人工智能导论

> 我们期末考试的题型主要包括了**选择、填空、判断、问答、证明**等题型。大家复习的时候需要多理解老师们在课上讲的关于传统人工智能、博弈论、新人工智能的部分中的**示例**，不需要死记硬背一些概念。其中博弈论部分复习要点为：
>
> 1.**基本概念 博弈 占优策略 纳什均衡 （混合）策略 囚徒困境**
>
> 2.**对抗算法基本原理 minimax 算法 alpha-beta 剪枝 蒙特卡洛树搜索算法**
>
> HTML公式整洁版

求star♡🤍💔😘

人工智能的概念

- **定义 智能机器**

  能够在各类环境中自主地或交互地执行各种**拟人任务**(anthropomorphic tasks)的机器。

- **定义 1 人工智能**

  AI 是关于知识的科学，怎样**表示知识**以及怎样**获得知识**并**使用知识**的科学。(Nilsson)
  AI 就是研究如何使计算机做过去**只有人才能做**的智能工作。(Winston)

- **定义 2 人工智能(学科)**

  人工智能(学科)是计算机科学中涉及研究、设计和应用智能机器的一个分支。它的近期主要目标在于研究用机器来**模仿和执行人脑的某些智力功能**，并开发相关理论和技术。

- **定义 3 人工智能(能力)**

  人工智能(能力)是智能机器所执行的通常与**人类智能有关的智能行为**，如判断、推理、证明、识别、感知、理解、通信、设计、思考、规划、学习和问题求解等思维活动。

- **定义 4**

  人工智能是一种**使计算机能够思维**，使机器具有智力的激动人心的新尝试(Haugeland,1985)。

- **定义 5**

  人工智能是那些与人的思维、决策、问题求解和学习等有关**活动的自动化**(Bellman,1978)。

- **定义 6**

  人工智能是用计算模型研究智力行为 (Charniak 和 McDermott,1985)。

- **定义 7**

  人工智能是研究那些使理解、推理和行为成为可能的计算(Winston,1992)。

- **定义 8**

  人工智能是一种能够执行需要人的智能的创造性机器的技术（Kurzwell,1990)。

- **定义 9**
- 人工智能研究如何使计算机做事让人过得更好（Rick 和 Knight,1991)。
- **定义 10**

  人工智能是一门通过计算过程力图**理解和模仿智能行为**的学科（Schalkoff,1990)。
- **定义 11**

  人工智能是计算机科学中与智能行为的自动化有关的一个分支（Luger 和 Stubblefield,1993)。

> 定义 4 和定义 5 涉及**拟人思维**;
> 定义 6 和定义 7 与**理性思维**有关;
> 定义 8 和定义 9 涉及**拟人行为**;
> 定义 10 和定义 11 与**拟人理性行为**有关。

AI 研究如何用计算机来**表示和执行**人类的**智能活动**，以**模拟**人脑所从事的推理、学习、思考和规划等**思维活动**，并**解决**需要人类的智力才能处理的**复杂问题**等。AI 还涉及到脑科学、神经生理学、心理学、语言学、逻辑学、认知科学等许多学科领域。是一门综合性的交叉科学和边缘学科。

> 不属于人工智能:
>
> - 自动控制类
> - 科学计算类
> - 固定了算法的

强人工智能 *VS* 弱人工智能

- 弱人工智能: 某方面人工智能 强
- 强人工智能: 综合的多方面的人工智能 弱

人工智能的潜力和界限

- 机器的智能：不管什么计算机，都等价于**图灵机**
- 人的智能：还远远没有认识清楚

图灵机

- 一台图灵机是是一个数学概念，一个七元**有序**组

纸带	符号	读写头	规则	状态	起始	结束
存储	符号	读写	程序	数据	开始	结束

- **停机问题**
  - 就是判断任意一个程序是否能在有限的时间之内结束运行的问题
  - 等价于是否存在一个程序 P，对于任意输入的程序 w，能够判断 w 会在**有限时间内结束**或者**死循环**

```
def halt(func, input):
    '''判定函数func在输入input下是否能结束/停机'''
    if func(input): return True
    else: return False
def Turing(func):
    '''返回函数func在输入自己时是否能结束/停机的逆向结果'''
    if halt(func, func): return False
    else: return True
```

```
if Turing(Turing) == True:
==> halt(Turing, Turing) == False
==> Turing(Turing) == False
else Turing(Turing) == True:
==> halt(Turing, Turing) == False
==> Turing(Turing) == False
```

哥德尔不完备定理

- 任何**包含自然数定义的形式系统**都是不完全的，也就是存在不能证明为真，也不能证明为假的命题

可计算性

- 一个语言L是可以被图灵机所枚举（**enumerate**)的，如果存在一个图灵机M，使得输入是L中的串时，M 输出“接受”；而对非L中的串，M输出“拒绝”或不停机
- 存在语言L_d，是不能被图灵机所枚举的，以及存在语言L_u，是不能被图灵机所决定的

