CS181 人工智能I

Lec1 课程介绍

- 给分标准：
  - 作业(**10%**) 6次，在 Blackboard 上在线完成，无限次数提交
  - 编程作业(**25%**) 总共6个 Project，内容为 UC Berkeley CS188 Pacman 的改编
  - 合作Project(**15%**) 在后半学期，1-5人组队，五月中旬之前完成组队，在 5.18、5.20 会进行 proposal presentation，第18周 7.3 下午进行最终 presentation 并提交；必须使用课内讲过的内容；不要过于复杂；游戏Agent / 解决现实问题 / 实现不同机器学习方法比较优劣；评分标准：课程相关、合理性和完备性、报告和最终pre质量
  - 期中考试(**25%**) 4.20，闭卷，1张双面 A4-Cheatsheet；10不定选、4计算/简答；考到贝叶斯网络采样结束；带涂卡笔，不能带计算器
  - 期末考试(**25%**) 不涉及期中之前内容，略难于期中，1张双面A4-Cheatsheet，17周周一 6.22，10多选，5简答；带涂卡笔，允许带计算器
- 查重范围包括Github上的作业
- 课件传bb，有Piazza，有Gradescope
- 编程作业有总共5天迟交机会，使用时需要在提前5天内和所有助教、教授邮件通知

3种AI的方法

- **符号主义 Symbolism**
  - 用符号和表达式来表示知识
  - 在1950-1980年占主导，在1980-1990逐渐退出历史舞台，在2000-2010和统计学方法结合，在2010-2020和神经网络方法结合
  - 可解释性强
  - 难以学习，较为僵硬
- **连接主义 Connectionism**
  - 用简单单元的连接网络来表示知识
  - 1940年代开始发展，1970年AI寒冬，1980年被重新发现，1990-2000年被其它方法超越，2010年深度学习开始兴起，2012年统治了计算机图形学，2015年统治了自然语言处理(NLP)
  - 表现较好，适应性强
  - 黑箱，可解释性差
- **统计学方法 Statistical Approaches**
  - 用概率模型表示知识
  - 从1990年开始进入大众视野，在2000年占支配地位，在2010年后被深度学习超越
  - 可解释性强，可学习
  - 适应性差

Lec2 搜索

搜索问题的定义

- 状态空间 State space
- 后继函数 Successor function
- 起始状态和目标测试 Start state and goal test

搜索问题的分类

- 提示性搜索 Informed Search: Greedy, A*
- 盲目搜索 Uninformed Search: DFS, BFS, UCS

搜索算法

深度优先搜索 DFS

- 对于m层b叉树，时间复杂度 $O(b^m)$
- 空间复杂度 $O(bm)$
- 对于可避免循环的方法（深度有限），是完备的（一定能找到一个目标状态）
- 不一定第一次找到最优解

广度优先搜索 BFS

- 对于目标状态在第s层的b叉树，时间复杂度 $O(b^s)$
- 空间复杂度 $O(b^s)$
- 完备
- 仅对于所有边权相等的情况下，一定能找到最优解

均匀成本搜索 UCS (可理解为Dijkstra)

- 总是展开到目前为止所有未到达节点中，从根节点到达的总长度最小的节点
- 如果该解决方案的成本为C^*，而每条边的成本至少为ϵ，那么 “有效深度”大致为 $\frac{C^*}{\epsilon}$，时间复杂度 $O(b^{\frac{C^*}{\epsilon}})$
- 空间复杂度 $O(b^{\frac{C^*}{\epsilon}})$
- 边权为正时完备
- 能找到最优解

贪心搜索 Greedy Search

- 总是按照启发函数（到达最近目标状态距离的预估），DFS展开启发函数最低的节点
- 最坏情况是退化到DFS

A*搜索 A* Search

- 总是按照启发函数和从根节点到节点距离之和展开最低的节点

A*树搜索

- 将全图展开为搜索树，不进行去重，允许单一节点重复展开

- 需要一个可接受的(Admissible)启发函数
- 对于有限分支（任意节点都只有有限个后继节点）的搜索树，是完备且最优的

A* 图搜索

- 维护一个已搜索集(close set)，将展开过的节点放入已搜索集内，不再重复展开
- 需要一个一致的(Consistent)启发函数
- 完备且最优

启发函数 Heurisics

- 可接受的 Admissible：对于任何节点，启发函数总是不高估实际到最近目标状态的距离，即 $0 \leq h(n) \leq h^*(n)$，其中 $h^*(n)$ 表示到达最近目标状态的实际距离
- 一致的 Consistent：对于任何节点，两预估距离之差小于实际距离的第三边，即 $h(A) - h(B) \leq h^*(A \to B)$，其中 $h^*(A \to B)$ 表示节点A到节点B的实际距离

搜索问题总结

算法	时间复杂度	空间复杂度	完备性 (Completeness)	最优性 (Optimality)
DFS	$O(b^m)$	$O(bm)$	是（原图无环时）	否
BFS	$O(b^s)$	$O(b^s)$	是	是（原图边权全部相等时）
UCS	$O(b^{C^*/\epsilon})$	$O(b^{C^*/\epsilon})$	是（边权为正）	是
Greedy	$O(b^m)$	$O(b^m)$	是（原图无环时）	否
A* Tree	指数级	指数级	是	是（需启发函数 **Admissible**）
A* Graph	指数级	指数级	是	是（需启发函数 **Consistent**）

Lec3 约束满足问题 Constraint Satisfaction Problems (CSP)

问题定义

- 一种比起给出找到结果的路径，找到正确结果更为重要的问题
- 在有约束的前提下，构造一组变量的值的问题
- 描述为：变量Variables X_i，值域Domain D 或 D_i，约束Constraints
- 例如：
  - 离散有限值域：地图着色问题，N-皇后问题，选课排课问题，竖式填数问题等
  - 离散无限值域：作业调度问题
  - 连续值域：线性约束问题

- 状态 State 由目前已赋值的变量定义，后继函数 Successor function 由“给未赋值的变量赋值”定义，起始状态 Initial state 是空赋值状态，目标测试 Goal test 是满足所有约束的全赋值变量组

回溯搜索 Backtracking Search

- 每一步都检查当前已经赋值的变量是否符合约束，若不符合，则提前剪枝

滤波：正向检查 Filtering: Forward Checking

- 在遍历赋值的过程中，每一步尝试检查所有未赋值的变量，排除一些不符合约束的情况
- 只要存在某个变量值域为空，则证明当前无解，提前剪枝

滤波：约束传播 Filtering: Constraint Propagation

约束边（弧）一致性 Consistency of an arc

- 弧的尾部节点对应变量的每个取值，对应边的头部节点变量都存在合法的取值，则称这条约束边是一致(Consistent)的

约束传播算法

- 对于每一条约束边，都进行一致性检验；如果发现（弧尾部节点上的）非一致的取值，则删除非一致的取值，并重新检查与该变量相关的所有其它边
- 对于n个变量、每个变量有d个取值的CSP问题，时间复杂度：$O(n^2d^3)$，可优化为 $O(n^2d^2)$

排序：最小剩余值 Ordering: Minimum Remaining Values (MRV)

- 总是先选择检查剩余可能值最少的变量，以更容易排除错解，加速回溯

排序：最小约束值 Ordering: Least Constraining Value (LCV)

- 总是先选择尝试**能够给其它未赋值变量留下最多可选择空间**的值

结构：树状CSP问题

- 对于无环的CSP问题，时间复杂度为 $O(nd^2)$
- 算法
  - 先任意确定一个根，然后把图建立成一棵树 $O(n)$
  - 从最底层叶子结点开始，对于每个非根节点，检查其父亲节点与该节点之间弧的一致性，并删除不可能情况 $O(nd^2)$
  - 从根节点开始，任取合法取值，从上往下赋值 $O(n)$

迭代算法

- 先随便赋一组值给所有变量，再随机调整冲突边涉及的变量取值，直到满足所有约束
- 绝大多数情况下很快，但约束数量/变量数量为特定值附近时非常慢

局部搜索 Local Search

- 状态定义为一组完整的赋值
- 每次在局部（局部有多种定义）做一次优化

模拟退火 Simulated Annealing

- 每次尝试随机移动
- 对于向更好的方向的移动，总是接受 Accept
- 对于向更差的方向的移动（移动大小为T），有 $e^{-\frac{\Delta E}{T}}$ 概率接受，其中 ΔE 为 worse degree

遗传算法 Genetic Algorithms

- 保留暂时最好的若干个状态
- 交叉“杂交”各个状态，并发生突变得到新状态，然后重新检验

Lec4 对抗搜索 Adversarial Search

分类

- 确定性/随机性
- 玩家数量
- 是否零和博弈
- 信息是否完备

确定性游戏的定义

- 状态 State `S`
- 玩家 Player `P = {1, ..., N}`
- 动作 Action `A`
- 转移函数 Transition Function `SxA -> S`
- 终端测试 Terminal Test `S -> {true, false}` （判断是否结束）
- 终端效用 Terminal Utilities `SxP -> R` （最终根据结果分配利益）

状态树

- 由状态作为节点、状态直接转移作为边、初始状态为根的一棵树
- 每个非叶子节点的值定义为当前子树内叶子节点的最大值
- 每个叶子节点的值定义为当前状态的终端效用

Minimax

- 对于双人游戏，一人会最大化总体效用，一人会最小化总体效用
- 因此游戏轮流进行时，状态树为一层max（上三角）一层min（下三角）：每一层取其子节点的max/min，交替进行
- 复杂度类似于DFS：时间复杂度 $O(b^m)$，空间复杂度 $O(bm)$，其中树深度（最大操作次数）为 m，每个节点的子节点数量为 b
- 正确性可以保证，但是复杂度过高
- 代替：限制搜索深度，但是估计值有可能不准
- 代替：设计根据状态的估计函数 Evaluation Functions
- 代替：蒙特卡洛树搜索，全随机移动，限制采样数

