(s), validation accuracy using average model is 0.9472
After 9000 training step(s), validation accuracy using average model is 0.9498
After 10000 training step(s), test accuracy using average model is 0.9475
```

目录

其他深度学习框架

- Theano
- Torch
- Keras
- DeepLearning4j
- Caffe
- MXNet
- CNTK



深度学习框架硬件支持率

Property	Caffe	Neon	TensorFlow	Theano	Torch
Core	C++	Python	C++	Python	Lua
CPU	✓	✓	✓	✓	✓
Multi-threaded CPU	✓Blas	x Only data loader	✓Eigen	✓Blas, conv2D, limited OpenMP	**✓Widely used**
GPU	✓	✓customized Nvidia backend	✓	✓	✓
Multi-GPU	✓(only data parallel)	✓	**✓Most flexible**	x Experimental version available	✓
Nvidia cuDNN	✓	x	✓	✓	✓
Quick deploy. on standard models	**✓Easiest**	✓	✓	x Via secondary libraries	✓
Auto. gradient computation	✓	✓Supports Op-Tree	✓	**✓Most flexible** (also over loops)	✓

流行深度学习框架优缺点

- Caffe：
  - 优势：上手快，速度快，模块化，开放性，社区好。
  - 劣势：图层需要使用C++定义，而模型需要使用protobuf 定义。Caffe的配置文件不能用编程的方式调整超参数，仅支持单机多GPU的训练，没有原生支持分布式的训练。
- Torch：
  - 优势：构建模型简单，高度模块化，快速高效的GPU支持，通过LuaJIT接入C，数值优化程序等，可嵌入到iOS、Android和FPGA后端的接口 。
  - 劣势：某些接口不全面，需要LuaJIT支持用于Lua语言。通用性较差。

本章总结

- 本章主要介绍了TensorFlow的概念及相关特点，并介绍了TensorFlow基础知识，TensorFlow常用的模块。然后实战搭建开发环境并动手组建自己的TensorFlow程序。通过介绍TensorFlow的开发基本步骤，了解TensorFlow内部机制，最后介绍了其他深度框架。

思考题

1. TensorFlow的应用场景包括？（　　）

A. AlphaGo

B. 人脸识别

C. 图像风格变化

D. 自动驾驶

学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

深度学习预备知识和深度学习概览

www.huawei.com

HUAWEI

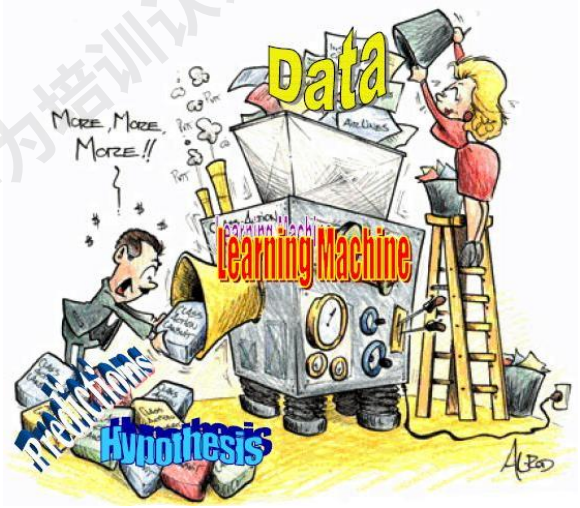
目标

- 学完本课程后，您将能够：
  - 了解学习算法及常用机器学习算法
  - 了解超参数与验证集的概念
  - 掌握最大似然估计与贝叶斯估计
  - 了解神经网络的定义与发展
  - 掌握神经网络的种类

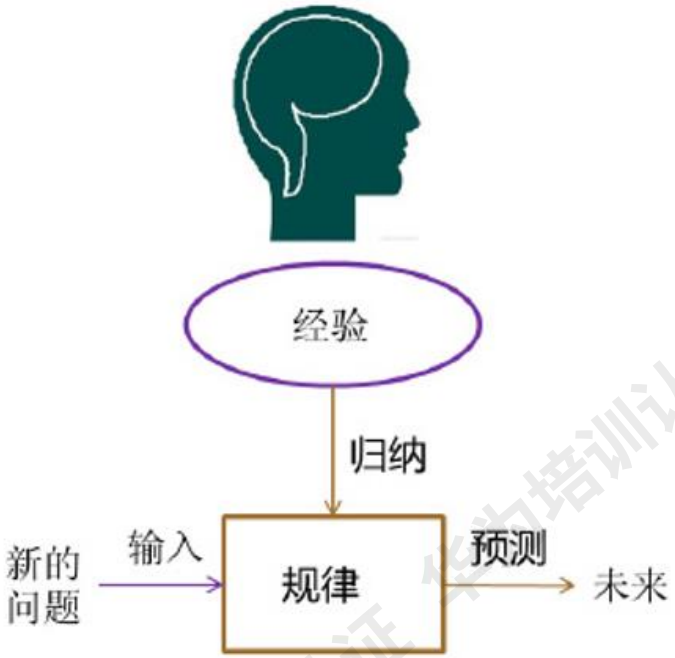
目录

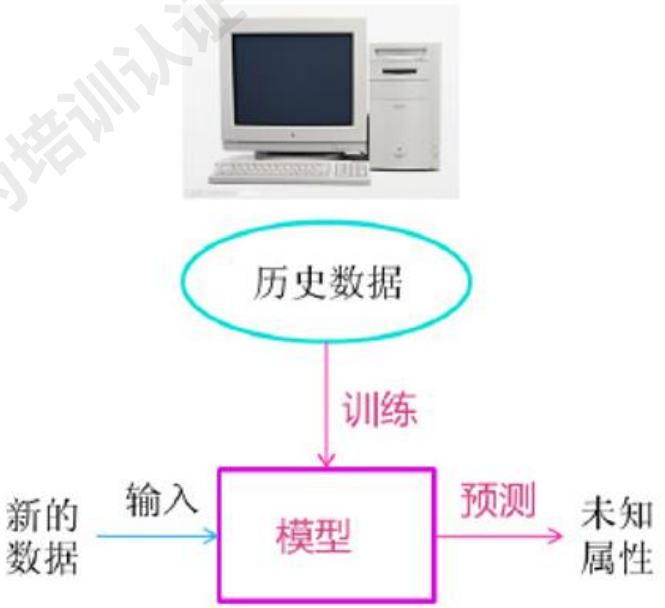
学习算法 (1)

- 机器学习（包括深度学习分支）是研究“学习算法”的一门学问。所谓“学习”是指：对于某类任务T和性能度量P，一个计算机程序在T上以P衡量的性能随着经验E而自我完善，那么我们称这个计算机程序在从经验E学习。



学习算法 (2)





基本术语和概念 (1)

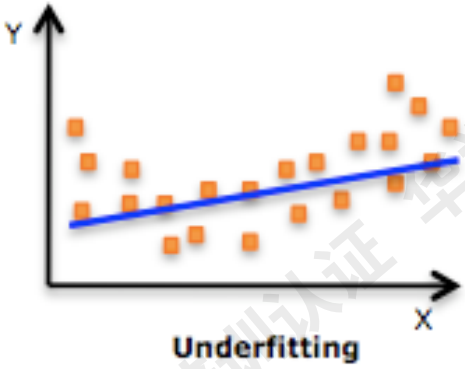
- **数据集**：在机器学习任务中使用的一组数据，其中的每一个数据称为一个样本。反映样本在某方面的表现或性质的事项或属性称为**特征**。
- **训练集**：训练过程中使用的数据集，其中每个训练样本称为训练样本。从数据中学得模型的过程称为**学习（训练）**。
- **测试集**：学得模型后，使用其进行预测的过程称为测试，使用的数据集称为测试集，每个样本称为测试样本。

基本术语和概念 (2)

- **泛化能力**：机器学习的目标是使学得的模型能够很好的适用于新的样本，而不是仅仅在训练样本上工作的很好，学得的模型适用于新样本的能力称为泛化能力。
- **误差**：学习到的模型在样本上的预测结果与样本的真实结果之间的差 。
  - 训练误差：模型在训练集上的误差。
  - 泛化误差：在新样本上的误差。显然，我们更希望得到泛化误差小的模型。
- **欠拟合**：如果训练误差很大的现象。
- **过拟合**：如果学得的模型的训练误差很小，而泛化能力较弱即泛化误差较大的现象。

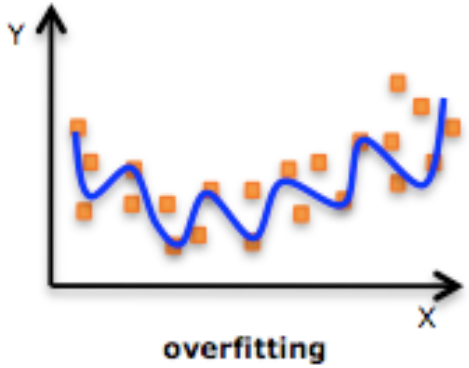
基本术语和概念 (3)

- **模型的容量**：指其拟合各种函数的能力。当机器学习算法的容量适合于执行任务的复杂度和所提供训练数据的数量时，算法效果通常会最佳；容量不足的模型不能解决复杂任务，可能出现欠拟合；容量高的模型能够解决复杂的任务，但是其容量高于任务所需时，有可能会过拟合。



欠拟合
没学到特征

好的拟合



过拟合
学习了噪声

基本术语和概念 (4)

- 机器学习可以解决多种类型的**任务**，下面列出最典型和常见的两种：
  - **分类**：计算机程序需要指定输入属于k 类中的哪一类。 为了完成这个任务，学习算法通常会输出一个函数$f: R^n \rightarrow (1,2,\ldots,k)$。比如计算机视觉中的图像分类算法解决的就是一个分类任务。
  - **回归**：这类任务中，计算机程序会对给定输入预测输出数值。学习算法通常会输出一个函数$f: R^n \rightarrow R$，这类任务的一个示例是预测投保人的索赔金额（用于设置保险费），或者预测证券未来的价格。
- **分类和回归是预测问题的两种主要类型，分类的输出是离散的类别值，而回归的输出是连续数值。**

基本术语和概念 (5)

- 术语：
  - P：正元组，感兴趣的主要类的元组。
  - N：负元组，其他元组。
  - TP：真正例，被分类器正确分类的正元组。
  - TN：真负例，被分类器正确分类的负元组。
  - FP：假正例，被错误地标记为正元组的负元组。
  - FN：假负例，被错误的标记为负元组的正元组。

预测 / 实际	yes	no	合计
yes	TP	FN	P
no	FP	TN	N
合计	P'	N'	$P+N$

混淆矩阵

- 混淆矩阵：是一个至少为$m \times m$的表。前m行和m列的表目$CM_{i,j}$指出类i的元组被分类器标记为j的个数。
  - 理想的，对于高准确率的分类器，大部分元组应该被混淆矩阵从$CM_{1,1}$到$CM_{m,m}$的对角线上的表目表示，而其他表目为0或者接近于0。即FP和FN接近0。

基本术语和概念 (6)

度量	公式
准确率、识别率	$\frac{TP+TN}{P+N}$
错误率、误分类率	$\frac{FP+FN}{P+N}$
敏感度、真正例率、召回率（$recall$）	$\frac{TP}{P}$
特效性、真负例率	$\frac{TN}{N}$
精度（$precision$）	$\frac{TP}{TP+FP}$
F、F_1、F分数精度和召回率的调和均值	$\frac{2\times precision\times recall}{precision+recall}$
F_β，其中β是非负实数	$\frac{(1+\beta^2)\times precision\times recall}{\beta^2)\times precision+recall}$

机器学习性能评估举例

- 我们训练了一个机器学习的模型用来识别图片中是不是一只猫，现在用200张图片来验证下模型的性能指标。这200张图片中，170张是猫，30张不是猫。模型的识别结果为160张是猫，40张不是猫。

精度：$P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5.\%$

召回率：$R = \frac{TP}{P} = \frac{140}{170} = 93.3\%.$

准确率：$ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 85\%$

预测 / 实际	*yes*	*no*	合计
yes	140	30	170
no	20	10	30
合计	160	40	200

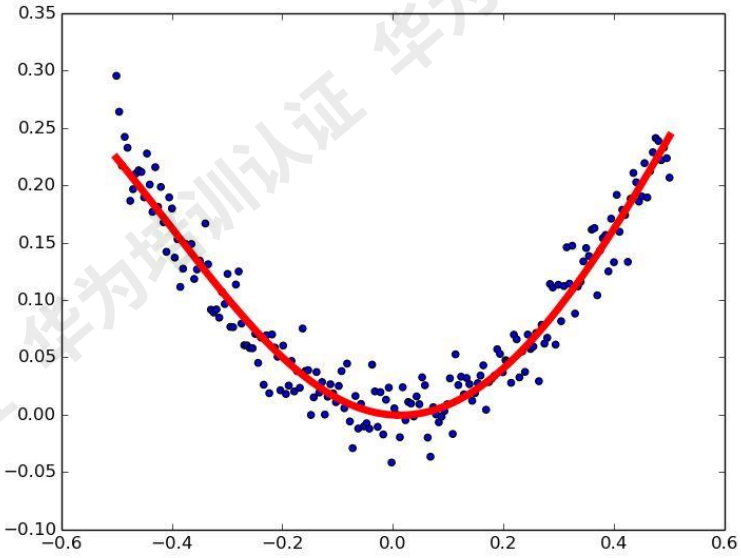
目录

机器学习分类

- **监督学习**：利用已知类别的样本，训练学习得到一个最优模型，使其达到所要求性能，再利用这个训练所得模型，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，即可以对未知数据进行分类。
- **无监督学习**：对于没有标记的样本，学习算法直接对输入数据集进行建模，例如聚类，即“物以类聚，人以群分”。我们只需要把相似度高的东西放在一起，对于新来的样本，计算相似度后，按照相似程度进行归类就好。
- **半监督学习**：试图让学习器自动地对大量未标记数据进行利用以辅助少量有标记数据进行学习。
- **强化学习**：学习系统从环境到行为映射的学习，以使奖励信号(强化信号)函数值最大，强化学习不同于连接主义学习中的监督学习，主要表现在教师信号上，强化学习中由环境提供的强化信号是对产生动作的好坏作一种评价(通常为标量信号)，而不是告诉强化学习系统如何去产生正确的动作。

常用机器学习算法 - 线性回归

- 线性回归（Linear regression）：线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。



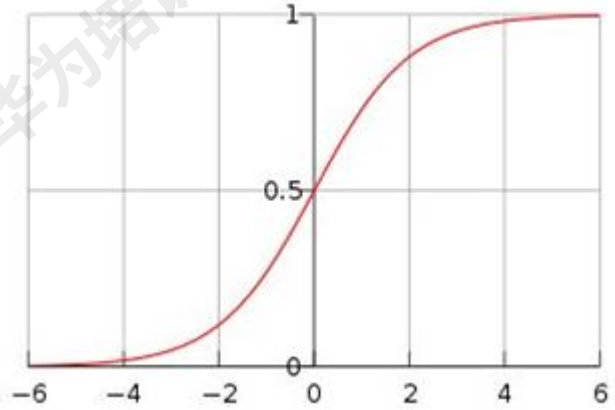
常用机器学习算法 - 逻辑回归

- 逻辑回归：逻辑回归模型是一种分类模型，用来解决分类问题。模型的定义：

$$P(Y=1|x)=\frac{e^{wx+b}}{1+e^{wx+b}}$$

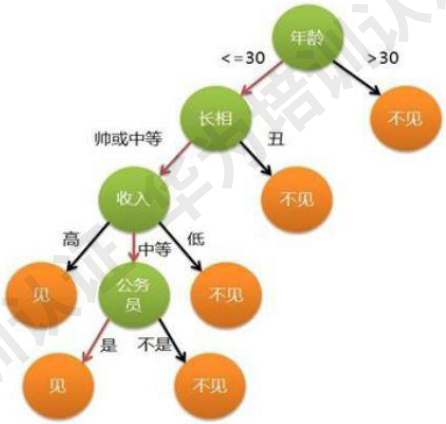
$$P(Y=0|x)=\frac{1}{1+e^{wx+b}}$$

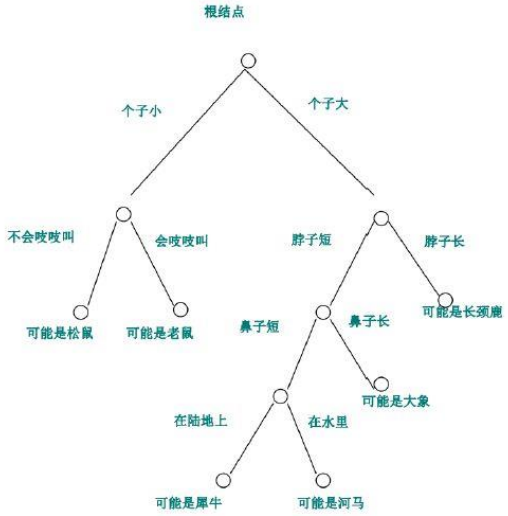
其中w称为权重，b称为偏置，其中的$wx+b$看成对x的线性函数。然后对比上面两个概率值，概率值大的就是x对应的类。



常用机器学习算法 - 决策树

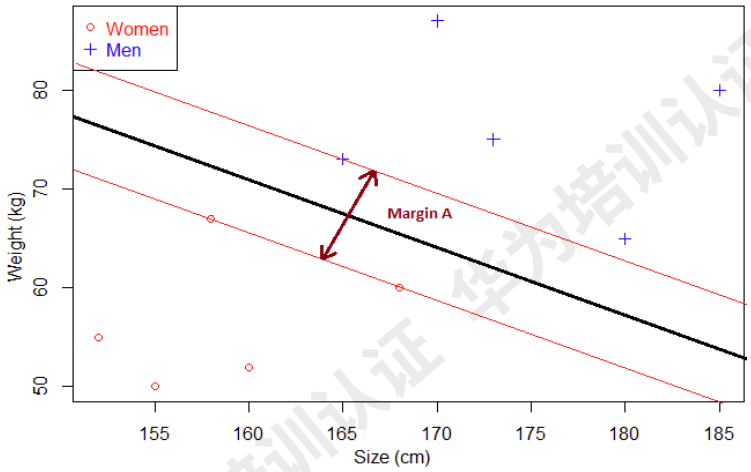
- 决策树：决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

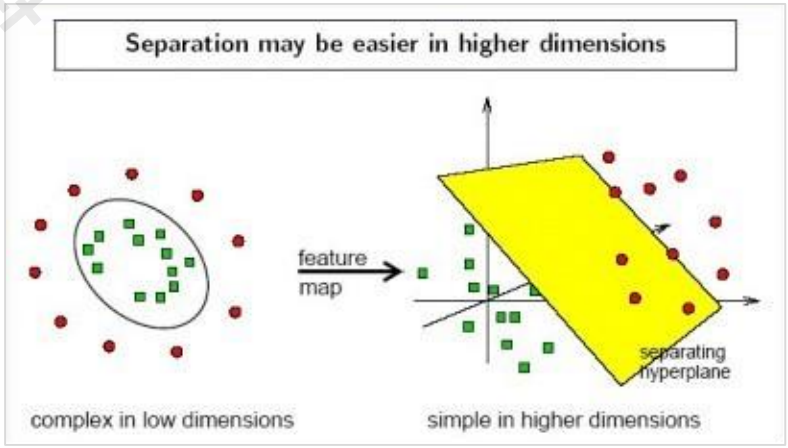




常用机器学习算法 - 支持向量机

- 支持向量机（support vector machine，SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器。SVM还包括核技巧，这使它成为实质上的非线性分类器。支持向量机的学习算法是求解凸二次规划的最优化算法。





常用机器学习算法 - 朴素贝叶斯

- **朴素贝叶斯算法（Naive Bayes）**：是基于**贝叶斯定理**与**特征条件独立假设**的分类方法。对于给定的训练数据集，首先基于特征条件独立假设学习输入/输出的联合概率分布，然后基于此模型，对给定的输入x，利用贝叶斯定理求出后验概率最大的输出y。根据贝叶斯定理，对一个分类问题，给定样本特征X，样本属于类别H的概率是：

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

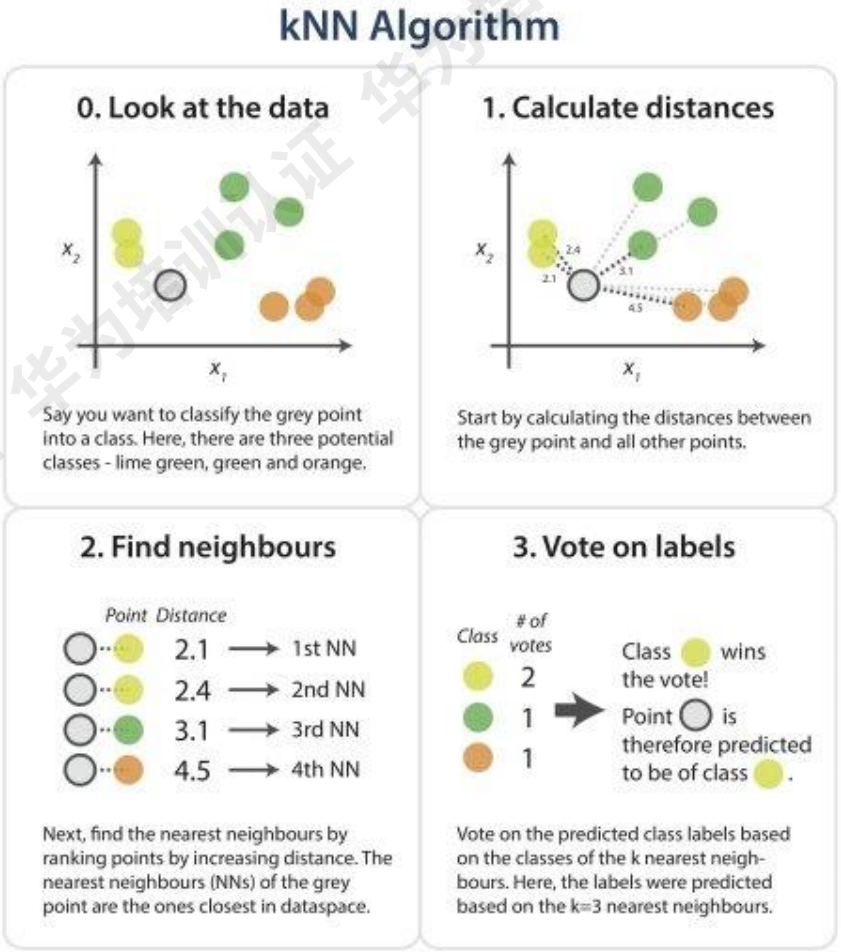
其中，X是数据元组，通常用n个属性集的测量值描述。H为某种假设，如数据元组X属于某个特定类C。$P(H|X)$是后验概率，或在条件X下，H的后验概率。$P(H)$是先验概率，或H的先验概率，$P(H)$独立于X。$P(X)$是X的先验概率。

- 因为朴素的假设，即特征条件独立，根据全概率公式展开，得出朴素贝叶斯法分类的基本公式：

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} = \frac{\prod_{k=1}^{M} P(X_i|C_k)P(C_k)}{\sum_k P(C_k)\prod_{k=1}^{M} P(X_i|C_k)}$$

常用机器学习算法 - KNN

- K最近邻(k-Nearest Neighbor，KNN)分类算法，是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。



目录

超参数和验证集 (1)

- 学习模型中一般有两种参数，一种**参数**是可以从学习中得到，还有一种无法靠数据里面得到，只能靠人的经验来设定，这类参数就叫做**超参数**。
- 模型超参数是模型外部的配置，具体特征有：
  - 模型超参数常应用于估计模型参数的过程中。
  - 模型超参数通常由实践者直接指定。
  - 模型超参数通常可以使用启发式方法来设置。
  - 模型超参数通常根据给定的预测建模问题而调整。
- 模型超参数举例：
  - 训练神经网络的学习速率，迭代次数，批次大小，激活函数，神经元的数量。
  - 支持向量机的C和σ超参数。
  - K近邻中的K。

超参数和验证集 (2)

- 超参数搜索的一般过程：
  - 将数据集分成训练集、验证集、测试集。
  - 在训练集上根据模型的性能指标对模型参数进行优化。
  - 在验证集上根据模型的性能指标对模型的超参数进行搜索。
  - 步骤2和步骤3交替迭代进行，最终确定模型的参数和超参数，并在测试集中评价模型的优劣。
- 其中，步骤3的搜索过程需要搜索算法，一般有如下：
  - 网格搜索
  - 随机搜索
  - 启发式智能搜索
  - 贝叶斯搜索

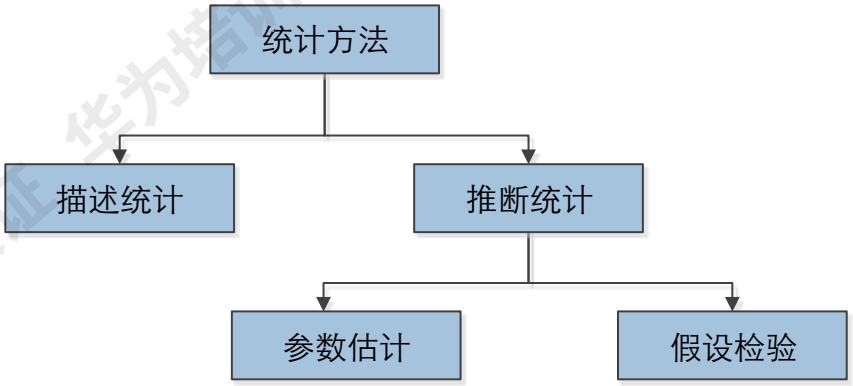
超参数和验证集 (3)

- **交叉验证**：是用来验证分类器的性能一种统计分析方法，基本思想是把在某种意义下将原始数据进行分组，一部分作为训练集，另一部分作为验证集，首先用训练集对分类器进行训练，再利用验证集来测试训练得到的模型，以此来做为评价分类器的性能指标。
- **k-折交叉验证（$K-CV$）**：
  - 将原始数据分成k组（一般是均分）。
  - 将每个子集数据分别做一次验证集，其余的$k-1$组子集数据作为训练集，这样会得到k个模型。
  - 用这k个模型最终的验证集的分类准确率的平均数作为此$K-CV$下分类器的性能指标。

目录

参数估计

- **参数估计**：设有一个统计总体，总体的分布函数为$F(x,\theta)$，其中θ为未知参数，现从该总体抽样，得样本$X_1, X_2, \ldots, X_n$，要依据该样本对参数θ作出估计$\hat{\theta}$，或估计θ的某个已知函数$g(\theta)$，这类问题称为**参数估计**。参数估计分为**点估计**和**区间估计**。
- **参数估计在统计方法中的地位：**



点估计量

- 已知某地区新生婴儿的体重$X \sim N(\mu, \sigma^2)$（μ，σ^2未知），随机抽查100个婴儿，得100个体重数据

  $$10,7,5,6,6,5.2, \ldots$$

  而全部信息就由这100个数组成，据此我们应如何估计μ和σ呢?

- **点估计量**：我们需要构造出适当的样本的函数$T(X_1, X_2, \ldots, X_n)$，每当有了样本，就代入到该函数中算出一个值，用来作为μ的估计值。
  $T(X_1, X_2, \ldots, X_n)$称为参数的点估计量。

- 问题是：使用什么样的估计量去估计μ? 如何评价估计量的好坏?
