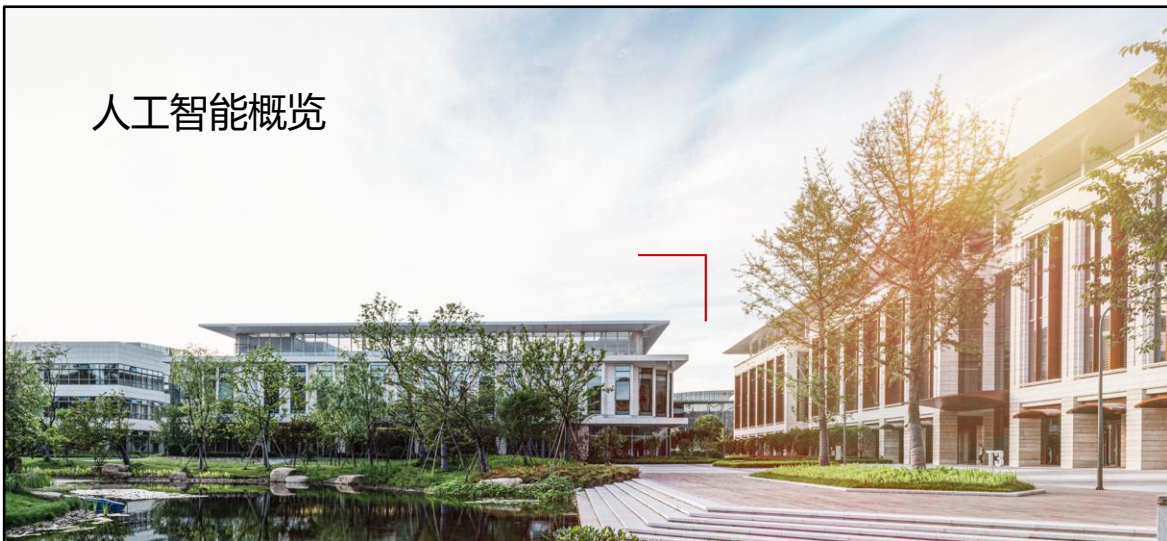


人工智能概览



前言

- 人工智能正以前所未有的速度推动着社会进步与产业升级。作为一门研究模拟人类智能的学科，人工智能不仅深刻改变了我们的生活方式，更在科学研究、工业生产、医疗健康、金融服务、教育娱乐等多个领域展现出其巨大的潜力和价值。
- 本章节将为您介绍人工智能的发展历程、关键技术及其在各领域的应用实践，将从人工智能的定义与范畴出发，逐步深入探讨人工智能技术应用领域与未来趋势。

目标

- 学完本课程后，您将能够：
 - 了解人工智能的基本概念。
 - 了解人工智能的相关技术及发展历史。
 - 了解人工智能的应用技术及应用领域。
 - 了解华为的人工智能解决方案。
 - 了解人工智能的发展趋势。

目录

1. **人工智能概述**
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

研究学者视角下的人工智能

“I propose to consider the question, ‘Can machines think?’”

——艾伦·图灵 1950

让机器达到同样的行为，即与人类做同样的行为。

——约翰·麦卡锡 1956

人工智能是一门科学，是使机器做那些人需要通过智能来做的事情。

——马文·闵斯基

- 人工智能的一个比较流行的定义，也是该领域较早的定义，是由约翰·麦卡锡（John McCarthy）在1956年的达特茅斯会议（Dartmouth Conference）上提出的：人工智能就是要让机器的行为看起来就象是人所表现出的智能行为一样。但是这个定义似乎忽略了强人工智能的可能性。另一个定义指人工智能是人造机器所表现出来的智能性（弱人工智能）。
- 艾伦·图灵在论文《计算机器与智能》中讨论过“机器能思考吗”问题。

何谓智能

- Howard Gardner教授提出多元智能理论，他指出了体现多元智能的八种能力：

- 语言 (Verbal/Linguistic)
- 逻辑 (Logical/Mathematical)
- 空间 (Visual/Spatial)
- 肢体动觉 (Bodily/Kinesthetic)
- 音乐 (Musical/Rhythmic)
- 人际 (Inter-personal/Social)
- 内省 (Intra-personal/Introspective)
- 自然 (Naturalist)

- 美国教育学家和心理学家加德纳

- 1、语言智能 (Linguistic intelligence)

是指有效的运用口头语言或及文字表达自己的思想并理解他人，灵活掌握语音、语义、语法，具备用言语思维、用言语表达和欣赏语言深层内涵的能力结合在一起并运用自如的能力。他们适合的职业是：政治活动家，主持人，律师，演说家，编辑，作家，记者，教师等。

- 2、数学逻辑智能 (Logical-Mathematical intelligence)

是指有效地计算、测量、推理、归纳、分类，并进行复杂数学运算的能力。这项智能包括对逻辑的方式和关系，陈述和主张，功能及其他相关的抽象概念的敏感性。他们适合的职业是：科学家、会计师、统计学家、工程师、电脑软体研发人员等。

- 3、空间智能 (Spatial intelligence)

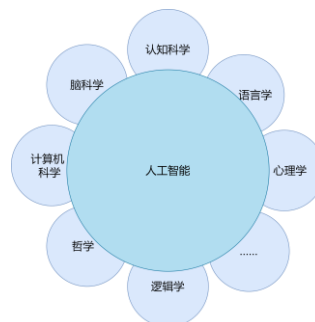
是指准确感知视觉空间及周围一切事物，并且能把所感觉到的形象以图画的形式表现出来的能力。这项智能包括对色彩、线条、形状、形式、空间关系很敏感。他们适合的职业是：室内设计师、建筑师、摄影师、画家、飞行员等。

- 4、身体运动智能 (Bodily-Kinesthetic intelligence)

是指善于运用整个身体来表达思想和情感、灵巧地运用双手制作或操作物体的能力。这项智能包括特殊的身体技巧，如平衡、协调、敏捷、力量、弹性和速度以及由触觉所引起的能力。他们适合的职业是：运动员、演员、舞蹈家、外科医生、宝石匠、机械师等。

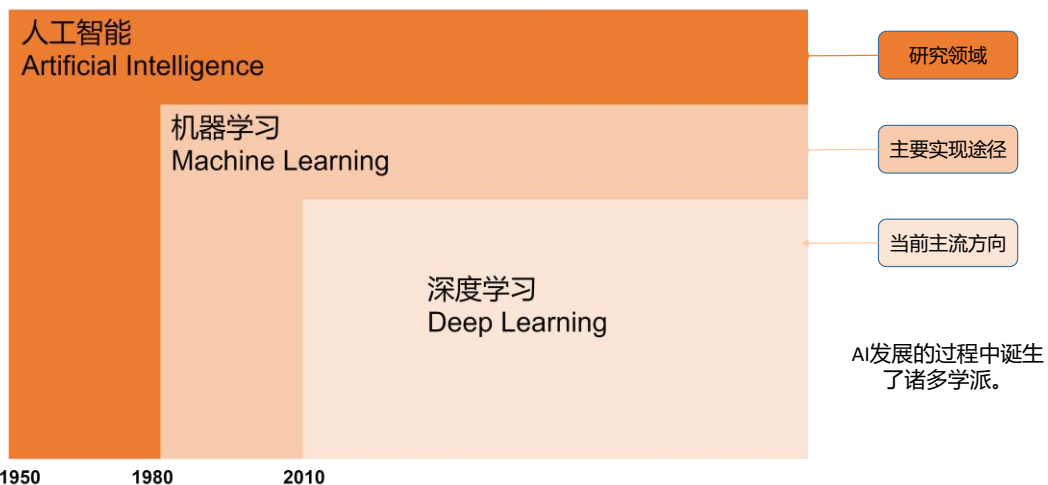
什么是人工智能

- 人工智能（Artificial Intelligence）的定义可以分为两部分，即“人工”和“智能”。“人工”即由人设计，为人创造、制造。“智能”即像人一样思考和行动。
- 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、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”。人工智能的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。



- 对机器学习的认识可以从多个方面进行，有着“全球机器学习教父”之称的 Tom Mitchell 则将机器学习定义为：对于某类任务 T 和性能度量 P ，如果计算，程序在 T 上以 P 衡量的性能随着经验 E 而自我完善，就称这个计算机程序从经验 E 学习。这些定义都比较简单抽象，但是随着对机器学习了解的深入，我们会发现随着时间的变迁，机器学习的内涵和外延在不断的变化。因为涉及到的领域和应用很广，发展和变化也相当迅速，简单明了地给出“机器学习”这一概念的定义并不是那么容易。

AI、机器学习、深度学习的关系 (1)



AI、机器学习、深度学习的关系 (2)

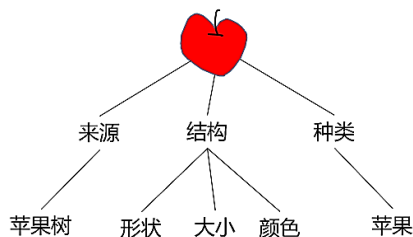
- 人工智能：是研究、开发用于模拟、延伸和扩展人的智能的理论、方法及应用系统的一门新的技术科学。
- 机器学习：研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。
- 深度学习：深度学习的概念源于人工神经网络的研究，是机器学习研究中的一个领域，它模仿人脑的机制来解释数据。

- “人工智能”一词以前曾被用来描述模仿和展示与人类思维相关的“人类”认知技能的机器。
- 机器学习算法主要是基于样本数据（称为训练数据）构建模型，以便能在没有明确编程的情况下做出预测或决策。
- 深度学习是机器学习的一种，而机器学习是实现人工智能的必经路径。深度学习的概念源于人工神经网络的研究，含多个隐藏层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，以发现数据的分布式特征表示。研究深度学习的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，声音和文本等。

人工智能的主要学派 - 符号主义

- 符号主义（Symbolicism），又称为逻辑主义（Logicism）、心理学派（Psychologism）或计算机学派（Computerism）。
 - 符号主义认为人工智能源于数理逻辑。他们认为人类的认知基元是符号，人类认知的过程是各种符号进行推理运算的过程，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，可以用计算机来模拟人的智能行为。

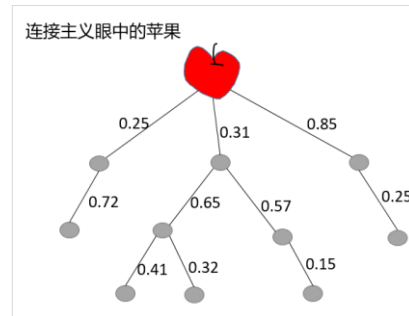
符号主义眼中的苹果



- 1956后，符号主义又发展了启发式算法>专家系统>知识工程理论与技术，并在1980年代取得重要发展。
- 专家系统的成功开发与应用，为人工智能走向工程应用和实现理论联系实际具有特别重要的意义。
- 别的人工智能学派出现后，符号主义仍是人工智能的主流派别。

人工智能的主要学派 - 连接主义

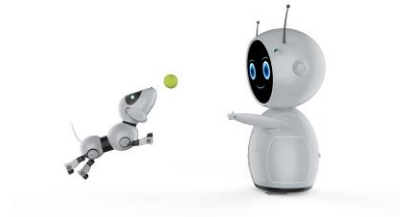
- 连接主义（Connection），又称为仿生学派（Bionicsism）或生理学派（Physiologism）。
 - 连接主义认为人工智能源于仿生学，特别是对人脑模型的研究。他们认为人类的思维基元是神经元，而不是符号处理过程。从神经元开始进而研究神经网络模型和脑模型，成功开辟了人工智能的又一发展道路。



- 网络中的单元可以代表神经元，而连接可以代表突触，就像在人脑中一样。
- 1986年，鲁梅尔哈特等人提出多层网络中的反向传播（BP）算法。此后，连接主义又逐渐兴起。从模型到算法，从理论分析到工程实现。

人工智能的主要学派 - 行为主义

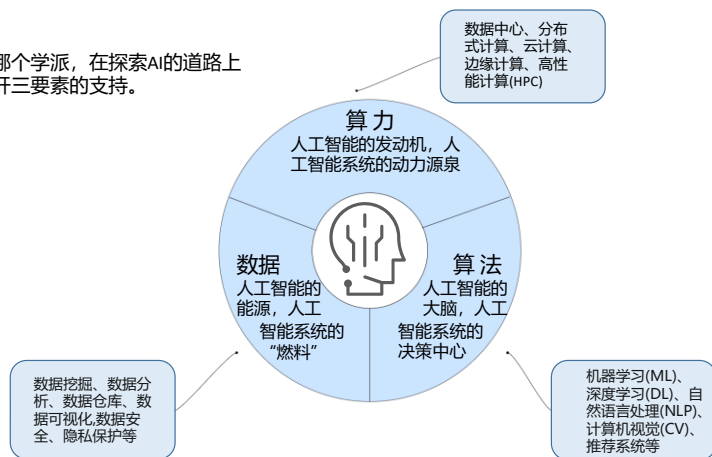
- 行为主义（Actionism），又称为进化主义（Evolutionism）或控制论学派（Cyberneticism）。
- 行为主义认为人工智能源于控制论。他们认为智能取决于感知和行动，智能不需要知识、不需要表示、不需要推理。人工智能可以像人类智能一样逐步进化。智能行为只能在现实世界中与周围环境交互作用而表现出来。



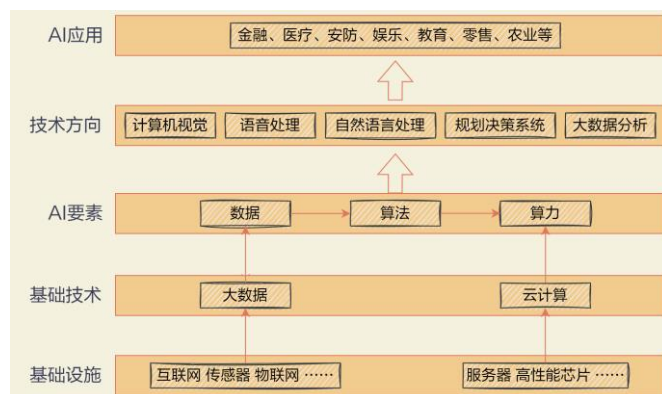
- 行为主义偏向于应用实践，从环境不断学习以不断修正动作。
- 学派代表作：布鲁克斯的六足行走机器人——一个基于感知-动作模式的模拟昆虫行为的控制系统。
- 该学派有点类似自适应控制系统，基于传感器（环境）收集数据并对系统进行作用。

AI发展三要素

不论是哪个学派，在探索AI的道路上都离不开三要素的支持。



AI产业生态



人工智能的分类

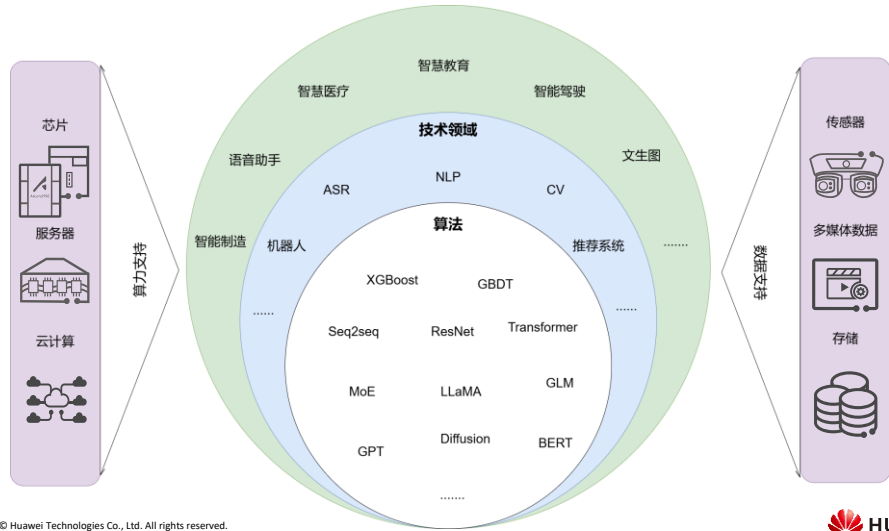
- 强人工智能：
 - 强人工智能观点认为有可能制造出真正能推理（Reasoning）和解决问题（Problem_solving）的智能机器，并且，这样的机器将被认为是有知觉的，有自我意识的。可以独立思考问题并制定解决问题的最优方案，有自己的价值观和世界观体系。有和生物一样的各种本能，比如生存和安全需求。在某种意义上可以看作一种新的文明。
- 弱人工智能：
 - 弱人工智能是指不能制造出真正地推理（Reasoning）和解决问题（Problem_solving）的智能机器，这些机器只不过看起来像是智能的，但是并不真正拥有智能，也不会有自主意识。

- 人工智能的一项重要指标是在没有任何先验知识的前提下，通过完全的self-learning，也就是自我学习，在极具挑战的领域里边达到超人的境界，
- 强人工智能是指在各方面都能和人类比肩的人工智能，因此强人工智能不是仅限于某一领域，而是让机器人全方位实现类人的能力。强人工智能能够进行思考、计划、解决问题、抽象思维、理解复杂理念、快速学习和从经验中学习。目前有一种认为是，如果能够模拟出人脑，并把其中的神经元、神经突触等全部同规模地仿制出来，那么强人工智能就会自然产生。
- 当前我们正处于弱人工智能阶段。弱人工智能的产生减轻了人类智力劳动，类似于高级仿生学。AI的能力仅在某些方面超过了人类。

目录

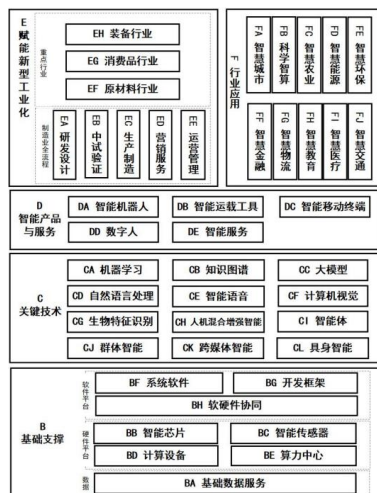
1. 人工智能概述
2. **人工智能技术领域**
 - AI技术领域介绍
 - NLP
 - CV
 - 大模型
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

AI技术领域&应用领域概览



国内人工智能应用技术分布

- 目前AI的通用技术方向主要为：
 - 计算机视觉
 - 是研究如何让计算机“看”并“看”地更快更精确。
 - 自然语言处理
 - 是研究让计算机来理解并运用自然语言。
 - 多模态
 -



- <https://www.gov.cn/zhengce/zhengceku/202407/P020240702716282797987.pdf>

目录

1. 人工智能概述
2. **人工智能技术领域**
 - AI技术领域介绍
 - NLP
 - CV
 - 大模型
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

自然语言处理

- 自然语言处理（Natural Language Processing, NLP）是一门计算机科学领域与人工智能的一个分支，它研究能实现人与计算机之间用自然语言进行有效通信的各种理论和方法。NLP的目标是让计算机理解和生成人类的语言。

- 发展：从基于规则和语言学理论的起步阶段，经过机器学习和深度学习的崛起，到如今大语言模型（LLM）。

- 应用场景：

- 情感分类
 - 文本挖掘
 - 机器翻译
 - 智能问答
 -



自然语言处理任务 - 文本分类

- 定义：

- 输入：

- 一篇文章 d
 - 固定的类别集合 $C = \{C_1, C_2, \dots, C_j\}$

买没几天就降价一点都不开心，闪存跑分就五百多点点 --- 😞

- 输出：

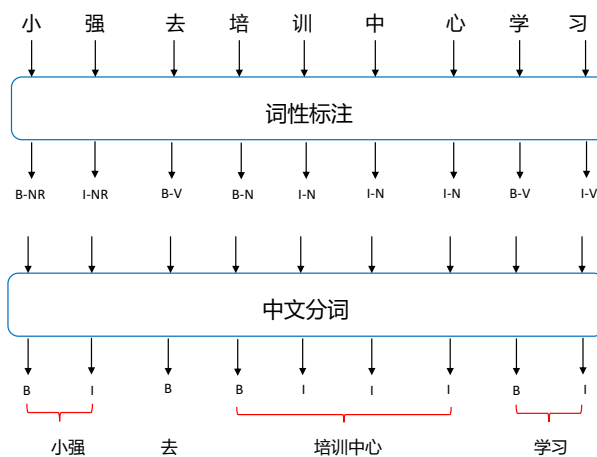
- 预测出的该文章所属的类别 $c \in C$

外观漂亮音质不错，现在电子产品基本上都是华为的了 --- 😄

评论	动力	外观	空间	油耗
2.0T涡轮增压发动机动力强，各种路况表现都不错，高速120超车没压力；外观是我和老婆都比较喜欢的款；后排空间有点小；有点费油啊。	😄	😄	😞	😞

自然语言处理任务 - 序列标注

- 序列标注是给定一个输入序列（通常是一个句子或文本中的字符、词等），使用模型对这个序列的每一个位置标注一个相应的标签，从而形成一个标签序列。这个过程可以看作是一个序列到序列的映射。



自然语言处理任务 - 文本生成

- 文本生成是算法自动产生符合语法规则且富有意义的文本内容。
 - 这一技术可以模拟人类的写作过程，生成各种类型的文本，如新闻报道、电子邮件、文章、社交媒体帖子、诗歌、小说章节、机器翻译等。



知识图谱

- 知识图谱是一种用于表示和组织结构化知识的图形化数据模型。它将现实世界中的实体（如人、地点、事物等）及其之间的关系（如朋友关系、从属关系等）以图的形式进行表示，从而构建一个大规模的、多领域的知识库。
 - NLP技术为知识图谱的构建提供了必要的手段。通过NLP，可以从文本中抽取实体、关系和属性等信息，从而构建和丰富知识图谱。



目录

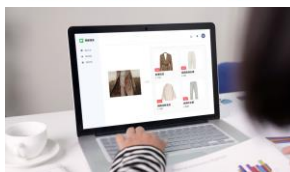
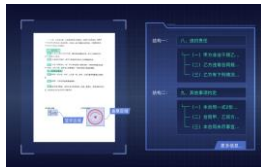
1. 人工智能概述
2. **人工智能技术领域**
 - AI技术领域介绍
 - NLP
 - CV
 - 大模型
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

计算机视觉

- 计算机视觉是一种利用计算机和相应的算法来模拟人类视觉的过程的技术。它涉及从图像或视频中提取信息、分析和理解内容，并做出相应的决策。

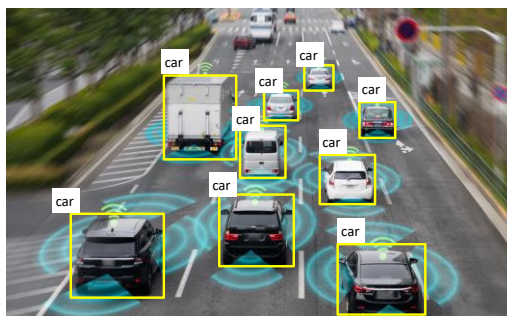
- 应用场景：

- 自动驾驶
- 身份验证
- 智慧城市
- 字符识别
-



计算机视觉任务 - 目标检测

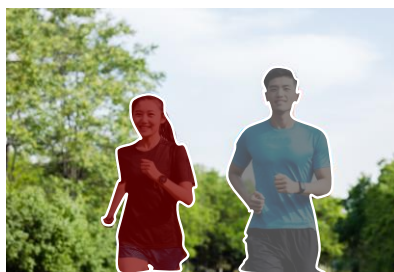
- 目标检测作为人工智能领域中图像处理和计算机视觉的经典技术，有着广泛的应用，例如交通监控、图像检索、人脸识别、人机交互等应用场景都使用了目标检测。目标检测是在图像中检测出人们感兴趣的目标，作为后续智能算法的处理对象。



- 由于各类物体有不同的外观、形状和姿态，加上成像时光照、遮挡等因素的干扰，目标检测一直是计算机视觉领域最具有挑战性的问题。
- 目标识别与目标检测略有不同，尽管它们使用类似的技术。给出一个特定对象，目标识别的目标是在图像中找出该对象的实例。这并不是分类，而是确定该对象是否出现在图像中，如果出现，则执行定位。例如，监控安防摄像头拍摄的实时图像以识别某人的面部。
- 输入输出：
 - 输入：图像。
 - 输出：类别与边界框的位置。

计算机视觉任务 - 图像分割

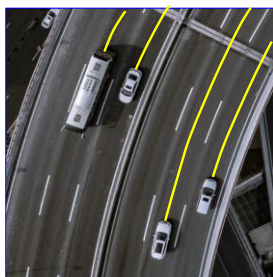
- 图像分割（Segmentation）是根据需要解决的问题，将图像细分为目标内容所构成的不同子区域。
- 图像分割的算法和应用方向有很多，常见的图像分割方法有连通域分割，运动分割，目标分割等。



- 目标分割：语义分割，实例分割。
 - 语义分割（Semantic Segmentation）是指像素级的识别图像，即标注出图像中每个像素所属的对象类别。
 - 实例分割（Instance Segmentation）是指从图像中识别物体的各个实例，并逐个将实例进行**像素级**标注的任务。
- 分割是对图像的像素级描述，它赋予每个像素类别（实例）意义，适用于理解要求较高的场景，如无人驾驶中对道路和非道路的分割。
- 输入输出：
 - 输入：图像。
 - 输出：与输入图像同分辨率的带有各像素类别标签的分割图像。

计算机视觉任务 - 目标跟踪

- 目标跟踪（Object Tracking）是计算机视觉中的基础研究方向，有着广泛的应用，如：智能交通，安防监控，人机交互，无人驾驶等。
- 跟踪算法能够得到感兴趣的目标在时间上的运动轨迹，从而对目标的运动行为进行分析。



计算机视觉任务 - 文字识别

- 文字识别（Optical Character Recognition, OCR）是将图片或扫描件中的文字识别成可编辑的文本，其中使用了文字目标检测，目标分类等图像处理技术。可代替人工录入，提升业务效率。常见的文字识别应用有身份证、驾驶证、行驶证、发票、英文海关单据、通用表格、通用文字等场景文字识别。



计算机视觉任务 - 图像生成

- 图像生成 (Image Generation) : 根据一张图片生成并修改部分区域的图片或者是全新的图片的任务。
 - 超分辨率 (Super Resolution) : 从低分辨率对应物估计高分辨率图像的过程, 以及不同放大倍数下图像特征的预测。
 - 风格迁移 (Style Migration) : 将一个领域或者几张图片的风格应用到其他领域或者图片上。比如将抽象派的风格应用到写实派的图片上。
 - 图像修复 (Image Inpainting) : 修复图像中缺失的地方, 比如可以用于修复一些老的有损坏的黑白照片和影片。



图像修复

- 输入输出:
 - 输入: 真实图像与随机噪声。
 - 输出: 生成图像。

目录

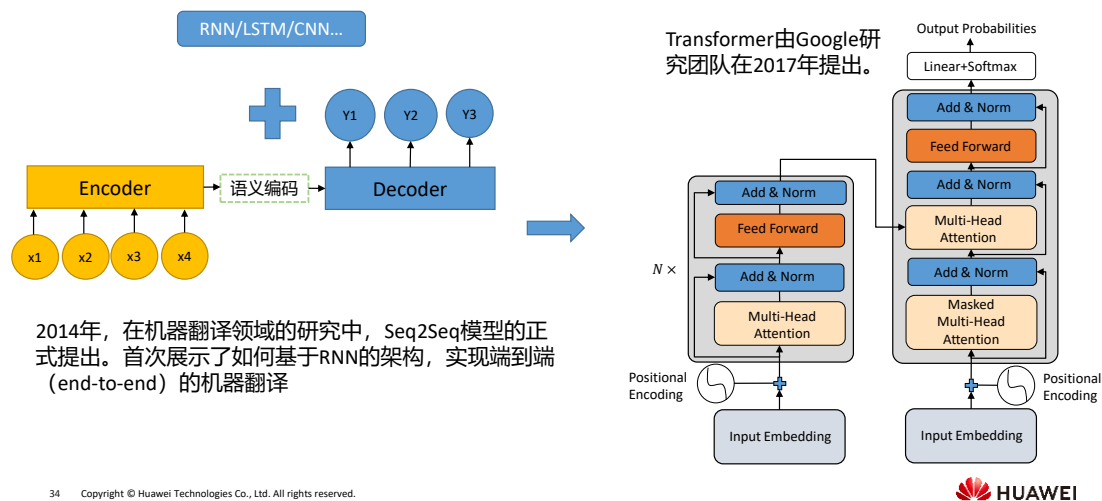
1. 人工智能概述
2. **人工智能技术领域**
 - AI技术领域介绍
 - NLP
 - CV
 - **大模型**
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

什么是大模型

- 在大规模数据上训练、具有海量参数、功能强大的模型统称为大模型。

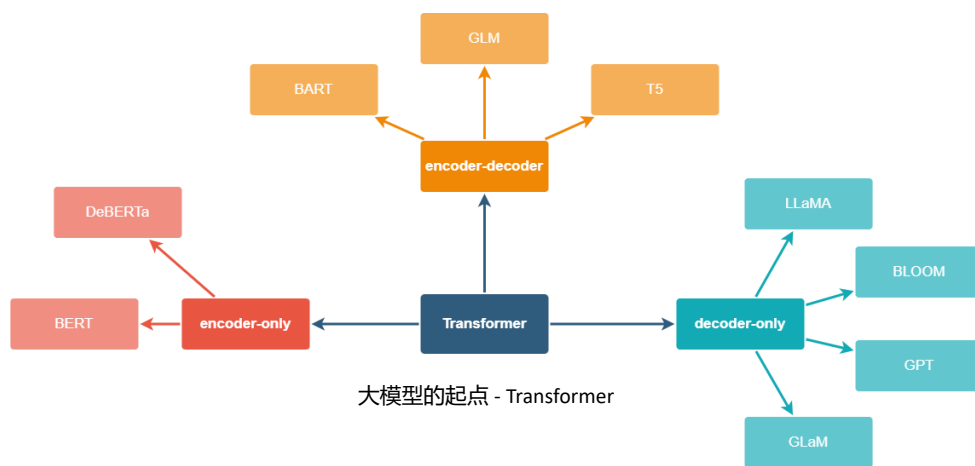


大模型从何而来 (1)



- Seq2Seq模型的核心思想是使用一个编码器网络将输入序列编码为一个固定维度的向量或一系列隐状态，然后解码器网络，逐词生成目标序列。

大模型从何而来 (2)



- 以Transformer为基础架构，诞生了不同分支的大模型。

大模型有多大？

- 大模型，通常参数量在亿级别以上，具有强大的表示能力和泛化能力。

模型	参数 (M)
ResNet101	44.55
ResNet152	60.19
Yolo v7	74.8
VGG16	138

模型	参数 (亿)	tokens
GPT3	1750	3000B
GPT4	1.76万 (猜测)	13T
LLaMA1	70~700	1.4T
LLaMA2		2T
LLaMA3	80~700	15T
GLM	60~1300	400B
T5	110	34B
PaLM2	3400	3.6T
Stable LM 2	210	2T

大模型和小模型

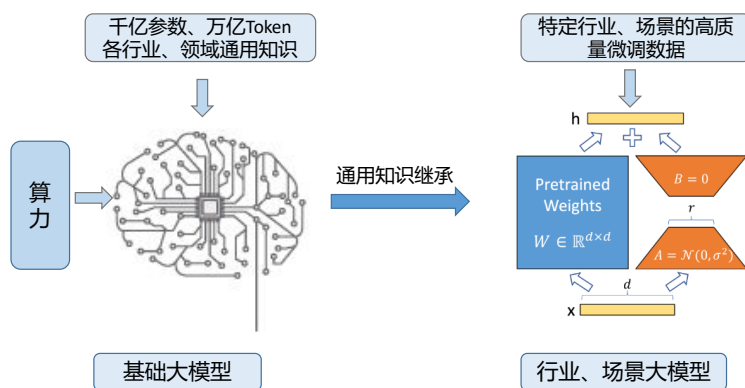


小模型具有轻量级、高效率的特点。
适用于资源受限和单一任务的场景。
如手表只能用来查看时间。



大模型具有更强的处理能力和更高的准确度。
适用于复杂和高度准确性的场景。
如智能手表可以用来查看时间、心跳、睡眠等。

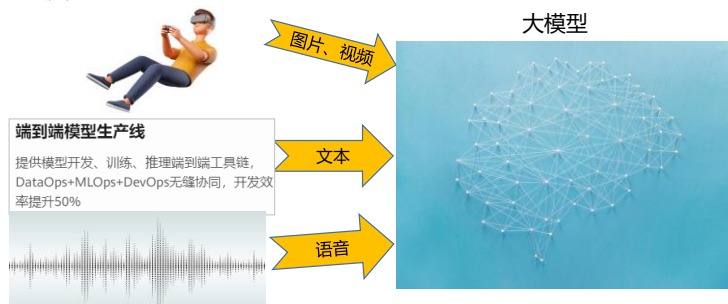
大模型如何学习？



- 大模型需要海量的数据进行学习，对每个行业和场景从零开始训练是需要耗费巨大成本的，因此，当前大模型的学习是基于一个事先在大型数据上训练好的基础模型，然后在一些行业数据上进行微调。

能说会画的大模型

- 与传统的单模态模型（只处理文本、图像或音频等单一类型信息）不同，大模型能够融合多种模态数据（文本、图像、视频、音频等），并进行综合理解和推理。
 - 跨模态理解：能够理解不同模态数据之间的联系，例如从图像中提取信息，并用文字描述出来，或者根据文本描述生成图像、视频。



大模型为AI带来的突破 - 涌现

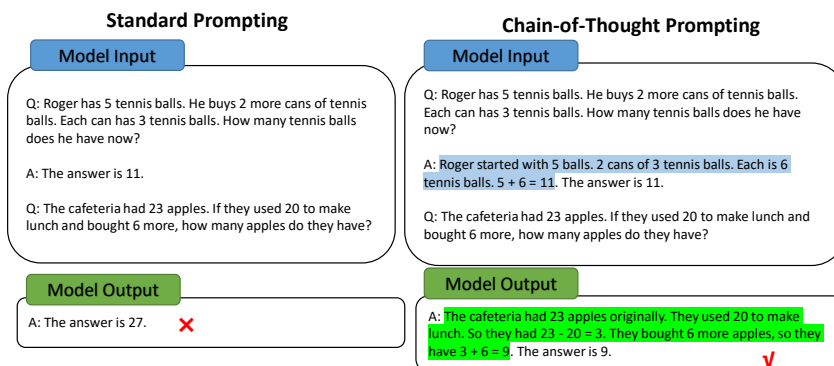
- 当一个复杂系统由很多微小个体构成，这些微小个体凑到一起，相互作用，当数量足够多时，在宏观层面上展现出微观个体无法解释的特殊现象，就可以称之为“涌现现象”。
- 目前大模型有两大类任务被认为具有涌现能力：
 - 上下文学习（ICL, In-Context Learning）。上下文学习可以帮助算法更好地理解一个句子中的词语含义和关系，或者帮助算法更好地识别图像中不同物体之间的关系。
 - 思维链（CoT, Chain-of-thought）。用户把一步一步的推导过程写出来，并提供给大语言模型，这样大模型就能做一些相对复杂的推理任务。



- 在日常生活中也有一些涌现现象，比如雪花的形成、堵车、动物迁徙、涡流形成等。这里以雪花为例来解释：雪花的构成是水分子，水分子很小，但是大量的水分子如果在外界温度条件变化的前提下相互作用，在宏观层面就会形成一个很规律、很对称、很美丽的雪花。
- 在在模型规模不够大的时候，各种任务都处理不好，但是当跨过某个模型大小临界值的时候，大模型就突然能比较好地处理这些任务。

一步一步的思考 - 思维链

- 思维链 (Chain-of-Thought, CoT) 是一种改进的提示策略，用于提高LLM在复杂推理任务中的性能，如算术推理、常识推理和符号推理。CoT没有像In-Context Learning那样简单地用输入输出对构建提示，而是**结合了中间推理步骤**，这些步骤可以将最终输出引入提示。

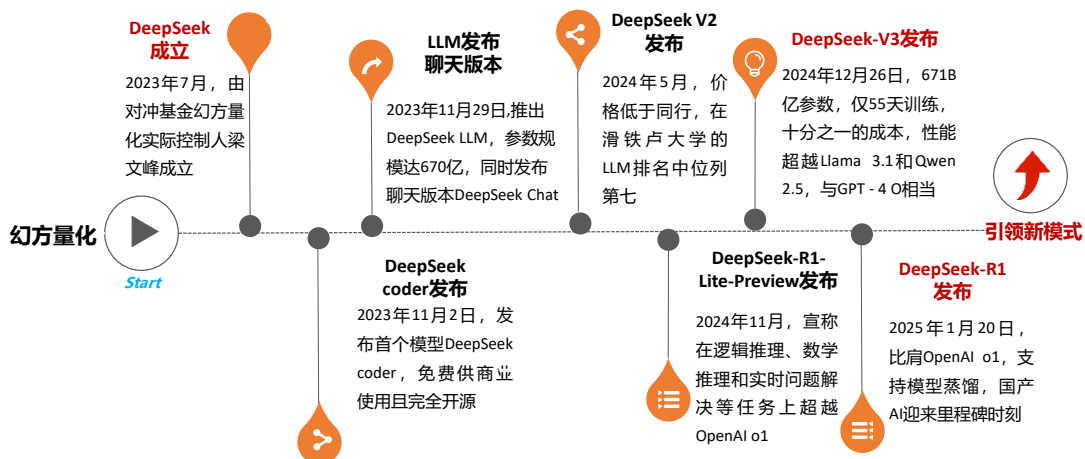


- 相比于之前传统的上下文学习（即通过 $x_1, y_1, x_2, y_2, \dots, x_{test}$ 作为输入来让大模型补全输出 y_{test} ），思维链多了中间的中间的推导提示。
- <https://bbs.huaweicloud.com/blogs/406077>。

目录

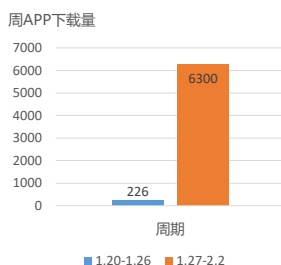
1. 人工智能概述
2. 人工智能技术领域
- 3. DeepSeek概览及其产业影响**
4. 人工智能应用
5. 人工智能的争议和未来
6. 华为人工智能解决方案

DeepSeek成长之路

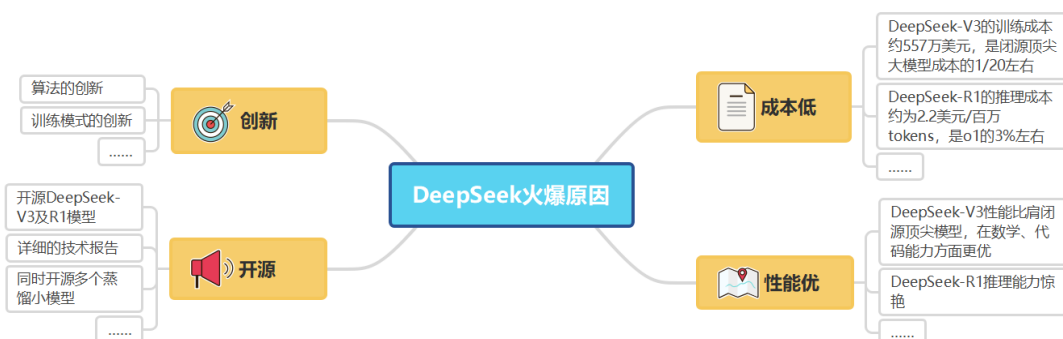


DeepSeek火到什么程度？

- 媒体2025年1月31日报道，DeepSeek的人工智能（AI）软件在全球140个市场中的移动应用下载量排行榜上位居榜首。同日，微软、英伟达、亚马逊宣布全部接入DeepSeek！
- 截至2025年2月9日，DeepSeek的App累计下载量已超1.1亿次，周活跃用户规模最高近9700万。其中，1月20~26日，DeepSeek的App周下载量达到226万次。次周，下载量则直接飙升至6300万次，环比增长超2700%。更重要的是，DeepSeek几乎没在营销上投入一分钱。



DeepSeek为什么这么火？



- 在性能上和世界目前顶尖的GPT-4o等大模型不相上下。但在成本上，OpenAI训练ChatGPT-4花费的成本高达7800万美元，还可能达到1亿美元。而DeepSeek-V3大模型训练成本仅557.6万美元（278.8万 H800 GPU 小时），仅为同性能模型的5%到10%。
- 算法：新的MoE架构，MLA，无辅助损失函数的负载均衡，MTP。
- 训练模式改变，并行方式的创新。

DeepSeek对AI产业的影响



- **挑战巨头垄断地位：**长期以来，AI领域被少数拥有大量资源和技术优势的巨头企业所主导。DeepSeek的出现，以其强大的技术实力和独特的开源策略，对现有巨头的垄断地位构成了挑战。它在短时间内取得的技术突破和广泛的市场应用，吸引了大量用户和开发者，打破了市场原有的平衡，促使巨头企业重新审视自身的技术优势和市场策略。
- **引发行业竞争加剧：**DeepSeek的成功激发了更多企业和研究机构在AI领域的投入和创新。为了在竞争中占据优势，各大企业纷纷加大研发力度，推出更具竞争力的产品和技术。这种竞争不仅体现在技术性能上，还包括成本控制、应用场景拓展等多个方面。例如，在大语言模型市场，DeepSeek引发了价格战，其他企业不得不降低价格以保持竞争力，这使得整个行业的竞争更加激烈，也推动了技术的快速迭代和成本的降低。
- **开源促进技术共享与创新：**DeepSeek采取开源策略，将其模型和技术向全球开发者开放。这一举措打破了AI领域长期存在的闭源模式，促进了全球范围内的技术共享与合作。开发者可以基于DeepSeek的开源模型进行二次开发和创新，加速了新技术的产生和应用。例如，许多开发者利用DeepSeek的模型开发出了针对特定行业的应用程序，推动了AI在各个领域的深入应用。
- **改变AI发展思路：**在DeepSeek出现之前，AI发展往往依赖大量的资金投入、强大的算力支持和大规模的模型训练。而DeepSeek证明了，通过创新的算法和高效的训练方法，即使在有限的资源条件下，也能取得卓越的技术成果。这促使AI行业从单纯追求算力和模型规模，转向更加注重算法创新、资源优化和应用落地，为AI的发展开辟了新的道路。

- 满足多样化应用需求：DeepSeek强大的语言理解和生成能力、数学推理能力以及代码处理能力，使其能够满足不同行业和领域的多样化应用需求。在金融领域，可用于风险评估、投资决策等；在医疗领域，辅助疾病诊断、药物研发等；在教育领域，实现个性化学习、智能辅导等。其广泛的应用场景为AI技术的普及和推广提供了有力支持。
- 降低应用开发门槛：开源的DeepSeek模型和相关技术，降低了企业和开发者进入AI领域的门槛。无需从头开始研发复杂的AI模型，开发者可以利用DeepSeek的技术快速构建应用程序，缩短了开发周期，降低了开发成本。这使得更多的中小企业和创新团队能够参与到AI应用的开发中，进一步拓展了AI市场的规模和范围。
- 推理算力重要性提升：未来大模型开发成本低，大家重心会转向推理，算力结构发生变化，推理算力受关注，且推理算力取决于用户需求。
- 国产替代机遇：DeepSeek开源后，众多芯片厂家为其做适配，全球芯片设计会考虑其架构变化，这对国产芯片设计公司利好，会带动生态发展，国产算力崛起迎来更多机会。

DeepSeek对AI产业发展的启示



- **算法创新：**DeepSeek通过对算法的升级和优化，如在Transformer架构基础上进行改进，采用独特的注意力机制和训练框架，实现了在有限资源下的高效训练和强大性能。这表明，不断进行算法创新是推动AI技术进步的关键。企业和研究机构应加大在算法研究方面的投入，鼓励创新思维，探索新的算法和技术路径，以提升AI系统的性能和效率。
- **应用创新：**除了技术本身的创新，AI的应用创新同样重要。DeepSeek在多个领域的成功应用，展示了AI与不同行业深度融合的潜力。企业应积极探索AI在自身业务中的创新应用，挖掘新的应用场景和商业模式，将AI技术转化为实际的生产力和商业价值。
- **技术共享与协同创新：**开源模式促进了全球AI开发者之间的技术共享和协同创新。通过开放源代码和模型，不同地区、不同背景的开发者的可以共同参与到AI技术的改进和应用开发中，形成一个庞大的创新生态系统。企业和研究机构应积极参与开源项目，分享自己的技术成果，同时吸收他人的经验和智慧，实现共同发展。
- **降低研发成本与风险：**对于中小企业和初创企业来说，参与开源项目可以降低研发成本和风险。他们可以利用开源的AI技术和模型，快速搭建自己的产品和服务，避免了从头开始研发的巨大投入和不确定性。同时，开源社区的支持和反馈也有助于企业不断改进产品，提高市场竞争力。
- **重视AI人才培养：**DeepSeek的成功离不开其背后的优秀AI人才团队。AI产业的快速发展对人才提出了更高的要求，需要培养大量具备扎实的数学、计算机科学基础，掌握AI核心技术，同时具有创新能力和实践经验的专业人才。政府、高校和企业应加强合作，共同推动AI人才培养体系的建设，加大在AI教育方面的投入，培养更多符合行业

需求的专业人才。

- 营造良好的人才发展环境：为了吸引和留住优秀的AI人才，企业和研究机构需要营造良好的人才发展环境。提供具有竞争力的薪酬待遇、广阔的发展空间和良好的科研氛围，鼓励人才创新和探索。同时，建立合理的人才评价机制，注重人才的实际能力和创新成果，为人才的成长和发展提供保障。

DeepSeek的应用

- 随着DeepSeek 的走红，国内外各大厂商密集宣布接入DeepSeek！
- DeepSeek强大的功能和其开源的特性，快速的融入了众多行业，为各领域带了重大的技术变革。



河南云飞科技发展有限公司与DeepSeek 合作，在农业植保领域展开探索。通过DeepSeek 强大的数据分析能力，快速识别病虫害发生规律，预测病虫害的暴发风险。当监测到可能发生病虫害时，会根据实时田间数据，为农户生成个性化植保建议。

2月5日，小艺App 正式上架了DeepSeek-R1 Beta版。此次强强联合，让小艺的能力得到全方位提升。无论是程序员在代码编写时遭遇的棘手难题，学生面对复杂数学计算的困扰，还是职场人需要进行逻辑推理分析，用户只需通过小艺App向DeepSeek-R1模型发起询问，就能迅速获得准确且专业的解答。



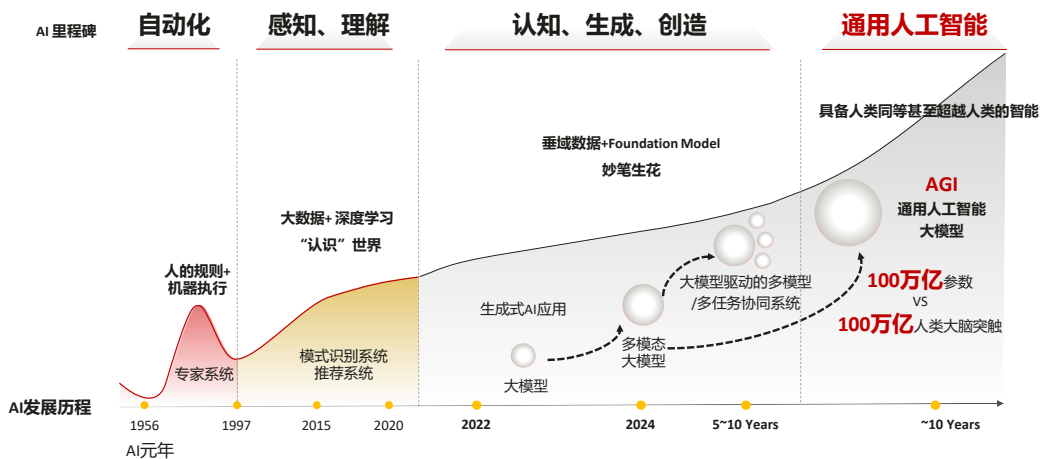
吉利汽车：2月7日，吉利汽车宣布其自研的星睿大模型与大模型完成深度融合。

东风汽车：2月7日，东风汽车宣布已完成DeepSeek全系列大语言模型的接入工作，猛士、奕派、风神、纳米等自主品牌车型将于近期陆续搭载应用。

目录

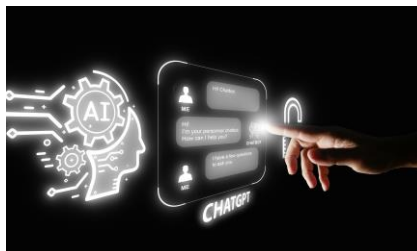
1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
- 4. 人工智能应用**
 - AI应用发展历程
 - AI应用案例
 - AI for Industries - 盘古大模型
5. 人工智能的争议和未来
6. 华为人工智能解决方案

AI应用发展历程 - 从感知理解到创造生成



一个影响重大的应用 - ChatGPT

- ChatGPT是OpenAI在其GPT语言模型系列产品基础上研发的新一代对话式人工智能系统。
 - 推出仅2个月后，活跃用户已经突破1亿，成为史上用户增长速度最快的消费级应用程序。
- ChatGPT具备类人化的语言、语境、意图、逻辑等理解与推理能力，在高质量、多轮次、长文本人机交互对话方面表现突出，且实现了初步的“智能涌现”。
- 目前已经到GPT4-o版本，具备多模态能力和多领域专业知识。