- 代替：卷积神经网络 Convolutional Neural Network

优化：剪枝 Game Tree Pruning

- 省去不必要的继续搜索

Alpha-Beta 剪枝

- 对于Minimax树的剪枝
- α 表示 `Max` 玩家从当前节点到根节点的路径上的最佳选择
- β 表示 `Min` 玩家从当前节点到根节点的路径上的最佳选择

```
def max_value(state, alpha, beta):
    initialize v = -inf
    for each successor of state:
        v = max(v, value(successor, alpha, beta))
        if v >= beta:
            return v
        alpha = max(alpha, v)
    return v

def min_value(state, alpha, beta):
    initialize v = inf
    for each successor of state:
        v = min(v, value(successor, alpha, beta))
        if v <= alpha:
            return v
        beta = min(beta, v)
    return v
```

- 本质上是进行一个前序DFS遍历，删去无意义的讨论
- 对于任意构造的搜索树，时间复杂度 $O(b^m)$
- 对于完美排序的搜索树，时间复杂度 $O(b^{m/2})$
- 可能发生 $\alpha-\beta$ 剪枝的位置（从左往右DFS）：自身不是左子节点，且存在某个祖先节点（不含自己和根节点）也不是左子节点

带随机：按期望搜索 Expectimax Search

状态树

- 一层max（上三角），一层chance（圆形）

Expectimax Search

- 对于对手的动作，返回所有子节点的期望
- 对于自己的动作，返回所有子节点的最大值

Lec5 命题逻辑 Propositional Logic

基于知识库的AI

- 知识库 Knowledge base （特有的）

- 推理引擎 Inference engine （共用的）

形式语言 Formal Language

- 语法：什么句子是合法的?
- 语义：在每个模型（情况）下，什么句子是 `True` 或 `False` 的?

命题逻辑 Propositional Logic

逻辑连词/逻辑操作

- $\neg$ 非 negation
- $\wedge$ 合取，与 conjunction
- $\vee$ 析取，或 disjunction
- $\Longrightarrow$ 蕴含 implication （单双箭头均可）
- $\Longleftrightarrow$ 双向蕴含 biconditional （单双箭头均可）

逻辑恒等式 Logical equivalence

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

合法与可满足 Validity and Satisfiability

- 合法 valid：在所有模型下都true（重言式）
- 可满足 satisfiability：在某些模型下true
- 不可满足 unsatisfiable：不在任何模型下true

推理

- $\models$ 语义蕴含，推出 Semantic Entailment：一种在所有可能模型下的推理
  - `Today is Sunday. |= Today is weekend.`
- $\vdash$ 证明 Proof（句法推导 Syntactic Derivation）：一种对推理的揭示
  - model checking 真值表检验
  - inference rules 推理规则证明

归结推理规则 Resolution inference rule for CNF

- 合取范式 Conjunctive Normal Form, CNF：（若干个命题的或）与起来
  - $(A \vee B) \wedge (\neg B \vee C \vee D)$
- 对于合取范式，如果两个括号内的某对命题是相反的，则合取范式的这两个括号可以改为两个括号内的其它命题或起来的结果
  - $(A \vee B) \wedge (\neg B \vee C) \models (A \vee C)$
  - $(B) \wedge (\neg B) \models \{\}$，空命题为 False

Horn 逻辑 Horn logic

- n个前提假设premises 同时成立，则能推出结论 conclusion：
  - $\alpha_1, \alpha_2, \cdots, \alpha_n,\ \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n \implies \beta \models \beta$
- n 可以为 0，但是 α, β 均为非逆的命题符号（必须是单独的命题，不能是命题的非或者命题的组合）

前向链接 Forward Chaining

- 条件驱动的推理
- 对于 Horn 逻辑，前向链接是可靠且完备的推理
- 从已知条件开始，每当一个 Horn 子句 (Horn clause) 的前提假设全部成立，则认为命题成立

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

反向链接 Backward chaining

- 目标驱动的推理，从结论开始反向推导：要证明结论，需要先证明什么
- 先从目标出发，自顶向下；再从已经满足的条件开始，沿着反向链接自下而上推理

逻辑编程 Logic programming

- 定义问题 Identify problem
- 收集信息 Assemble information
- 编码信息 Encode info in KB (Knowledge base)

- 编码事实 Encode problem instance as facts
- 提出查询，运行求解器 Ask queries

Lec6 一阶逻辑 First-Order Logic, FOL

一阶（谓词）逻辑 First-order (predicate) logic

- 组成成分
  - 对象 Objects
  - 关系 Relations
  - 函数 Functions
- 逻辑符号
  - 连接词 Connectives $\wedge, \vee, \neg, \implies, \iff$
  - 变量 Variables x, y, ...
  - 量词 Quantifiers $\forall, \exists$
  - 等号 Equality =
- 非逻辑符号
  - 常量 Constants `Shanghaitech, 2, ...`
  - 谓词（判断句） Predicates `>, isBrother(), ...`
  - 函数 Functions `Sqrt(), BrotherOf(), ...`

句子

- 原子句 Atomic sentences：`predicate(term1, ...)` 或 `term1 = term2`
- 项 Term：`constant` 或 `variable` 或 `function(term1, ...)`
- 复杂句 Complex sentences：用连接词 Connectives 连接原子句

量词

全称量词 $\forall$

- 语法：$\forall$ `<variables> <sentence>`
  - $\forall$x At(x, STU) $\implies$ isSmart(x)
- 一般来说，$\implies$ 是全称量词后面主要的连接词（不要使用 $\wedge$ 来代替 $\implies$）

存在量词 $\exists$

- 语法：$\exists$ `<variables> <sentence>`
  - $\exists$x At(x, STU) $\wedge$ isSmart(x)
- 一般来说，$\wedge$ 是存在量词后面主要的连接词（不要使用 $\implies$ 来代替 $\wedge$）

量词的属性

- 交换律：$\forall x \forall y = \forall y \forall x, \exists x \exists y = \exists y \exists x$，但是 $\forall x \exists y \neq \exists y \forall x$
- 任意句中的变量都应该被量词约束

推理

全称实例化 Universal Instantiation (UI)

- 对于语句 $\forall v, \alpha$，可以把 $\forall v$ 改为任意 v 的不含变量的子集 g（ground term基项），有 v, α
- 可以无限实例化替换

存在实例化 Existential Instantiation (EI)

- 对于语句 $\exists v, \alpha$，可以把 $\exists v$ 改为一个唯一的符号 C_1，有 C_1, α，其中 C_1 被称为 Skolem constant
- 这个过程只能进行一次，被称为Skolemization

一阶谓词逻辑的推理

- 穷举所有的量词实例化，将一阶谓词逻辑转化为命题逻辑（命题化 propositionalized），然后用命题逻辑来计算
- 定理：如果一个句子 α 能够被一个一阶谓词逻辑知识库 FOL KB 蕴含（推出），那句子 α 也能被这个知识库的某个有限子集蕴含（推出）
- 定理：一阶谓词逻辑的推理是半可确定的 semi-decidable：
  - 对于可以推出的结论，存在算法可以推出
  - 对于不可推出的结论，不存在算法可以确定无法推出
- 合一化 Unification：找到一种合适的替代（实例化），使得句中的两个子表达式变得完全一样
  - 标准化分离 standardized apart：避免变量重名导致合一化困难
  - 最一般合一 most general unifier(MGU)：存在最一般的合一，在变量重命名的意义下是唯一的

一阶谓词逻辑的Horn子句 Horn clauses

- $p_1 \wedge \cdots \wedge p_n \implies q$，其中 p_i 均为原子句，而所有变量都是全称量化的
- 广义肯定前件 Generalized Modus Ponens(GMP)：对于$p_1', \cdots, p_n'$，都存在$p_i'\theta = p_i\theta$，且有 $p_1 \wedge \cdots \wedge p_n \implies q$，那么我们可以推理出 $q\theta$

前向链接和反向链接 Forward and Backward Chaining

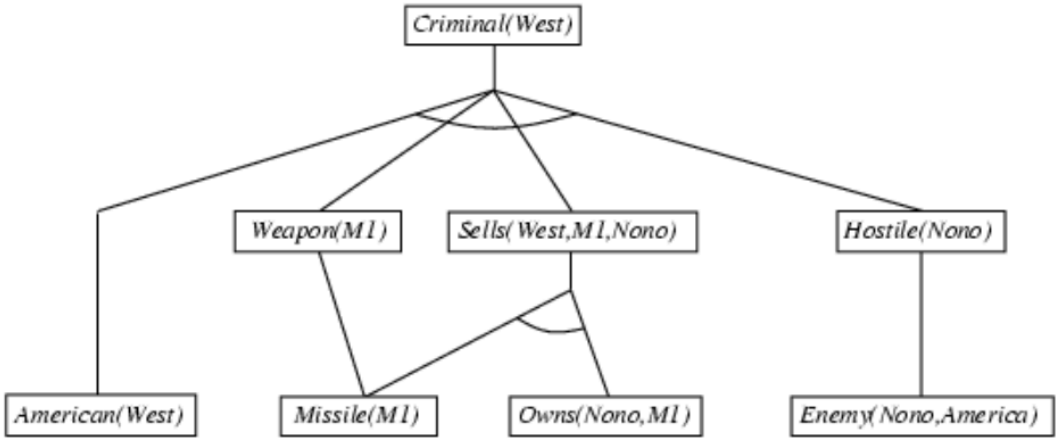
Example knowledge base

- The US law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

- Prove that Col. West is a criminal

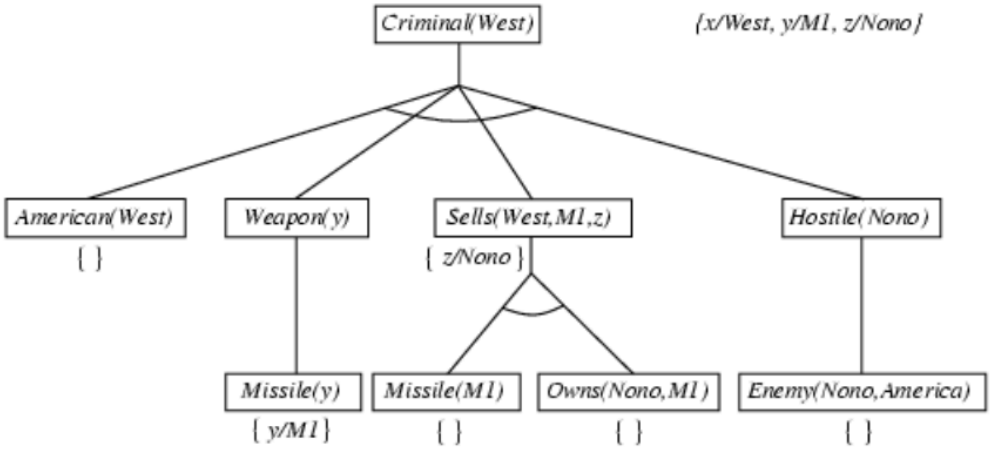
Forward chaining proof

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



Backward chaining example

- *American(x) ∧ Weapon(y) ∧ Sells(x,y,z) ∧ Hostile(z) ⇒ Criminal(x)*
- *Missile(x) ∧ Owns(Nono,x) ⇒ Sells(West,x,Nono)*
- *Missile(x) ⇒ Weapon(x)*
- *Enemy(x,America) ⇒ Hostile(x)*
- *Owns(Nono,M_1), Missile(M_1), American(West), Enemy(Nono,America)*