  - 可以用样本均值。
  - 也可以用样本中位数。
  - 或者是其他的统计量。

衡量统计量的标准 – 无偏性

- **无偏性**：估计的偏差被定义为：

$$bias(\hat{\theta}) = E(\hat{\theta}) - \theta$$

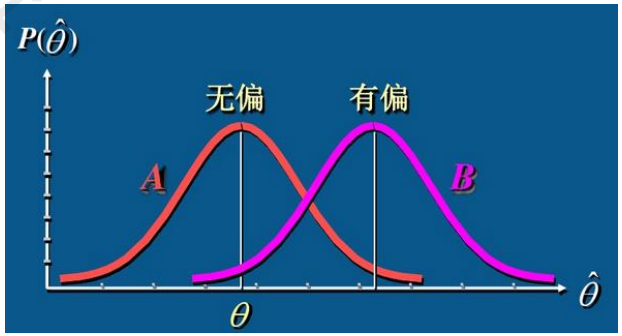
如果$bias(\hat{\theta})$=0，即$E(\hat{\theta}) = \theta$，那么估计量$\hat{\theta}$被称为是无偏的。

如果$\lim bias(\hat{\theta})$=0，即$\lim E(\hat{\theta}) = \theta$，$\hat{\theta}$被称为是渐进无偏的。

- 例：设$(X_1, X_2, \ldots, X_n)$是来自总体X的样本，$EX = \mu$，$DX = \sigma^2$.试证

（1）$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$是$\mu$的无偏估计量。

（2）$S^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2\right)$是$\sigma^2$的无偏估计量。



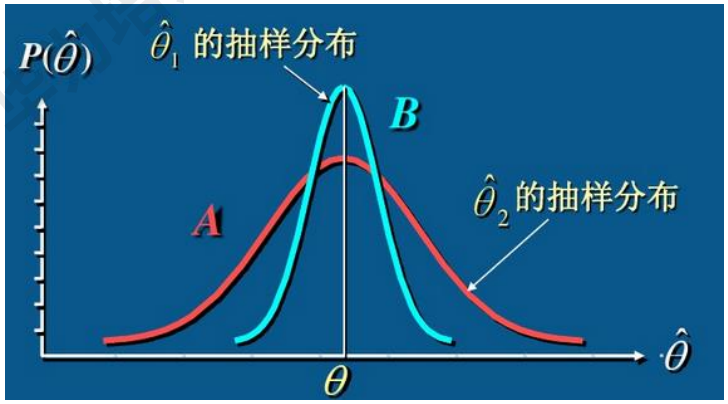
衡量统计量的标准 – 有效性

- 例：X_1，X_2，X_3是总体X的样本。下列统计量是否为总体均值μ的无偏估计量？谁更优？

$$\widehat{\mu_1}=\frac{2}{5}X_1+\frac{1}{10}X_2+\frac{1}{2}X_3,\ \widehat{\mu_2}=\frac{1}{3}X_1+\frac{3}{4}X_2-\frac{1}{12}X_3,$$

$$\widehat{\mu_3}=\frac{1}{2}X_1+\frac{1}{3}X_2+\frac{1}{6}X_3,\ \widehat{\mu_4}=\frac{1}{5}X_1+\frac{1}{10}X_2+\frac{7}{10}X_3.$$

- **有效性**：设$\widehat{\theta_1}$和$\widehat{\theta_2}$是θ的两个无偏估计量，若$D(\widehat{\theta_1})<D(\widehat{\theta_2})$，称$\widehat{\theta_1}$比$\widehat{\theta_2}$有效。



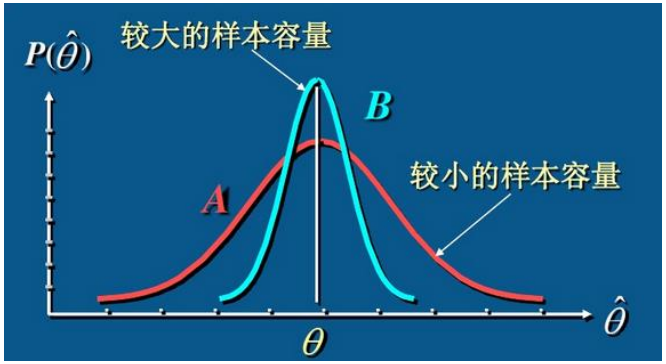
衡量统计量的标准 – 相合性

- **一致性（相合性）**：在概率P的意义下，$\hat{\theta} \to \theta(n \to \infty)$，即对于$\forall \epsilon > 0$，有

  $\lim_{n\to\infty} P(|\hat{\theta} - \theta| > \varepsilon)=0.$

- 例：设总体X的期望$EX = \mu$，方差$DX = \sigma^2$，X_1，X_2，…，X_n是来自X的样本，

  $\widehat{\mu_1}=\frac{1}{n}\sum_{i=1}^{n} X_i$，$\widehat{\mu_2}=\frac{1}{k}\sum_{i=1}^{k} X_i$，$k < n$，试证$\widehat{\mu_1}$，$\widehat{\mu_2}$都是$\mu$的无偏估计量，且$\widehat{\mu_1}$比$\widehat{\mu_2}$更有效。



证明：$E(\widehat{\mu_1}) = E\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n} \times n\mu = \mu.$

$E(\widehat{\mu_2}) = E\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) = \frac{1}{k} \times k\mu = \mu.$

$D(\widehat{\mu_1}) = D\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} D(X_i) = \frac{\sigma^2}{n}$，$D(\widehat{\mu_2}) = \frac{\sigma^2}{k}.$

由于$n > k$，故$D(\widehat{\mu_1})<D(\widehat{\mu_2})$所以估计量$\widehat{\mu_1}$更有效。

目录

最大似然估计 (1)

- 例：假设在一个袋子中有多个黑球和白球，并假定已知两种球的数目之比是1:3，但不知道哪种颜色的球多。如果在袋子中重复抽样n次，则其中黑球的个数为x的概率为：

$$p(x;p) = \binom{n}{x} p^x q^{n-x},$$

其中$q = 1 - p$，由假设知，$p = \frac{1}{4}$或$\frac{3}{4}$。若取$n = 3$，可计算得x在两种p值之下的概率：

x	0	1	2	3
$P(x,\frac{3}{4})$	$\frac{1}{64}$	$\frac{9}{64}$	$\frac{27}{64}$	$\frac{27}{64}$
$P(x,\frac{1}{4})$	$\frac{27}{64}$	$\frac{27}{64}$	$\frac{9}{64}$	$\frac{1}{64}$

上表观察可知，$\hat{p}(x) = \begin{cases} \frac{1}{4}, x = 0,1 \\ \frac{3}{4}, x = 2,3 \end{cases}$.

最大似然估计 (2)

- **最大似然估计原理**：设X_1，X_2，…，X_n是取自总体X的一个样本，样本的联合密度或联合分布律为$f(x_1, x_2, \cdots, x_n; \theta)$。定义似然函数为：

$$L(\theta) = f(x_1, x_2, \cdots, x_n; \theta),$$

其中，$x_1, x_2, \cdots, x_n$是样本的观察值；$L(\theta)$看做参数θ的函数，它可作为θ将以多大可能产生样本值$x_1, x_2, \cdots, x_n$的一种度量。

- 最大似然估计法就是用使$L(\theta)$达到最大值的$\hat{\theta}$去估计θ。

$$L(\hat{\theta}) = \max_{\theta} L(\theta),$$

称$\hat{\theta}$为θ的最大似然估计值。而相应的统计量$\hat{\theta}(X_1，X_2，\ldots，X_n)$称为$\theta$的最大似然估计量。

最大似然估计举例

- 例：设X_1，X_2，…，X_n是取自总体$X \sim B(1,p)$的一个样本，求参数p的最大似然估计量。

  解：似然函数：

$$L(p) = f(x_1, x_2, \cdots, x_n; p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{\sum_{i=1}^{n} x_i}(1-p)^{n-\sum_{i=1}^{n} x_i}.$$

  对数似然函数：

$$\ln L(p) = \sum_{i=1}^{n} x_i \ln(p) + (n - \sum_{i=1}^{n} x_i)\ln(1-p).$$

  对p求导并令其为0：

$$\frac{d \ln L(p)}{dp} = \frac{1}{p}\sum_{i=1}^{n} x_i - \frac{1}{1-p}(n - \sum_{i=1}^{n} x_i) = 0.$$

  得 $\hat{p} = \bar{x}$.从而p的最大似然估计量为：

$$\hat{p}(X_1, X_2, \ldots, X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}.$$

目录

贝叶斯定理

- 贝叶斯定理提供了一种由$P(D)$、$P(h)$、$P(D|h)$计算后验概率$P(h|D)$的方法。公式如下：

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

其中，D是数据元组，通常用n个属性集的测量值描述。h为某种假设，如数据元组D属于某个特定类C。$P(h \mid D)$ 是后验概率，或在条件D下，h的后验概率。$P(h)$是先验概率，或h的先验概率，$P(h)$独立于D。$P(D)$是D的先验概率。

极大后验概率假设

- $P(h|D)$是假设h的后验概率，使$P(h|D)$最大化的假设h，称为极大后验（maximum a posteriori，MAP）假设。形式化地，MAP假设 h_{MAP} 满足：