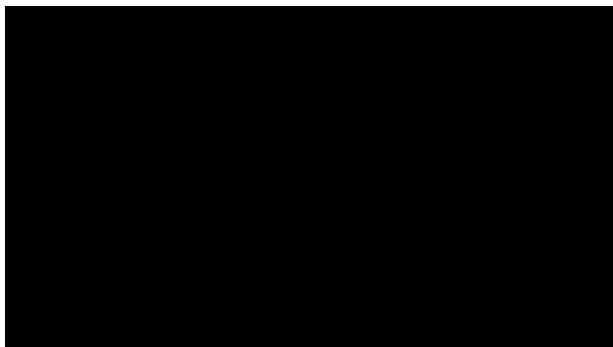
一众 “ChatGPT” 应用

- ChatGPT发布以后，国内外相继发布了众多类似的应用。
- 时至今日，这些应用已经集成了生成文本、代码、视频、图片等多种功能。



AI助力生产

- AI应用在教育、客服与支持、健康咨询、娱乐互动以及多语言翻译等领域都有广泛的应用潜力。
 - 辅助编码；
 - 自媒体文案生成；
 - 艺术创作；
 - 文档解读；
 -



目录

1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
- 4. 人工智能应用**
 - AI应用发展历程
 - AI应用案例
 - AI for Industries - 盘古大模型
5. 人工智能的争议和未来
6. 华为人工智能解决方案

智慧矿山

场景1: 机电硐室——机电硐室存放配电、通信等重要设备。

目标: 实现自动巡检到无人值守

挑战1: 需24小时监控硐室环境; 挑战2: 需人工定时巡检;



场景2: 井下智慧监控

挑战	
输煤皮带	输煤皮带超过20公里, 人工巡检耗时、不安全; 多段皮带需要协同工作, 防止煤出现堆积的情况
违规行为	人员违规行为监控: 佩戴安全帽、抽烟、从设备底下通过、巷道车道走人、皮带坐人等;
违规操作	如瓦斯抽采、探放水作业等;

场景1解决方案: 自动巡检机器人

视频监控: 通过AI机器视觉, 24小时监控, 进行异常报警、敏感区域检测等;

声音检测: 通过AI智能识别机器是否健康运行。



场景2

皮带撕裂监控



皮带跑偏监控



皮带堆煤监控



- <https://e.huawei.com/cn/material/wireless/4d008289c5424b31a055c71eaaed5790>

AI助力守护自然 - 智利生物多样性的保护伞

- 在纳韦尔布塔山脉部署运用华为云和AI技术的“自然守护者”项目，以研究和保护达尔文狐。雨林保护组织Rainforest Connection (RFCx) 开发的声学监测系统“自然守护者”已在多个项目落地且行之有效。
- 它由配备麦克风和天线的太阳能设备组成，这些设备收集周边环境中的声音数据，并通过无线网络将数据传输到云端，并由人工智能进行数据分析。每个设备能够覆盖3平方公里，全天候运行。
- “自然守护者”能监测盗猎的枪声、非法伐木者的卡车与电锯声、不同动物叫声，为研究提供数据。
- 训练后的人工智能分析能够识别不同动物的叫声，使专家能够研究特定物种的分布和行为，通过适应性管理更好地指导环境保护工作。
- 如果识别到了威胁，系统会向护林员的手机应用程序发送实时警报，使其能够就近响应。



摄影：Bujin Zoon & Nahuelbuta Foundation

达尔文狐

Source: <https://www.huawei.com/cn/tech4all/stories/nature-guardians-for-biodiversity-in-chile>

- <https://www.huawei.com/cn/tech4all/stories/nature-guardians-for-biodiversity-in-chile>

AI助力守护自然——声音盾牌守护希腊巴尔干羚羊



- <https://www.huawei.com/cn/tech4all/stories/wildlife-greece>

比特驱动瓦特，共建万物互联的智能电网 - 南方电网智能巡检



数字人



目录

1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
- 4. 人工智能应用**
 - AI应用发展历程
 - AI应用案例
 - AI for Industries - 盘古大模型
5. 人工智能的争议和未来
6. 华为人工智能解决方案

大模型现状 - 千行百业

• 工业生产

汇聚行业数据，使用基础模型构建行业大模型，解决场景碎片化问题，将“作坊式”AI开发转变为“工业化”生产。

• 设备巡检

使用基础模型结合行业知识，解决设备巡检场景训练样本少、故障种类多、目标尺度差异大等挑战。

• 工艺指标预测

工业生产过程工艺指标的预测，例如水泥回转窑质量预测、煤矿洗选煤精煤灰分预测。

• 智慧城市

利用万物检测、万物分割满足智慧城市海量长尾需求，通过语言交互方式高效发现城市事件。

• 智能创意营销

轻松完成多种风格类型的写作，提供创意的商业文案，帮助产品吸引更多的潜在客户，释放无穷创作活力。

• 政企知识检索

基于政企知识进行模型训练，通过对话交互，精准获取用户需求的信息，提高知识获取效率，带来有温度的人工智能服务体验。

• 会议助手

根据流水账改写成标准会议议程，基于会议内容生成文本摘要等，为会议组织者、与会者提供高效便捷的会议体验。

• 影视制作

面向行业提供分镜创作、IP角色设计、短视频生成、场景生成、视频编辑等技术。

- 参考：<https://www.huaweicloud.com/product/pangu.html>。
- **电商营销**：面向行业提供产品设计、品类包装、要素识别、素材生成、广告营销等技术。
- **游戏开发**：面向行业提供角色创作、原画设计、3D素材生成、手办创作、风格转换等技术。
- **平面设计**：面向行业提供广告设计、产品设计、电商/模特图、头像设计、插画设计等技术。

大模型使用层级

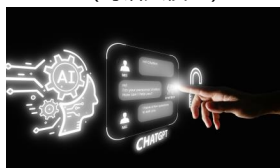
- 按照使用层级划分，大模型可分为基础模型，行业模型和场景模型三层。

基础模型
(基础LLM)



预训练模型，具备通用能力。如基础LLaMA、GLM等模型。

行业模型
(对话大模型)



在行业数据进行微调过的基础模型，针对某一行业属性有能力提升。

场景模型
(智能客服)



在行业场景的数据集上进行微调得到的模型。

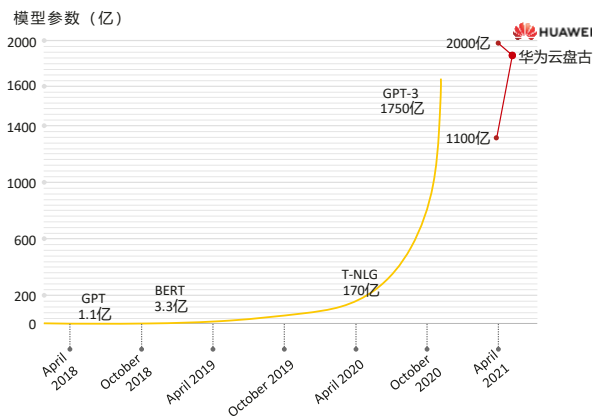
盘古大模型：解决行业难题，释放AI生产力

- 以华为盘古大模型解决方案为例，盘古大模型的应用场景广泛，覆盖了自动驾驶、工业设计、建筑设计、媒体生产、高铁、钢铁、气象、医药等多个领域。

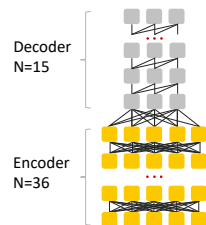


基础模型：盘古自然语言大模型

多轮对话 | 多重意图识别 | 小样本学习



文本数据~40TB



沉淀行业知识，解决低成本大规模定制

行业模型：盘古政务大模型



场景模型：政务智慧办公助手

- 盘古政务大模型基于20多个省市40多万篇公开公文进行训练，通过知识搜索与大模型结合，构建政务智慧办公助手，赋能政务办公一网协同领域。
- 以公文撰写场景为例，盘古政务智慧办公助手实现了多任务演进，支持基于15种公文，基于提纲、角色权责、参考文献照写，为每位公务员打造个人专属的智能办公助手，办公效率提高200%。

智慧办公助手

政务摘要生成

政务公文撰写

政务文档问答



- <https://www.huaweicloud.com/about/takeacloudleap2024/shenzhen-futian.html>

目录

1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
4. 人工智能应用
- 5. 人工智能的争议和未来**
 - 人工智能的争议
 - 人工智能的未来
6. 华为人工智能解决方案

人工智能的争议

- 随着人工智能技术的不断发展，生产力在不断提升，但也因此暴露出诸多争议：
 - 我们现在见到的、听到的还是真的吗？
 - 人工智能存在的伦理问题如何解决？
 - 未来大家都会失业吗？
 - 我们的隐私会泄露吗？怎么保证隐私的安全？
 - 未来人工智能可控吗？
 -



- 所见即所得——不一定是真实的
- 伦理问题——有政策法规约束
- 失业——部分失业
- 隐私有可能泄露——但可以通过差分隐私训练，模型指纹，模型加密等技术对隐私进行保护
- 可控性

还能眼见为实吗？

- 随着AI技术的发展，图像、音频、视频的可信度也越来越低。现在可以通过多模态大模型、GAN（生成对抗网络）等技术制作虚假图像、视频等，让人难分真伪。
 - 如Lyrebird 是一个可以从几分钟录音样本中自动模仿人声音的工具。
 - 利用deepfake网络生成换脸视频。

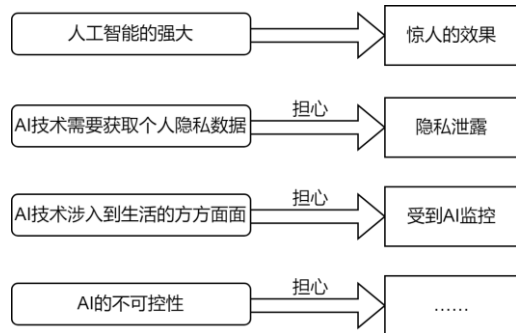
真？



假？

对人工智能的伦理问题的看法

- 歧视和污名化：威胁社会公平正义，限制人们的选择自由。
- AI自主决策机制：会危及人类的自主性，无法对结果进行追溯还原并对错误决策进行有效的补救措施。
- 存在缺点：人工智能系统的目标、方法和决策无法解释。
- 对人类的尊严、思想和生理上的伤害：不安全的环境（比如：与机器人共存），并且机器人容易被恶意利用。人类易失去隐私。容易受到欺骗和操纵。



会失业吗？

- 从人类的发展进程上看，我们一直在寻求提高效率的途径。用蒸汽机，减少了对马匹的需求。实现自动化的每一步都会改变工作生活。在AI的时代，哪些工作会被AI取代呢？
- AI可能会取代：
 - 重复性强的工作
 - 危险工作
 -

AI的发展又会带来哪些机遇和工作岗位？



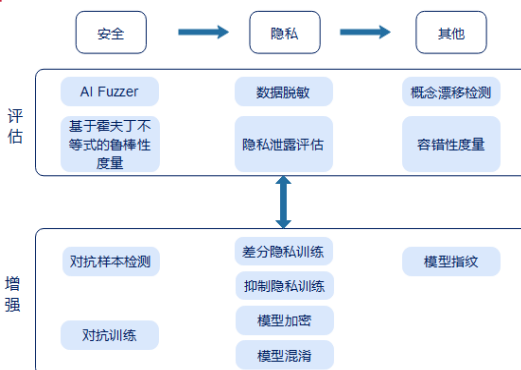
AI



隐私数据安全

- 在大数据和人工智能的时代，人们能够更方便高效地获取信息。然而在获得便利的同时，用户的行为无时无刻不在被记录、被使用。那么应该如何保护隐私数据？

- 一、对用户而言：手机APP在获取用户资料前需要征得用户同意（隐私条款），用户可以对APP需求资料范围进行拒绝，进行主动防护。
- 二、对社会各界而言：积极推进隐私保护相关条款的制定，落实业界基本守则，加强监管。
- 三、针对开发者而言：可以采用机密计算、模型隐私保护、联邦学习或对抗学习等防范逆向攻击及保护用户隐私。或使用成熟的隐私保护框架比如MindArmour。

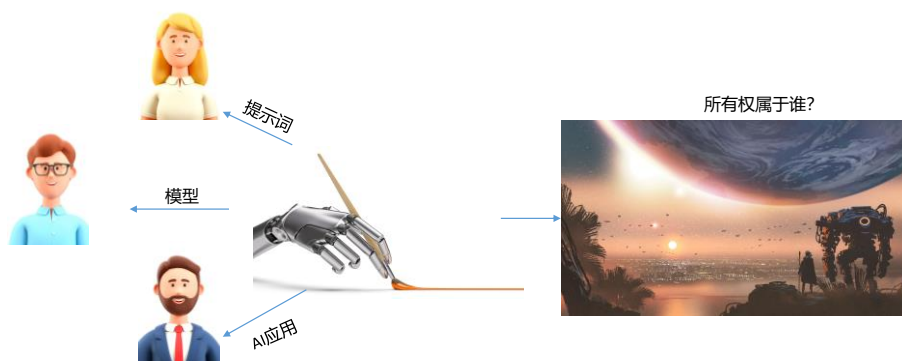


MindArmour典型应用场景

- 学习参考：
- <https://zhuanlan.zhihu.com/p/170475373>
- <https://gitee.com/mindspore/mindarmour>
- <https://mindspore.cn/mindarmour?version=/r1.5/>
- 机密计算：指数据的传输和计算过程是机密的，其保护隐私的代价较大。
- 模型隐私保护：常用到的方法有如数据脱敏、差分隐私训练、模型加密、模型混淆等。（差分隐私可以衡量和控制模型对训练数据的泄露）
- 联邦学习：通常是在不共享数据的情形下进行多方联合机器学习，本质上是一种数据访问受限的分布式机器学习框架。计算节点间只共享梯度不共享原始数据。

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

1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
4. 人工智能应用
- 5. 人工智能的争议和未来**
 - 人工智能的争议
 - 人工智能的未来
6. 华为人工智能解决方案

价值层面 - 广泛的应用场景

- 随着人工智能在各个垂直领域的不断探索，人工智能的应用场景也将不断更新：
 - 智慧交通——公路、机场、铁路、物流、港口.....
 - 能源——电力、油气.....
 - 金融——银行、保险、证券.....
 -

- 数字政府：智慧城市、智慧政务、智慧应急、阳光厨房、智慧水利.....譬如阳光厨房，通过AI机器视觉识别餐盘上食物类别、核算出价格，卡路里等。通过智慧水利水务解决1、黑水治理、排污监测（AI视频算法排污识别，卫星遥感监测，水泵水闸调度）2、水库（四无增强视频，智能供电，“三免”水情一体化杆站）3、河胡（光谱水质分析，5G接入无人机/船）
- 智慧矿山：井下5G，安全监测，违规检测等
- 智慧交通：机场（IOC排队管理——感知旅客超长排队等待：AI+视频分析，机位调度：运筹优化AI算法+大数据+IOT+GIS+视频+仿真.....），公路（全息路口，无人驾驶.....）

价值层面 - 不断增长的市场 (1)

市场规模

2024年全球人工智能市场规模预计达到6382亿美元（2023年5381亿美元）。

AI IT总投资

2022年全球人工智能IT总投资规模为1,288亿美元，2027年预计增至4,236亿美元，五年复合增长率（CAGR）约为26.9%

人工智能企业

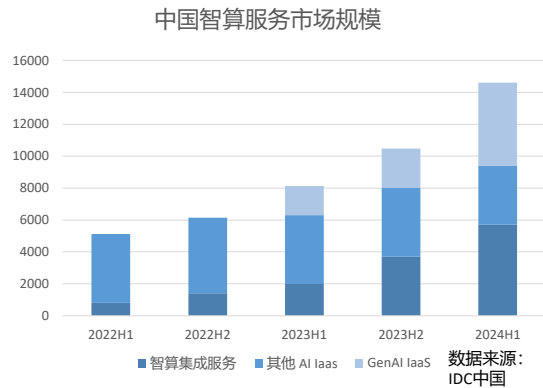
据全国组织机构代码数据服务中心统计，2024年上半年中国人工智能企业数量同比增长35.65%，预计后续会进一步提升。

大模型解决方案

2024年上半年中国AI大模型解决方案市场的规模为13.8亿元，预计在2024-2028年期间年均复合增长率为56.2%，到2028年市场规模将达到211亿元。

价值层面 - 不断增长的市场 (2)

- 2024上半年中国智算服务整体市场同比增长79.6%，市场规模达到146.1亿元人民币。其中，智算集成服务市场同比增长168.4%，市场规模达57.0亿元人民币；GenAI IaaS市场同比增长203.6%，市场规模达52.0亿元人民币；Other AI IaaS市场同比缩减13.7%，市场规模达37.1亿元人民币。



新旧动能快速切换，面向GenAI的算力支出已经成为智算服务市场主阵地

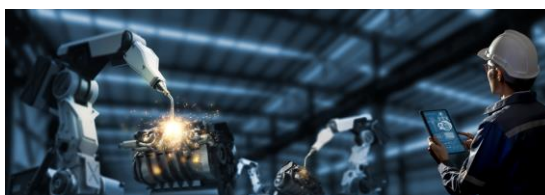
价值层面 - 生产效率提升

- 行业层面，人工智能创新对制造业推动作用最为突出，智能制造可显著提升产品质量和生产效率，据统计，AI改造后工厂研发周期缩短20.7%，生产效率提高34.8%。
- 企业层面，劳动密集型和技术密集型企业应充分发挥AI创新效应，前者可利用人工智能提高劳动生产率，后者本身研发实力强、AI创新难度较低。



一群人共同工作

↓
少量人控制一群机器工作



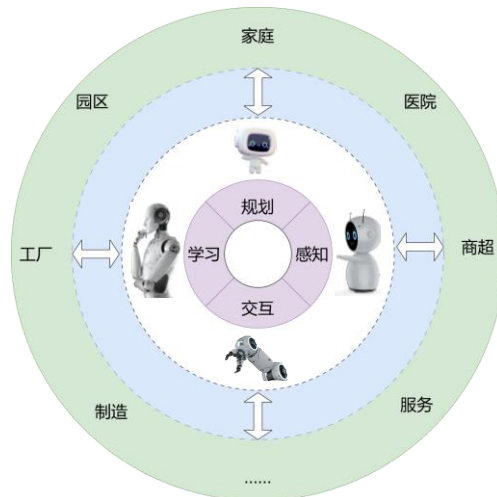
技术层面 - 通用人工智能

- AGI是一种能够执行任何人类智能活动的计算机系统，具有全方位的智能，包括自我学习、自我改进、自我调整及跨领域、跨模态的学习、推理和决策能力。



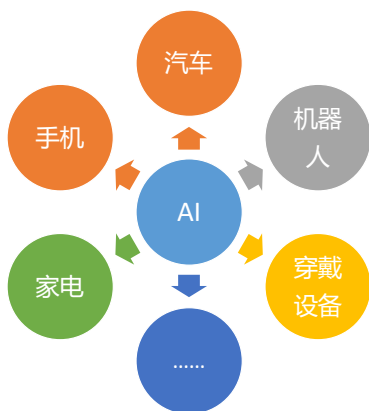
技术层面 - 具身智能

- 具身智能是将人工智能融入各类物理本体形成的智能系统，使本体具备在物理环境中拥有自主感知、学习、决策及行动的能力，从而完成物理环境及任务的灵活适应。



设备层面 - AI设备

- 从智能手机到AI手机，更自然的交互方式，让手机真正的智能化。搭配更多的穿戴设备获取全新的体验。
- 智能家电：食谱推荐、节能优化、语言交互...，获得更舒适的体验。



- 智能汽车，实现真正的智能驾驶，提高行车安全性和舒适性。
- 智能机器人，具备更强的感知和决策能力，胜任管家、护理等工作。

目录

1. 人工智能概述
2. 人工智能技术领域
3. DeepSeek概览及其产业影响
4. 人工智能应用
5. 人工智能的争议和未来
- 6. 华为人工智能解决方案**

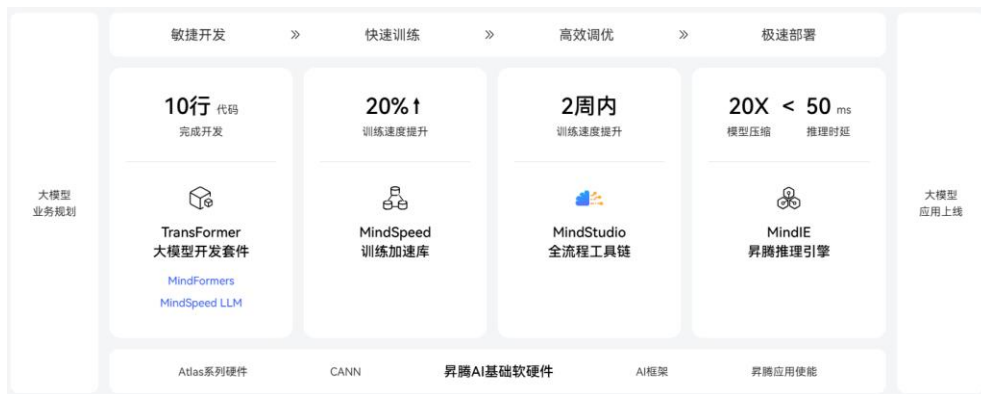
昇腾AI基础软硬件平台



- 全栈，是技术功能视角，是包括芯片、芯片使能、训练和推理框架和应用使能在内的全堆栈方案；
- 全场景，是包括公有云、私有云、各种边缘计算、物联网行业终端以及消费类终端等全场景的部署环境。
- 这其中，Atlas作为华为全栈全场景AI解决方案的基石，基于昇腾 AI处理器提供了模块、板卡、服务器等不同形态的产品，满足客户全场景的算力需求。
- **应用使能：**提供全流程服务（ModelArts），分层API和预集成方案。
- **MindSpore：**支持端、边、云独立的和协同的统一训练和推理框架。
- **CANN：**芯片算子库和高度自动化算子开发工具。
- **Atlas：**基于Ascend系列AI处理器，通过丰富的产品形态，打造面向“端、边、云”的全场景AI基础设施方案。
- 华为的AI全场景包括公有云、私有云、各种边缘计算、物联网行业终端以及消费类终端等端、边、云的部署环境。

昇腾AI全流程使能大模型创新落地

从大模型的开发训练到推理部署，昇腾AI全流程使能大模型创新落地



Ascend AI处理器，为计算注入强劲的智能因子



Ascend 推理系列

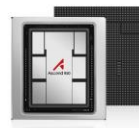
极致高效AI SoC



Atlas 800T A2训练服务器



Atlas 800 推理服务器



Ascend 训练系列

性能强悍AI处理器



Atlas 800T A2训练服务器



Atlas 900 AI 集群

- 基础这两个AI芯片的Atlas硬件有
- 推理系列：Atlas200加速模块，Atlas500智能小站，Atlas300I Pro推理卡，Atlas 300I Duo推理卡，Atlas 500Pro智能边缘服务器，Atlas800推理服务器，Atlas 300V Pro视频解析卡
- 训练系列：Atlas 300T Pro训练卡，Atlas 900Pod，Atlas 900T PoD Lite，Atlas 800训练服务器，Atlas 900AI集群。
- <https://www.hiascend.com/hardware/product>

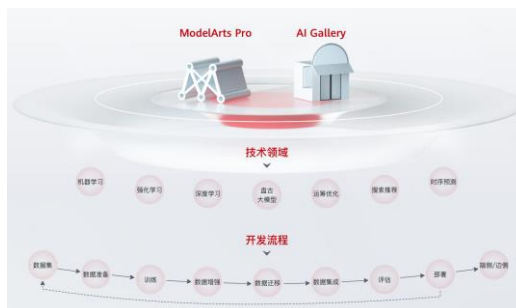
ModelArts：基于昇腾芯片的全栈自主可控

自主可控AI技术栈

智慧应用		科研	交通	教育	制造	金融	医疗	城市
全栈AI	软件平台	全流程AI使能平台 ModelArts						
		算法开发环境	自动学习	盘古大模型 (D)	行业数据集			
		数据标注	分布式调测	AI科学计算	昇腾预置算法			
		数据管理	AI训练	AI推理	AI资产市场			
	硬件基础设施	自主可控的国产芯片使能工具链						
		深度学习计算框架MindSpore			昇腾算子开发工具MindStudio			
		昇腾芯片异构计算驱动CANN						
	基建设施	自主知识产权的国产芯片						
		鲲鹏 (通用算力)		昇腾NPU (AI算力)				
	基建设施	规划与设计	机房土建	电气工程建设	风火水电建设			

ModelArts 是面向开发者的一站式 AI 开发平台，为机器学习与深度学习提供海量数据预处理及交互式智能标注、大规模分布式训练、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期 AI workflow。

企业级开发平台ModelArts Pro，满足行业定制化需求。



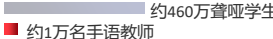
- ModelArts训练流程参考：<https://www.huaweicloud.com/product/modelarts.html>

华为AI助力各行各业

听障群体持续学习成为难题

听障人士长期手语学习过程中
三大挑战

1 词汇匮乏 |  69000个词语 《现代汉语词典》
8200个词语 新事物 新技术 ...

2 师资短缺 |  约460万聋哑学生
约1万名手语教师

3 标准不一 | 方言差异 语法差异 认知差异

AI赋能，让手语学习更简单、可持续



本章总结

- 本章首先介绍了人工智能的定义、人工智能的发展史，接着描述了人工智能的热门领域与应用场景，最后讨论了人工智能的争议话题和人工智能的发展趋势。

思考题

1. （多选）以下哪些属于计算机视觉的应用场景？
- A. 知识图谱
 - B. 语义分割
 - C. 智能驾驶
 - D. 视频分析

- 答案：
 - BCD

学习推荐

- 华为人才在线网站
 - <https://e.huawei.com/cn/talent/#/home>
- 昇腾官网
 - <https://www.hiascend.com/>
- MindSpore
 - <https://mindspore.cn/>
- 华为云
 - <https://www.huaweicloud.com/>

Thank you.

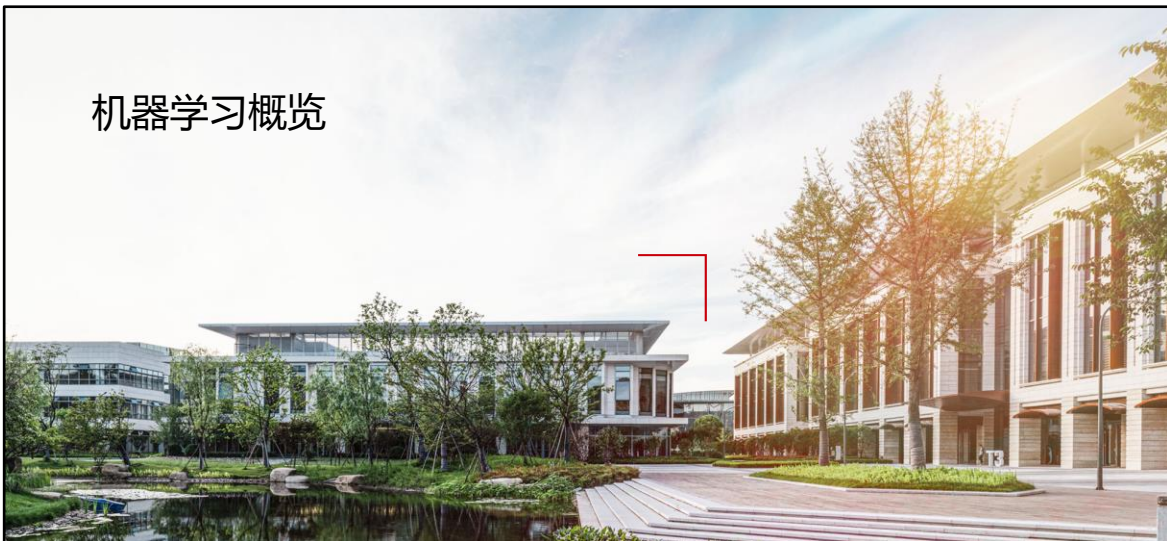
把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



机器学习概览



前言

- 机器学习基于大量数据，通过算法模型的训练与优化，实现了对未知信息的预测与判断，展现了强大的模式识别与决策能力。
- 本章节将介绍机器学习的基础知识，从机器学习的基本概念，逐步深入到其核心理论、算法、模型评估与优化策略，以及在实际应用场景中的实施与挑战。

目标

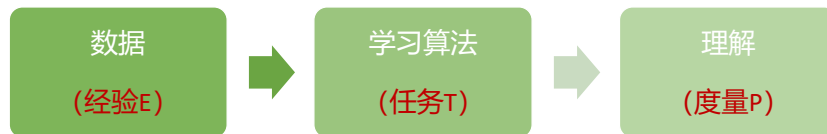
- 学完本课程后，您将能够：
 - 掌握学习算法定义与机器学习的流程；
 - 了解常用机器学习算法；
 - 了解超参数、梯度下降和交叉验证等概念。

目录

1. **机器学习算法**
2. 机器学习的分类
3. 机器学习的整体流程
4. 其他机器学习重要方法
5. 机器学习的常见算法

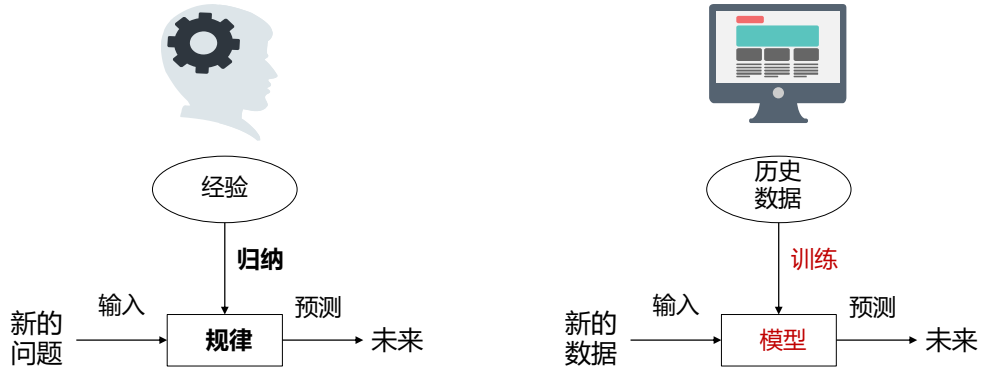
机器学习算法 (1)

- 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。

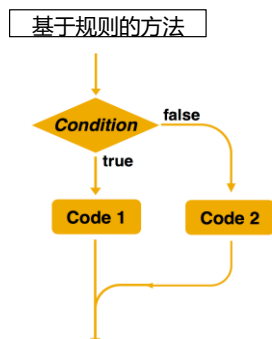


- 深度学习是机器学习的一个特定分支，要想充分理解深度学习，必须对机器学习的基本原理有深刻的理解。
 - 任务 T ：机器学习系统应该如何处理样本。
 - 性能度量 P ：评估机器学习算法的能力。如准确率、错误率。
 - 经验 E ：大部分学习算法可以被理解为在整个数据集上获取经验。有些机器学习的算法并不是训练于一个固定的数据集上，例如强化学习算法会和环境交互，所以学习系统和它的训练过程会有反馈回路。根据学习过程中的不同经验，机器学习算法可以大致分为无监督算法和监督算法。
- 学习下围棋问题：
 - 经验 E_1 ：自己和自己对弈 —— 无监督、间接学习
 - 经验 E_2 ：自对弈过程中询问人类 —— 半监督学习
 - 经验 E_3 ：人类历史对弈 —— 有监督、直接学习
- 手写识别问题：任务 T ：识别手写文字。性能标准 P ：分类准确率。经验 E ：已分类样例库（有监督、直接学习）。
- 机器人的上进心：寻找新的棋局，同样的局面下通过微小变化来磨砺技能，等效于扩充了训练样例。

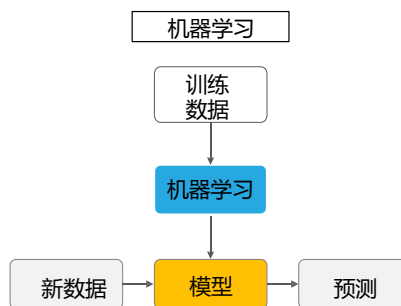
机器学习算法 (2)



机器学习算法与传统基于规则的区别



- 使用显性编程来解决问题
- 规则可以被人工明确



- 使用样本训练
- 决策的规则复杂或者难以描述
- 由机器自动学习规则

什么时候该使用机器学习 (1)

- 问题的解决方案很复杂，或者涉及到大量没有明确分布函数的数据。
- 遇到如下情况，可以考虑使用机器学习：

规则十分复杂或者无法描述，如语音识别



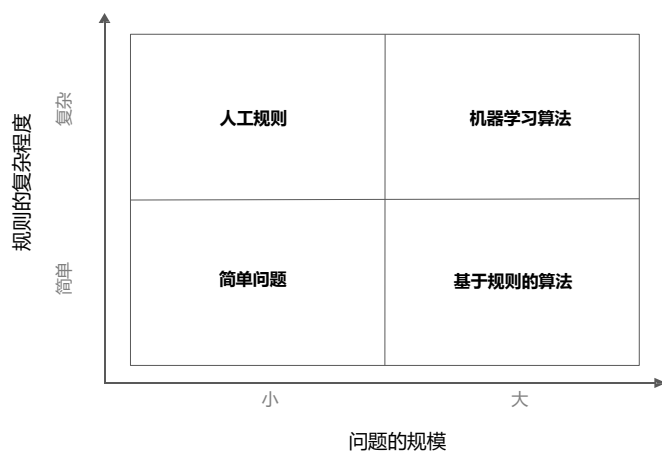
任务的规则会随时间改变，比如词性标注，随时都会产生新的词或词义。



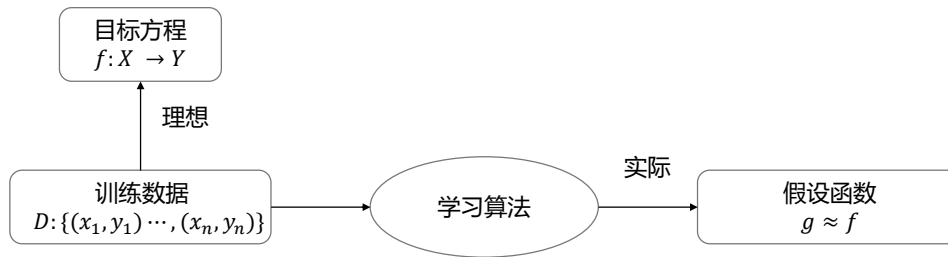
数据分布本身随时间变化，需要程序不停的重新适应，如预测商品销售的趋势



什么时候该使用机器学习 (2)



机器学习算法的理性认识



- 目标函数 f 未知，学习算法无法得到一个完美的函数 f
- 假设函数 g 逼近函数 f ，但是可能和函数 f 不同

机器学习解决的主要问题

- 机器学习可以解决多种类型的任务，下面列出常见的三种：
 - 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 $f: R^n \rightarrow (1, 2, \dots, k)$ 。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。
 - 聚类：对大量未知标注的数据集，按数据的内在相似性，将数据划分为多个类别，是类别内的数据相似程度较大，而类别间的相似性比较小。可以被运用在图片检索，用户画像等场景中。
- 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。



目录

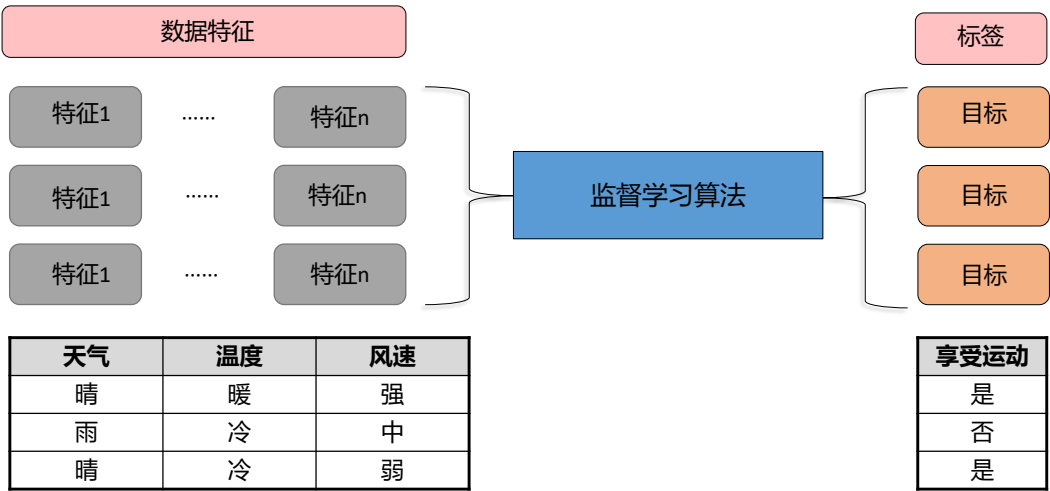
1. 机器学习算法
- 2. 机器学习的分类**
3. 机器学习的整体流程
4. 其他机器学习重要方法
5. 机器学习的常见算法

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：在一个任务中，试图让学习器自动地对大量未标记数据直接帮助少量有标记数据进行模型训练。
- 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。
- 随着算法的发展，除以上学习方式以外，还有自监督学习、对比学习、生成式学习等。

- 监督学习：通俗的来讲，我们给计算机一堆选择题（训练样本），并同时提供了它们的标准答案，计算机努力调整自己的模型参数，希望自己推测的答案与标准答案越一致越好，使计算机学会怎么做这类题。然后再让计算机去帮我们做没有提供答案的选择题（测试样本）。
- 非监督学习：通俗的来讲，我们给计算机一堆选择题（训练样本），但是不提供标准答案，计算机尝试分析这些题目之间的关系，对题目进行分类，计算机也不知道这几堆题的答案分别是什么，但计算机认为每一个类别内的题的答案应该是相同的。
- 半监督学习：传统监督学习通过对大量有标记的训练样本进行学习以建立模型用于预测新的样本的标记。例如，在分类任务中标记就是样本的类别，而在回归任务中标记就是样本所对应的实值输出。随着存储数据能力的高度发展，在很多实际任务中可以容易地获取大批未标记数据，而对这些数据赋予标记则往往需要耗费大量的人力物力。例如，在进行 Web 网页推荐时，需请用户标记出感兴趣的网页，但很少有用户愿意花很多时间来提供标记，因此有标记的网页数据比较少，但 Web 上存在着无数的网页，它们都可作为未标记数据来使用。
- 强化学习：通俗的来讲，我们给计算机一堆选择题（训练样本），但是不提供标准答案，计算机尝试去做这些题，我们作为老师批改计算机做的对不对，对的越多，奖励越多，则计算机努力调整自己的模型参数，希望自己推测的答案能够得到更多的奖励。不严谨的讲，可以理解为先无监督后有监督学习。

监督学习



监督学习 - 回归问题

- 回归：反映了样本数据集中样本的属性值的特性，通过函数表达样本映射的关系来发现属性值之间的依赖关系。
 - 下周股票能给我带来多少收益？
 - 周二的温度会是多少摄氏度？

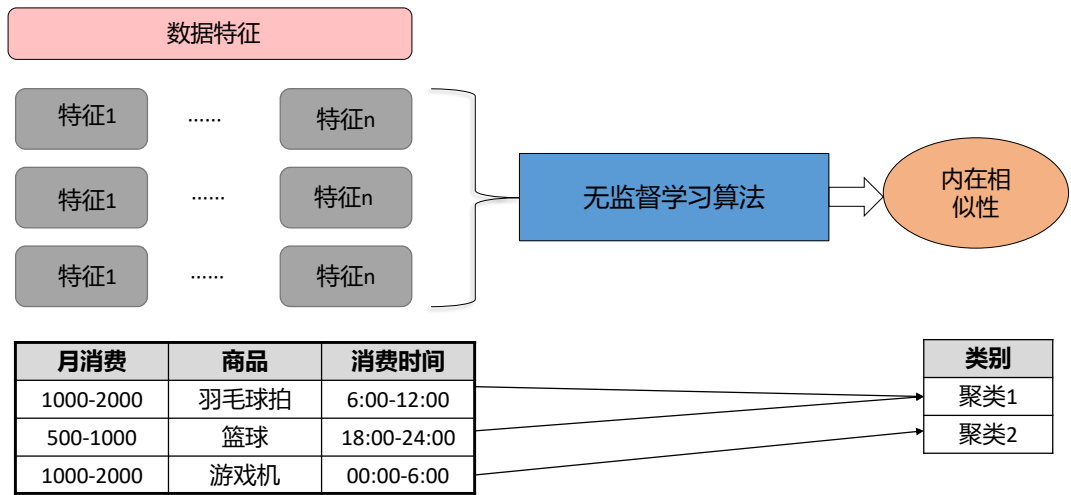
Monday	Tuesday
38°	?

监督学习 - 分类问题

- 分类：分类问题属于有监督学习，通过分类模型，将样本数据集中的样本映射到某个给定的类别中。
 - 饮料瓶属于哪一类垃圾？
 - 收到的邮件是否是垃圾邮？



无监督学习

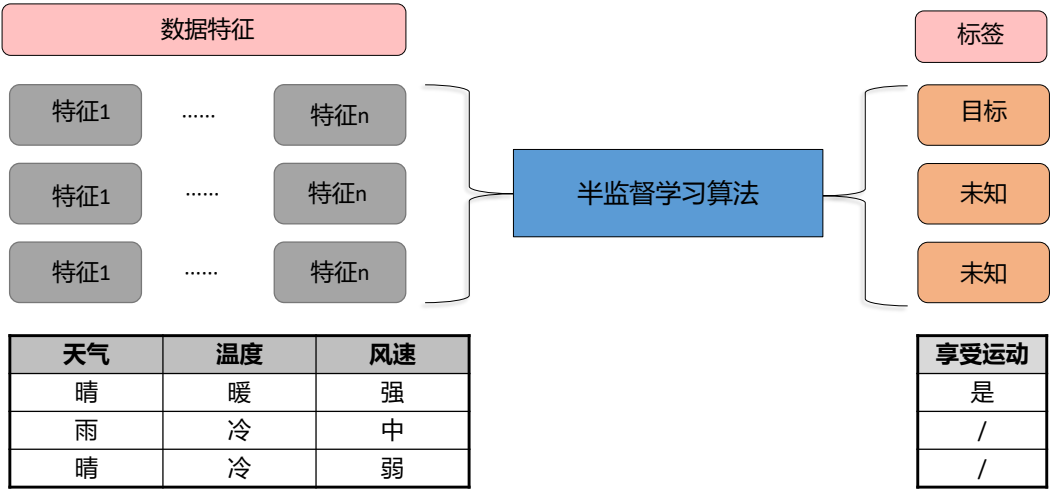


无监督学习 - 聚类问题

- 聚类：通过聚类模型，将样本数据集中的样本分为几个类别，属于同一类别的样本相似性比较大。
 - 找出相同族类的鱼？
 - 为用户推荐可能喜欢的电影？

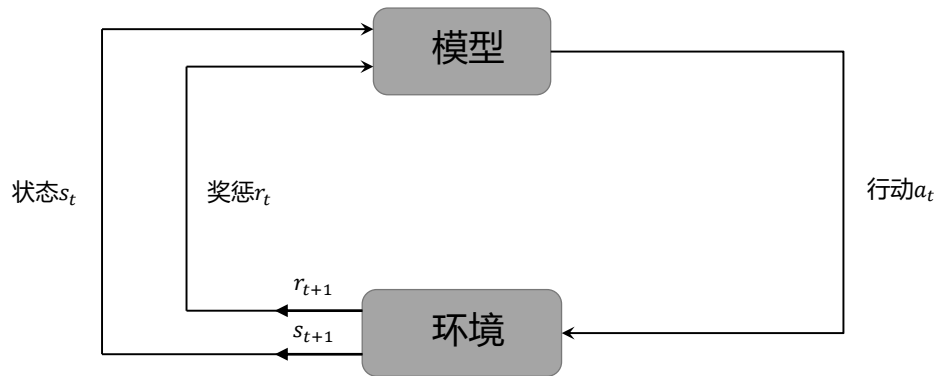


半监督学习



强化学习

- 模型感知环境，做出行动，根据状态与奖惩做出调整 and 选择。



- 强化学习通过一系列的行动最大化“奖励函数”来学习模型，
- 不管是好行为还是坏行为都可以帮助强化学习模型学习
- 例子，自动驾驶汽车通过不断与环境交互来学习

强化学习 - 最佳行为

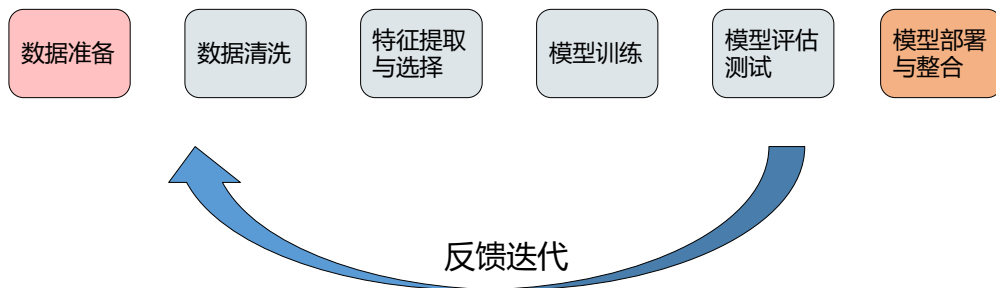
- 强化学习：强化学习总是在寻找什么样的行动才是最佳的？
 - 自动驾驶汽车：黄灯开始闪了，是刹车还是加速通过？
 - 对于一个吸尘机器人，电量剩余10%，还剩一小块区域未进行打扫，是继续吸尘还是回去充电？



目录

1. 机器学习算法
2. 机器学习的分类
- 3. 机器学习的整体流程**
4. 其他机器学习重要方法
5. 机器学习的常见算法

机器学习整体流程



机器学习基本概念 - 数据集

- 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。

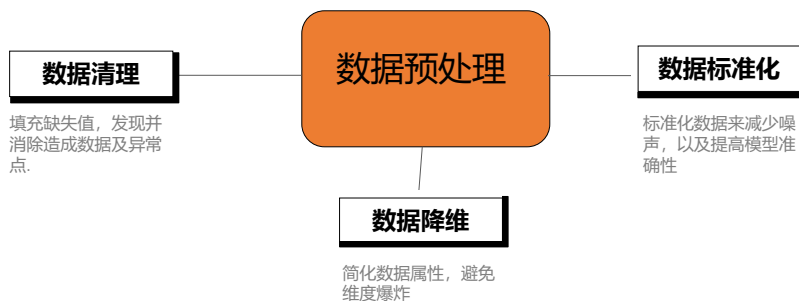
数据概览

- 典型的数据集的样式

		特征1	特征2	特征3	标签
	序号	面积	学区	朝向	房价
训练集	1	100	8	南	1000
	2	120	9	西南	1300
	3	60	6	北	700
	4	80	9	东南	1100
测试集	5	95	3	南	850

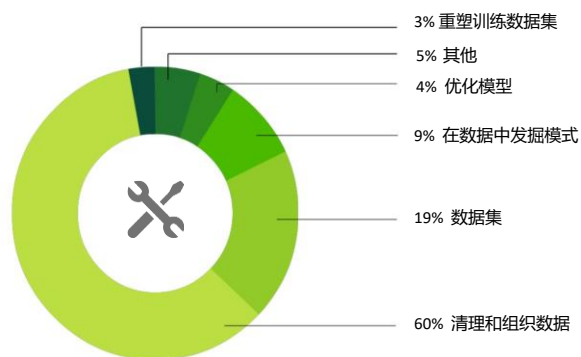
数据处理的重要性

- 数据对于模型来说就是至关重要的，是模型能力的天花板，没有好的数据，就没有好的模型。



数据清理的工作量

- 数据科学家在机器学习的工作统计



CrowdFlower Data Science Report 2016

数据清理

- 大部分的机器学习模型所处理的都是特征，特征通常是输入变量所对应的可用于模型的数值表示。
- 大部分情况下，数据需要经过预处理后才能够为算法所使用，预处理的操作主要包括以下几个部分：
 - 数据过滤
 - 处理数据缺失
 - 处理可能的错误或者异常值
 - 合并多个数据源数据
 - 数据汇总

“脏”数据

- 真实的数据中通常会出现一些数据质量问题：
 - 不完整: 数据中缺少属性或者包含一些缺失的值。
 - 多噪音: 数据包含错误的记录或者异常点。
 - 不一致: 数据中存在矛盾的、有差异的记录。

#	Id	Name	Birthday	Gender	IsTeacher?	#Students	Country	City
1	111	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	Spain	Madrid
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

无效重复项

格式错误

属性依赖

缺失值

无效的值

应该在另外一列的值

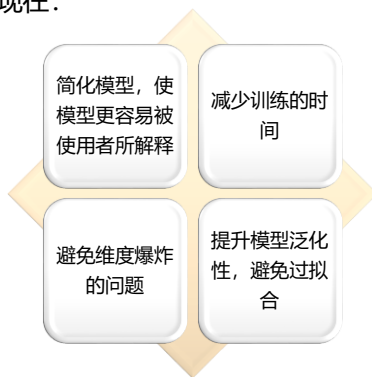
错误的拼写

数据的转换

- 对数据进行初步的预处理后，需要将其转换为一种适合机器学习模型的表示形式，以下是一些常见的数据转化的形式。
 - 在分类问题中，将类别数据编码成为对应的数值表示。
 - 数值数据转换为类别数据以减少变量的值（对年龄分段）。
 - 其他数据：
 - 在文本中，通过词嵌入将词转化为词向量（一般使用word2vec模型，BERT模型等）。
 - 处理图像数据（颜色空间，灰度化，几何变化，haar特征等，图像增强）。
 - 特征工程：
 - 对特征进行归一化、标准化，以保证同一模型的不同输入变量的值域相同。
 - 特征扩充：对现有变量进行组合或转换以生成新特征，比如平均数。

