- 图灵测试：理解其核心目的，即测试机器是否能展现出与人类无法区分的智能行为。
- 2、搜索算法
  - Slide 7 Page 6, Page 10-17
    - 第一部分：搜索算法分类与定义总结
      - 1. 搜索算法的分类
      - 2. Tree Search 与 Graph Search 的区别
    - 第二部分：A* 搜索与可采纳启发式详解
      - 1. A* 搜索算法核心
      - 2. 重点解答：可采纳启发式 (Admissible Heuristic)
- 3、人工智能中的规划技术
  - 1. 自动化规划的结构梳理
  - 2. 重点问题解析
    - Graphplan
    - STRIPS 规划（后向搜索中的相关动作）
    - 前向搜索与后向搜索
  - 3. STRIPS 规划具体实例：积木世界 (Blocks World)
    - 场景设定
    - STRIPS 动作定义
    - 规划过程演示 (状态转换)
- 4、不确定性决策与概率
  - 1. 期望效用 (Expected Utility)
  - 2. 独立事件概率 (Probability of Independent Events)
- 5、机器学习核心概念
  - 1. 偏差-方差权衡 (Bias-Variance Tradeoff)
  - 2. 学习范式 (Learning Paradigms) - 补充详情
    - 2.1 监督学习 (Supervised Learning)
    - 2.2 无监督学习 (Unsupervised Learning)
    - 2.3 强化学习 (Reinforcement Learning)
  - 3. 监督学习（回归问题）
    - 3.1 损失函数选择 (Loss Function Selection)
    - 3.2 目标函数 (Objective Function)
    - 3.3 梯度下降 (Gradient Descent)
- 6、约束满足问题（CSP）
  - 1. CSP定义 (Constraint Satisfaction Problems)
  - 2. 回溯算法 (Backtracking Algorithm)
  - 3. 回溯算法完整推导流程（结合课件案例）

- 7、数据科学与负责任的人工智能
  - 1. 数据偏差（Data Bias）
  - 2. 去偏方法（De-biasing Methods）
  - 3. 可解释人工智能（Explainable AI）

图灵测试：理解其核心目的，即测试机器是否能展现出与人类无法区分的智能行为。

Slide 1 Page20-36

1. 核心目的与定义

- **作为智能的操作性测试：** 图灵测试旨在为“智能”提供一个可操作的定义。艾伦·图灵（Alan Turing）在1950年的论文《计算机器与智能》中提出，与其纠结于抽象的哲学问题“机器能思考吗？”（Can machines think?），不如关注**“机器能否表现得像具备智能一样？”（Can machines behave intelligently?）**。
- **像人一样行动（Acting Humanly）：** 它是衡量机器是否具有**像人一样的行为能力**的标准。

2. 测试机制：模仿游戏（The Imitation Game）

- **设置：** 根据第20页的图示，测试包含三方：一个人类询问者（Human Interrogator）、一个人类（Human）和一个AI系统（AI System）。
- **过程：** 询问者在不可见的情况下与另外两者进行交流（通常通过文本）。
- **判断标准：** 如果人类询问者无法区分哪个是人类、哪个是机器，即机器成功“愚弄”了询问者，那么该机器就被认为表现出了与人类无法区分的智能行为。

3. 相关的预测与组成部分

- **预测：** 课件提到，图灵预测到2000年，机器可能有30%的几率在5分钟的测试中愚弄一个普通人。
- **所需能力：** 图灵指出，为了通过这个测试，AI需要具备以下主要组件：知识（knowledge）、推理（reasoning）、语言理解（language understanding）和学习（learning）。

2、搜索算法

Slide 7 Page 6, Page 10-17

第一部分：搜索算法分类与定义总结

1. 搜索算法的分类

根据 **Lecture 5 (Page 8)** 和 **Lecture 7 (Page 4)**：

- **无信息搜索 (Uninformed Search / Blind Search)**
  - **特点：** 不利用任何关于“目标在哪里”的额外信息，只利用问题定义中的信息（如起点、后继生成）。
  - **包括：**
    1. 广度优先搜索 (Breadth-First Search, BFS)
    2. 一致代价搜索 (Uniform-Cost Search, UCS)
    3. 深度优先搜索 (Depth-First Search, DFS)
    4. 深度受限搜索 (Depth-limited Search)
    5. 迭代加深搜索 (Iterative Deepening Search)
- **有信息搜索 (Informed Search / Heuristic Search)**
  - **特点：** 利用**启发式函数** $h(n)$ 来估计从当前节点到目标的距离，从而“智能”地选择更有希望的路径。
  - **包括：**
    1. 贪心最佳优先搜索 (Greedy Best-First Search)
    2. A* 搜索 (A* Search)