人工智能的三大流派

符号主义：最经典、接受程度最高的人工智能

- 把现实的物，映射到代表它的符号，在符号上完成所有的推理、计算等等，如**图灵测试**

连接主义：新人工智能最红火的部分

- **神经网络**模型，模拟人脑神经元之间的连接，从而模拟出人脑的功能
- **深度学习**的功能强大，问题也不少

行为主义：智能来自更低级的感知和行动

- 波士顿动力公司的机器狗、无人机、群体智能

人工智能的三个阶段

早期人工智能：1956 年创立到 80 年代早期

- 科技贵族的专属品
- 概念、机器定理证明，博弈，机器翻译，模式识别.....
- **摩尔定律**：当价格不变时，集成电路上可容纳的元器件的数目，约每隔 18-24 个月便会增加一倍，性能也将提升一倍。换言之，每一美元所能买到的电脑性能，将每隔 18-24 个月翻一倍以上

中期的人工智能：80 年代到 2010 年

- 以**专家系统**为代表的，以**知识**为核心
- 偏向控制，介于**自动化和计算机**之间

新人工智能：近几年开始

- 介于**计算机**和**软件**之间

人工智能的三次浪潮

20 世纪 50 年代末至 70 年代初

人工智能学科诞生，基于知识的方法和具有初步智能的机器出现。
由于数学模型存在缺陷，以及计算能力无法完成复杂度呈指数级增长的计算任务，使人工智能发展陷入低谷。

20 世纪 80 年代初至 90 年代初

数学模型实现了重大突破，诞生了多层神经网络、BP 反向传播算法和高度智能机器等。
由于人工智能计算机的成本与维护难度都较高，限制了其大规模商业应用和普及，使人工智能再次步入低谷。

21 世纪初至今

大数据、算法模型和计算能力的多重突破共同驱动了新一代人工智能的快速发展，尤其是 2006 年深度学习神经网络的提出，使人工智能的性能获得突破性进展，标志着人工智能迎来第三次高速成长期。

新一代人工智能技术的新特点：人机融合，大数据，群体性，自主化，跨媒体

新一代人工智能与前两轮浪潮的本质区别

- 已经可以**大规模应用于经济社会实践**，给企业带来盈利，给社会管理带来实质性改善。
- 从**学术驱动、研究驱动**转变为**需求驱动和产业驱动**， 企业的研发激励巨大、引领作用显著；
- **企业投资**及**研发主导**、**政府规划引领**，使新一代人工智能发展正进入良性循环；前两次浪潮中主要依赖**政府资助**。

人工智能的领域

常见的应用领域

- 机器定理证明(早期很红火，证明了四色定理)
- 博弈：人工智能的经典问题
  - **Alpha GO 成为新人工智能的标志事件**，从 Alpha Go 到 Alpha Go Zero 到 Alpha Zero
- 模式识别：声图文
  - 视频图像（目标跟踪、视频理解、检索等等)
  - 静态图像（人脸识别、图像理解、检索等等)
  - 声音识别（内容识别、声纹识别、语音矫正、语音合成、检索等)
  - 文字识别（手写、联机手写、印刷体、多语种等等)
- 自然语言处理（很多方向)
  - 翻译、理解、问答、作文作诗
  - 翻译是最复杂，有规则学派和统计学派
- 数据挖掘和知识发现（当年曾经号称发现了开普洛行星运行定律)
- 专家系统（费肯鲍姆)

自动程序设计

机器人

NP-NPC 问题

群体智能

智能+

- 教育、网络、CAD、绘画、作曲、决策、模拟……

从研究方向的分类

- 知识表示（IF–then，框架，三元组，统计模型，HMM 隐马尔可夫链）
- 推理方法（归结原理，Lisp 语言，Prolog 语言）
- 搜索技术
- 机器学习（发现规则，调整参数）
- 群智算法（简单智能间能否配合）
- 规划（机器人）
- 神经网络（自己发现规则）
- 图计算（用一个数代表一个节点）

大数据

大数据的 5V 特点（IBM 提出）

- Volume (大量)
- Velocity(高速)
- Variety(多样)
- Value(低价值密度)
- Veracity(真实性)

大数据时代大量的使用相关性而不是因果性

局部大数据和打通的大数据

合法的大数据与非法大数据

专家系统与产生式

专家系统（产生式系统)的组成三要素

1. 一个综合数据库——存放信息
2. 一组产生式规则——知识

规则的一般形式	简写
IF <前提> THEN <结论>	<前提> —> <结论>
IF <条件> THEN <行动>	<条件> —> <行动>

3. 一个控制系统——规则的解释或执行程序（控制策略）（推理引擎）

Example 1 字符转换（六六六）

- 综合数据库：{x}，其中 x 为字符
- 规则集
    1. IF A∧B THEN C
    2. IF A∧C THEN D
    3. IF B∧C THEN G
    4. IF B∧E THEN F
    5. IF D THEN E
- 控制策略：顺序排队
- 初始条件：{A，B}
- 结束条件：F∈{x}
- 求解过程