- 现在编程使用更多的是反向链接 Backward Chaining

一阶谓词逻辑的归结 Resolution

- 对于两个子句（文字/文字的否定的析取）$l_1 \vee \cdots \vee l_k$ 和 $m_1 \vee \cdots \vee m_n$，存在 i 使得 $Unify(l_i, \neg m_j) = \theta$（即$l_i\theta = (\neg m_j\theta)$），则可以推出 $l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n$

Lec7 贝叶斯网络 Bayesian network

概率 Probability

不确定性 Uncertainty

来源：

- 随机噪声 noisy sensors
- 不确定性结果 uncertain action outcome
- 部分可观测 partial observability
- 模型复杂性简化 immense complexity of modelling and predicting

决策 Decisions

- 效用理论 Utility theory 用于表示偏好
- 决策理论 Decision theory = 效用理论 + 概率论
- 决策为最大化期望效用 $a^* = argmax_a \sum_s P(s|a)U(s)$

概率分布

- 非负、和为1
- 对于n个事件，每个事件有d种情况，其联合概率的分布列有 d^n 行，这是不可接受的，因此需要优化
- 事件是一系列结果的集合
- 边际分布是联合概率表通过消除某些变量得到的子表
- 条件概率

概率推断 Probabilistic Inference

通过枚举推断 Inference by Enumeration

- 定义：从已知的概率推导出未知的概率
  - 证据变量 Evidence variables $E_1, \cdots, E_k = e_1, \cdots, e_k$
  - 询问变量 Query variable Q
  - 隐变量 Hidden variables $H_1, \cdots, H_r$
  - （其中大写是随机变量 RV，小写是结果 outcome）
- 我们需要的是 $P(Q|e_1, \cdots, e_k)$
- 步骤
  - 选择有关的条目
  - 对隐变量求和 $P(Q, e_1, \cdots, e_k) = \sum_{h1,\cdots,h_r} P(Q, h_1, \cdots, h_r, e_1, \cdots, e_k)$
  - 归一化得到后验概率
- 问题：时间和空间复杂度到达 $O(d^n)$，不可接受

贝叶斯推断 Inference with Bayes' Rule

- 链式法则 The Chain Rule $P(x_1, x_2, \cdots, x_n) = \Pi_i P(x_i|x_1, \cdots, x_{i-1})$

- 贝叶斯公式 Bayes' Rule $P(x|y) = \frac{P(y|x)}{P(y)}P(x)$
- $P(\text{因}|\text{果}) = \frac{P(\text{果}|\text{因})}{P(\text{果})}P(\text{因})$

贝叶斯网络 Bayesian Networks

独立性 Independence

- 两个变量是独立的，当且仅当 $\forall x, y$, 有 $P(x, y) = P(x)P(y)$
- 两个变量x, y是条件z下独立的，当且仅当 $\forall x, y, z$, 有 $P(x, y|z) = P(x|z)P(y|z)$

贝叶斯网络定义

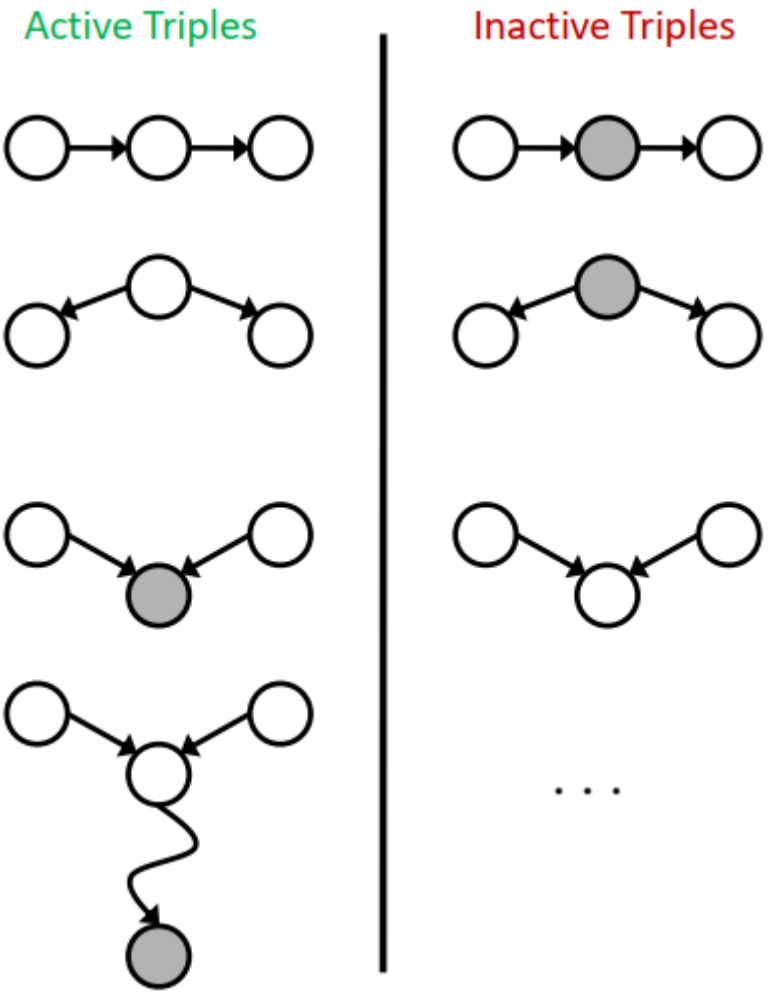
- 节点：随机变量（含有定义域）
- 边：直接关系，编码条件相关（每条边附带一张条件概率表）
- 有向无环图
- 条件概率表 Conditional probability table, CPT
  - 每一行表示一个子节点在给定所有父节点取值的情况下的分布
- 对于有n个变量、最大值域范围为d、最大父节点数量为k的贝叶斯网络，其大小为 $O(nd^{k+1})$
- 贝叶斯网络只反映相关性，不一定反映正确的因果性

独立/条件独立性

- 对于随机变量x，给定父节点的情况下，x与所有x的非后代节点（直系下方节点）条件独立
- **马尔可夫毯** Markov blanket：对于中心变量x，包括其父节点、子节点和子节点的其它父节点
  - 每个变量在给定它的马尔可夫毯的情况下，与所有其它变量条件独立
- 例（因果链）：有贝叶斯网络 `x` -> `y` -> `z`，则 `x` 与 `z` 不独立；但是给定 `y` 的前提下，有 $P(z|x, y) = \frac{P(x,y,z)}{P(x,y)} = \frac{P(x)P(y|x)P(z|y)}{P(x)P(y|x)} = P(z|y)$，故给定 `y` 时，`x` 与 `z` 条件独立
- 例（共因）：有贝叶斯网络 `x` <- `y` -> `z`，则 `x` 与 `z` 不独立；但是给定 `y` 的前提下，有 $P(z|x, y) = \frac{P(x,y,z)}{P(x,y)} = \frac{P(y)P(x|y)P(z|y)}{P(y)P(x|y)} = P(z|y)$，故给定 `y` 时，`x` 与 `z` 条件独立
- 例（共果）：有贝叶斯网络 `x` -> `z` <- `y`，则 `x` 与 `y` 独立；但是给定 `z` 的前提下，`x` 与 `y` 不条件独立

有向图中的分离 D-separation

- 问题：对于X,Y,Z三个没有交集的节点，在给定Z的前提下，X和Y是否条件独立?
- 一个三元组 triple 是激活的 active，当且仅当是以下三种情况之一：
  - 因果链 A->B->C ，且B是未被观察的
  - 共因 A<-B->C ，且B是未被观察的
  - 共果 A->B<-C ，且B或B的某个后代节点被观察了

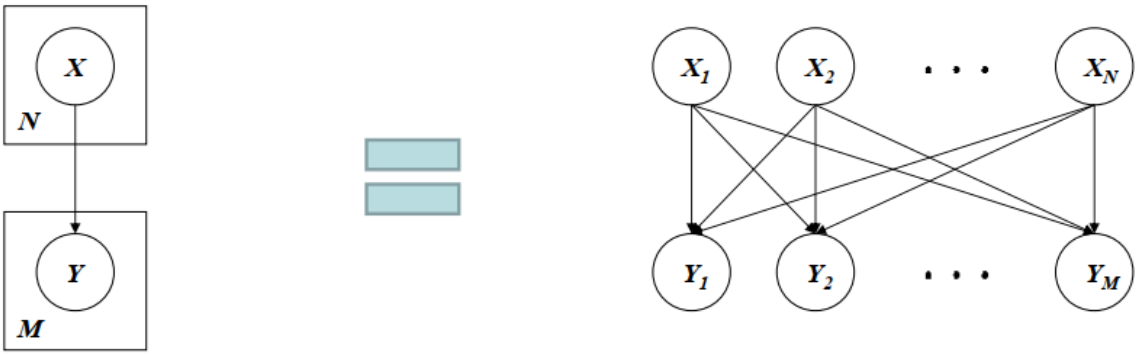


- 一条路径是激活的 active，当且仅当路径上的所有三元组都是激活的
- 一条路径是阻塞的 blocked，当且仅当路径上包含了至少一个非激活的 inactive 三元组
- 如果所有从 X 到 Y 的路径都是阻塞的，则称 X 与 Y 在 Z 条件下被有向分离 d-separated 了，此时 X 与 Y 在给定 Z 的条件下条件独立