2. Tree Search 与 Graph Search 的区别

根据 **Lecture 6 (Page 8)** 和 **Lecture 5 (Page 15-16)**：

- **Tree Search (树搜索)**
  - **定义：** 在搜索过程中维护一个边缘集（Fringe/Frontier），**不记录**已探索过的状态。
  - **缺点：** 遇到重复状态或环路时，会重复扩展，可能导致死循环或效率极其低下（搜索树可能无限大）。
- **Graph Search (图搜索)**

- **定义：** 在树搜索的基础上，增加一个 **Closed List (已探索集合)**。
- **机制：** 每当要扩展一个节点时，先检查它是否在 Closed List 中。如果在，直接丢弃；如果不在，才加入 Closed List 并扩展。
- **优点：** 防止重复处理相同状态，避免死循环。

第二部分：A* 搜索与可采纳启发式详解

1. A* 搜索算法核心

- **评价函数：** $f(n) = g(n) + h(n)$
  - $g(n)$：从起点到当前节点 n 的**实际**耗费。
  - $h(n)$：从节点 n 到目标节点的**估计**耗费（启发式）。
  - $f(n)$：经过节点 n 到达目标的**估计总耗费**。
- **出处：** **Lecture 7, Page 6**。

2. 重点解答：可采纳启发式 (Admissible Heuristic)

您问到：**为什么要“启发式估计值 $h(n)$ 永远不会超过从 n 到目标的实际最小代价”？它的重要性是什么?**

我们可以结合 **Lecture 7, Page 10 (定义)** 和 **Page 17 (证明逻辑)** 来解答。

A. 为什么要“不该高估”？（通俗理解） 想象你在地图上找路。

- **如果 $h(n)$ 高估了 (Overestimates)：** 你可能会看着一条明明很好的路，觉得它“太远了”（评估出来的总代价 $f(n)$ 很大），于是你放弃了这条路，转而去走另一条其实更远、但看起来似乎近一点的冤枉路。这就导致你错过了最优解。
- **如果 $h(n)$ 低估了 (Underestimates) 或准确：** 你会觉得这条好路“很有希望”，A* 算法就会优先去探索它。只要所有的估计都比实际值小（或者相等），A* 就绝不会因为“误以为太贵”而漏掉真正的最短路径。

B. 定义与数学表达

- **定义：** 一个启发式函数 $h(n)$ 是可采纳的，当且仅当对于所有节点 n:

  $$h(n) \leq h^*(n)$$

  其中 $h^*(n)$ 是从 n 到目标的**真实**最小代价。
- **出处：** **Lecture 7, Page 10** ("A heuristic is admissible... if it never overestimates the cost to reach the goal")。

C. 重要性：如何保证最优性（Optimality） 这是 A* 算法最核心的性质。根据 **Lecture 7, Page 17** 的逻辑：

1. **假设：** 假如 A* 找到了一个**次优目标** G（即找到了一条路，但不是最短的），其耗费为 $g(G)$。同时，真正的**最优解耗费为** C^*。那么显然 $g(G) > C^*$。
2. **存在性：** 在 A* 结束前，边缘集（Frontier）里一定还躺着某个节点 n，这个节点位于通往**真正最优解**的路径上。
3. **关键的不等式：**
   - 因为 h 是可采纳的（不通过高估），所以对于这个优良节点 n：$f(n) = g(n) + h(n) \leq g(n) + h^*(n) = C^*$。
   - 也就是说，**最优路径上的节点，其评估值** $f(n)$ **永远不会超过最优解的总代价** C^*。
4. **矛盾：**
   - 我们前面说了，次优目标 G 的评估值 $f(G) = g(G)$（因为到目标了 $h = 0$）。
   - 已知次优解比最优解差：$f(G) > C^*$。
   - 结合第3点：$f(n) \leq C^* < f(G)$。
5. **结论：** A* 算法总是优先扩展 f 值最小的节点。因为 $f(n) < f(G)$，所以 *A 永远会先去扩展那个通往最优解的节点 n，而不会先去选择那个次优的目标 G。**