数据库	可触发规则	被触发规则
A，B	(1)	(1)
A，B，C	(2) (3)	(2)
A，B，C，D	(3) (5)	(3)
A，B，C，D，G	(5)	(5)
A，B，C，D，G，E	(4)	(4)
A，B，C，D，G，E，F		

Example 2 M-C 问题 Missionaries and Cannibals

- 综合数据库：(m, c, b)，其中：$0 ≤ m, c ≤ 3, b ∈ {0, 1}$
- 规则集：
    1. $\bold{IF} (m, c, 1) \bold{AND} 1≤i+j≤2 \bold{THEN} (m-i, c-j, 0)$
    2. $\bold{IF} (m, c, 0) \bold{AND} 1≤i+j≤2 \bold{THEN} (m+i, c+j, 1)$
- 控制策略
- 初始状态：(3，3，1)
- 目标状态（结束状态）：(0，0，0)
- 求解过程

	M Go →	C Go →	M Come ←	C Come ←
33 00		2		1
32 01		2		1
31 02	2		1	1
22 11	2			1
03 30		2		1
02 31		2		
00 33				

Example 3 猴子摘香蕉问题

- 综合数据库：(M, B, Box, On, H)
  - M：猴子的位置
  - B：香蕉的位置
  - Box：箱子的位置
  - On=0：猴子在地板上
  - On=1：猴子在箱子上
  - H=0：猴子没有抓到香蕉
  - H=1：猴子抓到了香蕉
- 规则集：
  1. IF $(x, y, z, 0, 0)$ THEN $(w, y, z, 0, 0)$【猴子可移动】
  2. IF $(x, y, x, 0, 0)$ THEN $(z, y, z, 0, 0)$【猴子可以带同位置箱子动】
  3. IF $(x, y, x, 0, 0)$ THEN $(x, y, x, 1, 0)$【猴子可以爬上箱子】
  4. IF $(x, y, x, 1, 0)$ THEN $(x, y, x, 0, 0)$【猴子可以从箱子下来】
  5. IF $(x, x, x, 1, 0)$ THEN $(x, x, x, 1, 1)$【猴子、香蕉、箱子同一位置，且站在箱子上，可以摘香蕉】
- 控制策略
- 初始状态：$(c, a, b, 0, 0)$
- 结束状态：$(x1, x2, x3, x4, 1)$
- 求解过程
  $1. (x, y, z, 0, 0)$
  $2. (z, y, z, 0, 0)$
  $3. (y, y, y, 0, 0)$
  $4. (y, y, y, 1, 0)$
  $5. (y, y, y, 1, 1)$

产生式系统的特点

- 数据驱动
- 知识的无序性
- 控制系统与问题无关
- 数据、知识和控制相互独立

产生式系统的类型

- 正向、逆向、双向产生式系统
- 可交换的产生式系统
- 可分解的产生式系统

搜索策略

产生式系统的搜索策略

- 内容：状态空间的搜索问题
- 搜索方式：
  - 盲目搜索
  - 启发式搜索
- 关键问题：如何利用知识，尽可能有效地找到问题的解（最佳解）

回溯策略（皇后问题）

- Example 4 递归回溯的例子

```
int ListLenght(LIST *pList)
{
    if (pList == NULL) return 0;
    else return ListLength(pList->next) + 1;
}
```

```
procedure bt(c) is
    if reject(P, c) then return
    if accept(P, c) then output(P, c)
    s ← first(P, c)
    while s ≠ NULL do
        bt(s)
        s ← next(P, s)
```

图搜索策略

- 回溯搜索：只保留从初始状态到当前状态的一条路径；不是图搜索
- 图搜索：保留所有已经搜索过的路径。
  - 深度优先搜索 Depth First Search
    - 一般不能保证找到最优解
    - 当深度限制不合理时，可能找不到解
  - 宽度优先搜索 Breadth First Search
    - 当问题有解时，一定能找到解
    - 当问题为单位耗散值，且问题有解时，一定能找到最优解
  - 渐进式深度优先搜索方法

启发式图搜索：利用知识来引导搜索，达到减少搜索范围，降低问题复杂度的目的

- 启发信息的强度
  - 强：降低搜索工作量，但可能导致找不到最优解
  - 弱：一般导致工作量加大，极限情况下变为盲目搜索，但可能可以找到最优解
- 基本思想：定义一个评价函数f，对当前的搜索状态进行评估，找出一个最有希望的节点来扩展