节点顺序

- 不按照逻辑顺序建立节点关系也是可行的，但是有可能由于条件相关性，可能会需要更多的参数
- 找到最优逻辑顺序的算法是 NP-hard 的

盘状记号 Plate Notation



马尔可夫网络 Markov Networks

- 无向图（可能有环）
- 极大团 Maximal Clique：完全子图，其加入任何一个其它节点都不再是完全子图

- 归一化参数 $Z = \sum_x \Pi_C \psi_C(x_C)$ 其中C为马尔可夫网络的团，$\psi_C(x_C)$ 为团C的势 potential，x_C 是团C的所有点
- 联合概率正比于势之积 $p(x) = \frac{\Pi_C \psi_C(x_C)}{Z}$
- 节点x的马尔可夫毯 Markov Blanket 是与x有直接连边的所有节点的集合

贝叶斯网络 -> 马尔可夫网络

- 图的有向边转换为无向边；道德化 moralization（共有子节点的父节点之间连边）

贝叶斯网络，马尔可夫网络，逻辑

- 二者都是图模型 Graphical Models；建模的结果都是联合分布，但是计算过程不同；二者表示的条件独立性集合不同
- 贝叶斯网络/马尔可夫网络 可以看作命题逻辑的概率拓展
- 贝叶斯网络/马尔可夫网络 都是生成式模型 Generative Models（表示联合分布的模型）
- 判别模型 Discriminative Models 只关心用证据预测查询，建模结果是给定证据（条件）下的条件分布

Lec8 贝叶斯网络：精确推理 Exact Inference

变量消除 Variable Elimination，VE

- 在贝叶斯网络中，对于隐藏变量，我们需要通过求和式将其边缘化消除，但这个过程涉及到大量的含重复项的乘法、加法计算
- 于是可以考虑将求和过程尽可能放在乘法内部，先求和后相乘，可以减少计算量

因子 Factor

- 因子 Factor 是一个多维向量，代表条件概率表；当变量被赋值时，因子的维度会减小（`P(L|A, B)` 维度大于 `P(L|True, B)`）
- 因子的大小是对应条件概率表的行数，即因子的维数

枚举计算条件概率

- 给定多个因子，把因子中相关的变量提取出来，形成新的变量
- 把因子进行变量内积，通过贝叶斯公式计算结果因子

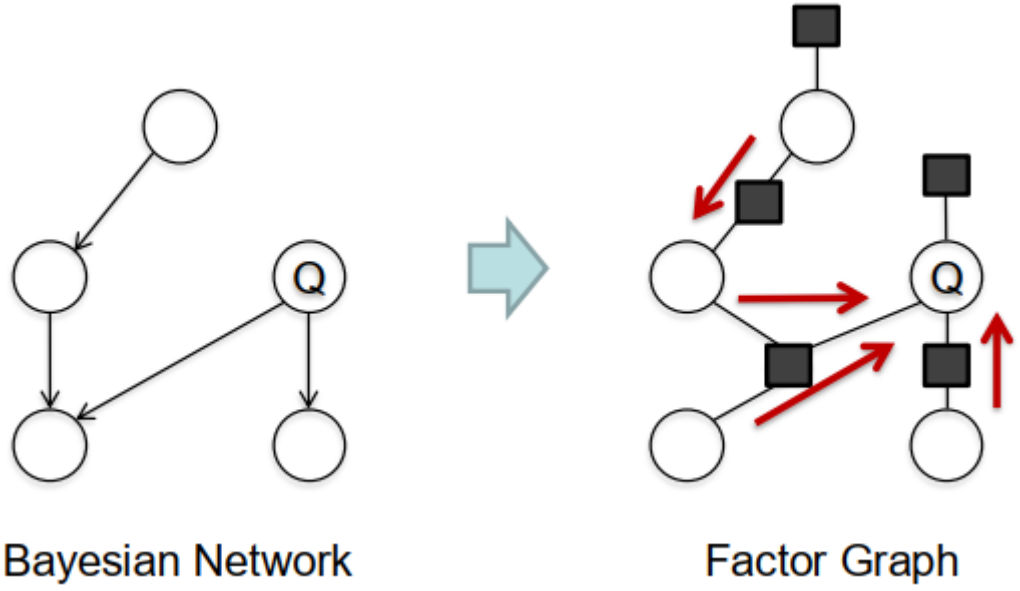
变量消除 Variable Elimination，VE

- 每次计算含隐藏变量的条件概率时，可以先把条件变量求和消除，以简化后续计算过程
- 本质上是一种 Σ 向内交换
- 然而，消除顺序不同可能导致计算复杂度不同
- 变量消除的时间/空间复杂度由最大因子规模决定
- 在一般图中，寻找最优消去顺序是NP-hard问题（可以由3-SAT归约而来）

多树结构 Polytrees

- 多树结构：有向图，在不考虑方向时也没有环
- 在多树上，VE的复杂度与节点数呈线性关系

- 因子图 Factor Graph：包含变量和因子两类节点的一个无向二分图，只存在变量和因子之间的连边；变量节点表示一个随机变量，因子节点表示一个概率（端末的方形节点）或条件概率（边上的方形节点）



- 消去顺序：（从外围向中心逐层展开）
  - 将贝叶斯网络转换为因子图
  - 将查询变量作为根节点
  - 从叶子结点向根节点，依次消去变量
- 对于带环的图，也可以按照不带环的图来计算，得到近似结果

Lec9 贝叶斯网络：估计推断

采样 Sampling

- 目标：获得概率 `P`
- 操作：从采样分布 `S` 中采样 `N` 个样本，计数并证明概率趋于 `P`
- 时间复杂度：$O(N)$

贝叶斯网络中的采样

- 直接推导贝叶斯网络的精确结果复杂度过高，故考虑用更快的近似方法
- 已知整个贝叶斯网络，以及所有单步条件概率，求某个条件概率

先验采样 Prior Sampling

- 不管条件如何，先按照贝叶斯网络的拓扑顺序，随机生成整个贝叶斯网络的值并采样，重复 N 次
- 对于 $P(Q)$，求和统计出 $N(Q)$，有估计 $\hat{P}(Q) = \frac{N(Q)}{N}$
- 对于 $P(Q|e)$，求和统计出 $N(e)$ 和 $N(Qe)$，有估计 $\hat{P}(Q|e) = \frac{N(Qe)}{N(Q)}$
- 采样过程是无偏的 consistent，即当样本数量足够大时，结果是精确的
- 但是会有很多无效的采样

拒绝采样 Rejection Sampling

- 对于目标为条件概率 $P(Q|e)$ 的问题，可以在生成整个贝叶斯网络的过程中，对于不满足条件 e 的情况就提前剪枝 Reject
- 但是对于稀少的样本，有可能会较难采到样

似然加权 Likelihood Weighting

计算过程

- 给定证据（条件变量） $e_1.\cdots,e_k$
- 对于每一次采样，对于每一个变量 x_i：
  - 设定初始权重 $w=1.0$
  - 如果变量 x_i 是证据变量，则令 $x_i=e_i$，并设定权重为 $w:=w\times P(x_i|Parents(x_i))$
  - 如果变量 x_i 不是证据变量，则正常采样 $P(X_i|Parents(x_i))$
- 对于每一次采样，返回 $(x_1,\cdots,x_n),w$
- 最后对于每个采样的结果，以权重 w 加权求和，占比即为条件概率

理解与优缺点分析

- 可以理解为一种对非条件变量采样、对条件变量计算的方法
- 是无偏的 consistent
- 优化了拒绝采样方法里，对于稀少样本很难采样的问题
- 问题：上游变量看不见下游的证据，导致生成的样本权重都不高：

  即对于极度稀少的样本（抗性略好于拒绝采样，但是好不到哪去），每个样本提供的权重也会很小，导致仍然存在较多低效计算

吉布斯采样 Gibbs Sampling

计算过程

- 固定所有证据变量为观测值
- 以任意值随机初始化每个非证据变量
- 循环更新：随机取一个非证据变量，固定其它变量不变，以它的马尔可夫毯内变量的值为条件，重新对它进行一次采样，用采样结果代替原来的值，此时所有变量的取值作为一组样本
- 最后对于所有样本直接计数，占比即为条件概率

优缺点分析

- 是无偏的 consistent
- 优化了似然加权方法里权重低的问题，这里的每一个样本都是权重为1的有效样本
- 初始值过于随机，可能产生偏差（可以丢弃前面一部分样本，减少这个偏差）
- 由于是演化产生的样本，样本之间是相关的，所以样本方差偏小
- 对于较为独立的几个高概率区，吉布斯采样可能被困在某个高概率区而难以到达另一个，导致偏差

马尔可夫链蒙特卡洛 Markov Chain Monte Carlo, MCMC

- 一类估计一系列值的随机化算法
- 每种状态仅依赖于前一种状态

- Metropolis-Hastings 算法是马尔可夫链蒙特卡洛（MCMC）方法的核心算法之一，用于从复杂的目标分布 P(x) 中采样
- Metropolis-Hastings 算法：
  - 从建议分布 $g(x'|x)$ 中采样一次
  - 以概率 $min(1, \frac{P(x')g(x|x')}{P(x)g(x'|x)})$ 接受这个采样
  - 循环这个过程
- 吉布斯采样是 Metropolis-Hastings 算法的一个特例