特征选择的必要性

- 通常情况下，一个数据集当中存在很多种不同的特征，其中一些可能是多余的或者与我们要预测的值无关的。
- 特征选择技术的必要性体现在：



特征选择的方法 - Filter

- 过滤法（Filter）在选择特征的时候是独立的，与模型本身无关。



通过评估每个特征和目标属性之间的相关性，这些方法应用一个统计度量来为每个特征赋值。然后根据分数对这些特性进行排序，这可以用来帮助保留或消除特定特征。

常见的方法：

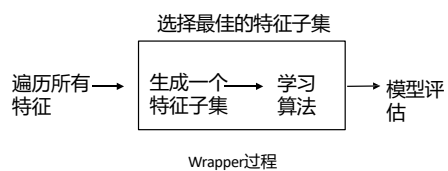
- 皮尔逊相关系数
- 卡方系数
- 互信息

过滤法的局限：

- 过滤方法倾向于选择冗余的变量，因为它们没有考虑特征之间的关系。

特征选择的方法 - Wrapper

- 包装器 (Wrapper) 方法使用一个预测模型来对特征子集进行评分。



包装器方法将特征选择问题视为一个搜索问题，在这个问题中，将评估和比较不同的组合。在这里，预测模型用于评估特征的组合的工具，根据模型的准确性进行评分。

常见的方法

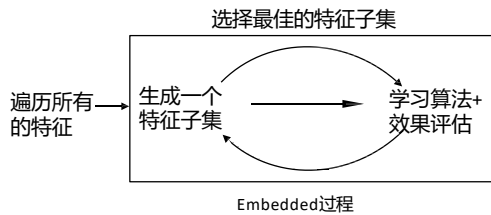
- 特征递归消除法

包装法的局限：

- 因为包装器方法为每个子集训练一个新模型时，所以它们的**计算量非常大**。
- 其特征选择的方法通常为**特定类型的模型**提供了性能最好的特性集。

特征选择的方法 - Embedded

- 嵌入法 (Embedded) 的方法将特征选择作为模型构建的一部分。



最常见的嵌入式特征选择方法是**正则化方法**。

正则化方法也称为惩罚方法，它在优化预测算法时引入额外的约束，使模型的复杂度降低，也就是减少了特征的数量。

常见的方法

- LASSO回归

- 强调利用了模型的惩罚属性，对不重要的特征就行消除

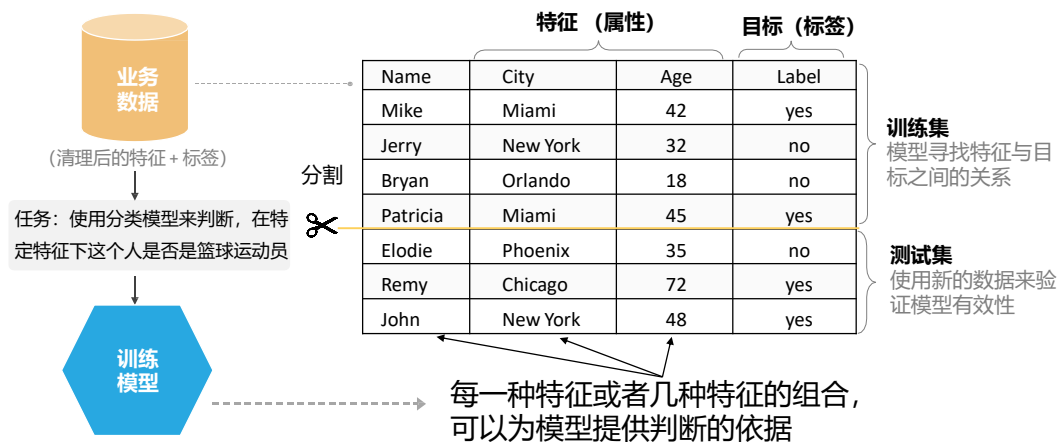
模型构建项目整体流程



- 在做完数据清理特征提取等操作后，我们要开始进行模型的构建了，整体上来说，模型构建一般是遵循如上步骤（有监督学习）。

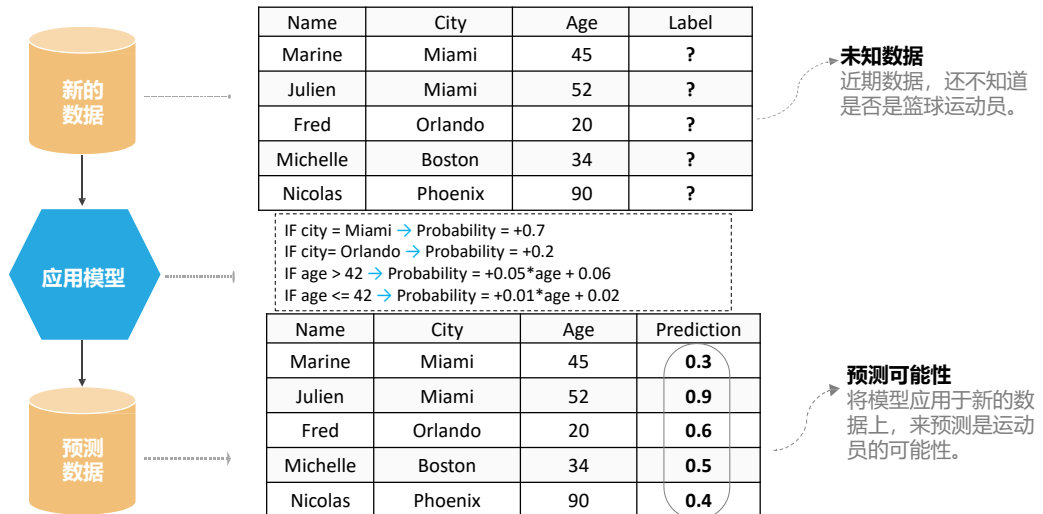
有监督学习示例 - 学习阶段

- 使用分类模型来判断，在特定特征下这个人是否是篮球运动员。



- 模型小例子

有监督学习示例 - 预测阶段



- 模型小例子
- 这里的模型可以认为是类似与决策树模型

什么是好的模型？



- **泛化能力**
能否在实际的业务数据也能预测准确
- **可解释性**
预测的结果是否容易被解释
- **预测速率**
每一条数据的预测需要多长时间

- 那么我们用什么指标来判断模型？
- 第一个泛化能力最重要

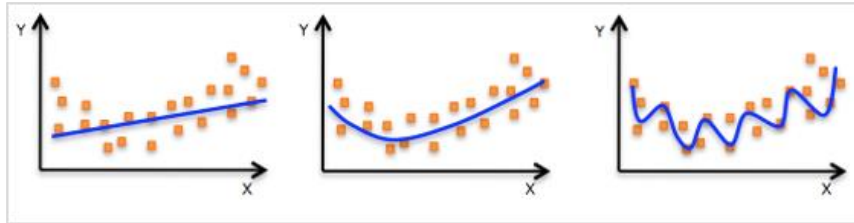
模型的有效性 (1)

- 泛化能力：机器学习的目标是使学得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，学得模型适用于新样本的能力称为泛化能力，也称为鲁棒性。
- 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。

- 一旦给定了问题假定的形式，所有可能的函数构成了一个空间，我们称之为假设空间。机器学习的问题可以看做是在假设空间中搜索合适的拟合函数的问题。
- 过拟合：过于复杂的数学模型比较容易发生过拟合，适当简化数学模型、在过拟合前提前结束训练、采用Dropout/Weight Decay手段等，可以减轻过拟合现象。
- 欠拟合：过于简单的数学模型，或训练时间太短，则可能引起欠拟合。对于前者，需使用更复杂的模型，对于后者，只需延长训练时间，即可有效缓解欠拟合现象。

模型的有效性 (2)

- 模型的容量：指其拟合各种函数的能力，也称为模型的复杂度。
 - 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

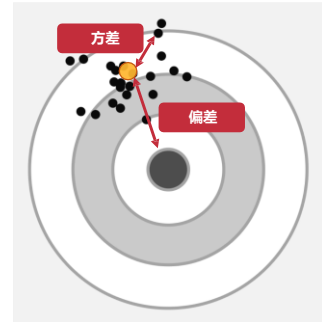
好的拟合

过拟合
学习了噪声

- 有效容量受限于算法、参数、正则化等。

过拟合的原因 - 误差

- 最终预测的总误差= 偏差² + 方差 + 不可消解的误差
- 预测误差总的来说可以被拆解出两种主要的我们需要关注的子形式：
 - 来源于“偏差”的误差
 - 来源于“方差”的误差
- 方差 (Variance):
 - 模型的预测结果在均值附近的偏移的幅度
 - 来源于模型在训练集上对小波动的敏感性的误差
- 偏差 (Bias):
 - 模型的预期(或平均)预测值与我们试图预测的正确值之间的差异。



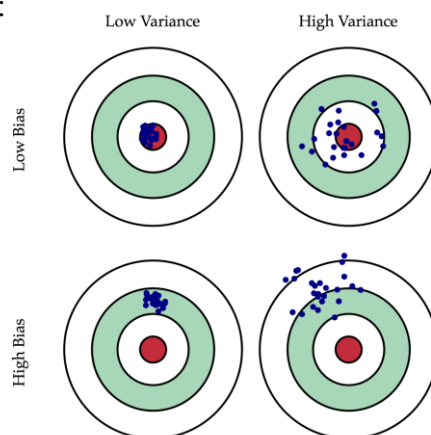
- 关于不可消解的误差
 - 理论上来说，如果有无限量的数据与完美的模型，这个不可消解的误差是可以被消解的，但实际上所有模型都是不完美的，并且我们的数据量是有限的。

方差与偏差

- 以下四种形式表示了方差与偏差不同组合方式：

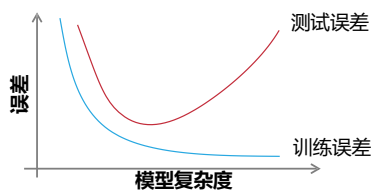
- 低偏差&低方差→好模型
- 低偏差&高方差→欠优模型
- 高偏差&低方差→欠优模型
- 高偏差&高方差→不良模型

- 理想状态下，我们想要选择一个模型，它既能准确地捕获训练数据中的规律，又能很好地概括不可见数据（即新数据）。但是，通常我们不可能同时完成这两件事。



模型的复杂度与误差

- 随着模型复杂度上升的增加，训练误差逐渐减小。
- 同时，测试误差会随着复杂的增大而减小到某一点，继而反向增大，形成一条凸曲线。



机器学习性能评估 - 回归

- MAE (Mean Absolute Error) 平均绝对误差, 越趋近于0表示模型越拟合训练数据。

$$MAE = \frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

- MSE (Mean Square Error)

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- R²取值范围[0,1], 值越大表示模型越拟合训练数据, 其中TSS表示样本之间的差异情况, RSS表示预测值与样本值之间的差异情况

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y})^2}$$

机器学习性能评估 - 分类 (1)

- 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。

机器学习性能评估 - 分类 (2)

度量	公式
准确率、识别率	$\frac{TP + TN}{P + N}$
错误率、误分类率	$\frac{FP + FN}{P + N}$
真正例率、查全率、召回率 (<i>recall</i>)	$\frac{TP}{P}$
特效性、真负例率	$\frac{TN}{N}$
查准率 (<i>precision</i>)	$\frac{TP}{TP + FP}$
F_1 值, 查准率和查全率的调和均值	$\frac{2 \times precision \times recall}{precision + recall}$
F_β 值, 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。

- 查准率： $P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5\%$
- 查全率： $R = \frac{TP}{P} = \frac{140}{170} = 82.4\%$
- 准确率： $ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 75\%$

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

目录

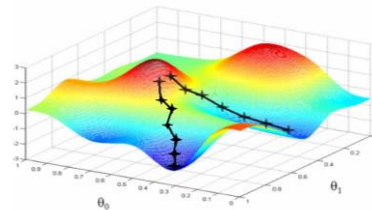
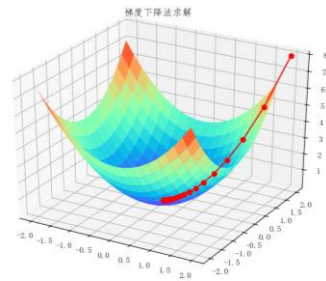
1. 机器学习算法
2. 机器学习的分类
3. 机器学习的整体流程
- 4. 其他机器学习重要方法**
5. 机器学习的常见算法

机器学习训练方法 - 梯度下降

- 梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，该方向为当前位置最快下降方向，公式如下：

$$w_{k+1} = w_k - \eta \nabla f_{w_k}(x^i)$$

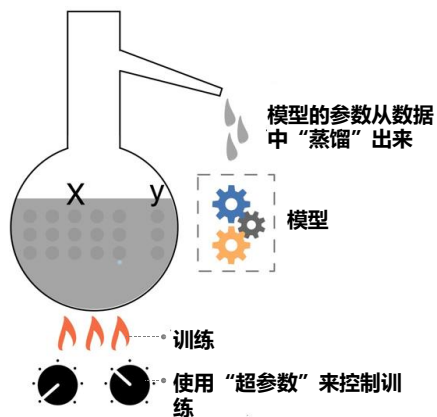
- 其中 η 成为学习率， i 表示第 i 条数据。权重参数 w 每次迭代变化的大小。
- 收敛：目标函数的值变化非常小或达到最大迭代次数。



- 通常情况下，在接近目标点区域的极小范围内，下降的幅度是越来越小的。但是，在目标点一定范围内可能存在梯度不稳定的情况。

模型中的参数与超参数

- 模型中不但有参数，还有超参数的存在。其目的是为了让模型能够学习到最佳的参数。
 - 参数由模型自动学习；
 - 超参数由人工手动设定。



模型的超参数

- 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

模型超参数是模型外部的配置

- Lasso/Ridge回归当中的 λ
- 训练神经网络的学习速率、迭代次数、批次大小、激活函数及神经元的数量
- 支持向量机的 C 和 σ 超参数
- KNN中的 K
- 随机森林当中的树的棵数

常见的模型超参数

- 参数作为模型从历史训练数据中学到的一部分，是机器学习算的关键。有以下特征：
 - 进行模型预测时需要模型参数。
 - 模型参数值可以定义模型功能。
 - 模型参数用数据估计或数据学习得到。
 - 模型参数一般不由实践者手动设置。
 - 模型参数通常作为学习模型的一部分保存。
- 模型参数的一些例子包括：
 - 人造神经网络中的权重。
 - 支持向量机中的支持向量。
 - 线性回归或逻辑回归中的系数。

超参数的搜索过程及方法

超参数搜索的一般过程

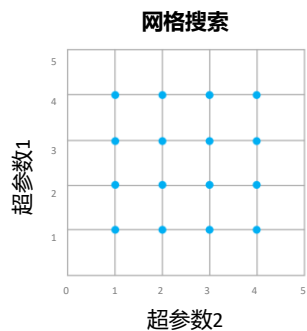
1. 将数据集分成训练集、验证集、测试集。
2. 在训练集上根据模型的性能指标对模型参数进行优化。
3. 在验证集上根据模型的性能指标对模型的超参数进行搜索。
4. 步骤2和步骤3交替迭代进行，最终确定模型的参数和超参数，并在测试集中评价模型的优劣。

搜索算法 (步骤3)

- 网格搜索
- 随机搜索
- 启发式智能搜索
- 贝叶斯搜索

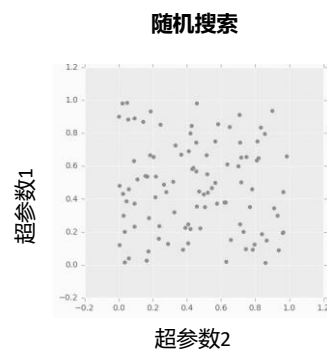
超参数的调节方法 - 网格搜索

- 网格搜索尝试穷举搜索所有可能的超参数组合形成超参数值网格。
- 在实践中，人工指定的范围和步骤之间的值。
- 网格搜索是一种昂贵且耗时的方法。
 - 这种方法在超参数数目相对较少的情况下工作良好。因此一般机器学习算法中可行，但在神经网络等情况下就行不通了（见深度学习部分）。



超参数的调节方法 - 随机搜索

- 在超参数搜索空间较大的情况下，采用随机搜索，会优于网格搜索。
- 随机搜索实现了对参数的随机搜索，其中每个设置都是从可能的参数值的分布中进行取样，试图找出最佳的参数子集。
- 注意：
 - 首先在一个粗范围内搜索，然后根据最佳结果出现的位置，缩小范围。
 - 某些超参数比其他的更加重要，随机搜索过程中会影响搜索的偏向。

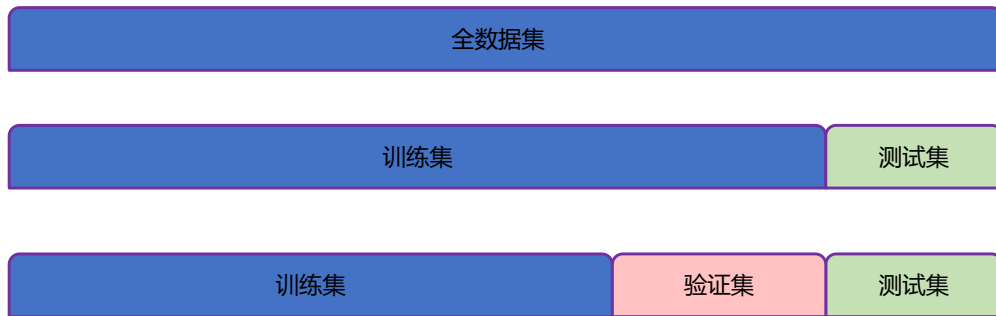


交叉验证 (1)

- 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。

- 将数据集分成固定的训练集和固定的测试集后，若测试集的误差很小，这将是有点问题的。一个小规模的测试集意味着平均测试误差估计的统计不确定性。使得很难判断算法 A 是否比算法 B 在给定任务上做得更好。当数据集有十万计或者更多的样本时，这不会是一个严重的问题。当数据集太小时，也有替代方法允许我们使用所有的样本估计平均测试误差，代价是增加计算量。
- k -折交叉验证： k 一般大于等于2，实际操作时一般从3开始取，只有在原始数据集合数据量小的时候才会尝试取2。 $K - CV$ 可以有效的避免过学习以及欠学习状态的发生，最后得到的结果也比较具有说服力。

交叉验证 (2)

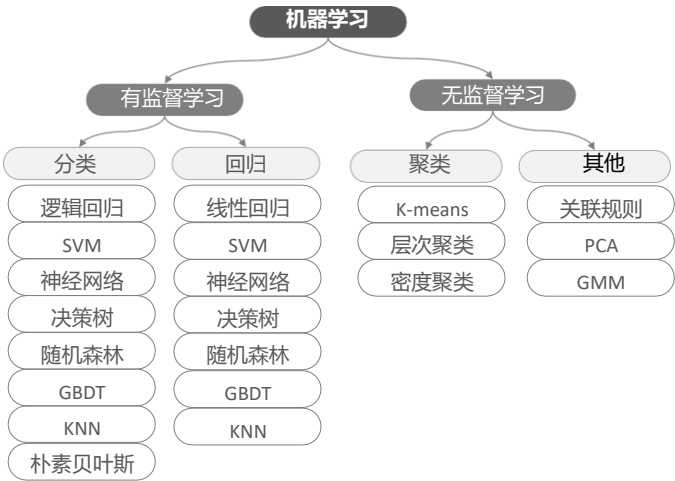


- 注意：K折交叉验证的K值，属于超参数

目录

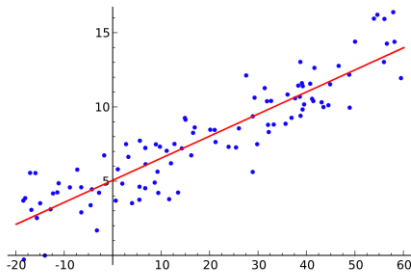
1. 机器学习算法
2. 机器学习的分类
3. 机器学习的整体流程
4. 其他机器学习重要方法
- 5. 机器学习的常见算法**

机器学习算法总览

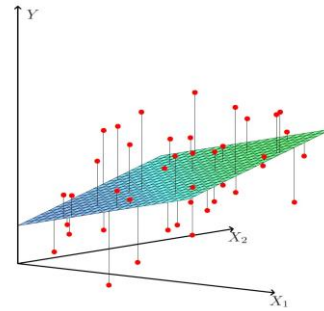


线性回归 (1)

- 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。
- 线性回归是一种有监督学习。



一元线性回归



多元线性回归

- 回归分析中, 只包括一个自变量和一个因变量, 且二者的关系可用一条直线近似表示, 这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量, 且因变量和自变量之间是线性关系, 则称为多元线性回归分析。学出来的不一定是一条直线, 只有在变量 x 是一维的时候才是直线, 高维的时候是超平面。比如房子的售价由面积, 户型, 区域等多种条件来决定, 通过这些条件来预测房子的售价可抽象为一个线性回归问题。
- 图中显示的是多项式线性回归

线性回归 (2)

- 线性回归的模型函数如下。其中 w 为权重参数， b 为偏置， x 表示样本。

$$h_w(x) = w^T x + b$$

- 模型预测与真实值的关系如下。其中 y 表示真实值， ε 表示误差。

$$y = w^T x + b + \varepsilon$$

- 误差 ε 受到众多因素独立影响，线性回归模型假设误差 ε 服从正态分布。根据正态分布函数与最大似然估计，可以求出线性回归的损失函数为：

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2$$

- 我们想要预测值与真实值最接近，就是求损失值最小。可以使用梯度下降的方法求出当损失函数达到最小的时候的权重参数 w ，继而完成模型构建。

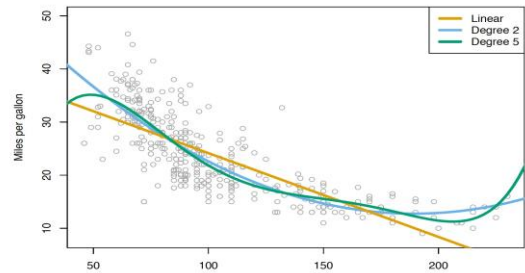
线性回归扩展 - 多项式回归

- 多项式回归，是线性回归的拓展，通常数据集的复杂度会超过用一条直线来拟合的可能性，也就是使用原始的线性回归模型会明显欠拟合。解决的办法就是使用多项式回归。

$$h_w(x) = w_1x + w_2x^2 + \dots + w_nx^n + b$$

这里n次方就是多项式回归维度（degree）。

多项式回归仍然是线性回归的一种，是因为其非线性体现在特征的维度上，而其权重参数 w 之间的关系仍然是线性的。



线性回归与多项式回归对比

线性回归与防止过拟合

- 正则项有助于减少过拟合， w 值在样本空间中不能过大/过小，可以在目标函数之上增加一个平方和损失：

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2 + \lambda \|w\|_2^2$$

- 正则项（norm）：这里这个正则项叫做L2-norm，这个使用这个损失函数的，线性回归也叫**Ridge回归**。

$$J(w) = \frac{1}{2m} \sum (h_w(x) - y)^2 + \lambda \|w\|_1$$

- 追加了绝对值损失的线性回归叫做**Lasso回归**。

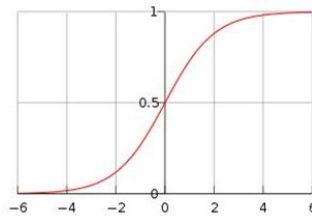
逻辑回归 (1)

- 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：

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

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

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。



逻辑回归 (2)

- 逻辑回归与线性回归都是广义线性模型。逻辑回归在线性回归的基础上引入了非线性因素(sigmoid函数)，并设定了阈值，因此可以处理二分类问题。
- 根据逻辑回归的模型函数，可以通过最大似然估计计算出逻辑回归的损失函数为：

$$J(w) = -\frac{1}{m} \sum (y \ln h_w(x) + (1-y) \ln(1-h_w(x)))$$

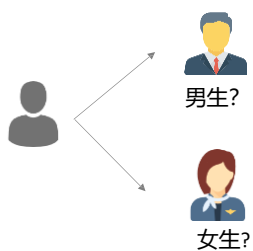
- 其中 w 为权重参数， m 为样本个数， x 表示样本， y 表示真实值。同样可以通过梯度下降算法求得所有权重参数 w 的值。

- 追加正则项之后，逻辑回归也可以防止过拟合。

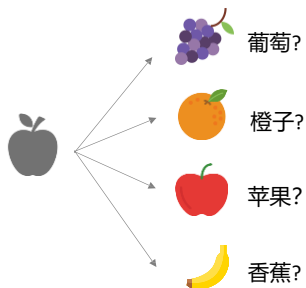
逻辑回归扩展 - Softmax函数 (1)

- 逻辑回归常用于二分类问题，如果是多分类问题，我们通常使用Softmax函数。

二分类问题



多分类问题



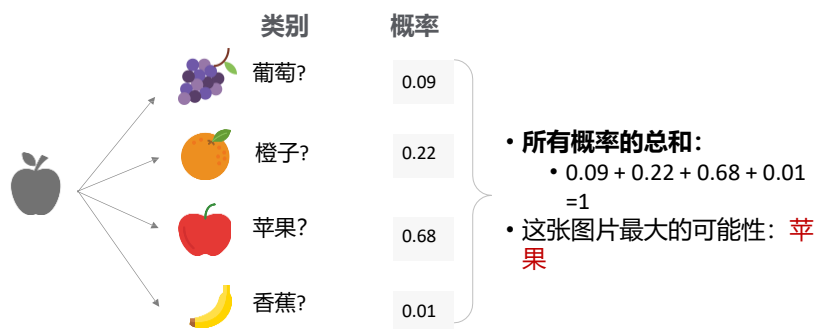
逻辑回归扩展 - Softmax函数 (2)

- Softmax回归是逻辑回归的一般化，适用于K分类的问题。
- Softmax函数的本质就是将一个K维的任意实数向量压缩（映射）成另一个K维的实数向量，其中向量中的每个元素取值都介于（0，1）之间。
- Softmax回归概率函数为：

$$p(y = k | x; w) = \frac{e^{w_k^T x}}{\sum_{l=1}^K e^{w_l^T x}}, k = 1, 2, \dots, K$$

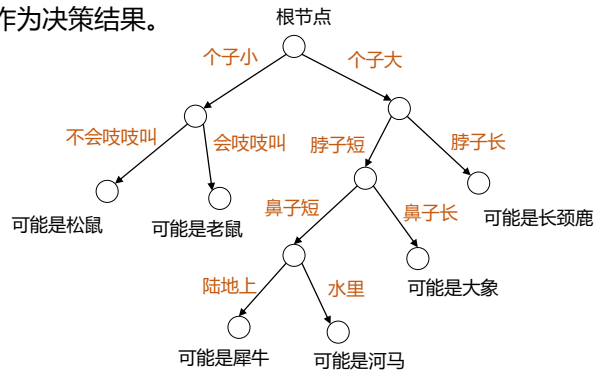
逻辑回归扩展 - Softmax函数 (3)

- Softmax为多类问题中的每个分类分配了概率值。这些概率加起来等于1。
 - Softmax可能产生一个属于特定类别的形式可能以如下方式展示:



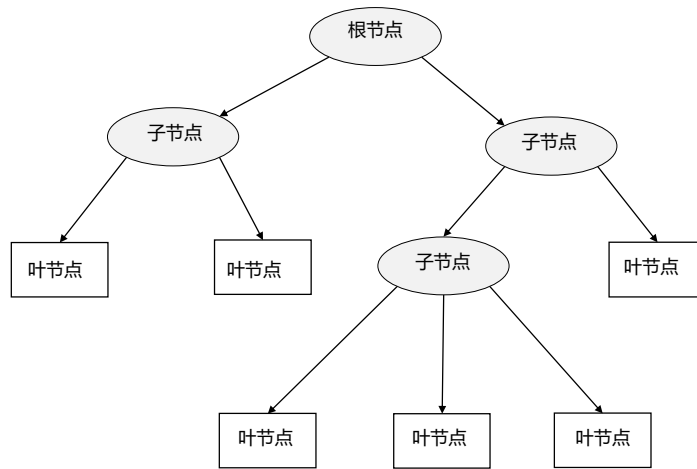
决策树

- 决策树 (decision tree) 是一个树结构。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。



- 决策树最重要的是决策树的构造。所谓决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。构造决策树的关键步骤是分裂属性，即在某个节点处按照某一特征属性的不同划分构造不同的分支。
- 决策树的学习算法用来生成决策树，常用的学习算法为ID3，C4.5，CART。

决策树的结构



- 除了根节点之外的其它节点统称为叶子节点。

决策树的构建关键点

- 决策树的构造就是进行属性的选择，确定各个特征属性之间的树结构。构建决策树的关键步骤就是按照所有的特征属性进行划分操作，对所有的划分操作的结果集的“纯度”进行比较，选择“纯度”最高的属性作为分割数据集的数据点。
- 纯度的量化指标主要通过信息熵与GINI系数，公式如下：

$$H(X) = - \sum_{k=1}^K p_k \log_2(p_k) \quad Gini = 1 - \sum_{k=1}^K p_k^2$$
$$\min_{j,s} [\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

- 其中 p_k 表示样本属于类别k的概率(共有K个类别)，分割前与分割后的纯度差异越大，决策树越好。
- 常见的决策树算法有ID3，C4.5，CART等。

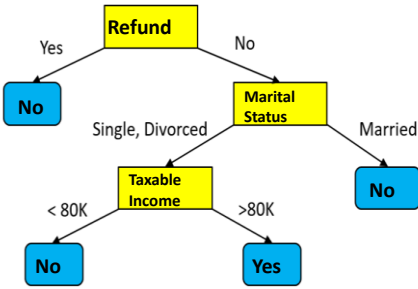
决策树的构建过程

- 特征选择：从训练数据的特征中选择一个特征作为当前节点的分裂标准（特征选择的标准不同产生了不同的特征决策树算法）。
- 决策树生成：根据所选特征评估标准，从上至下递归地生成子节点，直到数据集不可分则停止决策树停止生长。
- 剪枝：决策树容易过拟合，需要剪枝来缩小树的结构和规模（包括预剪枝和后剪枝）。

决策树案例

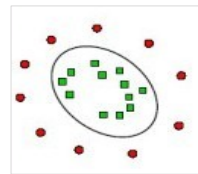
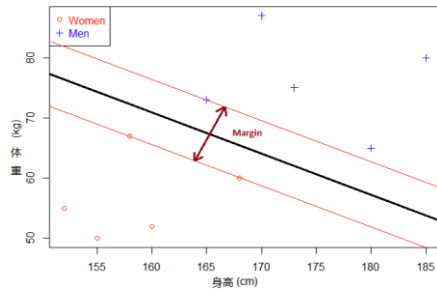
- 下图案例向我们展示了一个决策树进行的分类问题，分类结果受到三个属性的影响分别是Refund, Marital status, Taxable Income。

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divoiced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

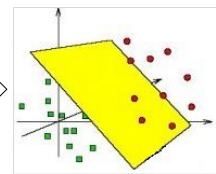
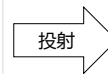


支持向量机

- 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。



低维空间分割复杂

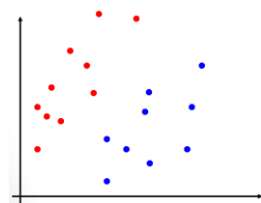


高维空间分割容易

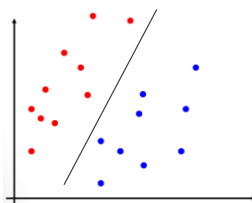
- SVM的主要思想可以概括为两点：
 - 对于线性不可分的情况，通过使用非线性映射算法将低维输入空间线性不可分的样本转化为高维特征空间使其线性可分，从而使得高维特征空间采用线性算法对样本的非线性特征进行线性分析成为可能。
 - 它基于结构风险最小化理论之上特征空间中构建最优超平面，使得学习器得到全局最优化，并且在整个样本空间的期望以某个概率满足一定上界。

线性支持向量机 (1)

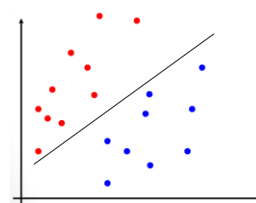
- 如何使用一条直线分割下面的红蓝数据集？



拥有二分类
二维数据集



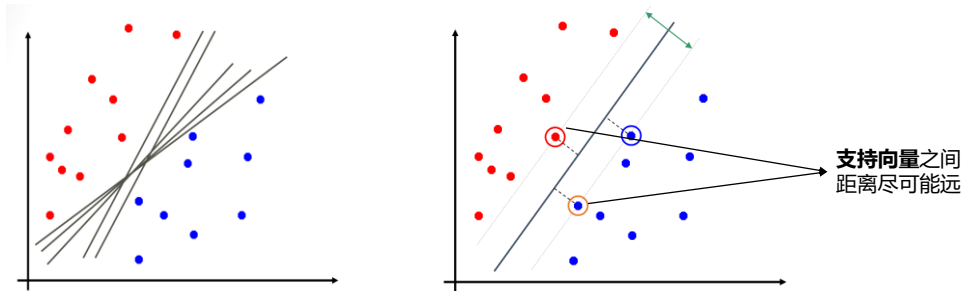
or



左边的分法与右边的分法都可以分割数据集，哪种是正确的呢？

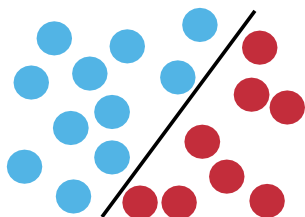
线性支持向量机 (2)

- 直线被用来将数据分割到不同类别中，而实际上我们可以多条直线将数据分开。SVM的核心思路是找到一个直线，使离直线比较近的点，尽可能**远离**这条直线，这可以使模型具有很强的泛化能力。而这些点就被称为**支持向量** (Support Vector) 。
- 在二维空间中，我们用直线来分割，如果是高维空间，我们使用**超平面**来分割。

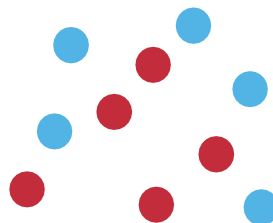


非线性支持向量机 (1)

- 如何分类一种非线性可分的数据集?



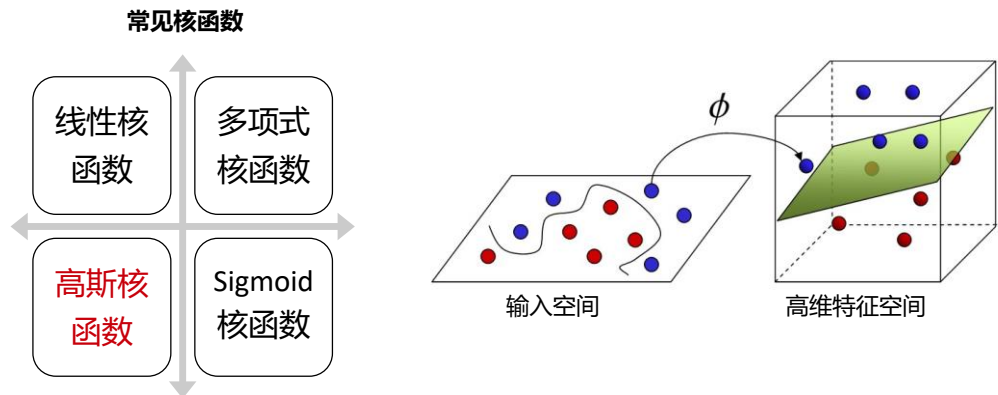
使用线性SVM可以很好的作用在
线性可分数据集上



我们无法用直线分割非线
性数据集

非线性支持向量机 (2)

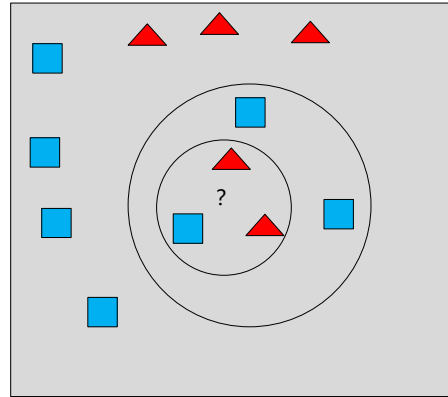
- 使用核函数来构建非线性支持向量机。
- 核函数允许算法在变换后的高维特征空间中拟合最大的超平面。



- 高斯核函数用的最多。

K最邻近算法 (1)

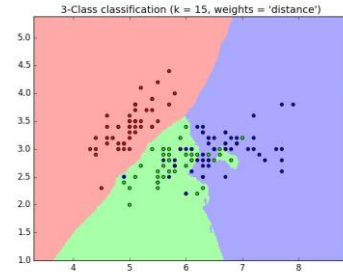
- K最近邻 (k-Nearest Neighbor, KNN) 分类算法，是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。



? 点的目标类别根据选取几个最邻近节点数而不同

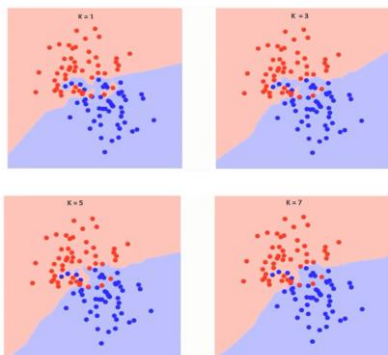
K最邻近算法 (2)

- KNN的核心思想是“近朱者赤，近墨者黑”，其在逻辑上十分的简洁。
- KNN是非参数方法，常用在决策边界非常不规则的数据集中使用。
 - KNN在分类预测时，一般采用多数表决法；KNN在做回归预测时，一般采用平均值法。
- KNN需要非常大的计算量。



K最邻近算法 (3)

- 一般来说，较大的k值会降低噪声对分类的影响，但会使类之间的边界不那么明显。
 - K值越大越容易欠拟合，因为分割过于粗糙，越小越容易过拟合，因为分割过于细腻。



- 可以发现随着K值的增加，边界会变得更平滑。
- 随着K值的增加到无穷大，最终会变成全蓝或全红

朴素贝叶斯 (1)

- 朴素贝叶斯算法 (Naive Bayes)：朴素贝叶斯是一种简单的多类分类算法，基于贝叶斯定理，并假设特征之间是独立的。给定样本特征 X ，样本属于类别 H 的概率是：

$$P(C_k | X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n | C_k) P(C_k)}{P(X_1, \dots, X_n)}$$

- $X_1, X_2, \dots, X_n$ 是数据的特征，通常用 m 个属性集的测量值描述。
 - 比如说颜色特征可能有红，黄和蓝三个属性。
- C_k 表示该数据属于某个特定类 C 。
- $P(C_k | X_1, X_2, \dots, X_n)$ 是后验概率，或在条件 C_k 下， H 的后验概率。
- $P(C_k)$ 是先验概率， $P(C_k)$ 独立于 $X_1, X_2, \dots, X_n$ 。
- $P(X_1, X_2, \dots, X_n)$ 是 X 的先验概率。

- 类条件独立性：贝叶斯分类法假定一个属性值在给定类上的影响独立于其他属性的值。做此假定是为了简化计算，并在此意义下成为“朴素的”。
- 贝叶斯分类器应用于大型数据库具有较高的准确性和快速的速度。

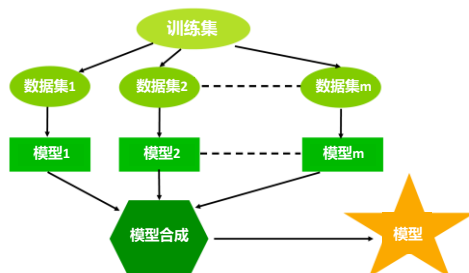
朴素贝叶斯 (2)

- 特征独立假设。
 - 例如，如果一个水果是红色的，圆圆的，直径大约10厘米，就可以被认为是苹果。
 - 一个朴素贝叶斯分类器认为这些特征中的每一个都独立地贡献了这个水果是一个苹果的概率，而不考虑颜色、圆度和直径特征之间的任何可能的相关性。



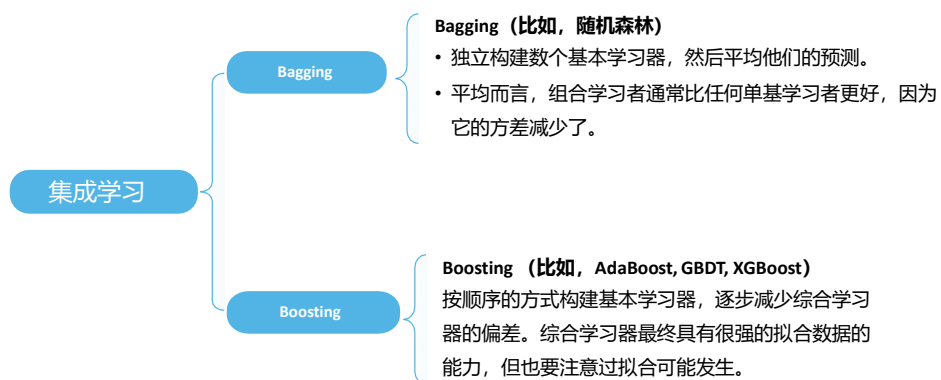
集成学习

- 集成学习是一种机器学习范式，在这种范式中，多个学习者被训练和组合以解决同一个问题。通过使用多个学习者，集成的泛化能力可以比单个学习者强得多。
- 如果随机向数千人提出一个复杂的问题，然后汇总他们的答案。在许多情况下，会发现这个汇总的答案比专家的答案更好。这就是群众智慧。



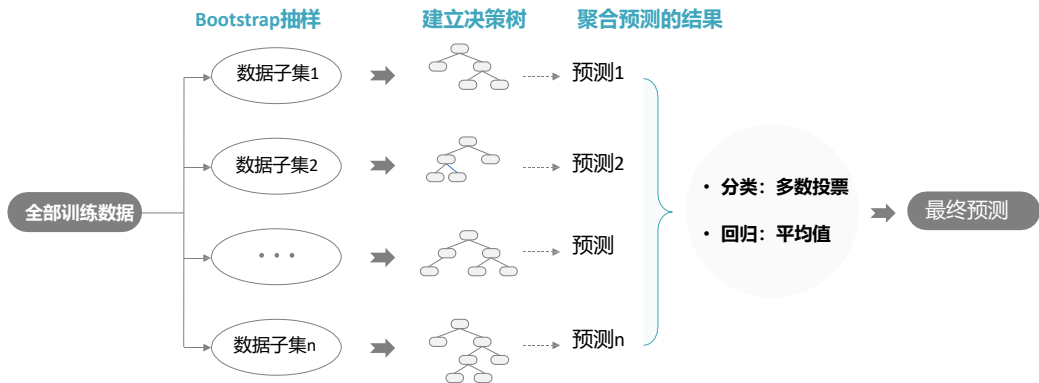
- http://www.datakit.cn/blog/2014/11/02/Ensemble_learning.html

集成学习的分类



集成学习 - 随机森林

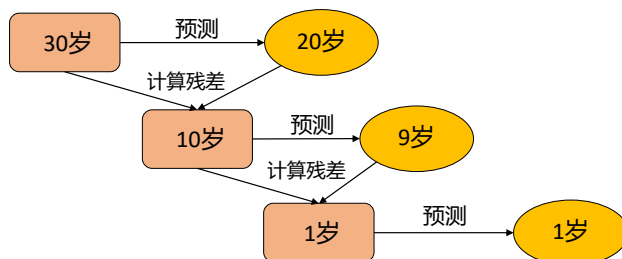
- 随机森林 (Random Forest) = Bagging + CART决策树。
- 随机森林建立多个决策树，并将它们合并在一起，以获得更准确和稳定的预测。



- 随机森林可以用于分类和回归问题。

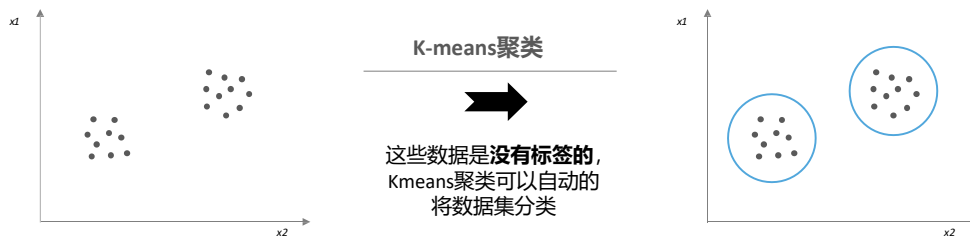
集成学习 - GBDT

- GBDT是Boosting算法的一种。
- 综合模型的结果是所有基础学习器的结果相加等于预测值，其本质是下一个基础学习器去拟合误差函数对预测值的残差（这个残差就是预测值与真实值之间的误差）。
- GBDT在模型训练的时候，是要求模型预测的样本损失尽可能的小。



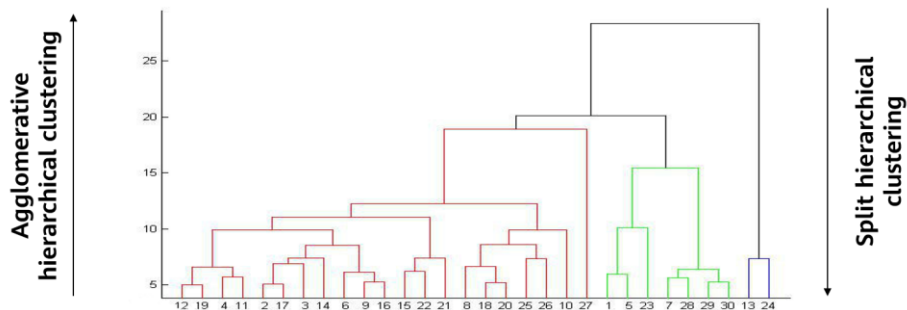
无监督学习 - K-means

- K-means算法是输入聚类个数 k ，以及包含 n 个数据对象的数据集，输出满足方差最小的标准的 k 个聚类的一种算法。
- K-means算法需要输入聚类的最终个数 k ，然后将 n 个数据对象划分为 k 个聚类，而最终所获得的聚类满足：（1）同一聚类中的对象相似度较高；（2）而不同聚类中的对象相似度较小。



无监督学习 - 层次聚类

- 层次聚类法试图在不同层次对数据集进行划分，从而形成树形的聚类结构，数据集的划分可采用“自下向上”的聚合策略，也可以采用“自顶向下”的分拆策略。聚类的层次被表示成树形图。树根拥有所有样本的唯一聚类，叶子是仅有一个样本的聚类。



本章总结

- 本章主要介绍了机器学习的定义、分类与要解决的主要问题，同时对机器学习的整体流程（数据准备，数据清理，特征提取，模型训练与评估，模型部署）、常见的机器学习算法（线性回归，逻辑回归，决策树，SVM，朴素贝叶斯，KNN，集成学习，K-means等），超参数等机器学习重要的知识点进行了梳理与学习。

思考题

1. (单选) 以下算法中不属于有监督学习的是? ()
 - A. 线性回归
 - B. 决策树
 - C. KNN
 - D. K-means
2. (判断题) 梯度下降迭代法是机器学习算法的唯一方法。 ()
 - A. True
 - B. False

- 答案: 1.D 2.B

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>

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



深度学习和大模型基础



前言

- 在当今数字化时代，数据量的爆炸式增长以及计算能力的显著提升，为人工智能领域带来了前所未有的发展机遇。深度学习是其中一个重要的分支。
- 随着研究的深入和技术的进步，深度学习从浅层架构向深层架构的转变，即“大模型”的兴起。它是极其庞大、结构更加复杂的深度神经网络。这类模型由于其卓越的表现力，在图像生成、语音识别、文本生成等方面取得了超越以往多数方法的成绩。
- 本章将从基础概念出发，逐步深入探讨神经网络与大模型领域的基础知识。

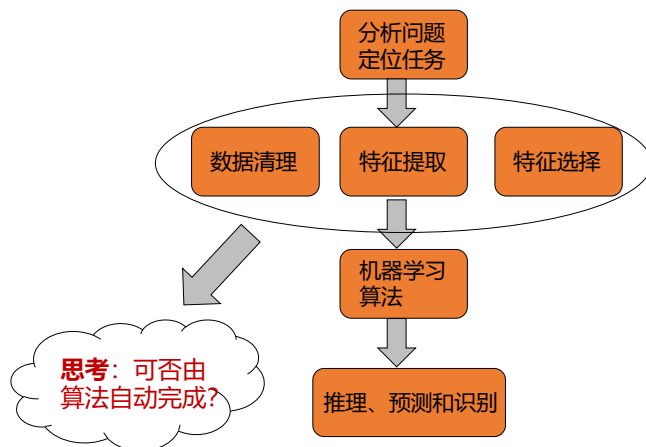
目标

- 学完本课程后，您将能够：
- 掌握深度学习基础
- 掌握卷积神经网络基础架构
- 掌握循环神经网络基础架构
- 了解Transformer架构
- 了解常见的大模型架构