启发式搜索算法A

- $f(n) = g(n) + h(n)$
  - $f(n)$：评价函数
  - $h(n)$：启发函数
- 在A算法中，如果满足条件：$h(n)≤h^(n)$则A算法称为A算法

$A*$算法的性质

- $A*$算法的假设：设$n_i、n_j$是任意两个节点，有：$C(n_i, n_j) > ε$其中 ε 为大于 0 的常数
- $f^(s) = f^(t) = h^(s) = g^(t) = f^*(n)$
- 其中s是初始节点，t是目标节点，n是s到t的最佳路径上的节点
- 定理 1：对有限图，如果从初始节点 s 到目标节点 t 有路径存在，则算法A一定成功结束
- 定理 2：对无限图，若从初始节点 s 到目标节点 t 有路径存在，则$A*$一定成功结束
  - 引理 2.1
    - 对无限图，若有从初始节点s到目标节点 t 的路径， 则A不结束时，在 *OPEN* 表中即使最小的一个f值也将增到任意大，或有$f(n)>f^(s)$
  - 引理 2.2：在A结束前，必存在节点n，使得$f(n) ≤ f^(s)$

    > 引理 2.2 证明：存在一个节点 n，n 在 最佳路径上。
    > f(n) = g(n) + h(n)
    > = g(n)+h(n) ≤g(n)+h(n) = f(n)
    > = f*(s)
    > 得证。

  - 推论 2.1：OPEN 表上任一具有$f(n)<f(s)$的节点n，最终都将被A选作扩展的节点。

    > 推论 2.1 证明：由定理 2，知 A一定结束，由 *A*的结束条件，OPEN 表中 f(t)最小时才结束。而 f(t) ≥ f(t) = f(s) 所以 f(n)<f*(s)的 n，均被扩展。得证。

- 定理 3（可采纳定理）：
  - 若存在从初始节点s到目标节点 t 有路径，则$A*$必能找到最佳解结束
  - 可采纳性的证明：$由定理1、2知A一定找到一条路径结束，设找到的路径s→ t不是最佳的(t为目标)，则:f(t) = g(t) > f(s)；由引理2.2知结束前OPEN中存在f(n)≤f(s)的节点n，所以f(n) ≤ f(s) < f(t)，因此A应选择n扩展，而不是t，与假设A选择t结束矛盾，得证$
  - 推论 3.1：A选作扩展的任一节点n，有$f(n)≤f^(s)$。

    > 由引理 2.2 知在 A结束前，*OPEN* 中存在节点 *n’*， *f(n’)*≤*f*(s)
    > 设此时 A选择 *n* 扩展。
    > 如果 *n=n’*，则 *f(n)*≤*f*(s)，得证。
    > 如果 n≠ n’，由于 A选择 *n* 扩展，而不是 *n’*， 所以有 *f(n)* ≤ *f(n’)*≤*f*(s)。得证。

- 定理 4：如果 h2(n) > h1(n) (目标节点除外)，则 A1 扩展的节点数 ≥A2 扩展的节点数

> 定理 4 证明：
> 使用数学归纳法，对节点的深度进行归纳
> (1)当 d(n)=0 时，即只有一个节点，显然定理成立。
> (2)设 d(n)≤k 时定理成立。(归纳假设)
> (3)当 d(n)=k+1 时，用反证法。
> 设存在一个深度为 k+1 的节点 n，被 A2 扩展，但没有被 A1 扩展。而由假设，A1 扩展了 n 的父节点，即 n 已经被生 成了。因此当 A1 结束时，n 将被保留在 OPEN 中。
> 所以有:f1(n) ≥ *f(s)*
> 即:*g1(n)+h1(n)* ≥ *f*(s)
> 所以: h1(n) ≥ *f(s) - g1(n)*
> 另一方面，由于 *A2* 扩展了 *n*，有 *f2(n)* ≤ *f*(s)
> 即: h2(n) ≤ *f(s) – g2(n) (A)*
> 由于 *d(n)=k* 时，*A2* 扩展的节点 *A1* 一定扩展，有
> *g1(n)* ≤ *g2(n)* (因为 *A2* 的路 *A1* 均走到了)
> 所以:*h1(n)*≥*f*(s)-g1(n)≥f*(s)–g2(n) (B)
> 比较 A、B 两式，有 h1(n) ≥ h2(n) ，与定理条件矛 盾。故定理得证。