总结： “$h(n)$ 不超过实际代价”的重要性在于：*它给 A 吃了一颗定心丸，确保算法绝不会因为“误判”而过早地终止在一条非最优的路径上，从而在树搜索（Tree Search）中严格保证找到最优解。**

3、人工智能中的规划技术

- Graph_Plan*(出处：Lecture 26, Slide 24 "Comparison with Plan-Space Planning")*
- STRIPS 规划*(出处：Lecture 27, Slide 16 "Backward Search")*
- 前向搜索*(出处：Lecture 27, Slide 14-15)*
- 后向搜索*(出处：Lecture 27, Slide 16-19)*

1. 自动化规划的结构梳理

自动化规划的知识体系可以分为**问题表示（Representation）**和求解算法（Algorithms）**两大类：

- **一、规划问题的表示 (Planning Problem Representation)**

- **STRIPS (Stanford Research Institute Problem Solver):**
  - 基础语言，使用受限的语言来保证算法效率。
  - 由状态（初始状态、目标状态）和动作模式（Action Schemas）组成。
  - 动作包含：前置条件（Preconditions）和效果（Effects，通常分为添加列表和删除列表）。
- **ADL (Action Description Language):**
  - 比 STRIPS 更丰富，允许负文字、开放世界假设、量词（forall/exists）、条件效果等。
- **PDDL (Planning Domain Definition Language):**
  - 现代标准语言，核心基于 STRIPS 和 ADL。
  - 文件结构分为：**Domain File**（定义谓词和动作/操作符）和 **Problem File**（定义对象、初始状态和具体目标）。

- **二、规划算法 (Planning Algorithms)**
  - **1. 状态空间搜索 (State-Space Search):**
    - **前向搜索 (Forward Search / Progression):** 从初始状态向目标搜索。
    - **后向搜索 (Backward Search / Regression):** 从目标向初始状态反向搜索。
  - **2. 规划图与Graphplan (Planning Graphs and Graphplan):**
    - 一种基于图结构的算法，用于减少搜索空间。
    - **规划图 (Planning Graph):** 交替的“状态层”和“动作层”，包含互斥关系 (Mutex)。
    - **解提取 (Solution Extraction):** 在图建立后进行反向搜索以提取计划。
  - **3. 启发式搜索 (Heuristic Search):**
    - 利用松弛问题 (Relaxed Problem)（如忽略前置条件）来自动生成启发式函数（如 A* 算法），指导搜索过程。
  - **4. 计划空间规划 (Plan-Space Planning, PSP):** (课件中作为对比提到)
    - 搜索的是部分计划的空间，而不是世界状态的空间。

---

2. 重点问题解析

Graphplan

- **优缺点：**
  - **优点：** 相比于传统的计划空间规划（PSP），Graphplan 速度非常快。它通过预先构建规划图，有效地排除了许多不可能的路径，从而显著缩小了搜索空

间。
    - **缺点：** 在生成规划图时，会产生大量的**基原子（ground atoms）**。它会尝试实例化所有可能的动作和状态参数组合，导致图变得非常巨大，包含大量现实中无关的节点。
- **缓解方法（数据类型）：**
    - 通过为变量和常量分配**数据类型（Data Types）**可以部分缓解此问题。
    - **原理：** 实例化变量时，强制要求变量只能绑定到相同类型的对象上。例如，动作 `Pickup(obj, room)` 中，若规定 `obj` 必须是物品类型，算法就不会生成 `Pickup(RoomA, RoomB)` 这样无意义的基原子，从而减少图的规模。
    - *(出处：Lecture 26, Slide 24)*

STRIPS 规划（后向搜索中的相关动作）

- **核心逻辑：**
    - 后向搜索从目标状态开始回溯。
    - **相关动作（Relevant Actions）：** 指那些**效果（Effect）能实现目标中某个未满足条件（conjunct）**的动作。
- **为何能降低复杂度：**
    - 在前向搜索中，任何可执行的动作都要尝试，导致分支因子（Branching Factor）极大（例如几千个航班）。
    - 在后向搜索中，只考虑那些能直接“贡献”于目标的动作，**分支因子急剧下降**（例如能把货物卸在目的地的动作可能只有几十个）。这使得搜索更加聚焦。
    - *(出处：Lecture 27, Slide 16)*