目录

1. **感知机**
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

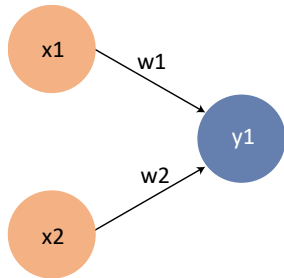
传统机器学习



- 分析问题定位任务:首先定位问题，将实际问题转化为机器学习问题，需要什么样的数据，然后进行数据的采集。包含数据探查、场景抽象、算法选择。比如逻辑回归可以决定我们是不是能喝咖啡；那另外朴素贝叶斯算法可以用来区分垃圾电子邮件和合法电子邮件。AI系统不是直接检测，而是医生会提供给系统一些数据，比如说人体对咖啡因的耐受度，会不会睡不着
- 数据清理：数据清洗——重新检查和验证数据的过程，旨在删除重复信息，纠正现有错误并提供数据一致性。采用（随即、系统、分层）、归一化、去除噪声、数据过滤
- 特征提取：特征抽象、特征重要性评估、特征衍生、特征降维。
- 特征选择
- 模型训练
- 推理，预测和识别
- 一般来说，机器学习中特征是由人工选定，而特征越多，给出的信息就越多，识别准确性会得到提升
- 但特征多，计算复杂度增加，搜索空间就打，训练数据会在全体特征向量中就会显得稀疏，影响相似性判断，即**维度爆炸**。

感知机是什么

- 感知机接收多个输入信号，输出一个信号。



- 左图一个接收两个输入信号的感知机的例子。 x_1 、 x_2 是输入信号， y 是输出信号， w_1 、 w_2 是权重（ w 是weight的首字母）。
- 图中的 $\circ$ 称为“神经元”或者“节点”。输入信号被送往神经元时，会被分别乘以固定的权（ w_1x_1 、 w_2x_2 ）。神经元会计算传送过来的信号的总和，只有当这个总和超过了某个界限值时，才会输出1。这也称为“神经元被激活”。

- 1957年,就职于康奈尔大学航空实验室的 Frank Rosenblatt,发明了感知器 人工神经网络。该网络被视为一种最简单形式的前馈神经网络模型,是一种二元线性分类器,其激活函数采用了符号函数 $\text{sign}(x)$ 。感知器是人工神经网络的第一个实际应用,标志着神经网络进入了新的发展阶段。Frank Rosenblatt通过软件实现感知器后,开始构建硬件版本的感知器 Mark I。他将400个光电设备作为神经元,将可调电位器作为突触权重,并且在学习期间权重的变化由电动机调整电位器实现,构造了硬件感知器,并用于图像识别。由于当时的技术限制,基于物理实现的感知器是比较罕见的。这种用于图像识别的系统受到了各方的关注,并因此从美国海军获得了大量资金。
- 这里所说的“信号”可以想象成电流或河流那样具备“流动性”的东西。像电流流过导线,向远方输送电子一样,感知机的信号也会形成流,向远方输送信息。但是,和实际的电流不同的是,感知机的信号只有“流/不流”（1/0）两种取值。在本书中,0对应“不传递信号”,1对应“传递信号”。

感知机的数学表示

- 感知机的多个输入信号都有各自固有的权重，这些权重决定各个信号的重要性。即权重越大，对应该权重的信号的重要性就越高。

$$y = \begin{cases} -1 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

权重相当于电流里所说的电阻。电阻是决定电流流动难度的参数，电阻越低，通过的电流就越大。而感知机的权重则是值越大，通过的信号就越大。不管是电阻还是权重，在控制信号流动难度（或者流动容易度）这一点上的作用都是一样的。

使用感知机模拟简单逻辑电路

- 逻辑电路的几种形式



与门

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

$$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$$



与非门

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

$$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$$



或门

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

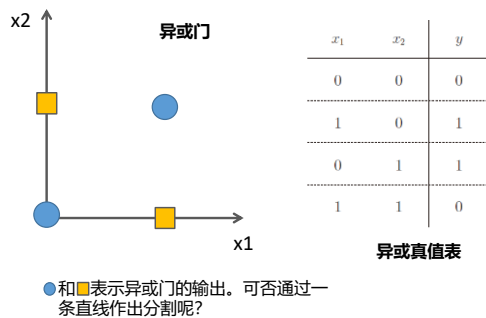
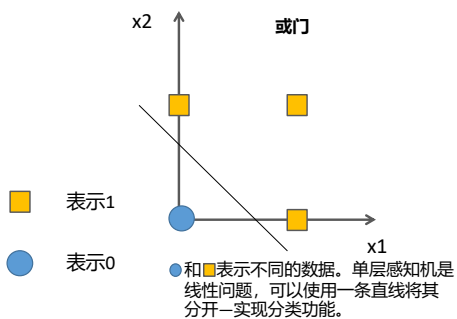
$$(w_1, w_2, \theta) = ?$$

- 在逻辑电路中我们将输出-1改为
- 还有哪些值组合满足不同的门
- 与门: (0.5, 0.5, 0.8) 或者(1.0, 1.0, 1.0)

如何用单层感知机解决异或问题？

- 将或门真值表在二维坐标系中表达出来，不难发现，单层感知机以线性的方式将两类数值分开。
- 异或门也被称为逻辑异或电路。仅当 x_1 或 x_2 中的一方为1时，才会输出1。

$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases}$$



- 两条线可以区分开，引出两层。

如何使用与门、与非门、或门组合实现异或门？

- 如果一层感知机无法解决异或问题，那是否可以组合已经实现的逻辑电路门来实现异或门呢？



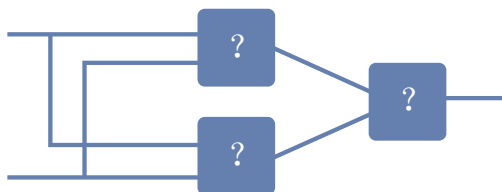
与门



与非门

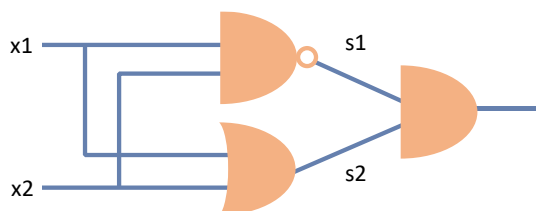


或门



将与门、与非门、或门代入到“？”中，就可以实现异或门！

异或门的实现 – 两层感知机



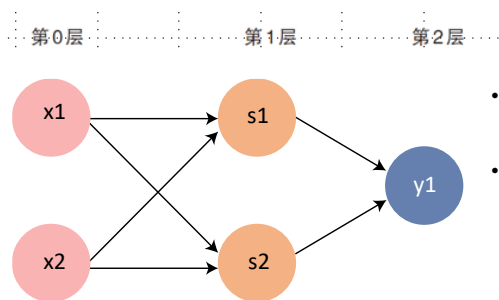
x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

通过组合与门、与非门、或门实现异或门

异或门实现过程的真值表

多层感知机

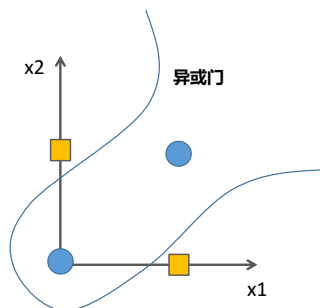
- 将前面的逻辑门表示转换为感知机形式表示：



- 1. 第0 层的两个神经元接收输入信号，并将信号发送至第1 层的神经元。
- 2. 第1 层的神经元将信号发送至第2 层的神经元，第2 层的神经元输出 y 。

非线性

- 解决异或问题，需要一条非线性的曲线将0、1分开，以达到模拟异或门的效果。

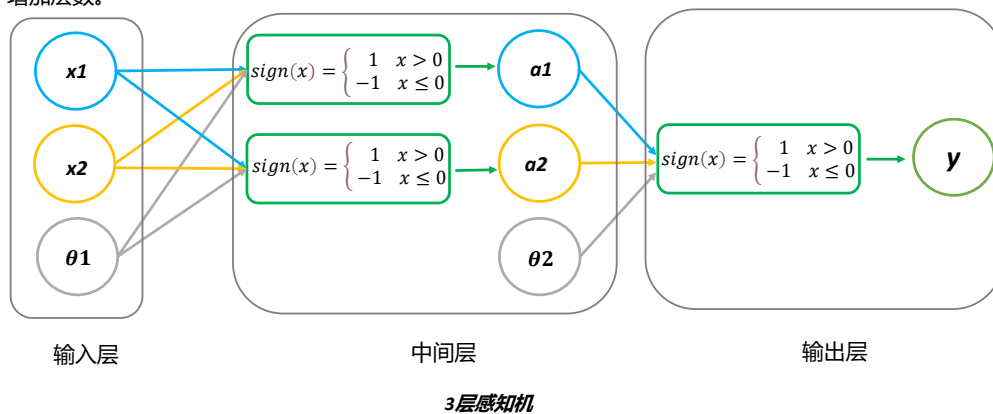


- 思考一个问题：多层感知机是如何将感知机的功能从线性转变为非线性的呢？

多层感知机非线性能力的来源 – 增加层数

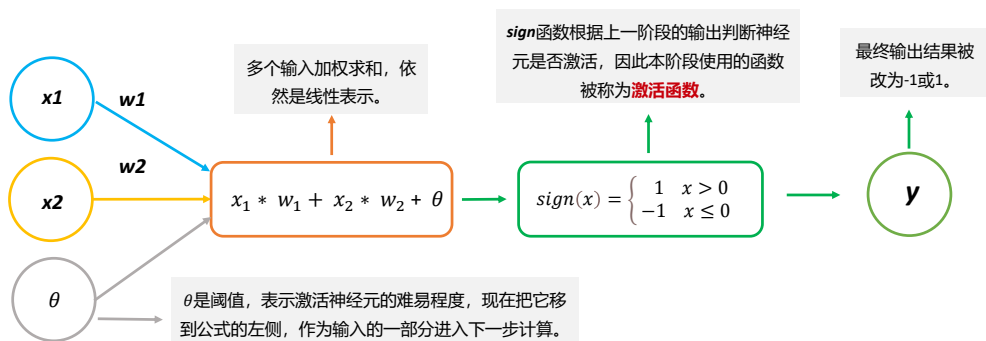
- 多层感知机能解决较为复杂的问题，主要是因为两点：

- 增加层数。



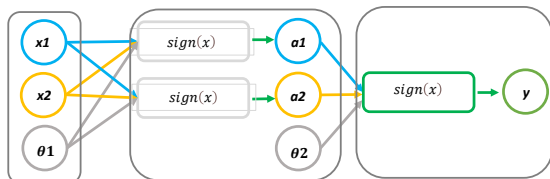
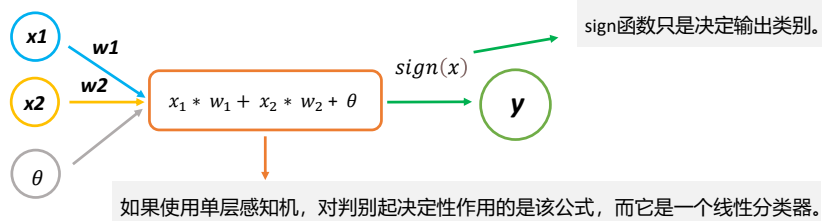
多层感知机非线性能力的来源 – 非线性激活函数

- 使用了非线性的函数来处理输出结果。



单层感知机计算过程

多层感知机和非线性激活函数的必要性



如果在中间层不使用sign函数，那么：

$$a_1 = x_1 * w_1 + x_2 * w_2 + \theta_1 - \text{线性}$$

$$a_2 = x_1 * w_3 + x_2 * w_4 + \theta_1 - \text{线性}$$

最终决定输出的值为：

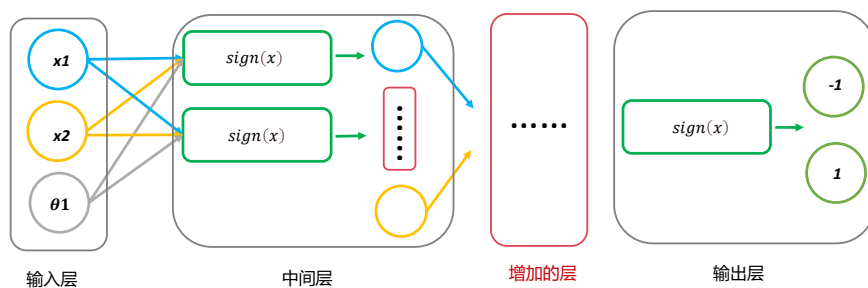
$$\begin{aligned} a_1 * w_5 + a_2 * w_6 + \theta_2 = & \\ & w_5(x_1 * w_1 + x_2 * w_2 + \theta_1) \\ & + w_6(x_1 * w_3 + x_2 * w_4 + \theta_1) + \theta_2 \end{aligned}$$

目录

1. 感知机
2. **全连接神经网络及训练流程**
 - 深度神经网络
 - 常见的激活函数
 - 训练神经网络
 - 优化器
 - 正则化
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

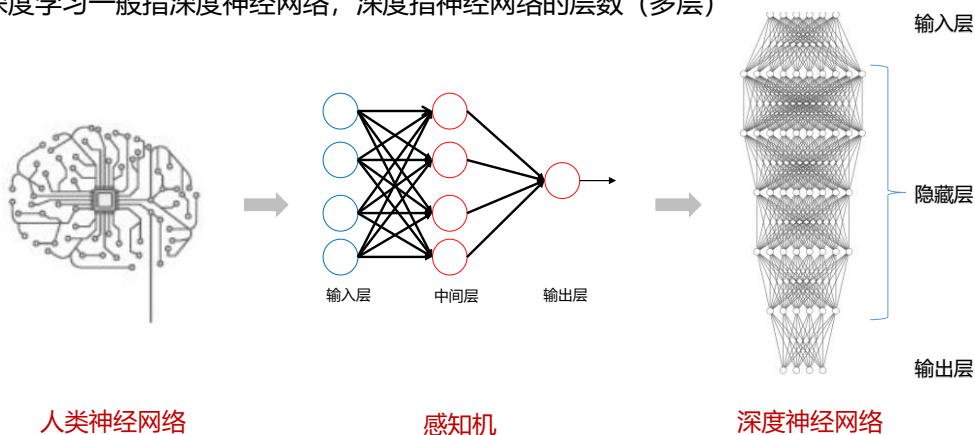
感知机的能力提升

- 多层感知机已经表现出了它能解决复杂问题（拟合复杂函数）的能力，但是它中间层只有两个单元，并且它只能解决二分类问题。
- 那继续增加中间层的节点数和网络结构层数，是否能进一步提升能力呢？



深度神经网络

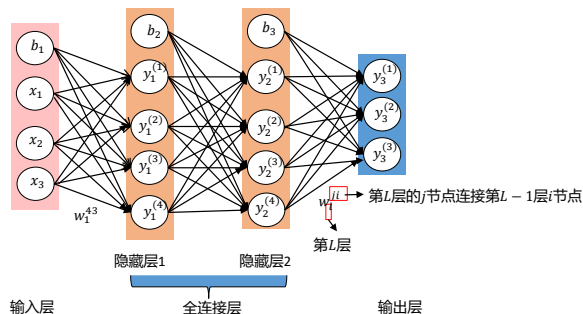
- 深度学习一般指深度神经网络，深度指神经网络的层数（多层）



- 何为深度学习呢？在前面的课程我们已经了解，深度学习是机器学习的一个子集。
- 深度学习一般指深度神经网络，深度指神经网络的层数（多层）。
- 这是模拟人类的神经网络而构建的
- 在人工神经网络设计及应用研究中，通常需要考虑三个方面的内容，即神经元作用函数、神经元之间的连接形式和网络的学习(训练)。

前馈神经网络结构

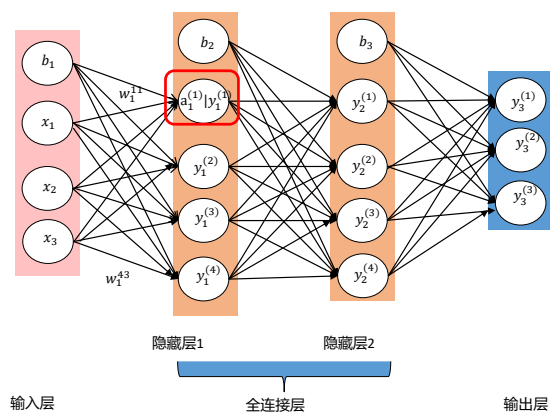
- 前馈神经网络（Feedforward Neural Network）是典型的深度学习模型。具有以下特点：
 - 第一层为输入层，是进入网络的初始数据源。
 - 隐藏层及输出层中的每个神经元都接收前一层全部神经元的加权和，经过激活函数后输出给下一层。
 - 采用一种单向多层结构；整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



- x 表示输入数据，
- w 表示权重
- b 是激活神经元的阈值，在前馈神经网络里称为偏置。
- y 是神经元的输出，它表示了输入经过激活函数的输出结果。

- 也叫前馈神经网络

前馈神经网络的计算过程



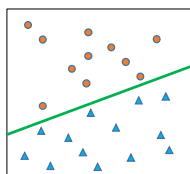
左图红框内将隐藏层1的第1个神经元展开，单个神经元的计算过程如下：

$$a_1^{(1)} = \sum_1^m (x_i * w_1^{1i}) + b_1$$

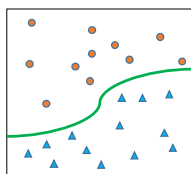
$$y_1^{(1)} = f(a_1^{(1)})$$

.....

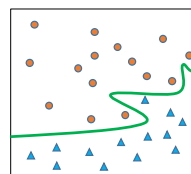
隐藏层数对神经网络的影响



隐藏层数=0



隐藏层数=3



隐藏层数=20

- 可以发现，隐藏层越多，神经网络的分辨能力越强。

增加单层节点数

- 通过单层感知机到多层感知机的变化发现，网络不仅可以增加层数，还可以增加单层上网络节点的数量。
- 那么，如果只增加单层网络中的节点数量，是否可以增强网络能力呢？

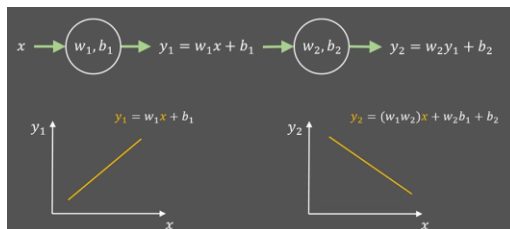


目录

1. 感知机
2. **全连接神经网络及训练流程**
 - 深度神经网络
 - 常见的激活函数
 - 训练神经网络
 - 优化器
 - 正则化
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

深度神经网络依然需要非线性激活函数

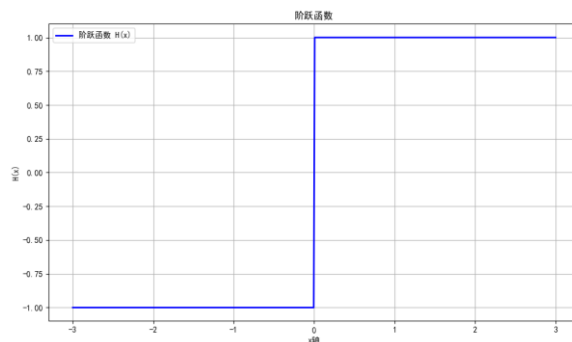
- 不仅是多层感知机，深度神经网络如果不使用非线性激活函数，依然不能产生非线性能力。
- 使用线性激活函数，例如： $f(x) = wx + b$



- 那常见的激活函数有哪些呢？

阶跃函数

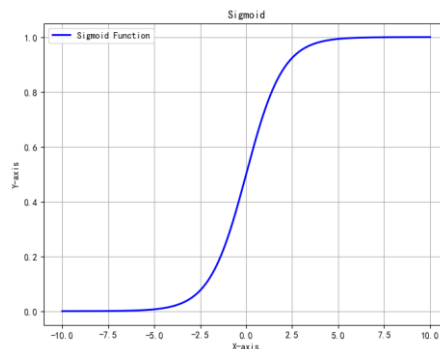
- 神经网络中的每个神经元节点接受上一层神经元的加权和作为本神经元的输入值，输入值经过一种非线性函数后，作为输入进入下一层，该非线性函数被称为激活函数。
- 在多层感知机中使用的sign函数作为激活函数，它是一种阶跃函数，其函数及其导数图形如下：



Sigmoid函数

- sigmoid函数是神经网络中经常使用的一个激活函数。
- 对于一个定义域在R中的输入，sigmoid函数将输入变换为区间(0, 1)上的输出。因此，sigmoid通常称为挤压函数：它将范围 $(-\infty, +\infty)$ 中的任意输入压缩到区间 $(0, 1)$ 中的某个值：

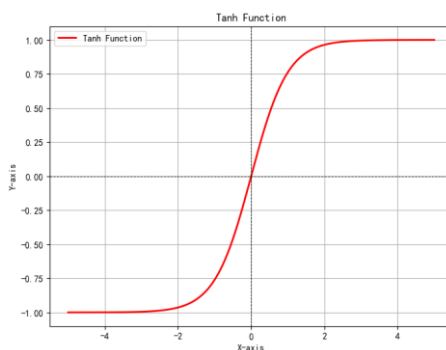
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



Tanh函数

- 与sigmoid函数类似，tanh(双曲正切)函数也能将其输入压缩转换到区间(-1, 1)上。tanh函数的公式如下：

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

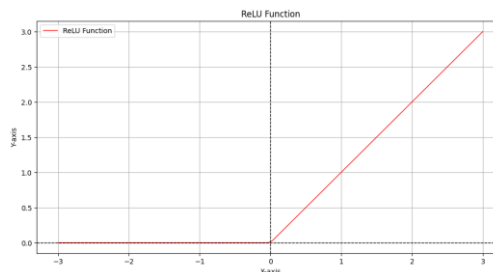


- 当输入在0附近时，tanh函数接近线性变换。函数的形状类似于sigmoid函数，不同的是tanh函数关于坐标系原点中心对称。

ReLU函数

- 修正线性单元 (Rectified linear unit, *ReLU*) 是最受欢迎的激活函数之一，因为它实现简单，同时在各种预测任务中表现良好。
- ReLU提供了一种非常简单的非线性变换。给定元素 x ，*ReLU*函数被定义为该元素与0的最大值：

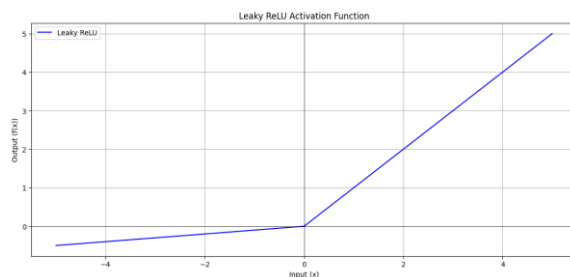
$$\text{ReLU}(x) = \begin{cases} \max(0, x) & , x \geq 0 \\ 0 & , x < 0 \end{cases}$$



Leaky ReLU函数

- Leaky ReLU函数是基于ReLU函数的改进版，它给所有负值赋予一个非零斜率。

$$\text{LeakyReLU}(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$

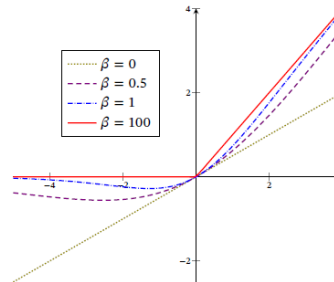


Swish函数

- Swish函数是一种自门控（Self-Gated）激活函数，定义为：

$$\text{swish}(x) = x\sigma(\beta x)$$

- 其中 $\sigma(\cdot)$ 为Sigmoid函数， β 为可学习的参数或一个固定超参数。 $\sigma() \in (0, 1)$ 可以看作一种软性的门控机制。当 $\sigma(\beta x)$ 接近于1时，门处于“开”状态，激活函数的输出近似于 x 本身；当 $\sigma(\beta x)$ 接近于0时，门的状态为“关”，激活函数的输出近似于0。

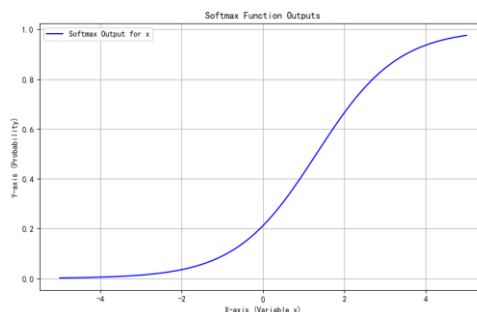


- 当 $\beta = 0$ 时，Swish 函数变成线性函数 $x/2$ 。当 $\beta = 1$ 时，Swish 函数在 $x > 0$ 时近似线性，在 $x < 0$ 时近似饱和，同时具有一定的非单调性。当 $\beta \rightarrow +\infty$ 时， $\sigma(\beta x)$ 趋向于离散的0-1函数，Swish 函数近似为ReLU 函数。因此，Swish 函数可以看作线性函数和ReLU 函数之间的非线性插值函数，其程度由参数 β 控制。

Softmax函数

- Softmax函数通常用于深度学习中的多分类问题。
- 它将一个含任意实数的K维向量转换为另一个在(0,1)范围内的K维向量，并且所有元素之和等于1。因此，Softmax函数输出的向量可以被解释为概率分布，这使得它非常适合用于表示分类模型中不同类别的概率。

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

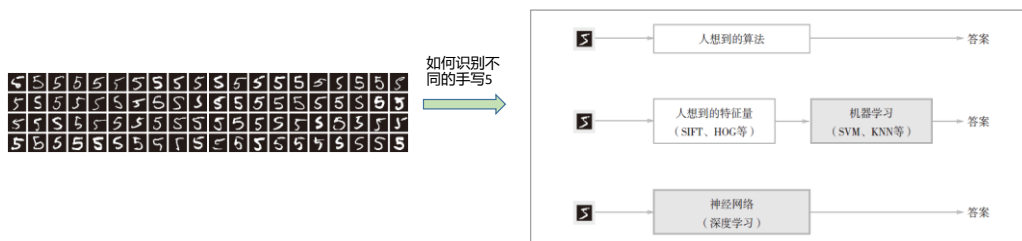


目录

1. 感知机
2. **全连接神经网络及训练流程**
 - 深度神经网络
 - 常见的激活函数
 - 训练神经网络
 - 优化器
 - 正则化
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

从数据中学习

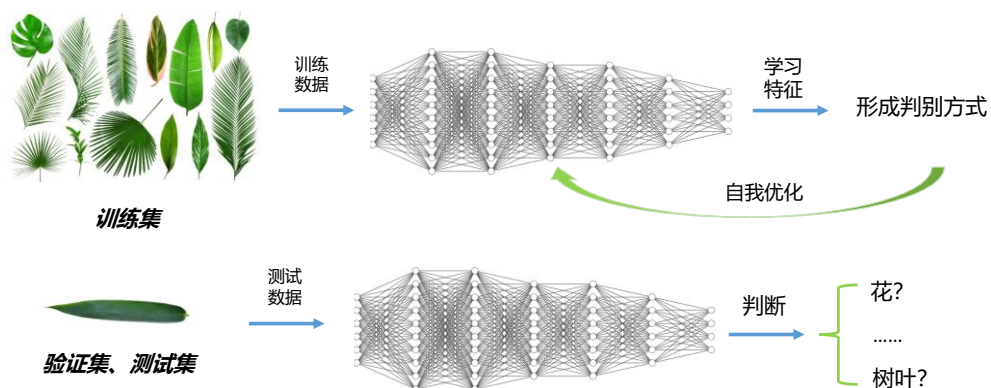
- 神经网络与多层感知机的最大的区别就是，在解决非线性问题时，它可以自动学习如何提取特征，从而形成解决问题的模型。而这种学习的根基，是数据！



- 数据是机器学习的命根子。从数据中寻找答案、从数据中发现模式、根据数据讲故事。这些机器学习所做的事情，如果没有数据的话，就无从谈起。因此，数据是机器学习的核心。这种数据驱动的方法，也可以说脱离了过往以人为中心的方法。

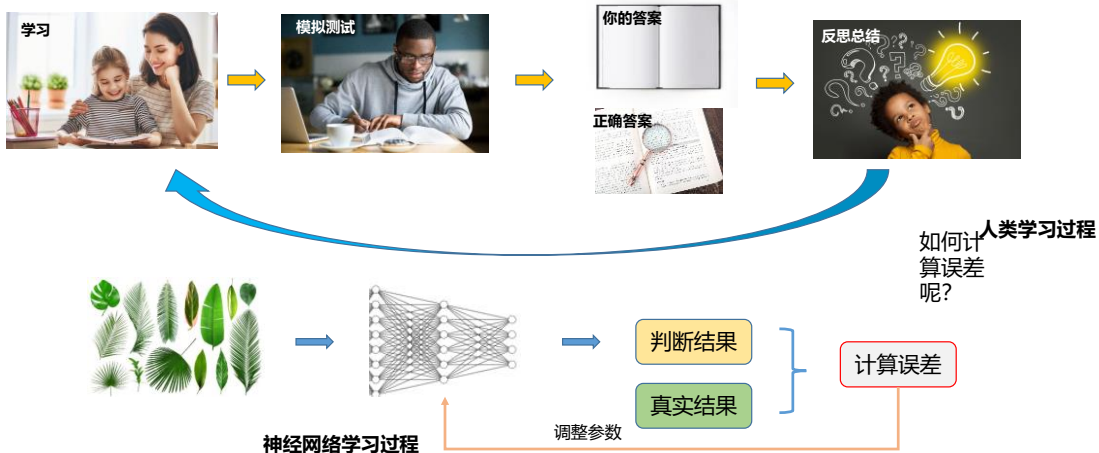
数据的划分

- 数据集一般分为训练集、验证集和测试集。



验证集一般用来阶段性测试模型训练效果，根据验证结果调整模型及训练方案。
测试集一般是模型正式工作过程中的模型从未见过的数据，模型在它上面的表现直接展示了其性能。

神经网络如何学习？



损失函数

- 损失函数 (Loss Function) 是机器学习和深度学习中用于衡量模型预测值与真实值之间差异的函数, 它量化了模型在训练过程中所犯错误的程度。
- 常用的损失函数主要有以下两种:

- 均方误差: 主要用于回归任务。

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

y_k 是表示神经网络的输出, t_k 表示监督数据, k 表示数据的维数。

- 交叉熵损失函数: 主要用于分类任务。

$$E = - \sum_k t_k \log y_k$$

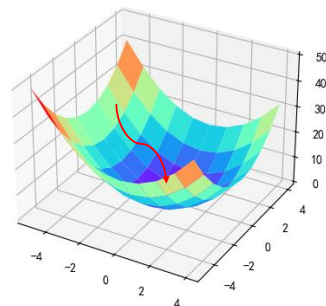
y_k 是神经网络的输出, t_k 是正确解标签。

梯度下降

- 神经网络学习的过程是在寻找最优参数的过程。随着参数的优化，神经网络的预测值会逐渐接近真实值，进而损失函数也会减小。因此，最优化参数即使损失函数取得最小值。
- 梯度是多元函数值增长最快的方向，因此需要所有参数向着负梯度方向优化。

对于多元函数 $C(W) = f(w_0, w_1, \dots, w_n)$ ，其在 $W' = [w_0', w_1', \dots, w_n']^T$ 处的梯度为：

$$\nabla f(w_0', w_1', \dots, w_n') = \left[\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \dots, \frac{\partial f}{\partial w_n} \right]^T \Big|_{W=W'},$$



随机梯度下降算法 (SGD)

- 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：

$$\Delta w_i = -\eta \frac{1}{n} \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i} \Rightarrow \Delta w_i = -\eta \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$$

- SGD:
 - 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:
 - 在训练集中随机出一个 $\langle x, t \rangle$ ，do:
 - 将 x 输入此单元，计算输出 o
 - 对于此单元的每个 w_i ，do: $w_i += -\eta \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$

- 这个版本的梯度下降算法走向了另一个极端，即每个样例都更新权值。因为训练样例中一般是含有噪声的，因此到了精确逼近极值的阶段，梯度往往会在极值附近横冲直撞，难以收敛到极值。

全局梯度下降算法 (BGD)

- 对于训练样例集中的每一个样例记为 $\langle X, t \rangle$, X 是输入值向量, t 为目标输出, o 为实际输出, η 是学习率, C 为损失函数。

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
- 对于训练集中每个 $\langle X, t \rangle$, do:
 - 将 X 输入此单元, 计算输出 o
 - 对于此单元的每个 w_i , do: $\Delta w_i += -\eta \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$
- 对于此单元的每个 w_i , do: $w_i += \Delta w_i$

- 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。

- 全量梯度下降法每次学习都使用整个训练集, 因此每次更新都会朝着正确的方向进行, 最后能够保证收敛于极值点, 凸函数收敛于全局极值点, 非凸函数可能会收敛于局部极值点, 缺陷就是学习时间太长, 消耗大量内存。

小批量梯度下降 (MBGD)

- 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。

初始化每个 w_i 为绝对值较小的随机值

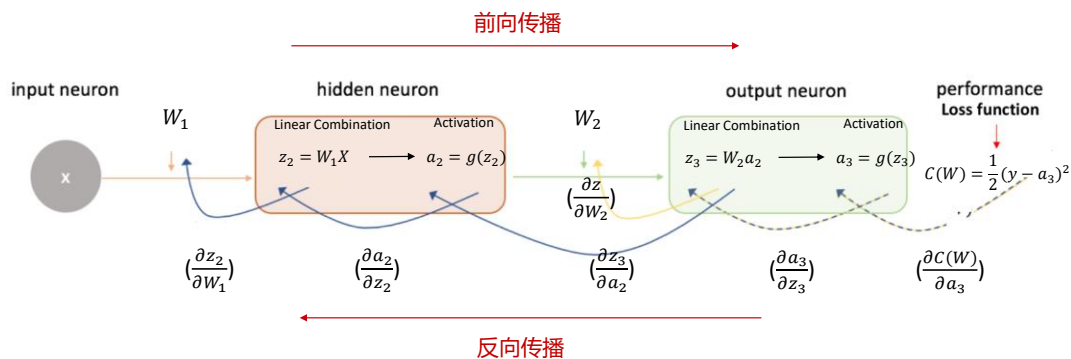
遇到终止条件前，do：

- 初始化每个 Δw_i 为零
- 从训练数据集中下一批（BS个）样例，对这批样例中的每一个 $\langle X, t \rangle$ ，do：
 - 将 X 输入此单元，计算输出 o
 - 对于此单元的每个 w_i ，do： $\Delta w_i += -\eta \frac{1}{n} \sum_x \sum_{d \in D} \frac{\partial C(t_d, o_d)}{\partial w_i}$
- 对于此单元的每个 w_i ，do： $w_i += \Delta w_i$
- 如果已经是最后一批，打乱训练样例的顺序。

- 这个版本的梯度下降兼顾了效率和梯度的稳定性，容易冲过局部极小值，是实际工作中最常用的梯度下降算法，BS的取值因具体问题而异，一般取32。

网络训练的流程

- 前向传播: $C(W) = \frac{1}{2}(y - a_3)^2 = \frac{1}{2}(y - (g(W_2 g(W_1 X))))^2$.
- 如果需要更新参数 W_1 , 根据 $W_{t+1} = W_t - \eta \frac{dC}{dW}$, 需要计算: $\frac{dC}{dW_1} = \frac{\partial C(W)}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial W_1}$.



- 如图所示, 反向传播的计算顺序是, 将信号 $loss$ 乘以节点的局部导数然后将结果传递给下一个节点。

链式法则

- 复合函数

$$z = (x + y)^2 \xrightarrow{\text{分解}} \begin{matrix} z = t^2 \\ t = x + y \end{matrix}$$

- 对复合函数求导：如果某个函数由复合函数表示，则该复合函数的导数可以用构成复合函数的各个函数的导数的乘积表示。

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \frac{\partial t}{\partial x} \quad \longrightarrow \quad \frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \cancel{\frac{\partial t}{\partial x}}$$

$\frac{\partial z}{\partial x}$ 的导数 (z关于x的导数) 可以用 $\frac{\partial z}{\partial t}$ (z关于t的导数) 乘以 $\frac{\partial t}{\partial x}$ (t关于x的导数) 来表示。

- 以上即是链式法则。

- 神经网络是逐层向前传播的，那如何根据损失函数的梯度计算出全部参数的梯度呢？
- 如果某个函数由复合函数表示，则该复合函数的导数可以用构成复合函数的各个函数的导数的乘积表示。

梯度消失和梯度爆炸问题（1）

- 梯度消失：当网络层数越多时，进行反向传播求导值越小，导致梯度消失。
- 梯度爆炸：当网络层数越多时，进行反向传播求导值越大，导致梯度爆炸。
- 导致原因：

$y_i = \sigma(z_i) = \sigma(w_i x_i + b_i)$ ，其中 σ 为sigmoid函数。

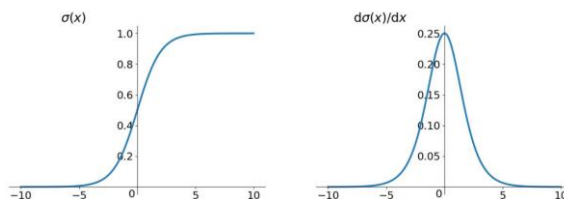


- 反向传播可以推导：

$$\begin{aligned}\frac{\partial C}{\partial W_1} &= \frac{\partial C}{\partial y_4} \frac{\partial y_4}{\partial z_4} \frac{\partial z_4}{\partial x_4} \frac{\partial x_4}{\partial z_3} \frac{\partial z_3}{\partial x_3} \frac{\partial x_3}{\partial z_2} \frac{\partial z_2}{\partial x_2} \frac{\partial x_2}{\partial z_1} \frac{\partial z_1}{\partial x_1} \\ &= \frac{\partial C}{\partial y_4} \sigma'(z_4) w_4 \sigma'(z_3) w_3 \sigma'(z_2) w_2 \sigma'(z_1) x\end{aligned}$$

梯度消失和梯度爆炸问题（2）

- $\sigma'(x)$ 的最大值为 $\frac{1}{4}$:



- 而网络权值 $|w|$ 通常都小于1, 因此 $|\sigma'(z)w| \leq \frac{1}{4}$, 因此对于上面的链式求导, 层数越多, 求导结果 $\frac{\partial C}{\partial b_1}$ 越小, 因而导致梯度消失的情况出现。
- 当网络权值 $|w|$ 比较大时, 导致 $|\sigma'(z)w| > 1$, 出现梯度爆炸。
- 解决方法: 梯度裁剪用于解决梯度爆炸, ReLU激活函数、LSTM神经网络等用于缓解梯度消失。

- 也就是比较大的情况。但对于使用sigmoid激活函数来说, 这种情况比较少。因为 $\sigma'(z)$ 的大小也与 w 有关 (公式), 除非该层的输入值 (公式) 在一直一个比较小的范围内。
- 其实梯度爆炸和梯度消失问题都是因为网络太深, 网络权值更新不稳定造成的, 本质上是因为梯度反向传播中的连乘效应。
- **梯度剪切**这个方案主要是针对梯度爆炸提出的, 其思想是设置一个梯度剪切阈值, 然后更新梯度的时候, 如果梯度超过这个阈值, 那么就将其强制限制在这个范围之内。这可以防止梯度爆炸。
- 另外一种解决梯度爆炸的手段是采用权重正则化 (weights regularization) 比较常见的是l1正则, 和l2正则, 在各个深度框架中都有相应的API可以使用正则化, 比如在tensorflow中, 若搭建网络的时候已经设置了正则化参数, 则调用以下代码可以直接计算出正则损失。
- ****Relu:****思想也很简单, 如果激活函数的导数为1, 那么就不存在梯度消失爆炸的问题了, 每层的网络都可以得到相同的更新速度, relu就这样应运而生。
- LSTM全称是长短期记忆网络 (long-short term memory networks), 是不那么容易发生梯度消失的, 主要原因在于LSTM内部复杂的“门” (gates), 如下图, LSTM通过它内部的“门”可以接下来更新的时候“记住”前几次训练的“残留记忆”, 因此, 经常用于生成文本中。目前也有基于CNN的LSTM, 感兴趣的可以尝试一下。

目录

1. 感知机
2. **全连接神经网络及训练流程**
 - 深度神经网络
 - 常见的激活函数
 - 训练神经网络
 - **优化器**
 - 正则化
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

加速训练

- 训练神经网络时，如果按部就班的训练，模型学习的速度较慢，因此，需要使用一些特殊的方法来加速训练。
- 这些加速的“工具”被称为优化器。



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，主要是学习率（Learning Rate）。
- 常见的优化器如：普通SGD优化器、动量优化器、Adagrad、RMSprop、Adam等。

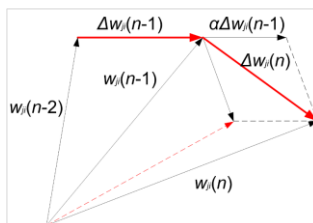
动量优化器

- 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：

$$\Delta w_{ji}^l(n) = -\eta \delta_i^{l+1} x_j^l(n) + \alpha \Delta w_{ji}^l(n-1)$$

其中， $0 \leq \alpha < 1$ 是一个常数，称为动量系数（Momentum Coefficient）。 $\alpha \Delta w_{ji}^l(n-1)$ 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：



- $\eta \delta_i^{l+1} x_j^l(n)$ 为原梯度大小和方向， j 表示第 l 层的第 j 个神经元， i 表示第 $l+1$ 层的第 i 个神经元

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。

Adagrad优化器 (1)

- 随机梯度下降算法 (SGD)、小批量梯度下降算法 (MBGD)、动量优化器的共同特点是：对于每一个参数都用**相同的学习率**进行更新。
- Adagrad的思想则是应该为不同的参数设置不同的学习率。

$$g_t = \frac{\partial C(t, o)}{\partial w_t} \quad \text{计算梯度}$$

$$r_t = r_{t-1} + g_t^2 \quad \text{累积平方梯度}$$

$$\Delta w_t = -\frac{\eta}{\varepsilon + \sqrt{r_t}} g_t \quad \text{计算更新}$$

$$w_{t+1} = w_t + \Delta w_t \quad \text{应用更新}$$

g_t 为第 t 次的梯度, r_t 为梯度累积变量, r 的初始值为0, **会一直递增**。 η 为全局学习率, 需要手动设置。 ε 为小常数, 为了数值稳定大约设置为 10^{-7} 。

Adagrad优化器（2）

- 从Adagrad优化算法中可以看出，随着算法不断迭代， r 会越来越大，整体的学习率会越来越小。这样做的原因是随着更新次数的增大，我们希望**学习率越来越慢**。因为我们认为在学习率的最初阶段，我们距离损失函数最优解还很远，随着更新次数的增加，越来越接近最优解，所以学习率也随之变慢。
- 优点：
 - 学习率自动更新，随着更新次数增加，学习率随之变慢。
- 缺点：
 - 分母会不断累积，最终学习率变得非常小，算法会失去效用。

RMSprop优化器

- RMSprop优化器是一种改进的Adagrad优化器，通过引入一个衰减系数，让 r 每回合都**衰减一定的比例**。
- RMSprop优化器很好的解决了Adagrad优化器过早结束的问题，很合适处理非平稳目标，对于RNN网络效果很好。

$$g_t = \frac{\partial C(t, o)}{\partial w_t}$$

计算梯度

$$r_t = \beta r_{t-1} + (1 - \beta) g_t^2$$

累积平方梯度

$$\Delta w_t = - \frac{\eta}{\varepsilon + \sqrt{r_t}} g_t$$

计算更新

$$w_{t+1} = w_t + \Delta w_t$$

应用更新

- g_t 为第 t 次的梯度， r_t 为梯度累积变量， r 的初始值为0，**未必递增，通过参数来调节**。 β 为衰减因子， η 为全局学习率，需要手动设置。 ε 为小常数，为了数值稳定大约设置为 10^{-7} 。

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_1 和 v_1 ，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 $\beta_1 = 0.9$ ， $\beta_2 = 0.999$ ， $\epsilon = 10^{-8}$ ， $\eta = 0.001$ 。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。

目录

1. 感知机
2. **全连接神经网络及训练流程**
 - 深度神经网络
 - 常见的激活函数
 - 训练神经网络
 - 优化器
 - 正则化
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

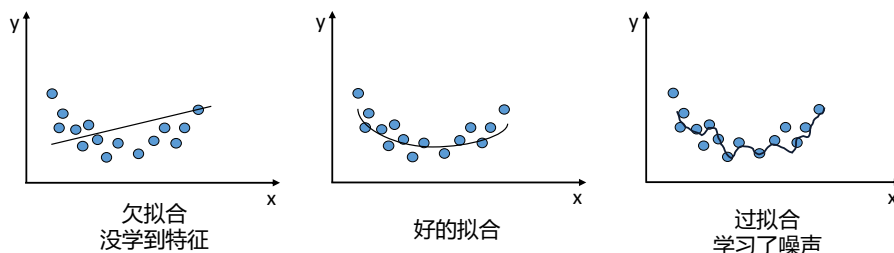
模型学习好了吗？

- 什么时候模型才算学习好了？
- 什么样的“状态”才是实际需要的？
- 完全学习到训练数据的特征就完美了吗？



过拟合问题

- 过拟合问题描述：模型在训练集表现优异，但在测试集上表现较差。
- 根本原因：特征维度过多，模型假设过于复杂，参数过多，训练数据过少，噪声过多，导致拟合出的函数几乎完美的对训练集做出预测，但对新数据的测试集预测结果差。过度的拟合了训练数据，而没有考虑到模型的泛化能力。



- 过拟合主要是有两个原因造成的：数据太少+模型太复杂：在深度学习中，经常出现的问题是，训练数据量小，模型复杂度高，这就使得模型在训练数据上的预测准确率高，但是在测试数据上的准确率低，这时就是出现了过拟合。
 - 获取更多数据：从数据源头获取更多数据；数据增强（Data Augmentation）。
 - 使用合适的模型：减少网络的层数、神经元个数等均可以限制网络的拟合能力。
 - Dropout。
 - 正则化，在训练的时候限制权值变大。
 - 限制训练时间；通过评估测试。
 - 增加噪声 Noise：输入时+权重上（高斯初始化）。
 - 数据清洗(data cleaning/Pruning)：将错误的label 纠正或者删除错误的的数据。
 - 结合多种模型：Bagging用不同的模型拟合不同部分的训练集；Boosting只使用简单的神经网络。

L_1 正则

- 对模型参数添加 L_1 范数约束，即

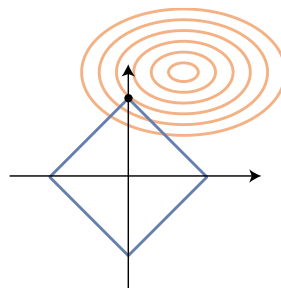
$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \|w\|_1,$$

- 如果通过梯度方法进行求解时，参数梯度为

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

- 参数优化方式为：

$$w_{t+1} = (w - \varepsilon \alpha \text{sign}(w)) - \varepsilon \nabla J(w)$$



- 特殊情况下，对于二次优化问题，并且假设对应的Hessian矩阵是对角矩阵，可以推导出参数递推公式为 $w_i = \text{sign}(w_i^*) \max(|w_i^*| - \frac{\alpha}{\lambda_i}, 0)$ ，从中可以看出当 $|w_i^*| < \frac{\alpha}{\lambda_i}$ 时，对应的参数会缩减到0，这也是和 L_2 正则不同地方。对比 L_2 优化方法， L_2 不会直接将参数缩减为0，而是一个非常接近于0的值。

L_2 正则

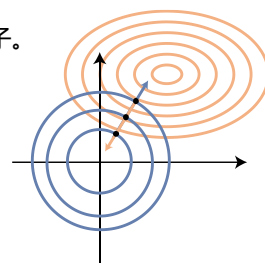
- 参数约束添加 L_2 范数惩罚项，该技术用于防止过拟合。

$$\tilde{J}(w; X, y) = J(w; X, y) + \frac{1}{2} \alpha \|w\|_2^2,$$

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：

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

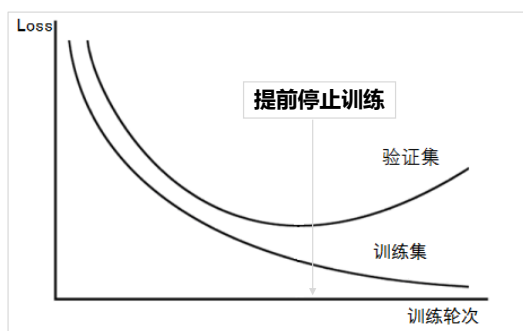
其中 ε 为学习率，相对于正常的梯度优化公式，对参数乘上一个缩减因子。



- 向目标函数添加一个正则项 $\Omega(\theta) = \frac{1}{2} \|w\|_2^2$ ，使权重更加接近原点
- 假设 J 是一个二次优化问题时，模型参数可以进一步表示为 $\tilde{w}_i = \frac{\lambda_i}{\lambda_i + \alpha} w_i$ ，即相当于在原来的参数上添加了一个控制因子，其中 λ 是参数Hessian矩阵的特征值。由此可见：
 - 当 $\lambda_i \gg \alpha$ 时，惩罚因子作用比较小。
 - 当 $\lambda_i \ll \alpha$ 时，对应的参数会缩减至0。