- 定理 5：若 h(n)是单调的，则 A*扩展了节点 n 之后，就已经找到了到达节点 n 的最佳路径。
- 定理 6：若 h(n)是单调的，则由 A*所扩展的节点序列其 f 值是非递减的。即 f(ni) ≤ f(nj)。
- 在修正的 A 算法中，fm 的含义是到当前为止，扩展的节点中，f 的最大值
- 对任意节点 n，设 m 是 n 的子节点，当 h 满足条件 h(n)-h(m) ≤ C(n, m), h(t) = 0 时，称 h 是单调的
- 用 A*算法求解问题时为什么会出现重复扩展节点问题，解决的方法有哪些?
  - 如果 h 函数定义的不合理，则当扩展一个节点时，不一定就找到了从初始节点到该节点的最优路径，对于这样的节点，就有可能被多次扩展。特别是如果这样的节点处于问题的最优解路径上时，则一定会被多次扩展
  - 使得 h 满足单调性；修正的 A*算法可以减少重复扩展节点问题

归结原理

- 定理证明方法，对机器定理证明问题起到了推动作用
- 化子句集
  1. 消蕴涵符 $A(x)→B(x) ≡ ~ A(x)∨B(x)$ $A(x)\to B(x) \Rightarrow \neg A(x) \cup B(x)$
  2. 移动否定符
     摩根定律:
     $ ~ (A(x)∨B(x)) ≡ ~ A(x)∧ ~ B(x)$ $\neg(A(x) \cup B(x)) \Rightarrow \neg A(x) \cap \neg B(x)$
     $ ~ (A(x)∧B(x)) ≡ ~ A(x)∨ ~ B(x)$ $\neg(A(x) \cap B(x)) \Rightarrow \neg A(x) \cup \neg B(x)$
     量词转换律表:
     $ ~ ((∃x)A(x)) ≡ (∀x)(~ A(x))$ $\neg((\exists x)A(x)) \Rightarrow (\forall x)(\neg A(x))$
     $ ~ ((∀x)A(x)) ≡ (∃x)(~ A(x))$ $\neg((\forall x)A(x)) \Rightarrow (\exists x)(\neg A(x))$
  3. 变量换名 $(∃x)A(x)∨(∃x)B(x) => (∃x)A(x)∨(∃y)B(y)$ $(\exists x)A(x) \cup (\exists x)B(x) \Rightarrow (\exists x)A(x) \cup (\exists y)B(y)$
  4. 量词左移 $(∃x)A(x)∨(∃y)B(y) => (∃x)(∃y){A(x)∨B(y)}$ $(\exists x)A(x) \cup (\exists y)B(y) \Rightarrow (\exists x)(\exists y)\{A(x) \cup B(y)\}$
     - 前束范式：所有量词均非否定的出现在公式的前部
     - 经过以上几步后，就将一个合式公式转化为了前束范式
  5. 消去存在量词（skolem 化）

$(∃z)(∀x)(∃y){[(~ P(x)∧ ~ Q(x))∨R(y)]∨U(z)} $
$\downarrow$
$(∀x){[(~ P(x)∧ ~ Q(x))∨R(f(x))]∨U(a)}$

6. 化为合取范式
   $(A(x)∧B(x))∧C(x) ≡ A(x)∧(B(x)∧C(x))$ } 结合律
   $(A(x)∨B(x))∨C(x) ≡ A(x)∨(B(x)∨C(x))$
   $A(x)∧(B(x)∨C(x)) ≡ (A(x)∧B(x))∨(A(x)∧C(x))$ } 分配律
   $A(x)∨(B(x)∧C(x)) ≡ (A(x)∨B(x))∧(A(x)∨C(x))$
7. 隐去全称量词

$(∀x){[~ P(x)∨R(f(x))∨U(a)]∧[~ Q(x)∨R(f(x))∨U(a)]}$
$\downarrow$
$[~ P(x)∨R(f(x))∨U(a)]∧[~ Q(x)∨R(f(x))∨U(a)]$

8. 表示为子句集

$[~ P(x)∨R(f(x))∨U(a)]∧[~ Q(x)∨R(f(x))∨U(a)]$
$\downarrow$
${~ P(x)∨R(f(x))∨U(a), ~ Q(x)∨R(f(x))∨U(a)}$

9. 变量换名
   ${~ P(x_1)∨R(f(x_1))∨U(a), ~ Q(x_2)∨R(f(x_2))∨U(a)}$

- 合一算法
  - 对于子句$C_1∨L_1$和$C_2∨L_2$，如果L_1与$~L_2$可合一，且s是其合一者，则$(C1∨C2)_s$是其归结式

传统人工智能考虑问题的方式

- OCR 光学字符识别 (Optical Character Recognition)

Game Theory Basics 博弈基础

博弈定义

- 研究智慧的理性决策者之间冲突与合作的数学模型

博弈要素

- 玩家 players：参与博弈的决策主体
- 策略 strategy：参与者可以采取的行动方案
  - 混合策略 mixed strategy：参与者可以通过一定概率分布来选择某确定的策略
  - 纯策略 pure strategy：参与者每次行动都选择某个确定的策略
  - 所有参与者各自采取行动后形成的状态被称为局势（outcome）
- 收益 payoff：各个参与者在不同局势下得到的利益
- 规则 rule：对参与者行动的先后顺序、参与者获得信息多少等内容的规定