前向搜索与后向搜索

- **前向搜索 (Forward Search):**
    - **定义：** 从**初始状态**出发，正向应用满足前置条件的动作，生成后继状态，直到遇到满足目标的状态。
    - **公式理解：** 新状态 $S' = (S \setminus \mathrm{Del}(a)) \cup \mathrm{Add}(a)$（减去动作的负效果，加上正效果）。
    - *(出处：Lecture 27, Slide 14-15)*
- **后向搜索 (Backward Search):**
    - **定义：** 从**目标描述**出发，通过**“回归（Regressing）”**操作逆向选择动作，生成前驱子目标，直到子目标被初始状态满足。
    - **公式理解：** 前驱子目标 $G' = (G \setminus \mathrm{Effects}^{+}(a)) \cup \mathrm{Precond}(a)$（减去动作实现的目标，加上动作需要的前置条件）。
    - *(出处：Lecture 27, Slide 16-19)*

---

3. STRIPS 规划具体实例：积木世界 (Blocks World)

使用经典的“积木世界”作为例子（基于 *Lecture 26, Slide 21-22* 和 *Lecture 27, Slide 21*）。

场景设定

假设桌子上有一些积木，机械臂可以移动它们。

- **谓词 (Predicates):**
  - `On(x, y)`: x 在 y 上面。
  - `Block(x)`: x 是一个积木。
  - `Clear(x)`: x 上面没有东西。
- **动作模式 (Action Schema):** `Move(b, x, y)` —— 把积木 b 从 x 移动到 y 上。

STRIPS 动作定义

在 STRIPS 中，动作由**前置条件 (Preconditions)** 和 **效果 (Effects)** 定义：

```
Action: Move(b, x, y)
  Preconditions:
    On(b, x)    ^  // b 目前在 x 上
    Clear(b)    ^  // b 上面没有东西（可以抓取）
    Clear(y)    ^  // y 上面没有东西（可以放下）
    Block(b)    ^  Block(y) // 都是积木
    (b ≠ x) ^ (b ≠ y) ^ (x ≠ y) // 变量不相等

  Effects:
    On(b, y)    ^  // 添加：b 现在在 y 上了
    Clear(x)    ^  // 添加：x 现在空出来了
    ¬On(b, x)   ^  // 删除：b 不再在 x 上了
    ¬Clear(y)      // 删除：y 不再是空的了
```

规划过程演示 (状态转换)

假设我们想把积木 **A** 从 **Table (桌子)** 移动到积木 **B** 上。

1. **初始状态 (Initial State, S_0):**
   - `{On(A, Table), On(B, Table), Clear(A), Clear(B), Block(A), Block(B)}`

- *解读：A和B都在桌子上，且它们上面都是空的。*

2. **执行动作 (Action):** `Move(A, Table, B)`
   - **检查前置条件：**
     - `On(A, Table)`? 是 (在 S_0 中)。
     - `Clear(A)`? 是。
     - `Clear(B)`? 是。
     - 条件满足，动作可执行。
3. **应用效果 (生成新状态 S_1):**
   - **删除列表 (Delete List - 负效果):** 移除 `On(A, Table)` 和 `Clear(B)`。
   - **添加列表 (Add List - 正效果):** 加入 `On(A, B)` 和 `Clear(Table)` (假设桌子总是Clear的特殊处理，或者此处简化为x变为Clear)。
   - **新状态 (S_1):** `{On(A, B), On(B, Table), Clear(A), Block(A), Block(B), ...}`

4、不确定性决策与概率

- 期望效用*(出处：Lecture 19, Page 9-10)*
- 独立事件概率*(出处：Lecture 19, Page 22, 25, 30)*

---

1. 期望效用 (Expected Utility)

- **定义与计算方法：**
  - **定义：** 期望效用是将**概率论**（Probability Theory）与**效用理论**（Utility Theory）相结合的概念。它反映了在不确定环境中，一个理性智能体（Rational Agent）对未来可能结果的平均满意度预估。
  - **计算原则：** 理性智能体遵循**最大期望效用原则（MEU, Maximum Expected Utility)**。这意味着智能体应选择那个能产生最高平均效用的行动。
  - **计算逻辑：** 虽然课件未列出具体的 Σ 数学算式，但文字描述了计算逻辑：即对行动所有可能结果的效用值进行加权平均（averaged over all the possible outcomes），权重为该结果发生的概率。
- **课件中的演示过程 (决策论智能体逻辑)：** 在 **Page 10** 中，课件通过 `DT-Agent` (Decision-Theoretic Agent) 函数演示了基于期望效用进行决策的真实过程：