提前停止训练

- 在训练过程中，插入对验证集数据的测试。当发现验证集数据的Loss上升时，提前停止训练。

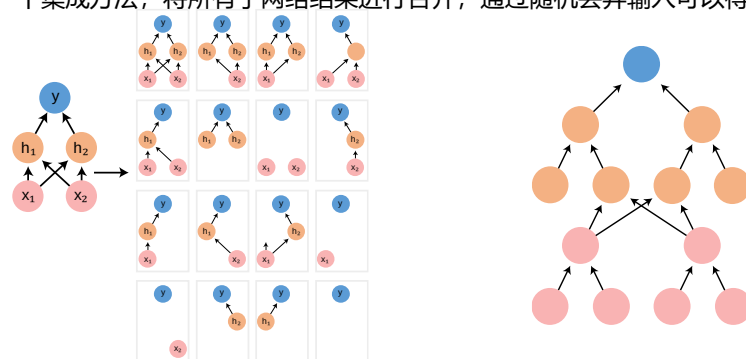


- 我们往往会设置一个比较大的迭代次数 n 。首先我们要保存好现在的模型(网络结构和权值),训练 num_batch 次(即一个epoch),得到新的模型。将测试集作为新模型的输入,进行测试。如果我们发现测试误差比上次得到的测试误差大,我们并不会马上终止测试,而是再继续进行几个epoch的训练与测试,如果测试误差依旧没有减小,那么我们就认为该试验在上一次达到最低测试误差时停下来训练模型时,
- 一般的做法是记录到目前为止最好的测试集准确率 p ,之后连续 m 个周期没有超过最佳测试集准确率 p 时,则可以认为 p 不再提高了,此时便可以提前停止迭代(Early-Stopping)。
- 从图中可以看出,测试误差在前几个epoch中逐渐减小,但是训练到某个epoch后,测试误差又有了小幅度的增大。这说明此时发生了过拟合。
- 发生过拟合是我们所不愿意看见的,我们可以利用提前终止(early stopping)来防止过拟合的发生。
- 提前终止是指:在测试误差开始上升之前,就停止训练,即使此时训练尚未收敛(即训练误差未达到最小值)。

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。

如图：



- 在每一轮的样本输入到神经网络进行训练时，设定一个概率 p ，使得每个神经元有一定的概率死亡，不参与网络的训练。
- 其流程为：1.首先以一定的概率 p 随机删除掉网络隐藏层中的神经元（暂时死亡），输入输出神经元保持不变。2.然后把输入 x 通过修改后的网络前向传播，把得到的损失结果通过修改的网络反向传播。一小批训练样本执行完这个过程后，在没有被删除的神经元上按照随机梯度下降法更新对应的参数（ w , b ）。3.然后继续重复这一过程：恢复被删掉的神经元（此时被删除的神经元保持原样，而没有被删除的神经元已经有所更新）从隐藏层神经元中随机选择一个一半大小的子集临时删除掉（备份被删除神经元的参数）。对一小批训练样本，先前向传播然后反向传播损失并根据随机梯度下降法更新参数（ w , b ）（没有被删除的那一部分参数得到更新，删除的神经元参数保持被删除前的结果）。
- 原始输入每一个节点选择概率0.8，隐藏层选择概率为0.5。
- Dropout的优点：
 - 相比于weight decay、范数约束等，该策略更有效。
 - 计算复杂度低，实现简单而且可以用于其他非深度学习模型。
 - 但是当训练数据较少时，效果不好。
 - Dropout训练过程中的随机过程不是充分也不是必要条件，可以构造不变的屏蔽参数，也能够得到足够好的解。
- 除上述的方法以外，还有半监督学习、多任务学习、提前停止、参数共享、集成化方法、对抗训练等方法。

目录

1. 感知机
2. 全连接神经网络及训练流程
- 3. 卷积神经网络**
4. 基于循环神经网络的模型架构
5. Transformer架构
6. 大模型基础架构

卷积神经网络

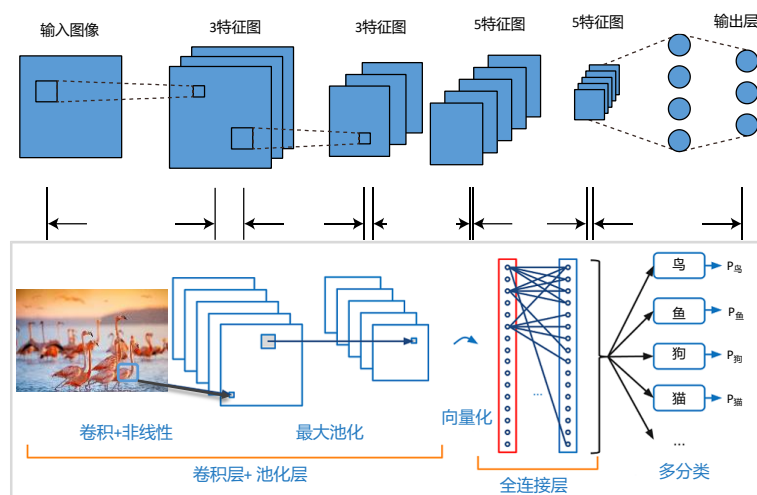
- 卷积神经网络 (Convolutional Neural Network, CNN) 是一种前馈神经网络，它的人工神经元可以**响应一部分覆盖范围内的单元**，对于图像处理有出色表现。它包括**卷积层** (convolutional layer)，**池化层** (pooling layer) 和**全连接层** (fully connected layer)。
- 20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络 (Convolutional Neural Networks-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。

- 对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的权重固定，所以又可以看做一个恒定的滤波器filter (kernel)）做内积（逐个元素相乘再求和）的操作就是所谓的卷积操作，也是卷积神经网络的名字来源。
- 局部感知，一般认为人对外界的认知是从局部到全局的，而图像的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。因而，每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部的信息综合起来就得到了全局的信息。网络部分连通的思想，也是受启发于生物学里面的视觉系统结构。视觉皮层的神经元就是局部接受信息的（即这些神经元只响应某些特定区域的刺激）。
- 参数共享，对输入的照片，用一个或者多个filter扫描照片，filter自带的参数就是权重，在同一个filter扫描的图层当中，每个filter使用同样的参数进行加权计算。权值共享意味着每一个过滤器在遍历整个图像的时候，过滤器的参数(即过滤器的参数的值)是固定不变的，比如我有3个特征过滤器，每个过滤器都会扫描整个图像，在扫描的过程中，过滤器的参数值是固定不变的，即整个图像的所有元素都“共享”了相同的权值。

卷积神经网络核心思想

- **局部感知**：一般认为，人对外界的认知是从局部到全局的，**而图像的空间联系也是局部的像素联系较为紧密，而距离较远的像素相关性则较弱**。因而，每个神经元没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部的信息综合起来就得到了全局的信息。
- **参数共享**：对输入的图片，用一个或者多个卷积核扫描照片，卷积核自带的参数就是权重，在同一个卷积核扫描的图层当中，每个卷积核使用同样的参数进行加权计算。权值共享意味着每一个卷积核在遍历整个图像的时候，**卷积核的参数是固定不变的**。

卷积神经网络架构



- 输入层，用于数据的输入。
- 卷积层 (Convolutional layer)，卷积神经网络中每层卷积层由若干卷积单元组成，每个卷积单元的参数都是通过反向传播算法优化得到的。卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。
- 线性整流层 (Rectified Linear Units layer, ReLU layer)，这一层神经的活性化函数 (Activation function) 使用线性整流 (Rectified Linear Units, ReLU) $f(x)=\max(0,x)$ 。
- 池化层 (Pooling layer)，通常在卷积层之后会得到维度很大的特征，将特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征。
- 全连接层 (Fully-Connected layer)，把所有局部特征结合变成全局特征，用来计算最后每一类的得分。
- 输出层，用于输出最终结果。
- 左侧图片：http://image.huawei.com/tiny-lts/v1/images/0281bccdff8137c18a163beba5a2650c_4715x2927.jpg

单卷积核计算 (1)

- 卷积计算描述

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

image 5*5

1	0	1
0	1	0
1	0	1

bias=0

filter 3*3

feature map 3*3

单卷积核计算 (2)

- 卷积计算结果演示

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

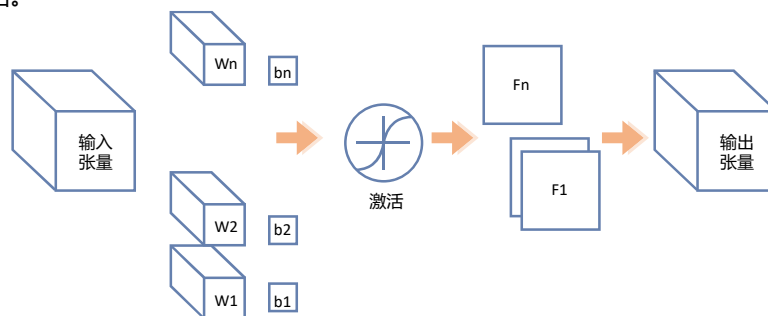
4		

Convolved
Feature

hanbingtao, 2017, 卷积神经网络

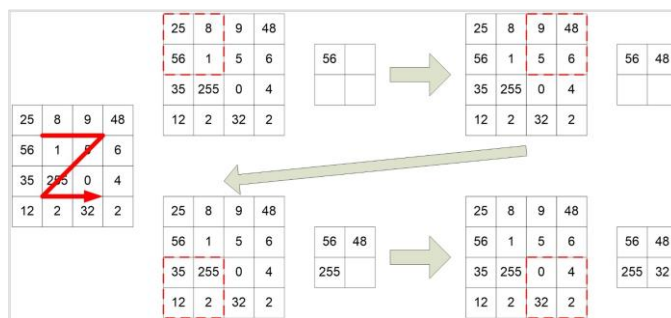
卷积层

- 卷积神经网络的基本结构，就是多通道卷积（由多个单卷积构成）。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。和全连接层类似，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

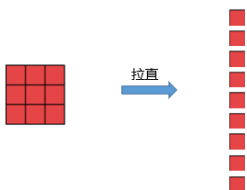
- 池化(Pooling)，它合并了附近的单元，减小了下层输入的尺寸，起到**降维**的作用。常用的池化有最大池化(Max Pooling)和平均池化(Average Pooling)，顾名思义，最大池化选择一小片正方形区域中最大的那个值作为这片小区域的代表，而平均池化则使用这片小区域的均值代表之。这片小区域的边长为池化窗口尺寸。下图演示了池化窗口尺寸为2的一般**最大池化**操作。



- 实际的分类网络，基本上都是卷基层和Pooling层交替互联，形成的前馈网络。Pooling层有如下作用：
 - 不变性：MaxPooling在一定范围内保证了不变性，因为一片区域的最大值无论在这篇区域的何处，最后输出都是它。
 - 减小下一层输入大小：Pooling有效降低了下一层输入数据的尺寸，减少了参数个数，减小了计算量。
 - 获得定长数据：通过合理设置Pooling的窗口尺寸和Stride，可以在变长输入中得到定长的输出。
 - 增大尺度：相当于扩大了尺度，可以从更大尺度上提炼上一层的特征。
 - 防止过拟合：Pooling简化了网络，降低了拟合的精度，因此可以防止过拟合（注意有可能带来的欠拟合）。

全连接层

- 全连接层实质上就是一个分类器，将前面经过卷积层与池化层所提取的特征，拉直后放到全连接层中，输出结果并分类。



- 通常我们使用Softmax函数作为最后全连接输出层的激活函数，把所有局部特征结合变成全局特征，用来计算最后每一类的得分。

$$\text{softmax} = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

目录

1. 感知机
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. **基于循环神经网络的模型架构**
 - RNN
 - LSTM
 - Seq2Seq
5. Transformer架构
6. 大模型基础架构

循环神经网络（RNN）

- 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。

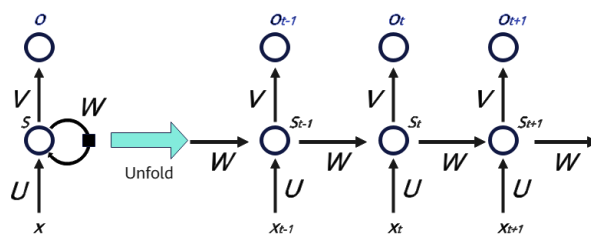
循环神经网络架构

- X_t 是输入序列在时间 t 处的输入
- S_t 是序列在时间 t 处的记忆单元，缓存了之前的信息：

$$S_t = \sigma(UX_t + WS_{t-1})。$$

- S_t 经过多个隐藏层，再通过全连接层 V 就可以得到该时序 t 的最终输出 O_t ：

$$O_t = \text{softmax}(S_t V)。$$



时序反向传播 (BPTT)

- 时序反向传播(BPTT):
 - 传统反向传播(BP)在时间序列上的拓展
 - 在t时序记忆单元的误差有两个来源，分别是t时序误差 C_t 对记忆单元的偏微分和记忆单元在下一个时序t+1的误差对t时序记忆单元的偏微分，需要将两者相加
 - 时序越长，最后一个时序的损失对第一个时序中w的梯度越容易出现梯度消失或梯度爆炸现象
 - 某个权重w的梯度是该权重在所有时序上梯度的累加
- BPTT的三个步骤：
 - 前向计算每个神经元的输出值
 - 反向计算每个神经元的误差值 δ_j
 - 计算每个权重的梯度
- 最后使用随机梯度下降算法更新权重。

- z_t 为memory cell进入激活函数前的值, $\delta_t = \frac{\partial C_t}{\partial z_t} + w^T \times f'_t(z_t) \times \delta_{t+1}$

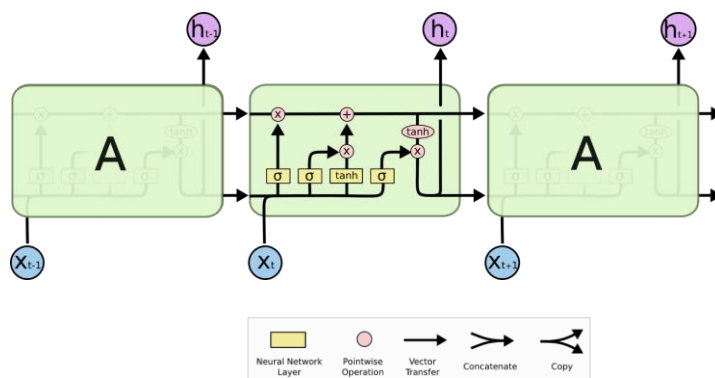
循环神经网络问题

- $S_t = \sigma(UX_t + WS_{t-1})$ 在时间序列上展开
- $S_t = \sigma\left(UX_t + W\left(\sigma\left(UX_{t-1} + W\left(\sigma\left(UX_{t-2} + W(\dots)\right)\right)\right)\right)\right)$
- 标准RNN结构解决了信息记忆的问题，但是对长时间**记忆的信息会衰减**。
 - 很多任务需要保存长时间的记忆信息，如推理小说开头埋下的伏笔，可能到结尾时候才解答。
 - 在记忆单元容量有限的情况下，RNN会丢失长时间间隔的信息。
 - 我们希望记忆单元能够选择性记住重点信息。

目录

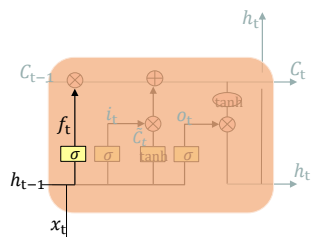
1. 感知机
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. **基于循环神经网络的模型架构**
 - RNN
 - LSTM
 - Seq2Seq
5. Transformer架构
6. 大模型基础架构

长短记忆性网络 (LSTM)



LSTM核心知识点：遗忘门

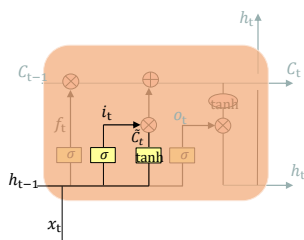
- LSTM中的第一步是决定从细胞状态中丢弃什么信息。
- 这个决定通过**遗忘门**完成。该门会读取 h_{t-1} 和 x_t ，输出一个在0到1之间的数值给每个在细胞状态 C_{t-1} 中的数字。1表示“完全保留”，0表示“完全舍弃”。



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM核心知识点：输入门

- 这一步确定什么样的新信息被存放在细胞状态中。
- 这里包含两个部分：
 - Sigmoid层称“输入门层” 决定什么值将要更新；
 - 一个tanh层创建一个新的候选值向量，会被加入到状态中。

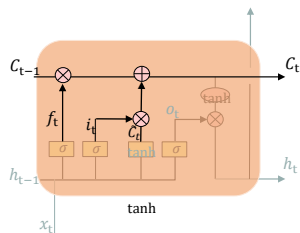


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \sigma(W_c \cdot [h_{t-1}, x_t] + b_c)$$

LSTM核心知识点：更新信息

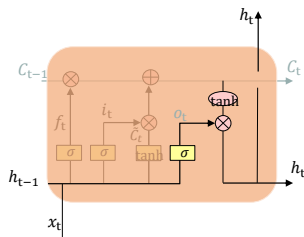
- 更新旧细胞状态的时间： C_{t-1} 更新为 C_t 。
- 我们把旧状态与 f_t 相乘，丢弃掉我们确定需要丢弃的信息。接着加上 $i_t * C_t$ 。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM核心知识点：输出门

- 运行一个Sigmoid层来确定细胞状态的哪个部分将输出出去。
- 接着，细胞状态通过tanh进行处理（得到一个在-1到1之间的值）并将它和Sigmoid门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。



$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

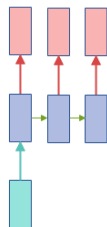
$$h_t = o_t * \tanh(C_t)$$

循环神经网络类型

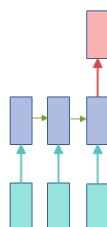
one to one



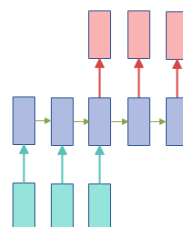
one to many



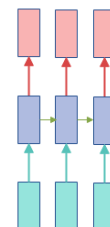
many to one



many to many



many to many

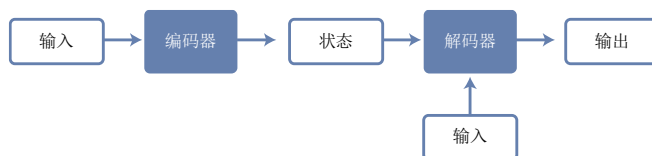


目录

1. 感知机
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. **基于循环神经网络的模型架构**
 - RNN
 - LSTM
 - Seq2Seq
5. Transformer架构
6. 大模型基础架构

Encoder – Decoder架构

- 机器翻译是序列转换模型的一个核心问题，其输入和输出都是长度可变的序列。为了处理这种类型的输入和输出，可以设计一个包含两个主要组件的架构：
 - 第一个组件是一个编码器（Encoder）：它接受一个长度可变的序列作为输入，并将其转换为具有固定形状的编码状态。
 - 第二个组件是解码器（Decoder）：它将固定形状的编码状态映射到长度可变的序列。
- 这被称为编码器-解码器（Encoder-Decoder）架构：

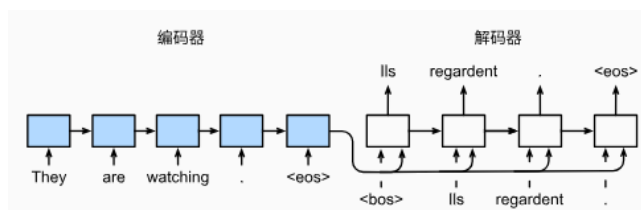


- 编码器和解码器可以选择RNN和LSTM等模型。

- 编码器和解码器可以选择RNN和LSTM等都可以。

序列到序列学习 (Seq2Seq)

- Seq2Seq (是 Sequence-to-sequence 的缩写)，输入一个序列，输出另一个序列。它是一个典型的 Encoder-Decoder 模型，编码器对输入数据进行编码，解码器对被编码的数据进行解码。同时，它也是一种时序问题解决方案。



- 蓝色为编码部分，第一个<eos>表示编码结束。白色为解码部分，<bos>是第一个输入，表示解码开始，第二个<eos>表示解码结束。

编码器计算过程

- 编码器将长度可变的输入序列转换成形状固定的上下文变量c，并且将输入序列的信息在该上下文变量中进行编码。
 - 假设输入序列是 $x_1, \dots, x_t$ ，其中 x_t 是输入文本序列中的第t个词元。在时间步t，循环神经网络将词元 x_t 的输入特征向量 x_t 和 h_{t-1} （即上一时间步的隐状态）转换为 h_t （即当前步的隐状态）。使用一个函数f来描述循环神经网络的循环层所做的变换：

$$h_t = f(x_t, h_{t-1})$$

- 编码器通过选定的函数，将所有时间步的隐状态转换为上下文变量：

$$c = q(h_1, \dots, h_T)$$

解码器计算过程

- 编码器输出的上下文变量 c 对整个输入序列 $x_1, \dots, x_T$ 进行编码。来自训练数据集的输出序列 $y_1, y_2, \dots, y_{T'}$ ，对于每个时间步 t' （与输入序列或编码器的时间步 t 不同），解码器输出 $y_{t'}$ 的概率取决于先前的输出子序列 $y_1, \dots, y_{t'-1}$ 和上下文变量 c ，即 $P(y_{t'} | y_1, \dots, y_{t'-1}, c)$ 。
- 在输出序列上的任意时间步 t' ，循环神经网络将来自上一时间步的输出 $y_{t'-1}$ 和上下文变量 c 作为其输入，然后在当前时间步将它们和上一隐状态 $s_{t'-1}$ 转换为隐状态 $s_{t'}$ 。因此，可以使用函数 g 来表示解码器的隐藏层的变换：

$$s_{t'} = g(y_{t'-1}, c, s_{t'-1})$$

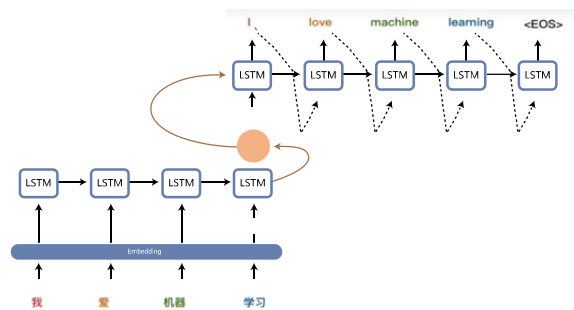
- 在获得解码器的隐状态之后，我们可以使用输出层和softmax操作来计算在时间步 t' 时输出 $y_{t'}$ 的条件概率分布 $P(y_{t'} | y_1, \dots, y_{t'-1}, c)$ 。

Seq2Seq的应用 – 机器翻译

中文(简体) ▾ ≡ 英文 ▾ 通用领域 ▾

机器翻译 术语和关键词

我爱机器学习 × I love machine learning.



Seq2Seq的缺陷

- 虽然Seq2Seq能解决序列问题，但是它依然存在一些问题：
 - Encoder 和 Decoder 之间只有一个向量 C 来传递信息，且 C 的长度固定。
 - 语义向量无法完全表示整个序列的信息： c 长度固定，如果输入数据过长，数据被压缩到指定长度，会丢失信息，输入越长，压缩越大，丢失的信息越多。
 - 当句子长度较大时，容易丢失信息：通过RNN编码得到的 c ，会逐渐丢失前面的信息。
 - 基于 RNN 的 Seq2Seq 模型不能并行计算，效率低。
 - 由于RNN计算 t 时刻数据时，都需要上一时刻的输出，因此无法实现高度并行。
 - 基于CNN的Seq2Seq参数量较大，训练复杂。
 - 推理过程中，必须要先输出前一个词才能预测下一个词，推理速度慢。

目录

1. 感知机
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. 基于循环神经网络的模型架构
- 5. Transformer架构**
6. 大模型基础架构

如何解决Seq2Seq的缺陷

- 按顺序将全部信息逐步保存下来的效率比较慢，是否有一种方式可以优化这种方案呢？
- 参考一下人类观察事物的方式：通常人类的视野比较广，第一眼会观察到视野内全部信息，然后迅速捕捉重要信息，并忽略其他信息。是否可以将这种机制应用到序列模型中？



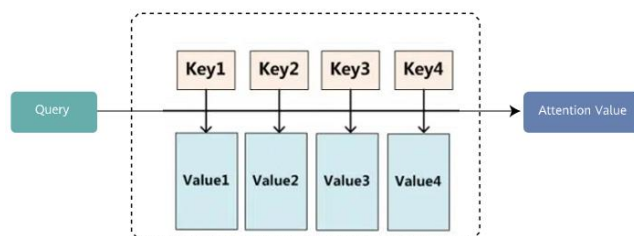
Attention机制

- Attention（注意力）机制其实来源于人类的认知能力。比如当人们观察一个场景或处理一件事情时，人们往往会关注场景的显著性物体，处理事情时则希望抓住主要矛盾。注意力机制使得人类能够关注事物的重要部分，忽略次要部分，更高效的处理所面临的各种事情。



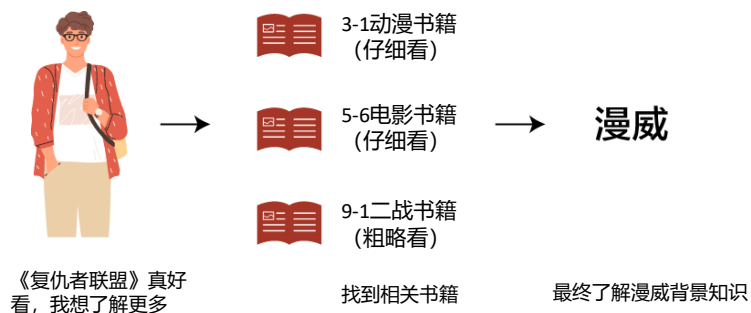
Attention原理

- Attention可以脱离Encoder-Decoder框架使用，下图是它的脱离Encoder-Decoder框架后的原理图解：



Attention在现实中的理解

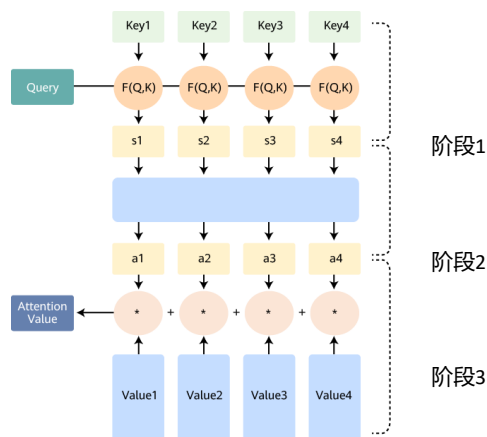
- 图书馆 (source) 里有很多书 (value)，为了方便查找，我们给书做了编号 (key)。当我们想要了解漫威 (query) 的时候，我们就可以看看那些动漫、电影、甚至二战 (美国队长) 相关的书籍。



Attention计算过程

- Attention 原理的3步：

- 第一步：Query与每一个Key计算相似性得到相似性评分s
- 第二步：将s评分进行softmax转换成[0,1]之间的概率分布，得到直接可用的权重
- 第三步：将[a1,a2,a3...an]作为权值矩阵对Value进行加权求和得到最后的Attention值



- 从上面的建模，我们可以大致感受到 Attention 的思路简单，四个字“带权求和”就可以高度概括，大道至简。做个不太恰当的类比，人类学习一门新语言基本经历四个阶段：死记硬背（通过阅读背诵学习语法练习语感）->提纲挈领（简单对话靠听懂句子中的关键词汇准确理解核心意思）->融会贯通（复杂对话懂得上下文指代、语言背后的联系，具备了举一反三的学习能力）->登峰造极（沉浸地大量练习）。
- 这也如同attention的发展脉络，RNN 时代是死记硬背的时期，attention 的模型学会了提纲挈领，进化到transformer，融汇贯通，具备优秀的表达学习能力，再到 GPT、BERT，通过多任务大规模学习积累实战经验，战斗力爆棚。

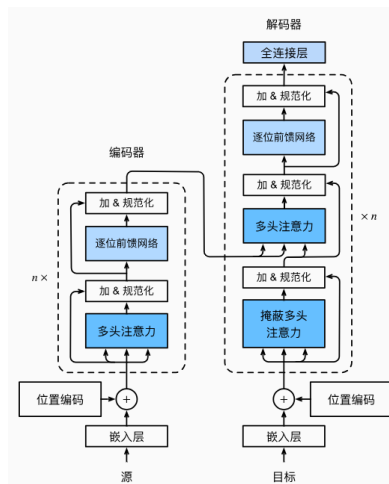
Attention的优点

- 注意力机制的三大优点是：
 - 参数少：模型复杂度跟 CNN、RNN 相比，复杂度更小，参数也更少。所以对算力的要求也就更小。
 - 速度快：Attention 解决了 RNN 不能并行计算的问题。Attention机制每一步计算不依赖于上一步的计算结果，因此可以和CNN一样并行处理。
 - 效果好：在 Attention 机制引入之前，长距离的信息会被弱化，就好像记忆能力弱的人，记不住过去的事情是一样的。



Transformer

- 2017年谷歌在一篇名为《Attention Is All You Need》的论文中,提出了一个基于Attention(自注意力机制)结构来处理序列相关的问题的模型,名为Transformer。
- Transformer在很多不同NLP任务中获得了成功,例如:文本分类、机器翻译、阅读理解等。
- 在解决这类问题时,Transformer模型摒弃了固有的定式,并没有用任何CNN或者RNN的结构,而是使用了Attention注意力机制,自动捕捉输入序列不同位置处的相对关联,善于处理较长文本,并且该模型可以高度并行地工作,训练速度很快。



Self-Attention

- Attention机制发生在Target的元素和Source中的所有元素之间。在一般任务的Encoder-Decoder框架中，输入Source和输出Target内容是不一样的，比如对于英 - 中机器翻译来说，Source是英文句子，Target是对应的翻译出的中文句子。

I love machine learning.

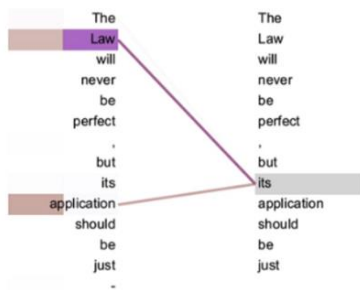


我爱机器学习

- Transformer中使用了Self-Attention机制，Self-Attention顾名思义，指的不是Target和Source之间的Attention机制，而是Source内部元素之间或者Target内部元素之间发生的Attention机制，其具体计算过程是一样的，只是计算对象发生了变化。

Self-Attention示例

- 自注意力机制是注意力机制的变体，其减少了对外部信息的依赖，更擅长捕捉数据或特征的内部相关性。
- 自注意力机制在文本中的应用，主要通过计算单词间的互相影响，解决长距离依赖问题：



若想知道在这句话里 its 指代的是什麼，与哪一些单词相关，那么就可以将 its 作为 Query，然后将这一句话作为 Key 和 Value 来计算 attention 值，找到与这句话中 its 最相关的单词。通过 self-attention 我们发现 its 在这句话中与之最相关的是 Law 和 application

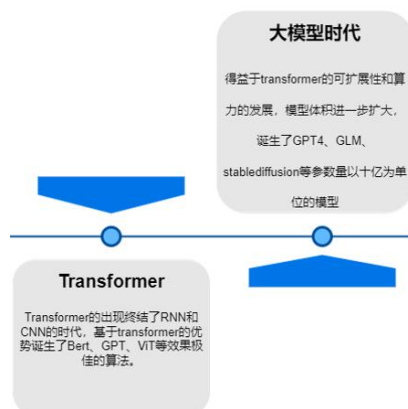
- 自注意力机制更注重自身的关系，因此它的encoder和decoder部分可以拆分开使用，这也产生了不同类型的网络结构。

目录

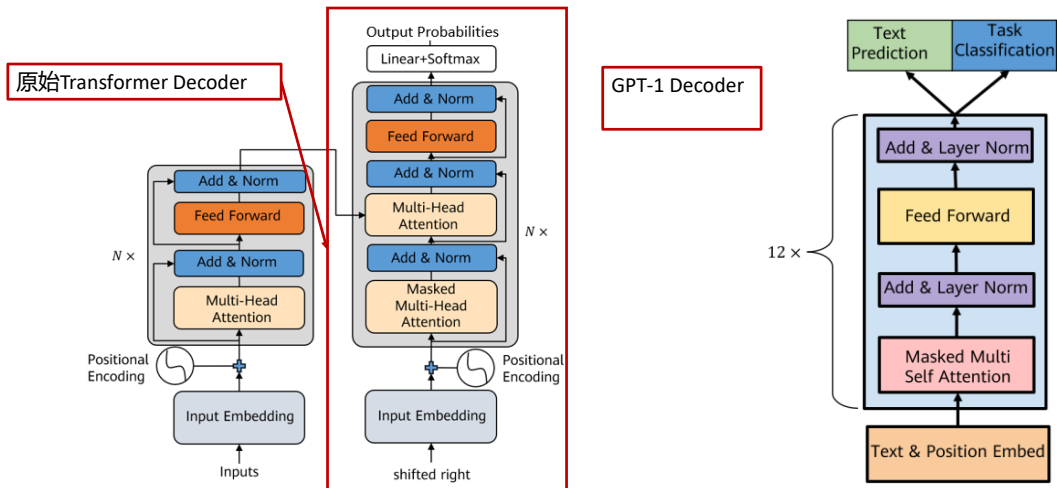
1. 感知机
2. 全连接神经网络及训练流程
3. 卷积神经网络
4. 基于循环神经网络的模型架构
5. Transformer架构
- 6. 大模型基础架构**

大模型阶段

- 随着计算能力的提升和数据量的增长，训练更大更深的神经网络成为了可能。以GPT系列、LLaMA、GLM等为代表的大模型，通过海量的数据和计算资源，训练出了具有强大生成能力和泛化能力的模型。
- 这些模型不仅能够在单一任务上取得优异性能，还能够通过微调的方式，快速适应新的任务和数据。



GPT - 1



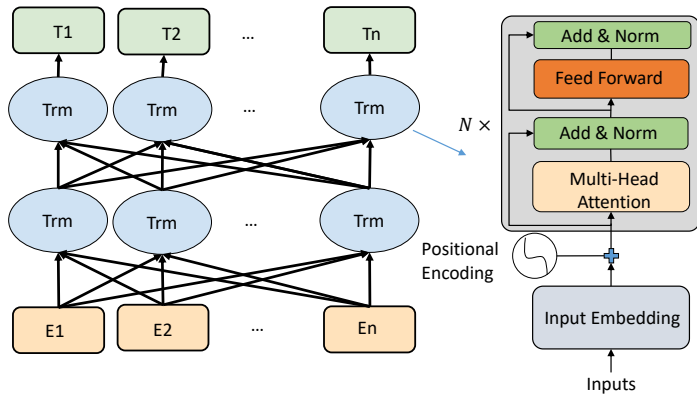
103 Copyright © Huawei Technologies Co., Ltd. All rights reserved.



- OpenAI的第一个版本的GPT-1当时称之为GPT。
- 使用Transformer而不使用LSTM在于，虽然预训练有助于捕获一些语言信息，但LSTM模型限制了他们在短期内的预测能力。相反，Transformer的选择使我们能够捕获更长的语言结构，同时在其他不同的任务中泛化性能更好。
- GPT-1使用的是Transformer的Decoder框架。
- GPT-1 保留了Decoder的Masked Multi-Attention 层和Feed Forward层，并扩大了网络的规模。将层数扩展到12层，GPT-1还将Attention 的维数扩大到768（原来为512），将Attention的头数增加到12层（原来为8层），将Feed Forward层的隐层维数增加到3072（原来为2048），总参数达到1.17亿。
- 对于位置编码的部分，实际上GPT-1和普通的Transformer的区别还是很大的，普通的Transformer的位置编码，是由余弦+正弦的方式表示出来的，而GPT-1中，采用与词向量相似的随机初始化，并在训练中进行更新，即是把每一个位置当做一个要学习的embedding来做。

Bert

- BERT于2018年发布，是一种双向模型，它分析完整序列的上下文，然后进行预测。该模型在纯文本语料库和 Wikipedia 上进行训练，使用了 33 亿个单词和3.4亿个参数。
- BERT 可以回答问题、预测语句和翻译文本。



多领域任务处理

- 以往神经网络模型基本上是针对单一场景处理专业问题，每个模型可能只被应用在一个或者少数几个任务中。



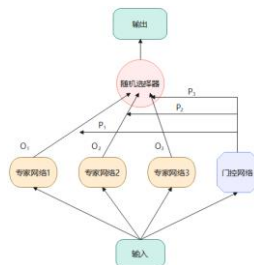
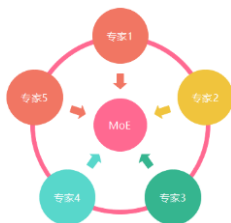
- 是否可以一个模型处理多个领域的任务呢？



- 在实际的数据分布中，存在着很多天然的子集，如不同的domain、topic，不同的语言、不同的模态等等，在使用单个模型进行学习，且模型capacity较小的情况下，不同子集会起到噪声的作用，干扰模型的拟合，导致模型训练慢、泛化困难。
- 对于传统的学习模型来说，训练的主要目的是使最终模型能够在不同的场景下执行多种任务。但这种训练方式也使得模型在对相应场景进行权重更新的同时，也会影响到模型对其它场景的权重。这种权重的互相影响称为**干扰效应 (interference effect)**。**干扰效应越强，模型的学习速率(Learning Speed)越慢，而且最终模型的泛化性(Generalization)也会越差。**

MoE (Mixture-of-Experts)

- 混合专家模型，英文叫Mixture of Experts (MoE) 是一种模型设计策略，它通过将多个模型（称为"专家"）直接结合在一起，以获得更好的预测性能。
- 在大模型中，MoE方案可以有效地提高模型的容量和效率。一般而言，大模型的MoE有一个门控机制和一套门控输出机制来合并和平衡专家的选择，用于决定每个专家对最终预测的；有一套专家选择机制，会根据门控机制的输出选择一部分专家进行预测。



106 Copyright © Huawei Technologies Co., Ltd. All rights reserved.



- 模型规模是提升模型性能的关键因素之一，这也是为什么今天的大模型能取得成功。在有限的计算资源预算下，用更少的训练步数训练一个更大的模型，往往比用更多的步数训练一个较小的模型效果更佳。
- MoE 的一个显著优势是它们能够在远少于 Dense 模型所需的计算资源下进行有效的预训练。这意味着在相同的计算预算条件下，您可以显著扩大模型或数据集的规模。特别是在预训练阶段，与稠密模型相比，混合专家模型通常能够更快地达到相同的质量水平。

思考题

1. (单选) 深度学习神经网络种类繁多，不属于深度学习神经网络的是？（ ）
- A. CNN
 - B. RNN
 - C. LSTM
 - D. Logistic

- 1.D

思考题

2. (多选) 卷积神经网络架构中有很多重要的“部件”，以下属于卷积神经网络“部件”是？（ ）
- A. 激活函数
 - B. 卷积核
 - C. 池化
 - D. 全连接层

- 2.ABCD

思考题

3. (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

- 3.A

思考题

4. (判断题) Transformer使用了encoder-decoder架构，因此它和Seq2Seq模式有类似的特性。
()
- A. True
- B. False

- B

本章总结

- 本章深入探讨了深度神经网络的训练流程，涵盖了从传统人工神经网络发展而来的多种网络架构，并最终阐述了构建大型模型的基础结构。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



人工智能开发框架



前言

- 深度学习研究和应用在近几十年得到了爆炸式的发展，掀起了人工智能的第三次浪潮，并且在图像识别、语音识别与合成、无人驾驶、机器视觉等方面取得了巨大的成功。这也对算法的应用以及依赖的框架有了更高级的要求。
- 深度学习框架的不断发展使得在大型数据集上训练神经网络模型时，可以方便地使用大量的计算资源。本章节将介绍人工智能框架的使用和AI应用开发流程。

目录

1. 学完本课程后，您将能够：

- 了解当前主流开发框架；
- 掌握AI应用开发流程；
- 了解MindSpore结构和特性；
- 了解PyTorch框架的使用。

目录

1. 人工智能框架

- AI开发框架的作用
- 主流AI开发框架介绍

2. AI开发框架基础知识

3. PyTorch

4. MindSpore

5. AI应用开发流程

AI开发框架



01. 数据预处理

对音、视、图、文等不同格式的数据进行预处理（数据转换、数据增强）

02. 开发接口

为开发者提供统一的AI开发全流程接口，加快科研、开发进展

03. 调试调优

提供可视化调试调优能力，便于查看和跟踪模型训练过程

04. 编译&执行

对用户代码进行编译，全栈软硬协同实现高性能训练

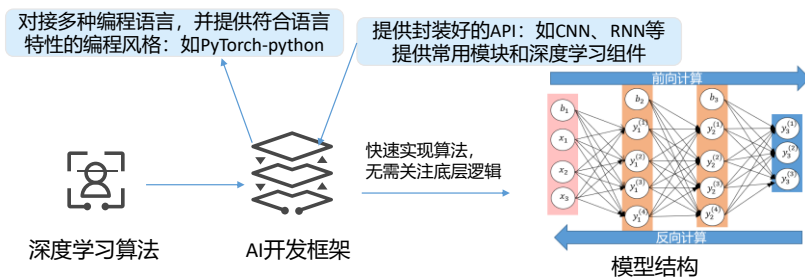
05. 推理部署

实现模型下发与计算调度，支撑行业客户快速部署，最大程度发挥昇腾芯片算力

- 深度学习框架的出现降低了入门的门槛。我们不需要从复杂的神经网络和反向传播算法开始编代码，可以依据需要，使用已有的模型配置参数，而模型参数自动训练得到。我们也可以在已有模型的基础上增加自定义网络层，或者是在顶端选择自己需要的分类器和优化算法。

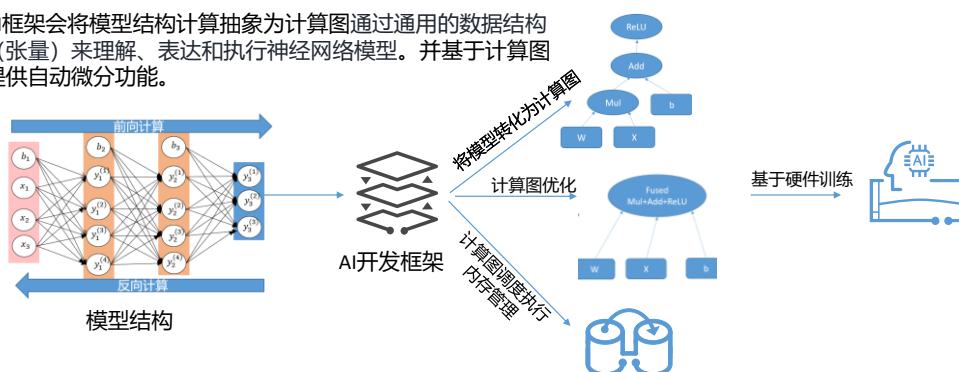
前端编程接口

AI框架可以让开发者更加专注于应用程序的业务逻辑，而不需要关注底层的数学和计算细节。



计算优化与调度

AI框架会将模型结构计算抽象为计算图通过通用的数据结构（张量）来理解、表达和执行神经网络模型。并基于计算图提供自动微分功能。

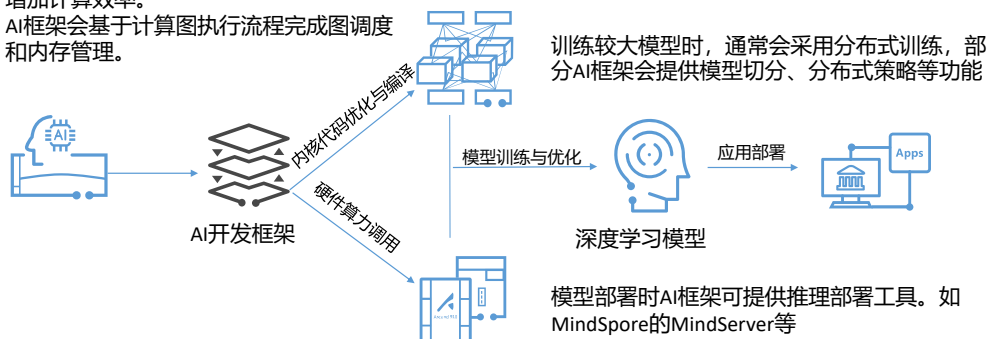


AI框架通常会对计算图做进一步的优化来增加计算效率。
AI框架会基于计算图执行流程完成图调度和内存管理。

- 模型结构构建完成后，AI框架会在执行前将模型转换为计算图的形式，并对计算图做进一步优化。

模型训练与部署

AI框架通常会对计算图做进一步的优化来增加计算效率。
AI框架会基于计算图执行流程完成图调度和内存管理。



目录

1. 人工智能框架

- AI开发框架的作用
- 主流AI开发框架介绍

2. AI开发框架基础知识

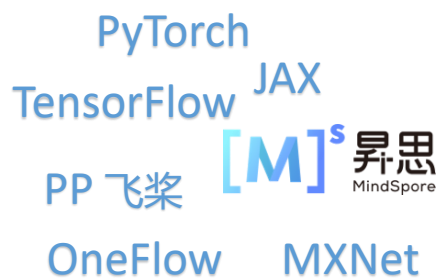
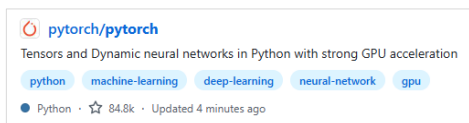
3. PyTorch

4. MindSpore

5. AI应用开发流程

百家争鸣 - AI框架

- 当前AI开发框架众多，每个框架特点不同，适配与不同的领域和场景。
- 当前大模型领域使用较多的开发框架为PyTorch。



PyTorch

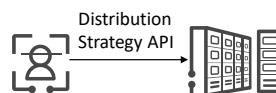
- PyTorch是由Meta发布的机器学习计算框架。它的前身是Torch。Torch是一个有大量机器学习算法支持的科学计算框架，是一个与Numpy类似的张量（Tensor）操作库，其特点是特别灵活，但因其采用了小众的编程语言是Lua，所以流行度不高，于是就有了基于Python的PyTorch。
- PyTorch 是一个针对深度学习，并且使用GPU和NPU来优化的tensor library。是当前的主流深度学习框架之一。

PyTorch特点

- Python优先：PyTorch不是简单地在C++框架上绑定 Python。 PyTorch从细粒度上直接支持Python的访问。可以像使用Numpy或者Scipy那样轻松地使用PyTorch。
- 动态神经网络： PyTorch默认使用动态计算图，程序可以在执行时动态构建/调整计算图。
- 易于Debug：PyTorch在运行时可以生成动态图，开发者可以在调试器中停掉解释器并查看某个节点的输出。
- 丰富的生态系统：PyTorch拥有一个活跃的社区，提供了大量的文档、教程、模型和工具。

TensorFlow

- TensorFlow是一个端到端开源机器学习平台，由Google团队设计，目前主要版本为TensorFlow2.X。
- TensorFlow提供多个级别的API，可以根据需求选择合适的级别。
 - 如可以使用高阶Keras API快速构建和训练模型。
- 对于大型深度学习学习训练任务，可以使用Distribution Strategy API（用于跨多个处理单元进行分布式训练）在不同的硬件配置上进行分布式训练，而无需更改模型定义。



TensorFlow生产工具

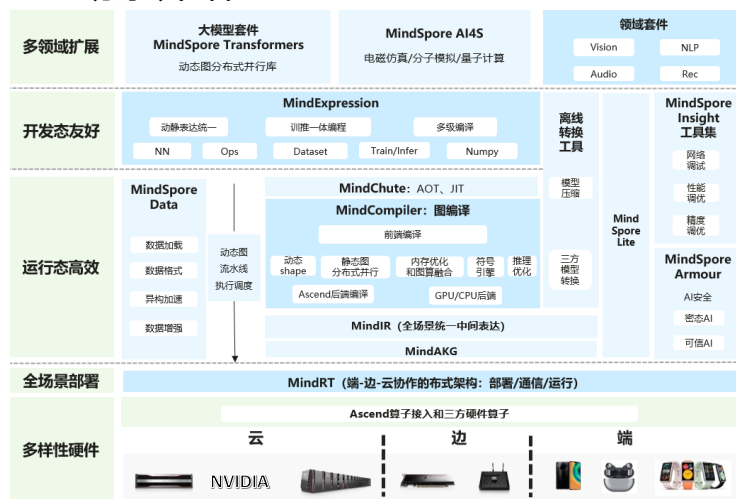
- TensorFlow.js：是一个用于在JavaScript中开发和训练ML模型，并在浏览器或Node.js上部署的库。
- TensorFlow Lite：是一个用于设备推理的开源深度学习框架。它可以满足在移动和物联网设备上部署机器学习模型。
- TensorFlow Extended (TFX)：是用于部署生产ML管道的端到端平台，当训练好的模型准备从研究转移到生产时，使用TFX创建和管理生产管道。

MindSpore

- MindSpore作为新一代深度学习框架，是源于全产业的最佳实践，最佳匹配昇腾处理器算力，支持终端、边缘、云全场景灵活部署，开创全新的AI编程范式，降低AI开发门槛。
 - MindSpore旨在实现易开发、高效执行、全场景覆盖三大目标。
 - MindSpore可以支持数据并行、模型并行和混合并行训练，具有很强的灵活性。
 - MindSpore有“自动并行”能力，它通过在庞大的策略空间中进行高效搜索来找到一种快速的并行策略。