博弈六特征 Features of Games

- 2 或多玩家
- 轮流 vs. 同时
- 完整信息 vs.不完整信息
- 确定的 vs. 随机的
- 合作 vs. 竞争
- 零和 vs. 非零和
  - 零和：一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加的总和永远为 0

> Example 5：石头剪刀布
> 2 & 同时 & 不完整 & 随机 & 竞争 & 零和

Dominant Strategy 占优策略

- 对于玩家 i，无论其他玩家做什么，如果策略 x 比策略 y 更好，策略 x 将主导策略 y
- 不满足**帕雷托效率 Pareto efficiency**
  - 在没有使任何人境况变坏的前提下，使得至少一个人变得更好

Nash Equilibrium 纳什均衡:

- 若任何参与者单独改变策略都不会得到好处，则该情形下的策略组合就是一个纳什均衡
- Dominant Strategy 是 Nash Equilibrium
- Nash 定理：有限博弈（参与者有限，每位参与者的策略集有限，收益函数为实值函数）必存在混合策略意义下的纳什均衡

> Example 6 囚徒困境

	Deny	Confess
Deny	$(-1, -1)$	$(-10, 0)$
Confess	$(0, -10)$	$\bold{(-8, -8)}$

Adversarial Search 对抗搜索（博弈搜索）

搜索 vs. 博弈

- 搜索: 非对抗性的
  - 解决方案：（启发式）发现目标的方法
  - 启发式和约束满足问题(CSP, Constraint Satisfaction Problem)技术->最佳解决方案
  - 评估函数：通过给定节点从开始到目标的成本估算
  - 示例：路径规划，安排活动
- 博弈：对抗性的
  - 解决方案：策略
  - 策略针对每个对手的可能反馈作出动作
  - 有限时间->近似解决方案
  - 评估函数：评估游戏位置的好坏
  - 例如：国际象棋，西洋跳棋，黑白棋，五子棋

对抗搜索的六部分形式化描述

- 初始状态 S: 游戏所处于的初始状态
- 玩家 $PLAYER$ (s): 在当前状态 s 下，该由哪个玩家采取行动
- 行动 $ACTIONS$ (s): 在当前状态 s 下所采取的可能移动
- 状态转移模型 $RESULT$ (s, a): 在当前状态 s 下采取行动 a 后得到的结果
- 终局状态检测 $TERMINAL-TEST$ (s): 检测游戏在状态 s 是否结束
- 终局得分 $UTILITY$ (s, p): 在终局状态 s 时，玩家 p 的得分

1、最小最大搜索(Minimax Search):

- 最小最大搜索是在对抗搜索中最为基本的一种让玩家来计算最优策略的方法

> **Function** MINIMAX(s) **returns** an action
> **if** TERMINAL-TEST(s) **then return** UTILITY(s)
> **if** PLAYER(s) = MAX **then return** $max{a∈ACTIONS(s)}$MINIMAX(RESULT(s, a))
> ***if*** *PLAYER(s) = MIN* ***then return*** *$min*{a∈ACTIONS(s)}$MINIMAX(RESULT(s, a))

2、$α-β$ 剪枝搜索(Pruning Search)

- 一种对最小最大搜索进行改进的算法，即在搜索过程中可剪除无需搜索的分支节点，且不影响搜索结果
- 极大节点的下界为 $α$，极小节点的上界为 $β$
- 剪枝的条件 $α≥ β$:
  - max节点的 $α$ 值 ≥ 后辈min节点的 $β$ 值时，$α$ 剪枝
  - min节点的 $β$ 值 ≤ 后辈max节点的 $α$ 值时，$β$ 剪枝
- 一般原则：若玩家是在位于n的父节点或更上层出有更好的选择m，则在实战中没必要抵达n

3、蒙特卡洛树搜索(Monte-Carlo Tree Search)

- 通过采样而非穷举方法来实现搜索
- Algorithm of AlphaGo
  - 深度神经网络 Deep Neural Networks
    - 价值网络 Value Networks: 评估
    - 策略网络 Policy Networks: 移动
  - 蒙特卡洛搜索 Monte-Carlo Tree Search (MCTS)
    - 结合蒙特卡洛模拟、价值网络与策略网络
  - 强化学习 Reinforcement Learning
- 蒙特卡洛方法：依靠重复随机采样来获得数值结果的计算算法
- 四个步骤创建决策树
  - 选择 $Selection$
  - 扩展 $Expansion$
  - 模拟 $Simulation$
  - 反向传播 $BackPropagation$

机器学习简介

定义

- Machine Learning is the study of algorithms that
  - 提升表现 P improve their performance P
  - 在某些任务 T at some task T
  - 通过经验 E with experience E.
  - A well-defined learning task is given by <P, T, E>.