1. **更新信念 (Update Belief State)：** 基于当前的感知（percept）和行动，更新对当前世界状态的概率信念。
2. **计算概率 (Calculate Probabilities)：** 给定当前信念状态和行动描述，计算各种可能结果发生的概率。
3. **选择行动 (Select Action)：** 结合结果的概率和效用信息（Utility information），选择那个能带来**最高期望效用**的行动。
4. **执行：** 返回该行动。

---

2. 独立事件概率 (Probability of Independent Events)

- **定义与计算方法：**
  - **定义：** 如果两个事件 a 和 b 是相互独立的（即一个事件的发生不影响另一个事件的发生概率），那么它们同时发生的联合概率（Joint Probability）等于它们各自边缘概率的乘积。
  - **计算公式：** $P(a \wedge b) = P(a) \times P(b)$。
- **课件中的演示过程 (抛硬币例子)：** 课件使用了“连续抛两次硬币”的例子来演示这一计算过程。

  示例场景：不均匀/灌铅硬币 (Loaded Coin)

  - **出处：** **Lecture 19, Page 30**
  - **设定：** 这是一个特制的硬币，抛出正面的概率是 0.6，反面的概率是 0.4。
    - $P(H) = 0.6$
    - $P(T) = 0.4$
  - **问题：** 两次都抛出正面（Heads, Heads）的概率是多少？即求 $P(H, H)$。
  - **计算演示：** 由于两次抛掷是独立事件，课件展示了如下计算：

$$P(H, H) = P(H) \times P(H)$$

$$0.36 = 0.6 \times 0.6$$

  - **结果：** 两次都抛出正面的概率为 **0.36**。

5、机器学习核心概念

- Lecture 37: Introduction to Machine Learning
  - Slide 24 (PDF 第66页)：定义了偏差、方差以及权衡。
  - Slide 21 (PDF 第57-59页)：曲线拟合的例子。
  - Slide 22 (PDF 第60-63页)：奥卡姆剃刀与拟合度/复杂度的关系图。
- Lecture 37: Introduction to Machine Learning：
  - Slide 15-17 (PDF 第40-46页)：定义监督学习和无监督学习及应用。；
  - Slide 13 (PDF 第36页)：列举了三种学习场景。
- Lecture 44: Reinforcement Learning - I：
  - Slide 9 (PDF 第20-21页)：定义马尔可夫决策过程（MDP）和强化学习的目标。
- Lecture 39: Machine Learning – Regression II, Slide 9 (PDF 第10页)。

1. 偏差-方差权衡 (Bias-Variance Tradeoff)

- **详细解释与例子：**
  - **核心思想：** 在选择模型时，我们不能只关注模型在训练集上的表现。
    - **偏差 (Bias)** 指的是假设（模型）偏离不同训练集期望的程度。高偏差通常意味着**欠拟合 (Underfitting)**，即模型太简单，无法捕捉数据的特征。
    - **方差 (Variance)** 指的是当数据发生变化时，假设（模型）发生改变的程度。高方差通常意味着**过拟合 (Overfitting)**，即模型太复杂，对训练数据中的噪声也进行了学习。
  - **权衡 (Tradeoff)：** 我们需要在训练集拟合得好（低偏差）和在测试集表现好（低方差）之间做出选择。
  - **课件例子：**
    - 在 **Slide 21** 中展示了一个“曲线拟合”的例子。
      - 如果在这些点上画一条直线（Linear），虽然简单，但无法穿过所有点，这就是**欠拟合（高偏差）**。
      - 如果用一条非常复杂的曲线（如高阶多项式）连接每一个点，虽然在训练集上误差为0，但它不仅学到了规律，还学到了噪声，这就是**过拟合（高方差）**。
    - 在 **Slide 22** 的图表中，展示了随着模型“复杂度 (Complexity)”的增加，训练数据的误差（黑线）会一直下降，但泛化误差/测试误差（红线/蓝线）会先降后升。最佳的平衡点是泛化误差最低的那个点，而不是训练误差最低的点。