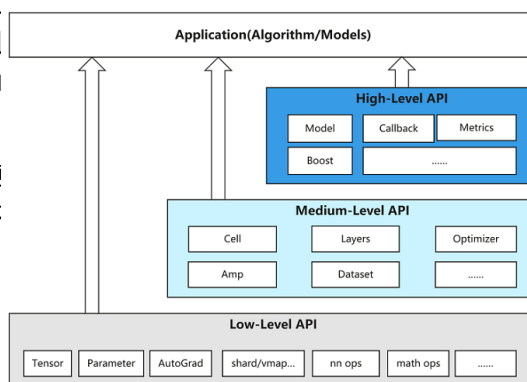
MindSpore全场景介绍



- 昇思MindSpore是一个全场景深度学习框架，旨在实现易开发、高效执行、全场景统一部署三大目标。其中易开发表现为API友好，调试难度低；高效执行包括计算效率、数据预处理效率和分布式训练效率；全场景则指框架同时支持云、边缘以及端侧场景。

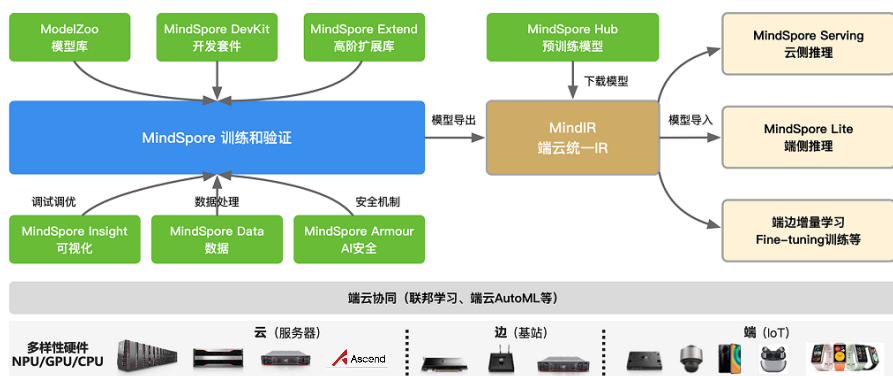
层次结构

- MindSpore向用户提供了3个不同层次的API，支撑用户进行AI应用（算法/模型）开发，从高到低分别为High-Level Python API、Medium-Level Python API以及Low-Level Python API。
- 高阶API提供了更好的封装性，低阶API提供更好的灵活性，中阶API兼顾灵活及封装，满足不同领域和层次的开发者需求。



- MindExpression层次结构。
- **High-Level Python API:**
 - 第一层为高阶API，其在中阶API的基础上又提供了训练推理的管理、混合精度训练、调试调优等高级接口，方便用户控制整网的执行流程和实现神经网络的训练推理及调优。例如用户使用Model接口，指定要训练的神经网络模型和相关的训练设置，对神经网络模型进行训练。
- **Medium-Level Python API:**
 - 第二层为中阶API，其封装了低阶API，提供网络层、优化器、损失函数等模块，用户可通过中阶API灵活构建神经网络和控制执行流程，快速实现模型算法逻辑。例如用户可调用Cell接口构建神经网络模型和计算逻辑，通过使用Loss模块和Optimizer接口为神经网络模型添加损失函数和优化方式，利用Dataset模块对数据进行处理以供模型的训练和推导使用。
- **Low-Level Python API:**
 - 第三层为低阶API，主要包括张量定义、基础算子、自动微分等模块，用户可使用低阶API轻松实现张量定义和求导计算。例如用户可通过Tensor接口自定义张量，使用ops.composite模块下的GradOperation算子计算函数在指定处的导数。

MindSpore全场景支持模式



- MindSpore作为全场景AI框架，所支持的有端（手机与IoT设备）、边（基站与路由设备）、云（服务器）场景的不同系列硬件，包括昇腾系列产品、英伟达NVIDIA系列产品、Arm系列的高通骁龙、华为麒麟的芯片等系列产品。
- 左边蓝色方框的是MindSpore主体框架，主要提供神经网络在训练、验证相关的基础API功能，另外还会默认提供自动微分、自动并行等功能。
- 蓝色方框往下是MindSpore Data模块，可以利用该模块进行数据预处理，包括数据采样、数据迭代、数据格式转换等不同的数据操作。在训练的过程会遇到很多调试调优的问题，因此有MindSpore Insight模块对loss曲线、算子执行情况、权重参数变量等调试调优相关的数据进行可视化，方便用户在训练过程中进行调试调优。
- AI安全最简单的方式就是从攻防的视角来看，例如，攻击者在训练阶段掺入恶意数据，影响AI模型推理能力，于是MindSpore推出了MindSpore Armour模块，为MindSpore提供AI安全机制。
- 蓝色方框往上的内容跟算法开发相关的用户更加贴近，包括存放大量的AI算法模型库ModelZoo，提供面向不同领域的开发工具套件MindSpore DevKit，另外还有高阶拓展库MindSpore Extend，这里面值得一提的就是MindSpore Extend中的科学计算套件MindSciences，MindSpore首次探索将科学计算与深度学习结合，将数值计算与深度学习相结合，通过深度学习来支持电磁仿真、药物分子仿真等等。
- 神经网络模型训练完后，可以导出模型或者加载存放在MindSpore Hub中已经训练好的模型。接着有MindIR提供端云统一的IR格式，通过统一IR定义了网络的逻辑结构和算子的属性，将MindIR格式的模型文件与硬件平台解耦，实现一次训练多次部署。因

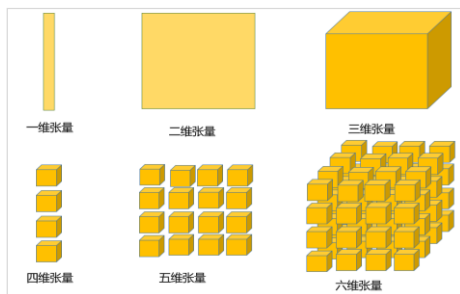
此如图所示，通过IR把模型导出到不同的模块执行推理。

目录

1. 人工智能框架
- 2. AI开发框架基础知识**
 - 基础知识介绍
 - 计算图
3. PyTorch
4. MindSpore
5. AI应用开发流程

tensor (张量)

- AI开发框架中最基础的数据结构就是张量（tensor）。所有数据都被封装在tensor中。
- tensor定义：一个多维数组。
 - 零阶张量：标量；
 - 一阶张量：向量；
 - 二阶张量：矩阵。

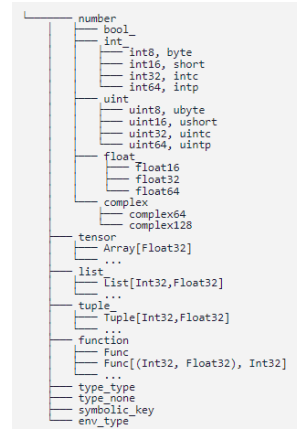


tensor的属性

tensor属性	功能
形状 (shape)	tensor的每个维度的大小，如[N,C,H,W]。
秩或维数 (dim)	表示tensor的维数，标量为0，向量为1。
数据类型 (dtype)	表示tensor的数据类型，如bool、uint8、int16、float32、float64等。
存储位置 (device)	tensor所在的设备位置，如NPU、GPU等。
名字 (name)	tensor的标识符。

数据类型

- 当前AI开发框架支持的数据类型包括数字（float32、float16、int8、bf16等）、bool、complex等（部分框架可能额外支持匹配某些硬件的数据类型）。
- AI框架的数据类型和使用到的编程语言类型可兼容使用。
 - 如python和numpy中的类型在PyTorch中使用（通过`tensor.numpy()`和`torch.from_numpy(data)`声明后使用）。



数据加载

- AI开发框架中提供数据加载模块，该模块可以直接加载一些开源数据集，如MNIST、CIFAR-10、CIFAR-100、VOC、COCO、ImageNet等。
 - 如PyTorch中`torchvision.datasets.CIFAR10`可以直接加载CIFAR10数据（本地不存在时会自动下载）。
 - MindSpore中`mindspore.dataset.CocoDataset`支持COCO2017数据集。

```
#（本地时不存在）下载并加载训练集
# transform为对数据做的装换操作
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
```

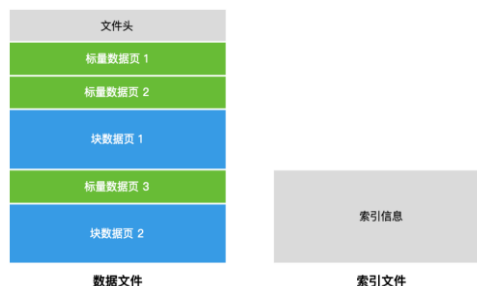
加载后的数据格式

- 数据集在加载以后通常以多维张量的形式存在。
 - 序列数据通常为3维张量，格式为：[batch_size, sequence_length, feature_length]。
 - 比如NLP中句子的嵌入表示 [批量大小，句子长度，词嵌入维度]。
 - 图像数据通常为4维张量，格式为[N,H,W,C]（N为批量数据大小，HW为图片像素高宽，C为通道数）。
 - 在PyTorch中，默认接收的图像数据张量格式为[N,C,H,W]。
 - 在TensorFlow中，默认接收的图像数据张量格式为[N,H,W,C]。

MindRecord、TFRecord

- 对于数据量较多的数据集，为了高效地读取数据，比较有帮助的一种做法是对数据进行序列化并将其存储在一组可线性读取的文件中。这尤其适用于通过网络进行流式传输的数据。对缓冲任何数据预处理也十分有用。

- 如MindSpore中的MindRecord和TensorFlow中TFRecord等。
- 这种数据格式可以减少磁盘IO、网络IO开销。



- 其中数据文件包含文件头、标量数据页、块数据页，用于存储用户归一化后的训练数据，且单个MindSpore Record文件建议小于20G，用户可将大数据集进行分片存储为多个MindSpore Record文件。而索引文件则包含基于标量数据（如图像Label、图像文件名等）生成的索引信息，用于方便的检索、统计数据集信息。数据文件中的文件头、标量数据页、块数据页的具体用途如下所示：
 - 文件头**：是MindSpore Record文件的元信息，主要用来存储文件头大小、标量数据页大小、块数据页大小、Schema信息、索引字段、统计信息、文件分区信息、标量数据与块数据对应关系等。
 - 标量数据页**：主要用来存储整型、字符串、浮点型数据，如图像的Label、图像的文件名、图像的长宽等信息，即适合用标量来存储的信息会保存在这里。
 - 块数据页**：主要用来存储二进制串、NumPy数组等数据，如二进制图像文件本身、文本转换成的字典等。
- MindSpore Record数据格式具备的特征如下：
 - 实现数据统一存储、访问，使得训练时数据读取更加简便。
 - 数据聚合存储、高效读取，使得训练时数据方便管理和移动。
 - 高效的数据编解码操作，使得用户可以对数据操作无感知。
 - 可以灵活控制数据切分的分区大小，实现分布式数据处理。

模型文件

AI开发框架可以基于训练好的模型，在不同的硬件平台上执行推理任务。

推理模型文件保存支持两种类型的数据：**训练参数和网络模型**。

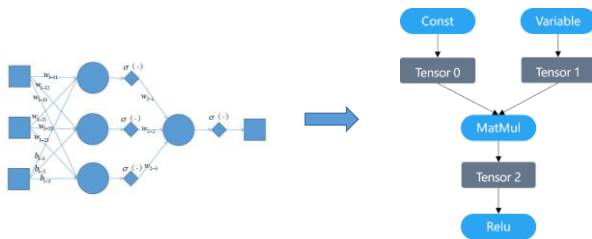
文件格式	基本概念	应用场景
Checkpoint (.ckpt)	采用了Protocol Buffers格式，默认存储了网络中所有的参数值（也可配置用于存储模型结构、训练状态等）。	一般用于训练任务中断后恢复训练，或训练后的微调（Fine Tune）任务。
ONNX (.onnx)	全称Open Neural Network Exchange，是一种针对机器学习模型的通用表达。	一般用于不同框架间的模型迁移或在推理引擎（TensorRT）上使用。
bin (.bin)	一种通用的二进制格式，可以用于保存和加载各种类型的模型和数据。	需要将 PyTorch 模型转换为通用的二进制格式的场景。
MindIR (.mindir)	全称MindSpore IR，是MindSpore的一种基于图表示的函数式IR，定义了可扩展的图结构以及算子的IR表示。	它消除了不同后端的模型差异，一般用于跨硬件平台执行推理任务。
.pt 或 .pth (.pt/.pth)	PyTorch 的默认模型文件格式，用于保存和加载完整的 PyTorch 模型，包含模型的结构和参数等信息。	需要保存和加载完整的 PyTorch 模型的场景，例如在训练中保存最佳的模型或在部署中加载训练好的模型。

目录

1. 人工智能框架
- 2. AI开发框架基础知识**
 - 基础知识介绍
 - 计算图
3. PyTorch
4. MindSpore
5. AI应用开发流程

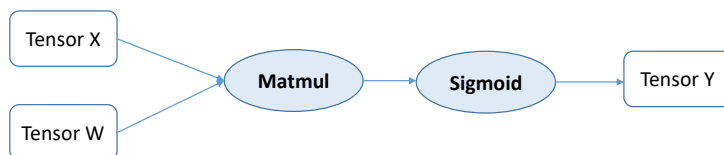
计算图

- 计算图提供了一种直观的方式来描述深度学习模型中的计算过程。每个节点代表一个计算操作（如加法、乘法、激活函数等），而边则表示数据在这些操作之间的流动。这种结构化的表示方法使得复杂的计算过程变得易于理解和分析。
 - 可以快捷地执行自动微分。
 - 方便寻找网络并行性和优化点。



计算图组成

- 计算图由基本数据结构张量 (Tensor) 和基本运算单元算子构成。
- 在计算图中通常使用节点来表示算子，节点间的有向边 (Directed Edge) 来表示张量状态，同时也描述了计算间的依赖关系。



- 节点代表变量或操作。输入节点表示输入数据或参数，而中间节点和输出节点表示计算结果。
- 边连接节点，表示数据流或计算依赖关系。每条边通常表示从一个操作的输出到另一个操作的输入。

从前端语言到计算图

- 计算图是用图论语言表示数学函数的一种方式，也是深度学习框架表达神经网络模型的统一方法。

```
class _DenseLayer(nn.Module):  
    def __init__(  
        self, num_input_features: int, growth_rate: int, bn_size:  
        int, drop_rate: float, memory_efficient: bool = False  
    ) -> None:  
        super().__init__()  
        self.norm1 = nn.Linear()  
        self.relu1 = nn.ReLU(inplace=True)  
        .....  
    def forward(self, x):  
        .....
```

前端编程语言、接口：
构建模型

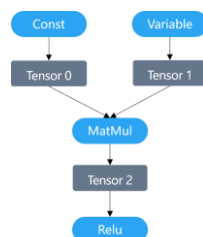
模型中间表示：
计算图 IR (DAG)

自动微分

图优化

优化、调度、编译

计算硬件：高效执行



动静态计算图

- 目前主流的深度学习框架可以生成静态图和动态图两种计算图。

	静态图	动态图
执行方式	程序在编译执行时先生成神经网络的图结构，然后再执行图中涉及的计算操作	程序按照代码的编写顺序执行，在执行正向过程中根据反向传播的原理，动态生成反向执行图
优点	编译器利用图优化等技术对执行图进行更大程度的优化，从而获得更好的执行性能	编译器将神经网络中的各个算子逐一下发执行，方便用户编写和调试神经网络模型
缺点	开发侧不友好，不易于调试	效率低，难于优化

动态计算图

- 动态计算图是在运行时构建的计算图，其主要特点包括：
 - 即时执行：操作是即时计算的，可以根据需要实时地改变图的结构。
 - 灵活性：能够轻松地处理可变大小的数据，并支持更复杂的控制流操作。
 - 易于调试：由于计算是按需进行的，开发者可以方便地使用Python等编程语言的调试工具进行错误检查。
 - 研究友好：研究者在进行实验时可以更灵活地尝试不同的想法，快速迭代模型。
 - 直观性：对于新手来说，动态计算图更容易理解和使用。
- 然而，动态计算图也存在一些局限性，如运行时可能引入额外的开销，导致性能不如静态计算图；由于图结构的动态性，优化可能不如静态计算图彻底。

静态计算图

- 静态计算图在运行之前就已经定义好，其主要特点包括：
 - 预先定义：在执行之前，整个计算图需要被定义好。
 - 优化：由于图结构是固定的，框架可以进行更多的优化，如内存分配和并行计算，从而提高运行时性能。
 - 部署友好：静态计算图适合用于生产环境和移动设备，因为它们通常有更高效率的运行时性能。
 - 可移植性：静态计算图可以被导出为独立于框架的格式，易于在不同的环境中部署。
- 但是，静态计算图在编写代码时通常更复杂，特别是涉及到复杂的控制流时；由于图是在运行前定义的，调试可能不如动态计算图直观。

静态图 - 以MindSpore为例

- 在MindSpore中，静态图模式又被称为Graph模式，可以通过set_context设置成静态图模式。
 - set_context(mode=GRAPH_MODE)，默认模式。
- 在Graph模式下，MindSpore通过源码转换的方式，将Python的源码转换成IR（MindIR），再在此基础上进行相关的图优化，最终在硬件设备上执行优化后的图。
 - 使用Graph模式时，需要使用nn.Cell类并且在construct函数中编写执行代码，或者调用@ms_function装饰器。
 - Graph模式是基于MindIR进行编译优化的。

- 静态图模式比较适合网络固定且需要高性能的场景。在静态图模式下，基于图优化、计算图整图下沉等技术，编译器可以针对图进行全局的优化，因此在静态图下能获得较好的性能，但是执行图是从源码转换而来，因此在静态图下不是所有的Python语法都能支持。
- ms_function是MindSpore中提供的用于加速动态图执行效率的工具。

MindSpore静态图使用

```
import numpy as np
import mindspore.nn as nn
import mindspore.ops as ops
import mindspore as ms

ms.set_context(mode=ms.GRAPH_MODE, device_target="CPU")

class Net(nn.Cell):
    def __init__(self):
        super(Net, self).__init__()
        self.mul = ops.Mul()
    def construct(self, x, y):
        return self.mul(x, y)

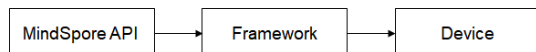
x = ms.Tensor(np.array([1.0, 2.0, 3.0]).astype(np.float32))
y = ms.Tensor(np.array([4.0, 5.0, 6.0]).astype(np.float32))

net = Net()
print(net(x, y)) # 在最终执行时才能获取输出
```

 [4. 10. 18.]

动态图 - 以MindSpore为例

- 在MindSpore中，动态图模式又被称为PyNative模式，可以通过set_context设置：
 - `set_context(mode=PYNATIVE_MODE)`。
- 在PyNative模式下，用户可以使用完整的Python API，此外针对使用MindSpore提供的API时，框架会根据用户选择的硬件平台（Ascend，GPU，CPU），将算子API的操作在对应的硬件平台上执行，并返回相应的结果。



- 通过前端的Python API，调用到框架层，最终到相应的硬件设备上计算。
- 在PyNative下执行正向过程完全是按照Python的语法进行执行。
- 拓展：在PyNative下是基于Tensor进行实现反向传播的，我们在执行正向过程中，将所有应用于Tensor的操作记录下来，并针对每个操作求取其反向，并将所有反向过程串联起来形成整体反向传播图（简称反向图）。最终，将反向图在设备上执行计算出梯度。

MindSpore动态图使用

```
import numpy as np
import mindspore.nn as nn
import mindspore as ms
import mindspore.ops as ops

ms.set_context(mode=ms.PYNATIVE_MODE, device_target="CPU")
x = ms.Tensor(np.ones([1, 3, 3, 4]).astype(np.float32)) # 创建一个张量
y = ms.Tensor(np.ones([1, 3, 3, 4]).astype(np.float32))
output = ops.add(x, y) # 相加
print(output.asnumpy())
```



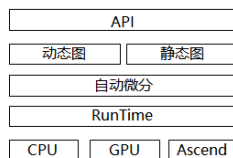
```
[[[2. 2. 2. 2.]
  [2. 2. 2. 2.]
  [2. 2. 2. 2.]
```

```
[[2. 2. 2. 2.]
 [2. 2. 2. 2.]
 [2. 2. 2. 2.]]
```

$$[[2. 2. 2. 2.]$$

动静统一

- 当前在业界支持动态图和静态图两种模式，动态图通过解释执行，具有动态语法亲和性，表达灵活；静态图使用jit编译优化执行，偏静态语法，在语法上有较多限制。动态图和静态图的编译流程不一致，语法约束不一致。
- MindSpore针对动态图和静态图模式，首先统一API表达，在两种模式下使用相同的API；其次统一动态图和静态图的底层微分机制。



- 在MindSpore中，我们可以通过控制模式输入参数来切换执行使用动态图还是静态图。但是，由于在静态图下，对于Python语法有所限制，因此从动态图切换到静态图时，需要符合静态图的语法限制，才能正确使用静态图来进行执行。

TensorFlow和PyTorch中的计算图

PyTorch	TensorFlow
默认为动态图模式	Tf2.x中默认为动态图， 也可以通过 <code>tf.enable_eager_execution()</code> 手动开启
可以通过 <code>torch.jit.trace</code> 和 <code>torch.jit.script</code> 函数实现静态图的效率提升	Tf1.x默认为静态图 Tf2.x可以通过 <code>tf.function</code> 实现静态图

目录

1. 人工智能框架
2. AI开发框架基础知识
- 3. PyTorch**
 - 基础模块介绍
 - 实现LeNet网络结构
4. MindSpore
5. AI应用开发流程

PyTorch常用模块

PyTorch常用模块	功能描述
torch	PyTorch的核心模块，提供了张量操作、线性代数等功能。
torch.nn	用于构建神经网络的模块，提供了神经网络层、损失函数和激活函数。
torch.utils.data	提供了数据加载和预处理的工具。
torch.autograd	PyTorch的自动微分引擎，支持梯度计算，主要用于在求得损失函数之后进行反向梯度传播。
torch.optim	提供了多种优化算法，用于模型的参数更新和优化。
torch.device	指定了张量和模型应该运行在哪个设备上，方便在CPU和GPU之间切换。
torch.distributions	提供了一系列类，使得PyTorch能够对不同的分布进行采样，并生成概率采样过程的计算图。
torch.jit	PyTorch的即时编译器模块，可以将动态图转换成可以优化和序列化的静态图。
torch.random	提供了一系列的方法来保存和设置随机数生成器的状态，有助于调试神经网络的结构和性能。
torch.onnx	定义了PyTorch导出和载入ONNX格式的深度学习模型描述文件，方便与其他深度学习框架交换模型。

torchvision

torchvision是PyTorch生态系统中专为计算机视觉设计的库

torchvision模块	功能描述
torchvision.datasets	提供了多种广泛使用的图像和视频数据集，如MNIST、CIFAR10/100、Fashion-MNIST、ImageNet、COCO等，这些数据集可以直接用于训练和评估图像分类、对象检测、语义分割等多种视觉模型。
torchvision.transforms	提供了丰富的数据增强和预处理操作，包括裁剪、旋转、翻转、归一化、调整大小、颜色转换等。这些操作对于训练稳健的深度学习模型至关重要，可以提高训练数据的质量和多样性。
torchvision.models	封装了大量预训练模型结构，例如AlexNet、VGG、ResNet、Inception系列、DenseNet、SqueezeNet等，以及用于目标检测和语义分割任务的模型。可以直接加载这些模型进行迁移学习或者作为基础网络结构进行扩展。
torchvision.utils	提供了一系列实用方法，比如将张量保存为图像文件、创建图像网格以便可视化多个样本等，有助于实验结果的可视化和分析。

数据加载

- PyTorch数据加载通过Dataset、DataLoader完成，Dataset定义好数据的格式和数据变换形式，DataLoader用iterative的方式不断读入批次数据。
 - torch.utils.data.Dataset是构造数据集的基类，通过继承这个类并重写__init__、__len__和__getitem__方法来提供自己的数据和标签。
 - torch.utils.data.DataLoader是一个用于加载数据集的迭代器，它提供了批量加载、打乱数据、多进程数据加载等功能。使用时需要传入一个Dataset对象。

```
#定义 一个Dataset，此处以pytorch中的开源数据为例
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
# 使用dataloader加载
trainloader = DataLoader(trainset, batch_size=64, shuffle=True)
```

其他构造Dataset的方式

- torchvision.datasets.ImageFolder：用于从文件夹中加载图像数据集的类。它假设所有的文件按文件夹保存，每个文件夹下的文件属于同一个类别。
- 其他开源数据集：

API名称	说明
torchvision.datasets.MNIST	用于加载MNIST手写数字数据集的类。用户只需指定数据的根目录和其他一些参数即可自动下载。
torchvision.datasets.CIFAR10	CIFAR10是torchvision库提供的另一个用于加载标准数据集（CIFAR-10）的类。这个数据集包含了60000张32x32的彩色图像，分为10个类别，每个类别有6000张图像。
torchvision.datasets.CIFAR100	CIFAR10类似，包含了100个类别，每个类别有600张图像。这个数据集也是用于图像分类任务的。
torchvision.datasets.ImageNet	ImageNet是一个大型图像数据集，包含了数以百万计的图像，涵盖了数千个类别。

模型创建

- `torch.nn.Module`是PyTorch中的模型构造类，是所有神经网络模块的基类，可以继承它来定义模型。
- 定义时需要重载`__init__`函数和`forward`函数，它们分别用于创建模型参数和定义前向计算（正向传播）。

```
import torch
from torch import nn

class MLP(nn.Module):
    # 声明带有模型参数的层，这里声明了两个全连接层
    def __init__(self, **kwargs):

        super(MLP, self).__init__(**kwargs)
        self.hidden = nn.Linear(784, 256)
        self.act = nn.ReLU()
        self.output = nn.Linear(256,10)
```

前向计算

```
class MLP(nn.Module):
    # 声明带有模型参数的层
    def __init__(self, **kwargs):
        .....

    # 模型的前向计算，即如何根据输入x计算返回所需要的模型输出
    def forward(self, x):
        o = self.act(self.hidden(x))
        return self.output(o)
```

- 无须定义反向传播函数。系统将通过自动求梯度而自动生成反向传播所需的backward函数。

二维卷积层

- `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)`:
 - 实现二维卷积操作的类。
 - `in_channels`: 输入图像的通道数。例如，对于彩色RGB图像，`in_channels` 为 3。
 - `out_channels`: 卷积操作后输出的通道数（卷积核数量）。
 - `kernel_size`: 卷积核的大小。
 - `stride`: 卷积时滑动的步长。
 - `padding`: 零填充的层数。

```
# 定义一个二维卷积层
conv_layer = nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, bias=True)

# 输入数据 (假设输入为 batch_size 个图像, in_channels 个通道, 大小为 height x width)
input_tensor = torch.randn(batch_size, in_channels, height, width)

# 通过卷积层
output_tensor = conv_layer(input_tensor)
```

最大池化

- `torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)`:
 - 实现二维最大池化操作。
 - `kernel_size` (int or tuple): 池化窗口的大小。
 - `stride` (int or tuple, optional): 池化时滑动的步长。默认为 `kernel_size`。
 - `padding` (int or tuple, optional): 输入特征图各边补充的零填充的层数。默认为 0。

```
conv_layer = nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, bias=True)

# 输入数据 (假设输入为 batch_size 个图像, in_channels 个通道, 大小为 height x width)
input_tensor = torch.randn(batch_size, in_channels, height, width)

# 通过卷积层
output_tensor = conv_layer(input_tensor)
```

循环神经网络

- `torch.nn.RNN(input_size, hidden_size, num_layers = 1, nonlinearity = 'tanh', bias = True, batch_first = False, dropout = 0.0, bidirectional = False, device = None, dtype = None)`:
 - 用于定义基础的循环神经网络。
 - `input_size`: 输入特征的维度。
 - `hidden_size`: 隐藏层特征的维度。
 - `num_layers`: RNN层的数量。
 - `bias`: 是否使用偏置。
 - `nonlinearity`: 非线性激活函数。可以是 'tanh' 或 'relu'。默认是 'tanh'。
 - `batch_first`: 如果为 `True`, 输入和输出的第一个维度将是 `batch size`。

```
# 定义RNN模型
rnn = nn.RNN(input_size=10, hidden_size=20, num_layers=1, batch_first=True)

# 创建输入数据
inputs = torch.randn(5, 3, 10) # 序列长度为5, batch_size大小为3, 输入特征维度为10
h0 = torch.zeros(1, 3, 20) # 初始隐藏状态, 形状为[num_layers, batch_size, hidden_size]
# 前向传播
output, hn = rnn(inputs, h0)
```

PyTorch激活函数

- PyTorch中使用激活函数：通过nn模块下的激活函数类和nn.functional下的激活函数。

激活函数	调用方法	调用方式说明
Sigmoid	nn.Sigmoid() F.sigmoid(input)	以Sigmoid为例： 通过nn.Sigmoid()构建Sigmoid层（实例化对象），然后调用该层； 或通过F.sigmoid(input)对输入input直接应用Sigmoid函数。 import torch import torch.nn as nn import torch.nn.functional as F # 创建一个输入张量 input_tensor = torch.tensor([[-1.0, 0.0, 1.0], [-2.0, -1.0, 0.0]]) # 创建一个Sigmoid层实例 sigmoid_layer = nn.Sigmoid() # 将输入数据传递给Sigmoid层 output_tensor = sigmoid_layer(input_tensor) # 直接对输入数据应用Sigmoid激活函数 output_tensor = F.sigmoid(input_tensor)
Tanh	nn.Tanh() F.tanh(input)	
ReLU	nn.ReLU() F.relu(input)	
LeakyReLU	nn.LeakyReLU(negative_slope=0.01) F.leaky_relu(input, negative_slope=0.01)	
ELU	nn.ELU() F.elu(input)	
PReLU	nn.PReLU() F.prelu(input)	
Softmax	nn.Softmax(dim=None) F.softmax(input, dim=None)	

损失函数

- PyTorch中提供了多种损失函数。

中文名称	调用方式	调用方式说明
L1损失（平均绝对误差）	<code>torch.nn.L1Loss(reduction='mean')</code>	计算预测值和真实值之间差的绝对值的平均值
均方误差损失	<code>torch.nn.MSELoss(reduction='mean')</code>	计算预测值和真实值之间差的平方的平均值
交叉熵损失	<code>torch.nn.CrossEntropyLoss(weight=None, reduction='mean')</code>	结合LogSoftMax和NLLLoss，用于多分类问题
二进制交叉熵损失	<code>torch.nn.BCELoss(weight=None, reduction='mean')</code>	计算目标值和预测值之间的二进制交叉熵，用于二分类问题
连接时序分类损失（CTCLoss）	<code>torch.nn.CTCLoss(blank=0, reduction='mean')</code>	对没有对齐的数据进行自动对齐，主要用于序列化数据训练

PyTorch中的优化器

优化器中文名称	调用方式	调用方式说明
SGD优化器	<code>torch.optim.SGD(params, lr=0.01, momentum=0, dampening=0, weight_decay=0, nesterov=False)</code>	params: 待优化参数; lr: 学习率; momentum: 动量因子, 用于加速SGD在相关方向上的收敛并抑制震荡; dampening: 动量的阻尼系数, 用于减小动量更新的幅度; weight_decay: 权重衰减系数, 用于L2正则化。
Adam优化器	<code>torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False)</code>	params: 待优化参数; lr: 学习率; betas: 用于计算梯度以及梯度平方的运行平均值的系数; weight_decay: 权重衰减系数。
Adagrad优化器	<code>torch.optim.Adagrad(params, lr=0.01, lr_decay=0, weight_decay=0, initial_accumulator_value=0, eps=1e-10)</code>	params: 待优化参数; lr: 学习率; lr_decay: 学习率衰减; weight_decay: 权重衰减系数; initial_accumulator_value: 累加器的初始值。
AdamW优化器	<code>torch.optim.AdamW(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0.01, amsgrad=False)</code>	params: 待优化参数; lr: 学习率; betas: 用于计算梯度以及梯度平方的运行平均值的系数; eps: 为了增加数值计算的稳定性而加到分母里的项; weight_decay: 权重衰减系数; amsgrad: 是否使用AMSGrad变体。
RMSprop优化器	<code>torch.optim.RMSprop(params, lr=0.001, alpha=0.99, eps=1e-08, weight_decay=0, momentum=0, centered=False)</code>	params: 待优化参数; lr: 学习率; alpha: 平滑项的系数; weight_decay: 权重衰减系数; momentum: 动量因子; centered: 是否计算中心化的RMSprop。

添加损失函数和优化器进行训练

```
# 实例化一个模型结构
model = MLP()
# 创建损失函数
criterion = nn.CrossEntropyLoss()
# 创建优化器
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
# 给模型输入数据
output = model(trainset)
# 计算损失
loss = criterion(output, target)
loss.backward()
optimizer.step()
```

目录

1. 人工智能框架
2. AI开发框架基础知识
- 3. PyTorch**
 - 基础模块介绍
 - 实现LeNet网络结构
4. MindSpore
5. AI应用开发流程

构造一个完整的神经网络 (1)

- 使用PyTorch构建一个完整网络LeNet，并使用MNIST进行训练。
 - 构建LeNet网络。
 - 包含两个卷积层:第一个卷积层使用6个5x5的卷积核，步长为1，填充为2，输出6个特征图。第二个卷积层使用16个5x5的卷积核，步长为1，没有填充，输出16个特征图。
 - 用平均池化层来减少特征图的维度。
 - 卷积层之后使用ReLU，全连接层之后使用sigmoid或softmax（在输出层）。

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms

# 定义LeNet网络结构
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # 定义二维卷积层
        self.conv1 = nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=2)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5, stride=1)
        # 定义全连接层
        self.fc1 = nn.Linear(16*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
        # 定义池化层
        self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        # 激活函数
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
```

构造一个完整的神经网络 (2)

```
Conv1 (第一个卷积层)
|---5x5卷积核, 6个, 步长为1
|---输出特征图: 28x28x6
Pool
|---2x2池化窗口, 步长为2
|---输出特征图: 14x14x6
Conv2 (第二个卷积层)
|---5x5卷积核, 16个, 步长为1
|---输出特征图: 10x10x16
Pool
|---2x2池化窗口, 步长为2
|---输出特征图: 5x5x16
Conv3 (第三个卷积层, 也可视为全连接层)
|---5x5卷积核, 120个, 步长为1
|---输出特征图: 1x1x120 (全连接)
FC1 (第一个全连接层)
|---输入: 120个神经元
|---输出: 84个神经元
FC2 (第二个全连接层/输出层)
|---输入: 84个神经元
|---输出: 10个神经元 (对应于10个数字类别)
OUTPUT (输出层)
```

定义LeNet前向计算过程。

```
# 定义LeNet网络结构
class LeNet(nn.Module):
    def __init__(self):
        .....

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.pool(x)
        x = self.conv2(x)
        x = self.sigmoid(x)
        x = self.pool(x)
        x = x.view(-1, 16*5*5)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        return x
```

构造一个完整的神经网络 (3)

- 定义dataset，数据为开源数据集MNIST。
- 定义数据预处理方式：通过transforms.Compose定义，定义后直接通过transform参数用于数据集。

```
# 数据预处理
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

# 加载MNIST数据集
trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = torchvision.datasets.MNIST(root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=1000, shuffle=False)
```

构造一个完整的神经网络 (4)

- 初始化网络、损失函数和优化器。
- 通过for循环执行训练。
 - 训练完成以后需要进行测试，此处省略测试过程。

```
# 初始化网络、损失函数和优化器
net = LeNet()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.9)
# 训练网络
for epoch in range(10): # 训练10个epoch
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data

        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    running_loss += loss.item()
    if i % 100 == 99: # 每100个batch打印一次损失
        print(f'Epoch {epoch+1}, Batch {i+1} loss: {running_loss/100:.3f}')
    running_loss = 0.0
```


构造一个完整的神经网络 (5)

- 保存训练好的模型。
 - 在PyTorch 中，保存和加载模型通常涉及两个主要部分：模型的状态字典（state dictionary）和模型的架构。模型的状态字典包含了模型参数（如权重和偏置），而模型的架构定义了模型的层次结构。

```
# 保存模型参数
torch.save(net.state_dict(), 'lenet_mnist.pth')

# 加载模型时需要重新实例化为模型对象
net = LeNet()
net.load_state_dict(torch.load('lenet_mnist.pth'))
net.eval() # 如果要进行推理，切换到 eval 模式
```

```
# 保存整个模型
torch.save(net, 'lenet_mnist.pth')

# 加载模型
net = torch.load('lenet_mnist.pth')
model.eval() # 如果要进行推理，切换到 eval 模式
```

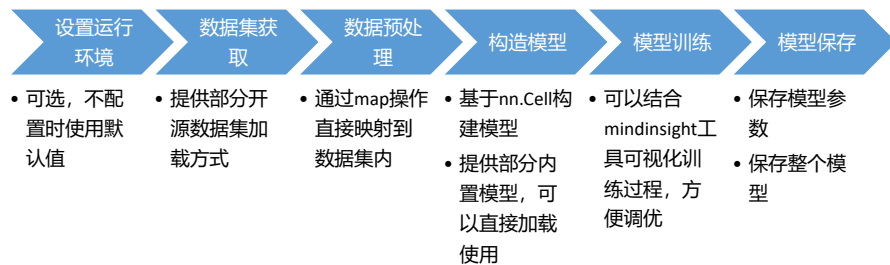
目录

1. 人工智能框架
2. AI开发框架基础知识
3. PyTorch
- 4. MindSpore**
5. AI应用开发流程

MindSpore

模块名	模块说明
mindspore	框架基础接口。
mindspore.nn	神经网络层，用于构建神经网络中的预定义构建块或计算单元。
mindspore.ops	函数接口。
mindspore.train	训练接口。
mindspore.common.initializer	参数初始化。
mindspore.dataset	加载和处理各种数据集的接口。
mindspore.dataset.transforms	通用数据变换。
mindspore.mindrecord	MindSpore开发的高效数据格式MindRecord相关的操作接口。
mindspore.nn.probability	可参数化的概率分布和采样函数。
mindspore.multiprocessing	多进程接口。

MindSpore使用流程



运行环境配置

- MindSpore在运行时需要指定运行时的具体环境参数，包括计算图的模式、运行时的设备、内存空间大小等，同时也可以使用API获取当前运行环境的具体信息。
- 运行环境相关API：
 - `mindspore.set_context`，设置运行环境的context。
 - `mindspore.get_context`，根据输入key获取context中的属性值。
 - `mindspore.ParallelMode`，并行模式。
 - `mindspore.set_ps_context`，设置参数服务器训练模式的上下文。
 -

Context配置项

- 可配置项包括当前执行环境，包括执行模式、执行后端和其他特性开关。

▫ `mindspore.set_context(**kwargs):`

参数	作用	取值
<code>device_id</code>	目标设备的ID	取值范围：[0, 4096-1]，默认0。
<code>device_target</code>	待运行的目标设备	取值范围：'Ascend'、'GPU'和'CPU'。
<code>mode</code>	运行模式选择	默认值：GRAPH_MODE。 (GRAPH_MODE/0或PYNATIVE_MODE/1)
<code>enable_sparse</code>	是否启用稀疏特征	默认值：False。
<code>save_graphs</code>	是否保存计算图	默认值：False。
<code>runtime_num_threads</code>	运行时线程池的线程数控制	默认值为30。

- 在运行程序之前，应配置context。如果没有配置，默认情况下将根据设备目标进行自动设置。

MindSpore数据处理模块

- 在MindSpore中提供了对于数据集处理的子模块dataset。
 - mindspore.dataset: 该模块提供了加载和处理各种通用数据集的API, 如MNIST、CIFAR-10、CIFAR-100、VOC、COCO、ImageNet、CelebA、CLUE等, 也支持加载业界标准格式的数据集, 包括MindRecord、TFRecord、Manifest等。此外, 用户还可以使用此模块定义和加载自己的数据集。
 - 此外, 该模块下还提供了针对于语音、文本、图像等数据的数据增强功能。

数据集获取

- 开源数据集调用API。

dataset.Cifar10Dataset	CIFAR-10数据集。
dataset.Cifar100Dataset	CIFAR-100数据集。
dataset.CityscapesDataset	Cityscapes数据集。
dataset.CocoDataset	COCO (Common Objects in Context) 数据集。
dataset.DIV2KDataset	DIV2K (DIverse 2K resolution image) 数据集。
dataset.EMNISTDataset	EMNIST (Extended MNIST) 数据集。
dataset.FakeImageDataset	生成虚假图像构建数据集。
dataset.FashionMnistDataset	Fashion-MNIST数据集。

- 其他构造数据集API。

dataset.FakeImageDataset	生成虚假图像构建数据集。
dataset.ImageFolderDataset	从树状结构的文件目录中读取图片构建源数据集。

- 除视觉数据集外dataset模块下还包括文本、语音、CSV等格式的数据集。
- 通过GeneratorDataset、RandomDataset等类，可以自定义数据集。

数据预处理

- mindspore.dataset.transforms, 此模块用于通用数据增强, 其中一部分增强操作是用C++实现的, 具有较好的高性能, 另一部分是基于Python实现, 使用了NumPy模块作为支持。

子模块	功能
mindspore.dataset.vision	用于图像数据增强。
mindspore.dataset.text	用于文本数据增强。
mindspore.dataset.audio	用于音频数据增强。
mindspore.dataset.transforms	通用数据增强。

```
import mindspore.dataset.vision as vision
vision.Normalize(mean, std) #根据均值和标准差对输入图像进行归一化
# std/mean 图像每个通道的标准差/均值组成的列表或元组。标准差/平均值必须在 [0.0, 255.0] 范围内

vision.HWC2CHW() #将输入图像的shape从 <H, W, C> 转换为 <C, H, W>。输入图像应为 3 通道图像

vision.CenterCrop(size) #对输入图像中心区域裁剪。输入图像尺寸小于输出尺寸, 则在裁剪前对输入图像边界填充0像素
```

数据预处理代码示例

场景：数据为图片数据集，共五个类别，分别为daisy、dandelion、roses、sunflowers、tulips，每个类别为一个文件夹名称，该文件夹下是同一类别的数据。

操作：读取数据，对图片数据做预处理操作，操作包含：解码为RGB格式数据、正则化、裁剪等。

```
def read_data(path, config, usage="train"):
    dataset = ds.ImageFolderDataset(path, class_indexing={'daisy':0, 'dandelion':1, 'roses':2, 'sunflowers':3, 'tulips':4})
    decode_op = vision.Decode() # 图像解码算子
    normalize_op = vision.Normalize(mean=[cfg._R_MEAN, cfg._G_MEAN, cfg._B_MEAN], std=[cfg._R_STD, cfg._G_STD, cfg._B_STD]) # 图像正则化算子
    resize_op = vision.Resize(cfg._RESIZE_SIDE_MIN) # 图像调整大小算子
    center_crop_op = vision.CenterCrop((cfg.HEIGHT, cfg.WIDTH)) # 图像裁剪算子
    horizontal_flip_op = vision.RandomHorizontalFlip() # 图像随机水平翻转算子
    channelswap_op = vision.HWC2CHW() # 图像通道数转换算子
    # 图像随机裁剪解码编码调整大小算子
    random_crop_decode_resize_op = vision.RandomCropDecodeResize((cfg.HEIGHT, cfg.WIDTH), (0.5, 1.0), (1.0, 1.0), max_attempts=100)
    .....
    dataset = dataset.repeat(1) # 数据增强
    dataset.map_model = 4
    return dataset
```

MindSpore构建网络

- mindspore.nn类是构建所有网络的基类，也是网络的基本单元。当用户需要自定义网络时，可以继承nn.Cell类，并重写__init__方法和construct方法。__init__包含所有网络层的定义，construct中包含数据（Tensor）的变换过程（前向计算，和PyTorch中nn.Module类似）。

```
# 代码实现参考实验手册
class Net(nn.Cell):
    def __init__(self, *args, **kwargs):
        super(ResNet, self).__init__()
        .....
        网络中的层
        .....
    def construct(self, x):
        .....
        前向计算
        .....
```

- MindSpore中神经网络的基本构成单元为nn.Cell。模型或神经网络层应当继承该基类。基类的成员函数**construct**是定义要执行的计算逻辑，所有继承类都必须重写此方法。

定义网络

- 构建一个两层全连接层的网络。
 - nn.SequentialCell，将按照传入顺序依次将Cell添加。用于模型结构较为复杂时。

```
# Define model
class Network(nn.Cell):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.dense_relu_sequential = nn.SequentialCell(
            nn.Dense(28*28, 512),
            nn.ReLU(),
            nn.Dense(512, 512),
            nn.ReLU(),
            nn.Dense(512, 10)
        )

    def construct(self, x):
        x = self.flatten(x)
        logits = self.dense_relu_sequential(x)
        return logits
```

网络结构API介绍 (1)

- mindspore.nn, 神经网络Cell, 用于构建神经网络中的预定义构建块或计算单元。
 - mindspore.nn.Cell, MindSpore的Cell类是构建所有网络的基类, 也是网络的基本单元。
 - mindspore.nn.LossBase, 损失函数的基类。
- 神经网络结构API:
 - mindspore.nn.Dense(in_channels, out_channels, weight_init='normal', bias_init='zeros', has_bias=True, activation=None): 全连接层。
 - in_channels (int), Dense层输入Tensor的空间维度。
 - out_channels (int), Dense层输出Tensor的空间维度。
 - weight_init, 权重参数的初始化方法。

- 在MindSpore中, 自定义网络结构、损失函数等可以通过继承对应的父类来快速创建。
- 具体函数及类的使用请参看MindSpore文档: <https://www.mindspore.cn/docs>。

网络结构API介绍 (2)

- `mindspore.nn.Conv2d`: (`in_channels`, `out_channels`, `kernel_size`, `stride=1`, `pad_mode="same"`, `padding=0`, `dilation=1`, `group=1`, `has_bias=False`, `weight_init="normal"`, `bias_init="zeros"`, `data_format="NCHW"`): 二维卷积层。
 - `kernel_size`, 指定二维卷积核的高度和宽度。
 - `stride`, 二维卷积核的移动步长。
 - `pad_mode`, 指定填充模式。可选值为`"same"`、`"valid"`、`"pad"`。默认值: `"same"`。
- `mindspore.nn.MaxPool2d`(`kernel_size=1`, `stride=1`, `pad_mode='valid'`, `data_format='NCHW'`): 二维最大池化层。

网络结构API介绍 (3)

- `mindspore.nn.RNN(*args, **kwargs)`: 循环神经网络 (RNN) 层, 其使用的激活函数为tanh或relu。
 - `input_size` (int), 输入层输入的特征向量维度。
 - `hidden_size` (int), 隐藏层输出的特征向量维度。
 - `num_layers` (int), 堆叠RNN的层数。默认值: 1。
- `mindspore.nn.Dropout(keep_prob=0.5, dtype=mstype.float32)`: 根据丢弃概率 $1 - \text{keep_prob}$, 在训练过程中随机将一些神经元输出设置为0。
 - `keep_prob` (float), 输入神经元保留率, 数值范围在0到1之间。
- `mindspore.nn.ReLU`: 修正线性单元激活函数。
- `mindspore.nn.Softmax(axis=-1)`: Softmax激活函数
 - `axis`, 指定Softmax运算的轴axis。

训练API介绍

- `mindspore.train.Model(network, loss_fn=None, optimizer=None, metrics=None, **kwargs)`:
Model 会根据用户传入的参数封装可训练或推理的实例。
 - `network (Cell)`, 用于训练或推理的神经网络。
 - `loss_fn (Cell)`, 损失函数。
 - `optimizer (Cell)`, 用于更新网络权重的优化器。
 - `metrics (Union[dict, set])`, 用于模型评估的一组评价函数。

模型保存与加载

- 在训练完网络完成后，将网络模型以文件的形式保存下来。保存模型的接口主要有2种：
 - 简单的对网络模型进行保存，可以在训练前后进行保存。