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h)$$

- 在实际使用中，我们往往不能得到各个假设的先验概率，我们只能认为假设空间中的所有假设都是等可能的，此时，$P(h)$可认为是常数，有：

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) = \underset{h \in H}{\operatorname{argmax}} P(D|h) = h_{ML}$$

其中，$P(D \mid h)$常被称为给定h时数据D的似然度，因此，使得$P(D|h)$最大的假设称为极大似然假设h_{ML}。

- 当假设空间中的假设都是等可能的假设时，h_{MAP}=h_{ML}。

促使深度学习发展的挑战

- 维数灾难
- 局部不变性和平滑正则化
- 流形学习

目录

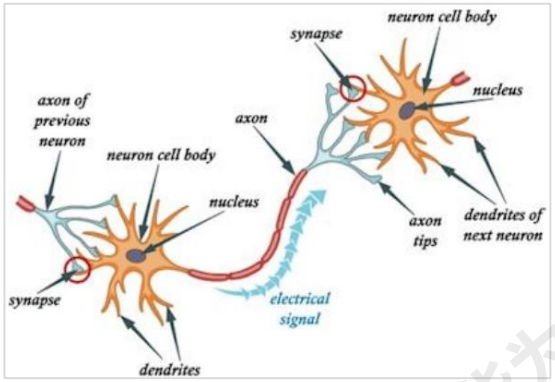
HUAWEI

神经网络定义

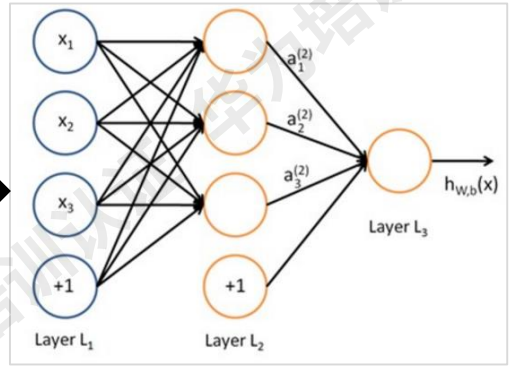
- 目前，关于神经网络的定义尚不统一，按美国神经网络学家Hecht Nielsen的观点，神经网络的定义是：“神经网络是由多个非常简单的处理单元彼此按某种方式相互连接而形成的计算机系统，该系统靠其状态对外部输入信息的动态响应来处理信息”。
- 综合神经网络的来源、特点和各种解释，它可简单地表述为：**人工神经网络是一种旨在模仿人脑结构及其功能的信息处理系统。**
- **人工神经网络（简称神经网络）**：是由人工神经元互连组成的网络，它是从微观结构和功能上对人脑的抽象、简化，是模拟人类智能的一条重要途径，反映了人脑功能的若干基本特征，如并行信息处理、学习、联想、模式分类、记忆等。

深度学习

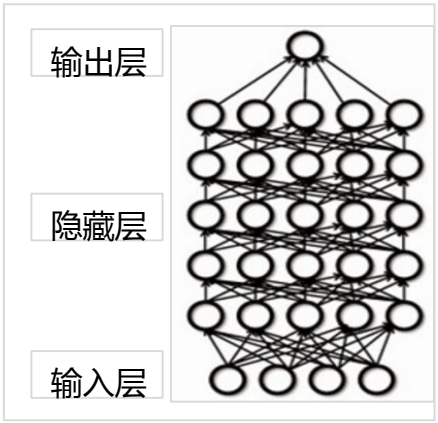
- 深度学习一般指深度神经网络，这里的深度指神经网络的层数（较多）。



人类神经网络

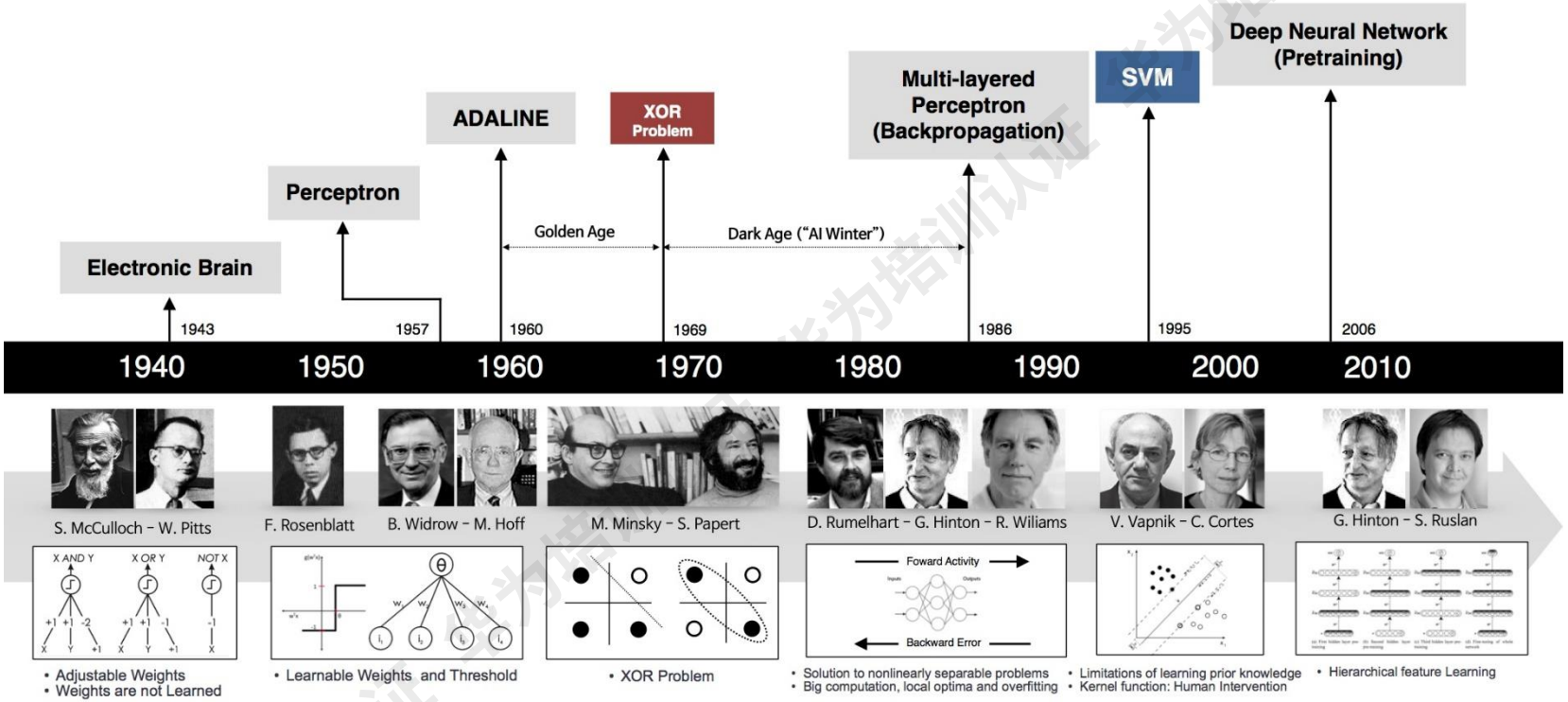


感知器



深度神经网络

深度学习里程碑

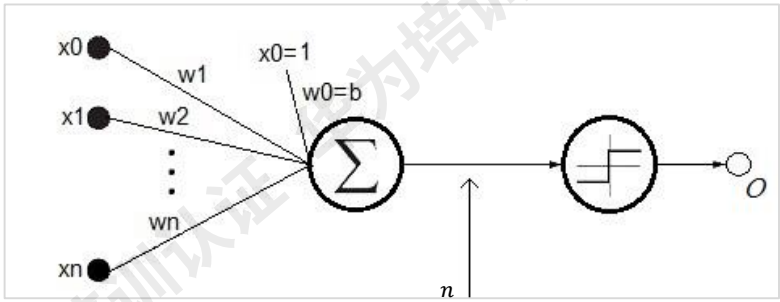


HUAWEI

目录

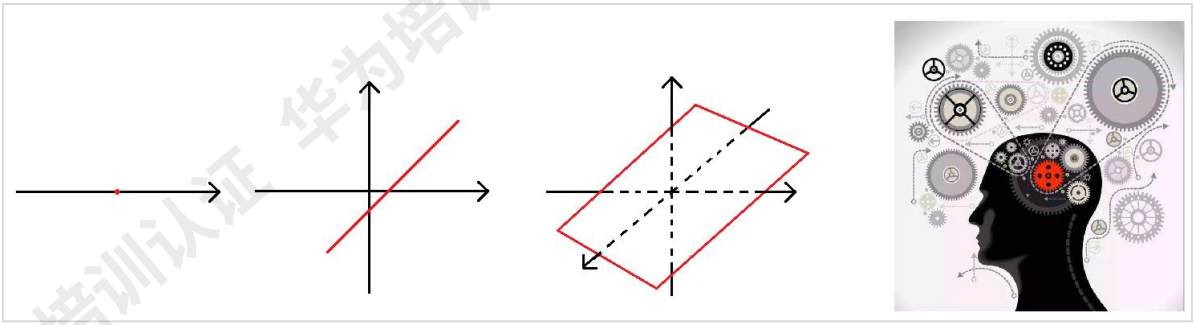
感知器

- **输入向量**：$X = [x_0, x_1, \ldots, x_n]^T$.
- **权值**：$W = [\omega_0, \omega_1, \ldots, \omega_n]^T$，其中$\omega_0$称为偏置。
- **激活函数**：$O = sign(net) = \begin{cases} 1, net > 0, \\ -1, otherwise. \end{cases}$



$$net = \sum_{i=0}^{n} \omega_i x_i = \boldsymbol{W^T X}$$

- 上面的感知器，相当于一个分类器，它使用高维X向量做输入，在高维空间对输入的样本进行二分类：当$\boldsymbol{W^T X} > 0$时，$o = 1$，相当于样本被归类为其中一类。否则，$o = -1$，相当于样本被归类为另一类。这两类的边界在哪里呢？就是$\boldsymbol{W^T X} = 0$，这是一个高维超平面。



分割点 $Ax + B = 0$　　分割直线 $Ax + By + C = 0$　　分割平面 $Ax + By + Cz + D = 0$　　分割超平面 $W^T X + b = 0$

感知器的训练法则

- 感知器的训练法则：对于每一个训练样例$< X, t >$
  - 使用当前的权值计算感知器输出o；
  - 对于每一个权值做如下的更新：

$$\omega_i \leftarrow \omega_i + \Delta\omega_i$$

$$\Delta\omega_i = \eta[t - o]x_i$$

  - 其中，X为输入向量，t为目标值，o为感知器当前权值下的输出，η为学习率，x_i和ω_i为向量X和W的第i个元素。
- 当训练样例线性可分时，反复使用上面的方法，经过有限次训练，感知器将收敛到能正确分类所有训练样例的分类器。
- 在训练样例线性不可分时，训练很可能无法收敛。因此，人们设计了另一个法则来克服这个不足，称为**delta法则**。它使用**梯度下降**（Gradient Descent）的方法在假设空间中所搜可能的权向量，寻找到最佳拟合训练样例的权向量。

梯度下降与损失函数

- 对于多元函数$o = f(x) = f(x_0, x_1, \dots, x_n)$，其在$X' = [x_0', x_1', \dots, x_n']^T$处的梯度为：

$$\nabla f(x_0', x_1', \dots, x_n') = [\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}]^T |_{X=X'},$$

梯度向量的方向，指向函数增长最快的方向。因此，负梯度向量$-\nabla f$，则指向函数下降最快的方向。

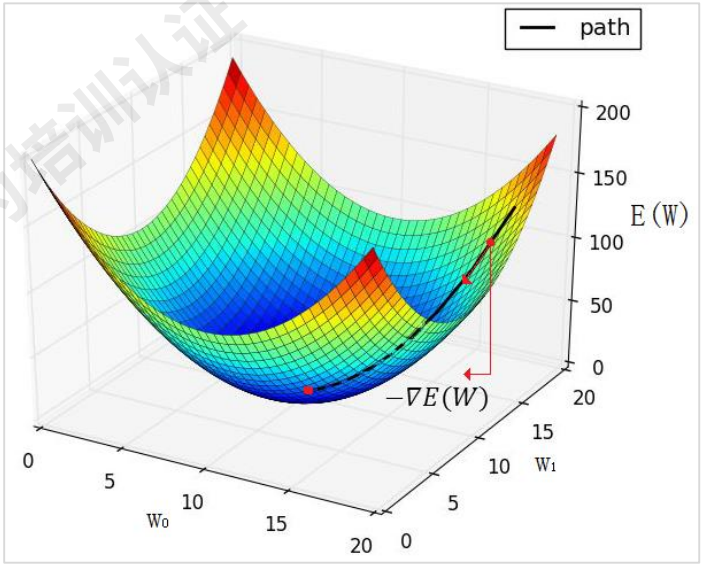
- 当训练样例线性不可分的时候，我们无法找到一个超平面，令感知器完美分类训练样例，但是我们可以近似地分类他们，而允许一些小小的分类错误。怎样让这个错误最小呢，首先要参数化描述这个错误，这就是**损失函数（误差函数）**，它反映了感知器目标输出和实际输出之间的误差。最常用的误差函数为**L2误差**：

$$E(w) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2,$$

其中，d为训练样例，D为训练样例集，t_d为目标输出，o_d为实际输出。

损失函数的极值

- 既然损失函数$E(W)$的自变量是权值，因此他是定义在权值空间上的函数。那么问题就转化成了在权值空间中，搜索使得$E(W)$最小的权值向量W。然而不幸的是，$E(W)=\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2$定义了一个非常负杂的高维曲面，而**数学上，对高维曲面的极值求解还没有有效的方法**。既然负梯度方向是函数下降最快的方向，那我们可以从某个点开始，沿着$-\nabla E(W)$方向一路前行，期望最终可以找到$E(W)$的极小值点，这就是梯度下降法的核心思想。



二元抛物面梯度下降示例

线性单元全局梯度下降算法

- 对于训练样例集D中的每一个样例记为$< X, t >$，X是输入值向量，t为目标输出，η是学习率。
  - 初始化每个w_i为绝对值较小的随机值
  - 遇到终止条件前，do：
    - 初始化每个Δw_i为零
    - 对于D中每个$< X, t >$，do：
      - 将X输入此单元，计算输出o
      - 对于此单元的每个w_i，do：Δw_i+= $\eta(t-o)\ x_i$
    - 对于此单元的每个w_i，do：w_i+= Δw_i
- 这个版本的梯度下降算法，实际上并不常用，它的主要问题是：
  - 收敛过程非常慢，因为每次更新权值都需要计算所有的训练样例；
  - 如果误差曲面上有多个局部极小值，那么这个过程极易陷入局部极值。

随机梯度下降算法和在线学习

- 针对原始梯度下降算法的弊端，一个常见的变体称为增量梯度下降（Incremental Gradient Descent），亦即随机梯度下降（SGD：Stochastic Gradient Descent）。其中一种实现称为在线学习（Online Learning），它根据每一个样例来更新梯度：

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \Longrightarrow \Delta w_i = \eta (t_d - o_d) x_{id}.$$

- ONLINE-GRADIENT-DESCENT(D, η)
  - 初始化每个w_i为绝对值较小的随机值
  - 遇到终止条件前，do：
    - 对于D中每个<X, t>，do：
      - 将X输入此单元，计算输出o
      - 对于此单元的每个w_i，do：w_i += η(t-o) x_i

Mini-Batch梯度下降

- 针对上两种梯度下降算法的弊端，提出了一个实际工作中最常用的梯度下降算法，即Mini-Batch SGD。它的思想是每次使用一小批固定尺寸（BS：Batch Size）的样例来计算Δw_i，然后更新权值。
- BATCH-GRADIENT-DESCENT(D, η, BS)
  - 初始化每个w_i为绝对值较小的随机值