---

2. 学习范式 (Learning Paradigms) - 补充详情

这三种范式的本质区别在于**数据中包含的信息量（是否有标签）**以及**反馈机制（Feedback Mechanism）**的不同。

2.1 监督学习 (Supervised Learning)

- **适用场景 (Scenarios):**
  - **预测任务：** 当你需要根据已知特征预测一个特定的结果时。
  - **分类 (Classification):** 预测离散的类别。
    - *例子:* 垃圾邮件过滤（Spam vs Not Spam）、诊断病人是否患有糖尿病。(**Lecture 37, Slide 18**)
  - **回归 (Regression):** 预测连续的数值。
    - *例子:* 预测房价（House Prices）、根据司机年龄预测汽车维修费用。(**Lecture 37, Slide 15**)
- **如何区分 (Distinction):**
  - **特征：** 训练数据由**输入特征 (x)** 和对应的**正确答案/标签 (y)** 组成。即数据是成对出现的 (x, y)。
  - **出处： Lecture 37, Slide 15 & 16** ("Labelled examples").
- **为什么 (Why - 核心逻辑):**
  - **目标：** 学习一个函数映射 $f(x) \rightarrow y$。
  - **原因：** 既然我们已经知道了一部分数据的“正确答案”，算法的目标就是通过最小化预测值与真实标签之间的误差（Loss），来找到一种规律，从而能对**未来未见过的数据**做出准确预测。

2.2 无监督学习 (Unsupervised Learning)

- **适用场景 (Scenarios):**
  - **发现结构/聚类：** 当你有一堆数据，但不知道它们代表什么，或者不知道它们之间有什么关系时。
  - *例子:* **市场细分 (Market segmentation)**（自动将客户分组）、社交网络分析（发现社交圈子）、整理计算机集群、将新闻文章按主题归类。(**Lecture 37, Slide 17 & 18**)
- **如何区分 (Distinction):**
  - **特征：** 训练数据**只有输入特征 (x)，没有标签 (y)**。数据是未标记的 (Unlabeled)。
  - **出处： Lecture 37, Slide 16** ("Unlabeled examples").
- **为什么 (Why - 核心逻辑):**

- **目标：** 发现数据内部的**隐藏结构 (Hidden Structure)** 或概率分布 $P(X)$。
- **原因：** 因为没有老师告诉算法什么是“对”的，算法必须自己去寻找数据点之间的相似性。例如，它不知道这群人是“高消费人群”，但它能发现这群人的消费行为非常相似，从而将其归为一类。

2.3 强化学习 (Reinforcement Learning)

- **适用场景 (Scenarios):**
  - **序贯决策 (Sequential Decision Making):** 需要在连续的时间步中做出一系列决定，而不是一次性的预测。
  - *例子:* **网格世界 (Grid World)** 导航、机器人控制、下棋（Chess/Backgammon）、电梯控制。 (**Lecture 37, Slide 8**; **Lecture 44, Slide 6**)
- **如何区分 (Distinction):**
  - **特征：** 数据不是静态的 x 和 y，而是**状态 (State) - 动作 (Action) - 奖励 (Reward)** 的交互序列。
  - **反馈机制：** 没有直接的“标签”告诉每一步是对是错，而是通过**偶尔的奖励 (Occasional Rewards)** 或惩罚作为延迟反馈。
  - **出处： Lecture 37, Slide 12** ("occasional rewards"); **Lecture 44, Slide 9** (MDP定义).
- **为什么 (Why - 核心逻辑):**
  - **目标：** 学习一个**策略 (Policy, π)**，即在什么状态下采取什么动作。
  - **原因：** 在复杂环境中（如走迷宫或下棋），告诉每一步绝对的“正确答案”是不现实的。因此，算法必须通过**试错 (Trial and Error)** 和**探索 (Exploration)**，根据最终的结果（比如赢了棋或撞墙了）来反推之前的哪些动作是好的，从而最大化**长期的累积奖励**。

---

3. 监督学习（回归问题）

3.1 损失函数选择 (Loss Function Selection)

- **解释与例子：**
  - 课件明确对比了线性回归（Linear regression）和逻辑回归（Logistic regression/分类）的损失函数。
  - 对于**回归问题**（预测连续值，如房价），课件展示的损失函数是**均方误差**（Mean Squared Error, MSE）的形式：

$$Loss(w) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_w(x^{(i)}) - y^{(i)})^2$$