Lec10 时序概率推理 Probabilistic Reasoning over Time

马尔可夫模型 Markov Models

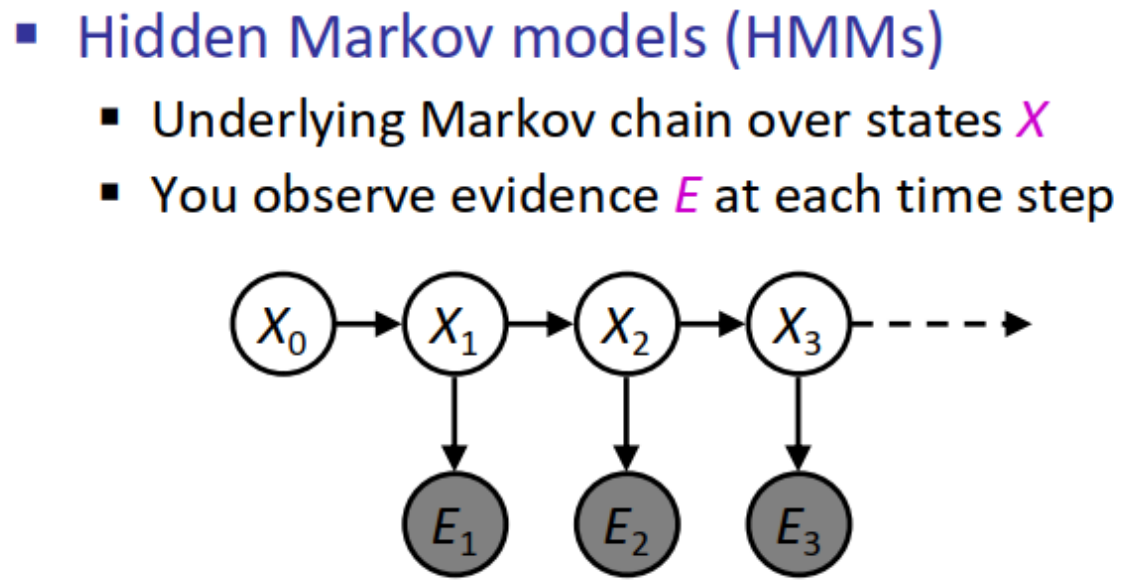
- 对于一列离散的变量 $X_0, X_1, \cdots$ 后面的变量取决于前面的变量，且所有变量共享无穷大的值域
- 离散变量的数量一般很大，甚至可以不收敛
- 由 $P(X_0), P(X_t|X_{t-1})$ 建模
- 联合分布：$P(X_0, \cdots, X_T) = P(X_0)\Pi_t P(X_t|X_{t-1})$
- 平稳性假设 Stationarity Assumption：$P(X_t|X_{t-1}) = Const$，即状态随时间演变的规则保持不变
- （一阶）马尔可夫假设 Markov Assumption：X_{t+1} 在给定 X_t 的前提下，与 $X_0, \cdots, X_{t-1}$ 条件独立
- n阶马尔可夫假设为 X_t 仅依赖于前n个变量的取值（语言模型中，$n \to \infty$）

稳态分布 Stationary Distribution

- 马尔可夫模型在 $t \to \infty$ 时概率 $P(X_\infty)$ 会趋于定值，且与初始状态 $P(X_0)$ 无关
- 可以用 $P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$ 求解

隐马尔可夫模型 Hidden Markov Models, HMMs

- 有隐含的马尔可夫模型，但是不能直接观测；只能观测到马尔可夫模型每一步的证据 evidence E_i



- 隐马尔可夫模型 HMM：

    - 初始分布 Initial distribution $P(X_0)$
    - 转移模型 Transition model $P(X_t|X_{t-1})$
    - 发射模型 Emission model $P(E_t|X_t)$
- 联合分布：$P(X_0, X_1, E_1, \cdots, X_T, E_T) = P(X_0)\Pi_{t=1:T}P(X_t|X_{t-1})P(E_t|X_t)$
- 独立性：当前状态取值仅与前一状态有关；证据变量仅与当时状态有关
- 简写 $X_a, \cdots, X_b$ 为 $X_{a:b}$

推断任务 Inference tasks

滤波 Filtering

- 已知 $e_{1:t}$, $P(X_{t+1}|X_t)$, $P(X_0)$, $P(E_t|X_t)$，求 $P(X_t|e_{1:t})$
- 给定至今所有证据 evidence，推断当前状态
- 前向算法 Forward algorithm：

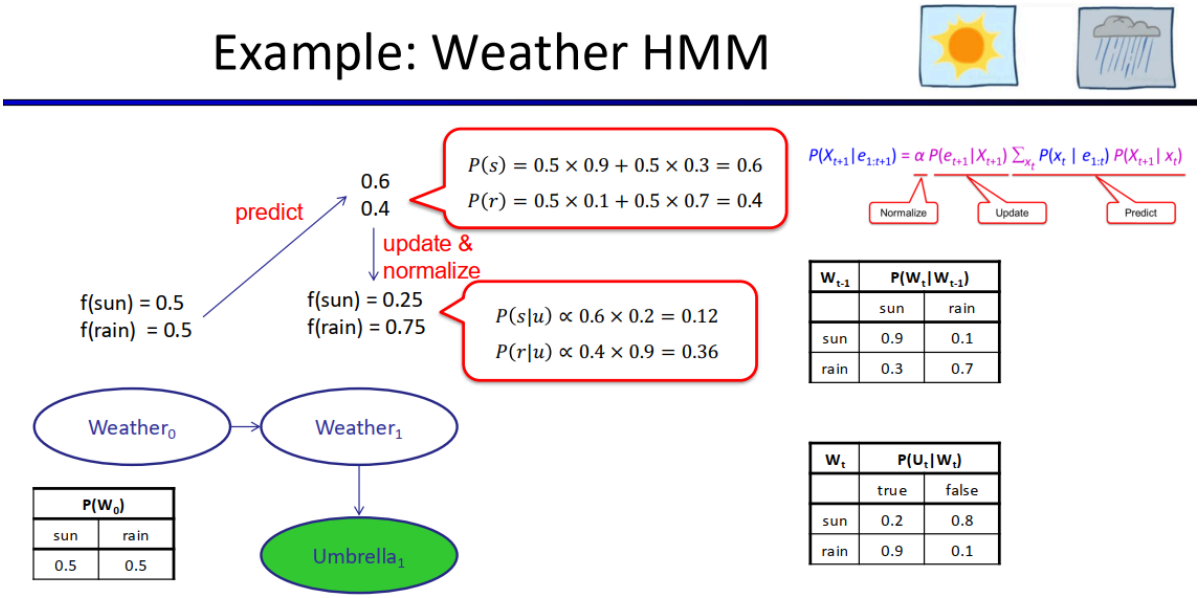
  $P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{1:t}, e_{t+1})$

  $= \alpha P(e_{t+1}|X_{t+1}, e_{1:t})P(X_{t+1}|e_{1:t})$

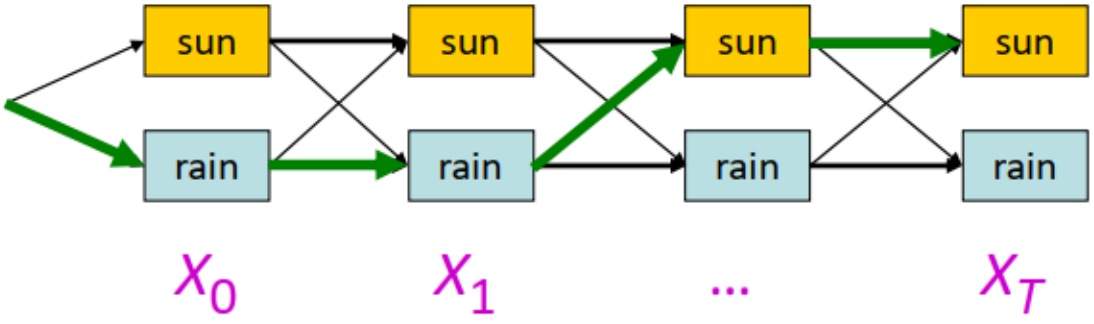
  $= \alpha P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})$

  $= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(x_t|e_{1:t})P(X_{t+1}|x_t, e_{1:t})$

  $= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(x_t|e_{1:t})P(X_{t+1}|x_t)$
    - 其中，α 为正则项 Normalize，$P(e_{t+1}|X_{t+1})$ 为更新 Update，$\sum_{x_t} P(x_t|e_{1:t})P(X_{t+1}|x_t)$ 为预测 Predict
- 其中 $\alpha = \frac{1}{P(e_{t+1}|e_{1:t})}$，一般不计算，而是直接求和归一化
- 故我们可以从 $f_{1:0} = P(X_0)$ 开始，使用 $f_{1:t+1} = FORWARD(f_{1:t}, e_{t+1})$ 迭代转移
- 时间复杂度为 $O(t|X|^2)$，其中 $|X|$ 为状态数量



- 状态转移图 State trellis：每条边表示转移，边权表示 $P(x_t|x_{t-1})P(e_t|x_t)$，路径上边权的连乘积表示沿这条路径到达指定节点的概率



预测 Prediction

- 已知 $P(X_t|e_{1:t})$（滤波的结果），$P(X_{i+1}|X_i)$，求$P(X_{t+k}|e_{1:t})$ for $k > 0$
- 结果：$P(X_{t+1}|e_{1:t}) = \sum_{x_t} P(x_t|e_{1:t})P(X_{t+1}|x_t)$，递推k次得到$P(X_{t+k}|e_{1:t})$
- 时间复杂度：滤波 $O(t|X|^2)$ + 预测 $O(k|X|^2)$