机器学习算法分类

- 有监督学习 Supervised (inductive) learning: **training data $\And$ desired outputs (labels)**
  - 回归 Regression
  - 分类 Classification
  - 支持向量机 Support Vector Machines & Kernel Methods
  - 决策树 Decision Tree Induction
  - 贝叶斯学习 Bayesian Learning
  - 神经网络&深度学习 Neural Networks & Deep Learning
  - 学习理论 Learning Theory
- 无监督学习 Unsupervised learning: **training data**
  - 聚类 Clustering
  - 维度约减 Dimensionality reduction
- 半监督学习 Semi-supervised learning: **training data $\And$ a few desired outputs**
- 强化学习 Reinforcement learning
  - 给定一系列具有（延迟）奖励的状态和动作，输出策略；策略是状态到动作的映射
  - 时间差异学习 Temporal Difference Learning
  - Q 学习 Q Learning

机器学习算法组成

- 表示 Representation
  - 数值函数 Numerical functions
    - 线性回归 Linear regression
    - 神经网络 Neural networks
    - 支持向量机 Support vector machines
  - 符号函数 Symbolic functions
    - 决策树 Decision trees
    - 命题逻辑规则 Rules in propositional logic
    - 一阶谓词逻辑的规则 Rules in first-order predicate logic
  - 基于实例的函数 Instance-based functions
    - 最近邻 Nearest-neighbor
    - 基于案例 Case-based
  - 概率图模型 Probabilistic Graphical Models
    - 朴素贝叶斯 Naive Bayes
    - 贝叶斯网络 Bayesian networks
    - 隐马尔可夫链 Hidden-Markov Models (HMMs)
    - 概率上下文无关语法 Probabilistic Context Free Grammars (PCFGs)
    - 马尔可夫网络 Markov networks
- 优化 Optimization
  - 梯度下降 Gradient descent

        - 感知器 Perceptron
        - 反向传播 Backpropagation
    - 动态变成 Dynamic Programming 动态规化
        - 隐马尔可夫学习 HMM Learning
        - 概率上下文无关语法学习 PCFG Learning
    - 分而治之 Divide and Conquer
        - 决策树归纳 Decision tree induction
        - 规则学习 Rule learning
    - 进化计算 Evolutionary Computation
        - 遗传算法 Genetic Algorithms (GAs)
        - 遗传编程 Genetic Programming (GP)
        - 神经进化 Neuro-evolution
- 评价 Evaluation
    - 精度 Accuracy
    - 精确度和召回率 Precision and recall
    - 平方误差 Squared error
    - 相似性 Likelihood
    - 后验概率 Posterior probability
    - 成本/效用 Cost/Utility
    - 边距 Margin
    - 熵 Entropy
    - K-L 散度 K-L divergence