  - 遇到终止条件前，do：
    - 初始化每个Δw_i为零
    - 从D中下一批（BS个）样例，对这批样例中的每一个$< X, t >$，do：
      - 将X输入此单元，计算输出o
      - 对于此单元的每个w_i，do：Δw_i += $\eta(t-o)\ x_i$
    - 对于此单元的每个w_i，do：w_i += Δw_i
    - 如果已经是最后一批，打乱训练样例的顺序。

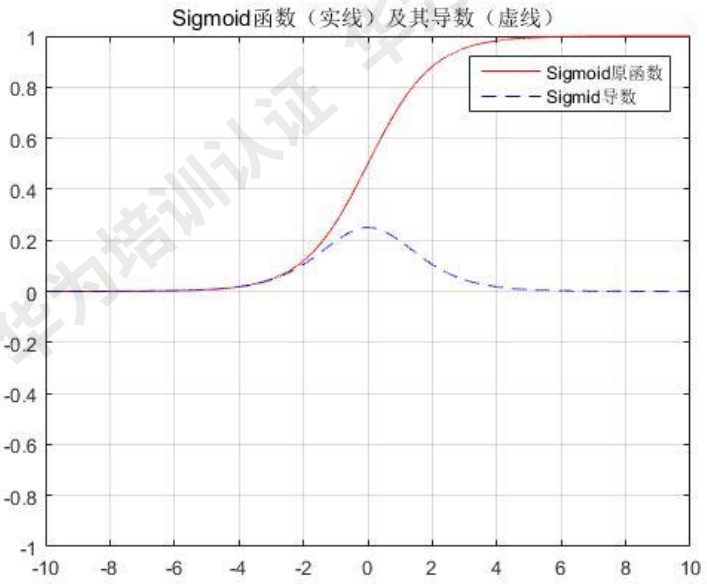
目录

HUAWEI

激活函数 - Sigmoid函数

- Sigmoid函数：

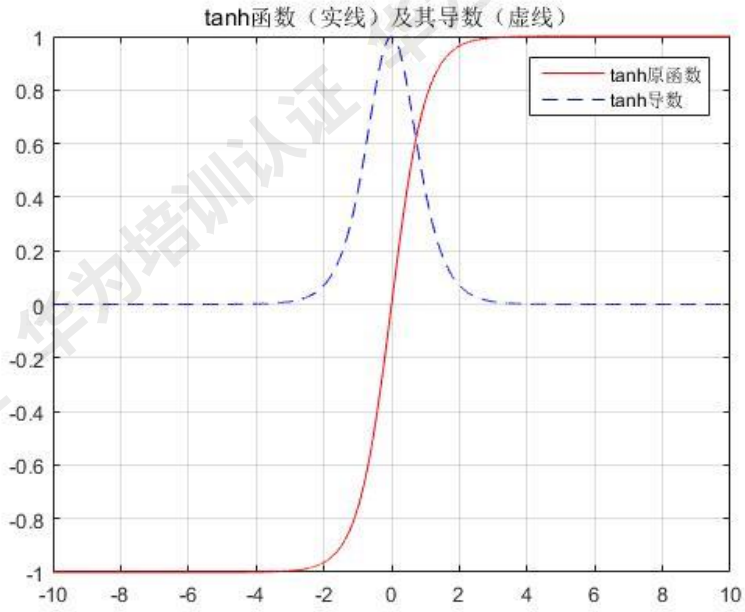
$$f(x) = \frac{1}{1 + e^{-x}}$$



激活函数 - tanh函数

- tanh函数:

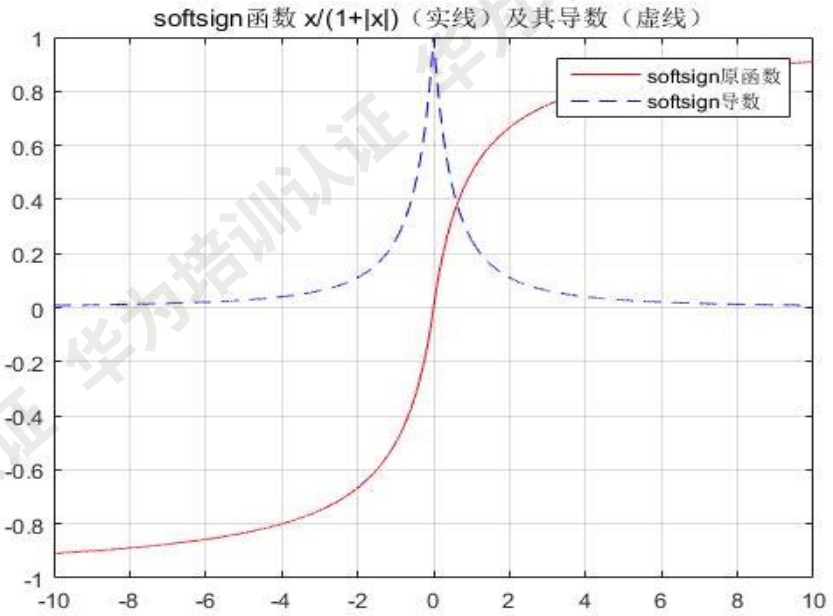
$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



激活函数 - Softsign函数

- Softsign函数：

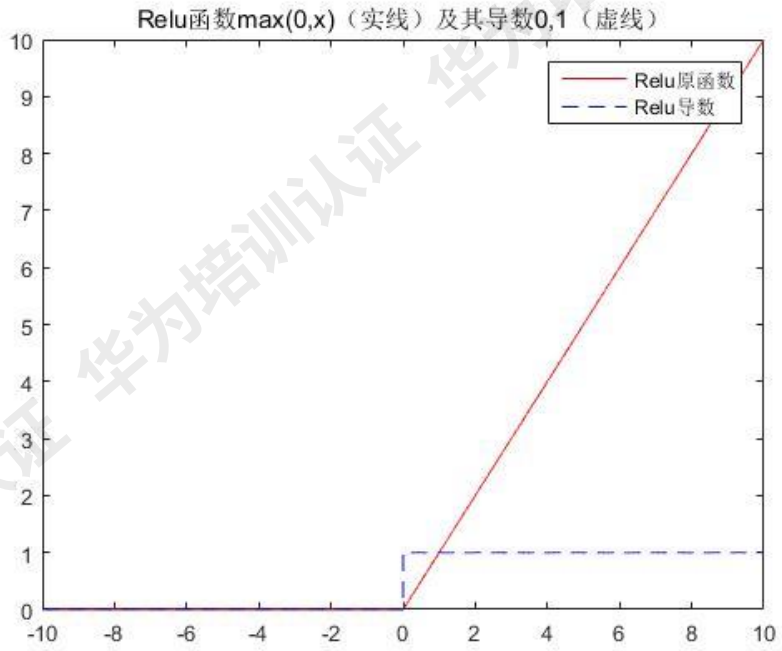
$$f(x) = \frac{x}{|x| + 1}$$



激活函数 - ReLU函数

- ReLU（Rectified Linear Unit）函数：

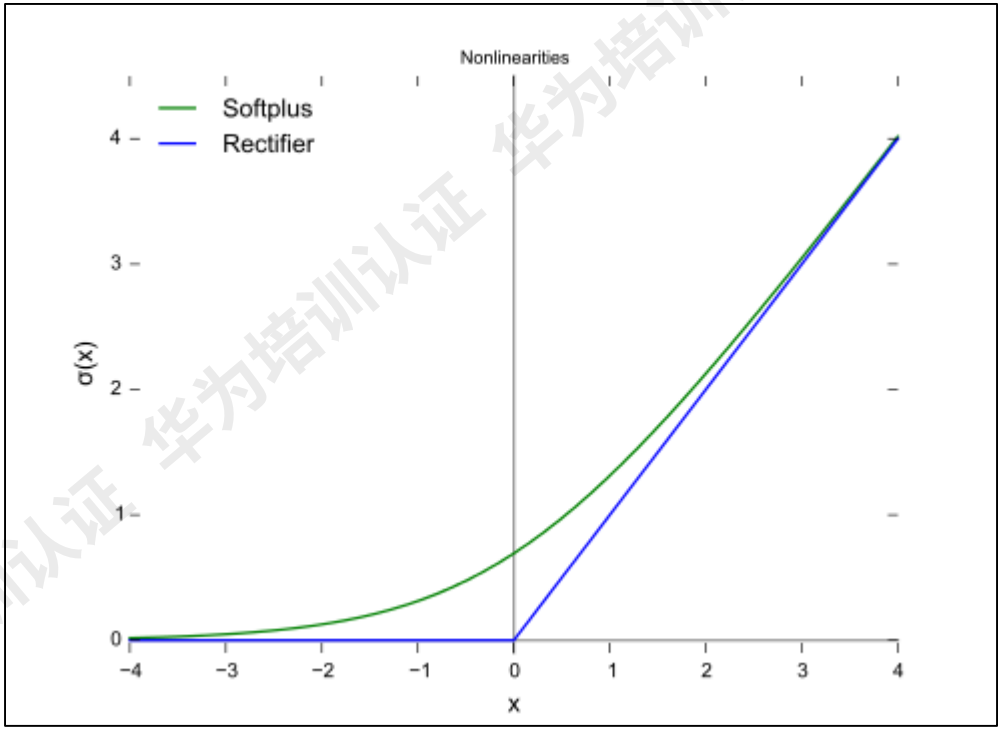
$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



激活函数 - Softplus函数

- Softplus函数：

$$f(x) = \ln(e^x + 1)$$



激活函数设计需考虑的因素

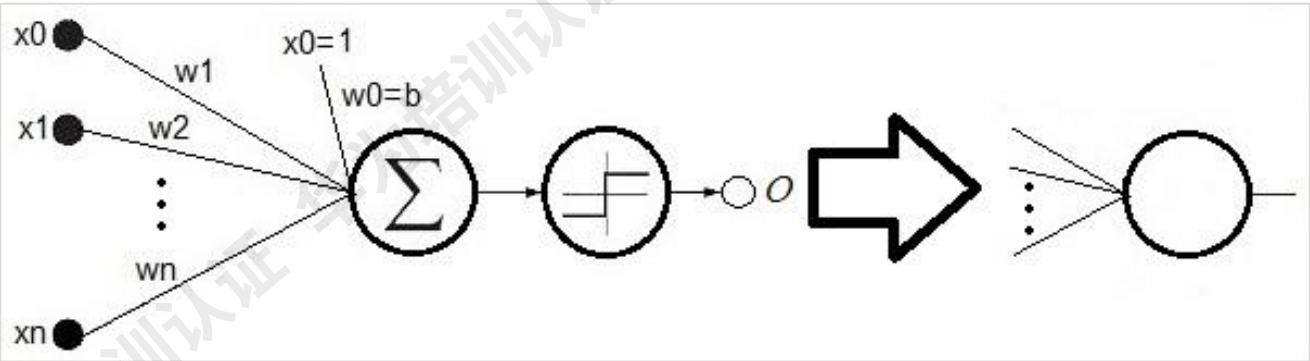
- **非线性**：当激活函数是非线性的，一个两层神经网络可以证明是一个通用函数近似值，如果失去了非线性，整个网络就相当于一个单层的线性模型。
- **连续可微性**：这个属性对基于梯度优化方法是必要的，如果选择了一些具有局部不可微的函数，则需要强行定义此处的导数。
- **有界性**：如果激活函数有界的，基于梯度的训练方法往往更稳定；如果是无界的，训练通常更有效率，但是训练容易发散，此时可以适当减小学习率。
- **单调性**：如果激活函数是单调的，与单层模型相关的损失函数是凸的。
- **平滑性**：有单调导数的平滑函数已经被证明在某些情况下泛化效果更好。
- **原点附近近似Identity**：当激活函数有这个特点时，对于小的随机初始化权重，神经网络能够更有效地学习。否则，在初始化权值时往往需要进行特殊设计。

目录

HUAWEI

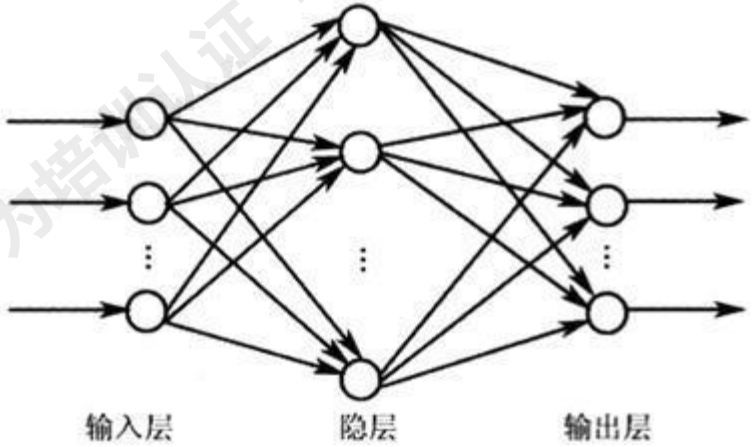
多层全连接人工神经网络

- 单个感知器的表达能力有限，它只能表达线性决策面（超平面）。如果我们把众多的感知器互联起来，就像人的大脑做所的那样，再将激活函数更换为非线性函数，我们就可以表达种类繁多的非线性曲面。



前馈神经网络

- **前馈神经网络**是一种最简单的神经网络，各神经元分层排列。是目前应用最广泛、发展最迅速的人工神经网络之一。
- 可以看出，输入节点并无计算功能，只是为了表征输入矢量各元素值。
- 各层节点表示具有计算功能的神经元，称为计算单元。每个神经元只与前一层的神经元相连。
- 接收前一层的输出，并输出给下一层，采用一种单向多层结构，每一层包含若干个神经元，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。



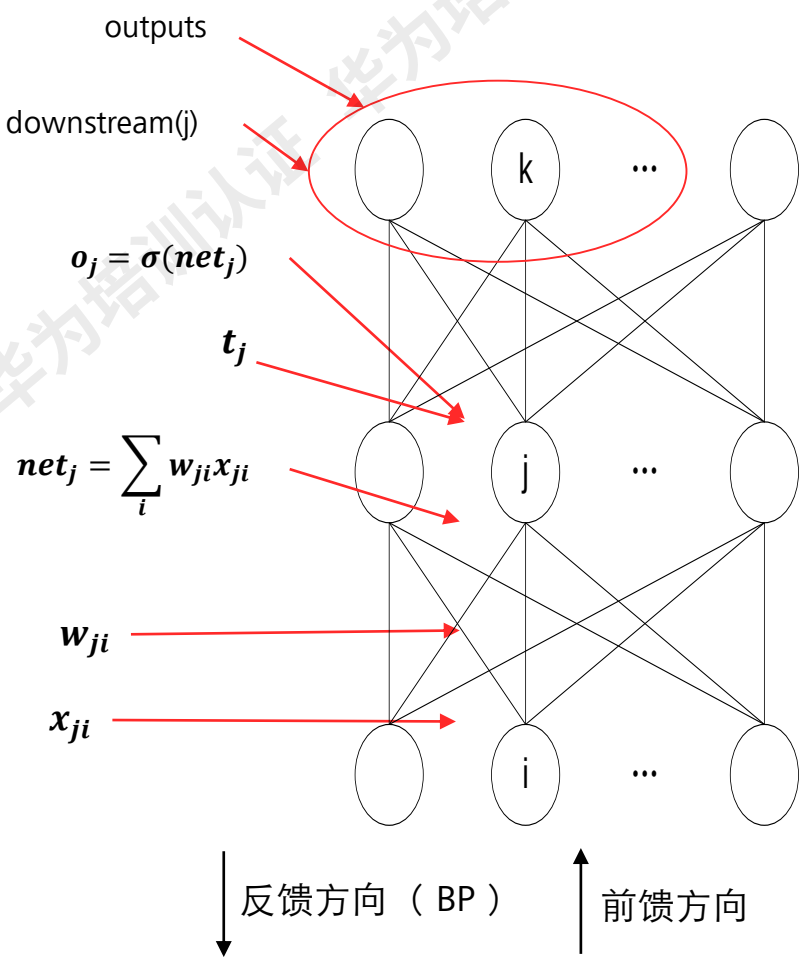
反向传播算法

- 有了这个公式，我们就可以训练神经网络了，公式重列如下：

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

$$\delta_j = \begin{cases} o_j(1-o_j)(t_j - o_j), & j \in outputs \quad (1) \\ o_j(1-o_j)\sum_{k \in DS(j)} \delta_k w_{kj}, & otherwise \quad (2) \end{cases}$$

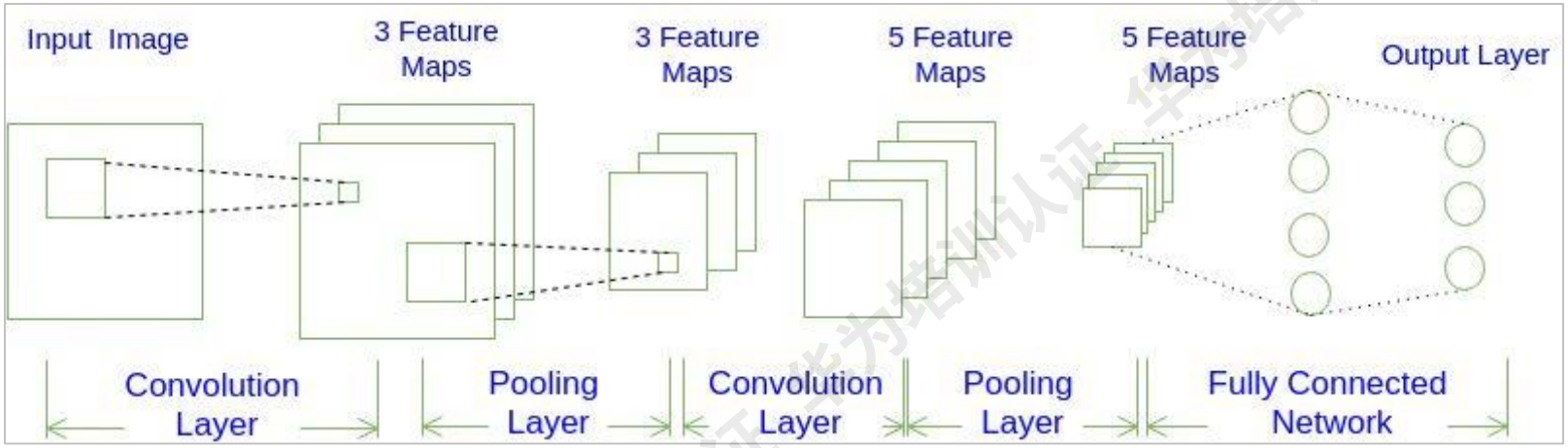
- BP算法训练网络的步骤如下：
  1. 取出下一个训练样例$< X, T >$，将X输入网络，得到实际输出O。
  2. 根据输出层误差公式（1）求取输出层δ，并更新权值。
  3. 对于隐层，根据隐层误差传播公式（2）从输出往输入方向反向、逐层、迭代计算各层的δ，每计算好一层的δ，更新该层权值，直至所有权值更新完毕。
  4. 返回1中继续。

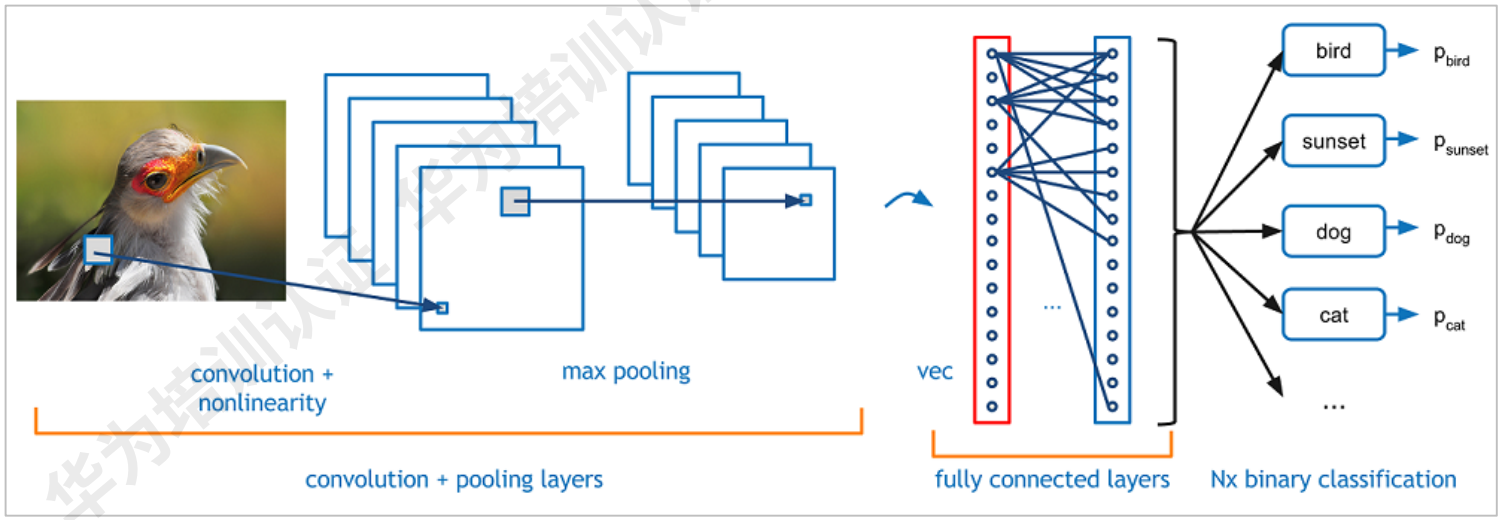


卷积神经网络 (1)

- 卷积神经网络（Convolutional Neural Network,CNN）是一种前馈神经网络，它的人工神经元可以**响应一部分覆盖范围内的周围单元**，对于图像处理有出色表现。它包括**卷积层**(convolutional layer)，**池化层**(pooling layer)和**全连接层**(fully_connected layer)。
- 20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称CNN）。
- 现在，CNN已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。

卷积神经网络 (2)

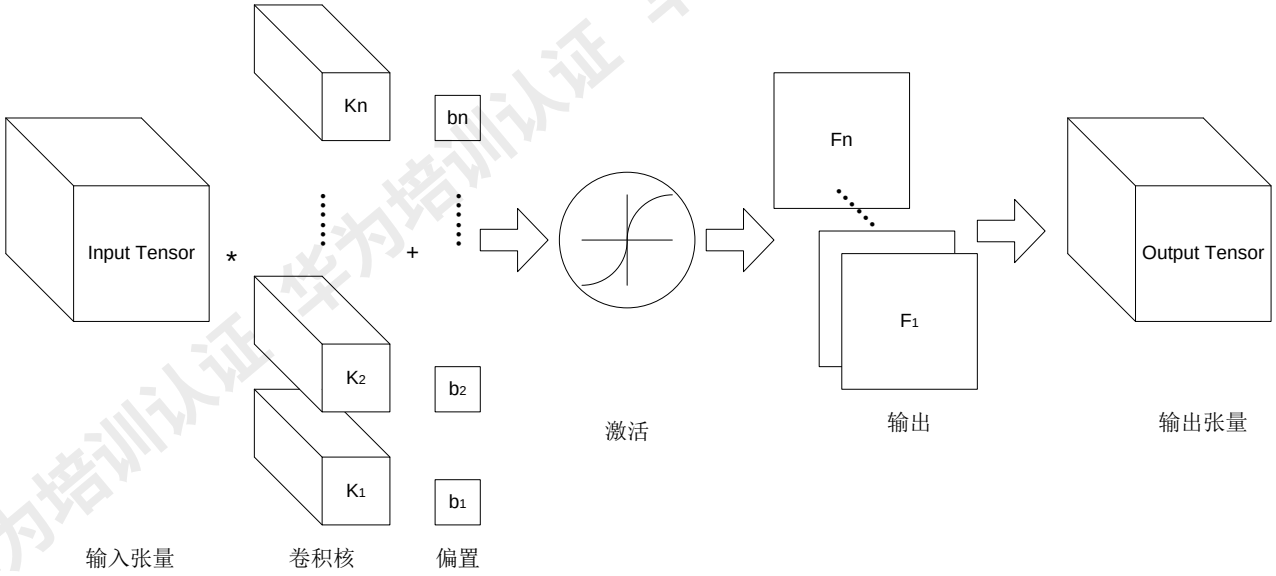




HUAWEI

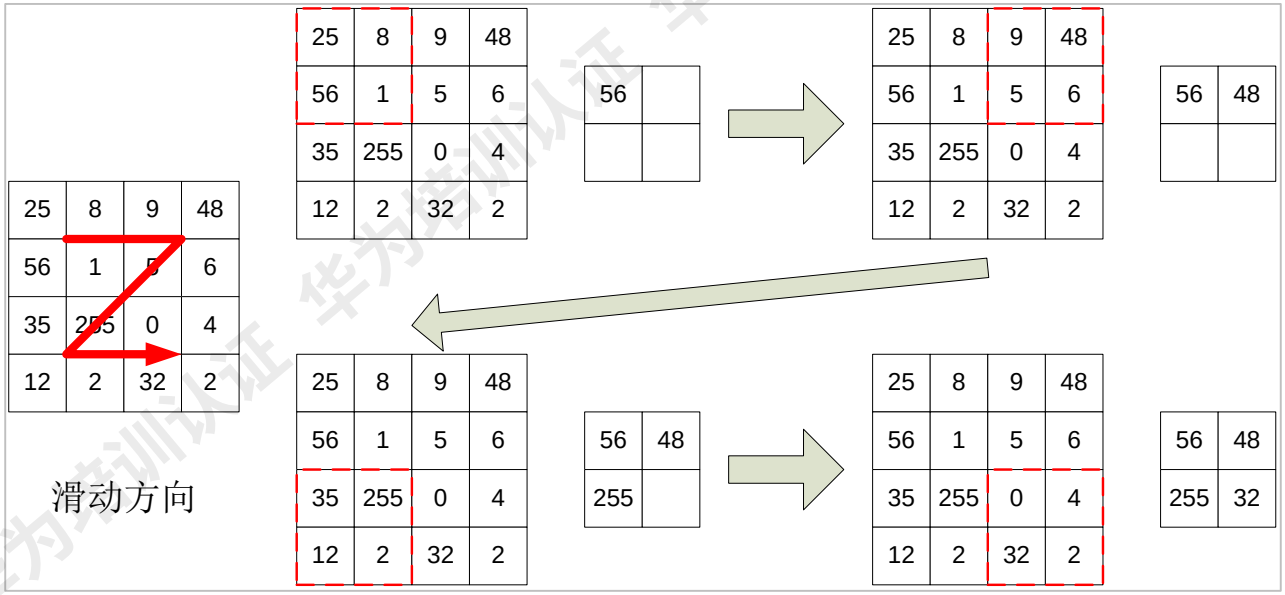
卷积层

- 卷积神经网络的基本结构，就是前面说的多通道卷积。上一层的输出（或者第一层的原始图像），作为本层的输入，然后和本层的卷积核卷积，作为本层输出。而各层的卷积核，就是要学习的权值。和FCN类似，卷积完成后，输入下一层之前，也需要经过偏置和通过激活函数进行激活。



池化层

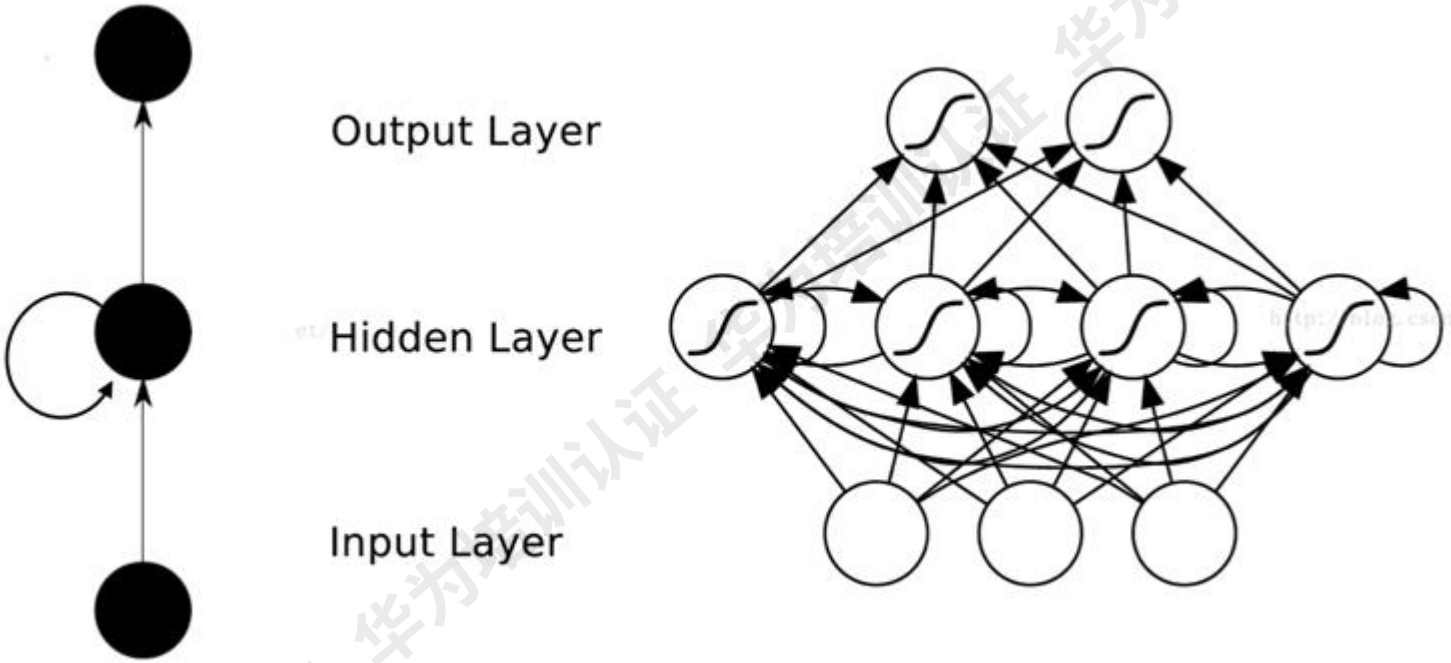
- Pooling的中文名为池化，它合并了附近的单元，减小了下层输入的尺寸。常用的Pooling有Max Pooling和Average Pooling，顾名思义，Max Pooling选择一小片正方形区域中最大的那个值作为这片小区域的代表，而Average Pooling则使用这篇小区域的均值代表之。这片小区域的边长为池化窗口尺寸。下图演示了池化窗口尺寸为2的一般Max池化操作。



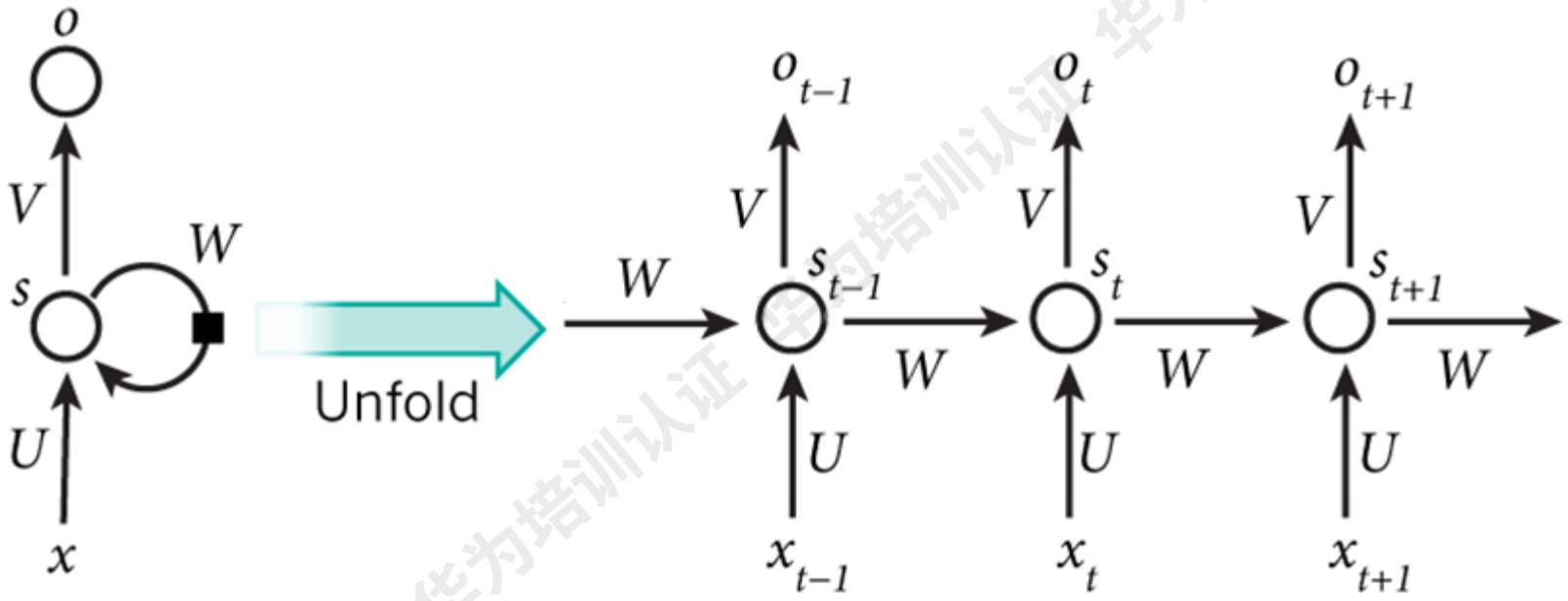
循环神经网络 (1)

- 循环神经网络（Recurrent neural networks，简称RNN）是一种通过隐藏层节点周期性的连接，来捕捉序列化数据中动态信息的神经网络，可以对序列化的数据进行分类。
- 和其他前向神经网络不同，RNN可以保存一种上下文的状态，甚至能够在任意长的上下文窗口中存储、学习、表达相关信息，而且不再局限于传统神经网络在空间上的边界，可以在时间序列上有延拓，直观上讲，就是本时间的隐藏层和下一时刻的隐藏层之间的节点间有边。
- RNN广泛应用在和序列有关的场景，如如一帧帧图像组成的视频，一个个片段组成的音频，和一个个词汇组成的句子。

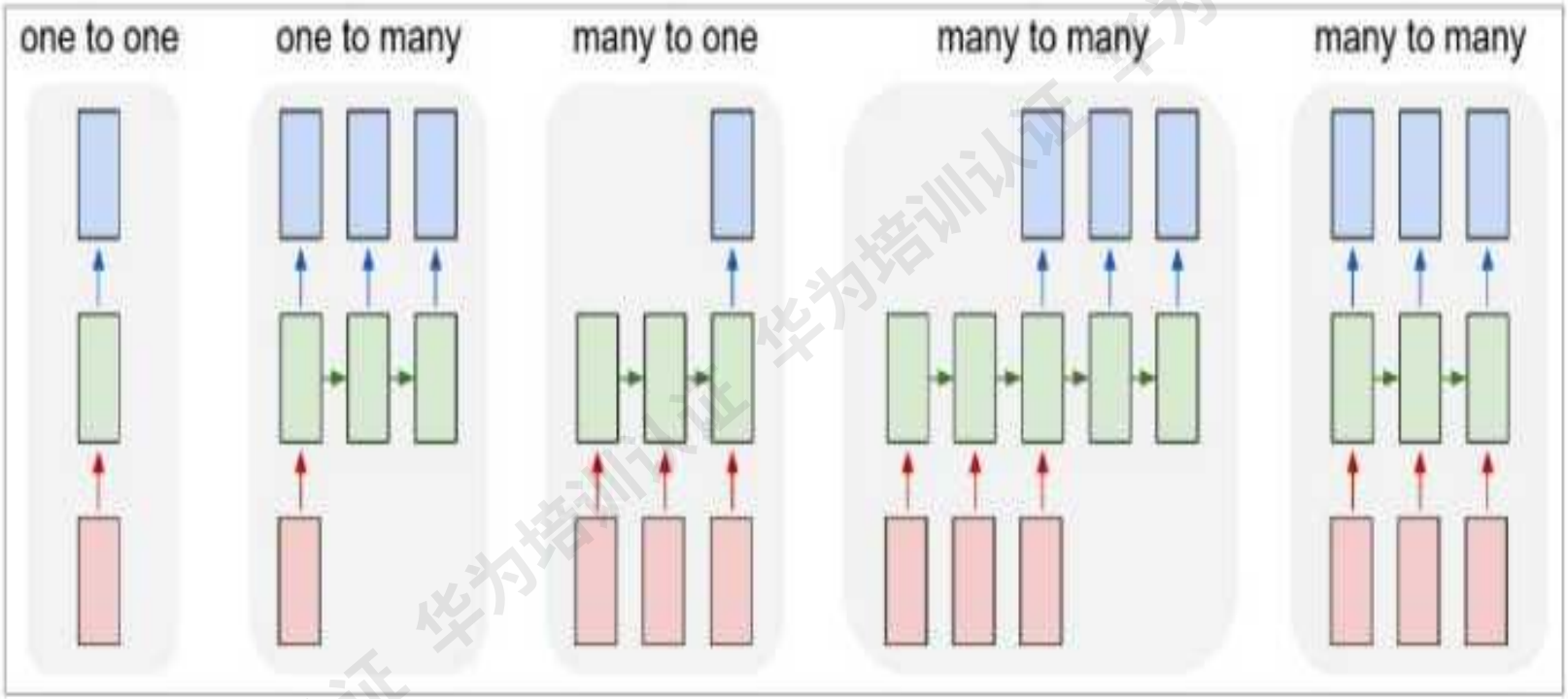
循环神经网络 (2)



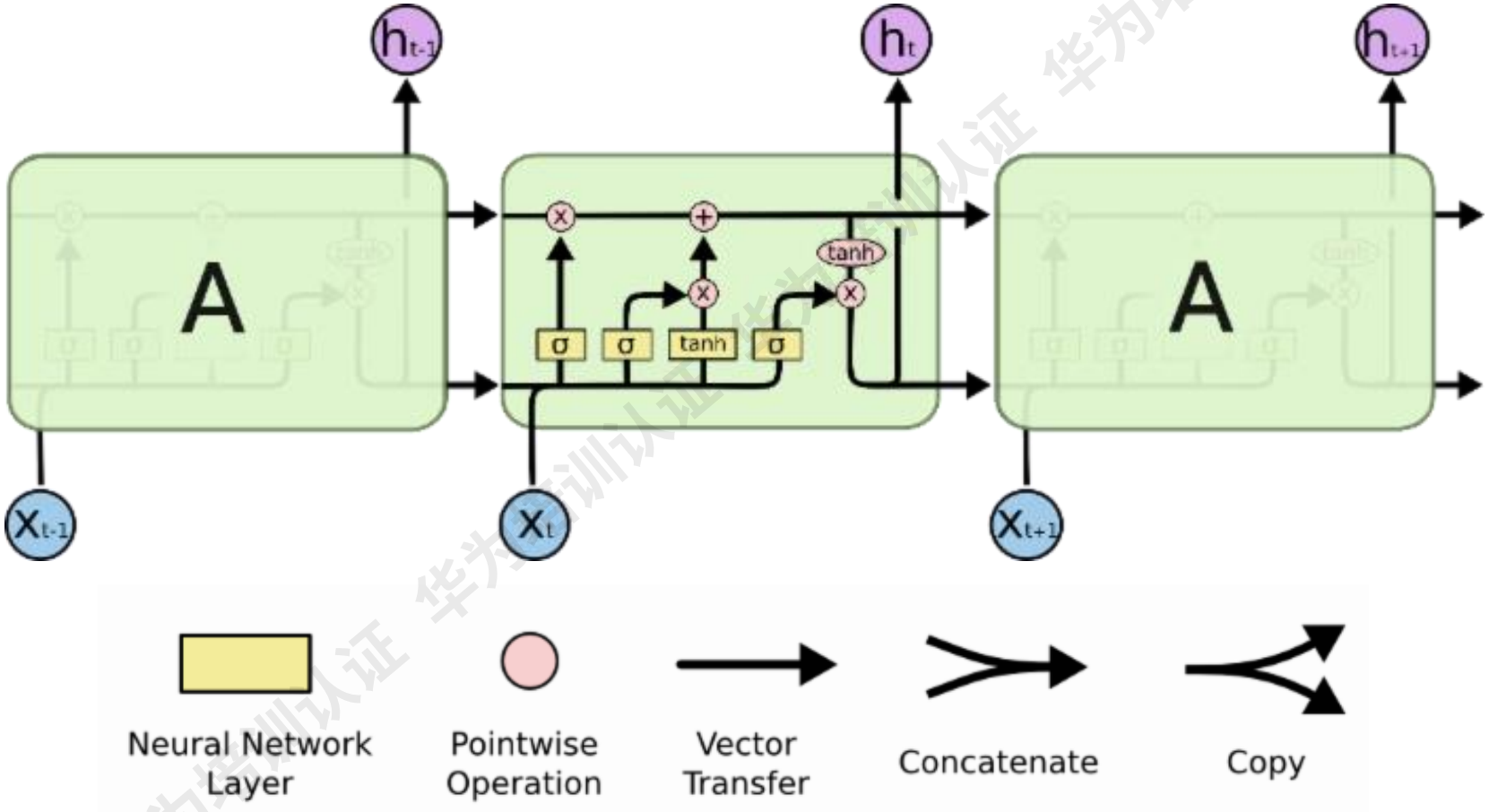
展开的循环神经网络



循环神经网络类型



标准LSTM结构



HUAWEI

目录

深度学习中的正则化

- 正则化是机器学习中非常重要并且非常有效的减少泛化误差的技术，特别是在深度学习模型中，由于其模型参数非常多非常容易产生过拟合。因此研究者也提出很多有效的技术防止过拟合，比较常用的技术包括：
  - 参数添加约束，例如L_1、L_2范数等。
  - 训练集合扩充，例如添加噪声、数据变换等。
  - Dropout

参数惩罚

- 许多正则化方法通过对目标函数J添加一个参数惩罚$\Omega(\theta)$，限制模型的学习能力。我们将正则化后的目标函数记为$\tilde{J}$。

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta),$$

其中$\alpha\epsilon[0,\infty)$是权衡范数惩罚项Ω和标准目标函数$J(X;\theta)$相对贡献的超参数。将α设为0表示没有正则化。α越大，对应正则化惩罚越大。