平滑 Smoothing

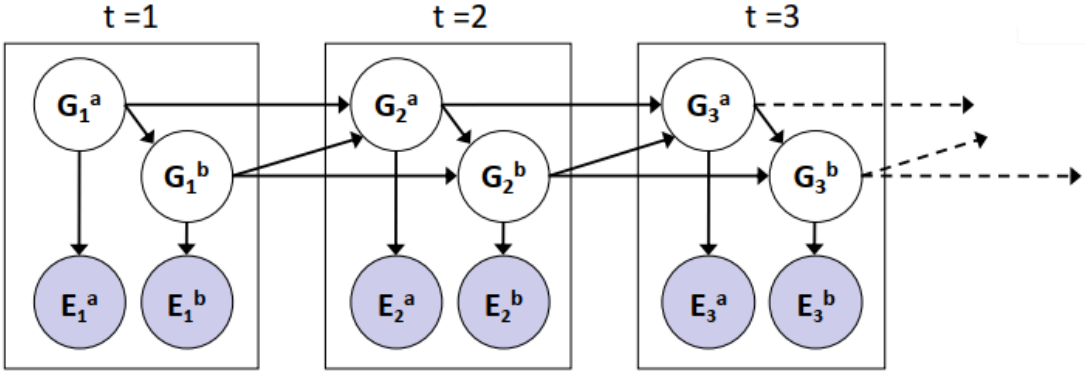
- 已知 $e_{1:t}$, $P(X_0)$, $P(X_{i+1}|X_i)$, $P(E_t|X_t)$，求 $P(X_k|e_{1:t})$ for $0 \le k < t$
- 前向-后向算法 Forward-Backward Algorithm：
  - $P(X_k|e_{1:t}) = \alpha \cdot P(X_k|e_{1:k}) \cdot P(e_{k+1:t}|X_k)$
  - 对于 $P(e_{k+1:t}|X_k)$，有递归计算
    $P(e_{k+1:t}|X_k) = \sum_{x_{k+1}} P(x_{k+1}|X_k) \cdot P(e_{k+1}|x_{k+1}) \cdot P(e_{k+2:t}|x_{k+1})$
- 时间复杂度 $O(t|X|^2)$，空间复杂度 $O(t|X|)$

最可能解释 Most likely explanation

- 已知$P(X_0)$, $P(X_t|X_{t-1})$, $P(E_t|X_t)$, $e_{1:t}$，求$argmax_{x_{0:t}} P(x_{0:t}|e_{1:t})$
- Viterbi algorithm：对于时间t的每个状态 x_t，保存到达该状态的最大可能性路径：
  - $m_{1:t}(x_t) = max_{x_{1:t-1}} P(x_{1:t}|e_{1:t})$
  - 更新转移：$m_{1:t+1} = VITERBI(m_{1:t}, e_{t+1})$
    $= P(e_{t+1}|X_{t+1}) max_{x_t} P(X_{t+1}|x_t) m_{1:t}[x_t]$
  - 时间复杂度 $O(|X|^2 T)$，空间复杂度 $O(|X|T)$

动态贝叶斯网络 Dynamic Bayes Nets, DBNs

- 相比于隐马尔可夫模型HMM，DBN在每一时刻可以有多个变量和多个对应证据



粒子滤波 Particle Filtering

背景引入

- 当状态数 $|X|$ 过大，或状态连续，此时直接精确推理 Exact inference 难以计算
- 考虑一个扫地机器人，在已知地图，但不知道当前位置的前提下想通过探索来推断自己的位置

算法流程

- 已知一个样本空间（对应机器人的“地图”），每个样本被称为一个粒子 particle
- 每次采样基于当前粒子的位置进行移动（根据机器人的移动，添加小幅度噪声进行采样），把粒子移动到目标地点，每一轮总共采样 N 次（由于优化需要，$N << |X|$）
- 每个粒子以概率分布 $x_{t+1} \sim P(X_{t+1}|x_t)$ 发生定向转移
- 观察，并根据观察到的证据 evidence 设定每个样本粒子的权重 $W = P(e_t|x_t)$
- 重采样 resample：从有权重的样本上进行采样，让每个粒子权重回到1，但总体分布模拟带权分布

Lec11 马尔可夫决策过程 Markov Decision Processes

不确定搜索 Non-Deterministic Search

- 在一个迷宫问题中，有墙
- 对于到达某些地点，存在大的激励
- 对于每一步移动，都存在一些小激励 living reward（可正可负）
- 对于每次移动，动作不一定是按照计划进行的：存在一个概率分布来模拟某次移动的可能性

马尔可夫决策过程 Markov Decision Process, MDP

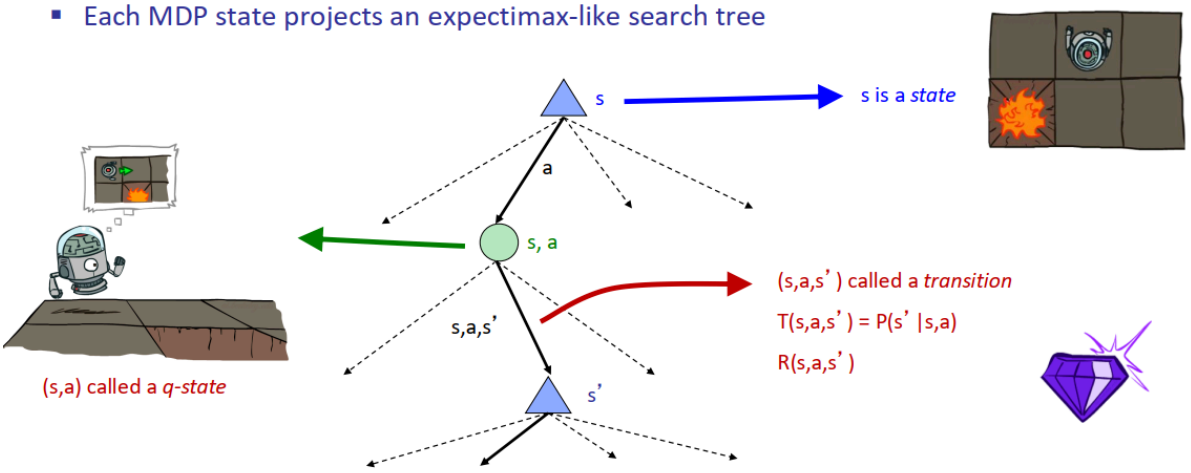
- 一个MDP（的环境）定义如下：
  - 一个状态集合 $s \in S$
  - 一个动作集合 $a \in A$
  - 一个转移方程 $T(s, a, s')$
  - 一个奖励函数 $R(s, a, s')$
  - 一个初始状态 Start state（可能有终末状态 terminal state）
- 马尔可夫性质：在确定当前状态的前提下，未来状态与过去状态独立
- 我们需要找出一个最优计划 optimal plan，即找一个最优策略 optimal policy $\pi^*: S \to A$（看到什么状态，就采取什么动作）

马尔可夫决策树 MDP Search Tree

- 决策过程可以转化为一个类似于 expectimax 的搜索树：AI需要判断未来期望收益的总和最大的计划

MDP Search Trees

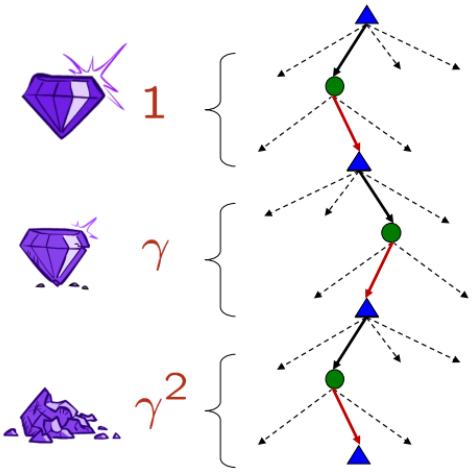
- Each MDP state projects an expectimax-like search tree



- 对于激励，我们倾向于获得即时收益，故设置折扣因子 discounting factor γ，使得第 i 步之后的激励函数 Utility function 乘上系数 γ

Discounting

- How to discount?
  - Each time we descend a level, we multiply in the discount once
- Why discount?
  - Sooner rewards probably do have higher utility than later rewards
  - Also helps our algorithms converge
- Example: discount of 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3=2.75
  - U([3,2,1]) = 1*3 + 0.5*2 + 0.25*1=4.25
  - U([1,2,3]) < U([3,2,1])



- 定义从状态 s 出发，以最优操作得到的期望效用为 $V^*(s)$
- 定义从状态 s 出发，先操作 a，然后最优操作，得到的期望效用为 $Q^*(s,a)$
- 定义从状态 s 出发的最优策略为 $\pi^*(s)$
- 故存在以下转移状态方程：
  - $V^*(s) = max_a Q^*(s,a)$
  - $Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
- 故可以得出 Bellman Equation：$V^*(s) = max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
- $\gamma < 1$ 也可以保证 Bellman Equation 的收敛性

值迭代算法 Value Iteration

- 已知 MDP 的全部环境参数，求最优解/最优解法
- 定义 $V_k(s)$ 为再走 k 步结束游戏、当前状态为 s 的前提下，获得效用的最大值

- 每一步转移时，取当前状态 s 条件下，对于每个下一状态，把 即时收益 $+\gamma\times$ 下一状态最大效用 中的最大值作为当前状态最大效用 $V_{k+1}(s)=max_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V_k(s')]$
- 全部迭代完成（迭代有限步就可以认为收敛）后，对于每个状态，取最优策略 $\pi^*(s)=argmax_aQ^*(s,a)$
- 时间复杂度：每次迭代 $O(S^2A)$
- Q值迭代算法 Q-Value Iteration：使用 $Q_k(s,a)$ 来代替 $V_k(s)$ 迭代，有转移方程 $Q_{k+1}(s,a)=\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma max_{a'}Q_k(s',a')]$

决策迭代 Policy Iteration

- 收敛速度快于值迭代算法
- 定义 $V^\pi(s)$ 为使用决策 π 的期望效用

第一步：决策估计 Policy Evaluation

迭代法

- 大规模MDP适用
- 初始化 $V_0^\pi(s)=0$
- 对于每个 V_k^π，计算 $V_{k+1}^\pi(s)=\sum_{s'}T(s,\pi(s),s')[R(s,\pi(s),s')+\gamma V_k^\pi(s')]$
- 时间复杂度：每次迭代 $O(S^2)$

直接求解法

- 小规模MDP适用
- 使用线性求解器直接求解 $V^\pi(s)=\sum_{s'}T(s,\pi(s),s')[R(s,\pi(s),s')+\gamma V^\pi(s')]$
- 时间复杂度：$O(S^3)$

第二步：决策优化 Policy Improvement

- 迭代以优化决策：$\pi_{i+1}(s)=argmax_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V^{\pi_i}(s')]$

Lec12 强化学习 Reinforcement Learning

引入：老虎机 Bandits

- 对于多臂老虎机，可以建模老虎机是 MDP，通过计算解决问题
- 但是对于全新的环境，我们不知道 MDP 的具体参数，所以无法通过线下规划 Offline Planning 来求解，必须做线上规划 Online Planning，即一边玩老虎机（交互），一边学习 MDP 的参数，然后做出决策
- 强化学习一般认为是在线的 Online