- 这里 $(h_w(x^{(i)}) - y^{(i)})^2$ 就是预测值与真实值之差的平方。课件指出这与用于分类（Logistic regression）的对数损失（Cross-entropy）不同。

3.2 目标函数 (Objective Function)

- **解释与例子：**
  - **目标：** 找到一组参数 w（例如 w_0, w_1），使得在所有训练样本上的平均损失最小化。
  - **公式：** 课件中写道：

$$\text{Goal: } \min_{w_0, w_1} Loss(w_0, w_1)$$

  - 这意味着我们的优化目标是让预测出的房价曲线与真实房价数据点之间的总距离（误差平方和的平均）尽可能小。

3.3 梯度下降 (Gradient Descent)

- **解释与例子：**
  - **概念：** 梯度下降是一种用于最小化损失函数的优化算法。
  - **梯度：** 课件在 **Slide 12** 展示了损失函数对参数 w 的偏导数 $\frac{\partial}{\partial w} Loss(w)$。这个偏导数代表了损失函数在当前参数位置的斜率或变化率。
  - **更新方向：** 课件 **Slide 11** 通过图示解释了导数行为：
    - 如果导数为正（斜率向上），我们需要减去这个值（$w - \alpha \times \text{positive}$）让 w 变小，从而向左移动到谷底。
    - 如果导数为负（斜率向下），减去负值等于加上正值，让 w 变大，向右移动到谷底。
  - **可视化例子：** **Slide 6** 展示了一个三维地形图（3D Plot），梯度下降就像是从山顶沿着最陡峭的下坡方向一步步走到山谷最低点（最小损失点）。

6、约束满足问题（CSP）

- **出处：**
  - **课件：** Lecture 13: Constraint Satisfaction Problems I
  - **页码：** **Slide 7** (标题: Defining Constraint Satisfaction Problems)
- **出处：**

- **课件：** Lecture 14: Constraint Satisfaction Problems II
- **页码：**
  - **Slide 6** (原理): 提到 "Depth-first search... is called backtracking search"。
  - **Slide 7** (伪代码): 展示了 `Backtrack` 函数的递归与撤销逻辑 (`remove {var = value}`)。
  - **Slide 8-11** (可视化): 展示了澳大利亚地图着色的搜索树过程。

---

1. CSP定义 (Constraint Satisfaction Problems)

- **定义解答：** 约束满足问题 (CSP) 是一个标准化的搜索问题框架，由三个核心部分组成：
  1. X **(Variables)**：一组**变量**，即 $\{X_1, \ldots, X_n\}$。
  2. D **(Domains)**：一组**域**，即 $\{D_1, \ldots, D_n\}$，每一个变量都有对应的取值范围（例如：红、绿、蓝）。
  3. C **(Constraints)**：一组**约束**，用于指定变量之间允许的值的组合（例如：相邻变量不能取相同颜色）。
  - **目标：** 找到一个**完整赋值 (Complete Assignment)**，即为每个变量赋一个值，且满足所有约束条件。

---

2. 回溯算法 (Backtracking Algorithm)

- **核心思想：**
  - 回溯搜索本质上是一种**深度优先搜索 (DFS)**。
  - 它采用**单变量赋值 (Single-variable assignments)** 的策略：在搜索树的每一层，只为一个变量选择一个值。
  - **核心逻辑：** 递归地为变量赋值。如果当前赋值满足约束，则继续递归；如果发现当前路径无法找到解（即某个变量无合法取值），则**撤销上一步赋值（回溯）**，尝试下一个可能的值。

---

3. 回溯算法完整推导流程（结合课件案例）

案例背景 (Lecture 14, Slide 8-11)：

- **问题：** 对澳大利亚地图进行着色。
- **变量 X：** $\{WA, NT, Q, NSW, V, SA, T\}$
- **域 D：** $\{Red, Green, Blue\}$ (红, 绿, 蓝)
- **约束 C：** 相邻区域不能同色 (例如 $WA \neq NT$)。