机器学习系统的设计

- 选择训练经验
- 选择学习对象，如目标函数
- 选择如何表示目标函数
- 选择学习算法从经验推断目标函数

机器学习实践

- 了解领域，先验知识和目标
- **数据集成，选择，清洁，预处理**
- 学习模型
- 解释结果
- 巩固和部署发现的知识

深度学习

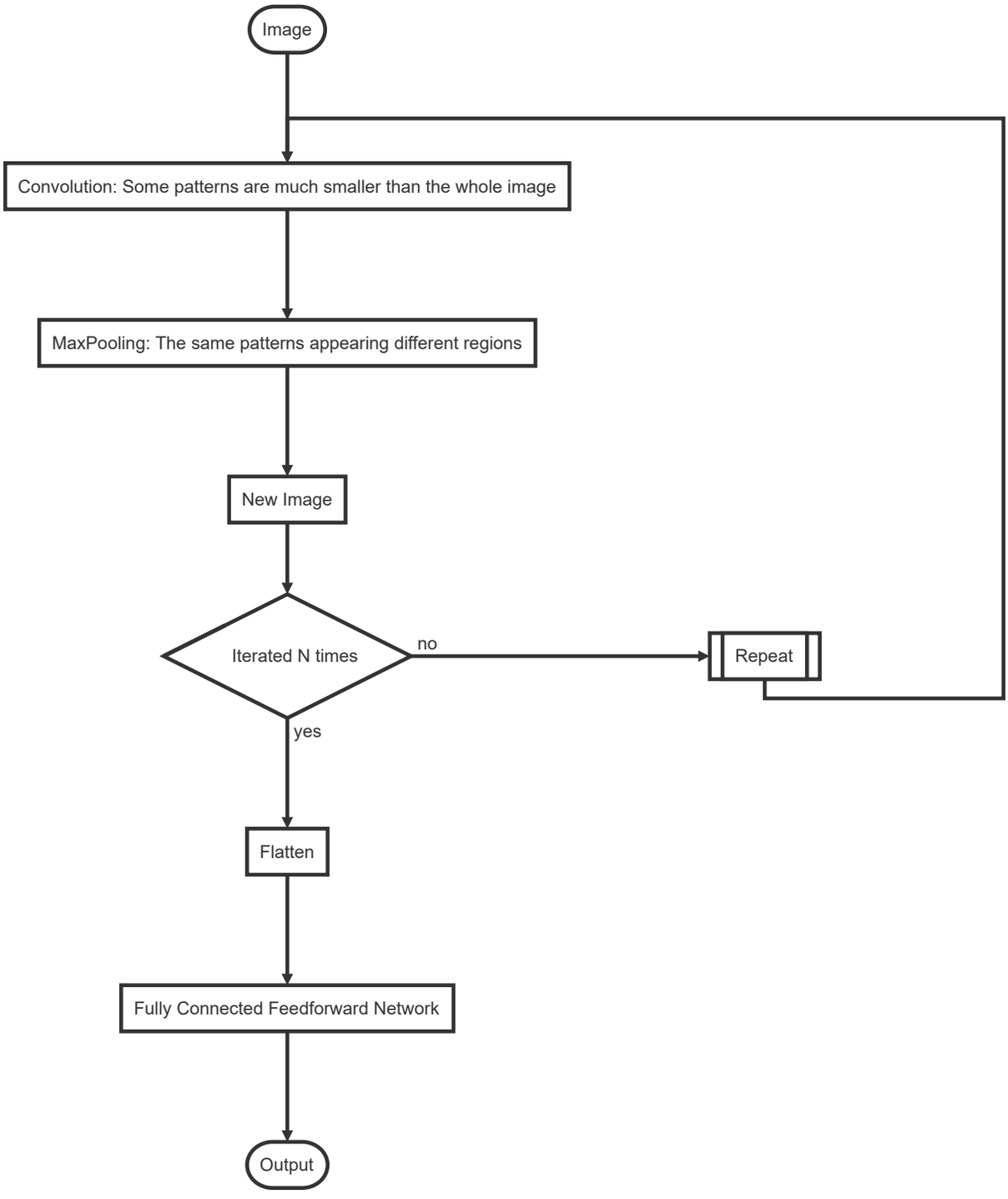
Three Steps for Deep Learning

1. Define a set of function
    - Network Structure
2. Goodness of function
    - Training Data
    - Learning Target
    - Total Loss
3. Pick the best function
    - Gradient Descent

Supervised Neural Network

> *e.g.* Fully Connect Feedforward Network

- Neuron, Weights, Bias, Activation Function (Sigmoid Function $sigma(z) = \frac{1}{1+e^{-z}}$), Output Layer (Softmax Layer)
- Convolutional Neural Network (CNN)
    - Convolution
        - Stride
        - Zero Padding
    - MaxPooling
    - Flatten



- Recurrent Neural Network (RNN): Neural Network with Memory

Unsupervised Neural Networks

- Embedding (word2vec)

生成模型 Generative Models

- 给定训练数据，从同一分布生成新样本
- 解决密度估计问题，这是无监督学习中的核心问题

主要方向

- 显式密度估计：显式定义并求解$p_{model}(x)$
- 隐式密度估计：非显示地定义模型，而是学习可从$p_{model}(x)$采样的模型

应用 Application

- 用于艺术品，超分辨率，着色的**真实采样**
- **时序数据**的生成模型可用于**仿真和计划**（强化学习应用程序）
- 推断潜在的一般**特征表示**

变分自动编码器 Variational Autoencoders (VAE)

- 无监督，用于学习未标注数据的低维特征表示

生成对抗网络 Generative Adversarial Networks (GAN)

- 不使用任何显式的密度函数
- 采取博弈论的方法：通过双人游戏，从训练分布中学习并生成
- 组成
    - 生成器网络 Generator Network：尝试通过生成逼真的图像来欺骗鉴别器
    - 鉴别器网络 Discriminator Network：尝试区分真实图像和伪造图像

Deep Learning on Graphs

Graph Applications

- Link Prediction
- Node Classification
- Node Importance
- Graph Classification

网络嵌入 Network Embedding

> De-coupling the links

- **Goal 1**：Reconstruct the original network
- **Goal 2**：Support network inference (Community detection, Network distance, Network evolution)

图神经网络 Graph Neural Networks

> Design new algorithms that can incorporate links

- Graph Recurrent Neural Networks: Recursive definition of states, e.g. `PageRank`
  - Most primitive methods
  - Unified frameworks with GCN
- Graph Convolutional Networks: Common local and global patterns
  - Convolutions: spectral, spatial
  - Readout
  - Improvements