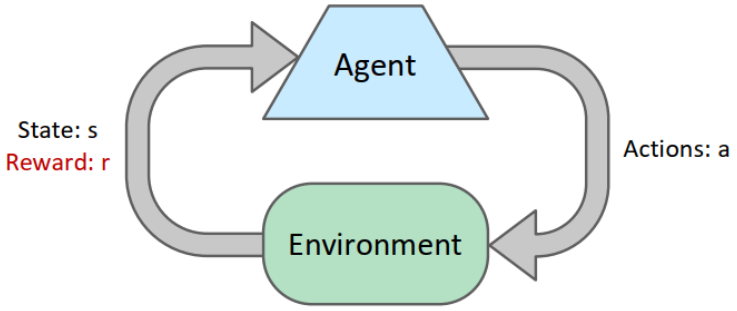
强化学习 Reinforcement Learning

- 建模一个马尔可夫决策过程 MDP：
  - 一个状态集合 $s\in S$
  - 一个动作集合 $a\in A$
  - 一个转移方程 $T(s,a,s')$

- 一个奖励函数 $R(s,a,s')$
- 一个衰减系数 γ (若未提及，默认为1)

- 我们期望找到策略 $\pi(s)$，但是我们不知道转移方程 T 和激励函数 R

Reinforcement Learning



- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards
  - All learning is based on observed samples of outcomes!

Basis of reinforcement learning*

- Markov decision process (MDP) assumption
- At timestep t, agent following a policy $\pi_\omega(s_t)$,
  - Obtains an observation s_t of the surrounding environment,
  - produces action a_t,
  - then, environment will transmit to s_{t+1},
  - and agent will receive reward r_t,

  - The goal of the learning agent is to *maximize* the expected cumulative rewards as

$$R = \mathrm{E}_\pi\left[\sum_{t=0}^{T-1} r_t(s_t, a_t)\right]$$

Interaction data stored in the transition dataset.

$$\{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, \underbrace{s_t, a_t, r_t, s_{t+1}}_{\text{Transition}}, \ldots s_{T-1}, a_{T-1}, r_{T-1},\}$$

* Episodic environment with limited timesteps

Reinforcement Learning

- What if the MDP is initially unknown? Lots of things change!
  - ***Exploration***: you have to ***try unknown actions*** to get information
  - ***Exploitation***: eventually, you have to use what you know
  - ***Regret***: early on, you inevitably "make mistakes" and lose reward
  - ***Sampling***: you may need to repeat many times to get good estimates
  - ***Generalization***: what you learn in one state may apply to others too

基于模型的 Model-based（模型 model 是对于环境的建模，不是策略 policy)

- 先学习一个马尔可夫决策过程模型 MDP model（即学习转移概率 $T(s,a,s)$ 和奖励 $R(s,a,s')$），然后假设这个模型是准确的，进行求解
- 优势：会高效使用所有的经验
- 劣势：会放大模型的误差；状态不完全可知时较为困难；复杂度较高

不基于模型的 Model-free

- 不学习复杂的转移概率分布，而是直接从样本估计分布，并计算期望

Model-Based vs. Model-Free

Goal: Compute expected age of ShanghaiTech students

Known P(A)

$$E[A] = \sum_a P(a) \cdot a \quad = 0.35 \times 20 + \ldots$$

Without P(A), instead collect samples [a_1, a_2, … a_N]

Unknown P(A): "Model Based"

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Why does this work? Because eventually you learn the right model.

Unknown P(A): "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work? Because samples appear with the right frequencies.

被动强化学习 Passive RL

- 智能体遵循一个固定策略 $\pi(s)$ 进行观察，估计给定策略下的状态价值 $V^{\pi}(s)$，即按照固定策略 π 从节点 s 开始走，直到结束得到的奖励 $V^{\pi}(s)$

- 本质是策略评估 Policy Evaluation

直接评估 Direct Evaluation

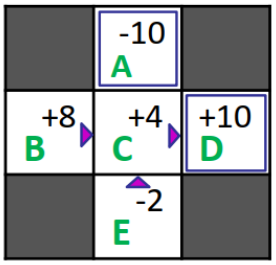
- 记录从某个状态 s 开始到结束的实际总奖励，多次试验结果取平均

Problems with Direct Estimation

- What's good about direct estimation?
  - It's easy to understand
  - It doesn't require any knowledge of *T* and *R*
  - It converges to the right answer in the limit
- What's bad about it?
  - Each state must be learned separately (fixable)
  - It ***ignores information about state connections***
  - So, it takes a long time to learn

E.g., B=at home, study hard
E=at library, study hard
C=know material, go to exam

Output Values



If B and E both go to C under this policy, how can their values be different?

时序差分学习 Temporal difference learning, TD

- 利用 Running Average 思想，即前 n 项均值 $\mu_n = ((n-1)\mu_{n-1} + x_n)/n$
- 智能体每观察到一个转移 (s, a, s', r)，就进行一次更新
  $V^\pi(s) \leftarrow V^\pi(s) + \alpha[R(s, \pi(s), s') + \gamma V^\pi(s') - V^\pi(s)]$
- 其中，TD error 为 $[R(s, \pi(s), s') + \gamma V^\pi(s') - V^\pi(s)]$

主动强化学习/价值决策 Active RL：Q-learning

- 不仅评估状态的价值，还学习如何进行下一步操作
- Q-learning：基于 TD learning，考虑直接用 $Q(s, a)$ 来计算
- $Q(s, a)$：在状态 s 下，执行动作 a 后，之后一直按照最优策略行动所能获得的期望总收益
- $Q(s, a) \leftarrow Q(s, a) + \alpha[R(s, a, s') + \gamma max_{a'} Q(s', a') - Q(s, a)]$
- 更新的策略与实际采样的策略 policy 不一定是同一个策略，可以从非最优策略中学习到最优策略 off-policy learning

探索与利用 Exploration vs. Exploitation

- Exploration 探索：尝试新的方法
- Exploitation 利用：使用当前学习到最优的策略

ϵ-greedy

- 在每一步中，以 ϵ 的概率随机行动，以 $1-\epsilon$ 概率按照当前策略执行

乐观探索函数 Optimistic Exploration Functions

- 设计一个探索方程 Exploration Function，如 $f(u, n) = u + \frac{k}{\sqrt{n}}$，其中 k 为超参数，$n = n(s, a)$ 表示历史探索中在 s 处做 a 操作的次数，鼓励尝试去过次数更少的地方

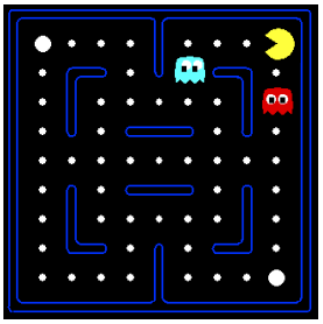
- 将 Q-learning 的转移方程改为
  $Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma max_{a'} f(Q(s',a'), n(s',a')) - Q(s,a)]$
- 这会在原来的基础上，更加鼓励尝试新的方法

基于特征的表征 Feature-Based Representations

- 我们希望把相似（从局势角度来说）的状态表示成相似或相同的状态，以增加模型的泛化性能

Let's say we discover through experience that this state is bad:

In naïve q-learning, we know nothing about this state:



Or even this one!

- 加入人类专家设计的特征，可能会极大简化学习的状态空间

近似Q学习 Approximate Q-learning

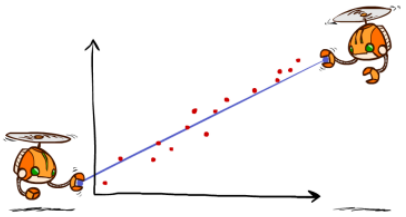
- Q-learning 需要存储一整张 $|S| \times |A|$ 的表格，时空复杂度过高

Imagine we had only one point x, with features f(x), target value y, and weights w:

$$\text{error}(w) = \frac{1}{2}\left(y - \sum_k w_k f_k(x)\right)^2$$

$$\frac{\partial\ \text{error}(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

$$w_m \leftarrow w_m + \alpha\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

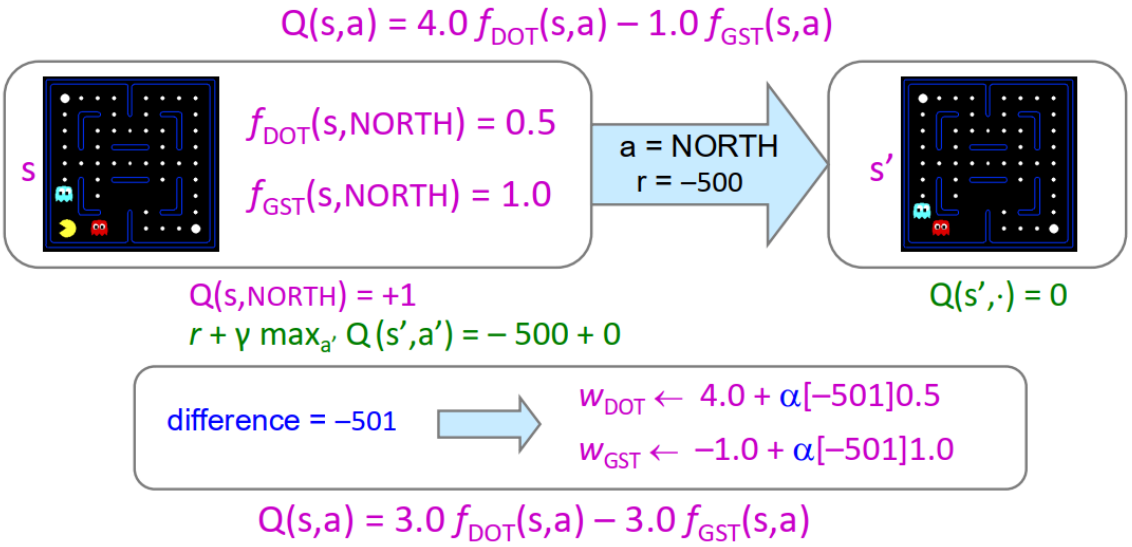


Approximate q update explained:

$$w_m \leftarrow w_m + \alpha\left[r + \gamma \max_a Q(s',a') - Q(s,a)\right] f_m(s,a)$$