推导流程演示：

我们按照 **Lecture 14 Slide 7** 的算法逻辑进行推导：

1. **初始状态：**
    - 赋值集合 `assignment = {}`。
    - 选择第一个未赋值变量：**WA** (Western Australia)。
2. **第一层递归 (Assign WA)：**
    - 尝试值 `Red`。
    - 检查约束：WA 周围没有已赋值邻居，**一致 (Consistent)**。
    - 更新赋值：`assignment = {WA = Red}`。
    - **向下递归...**
3. **第二层递归 (Assign NT)：**
    - 选择变量：**NT** (Northern Territory)。
    - 尝试值 `Red`：与 $WA = Red$ 冲突 $\rightarrow$ **失败**。
    - 尝试值 `Green`：与 $WA = Red$ 不冲突 $\rightarrow$ **一致**。
    - 更新赋值：`assignment = {WA = Red, NT = Green}`。
    - **向下递归...**
4. **第三层递归 (Assign Q)：**
    - 选择变量：**Q** (Queensland)。
    - 尝试值 `Red`：与 $NT = Green$ 不冲突 $\rightarrow$ **一致**。
    - 更新赋值：`assignment = {WA = Red, NT = Green, Q = Red}`。
    - *(注意：这里我们选择了一条路径，假设继续往下走)*
    - **向下递归...**
5. **第四层递归 (Assign SA - 触发回溯的关键点)：**
    - 选择变量：**SA** (South Australia)。
    - **约束检查：** SA 的邻居有 WA(Red), NT(Green), Q(Red)。

- **尝试值域 D 中的所有可能：**
  - 尝试 `Red`：与 WA(Red) 和 Q(Red) 冲突 → **失败**。
  - 尝试 `Green`：与 NT(Green) 冲突 → **失败**。
  - 尝试 `Blue`：假设后续的 NSW 或 V 已经被赋值导致 Blue 也不可用（为了演示回溯，假设此处无解），或者如果在实际推导中 SA 是 Blue，我们继续看下一层。
- **假设死胡同 (Dead End)：** 如果在当前路径下，SA 的所有颜色都尝试完毕且都冲突（返回 `failure`）。

6. **回溯 (Backtracking)：**

- 算法执行 **Slide 7 第15-16行** 的逻辑。
- **撤销操作：** 从 `assignment` 中移除上一步的赋值 `remove {Q = Red}`。
- **返回上一层：** 回到 **第三层递归 (Q)**。

7. **重新尝试 (Retry at Q)：**

- 在 Q 的域中尝试下一个值：`Green`。
- 检查：与 $NT = Green$ 冲突 → **失败**。
- 尝试下一个值：`Blue`。
- 检查：与 $NT = Green$ 不冲突 → **一致**。
- 更新赋值：`assignment = {WA = Red, NT = Green, Q = Blue}`。
- **再次向下递归 (Assign SA)...**

8. **继续搜索：**

- 现在 SA 的邻居是 WA(Red), NT(Green), Q(Blue)。
- SA 无法取 Red, Green, Blue 中的任何一个（因为它与这三个都相邻）。
- **再次回溯：** 撤销 `Q=Blue`，撤销 `NT=Green`... 直到找到一个可行的分支。

7、数据科学与负责任的人工智能

- **出处： Lecture 46: Ethics in AI - I, Page 14** (Fairness and bias (3))
- **出处： Lecture 46: Ethics in AI - I, Page 14** (Fairness and bias (3))
- **出处： Lecture 47: Ethics in AI II, Page 17** & **Page 18** (Trust and transparency)

1. 数据偏差（Data Bias）

- **内容：** 课件明确指出了**样本量差异（Sample size disparity）会导致偏差结果。具体表现为：“In most data sets there will be fewer training examples of minority class”（在大多数数据集中，少数类别的训练样本会较少），这指的就是类别不平衡**问题。此外，机器学习算法通常在拥有更多训练数据的情况下准确率更高，这意味着少数类别的成员将体验到较低的准确率。

2. 去偏方法（De-biasing Methods）

- **内容：** 课件在最后一项中提出了具体的去偏方法：“**De-bias the data: over-sample from minority classes to defend against sample size disparity**”（对数据进行去偏：对少数类别进行**过采样**，以防御样本量差异带来的影响）。

3. 可解释人工智能（Explainable AI）

- **内容：**
  - **Page 17** 定义了：“An AI system that can explain itself is called explainable AI (XAI)”（一个能够解释自身的AI系统被称为可解释AI）。
  - **Page 18** 进一步阐述了一个好的解释应具备的属性：“**it should be understandable and convincing to the user**”（它应当对用户来说是**可理解**且令人信服的），这与您描述的目标一致。