L_2正则

- 参数约束添加L_2范数惩罚项，该技术用于防止过拟合。

$$\tilde{J}(w;X,y) = J(w;X,y) + \frac{1}{2}\alpha\|w\|^2,$$

通过最优化技术，例如梯度相关方法可以很快推导出，参数优化方式为

$$w = (1-\varepsilon\alpha)\omega - \varepsilon\nabla J(w),$$

其中ε为学习率，相对于正常的梯度优化公式，对参数乘上一个缩减因子。

L_1正则

- 对模型参数添加L_1范数约束，即

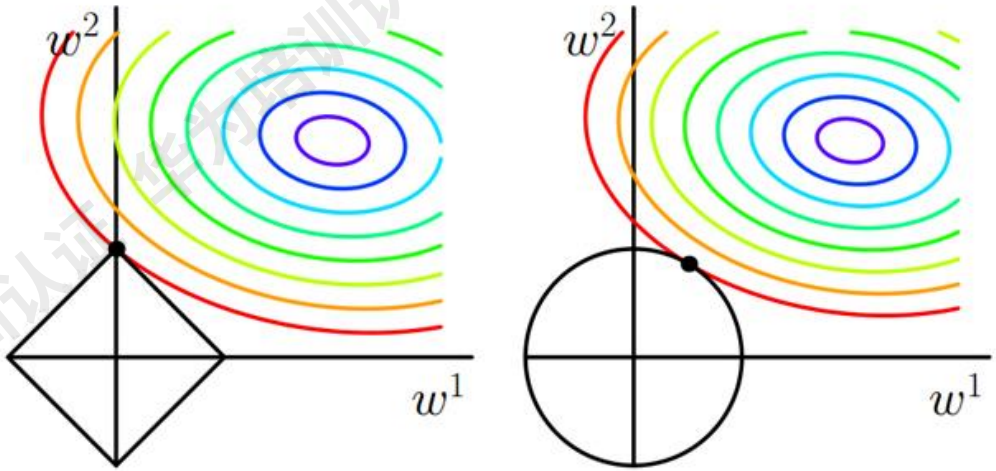
$$\tilde{J}(w;X,y) = J(w;X,y) + \alpha\|w\|_1,$$

- 如果通过梯度方法进行求解时，参数梯度为

$$\nabla\tilde{J}(w) = \propto sign(w) + \nabla J(w).$$

L_2 VS L_1

- L_2与L_1的主要区别如下：
  - 通过上面的分析，L_1相对于L_2能够产生更加稀疏的模型，即当L_1正则在参数w比较小的情况下，能够直接缩减至0，因此可以起到特征选择的作用。
  - 如果从概率角度进行分析，很多范数约束相当于对参数添加先验分布，其中L_2范数相当于参数服从高斯先验分布；L_1范数相当于拉普拉斯分布。

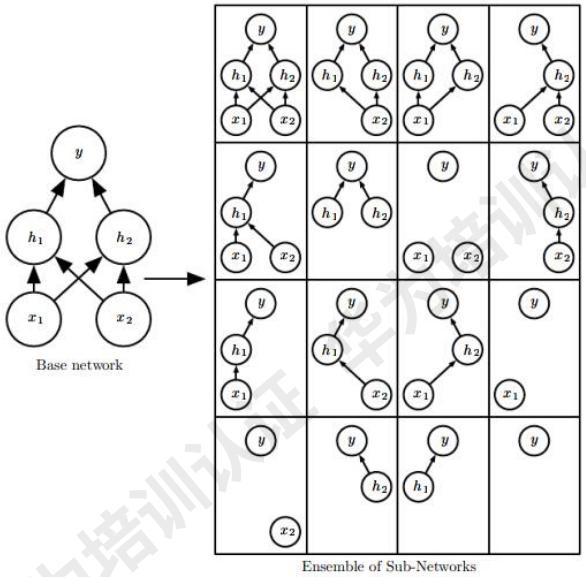


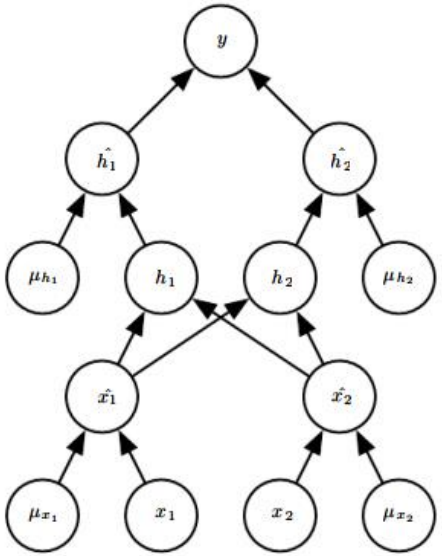
数据集合扩充

- 防止过拟合最有效的方法是增加训练集合，训练集合越大过拟合概率越小。数据集合扩充是一个省时有效的方法，但是在不同领域方法不太通用。
  - 在目标识别领域常用的方法是将图片进行旋转、缩放等（图片变换的前提是通过变换不能改变图片所属类别，例如手写数字识别，类别6和9进行旋转后容易改变类目）。
  - 语音识别中对输入数据添加随机噪声。
  - NLP中常用思路是进行近义词替换。
  - 噪声注入，可以对输入添加噪声，也可以对隐藏层或者输出层添加噪声。例如对于softmax 分类问题可以通过 Label Smoothing技术添加噪声，对于类目0-1添加噪声，则对应概率变成$\frac{\varepsilon}{k}$，$1-\frac{k-1}{k}\varepsilon$。

Dropout

- Dropout是一类通用并且计算简洁的正则化方法，在2014年被提出后广泛的使用。简单的说，Dropout在训练过程中，随机的丢弃一部分输入，此时丢弃部分对应的参数不会更新。相当于Dropout是一个集成方法，将所有子网络结果进行合并，通过随机丢弃输入可以得到各种子网络。例如





目录

HUAWEI

优化器

- 在梯度下降算法中，有各种不同的改进版本。在面向对象的语言实现中，往往把不同的梯度下降算法封装成一个对象，称为**优化器**。
- 算法改进的**目的**，包括但不限于：
  - 加快算法收敛速度；
  - 尽量避过或冲过局部极值；
  - 减小手工参数的设置难度，主要是Learning Rate（LR）。
- 常见的优化器如：普通GD优化器、动量优化器、Nesterov、**Adagrad**、**Adadelta**、**RMSprop**、Adam、 AdaMax、Nadam。

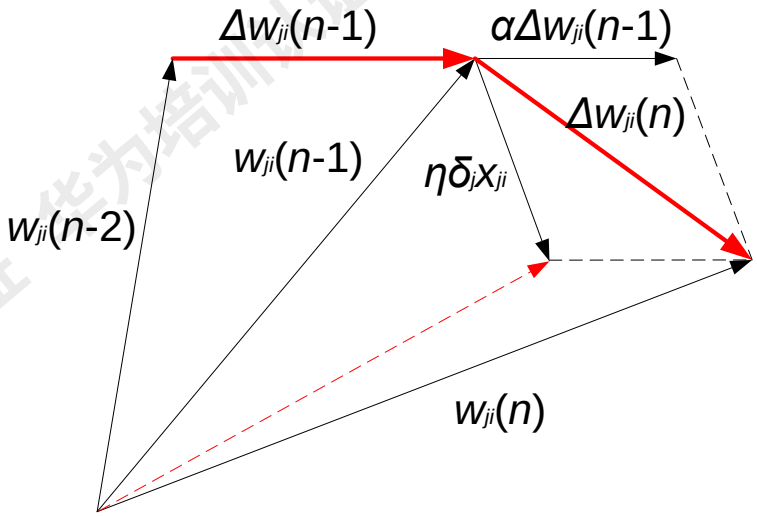
动量优化器

- 一个最基本的改进，是为Δw_{ji}增加动量项。记第n次迭代时的权值修正量为$\Delta w_{ji}(n)$，则权值修正法则变为：

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

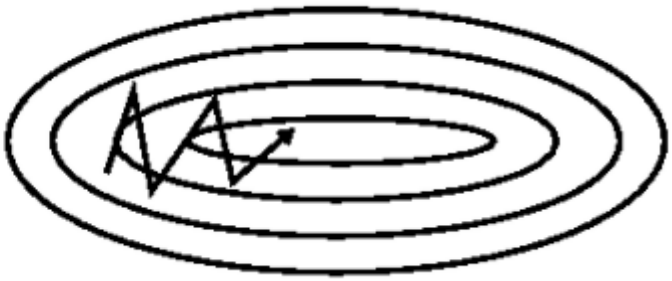
其中，0≤α<1是一个常数，称为动量（Momentum）。$\alpha \Delta w_{ji}(n-1)$称为动量项。

- 想象一个小球，从一个随机的点开始，沿着误差曲面滚下。动量项的引入相当于赋予了小球惯性：



动量优化器优缺点

- 动量优化器的优点是：
  - 增加了梯度修正方向的稳定性，减小突变。
  - 在梯度方向比较稳定的区域，小球滚动会越来越快（当然，因为$\alpha < 1$，其有一个速度上限），这有助于小球快速冲过平坦区域，加快收敛。
  - 带有惯性的小球更容易滚过一些狭窄的局部极值。
- 动量优化器的缺点是：
  - 学习率η以及动量α仍需手动设置，这往往需要较多的实验来确定合适的值



Adam优化器 (1)

- Adam（Adaptive Moment Estimation）：是从Adagrad、Adadelta上发展而来，Adam为每个待训练的变量，维护了两个附加的变量m_t和v_t：

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

其中t表示第t次迭代，g_t是本次计算出的梯度，从形式上来看m_t和v_t分别是梯度和梯度平方的移动均值。从统计意义上看，m_t和v_t是梯度的一阶矩（均值）和二阶矩（非中心方差）的估计，因此而得名。

Adam优化器 (2)

- 如果以0向量来初始化m_t和v_t，在开始的一些迭代，尤其是当β_1和β_2接近于1的时候，m_t和v_t将非常接近于0，为了解决这个问题，我们实际使用的是$\hat{m}_t$和$\hat{v}_t$：

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

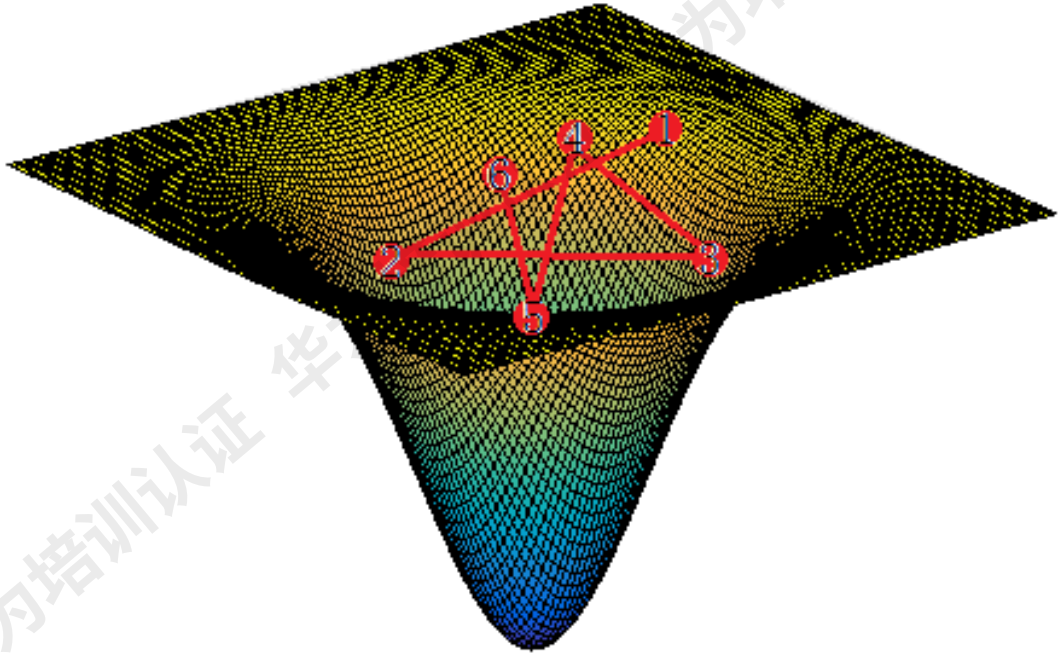
- 而Adam的权值更新法则是：

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

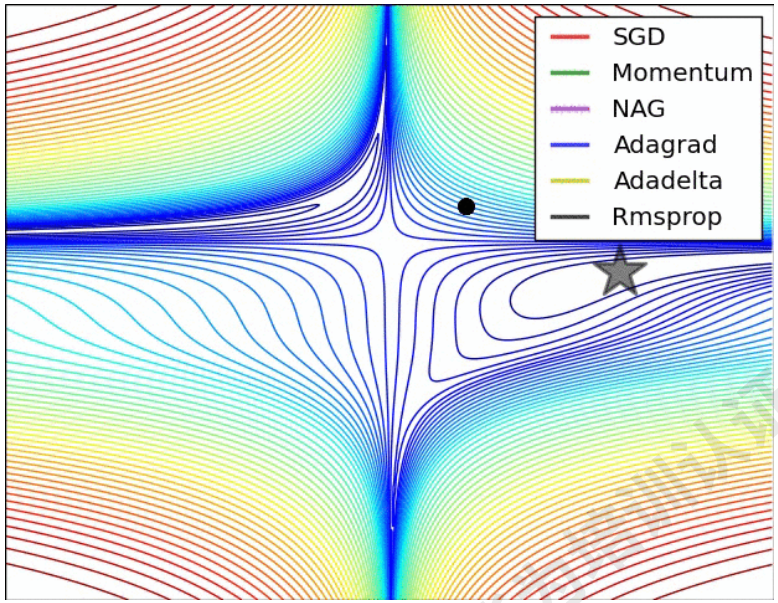
- 虽然上面的法则中依然存在人工设置η、β_1、β_2，但他们的设置难度大大降低。根据实验，一般取$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$，η=0.001。在实际使用过程中，Adam将迅速收敛，当收敛到饱和的时候，可以适当降低η，一般降低几次之后，即会收敛到满意的（局部）极值。其他参数一般不必调整。

Adam优化器 (3)

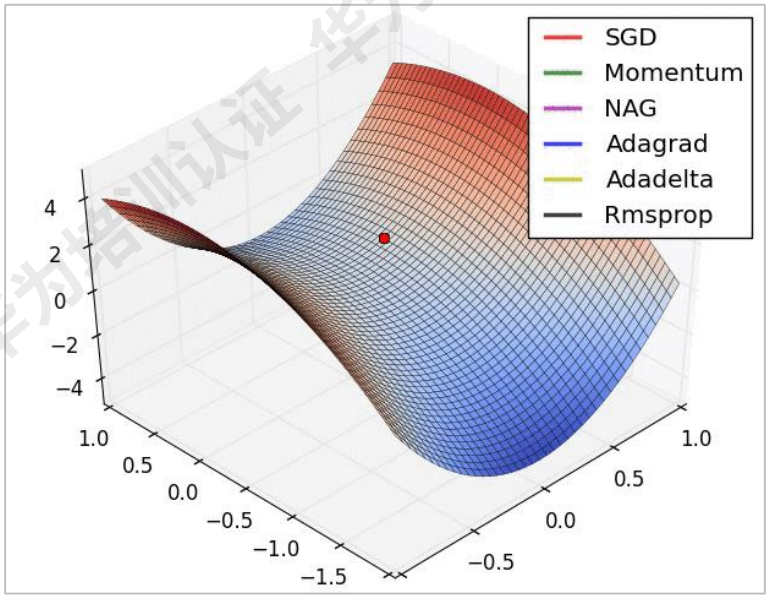
- 理解Adam：可以把$\hat{m}_t$看成是带动量项的当前梯度，而$\frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}$是等效的当前学习率。如果过去一段时间，梯度的模都很大，则权值可能在极值附近打转而难以收敛，此时应该降低学习率来促成收敛。



各优化器可视化比较



损失函数等高图上的各优化算法比较



鞍点处各优化算法比较

目录

HUAWEI

深度学习的应用

计算机视觉

- 图像分类
- 场景识别
- 人脸检测&识别
- 物体检测&识别
- 语义分割
- 风格化
- 超分辨率
- OCR
- ……

图像分类

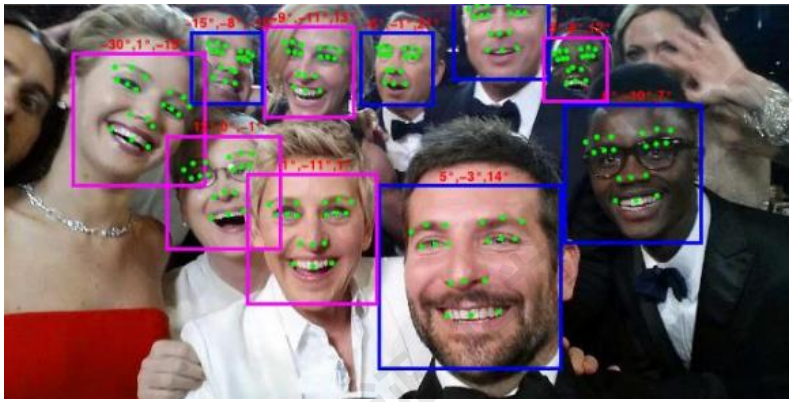
- 将图像映射到不同的类别集合，可用于图片检索、图片归档等。

场景识别

- 针对场景和环境的图片分类，可用于情景感知、拍照智能3A等。

人脸检测与人脸识别

- 人脸检测：发现并定位图片中的人脸及人脸特征点，可用于人脸对焦、美颜、增强现实等。
- 人脸识别：区分不同的人，可用于身份认证、隐私保护等。



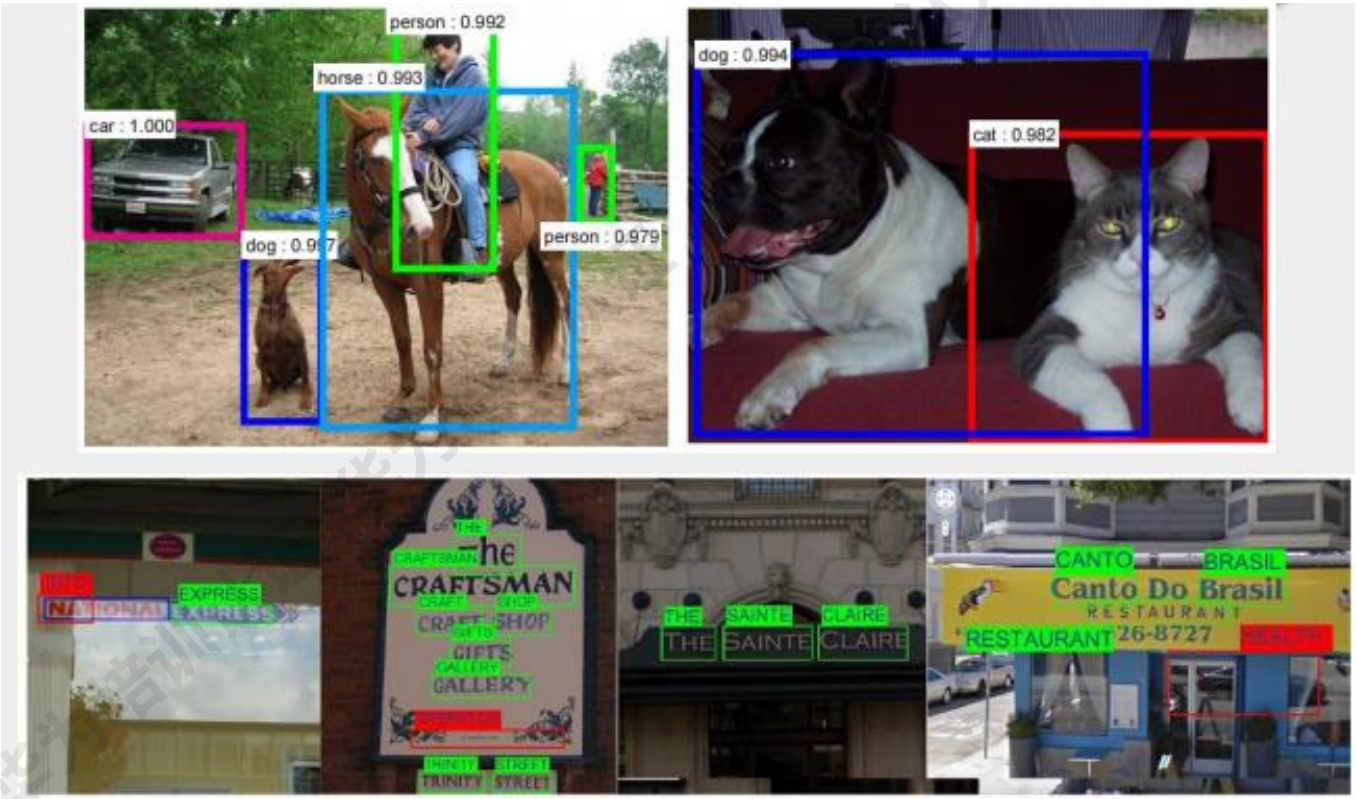
人脸检测

人脸识别

HUAWEI

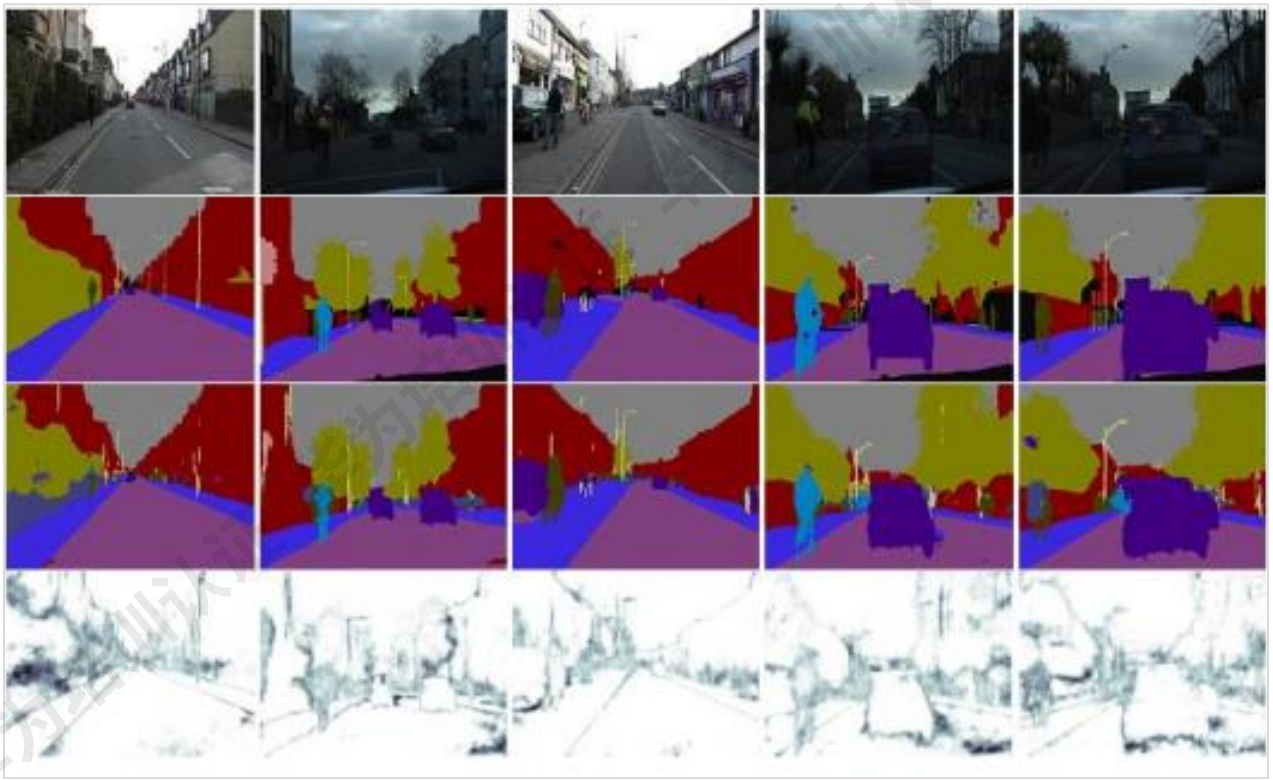
物体检测&识别

- 从图片中发现、定位并识别不同物体，包括数字文字检测、行人检测等可用于OCR、无人驾驶、图像智能裁剪等。



语义分割

- 预测图片中每个像素的标签，通常由分割和识别组成。可用于无人驾驶、增强现实、情景感知等。

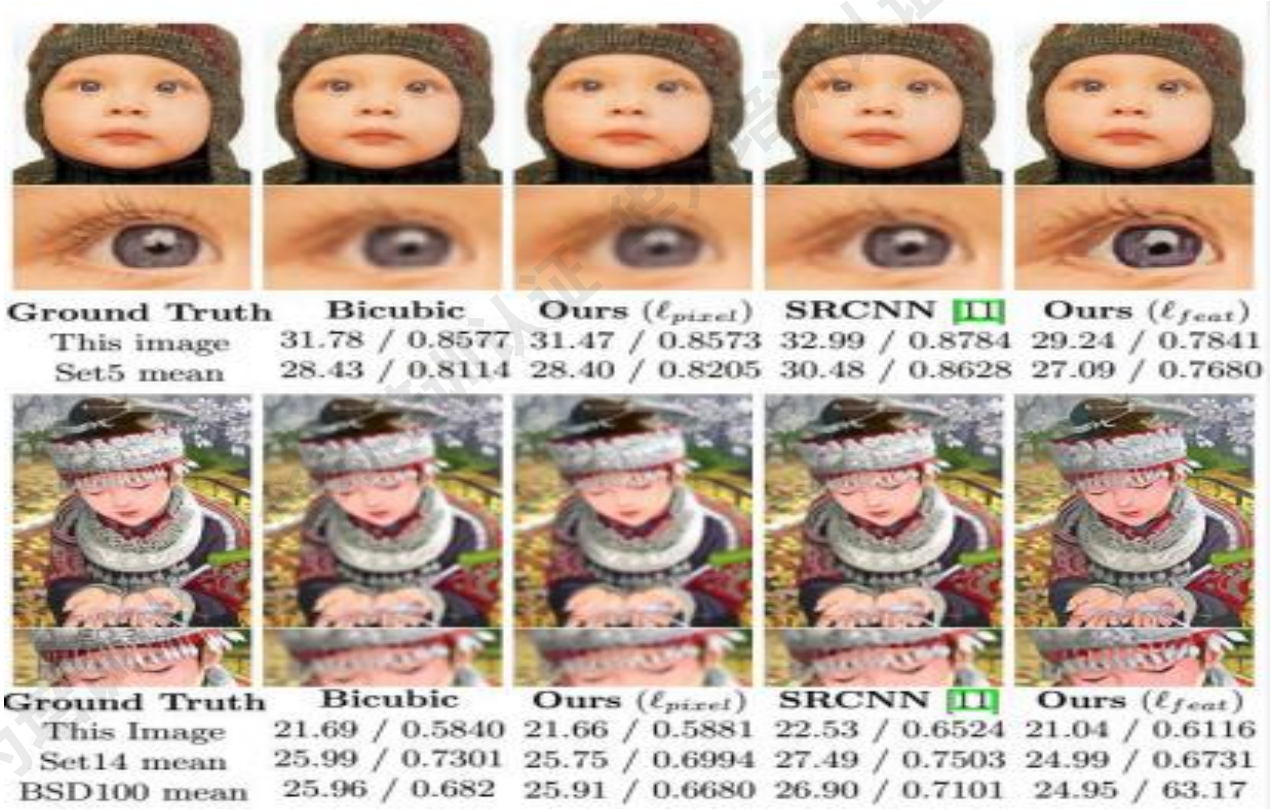


风格化

- 在保留图像内容的基础上改变图像风格，可用于图片艺术化处理等。

超分辨率

- 从低分辨率图像生成高分辨率图像，可用于照片处理、安防监控、医学图像等。



Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This image	31.78 / 0.8577	31.47 / 0.8573	32.99 / 0.8784	29.24 / 0.7841