```
import mindspore as ms
# 使用MindSpore提供的save_checkpoint保存模型，传入网络和保存路径
# 定义的网络模型为net，一般在训练前或者训练后使用
ms.save_checkpoint(network, "/MyNet.ckpt")
```

- 在网络模型训练中进行保存，自动保存训练时候设定好的epoch数和step数的参数，也就是把模型训练过程中产生的中间权重参数也保存下来，方便进行网络微调和停止训练。

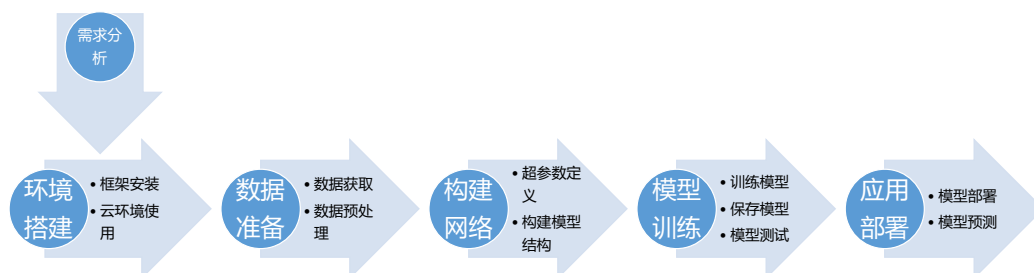
```
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig
# 设置epoch_num数量
epoch_num = 5
# 设置模型保存参数
config_ck = CheckpointConfig(save_checkpoint_steps=1875, keep_checkpoint_max=10)
# 应用模型保存参数
ckpt = ModelCheckpoint(prefix="lenet", directory="/lenet", config=config_ck)
model.train(epoch_num, dataset_train, callbacks=[ckpt])
```

- 训练网络模型的过程中，实际上我们希望保存中间和最后的结果，用于微调（fine-tune）和后续的模型部署和推理。且当模型过于庞大时，需要边训练边保存。
- 第二种保存方式首先需要初始化一个CheckpointConfig类对象，用来设置保存策略。
 - save_checkpoint_steps表示每隔多少个step保存一次。
 - keep_checkpoint_max表示最多保留CheckPoint文件的数量。
 - prefix表示生成CheckPoint文件的前缀名。
 - directory表示存放文件的目录。
- 创建一个ModelCheckpoint对象把它传递给model.train方法，就可以在训练过程中使用CheckPoint功能了。

目录

1. 人工智能框架
2. AI开发框架基础知识
3. PyTorch
4. MindSpore
- 5. AI应用开发流程**

AI应用开发流程



ResNet50图像分类应用

- 图像分类是最基础的计算机视觉应用，属于有监督学习类别，如给定一张图像（猫、狗、飞机、汽车等），判断图像所属的类别。本应用将介绍使用ResNet50网络对花卉数据集进行分类。

这是什么花？



环境搭建

- 在选择适合自己的环境条件（此处为Python3）后，获取命令并按照指南进行安装。
 - 后续以PyTorch为例，MindSpore实现流程相似（两种框架的具体实现参考实验手册）。

版本	2.4.1	2.3.1	master (Nightly build)
硬件平台	Ascend	CPU	GPU CUDA 10.1 GPU CUDA 11.1 GPU CUDA 11.6
操作系统	Linux-aarch64	Linux-x86_64	Windows-x64 MacOS-aarch64 MacOS-x86_64
编程语言	Python 3.8	Python 3.10	Python 3.11
安装方式	Pip	Conda	Source Docker
安装命令	<pre>pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/2.4.1/MindSpore/unified/aarch64/mindspore-2.4.1-cp38-cp38-linux_aarch64.whl --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple # 注意参考下方安装指南，添加运行所需的环境变量配置</pre>		

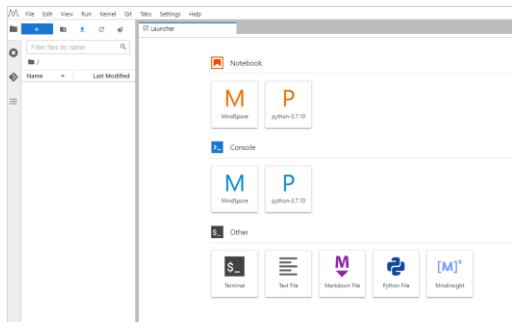
<https://www.mindspore.cn/install>

PyTorch Build	Stable (2.5.1)	Preview (Nightly)			
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python	C++ / Java			
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	ROCm v2	CPU
Run this Command:	<pre>pip3 install torch torchvision torchaudio</pre>				

<https://PyTorch.org/get-started/locally/>

云环境准备 (可选)

- 若当前无可用开发环境，可以使用云平台ModelArts进行开发、模型训练或部署。
 - ModelArts帮助用户快速创建和部署模型，管理全周期AI工作流。



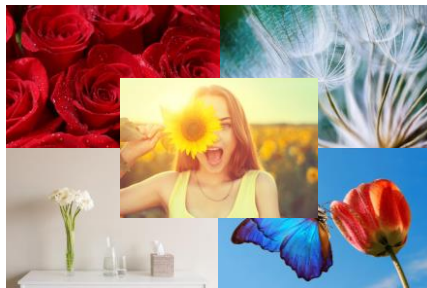
- 云环境相较于本地环境提供了充足的算力和存储空间，是开发过程中一个较好的选择。

数据准备

- 训练数据使用的是图像花卉数据集，该数据集是开源数据集，总共包括5种花的类型：分别是daisy（雏菊，633张），dandelion（蒲公英，898张），roses（玫瑰，641张），sunflowers（向日葵，699张），tulips（郁金香，799张），保存在5个文件夹当中，总共3670张。在这里分成了flower_photos_train和flower_photos_test两部分。