Set5 mean	28.43 / 0.8114	28.40 / 0.8205	30.48 / 0.8628	27.09 / 0.7680

Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [11]	Ours (ℓ_{feat})
This Image	21.69 / 0.5840	21.66 / 0.5881	22.53 / 0.6524	21.04 / 0.6116
Set14 mean	25.99 / 0.7301	25.75 / 0.6994	27.49 / 0.7503	24.99 / 0.6731
BSD100 mean	25.96 / 0.682	25.91 / 0.6680	26.90 / 0.7101	24.95 / 63.17

OCR

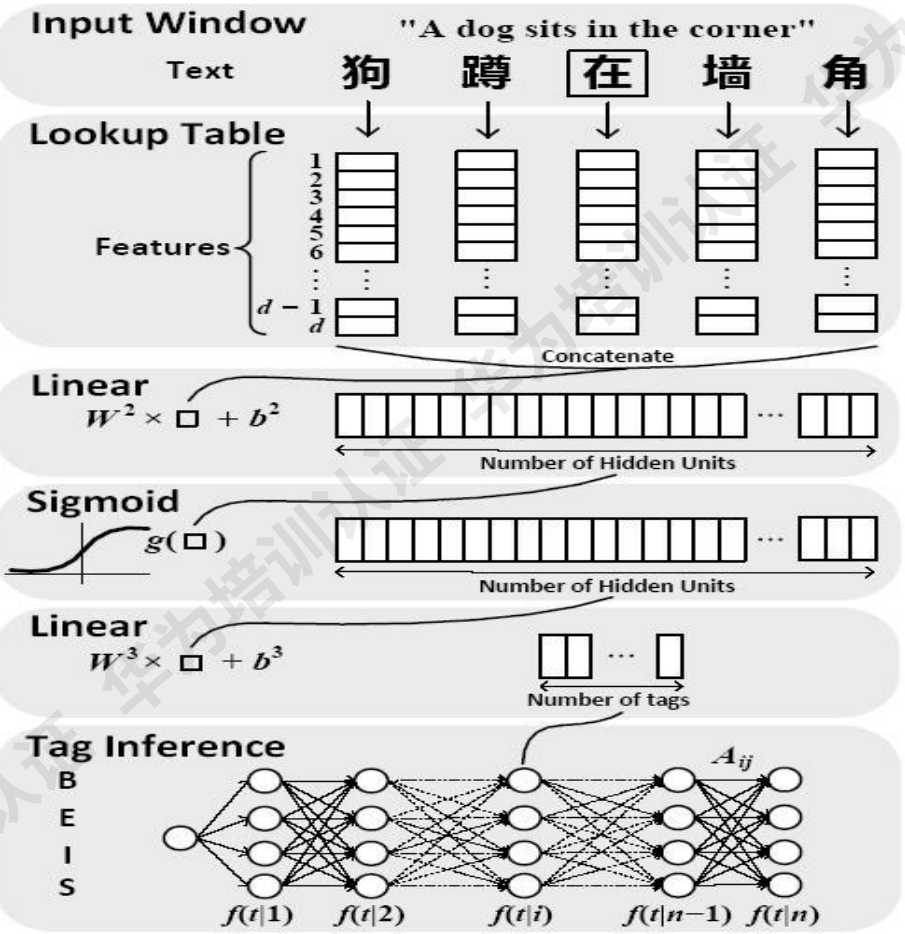
- 识别图片中的数字、文字等信息，用于图片或实体文档的数字化，如名片自动识别，票据扫描等。

自然语言处理

- 中文分词
- 知识挖掘
- 机器翻译
- 情感分析

中文分词算法

中文分词

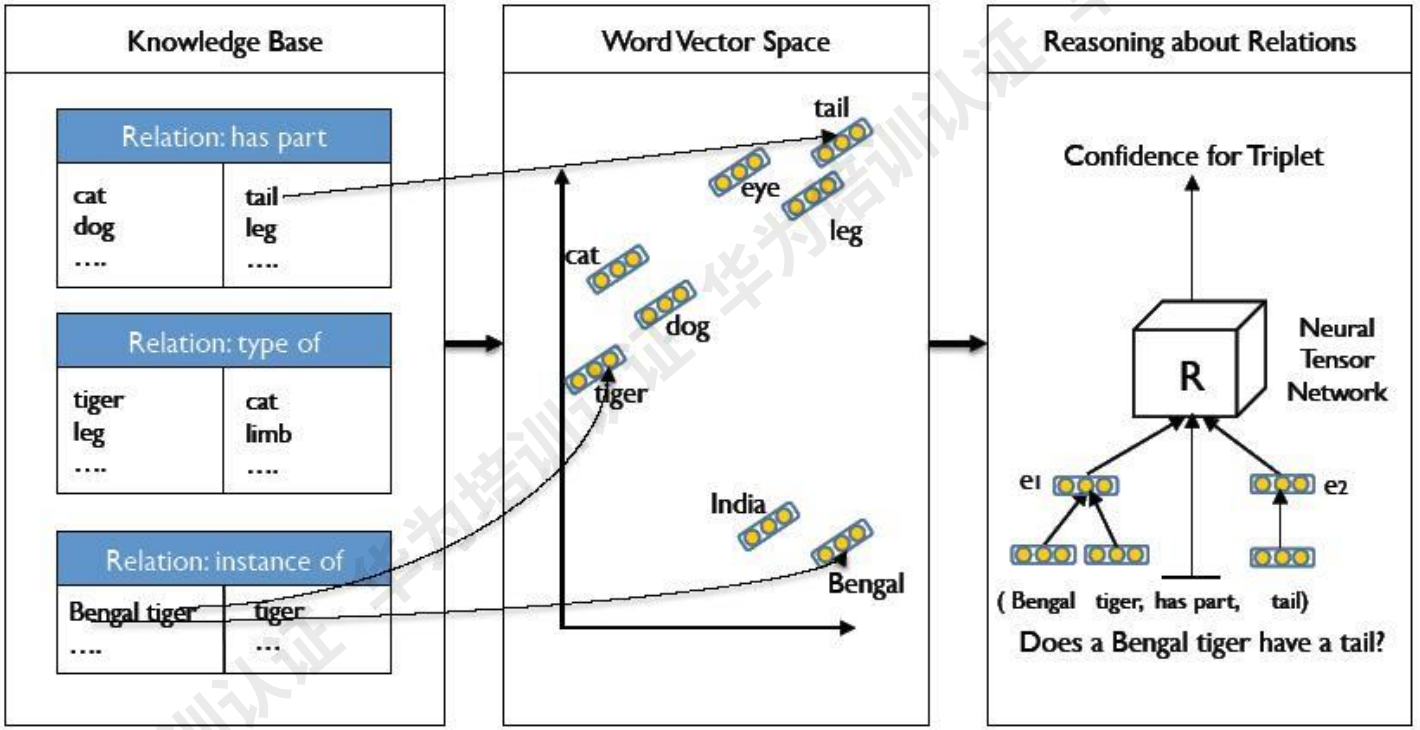


HUAWEI

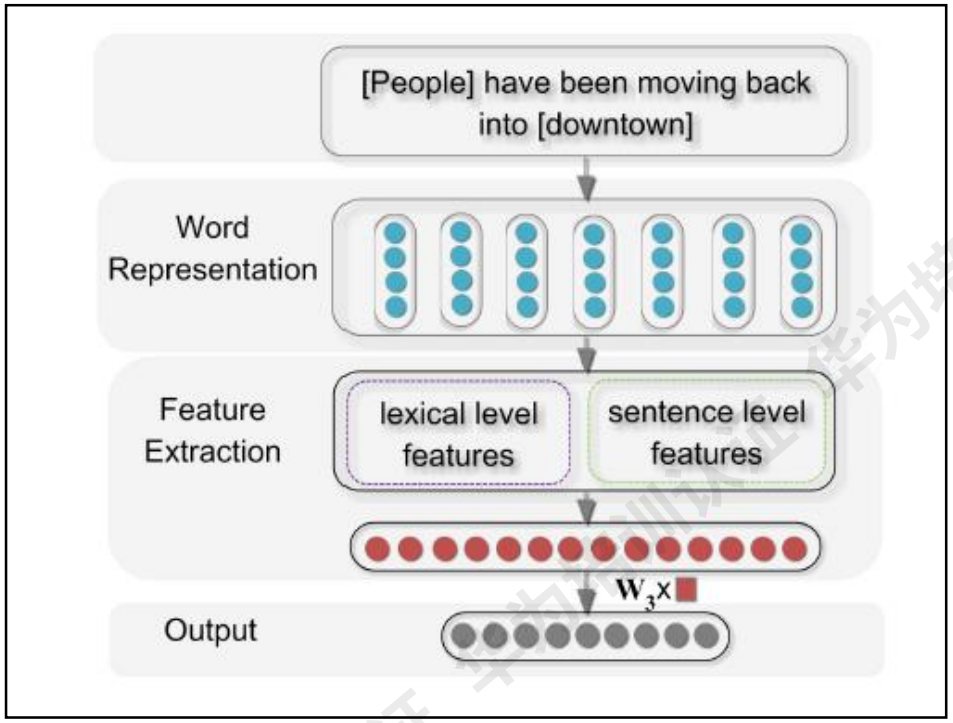
知识挖掘

- 两大类问题
  - 现有知识库的新知识推理
    - CYC,WordNet,FreeNet……
    - 目前的文献做法大思路基本一致
      - 已知实体用Word Embedding表示
      - 实体关系用Tensor Network建模
      - 后向传播+SGD训练
  - 从自由文本中挖掘结构化知识
    - 文本挖掘
    - TF-IDF

现有知识库的新知识推理



从自由文本中挖掘结构化知识 (1)



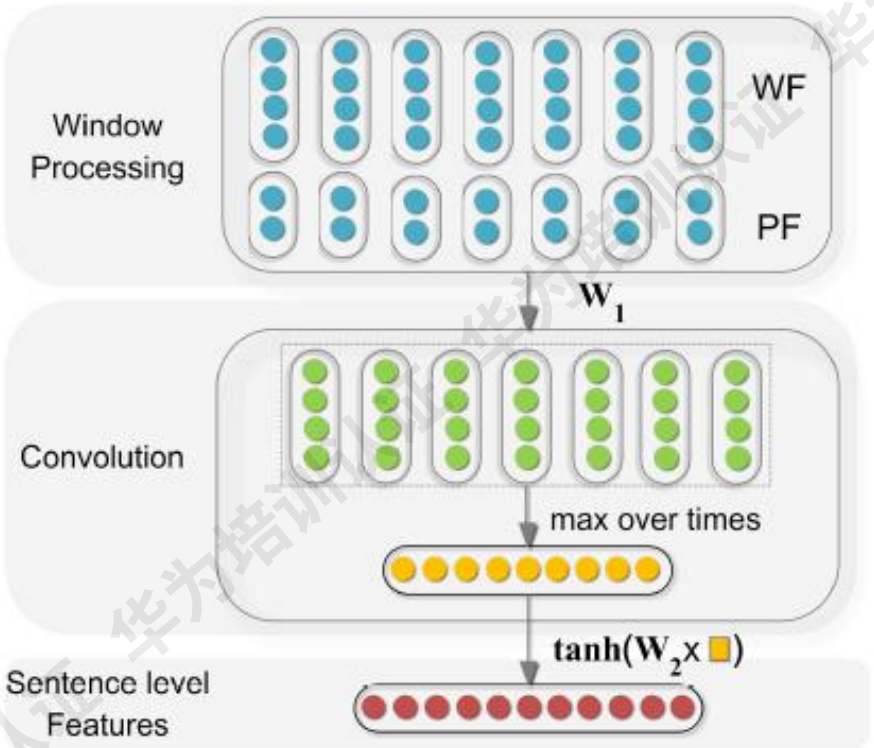
整体结构

Features	Remark
L1	Noun 1
L2	Noun 2
L3	Left and right tokens of noun 1
L4	Left and right tokens of noun 2
L5	WordNet hypernyms of nouns

词法级特征

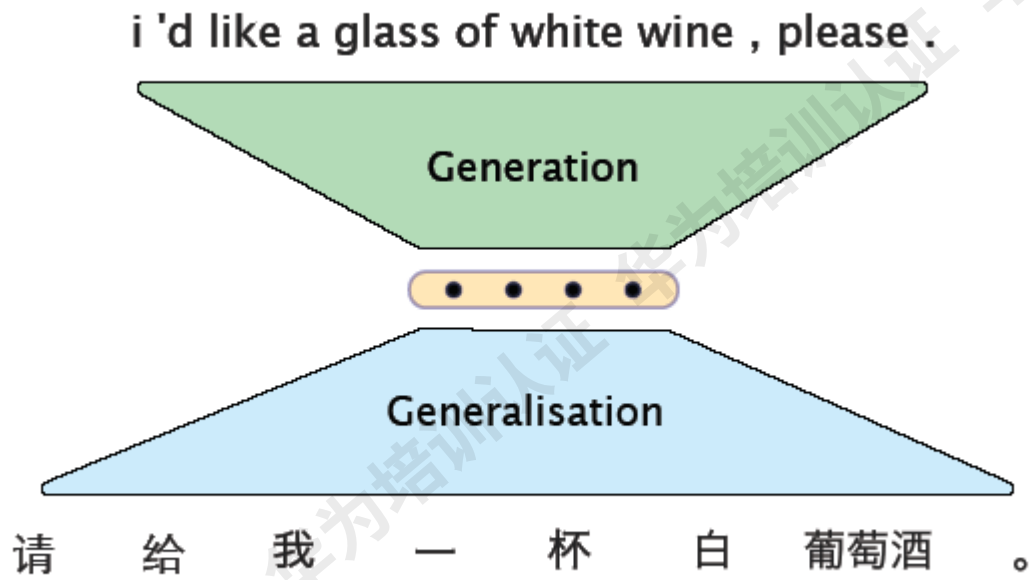
HUAWEI

从自由文本中挖掘结构化知识 (2)



句子级特征抽取：卷积网络

机器翻译（通用模型）



Decoder

语义向量

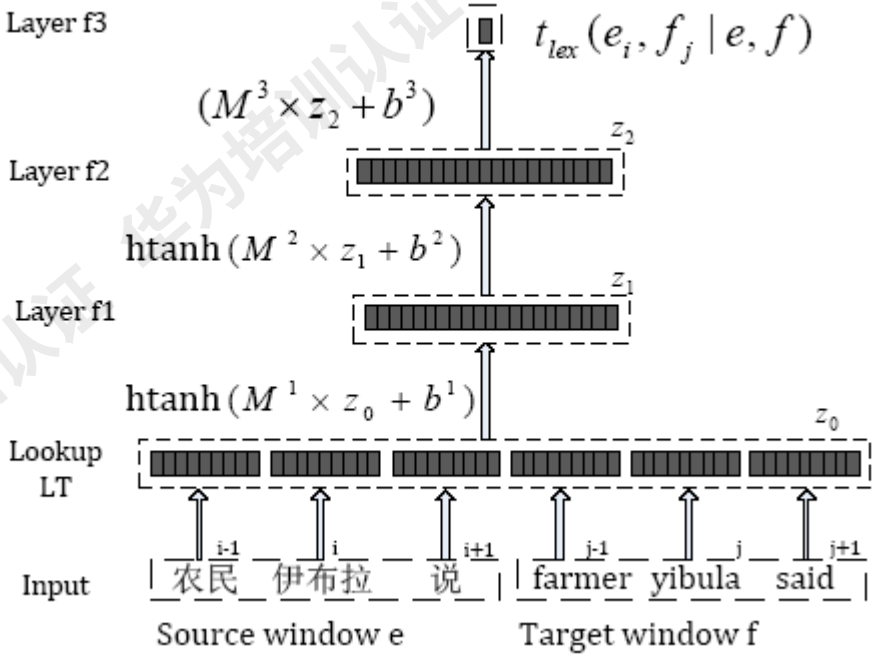
Encoder

最常见的通用模型：Encoder-Decoder Model

HUAWEI

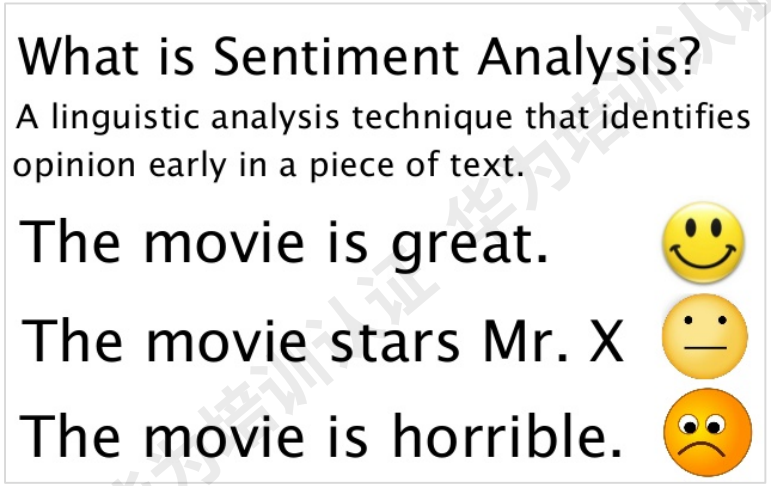
机器翻译 - 很多地方可以引入DL

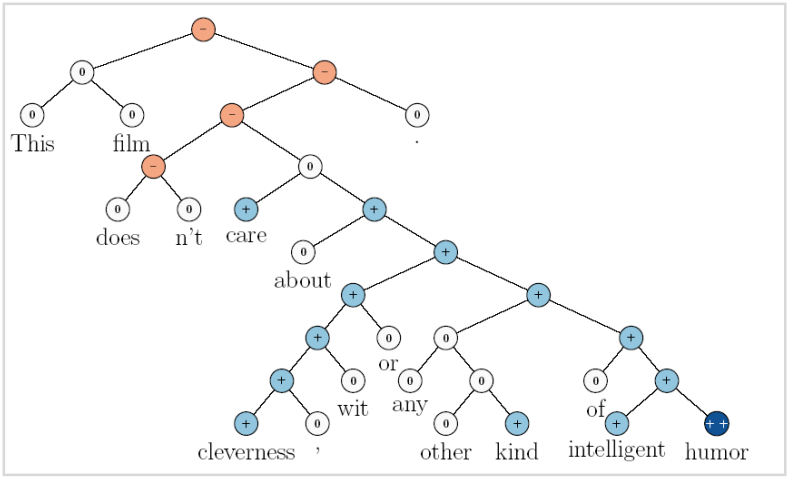
- 单词对齐
- 短语对齐
- 短语重排序
- 语言模型
- 翻译模型
- 联合模型
- 翻译结果重排序
- ……



情感分析

- 核心的两个问题
  - 句子级的Word Embedding表示
  - 如何将情感倾向编码到各级Word Embedding中
    - 半监督或者监督学习：通过训练过程将情感倾向编码到WE结构中





语音识别

推荐系统

HUAWEI

本章总结

- 本章主要介绍了深度学习预备知识，包括学习算法、机器学习常用算法、超参数和验证集、参数估计以及估计方法，进而讲述了神经网络的定义与发展、感知器及其训练法则、神经网络的种类等知识。

思考题

1. (单选) 不属于深度学习开发框架的是？（　）

   A. CNTK

   B. Keras

   C. BAFA

   D. MXNet

2. (判断题) GAN是一种深度学习模型，是近年来复杂分布上无监督学习最具前景的方法之一。（　）

   A. True

   B. False

思考题

3. (多选) 训练误差会降低模型的准确率，产生欠拟合，此时如何提升模型拟合度？（　　）

A. 增加数据量

B. 特征工程

C. 减少正则化参数

D. 增加特征

4. (判断题) 相比循环神经网络，卷积神经网络更适合处理图像识别问题。（　　）

A. True

B. False

学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

华为云EI概览

www.huawei.com

HUAWEI

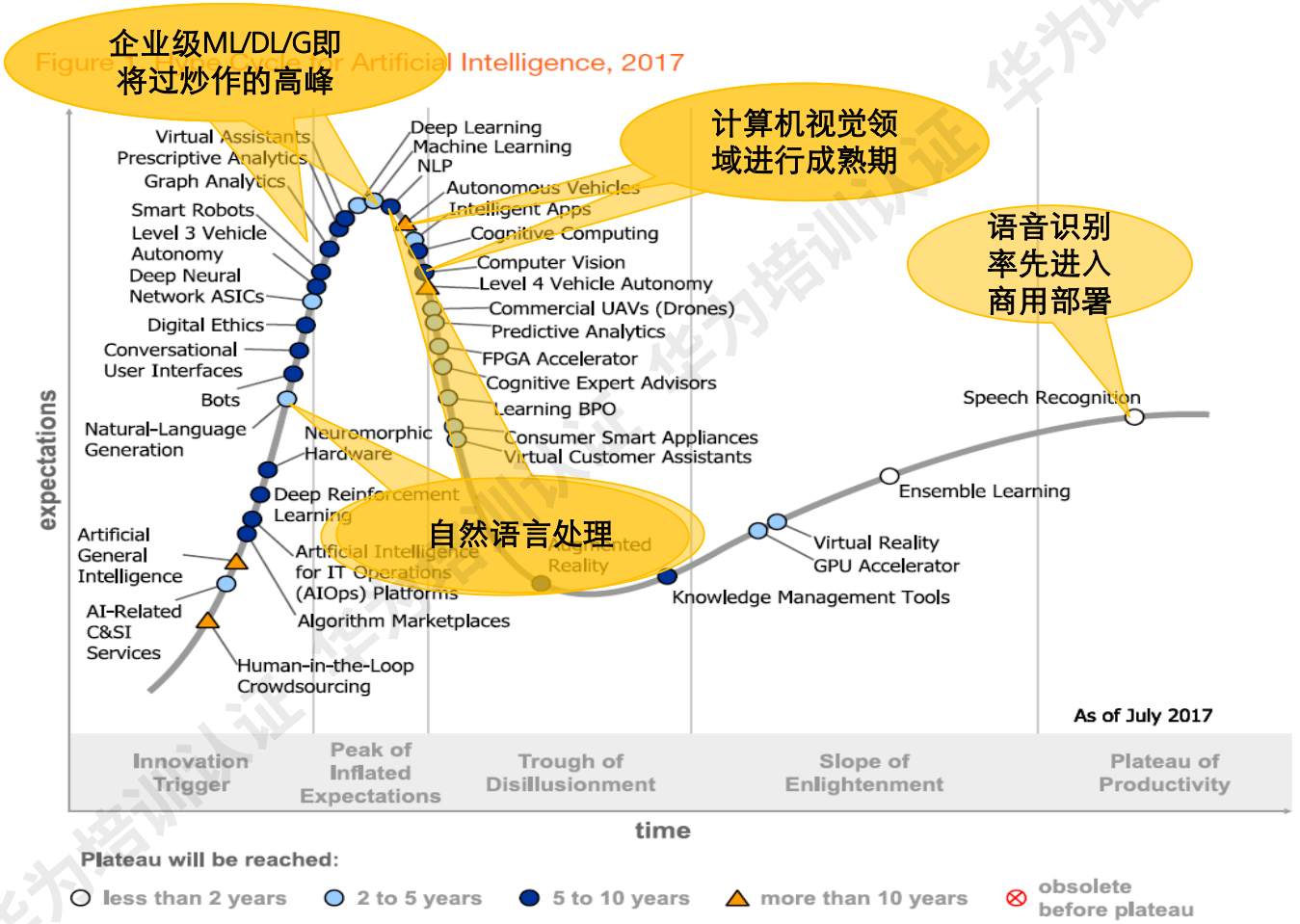
目标

- 学完本课程后，您将能够：
  - 了解华为云EI生态
  - 了解华为云EI服务产品能力
  - 了解华为云EI业务场景与解决方案

目录

HUAWEI

人工智能技术趋势



HUAWEI

华为AI发展思路

基础理论研究

- 诺亚方舟实验室：人工智能
- 罗素实验室：大数据
- 诺曼实验室：地图
- 香农实验室：算法

内部实践

- GTS：智能站勘、智能审核、网络设计、智能客服…
- 供应链：智能装车、路径优化、智能仓储…
- 终端公司：智能相册、风控、推荐、翻拍识别…
- 流程IT：智能运维、机器翻译…

外部赋能（EI企业智能）

- 异构计算：CPU、GPU、专用AI芯片…
- 通用服务：视觉、语音…
- 基础平台：机器学习、深度学习、图计算…
- 行业赋能：智能物流、智能问答、智能推荐…

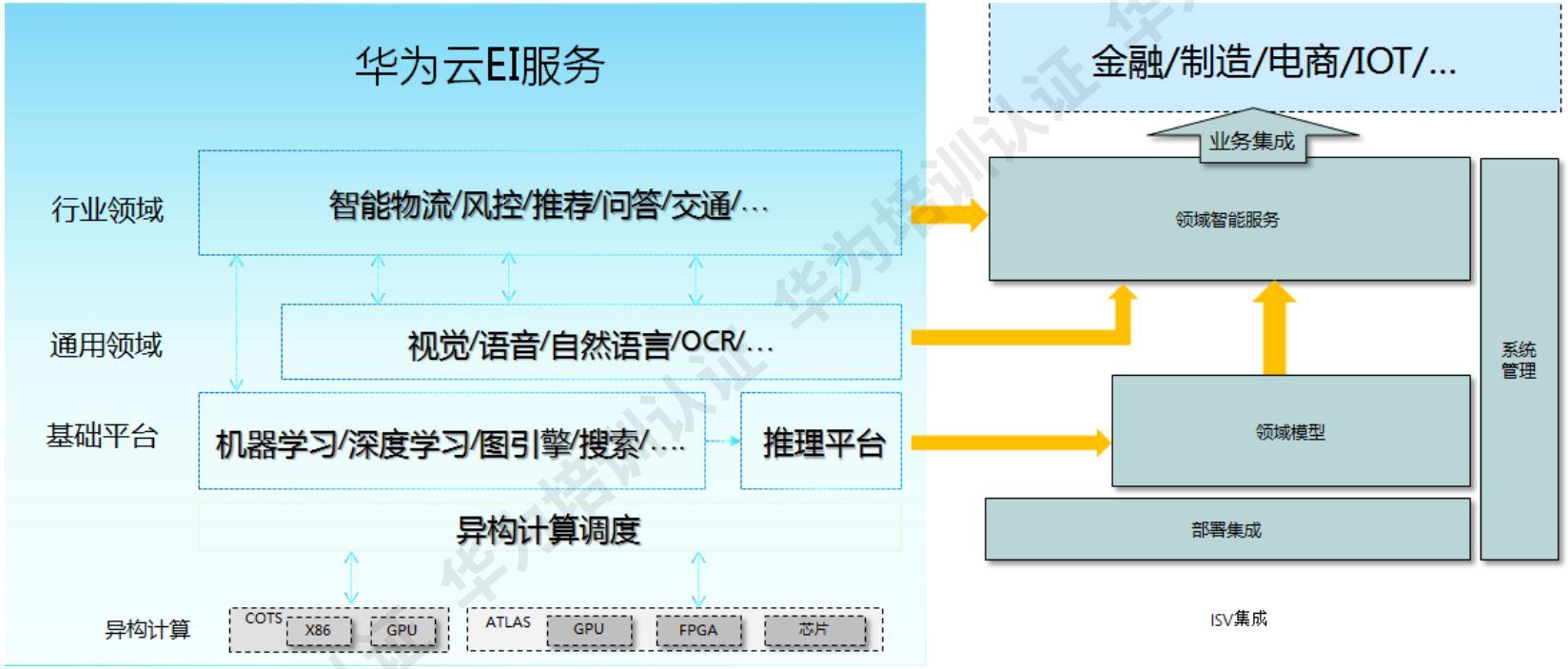
芯片

终端

云端

HUAWEI

华为云EI：让企业更智能



HUAWEI

华为云EI全球实践

60%
中国TOP 10金融企业

25%
全球TOP 50+电信运营商

30%
中国平安城市建设

HUAWEI

目录

华为云EI服务全景图

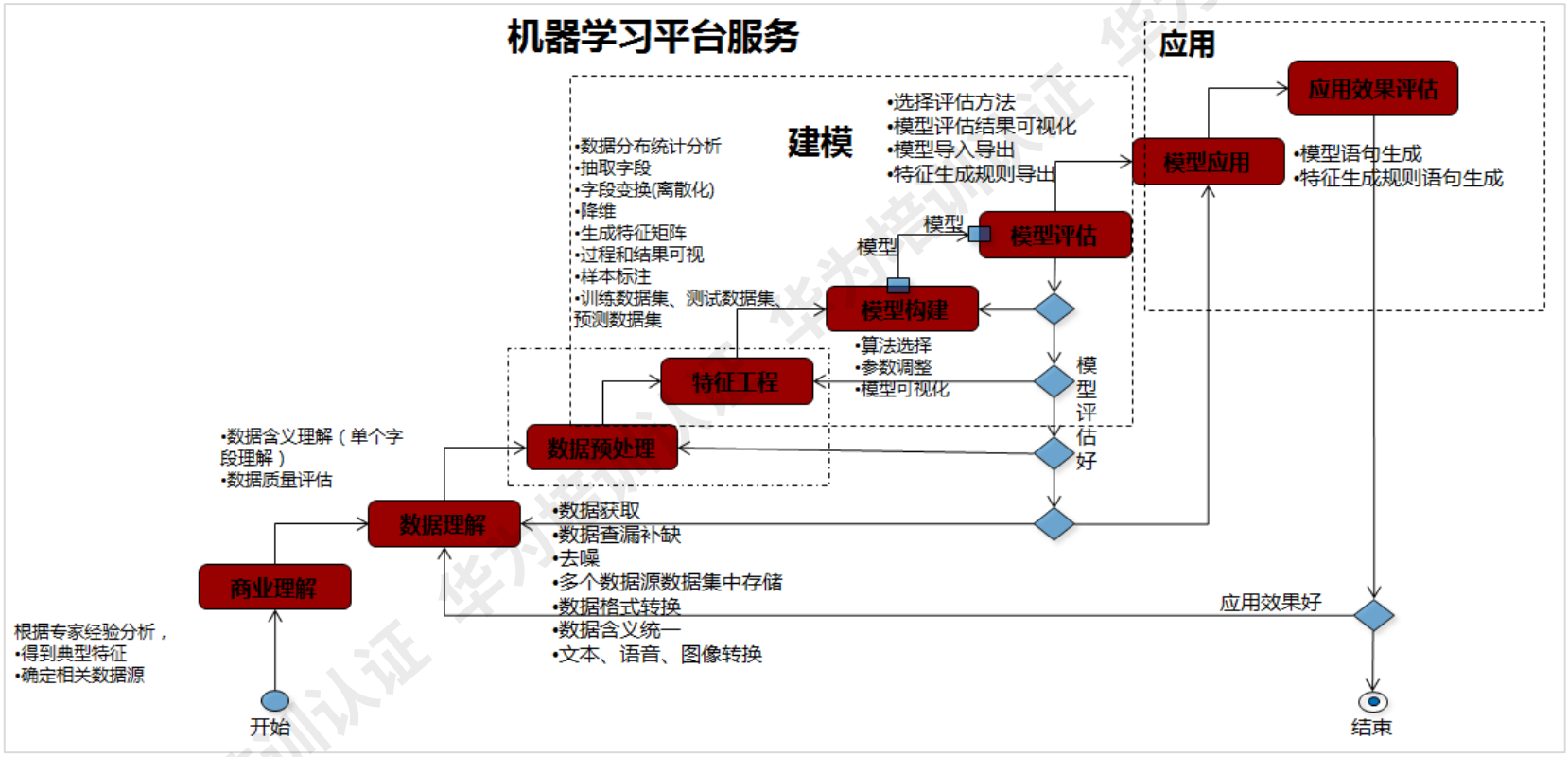
基础平台类服务 (1)

- **机器学习服务**（Machine Learning Service）：帮助用户通过机器学习技术快速发现数据规律和构建预测模型，并将其部署为预测分析解决方案。
- **深度学习服务**（Deep Learning Service）：是基于华为云强大高性能计算提供的一站式深度学习平台服务，内置大量优化的网络模型算法，以丰富、便捷、高效的品质帮助用户轻松使用深度学习技术，通过灵活调度按需服务化方式提供模型训练、评估与预测。
- **图引擎服务**（Graph Engine Service）：是针对以“关系” 为基础的“图”结构数据，进行查询、分析的服务。广泛应用于社交关系分析、推荐、精准营销、舆情及社会化聆听、信息传播、防欺诈等具有丰富关系数据的场景。

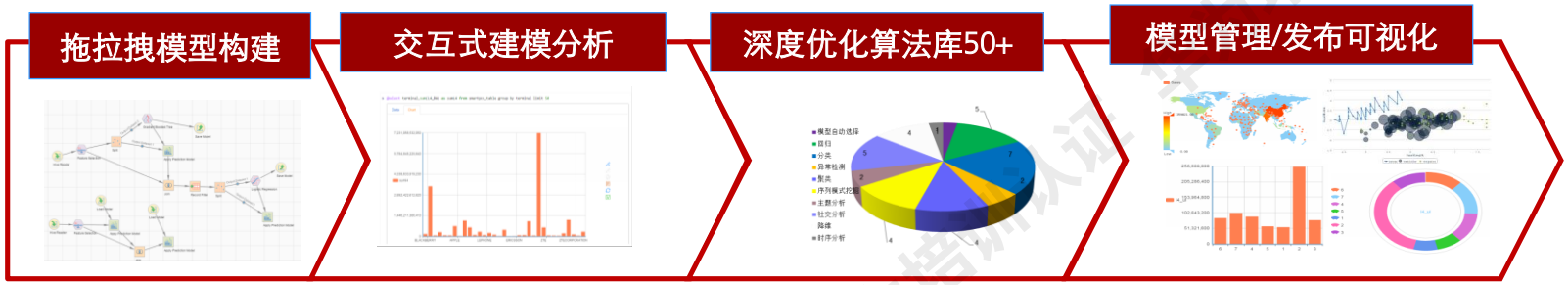
基础平台类服务 (2)

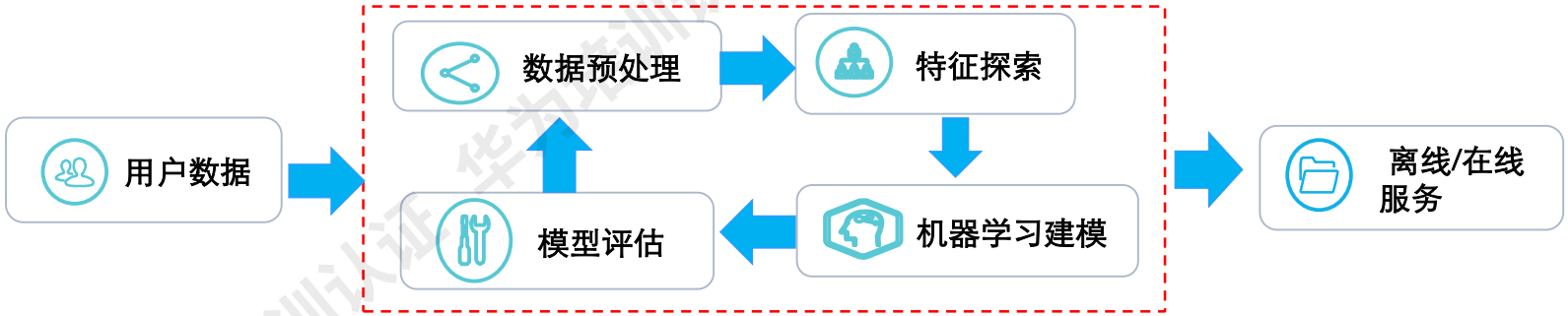
- **深度学习镜像服务（Deep Learning HMI）**：为数据科学家、深度学习从业者和研究科学家提供基础设施和工具以加速云计算上的深度学习工作。可以通过深度学习HMI快速启动华为ECS实例，预先安装流行的深度学习框架，方便用户训练复杂的自定义AI模型。深度学习HMI为您提供了TensorFlow、MXNet、CNTK、Caffe、Caffe2、Theano、PyTorch和Keras 。
- **批处理服务（Batch Service）**：是一种用于运行大规模并行批处理作业的分布式云计算服务。支持海量任务的自动调度、资源管理和数据加载，可广泛应用于深度学习、基因检测、视频分析等大规模并行计算场景。

MLS：解决特性到模型应用的完整过程

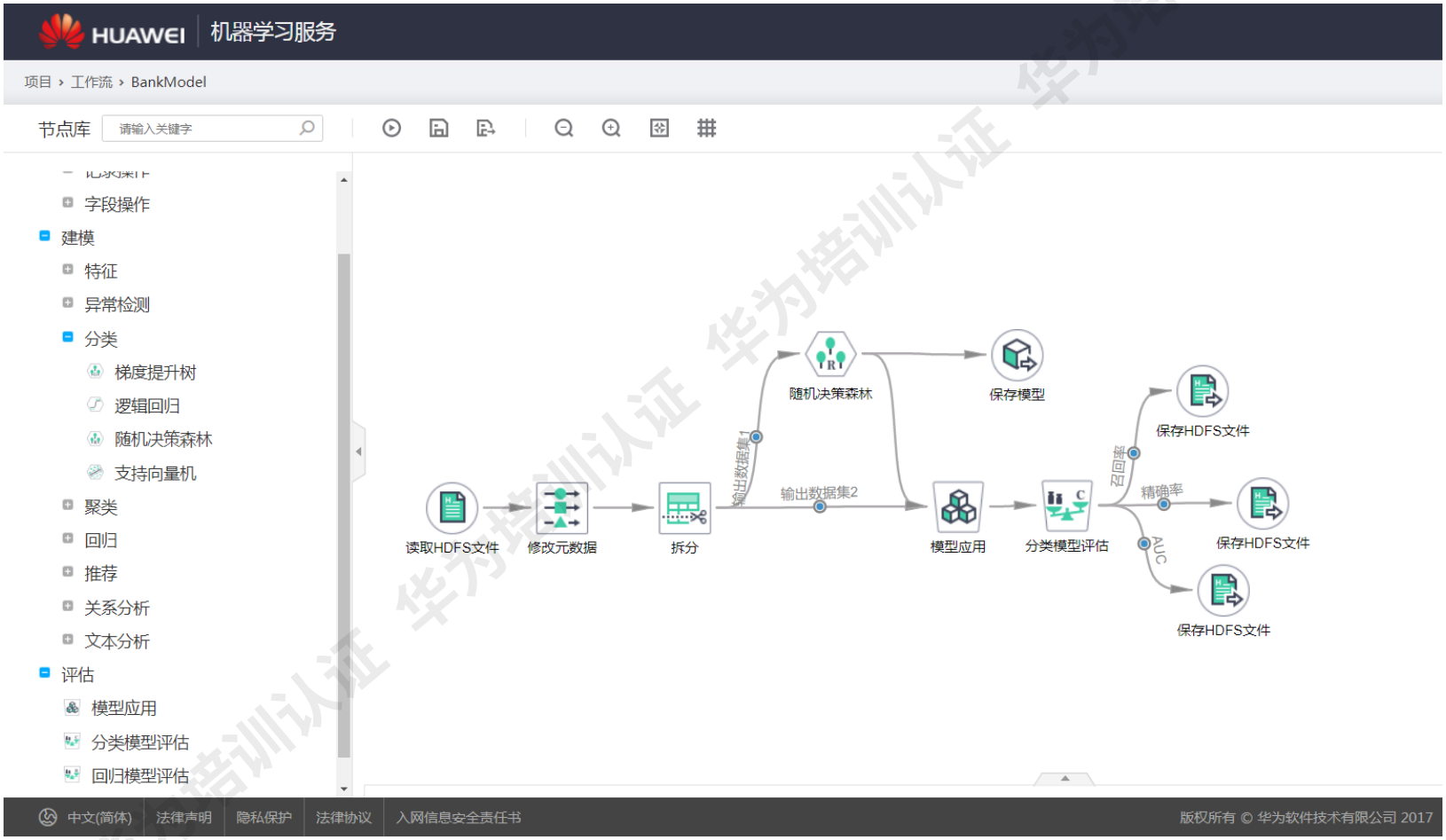


MLS：一站式平台支撑数据分析全流程





MLS：拖拉拽模型构建



MLS成功实践

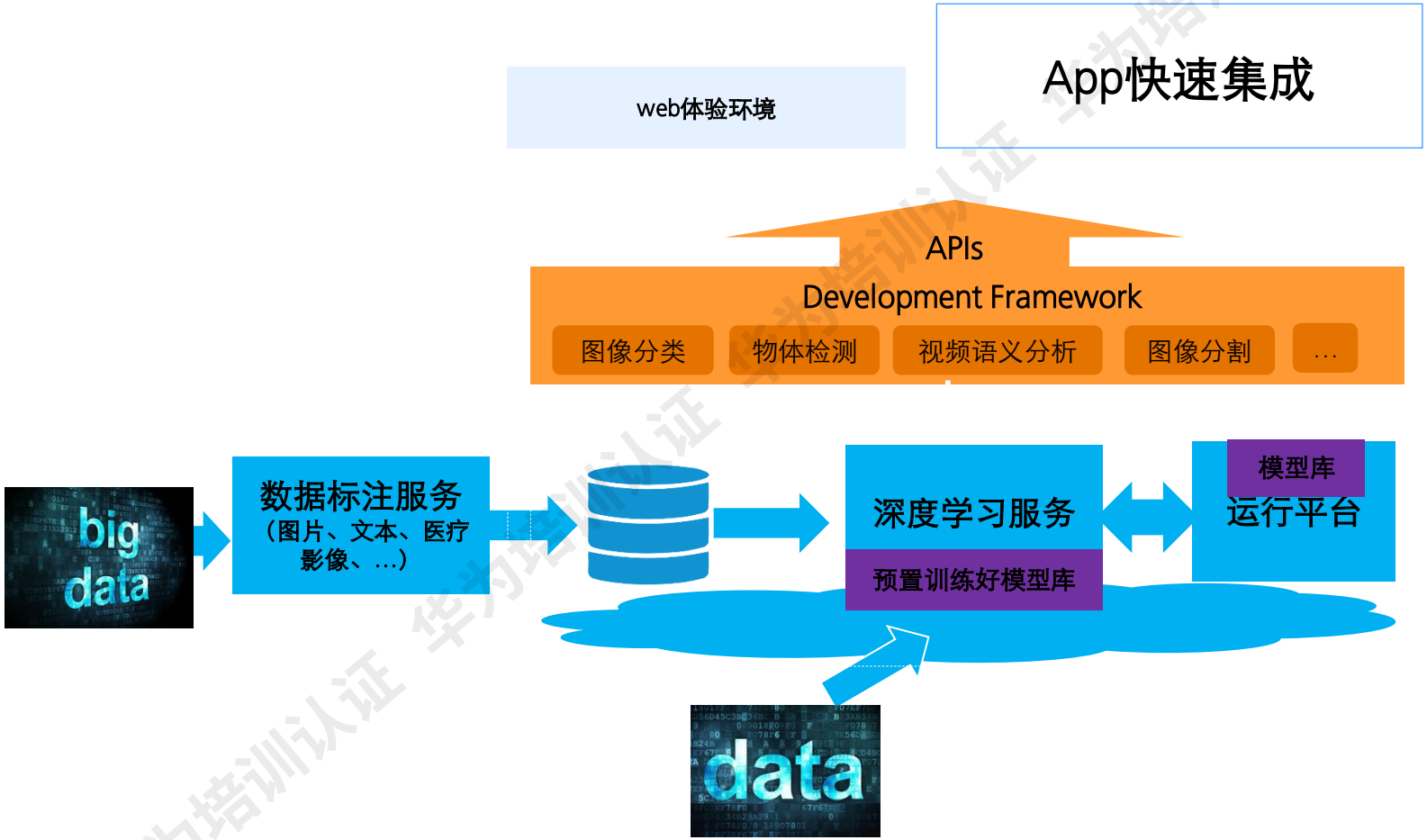
助力应用推荐

支持应用市场、音乐、新闻、游戏等服务的推荐，CTR提高30%

车联网分析

构建预测维护模型，减少故障发生几率，降低维护成本

DLS：预置模型开发框架，方便快速运用深度学习开发业务



丰富的模型，极致的性能和一键式开发环境，大大减低深度学习模型开发门槛，支撑快速业务上线。

DLS成功实践 (1)

- 哑设备识别与验收图片审核。

质量图像识别对象模型

EHS 15种智能对象模型（逐步扩展 共约50种）

人	安全帽	警示牌
手套	警示牌	急救箱
安全帽	滑轮	. . .

TE 25种智能对象模型（逐步扩展）

机柜	电池	DCDU
BBU	RRU	天线
GPS	防水	. . .

OSP、CW、能源（待扩展）

人工图片标注

EHS 15种识别对象，当前共标注6000+图片（首批与北京德科合作）

场景规则

EHS智能审核规则

编号	需要标注的物体	标准定义	业务审核规则	精度
1	人	person	有一个人	>80%
2	安全帽	helmet	有一只安全帽	>80%
3	人头+安全帽	helmet_on_head	每人一只安全帽	>80%
4	荧光服	high_visible_vest	每人一件安全服	>80%
5	手套	glove	有两只手套	>70%
6	塔工	person	有两只钩	>80%
		hook		>80%
7	急救箱	first_aid_box	有一个急救箱	>80%
8	灭火器	extinguisher	有一只灭火器	>80%
9	警示牌	warning_board	一个警示牌	>80%
10	警示标示	warning_sign	至少一个警示标示	>80%
11	警戒线	warning_line	至少一段警戒线	>80%
12	工具包	toolkit	一个工具包	>80%
13	安全鞋	safety_shoe	每人两只安全鞋	>70%
14	滑轮	pulley	有一个滑轮	>80%
15	绝缘手套	insulated_glove	有两只绝缘手套	>70%

TE、OSP、CW、能源规则（待扩展）

模型训练

识别出：
1.人
2.荧光马甲
3.安全帽

识别出：
1.警示牌
2.12个警示标识

序号	当前识别条目	精度-precision	召回率-recall
1	Person	100%	98.60%
2	Warning board	100%	98.36%
3	Extinguish	95.24%	97.56%
4	High visible vest	100%	97%
5	Helmet on head	100%	93.30%
6	Warning line	100%	92.85%
7	First aid box	100%	90%
8	Warning Sign	99.75%	88.89%
9	Toolkit	92.83%	76.47%
10	Helmet	97.70%	69.40%
11	Hook	85.18%	46.94%
12	Glove	96%	41.94%
	All	98.71%	84.91%

DLS成功实践 (2)

- 结合深度学习和统计学习识别勘测中的室内设备。

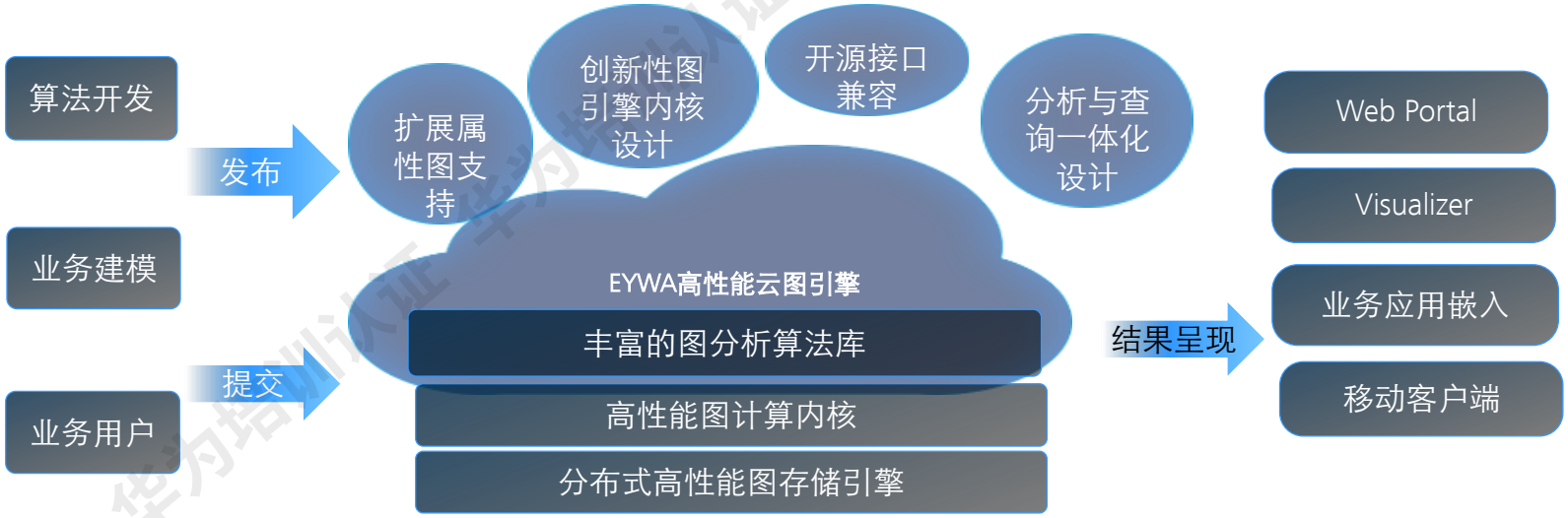
目前成果： 识别室内设备和哑设备，单站勘测时间从2个小时→10分钟，勘测准确率从67%→90%，减少重复上站，加快工期（减低人为的犯错，提升效率）

GES：超大规模一体化图分析与查询

GES提供一体化的关联数据查询、关系分析等平台能力，以及相关业务应用：**大规模、高并发、低延时**

10亿节点**1000亿+**边规模属性图，实现**~秒**级查询响应

支持**标准属性图**及其扩展，具备强大模型描述能力；兼容主流**图计算框架**与**图查询接口**



GES成功实践

路由规划

几十万级路由规划：
6.8小时->分钟级

车险防骗保

亿级节点十亿边：
为企业节省XXX万

目录

1. AI的认知及EI的由来
2. **华为云EI企业智能详细介绍**
   - 基础平台类服务
   - 通用领域类服务
   - 行业领域类服务

EI视觉认知

- **人脸识别服务**（Face Recognition Service）：能够在图像中快速检测人脸、分析人脸关键点信息、获取人脸属性、实现人脸的精确比对和检索。该服务可应用于身份验证、电子考勤、轨迹追踪、客流分析等场景。
- **图像识别服务**（Image Recognition Service）：基于领先的深度学习技术，可准确识别图像中的视觉内容，提供多达23000种物体、场景和概念标签，具备目标检测和属性识别等能力，帮助客户准确识别和理解图像内容。
- **文字识别服务**（Optical Character Recognition Service）：就是将图片或扫描件中的文字识别成可编辑的文本。可代替人工录入，提升业务效率。支持身份证、驾驶证、行驶证、发票、中英文海关单据、通用表格、通用文字等场景文字识别。
- **内容检测服务**（Content Moderation Service ）：基于领先的图像、文本、视频检测技术，可自动检测色情、广告、暴恐、涉政等内容，帮助客户降低业务违规风险。
- **图像处理服务**（Deblur Service ）：基于信号处理和领先的深度学习技术，提供图像低光照增强、去雾、超分辨率重建等能力。对光线不足或逆光、雾霾天气、分辨率不足等因素导致的模糊图像，重建出高清图像。

EI语音语义

- **机器翻译服务（Machine Translation Service）**：致力于为企业和个人提供不同语种间快速翻译能力，通过API调用即可实现源语言文本到目标语言文本的自动翻译。
- **语音合成服务（Text To Speech Service）**：是一种将文本转换成逼真语音的服务。通过音色选择、自定义音量、语速，为企业和个人提供个性化的发音服务(本服务核心能力由科大讯飞提供)。
- **语音识别服务（Automatic Speech Recognition Service）**：将口述音频转换为文本，通过API调用识别不同音频源发来的实时音频，或识别音频文件。识别的语言为中文普通话(本服务核心能力由科大讯飞提供)。
- **自然语言处理服务（Natural Language Processing Service）**：为用户提供包括分词、命名实体识别、关键词提取、短文本相似度等自然语言相关的API，可用于智能问答、对话机器人、舆情分析、内容推荐、电商评价分析等场景中。
- **对话机器人服务 (Conversational Bot Service)**：帮助企业快速构建，发布和管理智能问答系统。支持智能解析企业文档形成问答知识库，对接业务系统实现自助服务机器人，例如：电商智能客服、呼叫中心自助应答、智能语音助手等。

OCR：提供高精度光学文字自动识别能力

原始图片


中华人民共和国海关出口货物报关单

通用单据

身份证、名片

自然场景、通用文字

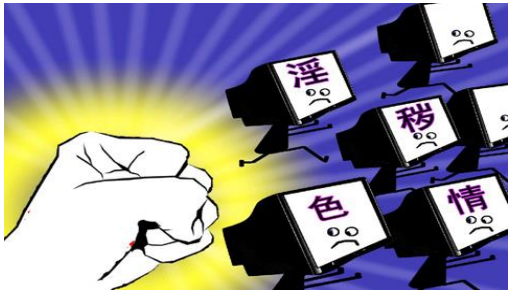
高精度OCR识别引擎

结构化文本

- Excel表格
- JSON数据
- 网页界面
- 移动客户端
- 存入数据库

HUAWEI

Moderation：提供自动内容检测



文本、图像鉴黄

准确识别文本、图片、视频中的涉黄内容，对文本、图片、视频进行筛选，解放审核人力。

涉政识别

识别包含敏感政治事件，反动反政府言论的文本文件。

暴恐识别

精准识别图片中是否包含杀人流血场景、 暴恐袭击场景、恐怖组织的旗帜等内容。

Deblur：图片低光照增强，去雾处理

低光照片

多维直方图均衡 vs 华为

低光照增

亮度失序误差率<3%

雾天照片

ECCV 2016方法 vs 华为

图像去雾

整体亮度真实，**不变暗**

HUAWEI

目录

1. AI的认知及EI的由来
2. **华为云EI企业智能详细介绍**
   - 基础平台类服务
   - 通用领域类服务
   - 行业领域类服务

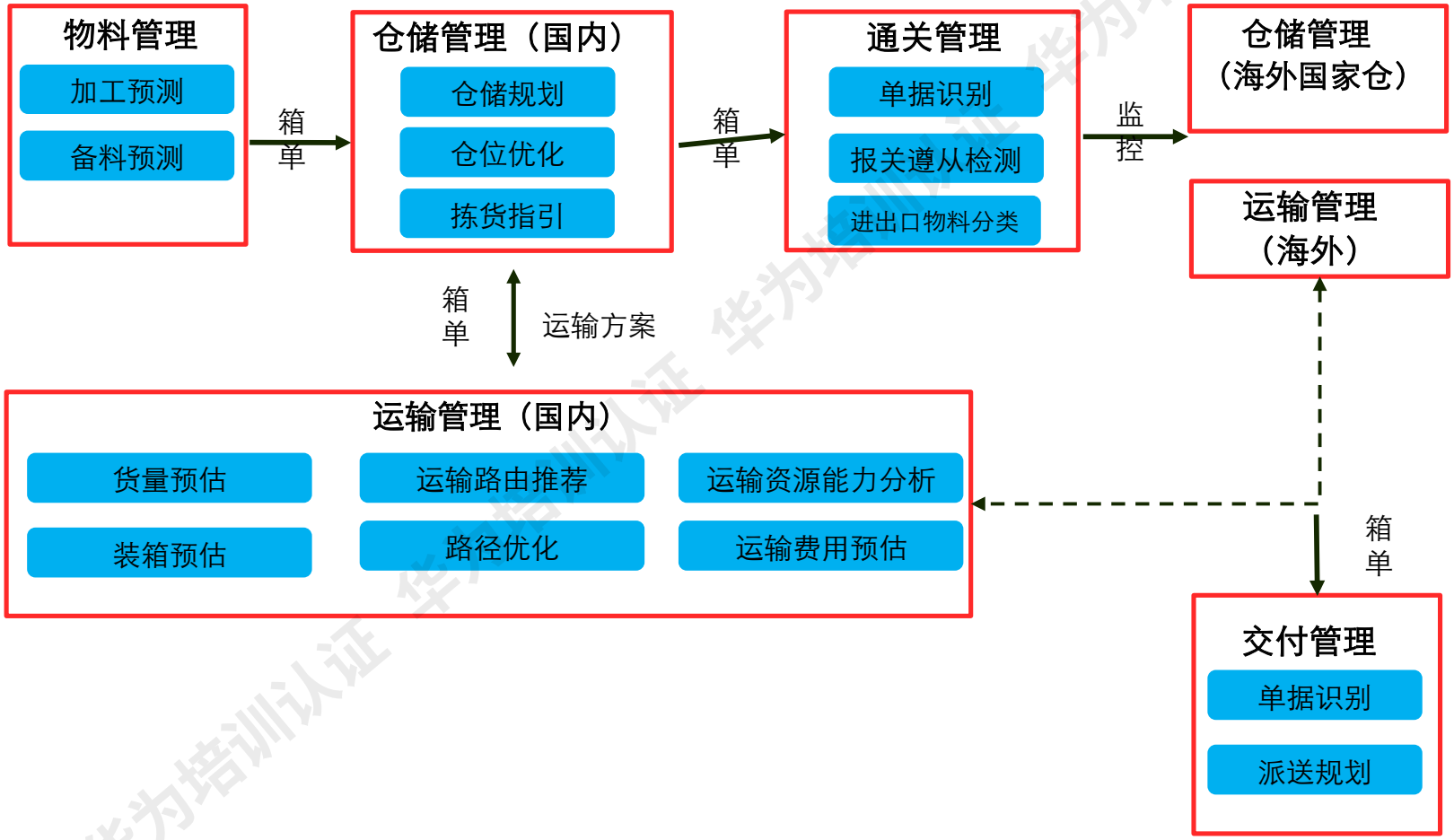
EI行业领域类服务 (1)

- **智能物流服务**（Intelligent Logistics Service）：利用人工智能及优化技术，实现企业物流数字化、智能化转型，降低物流运营成本、提升物流运营效率。智能物流涵盖物流链条中仓储、运输环节中的出库拣货路径规划、运输路径规划、智能装车等场景。
- **智能制造服务**（Intelligent manufacturing Service）：依托大数据&人工智能，提供设计、生产、物流、销售、服务全链式智能服务，挖掘海量数据价值，助力企业借助新技术，构筑领先优势。
- **智能金融服务**（Intelligent Finance Service)：基于大数据和人工智能技术，帮助金融客户快速构建安全合规的智能金融解决方案，助力金融企业向数字化、智能化转型。
- **智能零售服务**（Intelligent Retail Service）：基于人工智能技术帮忙新零售企业快速搭建智能化平台，提供应对以消费者体验为中心的、数据驱动的泛零售智能端到端解决方案。

EI行业领域类服务 (2)

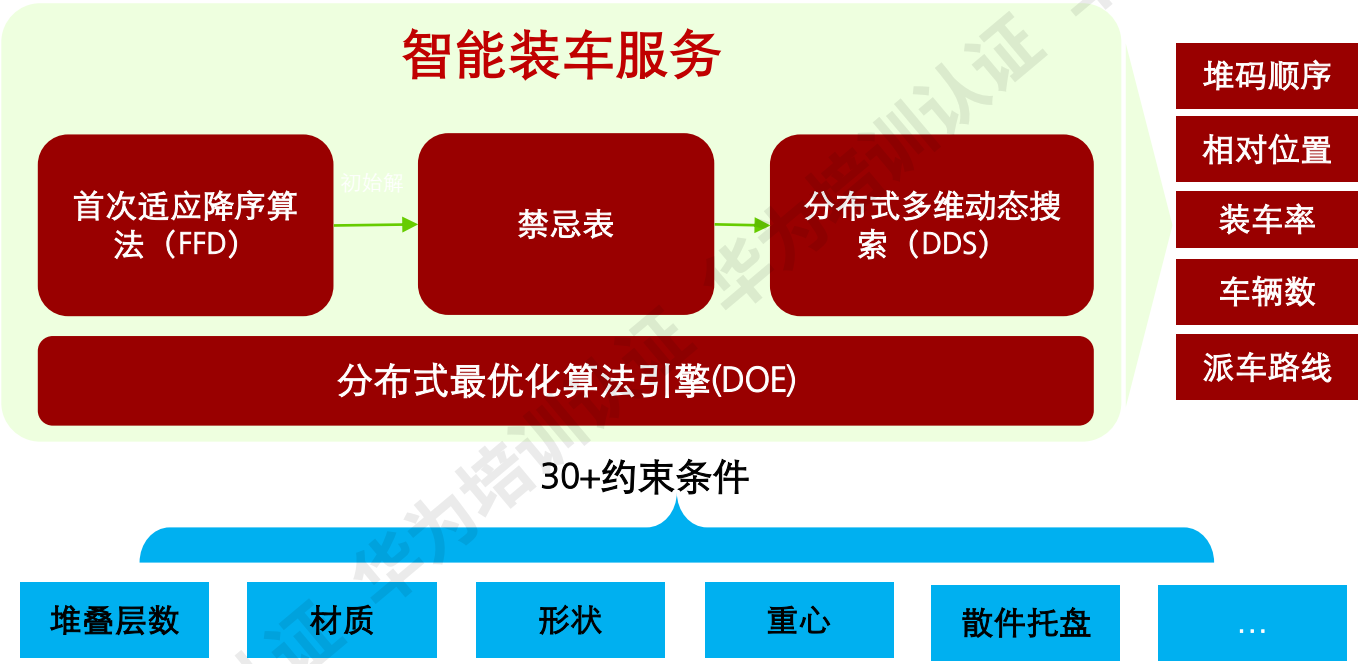
- **智能电力服务**（Intelligent Power Service）：结合大数据和人工智能，帮助电力企业快速搭建平台、应对发电、售电、充电、能效等场景的智能解决方案，涵盖发电站选址、设备故障预测、发电用电预测、充电桩管理等场景。
- **智能交通服务**（Intelligent Transport Service）：基于华为大数据和人工智能技术，提供违法识别、智慧信号灯、时空检索等功能，应用于违法智能识别、信号灯智能控制、套牌车智能发现等场景，降低交通管理成本，提升城市通行效率。