```
flower_photos_train-  
├─ daisy-  
├─ dandelion-  
├─ roses-  
├─ sunflowers-  
├─ tulips-  
└─ LICENSE.txt-  
flower_photos_test-  
├─ daisy-  
├─ dandelion-  
├─ roses-  
├─ sunflowers-  
├─ tulips-  
└─ LICENSE.txt-
```

文件结构



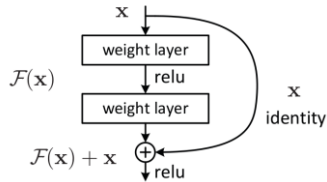
数据预处理

- 为了提升模型的准确率及保证模型的泛化能力，在使用数据训练模型前，通常会对数据做数据增强、标准化等操作。
- 数据集的加载及处理，包括以下功能：
 - 读取数据集。
 - 定义进行数据增强和处理所需要的一些参数。
 - 根据参数，生成对应的数据增强操作。
 - 对生成的数据集进行处理。

- 此次应用所使用的是开源数据集，所以不需要对数据集本身做太多处理。如果使用的数据是收集的相关业务数据，还需要对数据做清洗和整理等操作。

ResNet网络介绍

- ResNet50网络是2015年由微软实验室的何恺明提出，获得ILSVRC2015图像分类竞赛第一名。
- 残差网络结构（Residual Network）是ResNet网络的主要亮点，ResNet使用残差网络结构后可有效地减轻退化问题，实现更深的网络结构设计。



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
3×3 max pool, stride 2						
conv2.x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3.x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4.x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5.x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
1×1						
average pool, 1000-d fc, softmax						
FLOPs		1.8×10 ⁹	3.6×10 ⁹	3.8×10 ⁹	7.6×10 ⁹	11.3×10 ⁹

模型训练

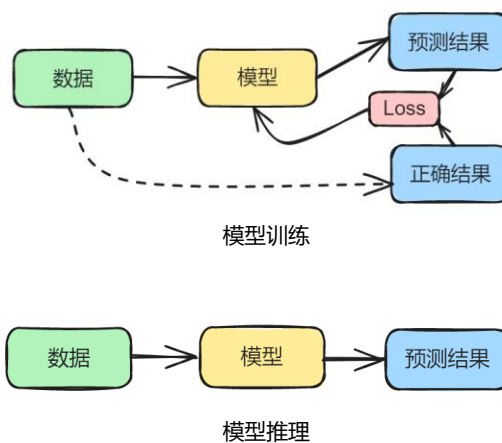
- 完成数据预处理、网络模型构建等步骤之后，开始模型训练。
- 模型训练可以简单的分为两种形式：
 - 基于自身数据集从零开始训练（PyTorch实验使用）；
 - 基于一个训练好的模型，在自身数据集上进行微调训练（MindSpore实验使用）。
 - 预训练模型：ImageNet数据集上训练好的ResNet模型文件。
 - 修改预训练模型参数的最后一层参数（预训练模型是在ImageNet数据集上训练的1001分类任务，而任务是实现flowers数据集的5分类任务）。
 - 基于自身训练集进行训练。
- 调参：训练时可以通过调整不同的超参数组合。
- 代码参考实验手册。

模型加载与预测

- 通过训练好的模型权重，对测试数据进行测试。
 - PyTorch训练完成后在禁用梯度计算的情况下进行推理测试（`torch.no_grad`）。
 - 如果在部署后推理的情况下，需要先开启eval模式（`model.eval()`）。
 - MindSpore训练完成后直接调用`model.predict`预测。
 - 加载权重文件后进行预测。
 - 要加载模型权重，需要先创建相同模型的实例。

模型部署

- 右图为简化后的模型训练与推理的流程，可以发现模型推理无需更新模型权重参数，因此对硬件的要求相对训练较低；
- 模型部署通常会部署在专用的推理服务器上，经过模型压缩等优化方法，也可以将模型部署在手机、摄像头等端边设备上。



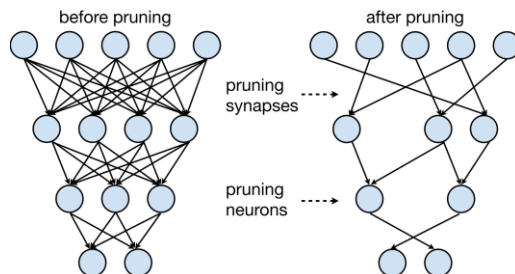
模型压缩

- 随着模型参数数量的增加，模型部署的成本和推理性能都受影响，部分影响如下：
 - 推理性能
 - 硬件访存带宽受限时，权重越大，数据搬运开销越大；
 - 显存容量不足时，只能CPU部署、推理性能较差；
 - 部署成本：
 - 权重越多，对存储容量、计算资源开销、显存容量要求就越高，需要更多的设备来部署；
- 目前大显存、访存带宽高的设备价格都较高，为了能在消费级设备上部署、推广AI应用，模型压缩必不可少。

- Batch Size、序列长度的增加，KV Cache也会不断上升，对访存也会造成较大压力。

模型压缩方法 - 模型剪枝

- 模型剪枝是模型压缩中一种重要的技术，其基本思想是将模型中不重要的权重和分支裁掉，将网络结构稀疏化，进而得到参数量更小的模型。然而，剪枝也可能导致模型性能的下降，因此需要在模型大小和性能之间找到一个平衡点。神经元在神经网络中的连接在数学上表示为权重矩阵，因此剪枝即是将权重矩阵中一部分元素变为零元素。剪枝过程如下图所示，目的是减去不重要的突触（Synapses）或神经元（Neurons）。



模型压缩方法 - 知识蒸馏

- 知识蒸馏（Knowledge Distillation），是一种带有迁移学习理念的模型压缩技术。
 - 较大、较复杂的网络虽然通常具有很好的性能，但是也存在很多的冗余信息，因此运算量以及资源的消耗都非常多。而所谓的Distilling就是将复杂网络中的有用信息提取出来迁移到一个更小的网络上，这样学习来的小网络可以具备和大的复杂网络相接近的性能效果，并且也大大的节省了计算资源。这个复杂的网络可以看成是一个教师，而小的网络则可以看成是一个学生。

常用部署工具介绍

- MindFormers
 - 基于MindSpore框架，支持Transformer类模型的微调与推理。
- MindSpeed-LLM/MM
 - 基于PyTorch框架，底层对接MindSpeed，分别支持大语言模型与多模态大模型的微调与推理。
- MindIE
 - 分层开放AI能力，向上支持多种主流AI框架（MindSpore、PyTorch），向下对接不同类型的昇腾AI处理器。
- Llama.cpp
 - 使用C++代码实现，目标是在各种硬件（本地和云中）上以最少的设置和最先进的性能实现LLM推理，无需AI框架。

本章总结

- 本章节主要介绍了AI开发框架的相关知识，包括了常用的AI框架、MindSpore框架、PyTorch框架的基础知识和AI应用开发流程。

思考题

1. （多选）以下哪些硬件设备可以用于MindSpore训练？
- A. CPU
 - B. GPU
 - C. NPU
 - D. CUDA

- 答案：ABC

学习推荐

- <https://www.mindspore.cn/>。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



华为人工智能平台



前言

- 本章主要讲述了华为昇腾基础软硬件及华为云的AI能力，包括基于达芬奇架构的昇腾AI处理器、昇腾大模型解决方案、华为云的AI服务等相关内容。

目标

- 学完本课程后，您将能够：
 - 了解昇腾芯片的硬件架构；
 - 了解昇腾AI基础软硬件；
 - 了解昇腾大模型解决方案的优势。

目录

1. 昇腾AI基础软硬件平台

- AI芯片概览
- 昇腾芯片硬件架构
- 昇腾AI基础硬件
- 昇腾AI基础软件

2. 华为云：AI Ready

AI芯片的定义

- AI任务中99%以上的运算都是矩阵运算。
- **AI芯片：**
 - 广义的AI芯片：专门用于处理人工智能应用中大量计算任务的模块，即面向人工智能领域的芯片均被称为AI芯片；
 - 狭义的AI芯片：针对人工智能算法做了特殊加速设计的芯片，也被称为AI加速器，即专门用于处理人工智能应用中的大量计算任务的功能模块。
 - An AI accelerator is a class of specialized hardware accelerator or computer system designed to accelerate artificial intelligence and machine learning applications, including artificial neural networks and machine vision. Typical applications include algorithms for robotics, internet of things, and other data-intensive or sensor-driven tasks. They are often manycore designs and generally focus on low-precision arithmetic, novel dataflow architectures or in-memory computing capability.

--维基百科

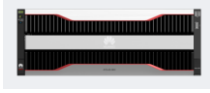
- 现在的AI模型中包含大量的矩阵运算，AI专用芯片都是对矩阵乘法做了优化，通过引入各种处理单元来高效进行矩阵运算。

AI芯片分类

- 从业务应用来看，可以分为Training（训练）和Inference（推理）两个类型：
 - Training环节通常需要通过大量的数据输入，或采取增强学习等非监督学习方法，训练出一个复杂的神经网络模型。训练过程涉及海量的训练数据和复杂的神经网络结构，运算量巨大，需要庞大的计算规模，对于处理器的计算能力、精度、可扩展性等性能要求很高。常用的例如华为的Atlas900集群、NVIDIA的GPU集群等。
 - Inference环节指利用训练好的模型，使用新的数据去“推理”出各种结论，如视频监控设备通过后台的神经网络模型，判断一张抓拍到的人脸是否属于特定的目标。虽然Inference的计算量相比Training少很多，但仍然涉及大量的矩阵运算。在推理环节，CPU、NPU、GPU和FPGA都有很多应用价值。

- 训练芯片需要考虑的因素更多，设计上也更加复杂，精度通常为FP32、FP16
- 推理芯片考虑的因素较少，对精度要求也不高，INT8即可

AI芯片应用领域



云端训练

- **芯片特征：**高功耗、高吞吐量、高精确率、分布式、可扩展性、高内存与带宽。
- **应用：**云/HPC/数据中心



云端推理

- **芯片特征：**低功耗、高吞吐量、高精确率、分布式、可扩展性、低延时。
- **应用：**云/HPC/数据中心

边缘计算



- **芯片特征：**低功耗、低延时、可单独部署或与其他设备组合、可将多个终端用户进行虚拟化、较小的机架空间
- **应用：**智能制造、智慧家居、智慧交通、智慧金融等众多领域

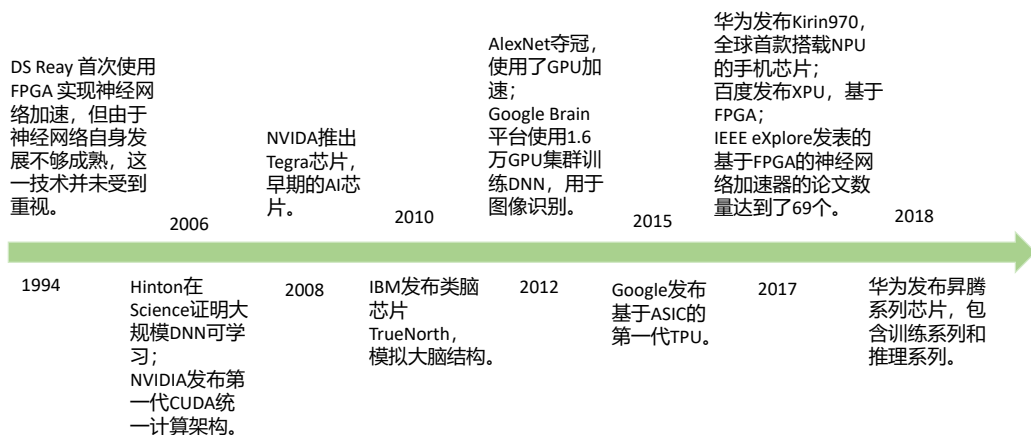
终端设备



- **芯片特征：**超低功耗、高能效、推理任务为主、较低的吞吐量、低延迟、成本敏感
- **应用：**各类消费电子，产品形态多样；物联网领域

- 云端芯片部署在专业机房，对环境要求不高；边缘计算通常部署的户外，需要适应高温和低温环境；终端设备主要考虑功耗和成本

AI芯片发展进程



- 现在芯片领域主流的为 GPU、FPGA、ASIC，后两者衍生出了 XPU、TPU 等一系列芯片

AI芯片技术路线

技术架构种类	定制化程度	可编辑性	算力	价格	优点	缺点	应用场景
GPU	通用型	不可编辑	中	高	通用性较强且适合大规模并行计算；设计和制造工艺成熟。	并行运算能力在推理端无法完全发挥。	高级复杂算法和通用性人工智能平台。
FPGA	半定制化	容易编辑	高	中	可通过编程灵活配置芯片架构适应算法迭代，平均性能较高；功耗较低；开发周期较短。	量产单价高；峰值计算能力较低；硬件编程困难。	适用于各种具体的行业。
ASIC	全定制化	难以编辑	高	低	通过算法固化实现极致的性能和能效，平均性能很强；功耗很低；体积小；量产后成本低。	前期投入成本高；研发时间长；技术风险大。	当客户处在某个特殊场景，可以为其独立设计一套专业智能算法软件。
类脑芯片	模拟人脑	不可编辑	高	-	功耗低；通信效率高；认知能力强。	目前仍处于探索阶段。	适用于各种具体的行业。

- 目前这几个技术方向都在发力，不存在某个技术方向被淘汰

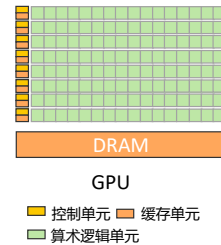
通用计算 vs. AI计算：“分工”不同，共建多样性计算

		硬件结构抽象	计算特点	适用场景
通用计算 以CPU提供算力	鲲鹏	Control 控制单元 Cache 缓存单元 ALU 控制单元 缓存单元 算术逻辑单元	适合复杂逻辑运算 比如大多数通用软件 70%以上晶体管用于构建Cache和控制单元，计算核心从几个到几十个。	通用应用 办公、数据库、数值计算（气象预测、流体仿真、电磁仿真）等。
AI计算 以GPU提供算力	NVIDIA AMD	TENSOR CORE 控制单元 缓存单元 算术逻辑单元	适合逻辑简单 计算密集型高并发任务 70%以上晶体管用于构建计算单元，	特定应用 图像识别：车牌识别、目标识别、目标检测； 自然语言处理：机器翻译、文本生成； 语言处理：语音识别、语音合成； 搜索推荐、辅助驾驶、趋势预测等。
AI计算 以NPU提供算力	昇腾	达芬奇 CORE 控制单元 缓存单元 算术逻辑单元	计算核心几千或上万个。	

- CPU考虑通用计算，GPU考虑数学计算，NPU考虑AI需要的矩阵计算

CPU、GPU设计比对（1）

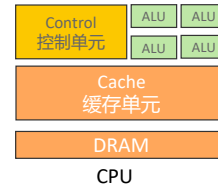
- GPU主要面对类型高度统一、相互无依赖的大规模数据和不需打断的纯净计算环境。
 - 拥有若干由数以千计的更小的核心（专为同时处理多重任务而设计）组成的大规模并行计算架构；
 - 基于大吞吐量设计；
 - 有很多ALU和很少cache（和CPU目的不同，为thread提高服务），缓存合并访问DRAM，带来时延问题；
 - 控制单元合并访问；
 - 大量ALU实现大量thread并行掩盖时延问题。
 - 擅长计算密集和易于并行的程序。



CPU、GPU设计比对（2）

- CPU需要很强通用性处理不同数据类型，同时需要逻辑判断，还会引入大量分支跳转和中断处理。

- 由专为串行处理而优化的几个核心组成；
- 基于低延时设计；
 - 强大的ALU单元，可在很短时钟周期完成计算；
 - 大量缓存降低延时；
 - 高时钟频率；
 - 复杂逻辑控制单元，多分支程序可通过分支预测能力降低时延；
 - 对于依赖之前指令结果的部分指令，逻辑单元决定指令在pipeline中的位置实现数据快速转发。
- 擅长逻辑控制、串行运算。



CPU、GPU设计演进

- CPU
 - 增加核数提升性能；
 - 增加指令（修改架构）的方式提升AI性能；
 - Intel（CISC架构）加入AVX512等指令，在ALU计算模块加入矢量运算模块（FMA）；
 - ARM（RISC架构）加入Cortex A等指令集并计划持续升级。
 - 提高频率提升性能；
 - 增加专用NPU单元。
- GPU
 - 增加AI专用计算单元（Tensor Core、Matrix Core）。

- 对于CPU和GPU，比较有效能够提高AI算力的方法都是加入AI专用计算模块

FPGA、ASIC比对

	FPGA	ASIC
运算速度	较低，FPGA结构上的通用性必然导致冗余；另外，不同结构间的时延也不可忽略。	较高，结构上无特殊限制，设计时也可将特定模块靠近减少延迟。
芯片规模	实现相同的功能时，需要更大的FPGA。	实现相同的功能时，ASIC的规模更小。
功耗	相同工艺条件下，功耗更大。	相同工艺条件下，功耗更小。
成本	研发风险较低，主要成本在单片上。	由于进入生产后硬件不可更改，开发工具和流片过程可能产生大量成本。
运行过程	加载配置进入存储器需要时间。	可立即运行。
产品定位	适用于项目产品需要灵活变动等方面的产品及产品要求快速占领市场的情况。	适用于设计规模较大，或应用于成熟的产品：如消费电子等。
发展方向	大容量、低电压、低功耗、SOC。	更大规模、IP复用技术、SOC。

- 两者目前发力的领域不同，并无明显的竞争关系

达芬奇架构：为AI计算而生，将AI计算效率做到极致

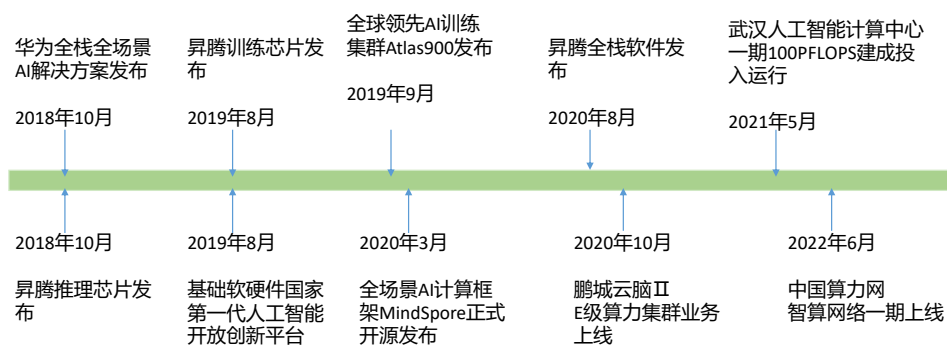
GPU 兼顾图像渲染与AI计算，Tensor Core单元占比低

昇腾NPU 专为AI计算设计，Cube占比高



- 达芬奇架构的核心就是CUBE计算单元

华为人工智能产业投入



- 科技部在2019世界人工智能大会宣布，将依托华为建设基础软硬件国家新一代人工智能开放创新平台，面向各行业、初创公司、高校和科研机构等的AI应用与研究，以云服务和产品软硬件组合的方式，提供全流程、普惠的基础平台类服务。

目录

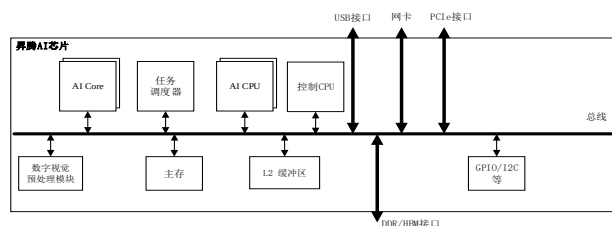
1. 昇腾AI基础软硬件平台

- AI芯片概览
- 昇腾芯片硬件架构
- 昇腾AI基础硬件
- 昇腾AI基础软件

2. 华为云：AI Ready

昇腾AI处理器逻辑架构

- 昇腾AI处理器的主要架构组成：
 - 芯片系统控制CPU（Control CPU）
 - AI计算引擎（包括AI Core和AI CPU）
 - 多层级的片上系统缓存（Cache）或缓冲区（Buffer）
 - 数字视觉预处理模块（Digital Vision Pre-Processing, DVPP）等



- 不同系列的芯片结构会略有差异

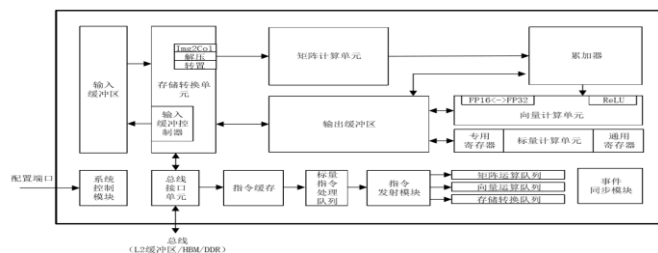
昇腾AI计算引擎—达芬奇架构

- 昇腾AI处理器的四大主要架构之一是AI计算引擎，AI计算引擎有包含了AI core（达芬奇架构）和AI CPU。因此达芬奇架构这一专门为AI算力提升所研发的架构也就是昇腾AI计算引擎和AI处理器的核心所在。

- AI Core负责高效矩阵运算，AI CPU负责通用运算

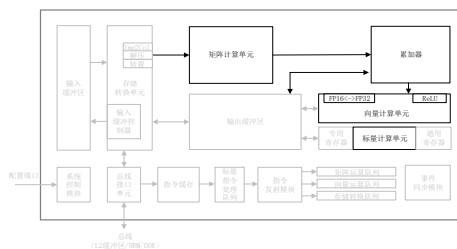
达芬奇架构 (AI Core)

- 达芬奇架构主要部分：
 - 计算单元：包含三种基础计算资源（矩阵计算单元、向量计算单元、标量计算单元）
 - 存储系统：AI Core的片上存储单元和相应的数据通路构成了存储系统。
 - 控制单元：整个计算过程提供了指令控制，相当于AI Core的司令部，负责整个AI Core的运行。



达芬奇架构 (AI Core) - 计算单元

- 三种基础计算资源：Cube Unit、Vector Unit和Scalar Unit，分别对应矩阵、向量和标量三种常见的计算模式。
 - 矩阵计算单元：矩阵计算单元和累加器主要完成矩阵相关运算。一拍完成一个fp16的 16×16 与 16×16 矩阵乘（4096）；如果是int8输入，则一拍完成 16×32 与 32×16 矩阵乘（8192）；
 - 向量计算单元：实现向量和标量，或双向量之间的计算，功能覆盖各种基本的计算类型和许多定制的计算类型，主要包括FP16/FP32/Int32/Int8等数据类型的计算；
 - 标量计算单元：相当于一个微型CPU，控制整个AI Core的运行，完成整个程序的循环控制、分支判断，可以为Cube/Vector提供数据地址和相关参数的计算，以及基本的算术运算。

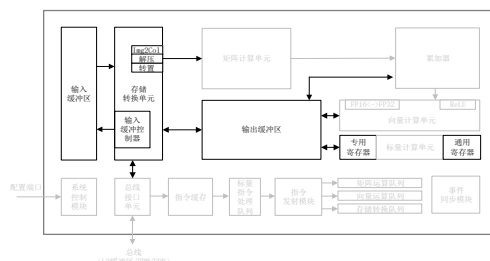


达芬奇架构（AI Core） - 存储系统（1）

- 存储系统由存储单元和相应的数据通路，构成了AI Core的存储系统
- 存储单元由存储控制单元、缓冲区和寄存器组成：
 - 存储控制单元：通过总线接口直接访问AI Core之外的更低层级的缓存，也可以直通到DDR或HBM直接访问内存。其中还设置了存储转换单元，作为AI Core内部数据通路的传输控制器，负责AI Core内部数据在不同缓冲区之间的读写管理，以及完成一系列的格式转换操作，如补零，Img2Col，转置、解压缩等；
 - 输入缓冲区：用来暂时保留需要频繁重复使用的数据，不需要每次都通过总线接口到AI Core的外部读取，从而在减少总线上数据访问频次的同时也降低了总线上产生拥堵的风险，达到节省功耗、提高性能的效果；
 - 输出缓冲区：用来存放神经网络中每层计算的中间结果，从而在进入下一层计算是方便的获取数据。相比通过总线读取数据的带宽低，延迟大，通过输出缓冲区可以大大提升计算效率；
 - 寄存器： AI Core中的各类寄存器资源主要是标量计算单元在使用。

达芬奇架构(AI Core) - 存储系统(2)

- 数据通路：是指AI Core在完成一次计算任务时，数据在AI Core中的流通过程。
 - 达芬奇架构数据通路的特点是多进单出，主要是考虑到神经网络在计算过程中，输入的数据种类繁多并且数量巨大，可以通过并行输入的方式来提高数据流入的效率。与此相反，将多种输入数据处理完成后往往只生成输出特征矩阵，数据种类相对单一，单输出的数据通路，可以节约芯片硬件资源。

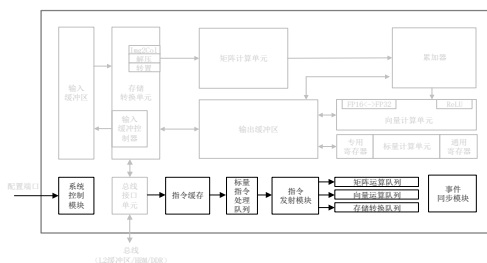


达芬奇架构（AI Core）- 控制单元（1）

- 控制单元主要组成部分为系统控制模块、指令缓存、标量指令处理队列、指令发射模块、矩阵运算队列、向量运算队列、存储转换队列和事件同步模块。
 - 系统控制模块：控制任务块（AI Core最小任务粒度）的执行进程，在任务块执行完成后，系统控制模块会进行中断处理和状态申报。如果执行过程出错，会把执行的错误状态报告给任务调度器；
 - 指令缓存：在指令执行过程中，可以提前预取后续指令，并一次读入多条指令进入缓存，提升指令执行效率；
 - 标量指令处理队列：指令被解码后便会被导入标量队列中，实现地址解码与运算控制，这些指令包括矩阵计算指令、向量计算指令以及存储转换指令等；
 - 指令发射模块：读取标量指令队列中配置好的指令地址和参数解码，然后根据指令类型分别发送到对应的指令执行队列中，而标量指令会驻留在标量指令处理队列中进行后续执行；

达芬奇架构 (AI Core) - 控制单元 (2)

- 指令执行队列：指令执行队列由矩阵运算队列、向量运算队列和存储转换队列组成，不同的指令进入相应的运算队列，队列中的指令按进入顺序执行；
- 事件同步模块：时刻控制每条指令流水线的执行状态，并分析不同流水线的依赖关系，从而解决指令流水线之间的数据依赖和同步的问题。



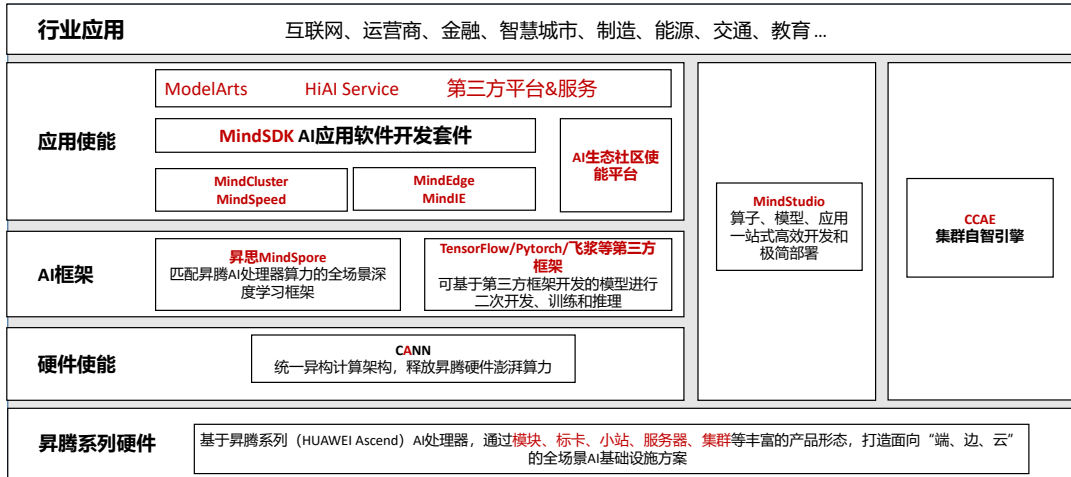
目录

1. 昇腾AI基础软硬件平台

- AI芯片概览
- 昇腾芯片硬件架构
- 昇腾AI基础硬件
- 昇腾AI基础软件

2. 华为云：AI Ready

昇腾全栈 AI 软硬件平台 构筑智能世界的基石



Atlas 200I A2 加速模块

- Atlas 200I A2 加速模块可以在边端侧实现目标识别、图像分类等AI应用加速，广泛用于智能边缘设备、机器人、无人机、智能工控等边端侧AI场景。

嵌入边缘设备，使能智能边缘



高度集成

- 单模块提供CPU、AI计算、视频图片编解码、ISP、图形输出等功能
- 支持直出多路PCIe / 以太网 / MIPI / SATA / USB / 音频等接口

超强性能

- 最大40路1080P 30FPS视频分析能力
- 最大20 TOPS INT8 算力

Atlas 300V Pro 视频解析卡

- **Atlas 300V Pro 视频解析卡**融合“通用处理器、AI Core、编解码”于一体，提供超强AI推理、视频图片编解码等功能，具有超大视频解析路数、高性能特征检索、安全启动等优势，支持128路高清视频实时分析，可广泛应用于智慧城市、智慧交通、智慧园区等诸多AI行业场景。



集成于服务器中，进行AI推理



超大视频解析路数

- 支持128路高清视频实时分析；
- 支持JPEG和视频硬件编解码，提升图片和视频类应用性能。

安全启动

- 设备启动链完整，启动初始状态确定，防止后端植入。

Atlas 300I Pro 推理卡

- **Atlas 300I Pro 推理卡**融合“通用处理器、AI Core、编解码”于一体，提供超强AI推理、目标检索等功能，具有超强算力、超高能效、高性能特征检索、安全启动等优势，可广泛应用于OCR识别、语音分析、搜索推荐、内容审核等诸多AI应用场景。



超强算力

- 为数据中心推理提供。

更强大支持

- 支持多核CPU计算能力。

超高能效

- 提供超高能效比，达到业界2.1倍。

安全启动

- 设备启动链完整，启动初始状态确定，防止后端植入。

Atlas 300I Duo 推理卡

- Atlas 300I Duo 推理卡融合“通用处理器、AI Core、编解码”于一体，提供AI推理、视频分析等功能，具有超强算力、超高能效、高性能视频分析等优势，可广泛应用于互联网、智慧城市、智慧交通等多场景，支持检索聚类、内容审核、OCR识别、语音分析、视频分析等多应用。



超强算力

超高能效

高性能视频分析

- 支持256路高清视频实时分析；
- 支持JPEG和视频硬件编解码，提升图片和视频类应用性能。

Atlas 500 A2 智能小站

- Atlas 500 A2智能小站是面向边缘应用的产品，具有环境适应性强、超强计算性能、云边协同等特点，可以在边缘环境广泛部署，满足在交通、社区、园区、商场、超市等复杂环境区域的应用需求。



边缘侧独立部署，使能智能边缘



环境适应性强

- 业界领先的集成AI处理能力的边缘产品
- 无风扇散热，支持-40℃至60℃室外工作

防护等级：IP40

超强计算性能

- 机顶盒大小即支持20 TOPS INT8算力
- 支持40路1080P 30FPS视频分析能力

云边协同

- 可选配4G或5G模块支持无线传输
- 云边协同，模型远程自动更新
- 可在云端统一进行设备管理和固件升级

Atlas 500 Pro 智能边缘服务器

- **Atlas 500 Pro 智能边缘服务器**（型号：3000）是面向边缘应用的产品，具有超强计算性能、高环境适应性、易于部署维护和支持云边协同等特点。可以在边缘场景中广泛部署，满足在安防、交通、社区、园区、商场、超市等复杂环境区域的应用需求。



边缘侧独立部署，使能智能边缘



超强算力

- 最大支持3张Atlas 300VI/V Pro推理卡，满足多场景推理需求；整机可提供384路高清视频实时分析（1080P 30 FPS）；
- 搭载鲲鹏920处理器，高效加速应用。

超高能效

- 发挥鲲鹏架构多核、低功耗优势，为推理场景构建高效能、低功耗的AI计算平台；
- Atlas 300I Pro单卡功耗仅72W，为AI服务器算力加速同时提供更优的能效比。

Atlas 800 推理服务器（型号：3000）

- **Atlas 800 推理服务器**（型号：3000）最大可支持8个Atlas 300I/V Pro，提供强大的实时推理能力和视频分析能力，广泛应用于中心侧AI推理场景。



超强算力

- 支持8张Atlas 300I/V Pro 推理卡，满足多场景推理需求；整机可提供1024路高清视频实时分析（1080P 30FPS）；
- 搭载超强算力的鲲鹏920处理器，高效加速应用。

超高能效

- 发挥鲲鹏架构多核、低功耗优势，为推理场景构建高效能、低功耗的AI计算平台；
- Atlas 300I/V Pro 单卡功耗仅72W，为AI服务器算力加速同时提供更优的能效比。

Atlas 800 推理服务器（型号：3010）

- **Atlas 800 推理服务器（型号：3010）** 是基于Intel处理器的推理服务器，最多可支持7个Atlas 300I/V Pro，支持896路高清视频实时分析，广泛应用于中心侧AI推理场景。



灵活配置，适配多项负载

- 支持SAS/SATA/NVMe/M.2 SSD硬盘多种组合灵活配置
- 支持板载网卡和灵活I/O卡，提供丰富多样的网络接口

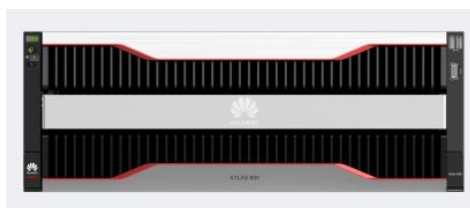


智能视频分析

- 最大支持7张Atlas 300I/V Pro，
- Atlas支持896路高清视频实时分析（1080P 30FPS）

Atlas 800I A2 推理服务器

- Atlas 800I A2 推理服务器采用8模组高效推理方式，提供强劲 AI 推理能力，在算力、内存带宽和互联能力方面具有优势，可广泛应用于生成式大模型推理，例如智能客服、文案生成、知识沉淀等内容生成类场景，支持NPU互联，提升大模型推理效率。



DeepSeek与Ascend碰撞的火花

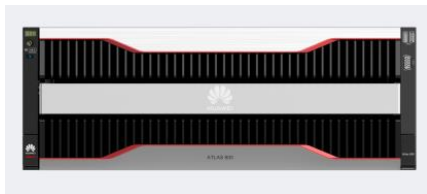
- DeepSeek-V3是由DeepSeek团队开发的一个强大的混合专家（MoE）语言模型，DeepSeek-R1模型是在DeepSeek-V3训练得到的推理模型，二者总参数量均为671B。
- 为了实现高效推理性能，Atlas 800I A2推理服务器进行了深度适配。



DeepSeek

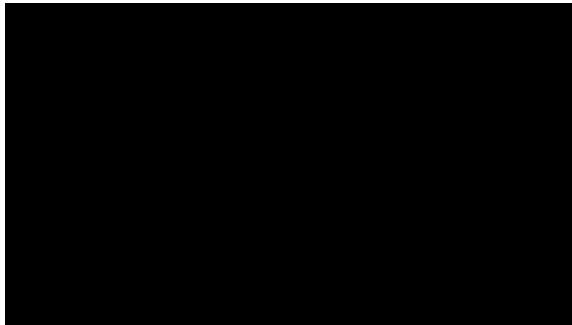
Atlas 800T A2 训练服务器

- Atlas 800T A2训练服务器具有更高算力密度、极致能效比与高速网络带宽等特点。该服务器广泛应用于深度学习模型开发和训练，适用于智慧城市、智慧金融、教育科研、运营商等需要大算力的行业领域。



Atlas 900 A2 PoD

- **Atlas 900 A2 PoD** 集群基础单元，具有极致算力密度、极高AI能效、极简交付部署等特点。该基础单元广泛应用于深度学习模型开发和训练，适用于人工智能计算中心、智慧城市、科研探索、运营商等需要重算力的场景；
- 其中Atlas 900 A2还使用了天成三总线。



目录

1. 昇腾AI基础软硬件平台

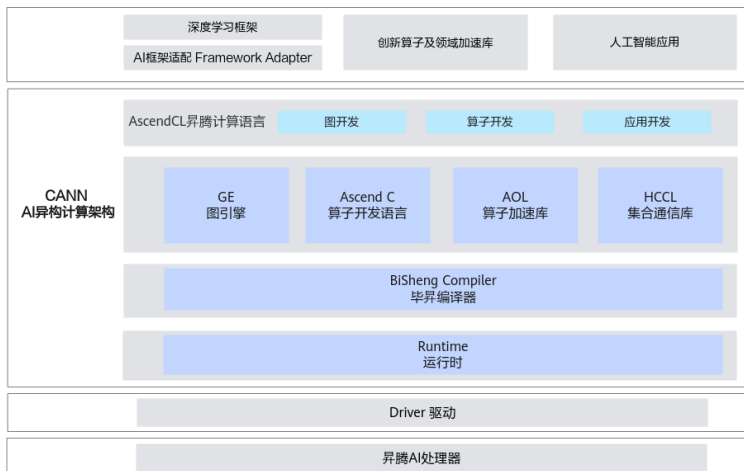
- AI芯片概览
- 昇腾芯片硬件架构
- 昇腾AI基础硬件
- 昇腾AI基础软件

2. 华为云：AI Ready

CANN AI异构计算架构（1）

- 异构计算架构CANN（Compute Architecture for Neural Networks）

是昇腾针对AI场景推出的异构计算架构，向上支持多种AI框架，包括 MindSpore、PyTorch、TensorFlow等，向下服务AI处理器与编程，发挥承上启下的关键作用，是提升昇腾AI处理器计算效率的关键平台。同时针对多样化应用场景，提供多层次编程接口，支持用户快速构建基于昇腾平台的AI应用和业务。



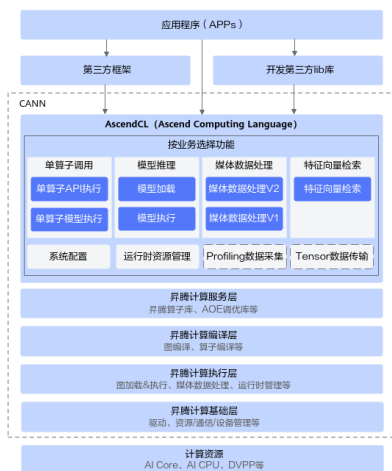
CANN AI异构计算架构（2）

- GE图引擎（Graph Engine）
 - 计算图编译和运行的控制中心，提供图优化、图编译管理以及图执行控制等功能。GE通过统一的图开发接口提供多种AI框架的支持，不同AI框架的计算图可以实现到Ascend图的转换。
- Ascend C算子开发语言
 - CANN针对算子开发场景推出的编程语言，原生支持C和C++标准规范，最大化匹配用户开发习惯；通过多层接口抽象、自动并行计算、孪生调试等关键技术，极大提高算子开发效率，助力AI开发者低成本完成算子开发和模型调优部署。
- AOL算子加速库（Ascend Operator Library）
 - 提供了丰富的深度优化、硬件亲和的高性能算子，包括神经网络（Neural Network，NN）库、线性代数计算库（Basic Linear Algebra Subprograms，BLAS）等，为神经网络在昇腾硬件上加速计算奠定了基础。

CANN AI异构计算架构 (3)

- HCCL集合通信库 (Huawei Collective Communication Library)
 - 是基于昇腾硬件的高性能集合通信库，提供单机多卡以及多机多卡间的数据并行、模型并行集合通信方案。HCCL支持AllReduce、Broadcast、Allgather、ReduceScatter、AlltoAll等通信原语，Ring、Mesh、HD等通信算法，在HCCS、RoCE和PCIe高速链路实现集合通信。
- BiSheng Compiler毕昇编译器
 - 提供Host-Device异构编程编译能力，利用微架构精准编译优化释放昇腾AI处理器性能，提供完备二进制调试信息与二进制工具链，支撑AI开发者自主调试调优。
- Runtime运行时
 - 提供了高效的硬件资源管理、媒体数据预处理、单算子加载执行、模型推理等开发接口，供开发者轻松构建高性能人工智能应用。

昇腾AI处理器神经网络软件流

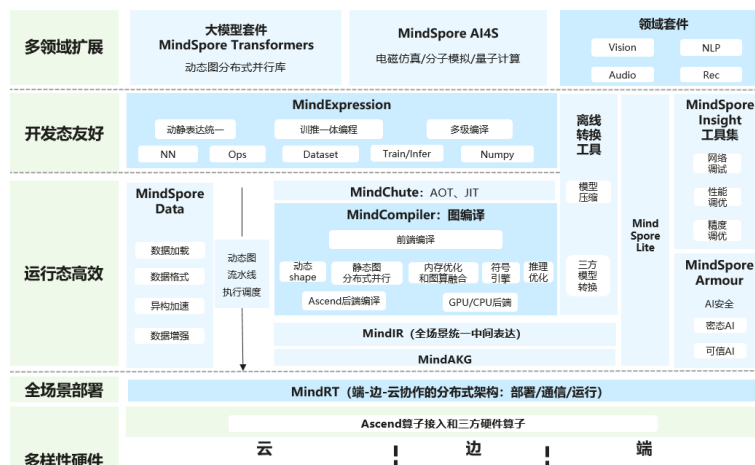


- 昇腾AI处理器神经网络软件流，是深度学习框架到昇腾AI芯片之间的一座桥梁，完成一个神经网络应用的实现和执行，具体流程如下。
- 按业务选择对应功能，然后可通过AOE算子调优等工具对APPs进行优化，同时通过图编译等完成图优化等操作，最后通过运行时管理调度硬件资源，如AI Core、DVPP等，完成程序的执行。

- 此处展示的是推理应用的神经网络软件流，非训练的

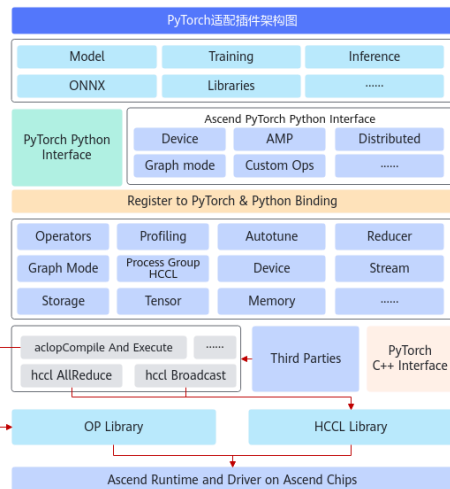
MindSpore

- 昇思MindSpore是一个全场景深度学习框架，旨在实现易开发、高效执行、全场景统一部署三大目标。其中易开发表现为API友好，调试难度低；高效执行包括计算效率、数据预处理效率和分布式训练效率；全场景则指框架同时支持云、边缘以及端侧场景。



Ascend Extension for PyTorch

- Ascend Extension for PyTorch 插件是基于昇腾的深度学习适配框架，使昇腾NPU可以支持PyTorch框架，为PyTorch框架的使用者提供昇腾AI处理器的超强算力。
- 当前阶段，本产品对PyTorch框架与昇腾AI处理器进行了在线对接适配。
- 昇腾AI处理器的加速实现方式是以算子为粒度进行调用（OP-based），即通过Ascend Computing Language（AscendCL）调用一个或几个亲和算子组合的形式，代替原有的算子实现方式。PyTorch适配插件架构如右图所示。



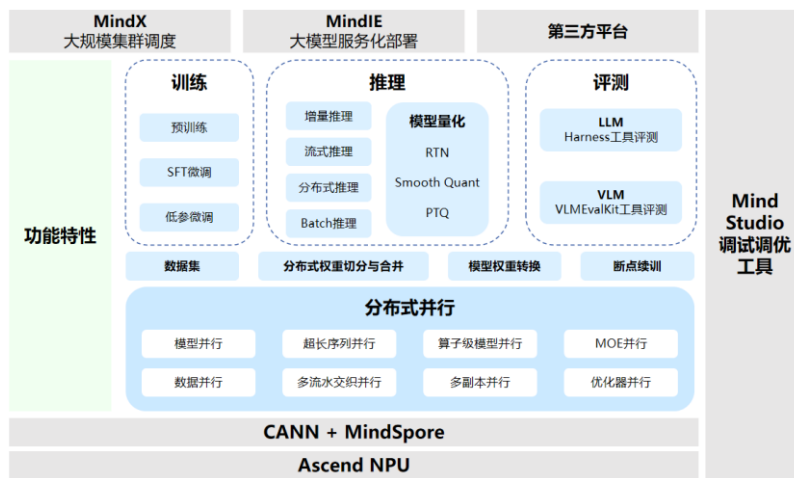
昇腾大模型解决方案概览



- 昇腾大模型解决方案涵盖了模型预训练、微调、精度与性能调优、推理部署等各个流程，并且同时支持业界使用较多的PyTorch框架与华为主导的MindSpore框架

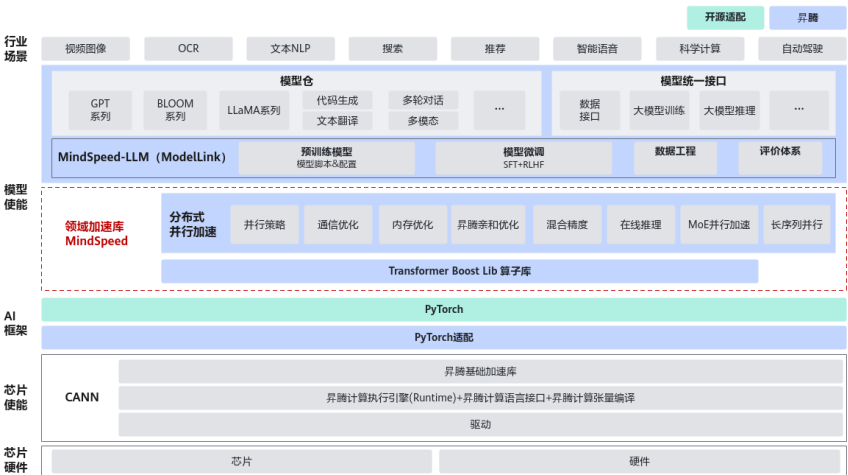
MindFormers

- MindFormers 通过 MindSpore 提供的 Python接口实现大模型相关代码，并由昇腾AI处理器配套软件包提供的算子库进行数据运算。



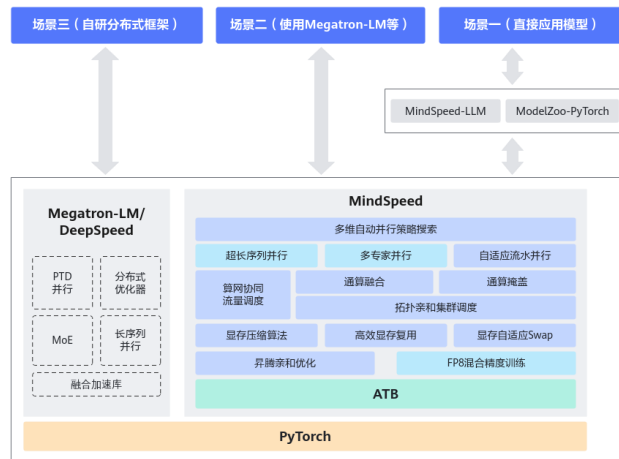
MindSpeed

- MindSpeed 是专为华为昇腾设备设计的大模型加速解决方案，以其卓越的性能表现和深度优化的算法体系，助力用户在昇腾设备上高效实现大模型训练。



MindSpeed-LLM/MM

- MindSpeed-LLM作为昇腾大语言模型套件，下层对接MindSpeed，提供了LLM的分布式预训练、分布式指令微调、分布式偏好对齐以及对应的开发工具链；
- MindSpeed-MM则是昇腾多模态大模型套件，提供了数据工程、分布式训练及加速、在线推理任务等特性；
- 以上两个套件对应于右图中的场景一。



MindIE

- MindIE（Mind Inference Engine，昇腾推理引擎）是华为昇腾针对AI全场景业务的推理加速套件。通过分层开放AI能力，支撑用户多样化的AI业务需求，使能百模千态，释放昇腾硬件设备算力。向上支持多种主流AI框架，向下对接不同类型昇腾AI处理器，提供多层次编程接口，帮助用户快速构建基于昇腾平台的推理业务。



目录

1. 昇腾AI基础软硬件平台
2. **华为云：AI Ready**

华为云：AI Ready，加速千行万业智能升级



AI重塑千行万业

- 20+行业大模型，1000+的行业AI成功实践
- 100+能力集，三层结构三种部署方式

AI重构企业核心产品

- 工具链全栈智能加持，CodeArts重塑软件开发
- PB级音视频数据预训练，MetaStudio带来数字人自由

AI重定义基础设施

- 分布式QingTian架构，突破算力/存储/网络边界
- AI云服务算力，即开即用

最专注行业：解难题，做难事，助力20+行业、1000+客户商业成功



盘古大模型 解难题 做难事 行致远

深耕行业：政务、金融、制造、
煤矿、铁路、制药、气象

煤矿生产少人无人，安全高效

AI-IS	TO-BE
每一公里一矿工巡检	主运减少20%井下人员
掘进人盯人作业	掘进作业序列智能检测
井下作业场景识别准确率	70% → 98%

金融专业知识繁多，业务办理时间长

AI-IS	TO-BE
查询多个业务系统	一次查询即可
花费20分钟	办结时间缩短5分钟
全行业务办理时间缩短	百万分钟

预测未来10天台风路径

AI-IS	TO-BE
3000台服务器数值计算	1卡大模型推理
5小时	10秒
预报速度	10,000x ↑

上亿小分子的成药筛选和优化

AI-IS	TO-BE
研发周期10年	先导药研发1个月
研发成本10亿	研发成本降低70%
先导药研发周期	年 → 月

感知-认知-行动-决策，全流程智能化

AI-IS	TO-BE
城市治理看得见	万物检测万物分割，看得懂
机器人在线	政务AI助手“在场”
城市治理，秒级发现，分钟级分拨	

让人人都能成为专业主播

AI-IS	TO-BE
直播间投入大	直播间全云上构建
好主播难培养难留	专业产品介绍，流利开播
真人最多播4H/天	24H直播，提升商品曝光/GMV
主播培养月	快速专业开播

秉承开放与安全，守护客户核心关切，帮助客户炼好自己的大模型



三层架构，100+能力集，分层解耦

三大工程化套件，5倍速开发行业大模型



安全可靠，三种模式，灵活部署

DataArts：数据治理生产线，一站式数智融合开发与治理



GaussDB：华为云分布式数据库，构建数字世界新底座



ModelArts：提供百模千态沃土，加速千行万业智能

大模型带来新挑战

大模型算力需求呈爆炸性增长

大模型的开发与应用是一个复杂的系统工程

AI落地行业的挑战

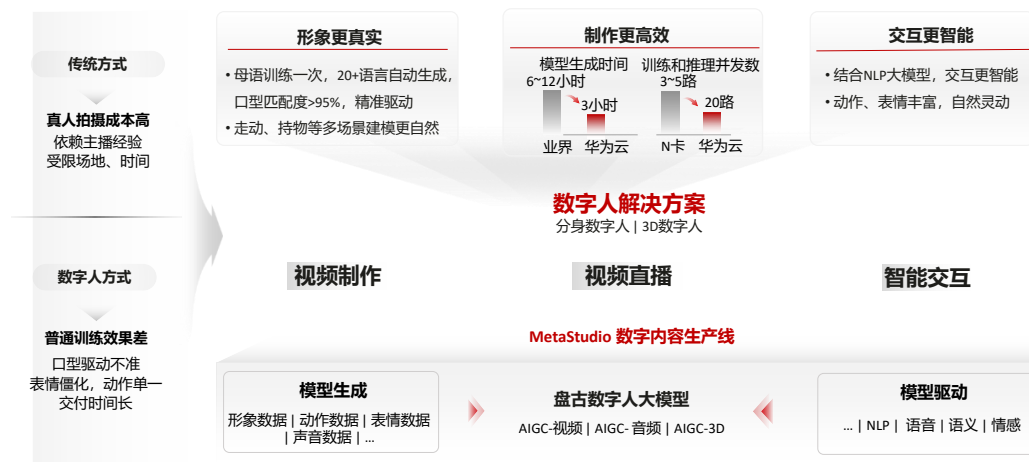
AI开发难：算法开发难、更新快

AI人才少：严重不足，分布不均

行业知识沉淀难，行业场景丰富



MetaStudio：改变内容生成方式，释放数字内容生产力



CodeArts：智能化软件开发生产线，加速企业应用现代化

智能研发

代码生成、代码解释等AI功能，代码产出量提升50%，研发综合效能提升30%以上

极速开发

- 分布式编译全球领先，亿行代码1小时完成；
- 千万级高并发能力，测试周期由周到天

软件供应链安全

- 7000+代码检查规则，11000+代码安全场景；
- 覆盖400万开源组件版本，分钟级漏洞感知



研发效率低

审核、中转节点多，耗时长，效率低

项目管理协作弱

需求协同受限，需求管理模式无法自定义

质量管理效能低

自动化测试率低，代码检查等质量门禁能力弱

着眼效率、质量、安全、组织认知
全面赋能企业快速实现数字化转型

周→天 80+ 71.74% ↓
版本交付周期 业务系统上线 代码问题

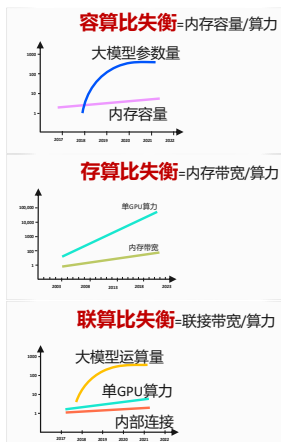
快运 整车 仓储 供应链

面向四大业务场景
提供80+业务系统应用

一站式研发管理解决方案

分布式QingTian：系统性架构升级，打破单服务器硬件限制

算力新挑战，需要系统新架构



分布式QingTian：系统性架构升级，打破单服务器硬件限制



昇腾AI云服务，澎湃算力支持大模型普惠



思考题

1. 昇腾芯片所采用的架构是什么名称？
2. CANN中负责图像预处理的功能单元是什么名称？

- 1、达芬奇
- 2、DVPP

本章总结

- 本章主要介绍了华为在人工智能领域的基础设施，包括基于达芬奇架构的各种昇腾硬件及相关软件、华为云AI服务。

学习推荐

- [昇腾开发者社区](https://www.hiascend.com/)
 - <https://www.hiascend.com/>
- MindSpore官网
 - <https://www.mindspore.cn/>

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



人工智能前沿应用场景



前言

- 在当今科技日新月异的时代，人工智能（AI）正以前所未有的速度改变着我们的世界。从智能助手到自动驾驶汽车，从气象预测到智能机器人，AI的应用领域日益广泛，不断推动着社会的进步和发展。

目标

- 学完本课程后，您将能够：
- 了解AI在日常生活中的应用
- 了解AI在前沿领域中的应用

目录

1. 语音助手
2. 智能家居
3. 智能汽车
4. 智能推荐
5. 智能机器人
6. AI4Science

AI - 无处不在

- 在当今快速发展的科技时代，人工智能（AI）已经成为我们生活中不可或缺的一部分。从智能手机的语音助手，到智能家居的自动化管理，再到医疗、教育和出行等领域的广泛应用，AI正在以各种形式助力人们的生活，提高生活质量和效率。



智慧家居

语音助手



高效出行

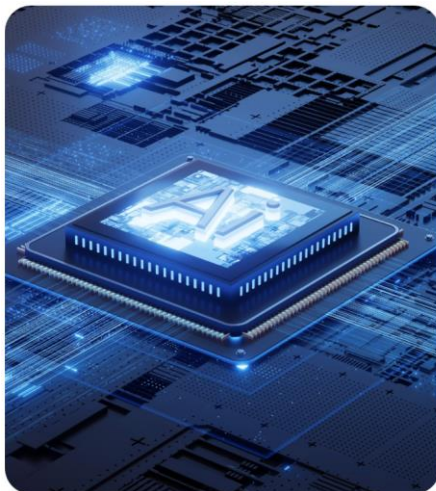
语音助手 - 小艺

- 小艺是华为全场景设备上的智慧助手，在AI大模型加持下，具备全场景解决方案能力。



自然流畅的对话交互，高效
便捷的生产力工具。

小艺 - AI引领科技革命



AI 引领科技革命

大模型技术

基于使用场景、数据预训练优化和增强，更加适用消费者场景。

对话系统

基于大模型意图理解的对话系统增强，实现知识检索和创意生成。

连续服务

基于鸿蒙生态的服务匹配，智能感知用户时空位置，实现更精准的服务推荐。

小艺 - 自然交互



随口说，小艺都懂

语音助手的核心功能之一是语音识别，即将用户的语音指令转换为文本。

小艺拥有自然语言理解能力，对话更轻松自在，可执行模糊询问或复杂任务。



小艺 - 让生活和工作更高效便捷

丰富的知识问答



高效的信息管理

给小艺一篇文档或一篇文章，它能帮你深度分析和翻译内容，快速摘要要点，更支持多轮互动提问。

文案创作，灵感无限



全场景智慧体验



目录

1. 语音助手
- 2. 智能家居**
3. 智能汽车
4. 智能推荐
5. 智能机器人
6. AI4Science

智能家居

- 智能家居，作为现代科技与日常生活完美融合的产物，正以前所未有的速度改变着我们的居住环境和生活方式。
- 它通过集成先进的物联网技术、人工智能算法以及大数据分析，将家中的各种设备连接成一个高度自动化、个性化的智能生态系统，为用户带来前所未有的便捷、舒适与安全体验。



华为全屋智能 - 让生活更便捷

- 在智能家居领域，AI技术的应用使得家居设备能够实现互联互通，形成一个智能化的生态系统。

步入家门，从玄关到客厅，温馨灯光次第渐亮；环境智控系统预判环境温度、湿度、空气质量，自动为你调配至适宜状态。卸下一日疲惫，独享一方天地。

一键回家模式



深夜归家或起夜，廊道夜灯随步行轨迹亮起，不用频繁开关灯，每一步都有光亮温暖陪伴。

华为全屋智能 - 让生活更贴心

让每一寸光，
舒适得恰到好处



遮阳智控，分组开合，根据光照强弱随心调整，过滤强光照射，为你营造自然柔和的采光环境。

触发回家模式，零冷水模式自动打开，智能系统的水温自适应能力让水温随四季、地理位置、个人喜好自动调整。

任凭四季变化，温水无需等待



居家安心，外出放心



每一次离家，摄像头自动开启，家中情况实时掌握，入侵及时告警。

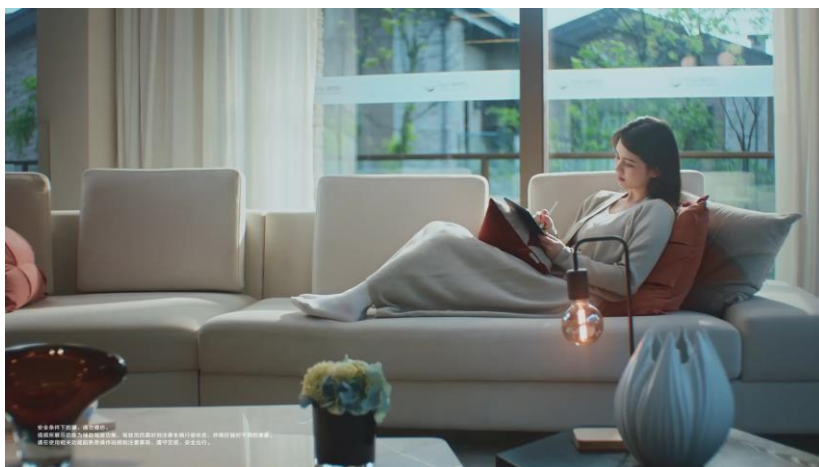
目录

1. 语音助手
2. 智能家居
- 3. 智能汽车**
4. 智能推荐
5. 智能机器人
6. AI4Science

智能驾驶

Advanced Driving System

(高阶智能驾驶系统) 简称ADS，融合了激光雷达、摄像头、毫米波雷达等多传感器，搭载高性能计算平台和全栈自研拟人化算法，致力于让人驾更安全、智驾更舒心、泊车更省心。



华为乾崮智驾

乾崮智驾

乾崮ADS

乾崮ADS 3.0具备强大的识别和处理能力，让异形障碍物识别率达到99.9%，不仅能够精准识别人、车等白名单障碍物，还能识别成排锥桶、静止水马、土堆杂物、大石碎石等异形障碍物。



18 Copyright © Huawei Technologies Co., Ltd. All rights reserved.

融合感知

传感器作为智能汽车的“眼睛”，是智能驾驶技术的核心组成部分，主要包括激光雷达、毫米波雷达和摄像头，通过将多类传感器的数据融合在一起，可以提高对目标的检出率、准确性和鲁棒性，从而实现全场景，全工况，全天候的看护。



智驾计算平台

智驾计算平台是智能汽车的“大脑”，它性能强劲，能够实时接收来自汽车周围的数十个各种传感器感知到的复杂路况信息；它反应灵敏，能够让部署其上的智能驾驶算法对复杂路况信息进行推理计算，并迅速作出驾驶行为决策；它稳定可靠，能够不知疲倦地长时间运行，减轻长途驾驶的疲劳，让我们享受更加轻松、舒适的智慧出行体验。



鸿蒙座舱

- HarmonyOS车机操作系统，强劲自研车机模组，重新定义智能化座舱体验。更懂你的小艺语音，更强大的视觉能力，让出行变得更智慧。开创性智慧互联能力，实现多人多音区，多设备联动。百款车机应用和澎湃车载音响，带来更精彩的移动影音娱乐新体验。



目录

1. 语音助手
2. 智能家居
3. 智能汽车
- 4. 智能推荐**
5. 智能机器人
6. AI4Science

智能推荐

- 智能推荐，是一项基于AI技术构建的推荐系统服务。多元化时代，凭借人工智能推荐可以对大数据进行自动化集成，并通过数据分析、数据过滤，持续洞察终端用户，赋能企业进阶为数据驱动决策型组织。
- 知识图谱是智能推荐系统里的一项技术，它的本质是一个存储实体之间关系的结构化网络，蕴含了推荐系统中对于目标事物的大量背景信息，以及事物之间的关系。知识图谱为推荐提供了目标事物的精确画像，例如用户行为、兴趣、需求等信息，从而实现准确的匹配和有针对性的推荐，实现场景化、任务型的个性化推荐。这种紧密的联系可以让知识图谱在智能推荐中大放异彩。



案例分析

- 下图中，知识图谱提供了不同的关系链接种类，为物品引入了更多的语义关系，有利于推荐结果的发散，避免推荐的局限性，能够更深层次地发现了用户的兴趣。该用户很有可能也喜欢《阿飞正传》《末代皇帝》、《搜索》等。同时，图谱可以连接用户的历史记录和推荐结果，提高用户对推荐结果的满意度和接受度，增强用户对推荐系统的信任。



- 此外，知识图谱作为混合推荐系统中一类有效的辅助信息，有效地解决推荐系统中存在的一系列关键问题，例如数据稀疏、冷启动、推荐多样性等。

目录

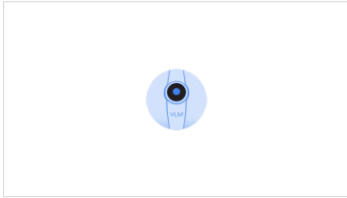
1. 语音助手
2. 智能家居
3. 智能汽车
4. 智能推荐
- 5. 智能机器人**
6. AI4Science

智能机器人

- 人形机器人指具有人的形态、能够完成人类肢体可完成的动作和作业、具有类人的感知学习和认知能力的新型机器人。
- 人形机器人将是“具身智能”最有价值载体，与快速发展的通用人工智能、AI 大模型能力结合，使机器能够以更自然、更智能的方式与环境进行互操作和互动、完成类人的各种复杂任务。

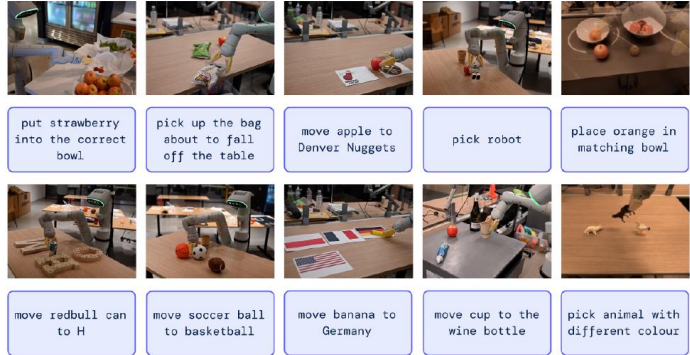


机器人理解指令和事物



将经过互联网规模数据训练的视觉-语言模型直接纳入端到端的机器人控制中，并将此类模型将此类模型称为视觉-语言-动作模型（VLA）。

这个大模型能显著提高机器人对新物体的泛化能力，能够解释机器人训练数据中不存在的命令（例如将物体放在特定的数字或图标上），以及能够执行基本的响应用户命令的推理（例如拿起最小或最大的物体，或者离另一个物体最近的那个）。



RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control -- DeepMind

- DeepMind研究了如何将经过互联网规模数据训练的视觉-语言模型直接纳入端到端的机器人控制中，以提高泛化能力并实现突发性语义推理。我们的目标是使一个经过端到端训练的单一模型既能学习将机器人观察到的信息映射为动作，又能享受大规模预训练在语言和视觉-语言数据上的好处。为此，我们提出对最先进的视觉-语言模型进行联合微调，包括机器人轨迹数据和互联网规模的视觉-语言任务（如视觉问答）。与其他方法不同，我们提出了一个简单、通用的方法来实现这一目标：为了将自然语言响应和机器人动作纳入同一格式，我们将动作表示为文本标记，并将它们以与自然语言标记相同的方式直接纳入模型的训练集。我们将此类模型称为视觉-语言-动作模型（VLA），并实例化了一个这样的模型，称为RT-2。我们的广泛评估（6k次评估试验）表明，我们的方法能够生成高效的机器人策略，并使RT-2从互联网规模的培训中获得一系列突发能力。这包括显著提高对新物体的泛化能力，能够解释机器人训练数据中不存在的命令（例如将物体放在特定的数字或图标上），以及能够执行基本的响应用户命令的推理（例如拿起最小或最大的物体，或者离另一个物体最近的那个）。

机器人像人类一样理解事物

0. 不同的杯子，包括机器人见过的和没见过的



1. 使用见过的杯子执行指令



RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation

2. 使用未见过的杯子1执行指令



3. 使用未见过的杯子2执行指令



- <https://arxiv.org/pdf/2410.07864>
- <https://rdt-robotics.github.io/rdt-robotics/>
- 具身智能

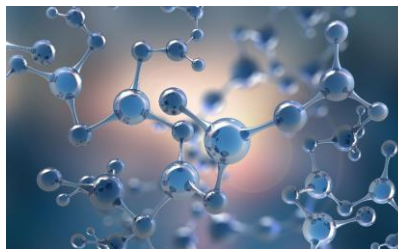
目录

1. 语音助手
2. 智能家居
3. 智能汽车
4. 智能推荐
5. 智能机器人
- 6. AI4Science**

AI在科学中的力量



2024年10月8日，瑞典皇家科学院宣布，将2024年诺贝尔物理学奖授予John J. Hopfield和Geoffrey E. Hinton，以表彰他们“为利用人工神经网络进行机器学习做出的基础性发现和发明”。



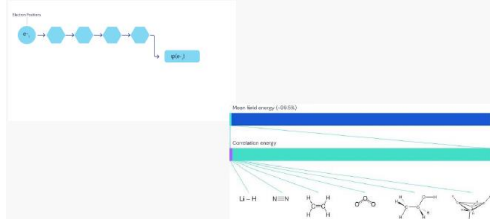
2024年10月9日，瑞典皇家科学院宣布，将2024年诺贝尔化学奖授予David Baker，以表彰其在计算蛋白质设计方面的贡献，另一半则共同授予Demis Hassabis和John M. Jumper，以表彰其开发的人工智能模型AlphaFold2在蛋白质结构预测方面的贡献。

- 诺贝尔物理学奖授予美国科学家约翰·霍普菲尔德和加拿大科学家杰弗里·辛顿，表彰他们“通过人工神经网络实现机器学习的基础性发现和发明”。两位人工智能先驱的工作，为深度学习和神经网络的发展奠定了基础，对人工智能如今的蓬勃发展起到了关键性作用。
- 诺贝尔化学奖中，美国科学家大卫·贝克因为在计算蛋白质设计方面的贡献获一半奖项，英国科学家德米斯·哈萨比斯和美国科学家约翰·江珀因为在蛋白质结构预测方面的成就共享另一半奖项。哈萨比斯和江珀开发了人工智能模型AlphaFold2，解决了一个已有50年历史的难题，能预测大约两亿种已知蛋白质的复杂结构，已被全球200多万人使用，对药物研发等领域具有革命性的意义。
- 这些奖项的颁发，体现了一大科研新趋势：AI技术作为一股不可忽视的力量，正在推动科学研究的范式转变。在物理学、化学、生物学、医药等多个领域，AI成为解决长期存在的复杂科学问题的重要工具。它建立了一个理论上可以解决所有科学问题的范式：即从实际问题出发，将其转变为AI可以处理的输入数据，再经过深度学习网络，最终输出结果。很多科学家认为，AI将推动科学研究不断突破传统框架，实现更加深远、更加广泛的创新。

AI + 量子力学

FermiNet (DeepMind2024)

FermiNet是由DeepMind 提出的神经网络架构，具有参数化的波函数表示方法，可以高效处理多电子量子系统，提升了电子结构计算的精度和效率。FermiNet计算原子和分子的能量，在双环丁烷30个电子组成的分子系统上仍然捕获了97%的相关能量。

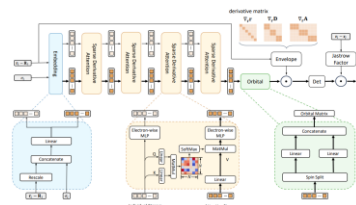


David Pfau, et.al. Accurate computation of quantum excited states with neural networks. Science. 2024

29 Copyright © Huawei Technologies Co., Ltd. All rights reserved.

LapNet (北大&字节2024)

LapNet是由字节跳动与北京大学开发的基于神经网络的量子变分蒙特卡罗NN-VMC的深度学习架构，团队设计了一个Forward Laplacian的计算框架，通过有效的前向传播过程来计算与神经网络相关的Laplacian（这是NN-VMC的瓶颈）。LapNet实现了显著的加速率，从而极大地将NN-VMC的适用性扩展到更大的系统。

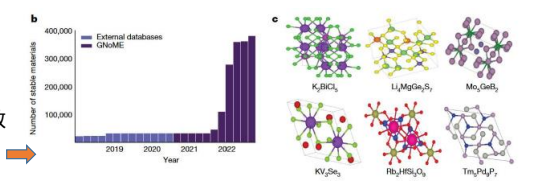


Ruichen Li, et.al. A computational framework for neural network based variational Monte Carlo with Forward Laplacian. Nature M. I., 2024

AI + 材料科学

GNoME (DeepMind, Nature 2023)

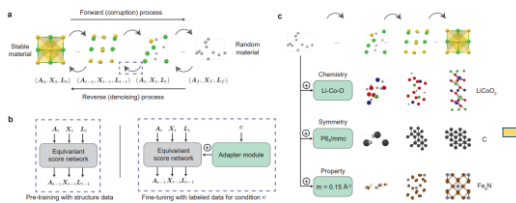
DeepMind发布的 GNoME 工具发现了 220 万种新晶体，其中 38 万个稳定结构有望合成，可能推动下一代电池、超导体等技术变革。GNoME 通过图神经网络（GNN）模型，可以高效地预测材料的晶体结构和稳定性，大幅加速新材料的发现过程。与传统的实验方法相比，AI 模型能在短时间内预测数百万种材料，从而缩短了研究周期，节省了大量实验成本。



Amil Merchant et al. Scaling deep learning for materials discovery, Nature (2023)

MatterGen (Microsoft, 2024)

MatterGen 是微软研究院开发的一种新型生成模型，用于设计具有所需性质的无机材料。MatterGen 能够生成稳定且多样的晶体结构，并通过新型的基于扩散的生成过程细化原子类型、坐标和周期性晶格。该模型可以根据特定属性要求进行微调，生成满足多种属性约束的新材料，如化学组成、对称性以及机械、电子和磁性等性能，进行定向材料设计。

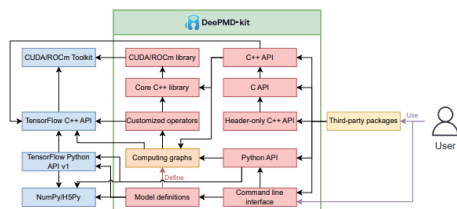


Tian Xie et al. MatterGen: a generative model for inorganic materials design. (2024)

AI + 分子动力学

DeePMD深势科技 2023

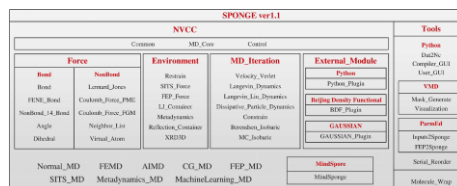
DeePMD-kit是深势科技开发的一个具有很大潜力的深度学习包，通过将第一性原理建模、以机器学习为代表的驱动方法和高性能计算紧密结合，能够以第一性原理的精度处理超大规模的分子动力学问题。DeePMD kit能够在提高模拟精度的同时，显著提升计算效率，解决了传统模拟方法中的精度与计算量之间的矛盾。



Weinan E, et. al. DeePMD kit v2: A software package for deep potential models. 2023

LapNet (北大&字节2024)

SPONGE是北大和深圳湾实验室开发的国内首个开源的通用分子模拟软件框架，旨在解决现有分子模拟软件的可靠性和效率问题，并融合人工智能技术推动分子模拟领域的变革。SPONGE 与华为MindSpore 合作，推出了可与 AI 算法无缝融合的分子模拟库MindSponge。SPONGE 的开发得到了学术界和产业界的支持，未来希望成为下一代分子模拟平台的领航者。

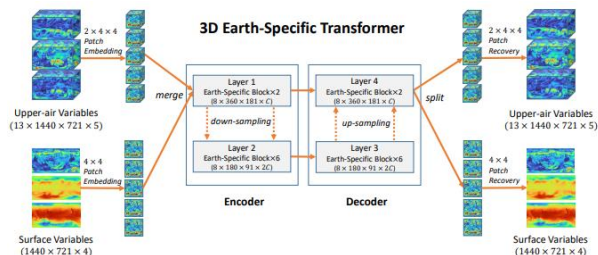


YQ Gao, et. al. SPONGE: A GPU Accelerated Molecular Dynamics Package with Enhanced Sampling and AI Driven Algorithms.

AI + 气象预测

- Pangu Weather (HUAWEI 2023, Nature)

- 华为云发布的盘古气象大模型突破了AI天气预测的精度和速度瓶颈，成为首个精度超越传统数值预报的AI模型。该模型能在秒级完成全球气象预测，速度比传统方法提升10000倍。使用高分辨率数据，盘古气象大模型在短期至中期天气预报中表现出色，并能精准预测极端天气事件。与现有AI模型相比，其准确性和效率显著提高，且能提供更为细致的天气预测细节，标志着气象预报技术的重大进步。



Qi Tian et al. Accurate medium range global weather forecasting with 3D neural networks. Nature (2023)

华为云盘古气象大模型 – 助力气象预测



华为云盘古气象大模型让人们重新审视气象预报模型的未来，模型的开放将推动该领域的发展。

——《Nature》审稿人

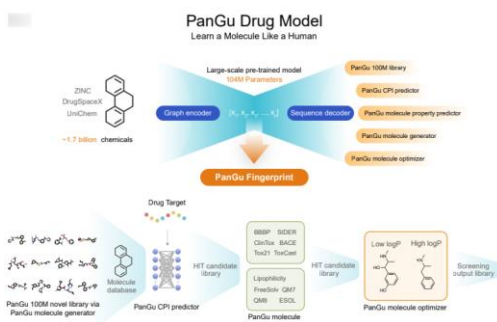
引入盘古气象大模型后，天气预报的时效由之前10天延长至15天，让用户能及早了解未来天气可能出现的变化。

——香港天文台

- <https://www.huaweicloud.com/intl/en-us/about/takeacloudleap2024/ai-weather-prediction.html>

AI + 药物研发

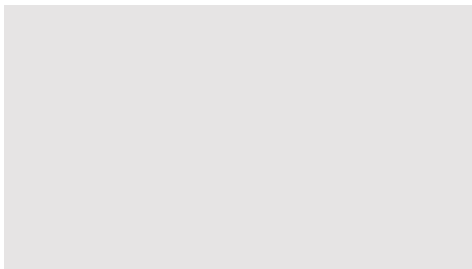
- 华为和中科院团队提出了盘古药物分子大模型，盘古采用不对称图到序列（graph2seq）结构，基于17亿类药化学分子进行预训练，能够预测分子性质、生成和优化分子。该模型在20个药物发现任务中取得领先成绩。盘古的应用有效提升了药物研发效率，推动了AI在制药领域的应用。



Nan, Qiao et al. PanGu Drug Model: learn a molecule like a human. SCLS (2021)

AI推动超级抗菌药Drug X研发加速

- 西安交通大学第一附属医院的刘冰教授利用基于华为云盘古药物分子大模型打造的 AI辅助药物设计服务，成功研制超级抗菌药Drug X，该药物通过靶向微生物类组蛋白HU，抑制细菌的DNA复制达到抗菌效果，是世界上首次发现噬菌体编码靶向细菌类组蛋白HU的抑菌抑制剂，有望成为全球近40年来首个新靶点、新类别的抗生素。在华为云盘古药物分子大模型的辅助下，Drug X先导药的研发周期从数年缩短至一个月，研发成本降低70%，大幅提升新药研发效率。



- <https://www.huawei.com/cn/huaweitech/industry-ecosystem/huawei-cloud-liubin>
- <https://www.huaweicloud.com/about/takeacloudleap2024/xi-an-jiaotong-university.html>

思考题

1. 请列举几个自己身边常见的AI应用。

- 智能音箱、文生图APP.....

本章总结

- 本章不仅介绍了AI在我们日常生活中的应用，给我们生活带来了超乎想象的变化，同时还介绍了AI在气象、机器人等领域的应用，展现了AI的魅力！

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



专业技术人员职业素养与组织赋能



前言

- 在技术高歌猛进、业务复杂多元的今天，我们坚信，卓越的专业追求绝非孤立存在，它既需要**内化于心的职业道德**——对专业诚信的坚守、对合规红线的敬畏、对技术伦理的审慎；也需要**外化于行的组织赋能**——打破技术孤岛，通过知识沉淀、经验传承，将个人能量转化为组织发展的动能。

目标

- 学完本课程后，您将能够：
 - 清晰界定技术应用的边界，识别风险；
 - 建立持续迭代的技术成长思维，以适应指数级发展趋势；
 - 掌握标准化带教方法，将个人技术传承延续。

目录

1. 职业道德
2. 终身学习
3. 培训指导

从业者的三重责任

三重责任

核心目标：明确责任边界与核心义务

1. **专业责任：**追求技术卓越，保持专业精进
2. **社会责任：**维护公共利益，坚守伦理底线
3. **带教责任：**传承知识与规范，赋能团队成长



三者间的关系

职业道德 → 技术技能 (约束关系)

职业道德为技术技能的应用划定边界。如生成式AI伦理约束AI模型开发方向；网络安全法规约束网络技术操作

培训指导 → 技术技能 (赋能关系)

培训指导为技术技能的习得提供方法论。如利用知识图谱工具构建能力图谱指导学习路径；“先培训再上岗”确保技术技能达标

技术技能 → 职业道德 (实践关系)

技术技能是践行职业道德的载体。如只有掌握了数据安全技术，才能真正保护用户隐私

技术技能 → 培训指导 (反馈关系)

技术技能的掌握程度反馈培训指导的有效性，驱动培训方案迭代

职业道德 ↔ 培训指导 (融合关系)

培训指导本身需遵循职业道德；同时培训指导也是传播职业道德的主渠道

职业道德核心原则解析

一般伦理原则

核心目标：确立社会底线

1. **人类福祉**：将公众安全、健康及福祉置于技术发展的首位
2. **诚实守信**：保持专业诚信，不隐瞒已知风险，不误导用户
3. **公平公正**：避免算法歧视，确保技术应用的包容性与公平性
4. **避免伤害**：预防技术滥用，主动规避对社会的潜在负面影响
5. **尊重权益**：尊重法律、知识产权以及用户隐私，严守数据边界

职业伦理原则

核心目标：确立行业标准

1. **追求卓越**：持续精进技术能力，保持对高质量交付的追求
2. **保持专业**：遵循行业规范，维护职业声誉，保持客观中立
3. **同行评议**：确保研究成果的质量和可信度，维护学术和职业的公平和正义
4. **尽职尽责**：对岗位负责，确保系统稳定与安全
5. **维护专业形象**：维护计算机行业的形象与声誉，促进计算机技术的普及及推广

一般伦理原则与职业伦理原则的双重规范

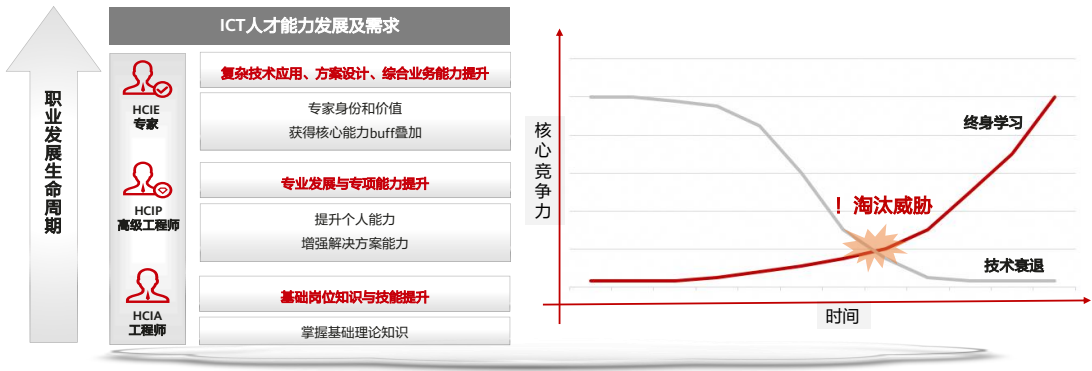
目录

1. 职业道德
- 2. 终身学习**
3. 培训指导

数智升级，对人才培养与技能评价提出更高要求



华为职业认证：终身学习



技术发展现状

- 迭代加速、跨界融合、技术颠覆成为常态，停止学习即被淘汰
- 从“碎片化输入”到“知识晶体”，真正的学习，是将外界的碎片重构成自身的“护城河”

目录

1. 职业道德
2. 终身学习
- 3. 培训指导**

角色升级：从技术能手到培训导师

“技术工作是把事做好，培训带教是把人带会”

	技术能手	培训导师
核心角色	解决特定技术问题	知识传递者、能力引导者、问题解决者
交付成果	高质量的代码与系统	独立规范上岗的专业人才
核心能力	专业硬技术、排障能力	会梳理知识、会标准示范、会纠错复盘



带教原则

实用为先

贴合岗位真实场景，不讲空话

实操为主

重动手，重细节，真操实练，夯实技术

问题导向

解决实操卡点与上岗痛点

全程闭环

课前有备

定目标，梳理重难点

课中有方

示范、纠错、控场、盯安全

课后有落

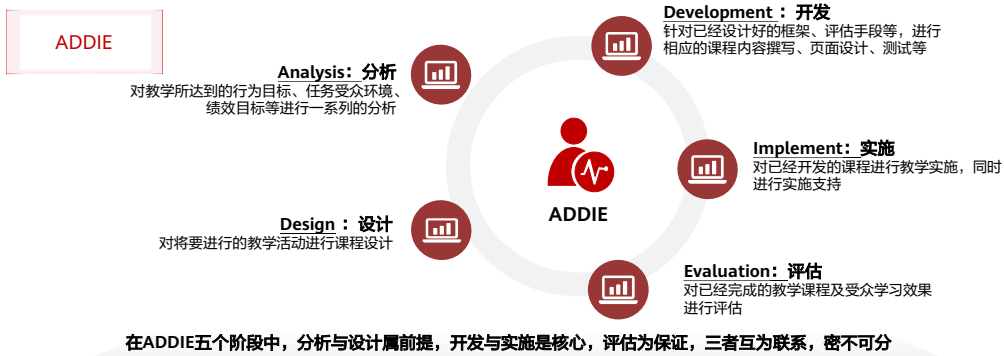
答疑、考核、建档、跟进成长

培训与指导

ADDIE：将A、D、D、I、E五个步骤综合起来考虑，形成系统，保障各个环节能进行及时有效的评估

SAM模型：提倡“快速迭代、小步快跑”，分为三阶段：准备、迭代设计、迭代开发

6D法则：以终为始，把培训的生命周期向后延伸，将“培训”转化为“商业结果”的全面流程



Analysis: 分析阶段

分析阶段是首要的环节，是确定培训目标、设计培训计划的前提，也是决定培训是否开展的关键因素

在培训的各个阶段都要有分析，通过问卷、约谈、电话等各种形式，了解对象、组织、课程的各方面需求



需求分析 (WHY/WHAT) :

组织需求 (公司层面) :

公司的战略目标是什么?
目前阻碍目标实现的“痛点”是什么?

岗位需求 (标准层面) :

目标岗位需要具备哪些核心胜任力?
标准的操作流程 (SOP) 或优秀员工的行为典范是怎样的?



需求对象分析 (WHO) :

基本属性 (画像) :

学员的年龄分布、学历、司龄、目前的岗位层级。

认知与经验起点:

他们对这个主题已经了解多少?
他们日常遇到的实际工作场景和痛点是什么?

学习动机与风格:

他们是主动想学，还是被公司强迫来的?



培训环境分析 (Where & How)

面授课堂: 教材需要配套 PPT、讲师手册、学员手册、通关演练通关单。

线上微课/E-learning: 教材需要拆解为结构化的“脚本”或“脚本大纲”，方便录制音频或制作动画。

混合式培训 (OMO) : 线上学理论 (教材偏精简、清单化)，线下做演练 (教材偏案例、工具化)。



培训内容分析 (Which)

知识类: 讲概念、原理 (要少而精)。

技能类: 讲步骤、方法、流程 (要多、要细、要有标准)。

态度类: 讲观念转变、企业文化 (要通过故事和案例来感染)。



培训资源分析 (With What)

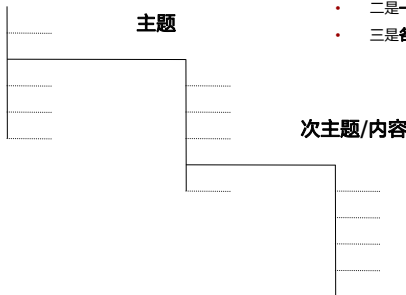
师资资源: 有没有内部的业务专家 (SME) 作为内容源头?

素材资源 (现成原料) : 公司现有的 SOP 手册、过往的失败案例库、竞品资料、行业标准等。

Design：设计阶段

设计阶段要制订一份完成培训项目所必需的说明提纲。基本内容包括必须制定的目标、目的和评估任务
基于培训项目的相关决策和其它实施问题的体系结构，确定各项任务的顺序

单元



课程单元结构的关系

- 一是**相对独立型**，各学习单元在位置上可以互换
- 二是一个单元是构成另一个单元的基础，这种类型在序列上极为严密
- 三是各单元学习内容的联系成**综合型**，即有的是相互独立关系，也有的是严格序列关系

主题的组织方式 - 螺旋式

- 即根据培训对象的智力发展水平，让学员尽早有机会在不同程度上去接触和掌握某门学科的基本结构，以后随着学员在智力上的成熟，围绕基本结构不断加深内容深度，使学员对知识有更深刻和有意义的理解。

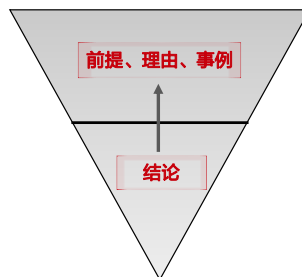
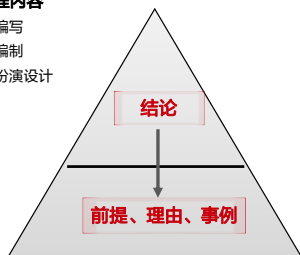
Development : 开发阶段

开发阶段的要点在于**开发创建教学材料**的方法和流程。根据之前总结的各干系方需求，开发培训教材（资料），设计好阶段性评估文件等。在这一阶段中，内容和材料从概念设计转变为可交付成果。

1: 收集素材

2: 组织课程内容

- 内容编写
- 案例编制
- 角色扮演设计



检查错误

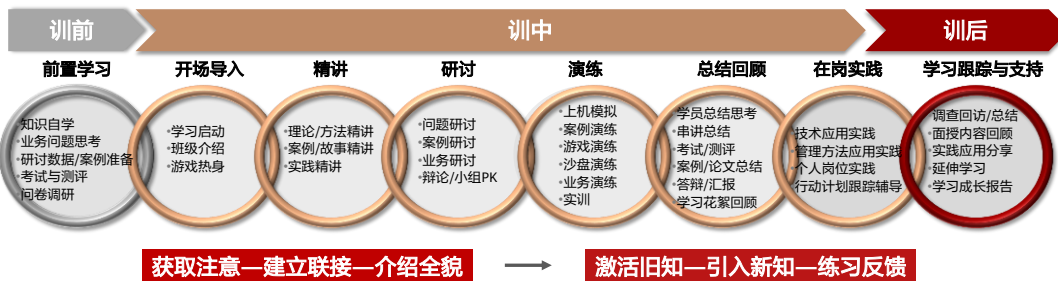
- 标题
- 段落
- 句子
- 表格
- 图形
- 缩略语/术语
- 数与数量
- 标点符号
- 单位符号

思考时多按金字塔顺序，但表述要用倒金字塔顺序

Implement: 实施阶段

- **教学程序**: 从把学员引入某个专题, 到掌握某个目标为止向学员呈现教学时教学活动的程序、步骤
 - **教学方式**: 呈现教学时运用的手段、方法, 如分组讨论、独自阅读、个案研究、讲授、计算机模拟、小组协作项目等
- 认知学习的一个关键概念是在信息从**工作记忆**迁移至**长期记忆**时, 对信息进行提取或演练, 从而强化长期记忆

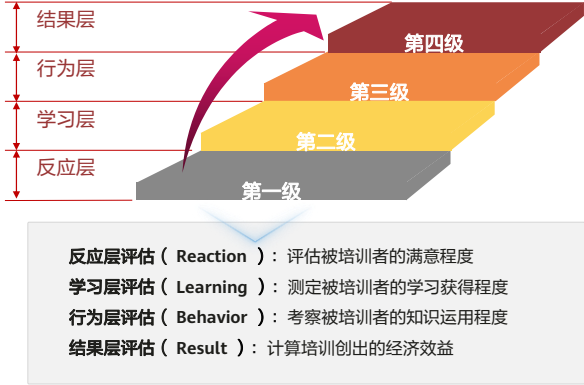
“三段八环六素”教学理论, 把学习过程分片, 根据不同技术场景, 适配不同教学活动, 提升学习效果。



Evaluation: 评估

主流评估模型：柯氏四级、菲利普斯五级、考夫曼五级、CIPP 模型等
行业地位：历经行业检验，**柯氏四级**时至今日仍是不可动摇的经典基石模型

层次	重点描述	衡量方法	评价人员
反应层	受训者满意程度：喜欢该项目吗？对培训人员和设施有什么意见吗？课程有用吗？	问卷	学员
学习层	受训者在培训前后，知识、技能、态度、行为方式方面的收获：有多大程度的提高？	笔试、实地操作和工作模拟	学员和培训人员
行为层	培训后，受训者行为的改进：行为有无不同？他们在工作是否使用了在培训中所学的知识？	问卷调查、面谈、直接观察、业绩检测、专题研究、实践计划	学员、学员上级、下属、同事、客户
结果层	受训者获得的经营业绩：组织是否因为培训变得更好？	问卷调查、业绩检测、业绩目标法	学员、学员上级



思考题

1. （单选题）项目经理小张突然接到紧急通知，需要当晚在公司内网加班处理一份涉及公司**核心商业秘密与关键技术参数**的涉密文件。由于公司保密机房的专用有线计算机键盘出了故障，为了不耽误进度，小张决定采取一些临时措施。请问以下哪**一项行为是明确禁止**的？
 - A. 临时将这份涉密文件拷贝到自己带有 Wi-Fi 芯片和蓝牙功能的个人便携式笔记本电脑上处理。
 - B. 立即联系IT安全部门，申请对当前的涉密系统环境进行一次全面的安全性评估。
 - C. 调整工位，移步到未链接互联网、且切断一切外网物理信号的涉密专用计算机上处理。
 - D. 确保待在公司规定的受控办公室内，在无无关人员围观的闭环环境下处理该涉密资料。
2. （判断题）培训完成后，进行了结业考试，考试通过就代表学员完全掌握了这项技能。

- A
- 错

本章总结

- 本章讲述了从业者的成长路径
 - 坚守底线：牢记行为红线，确保技术向善
 - 持续学习：践行终身学习，应对瞬息万变
 - 培训指导：应用教学实践，传承优秀同路人

学习推荐

- 《中国计算机学会职业伦理与行为守则》：
https://www.ccf.org.cn/About_CCF/CCF_Constitution/2023-07-26/794450.shtml

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

版权所有 © 华为技术有限公司 2026。

保留一切权利。

未经华为事先书面许可，任何人不得对本资料进行复制、修改、
整合、改编、翻译，也不得将本资料或其任何部分或基于本资
料的衍生作品提供、出售、转让、再许可给任何人用于华为
许可以外的任何其他用途。

Copyright © Huawei Technologies Co., Ltd. 2026.

All Rights Reserved.

Without prior written permission of Huawei, no one is
allowed to reproduce, modify, integrate, adapt, translate
this Document, or provide, sell, transfer, sublicense this
Document or any part of this Document or its derivative to
any third party for any other purpose otherwise permitted
by Huawei.