- **智能水务服务**（Intelligent Water Service）：利用华为物联网平台，依托大数据和人工智能技术，帮助水务企业实现智能化转型、降低能耗、提升运营效率。

智能物流解决方案全景图



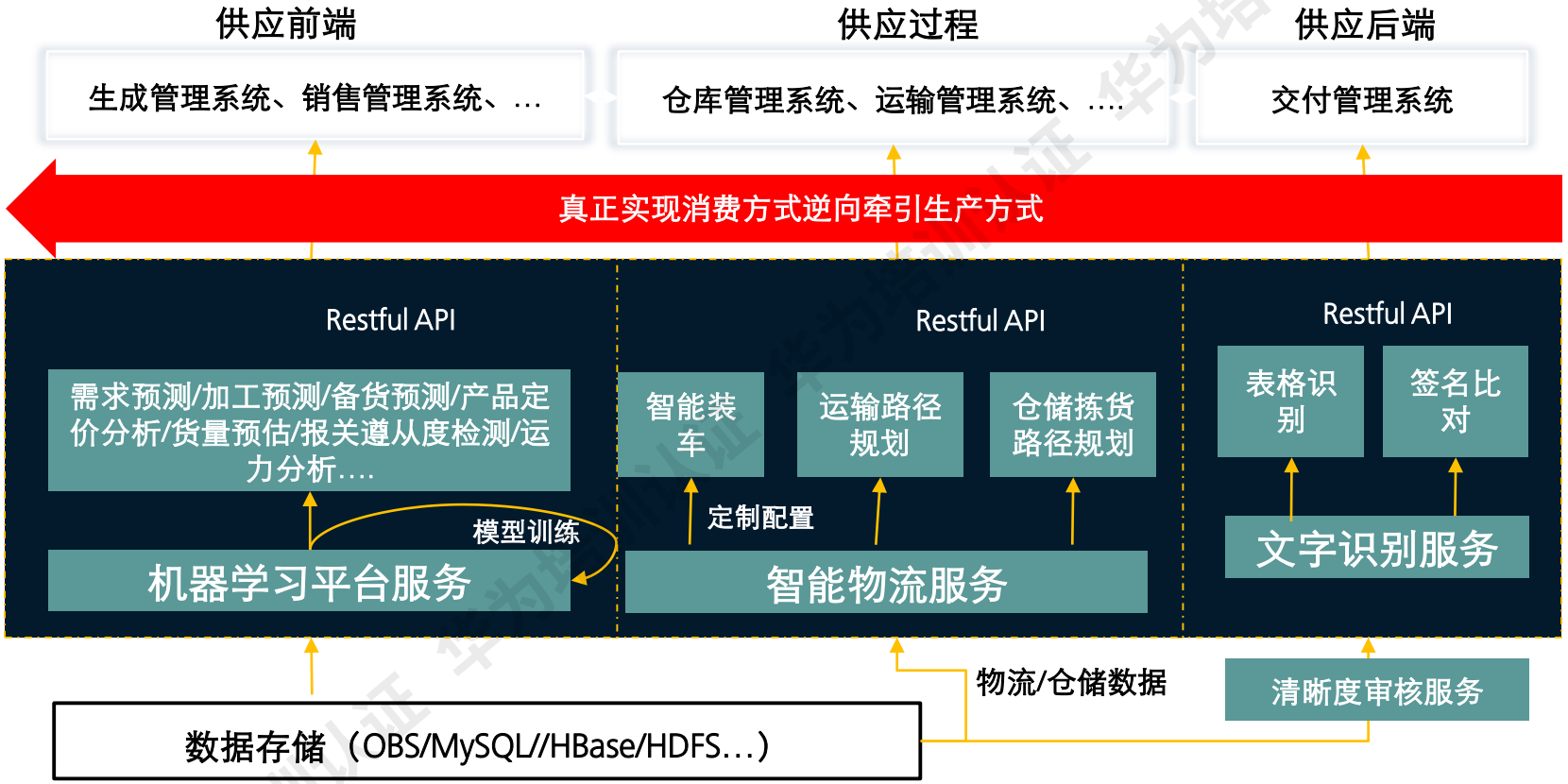
HUAWEI

智能装车服务



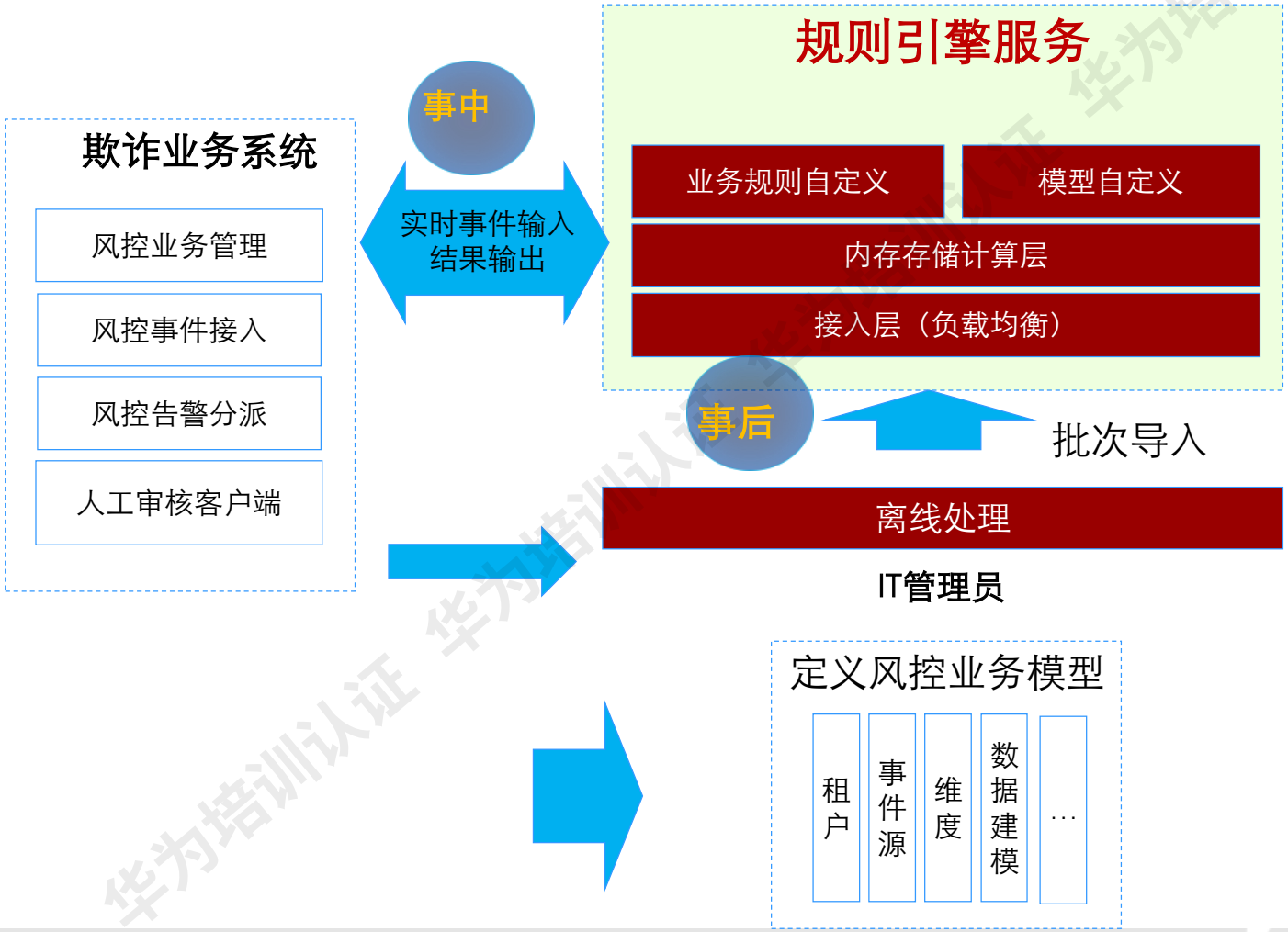
HUAWEI

云上制造行业智能解决方案



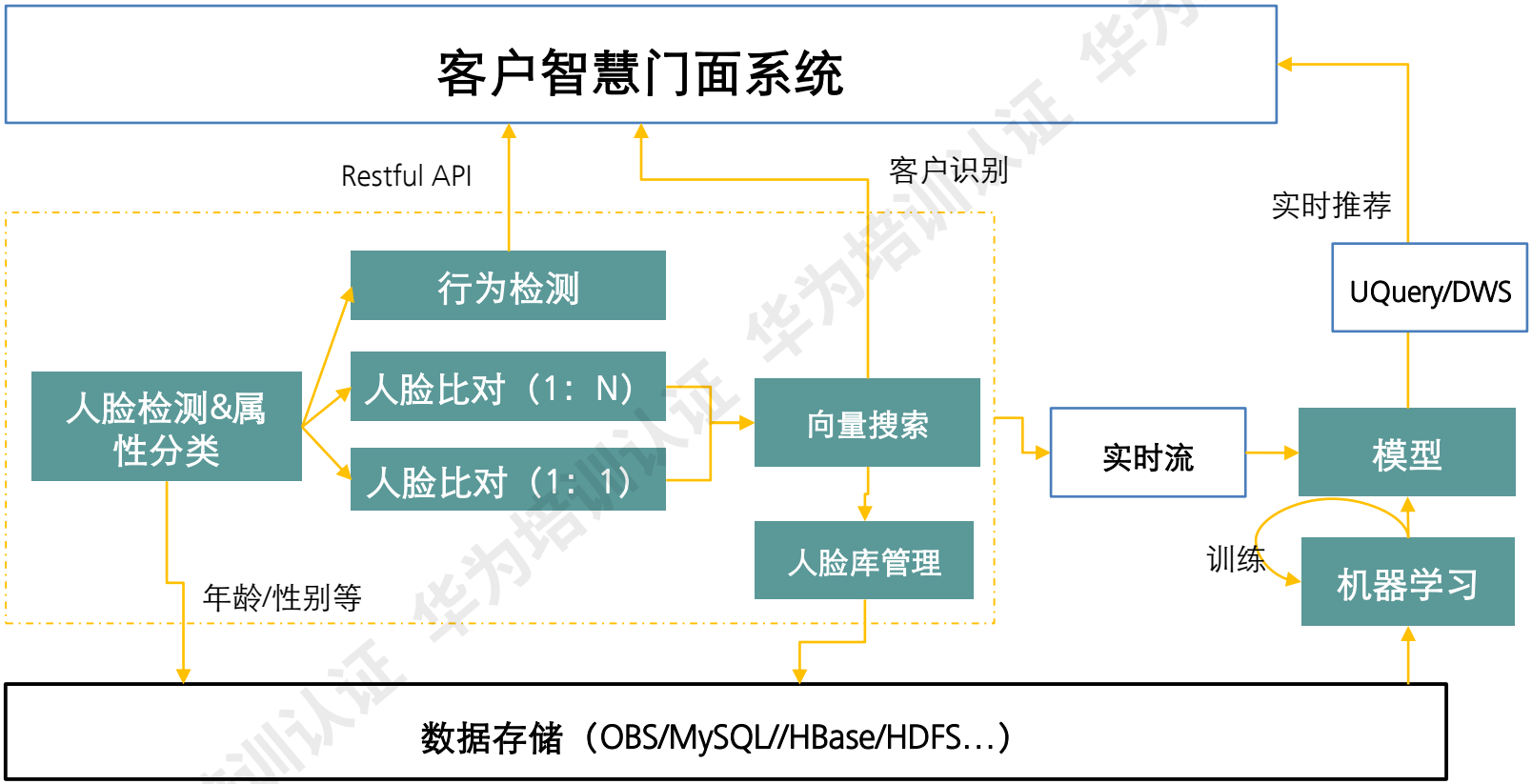
- **更精准的物流算法库** | 提供多种约束条件的最优化算法包，效率提升3X
- **表格文字识别** | 提供灵活通用表格识别，识别率>95%

智能风控



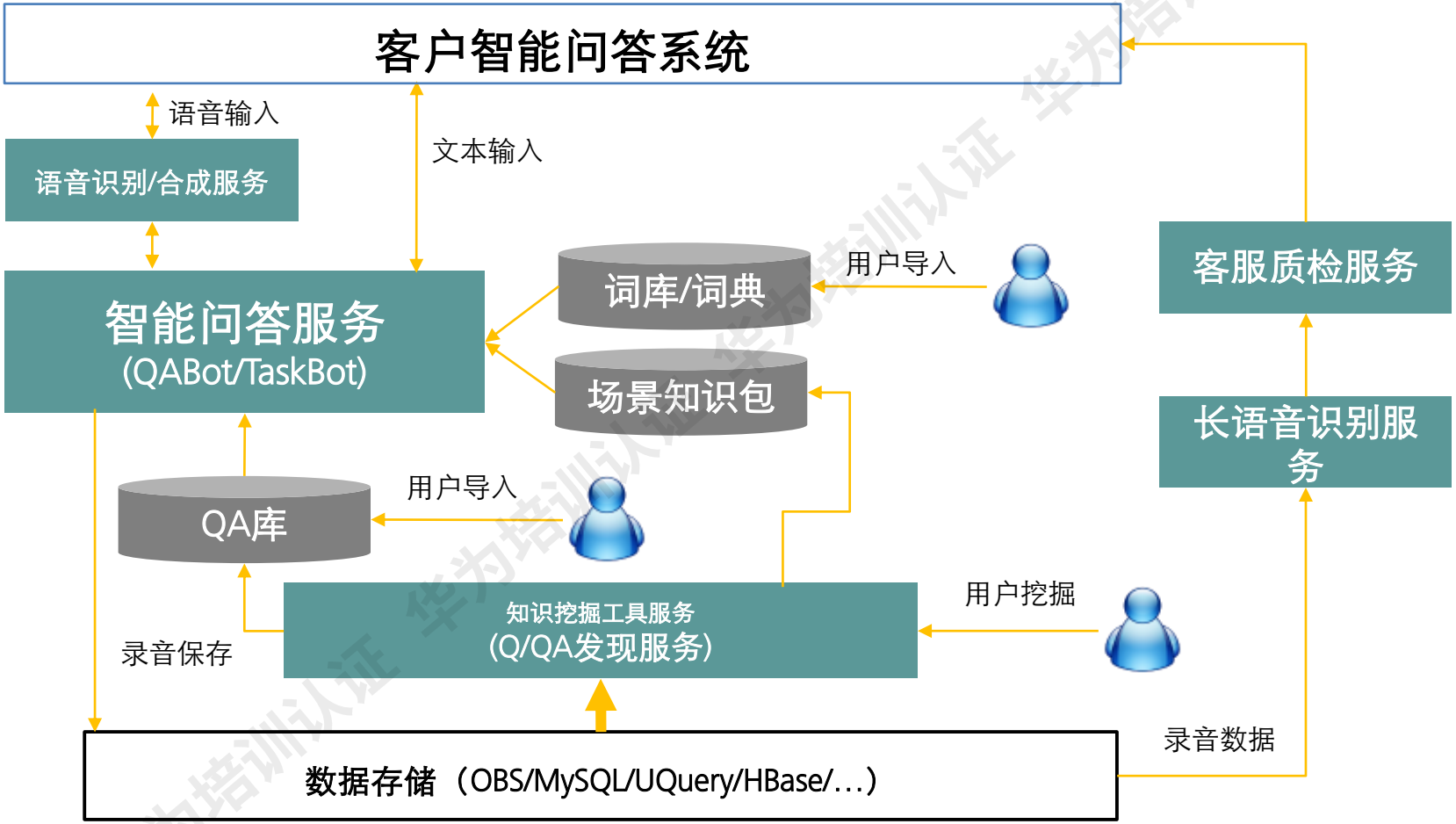
HUAWEI

云上新零售智能门面解决方案



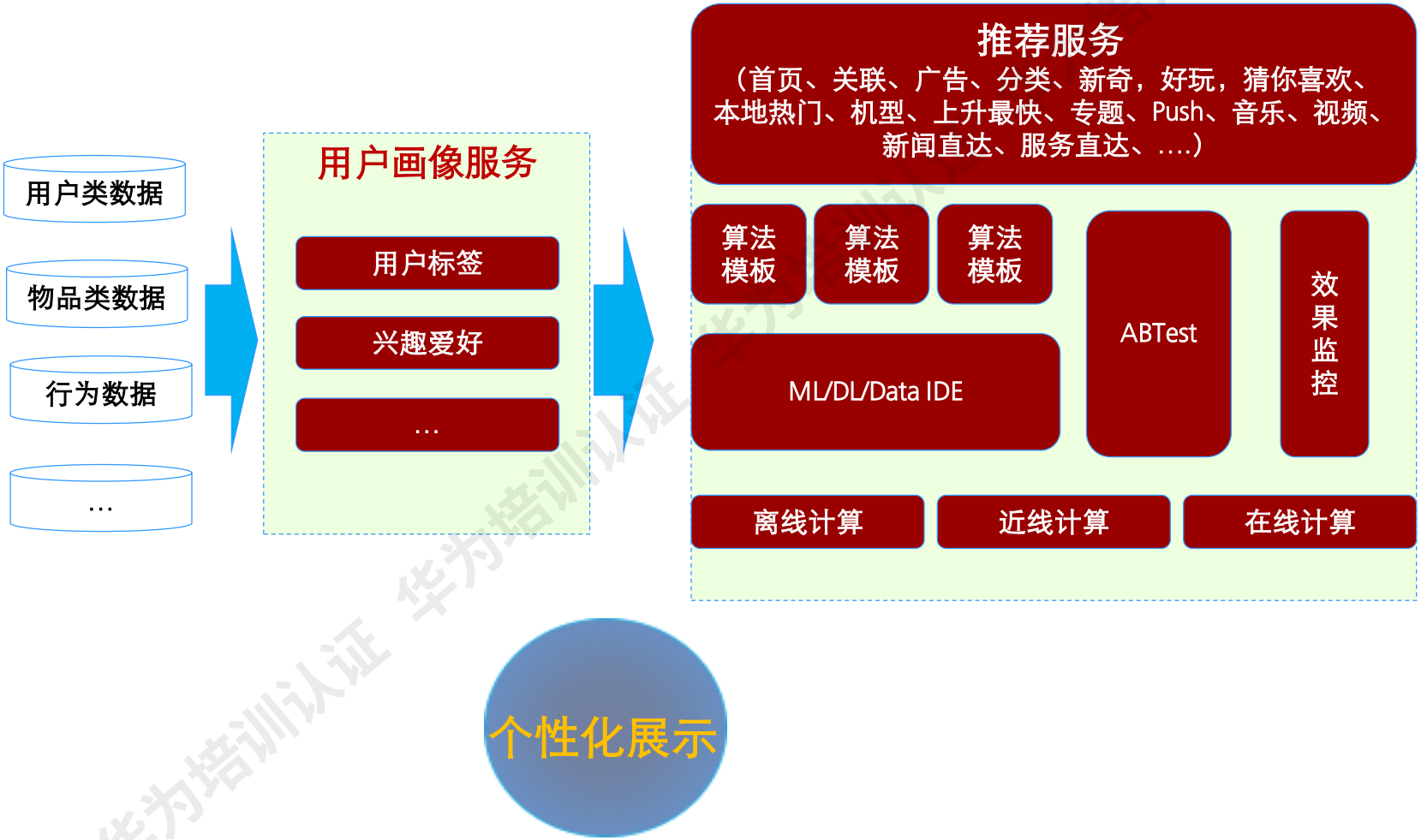
- 人脸识别服务 | 基于非约束条件下人像识别AUC > 99%，搜索命中率>85%

云上智能客服解决方案



HUAWEI

智能推荐：个性化推荐



智能小区规划参数调整

2D地图
3D建筑物
MR+OTT
网络信息

①虚拟路测

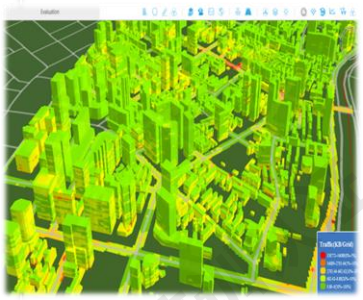
6200万公里/年 路测里程 ⇨ 20% 减少

②自动天线调优建议

2 Cluster /人月 ⇨ 6 Cluster/ 人月

电子地图
工程参数
MR+OTT
天线库

3D覆盖地图



路测报告自动化

- 方位角
- 天线高度
- 机械倾角
- 电调倾角

天线调整建议

HUAWEI

EI大数据服务 (1)

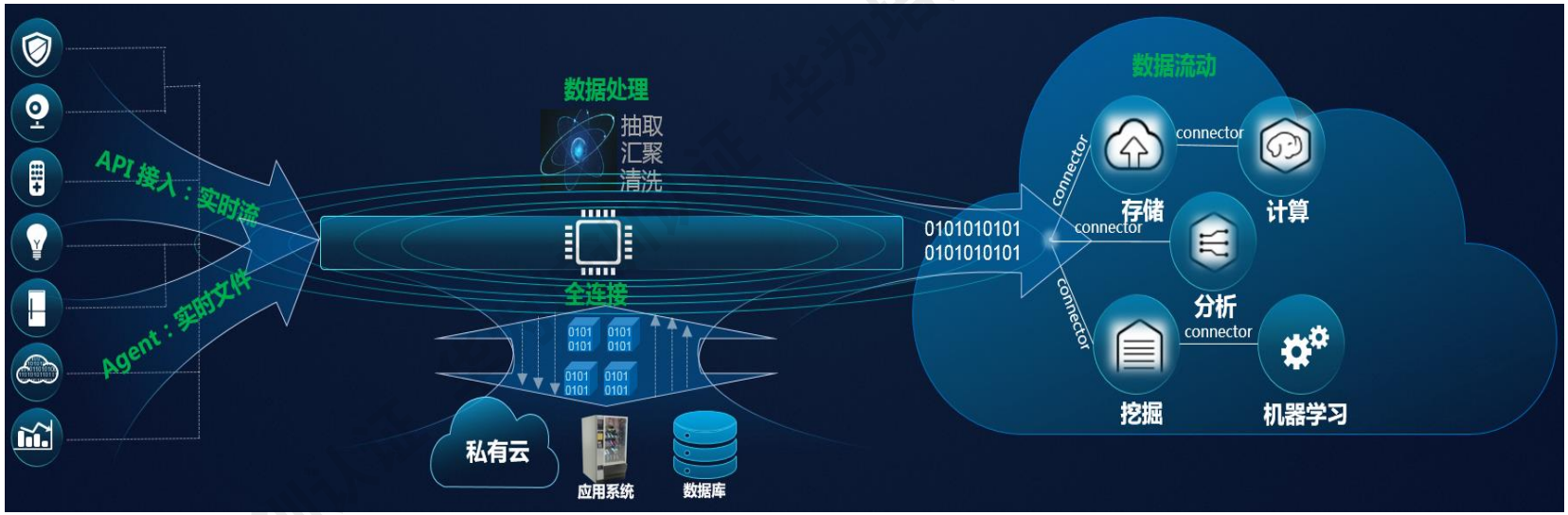
- **数据接入服务**（Data Ingestion Service）：是华为云提供的实时数据接入服务。它提供了灵活数据采集、高效数据传输、实时数据分发能力，让您可以轻松构建基于实时数据的分析和应用。
- **云数据迁移服务**（Cloud Data Migration Service）：提供同构/异构数据源之间批量数据迁移服务，帮助客户实现数据自由流动。支持客户自建和公有云上的文件系统，关系数据库，数据仓库，NoSQL，大数据云服务，对象存储等数据源。
- **实时流计算服务**（Cloud Stream Service）：提供实时处理流式大数据的全栈能力，简单易用，即时执行Stream SQL或自定义作业。无需关心计算集群，无需学习编程技能。完全兼容Apache Flink和Spark API。
- MapReduce服务（MapReduce Service）：提供租户完全可控的企业级大数据集群云服务，轻松运行Hadoop、Spark、HBase、Kafka、Storm等大数据组件。
- **推荐引擎服务**（Recommendation Engine Service）：提供低门槛、标准化的个性化推荐业务搭建，内置电商、新闻等行业模板，轻松实现高质量的推荐服务。

EI大数据服务 (2)

- **数据湖探索服务（Data Lake Insight Service）**：是完全托管的大数据处理分析服务，无需ETL,使用SQL和Spark程序就可以对华为云上多源异构数据进行探索。
- **表格存储服务(CloudTable Service)**：是基于Apache HBase提供的全托管NoSQL服务，集成OpenTSDB和GeoMesa，提供毫秒级随机读写能力，适用于海量(半)结构化数据存储。可被广泛应用于物联网、车联网、金融、智慧城市等行业。
- **数据仓库服务（Data Warehouse Service）**：基于华为FusionInsight LibrA企业级数据仓库内核，提供即开即用、可扩展且完全托管的分析型数据库服务。兼容PostgreSQL生态，您可基于标准SQL，结合商业智能(BI)工具，经济高效地对海量数据进行挖掘和分析，使用成本大大低于传统数仓。
- **云搜索服务**：是一个基于Elasticsearch且完全托管的在线分布式搜索服务，为用户提供结构化、非结构化文本的多条件检索、统计、报表。完全兼容开源Elasticsearch软件原生接口。
- **数据湖工厂服务（Data Lake Factory Service）**：提供一站式的大数据协同开发平台，帮忙用户轻松完成数据建模，数据集成，脚本开发，作业调度，运维监控等多项任务，可以极大降低用户使用大数据的门槛，帮助用户快速构建大数据处理中心。

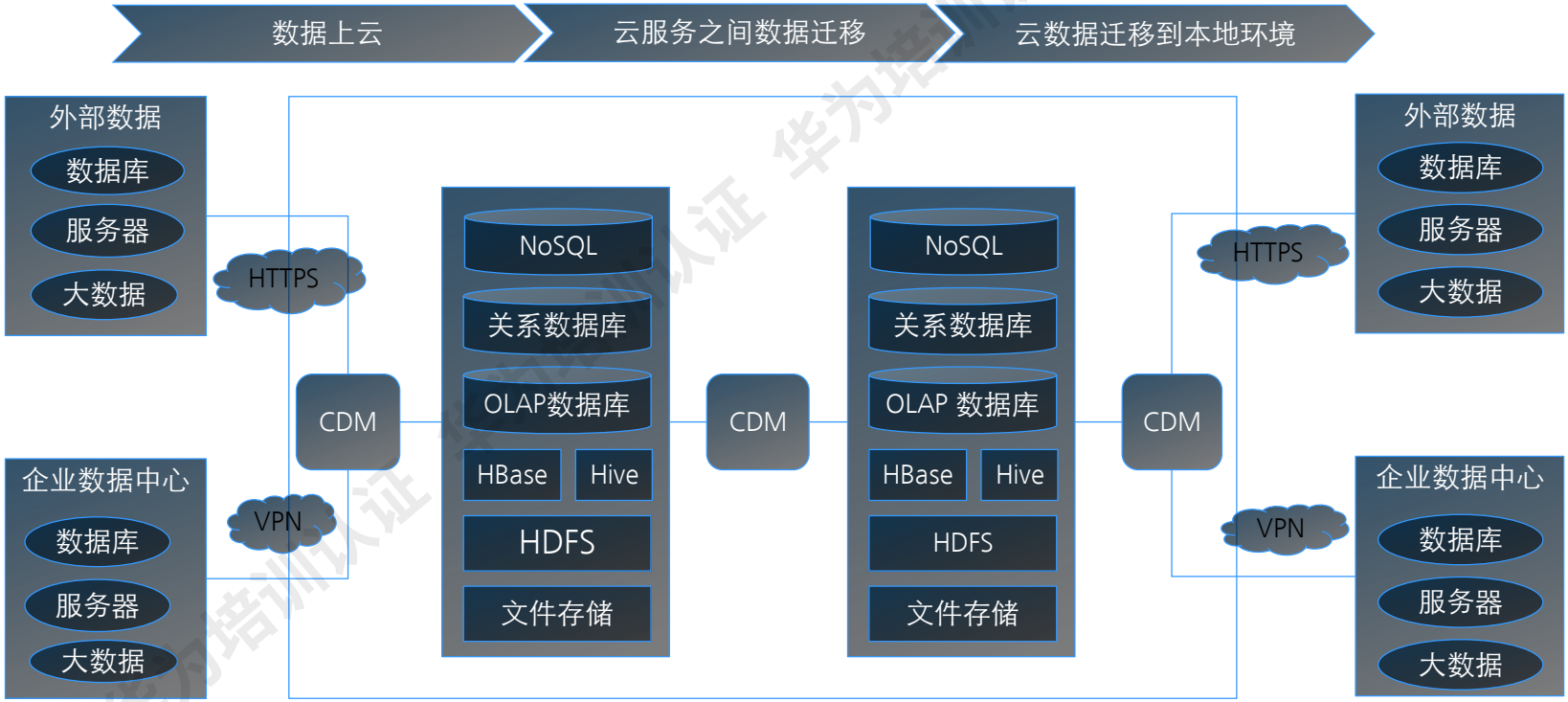
DIS：IoT实时数据全连接管道

- **数据采集**：RestAPI、Agent**对接**多种数据源，**兼容**开源Kafka API，线下数据**实时采集**。
- **数据传输**：实时数据管道，**高并发、低延时、高可靠**数据传输。
- **数据流动**：对接云上数据**存储、计算、分析**等多种服务，实时数据自由流动。
- **数据智能处理**：预置数据**抽取、格式转换、压缩**等处理算子，并支持用户**自定义**算子。



CDM：数据高效批量迁移

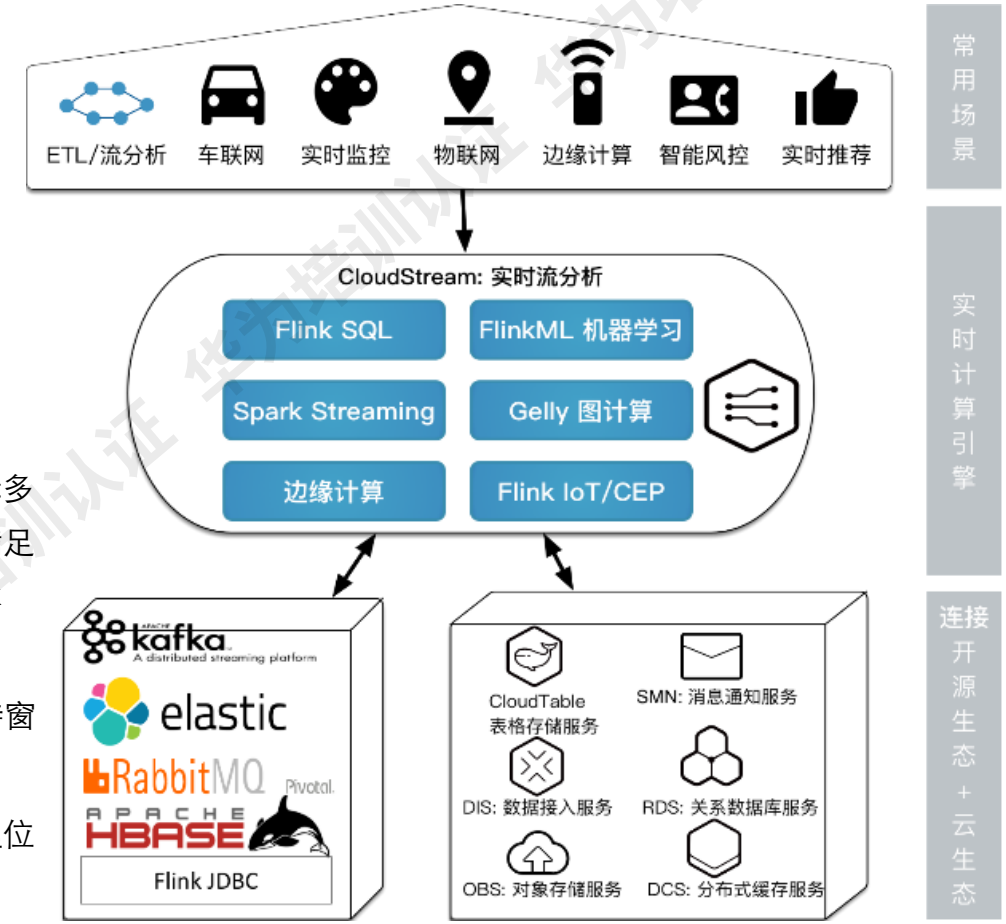
CDM基于分布式计算框架，实现数据高效批量迁移，支持包括关系数据库，OLAP数据库，NoSQL数据库，大数据存储，文件存储等常用数据源。提供云上和云下端到端数据迁移解决方案，满足大数据场景下的各种数据迁移诉求。



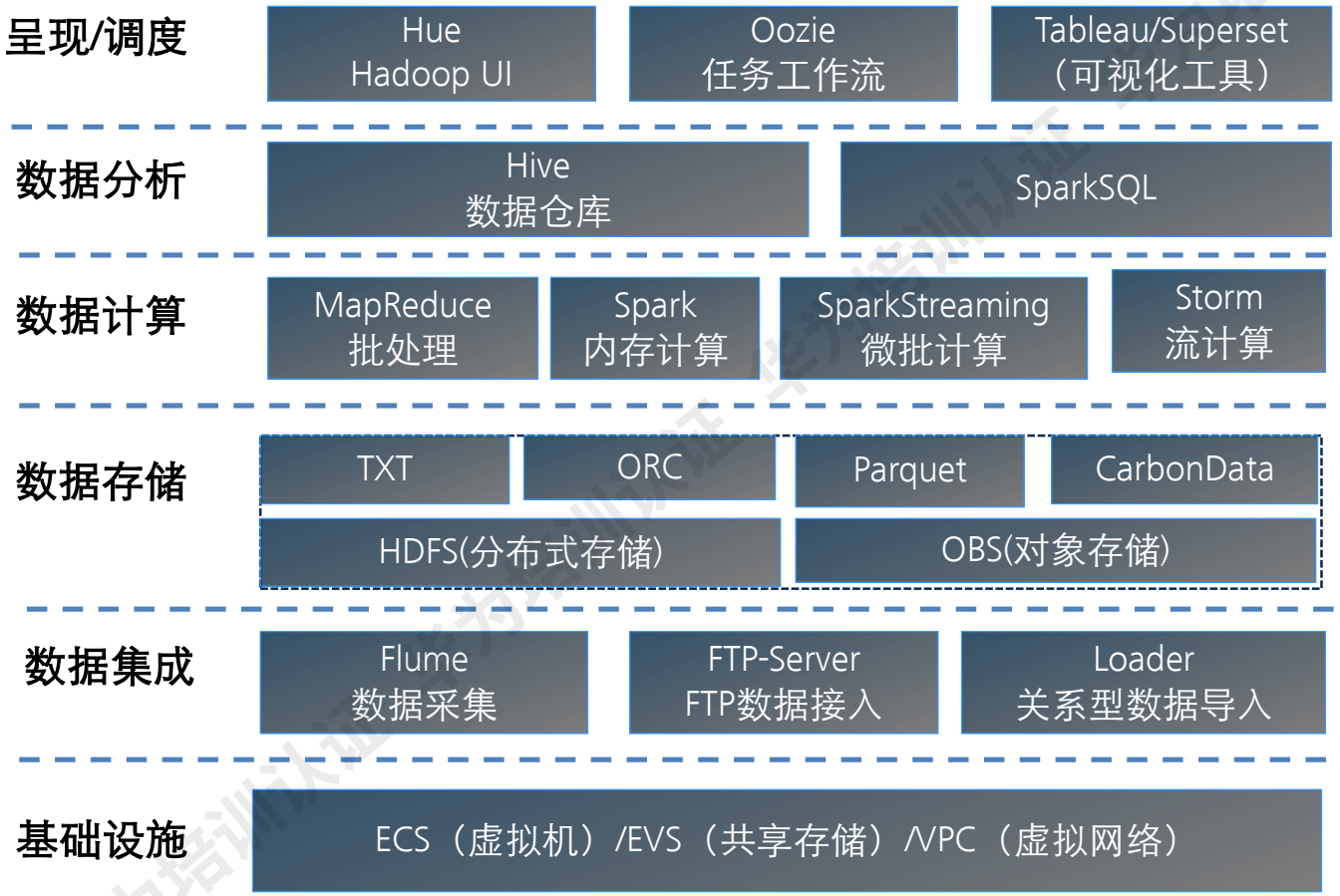
CloudStream：IoT领域核心服务

- 应用场景：
- IoT、车联网：偏航、电子围栏、异常检测实时告警、CEP。
- 金融/证券/电商/游戏/广告：智能风控，实时规则引擎、图计算、反欺诈、实时推荐、实时监控。

- 关键竞争力：
- 双引擎+双服务模式：同时支持逻辑多租+物理多租，降低使用门槛，并满足大客户安全性需求；支持Flink+spark streaming双引擎；业界领先。
- 简单易用：流式SQL能力丰富，支持窗口/CEP，场景模板化，使用零门槛
- 智能流：ML参数抽取函数、IoT地理位置函数、FlinkML、Gelly图计算、GraphX，连接AI智能服务（规划）

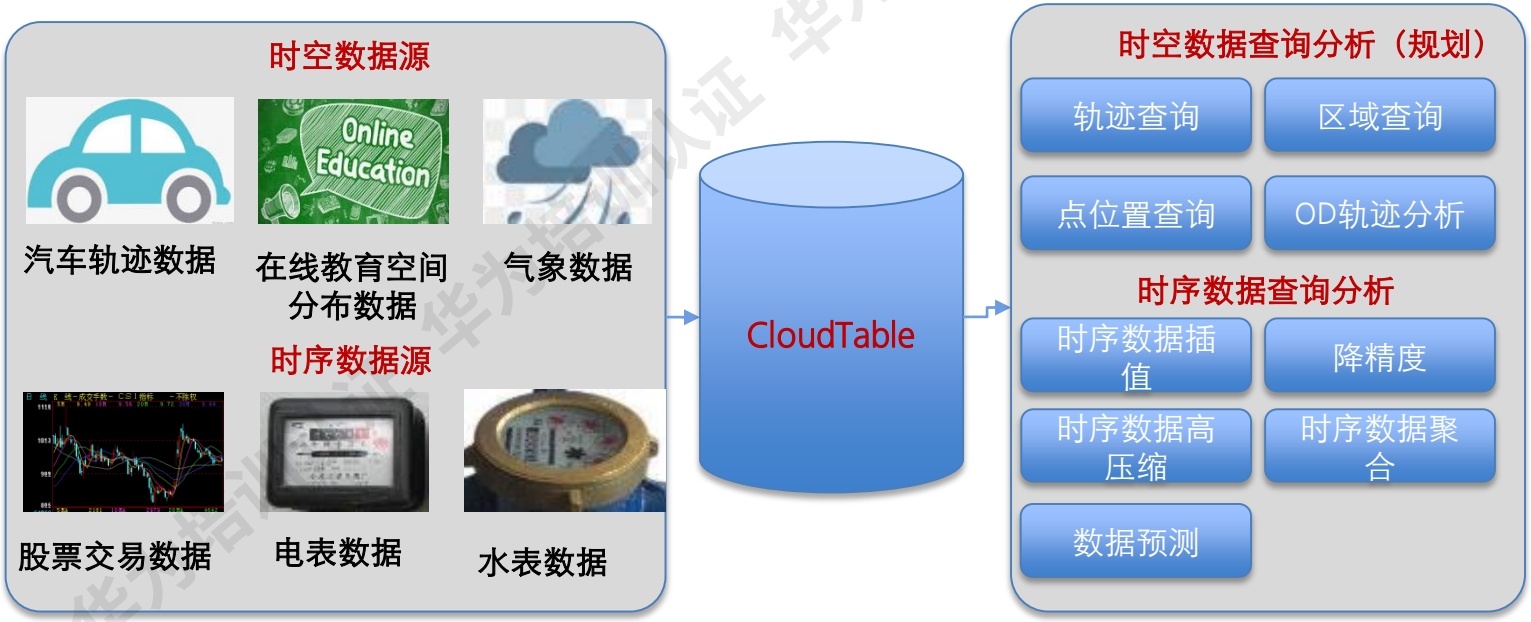


MRS：企业级大数据集群云服务

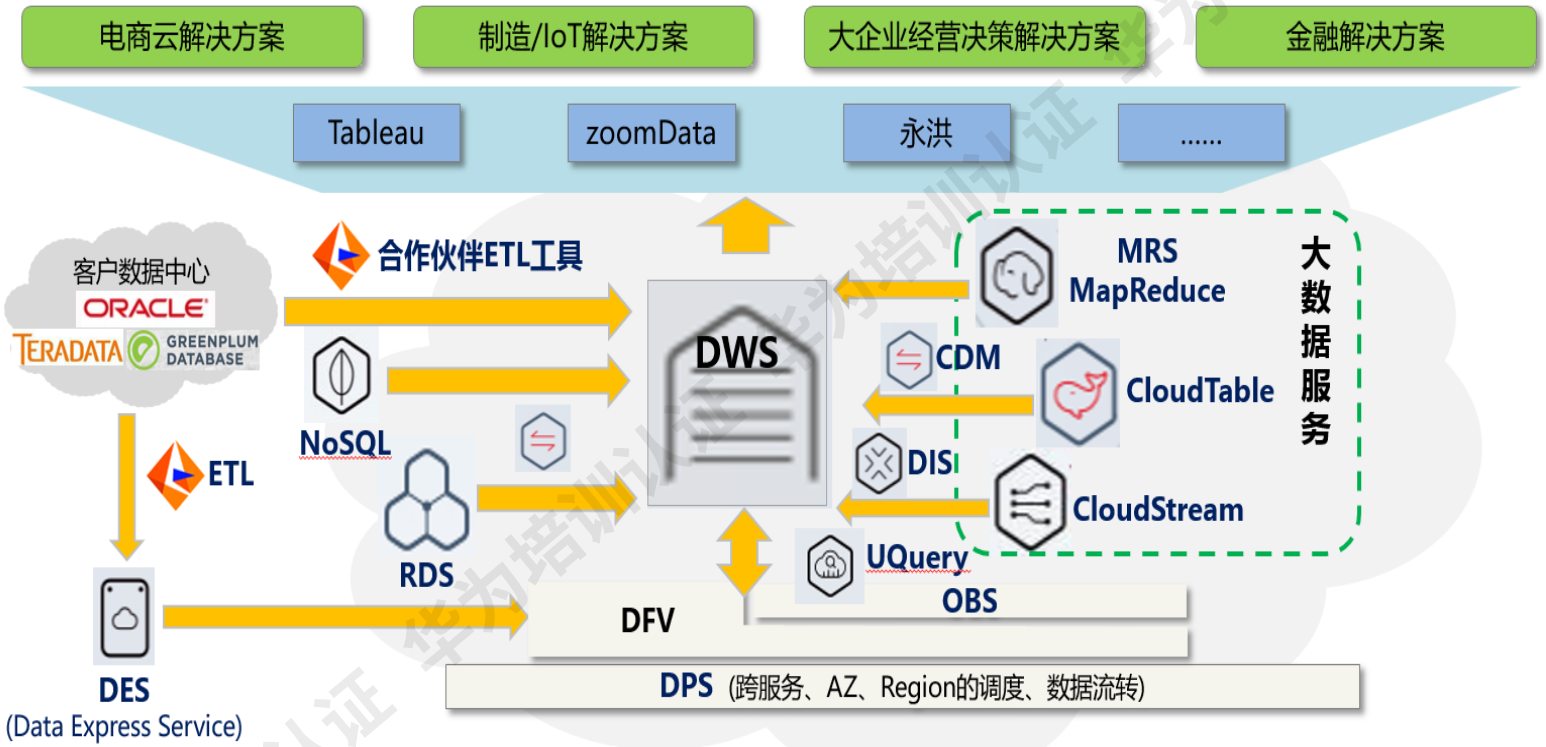


CloudTable：毫秒级NoSQL数据库

- **车联网**：车辆产生的属性数据和地理位置变化数据。
- IoT：燃气、水务、电力、化工设备、计算机设备、智能家居，等IoT设备监控分析。
- **在线教育/零售/旅游**：销售的区域分析，为优化区域销售提供决策支撑。
- **气象**：支持气象数据的三维空间+时间的数据存储、查询和分析。



DWS：企业级云上数据仓库



本章总结

本章首先介绍了华为云EI生态，使大家了解了当今大数据及人工智能技术的发展趋势，接着介绍了华为云EI服务产品能力，最后介绍了华为云EI业务场景与解决方案。

思考题

1. 什么是人工智能?
2. 华为云EI可以提供哪些功能服务?

思考题

1. (单选) 华为于哪一年开始正式以云服务方式提供服务，联合更多的伙伴一起对外提供更丰富的人工智能实践？（　）

   A. 2003

   B. 2012

   C. 2015

   D. 2017

2. (判断题) 机器学习服务是一项数据挖掘分析平台服务，帮助用户通过机器学习技术迅速发现数据规律和构建预测模型，并将其部署为预测分析解决方案。（　）

   A. True

   B. False

学习推荐

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh

谢谢

www.huawei.com

学习推荐

- 华为培训与认证官方网站
  - http://learning.huawei.com/cn/
- 华为在线学习
  - http://support.huawei.com/learning/elearning?lang=zh
- 华为职业认证
  - http://support.huawei.com/learning/NavigationAction!createNavi?navId=_31&lang=zh
- 查找培训入口
  - http://support.huawei.com/learning/NavigationAction!createNavi?navId=_trainingsearch&lang=zh

更多信息

- 华为培训APP