"target" "prediction"

Example: Q-Pacman



策略梯度 Policy Gradients

- 跳过所有值和路径 V/Q，直接估计策略 policy
- 从初始的一个基础解开始，通过微调（梯度上升）来优化；不再评估策略的好坏
- 我们希望找到策略 π_θ，对于每种策略，定义价值为 $J(\theta) = \mathbb{E}\left[\sum_{t\geq 0}\gamma^t r_t|\pi_\theta\right]$
- 我们需要找到使得 $J(\theta)$ 最大的 $\theta = \theta^*$

REINFORCE algorithm

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[r(\tau)\right]$$
$$= \int_\tau r(\tau)p(\tau;\theta)\mathrm{d}\tau$$

Where $r(\tau)$ is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

REINFORCE algorithm

Expected reward: $J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau)]$

$$= \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Now let's differentiate this: $\nabla_\theta J(\theta) = \int_\tau r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$

Intractable! Gradient of an expectation is problematic when p depends on θ

However, we can use a nice trick: $\nabla_\theta p(\tau;\theta) = p(\tau;\theta) \frac{\nabla_\theta p(\tau;\theta)}{p(\tau;\theta)} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$

If we inject this back:

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau) \nabla_\theta \log p(\tau;\theta) \right) p(\tau;\theta) \mathrm{d}\tau$$

$$= \mathbb{E}_{\tau \sim p(\tau;\theta)} [r(\tau) \nabla_\theta \log p(\tau;\theta)]$$

Can estimate with Monte Carlo sampling

REINFORCE algorithm

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$

Thus, $\log p(\tau;\theta) = \sum_{t \geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Doesn't depend on transition probabilities!

Therefore when sampling a trajectory τ, we can estimate $J(\theta)$ with

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance reduction

Gradient estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- 对于稀疏奖励的游戏（如围棋），可以采用 Q 和 V 来代替奖励

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)$

Actor-Critic Algorithm

Problem: we don't know Q and V. Can we learn them?

Yes, using Q-learning! We can combine Policy Gradients and Q-learning by training both an **actor** (the policy) and a **critic** (the Q-function).

- The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust
- Also alleviates the task of the critic as it only has to learn the values of (state, action) pairs generated by the policy
- Can also incorporate Q-learning tricks e.g. experience replay
- **Remark:** we can define by the **advantage function** how much an action was better than expected $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$

Lec13 监督学习 Supervised Machine Learning

学习的分类 Types of learning

- 监督学习 Supervised learning：训练数据包括标签（需要的结果）
- 无监督学习 Unsupervised learning：训练数据不包括标签
- 半监督学习 Semi-supervised learning：训练数据包括部分标签
- 强化学习 Reinforcement learning：由一系列操作来获得奖励

监督学习的定义 Supervised learning

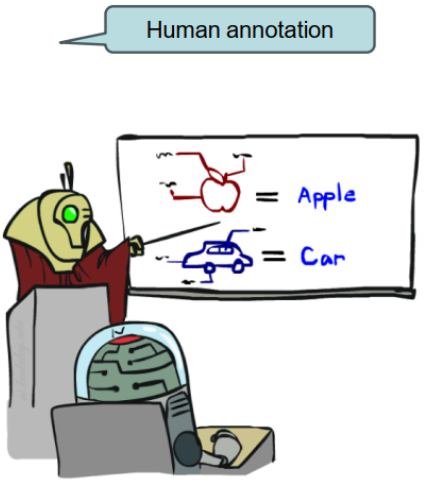
- 输入：训练集，包含带标签的样本 (x_i, y_i)
- 目标：学习一个未知的目标函数 f
- 输出：产生一个接近 f 的假设 h
- 两大任务：分类 Classification（输出离散值），回归 Regression（输出连续值）

- 工作流：给定输入，提取输入信息的特征 feature，使用机器学习算法在这些特征上学习，并输出预测的标签 y

Supervised learning

- To learn an unknown *target function* f
- Input: a *training set* of *labeled examples* (x_j, y_j) where $y_j = f(x_j)$
- Output: *hypothesis* h that is "close" to f

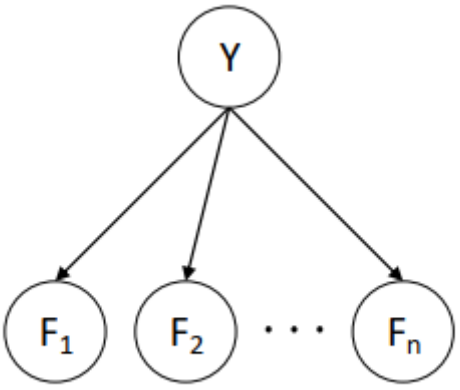
- Typical types of supervised learning
  - Classification = learning f with discrete output value
  - Regression = learning f with real-valued output value
  - Structured prediction = learning f with structured output



分类问题 Classification

朴素贝叶斯网络 Naive Bayes Model

- 生成式模型；本质上是求最大后验概率
- 给定标签 Y 及其概率 $P(Y)$，每个特征 F 都有条件概率 $P(F_i|Y)$



- 使用训练集来估计概率统计表 probability table，总共 $n+1$ 张表：估计 $P(Y)$，估计 $P(F_i|Y)$
- 我们将这些概率统计表的概率值称为参数 parameter
- 进行分类时，使用推断算法 inference algorithm（如变量消除 variable elimination）直接计算 $P(Y|f_1, \cdots . f_n)$
- 假设了特征变量是关于标签条件独立的，简化以达成计算的可行性

参数估计 Parameter Estimation

- 通过统计计数来估计概率，本质上是最大似然估计 Maximum Likelihood
- 我们使用训练数据来估计参数的值，这个过程叫作学习 Learning
- 假设存在参数，计算概率 $P(observation|\theta)$（是关于 θ 的函数）
- 计算最大后验估计 $argmax_\theta P(observation|\theta)$

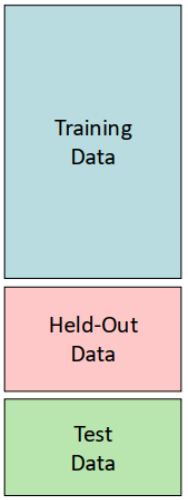
- 我们将数据集分为三部分：训练集 Training set，保留集/验证集 Held-out set，测试集 Test set
- 在一个训练周期中，在训练集上训练模型参数，并使用测试集来检验模型准确性
- 永远不要用测试集数据来修正模型参数
- 过拟合 Overfitting：过度拟合了当前的训练集，导致泛化性能差
- 欠拟合 Underfitting：与训练集拟合程度低

拉普拉斯平滑 Laplace Smoothing

- 假装模型见过每个输出比实际多 `k` 次，使用 $P_{LAP,k}(x) = \frac{c(x)+k}{N+k\|X\|}$
- `k` 为超参数
- 拉普拉斯估计本质上是一种基于数据集的最大后验概率 θ_{MAP}

典型问题

- Typical problems
  - Underfitting: fitting the training set poorly
  - Overfitting: fitting the training data very well, but not the test data

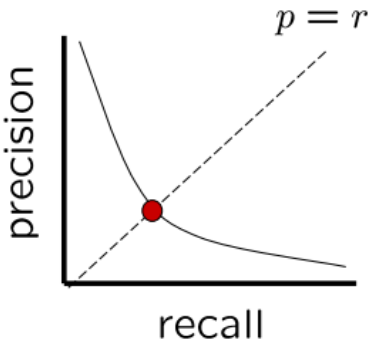


判断指标

- 设定一个基线模型 Baseline，用以对比判定模型的性能是好是坏
- 可以使用全随机作为基线模型，也可以使用之前的研究结果作为基线模型

Precision vs. Recall

- Precision/recall tradeoff
  - Often, you can trade off precision and recall
  - Only works well with weakly calibrated classifiers



- To summarize the tradeoff:
  - Break-even point: precision value when p = r
  - F-measure: harmonic mean of p and r:

$$F_1 = \frac{2}{1/p + 1/r}$$

线性分类器（感知机） Linear Classifier (Perceptrons)

- 输入特征值 feature values，每个特征都有权重 weight，和即为激活函数
- $activation_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$
- 对于二分类问题，输出函数可以是 $-1/1$ 的
- 有时我们加上偏移项 bias $\sum_{i=1}^{n+1} w_i \cdot f_i(x) = \sum_{i=1}^{n} w_i \cdot f_i(x) + w_{n+1}$
- 决策边界 Decision Boundary：如果一个数据点的特征向量点乘分类器 $f(x)$ 为正，则分类为正样本；否则分类为负样本，即 $y_i = sign(w_*^T f(x_i))$

感知机学习算法 Perceptron Learning Algorithm, PLA

- 所有参数 0 初始化
- 对于每个训练集数据点：
  - 如果分类正确，不作改变
  - 如果分类不正确，做出调整：$w = w + y^* \cdot f$ （其中 y^* 是正确的标签答案）
- 循环这个过程，直到对于整个训练集的数据点都能分类正确（依赖于训练集线性可分）

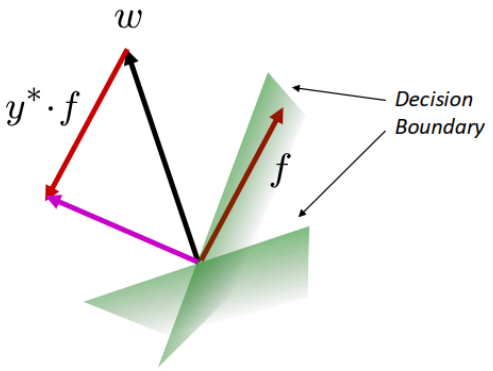
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector.

$$w = w + y^* \cdot f$$



Before: $w \cdot f(x)$
After: $w \cdot f(x) + y^* f(x) \cdot f(x)$ ≥ 0

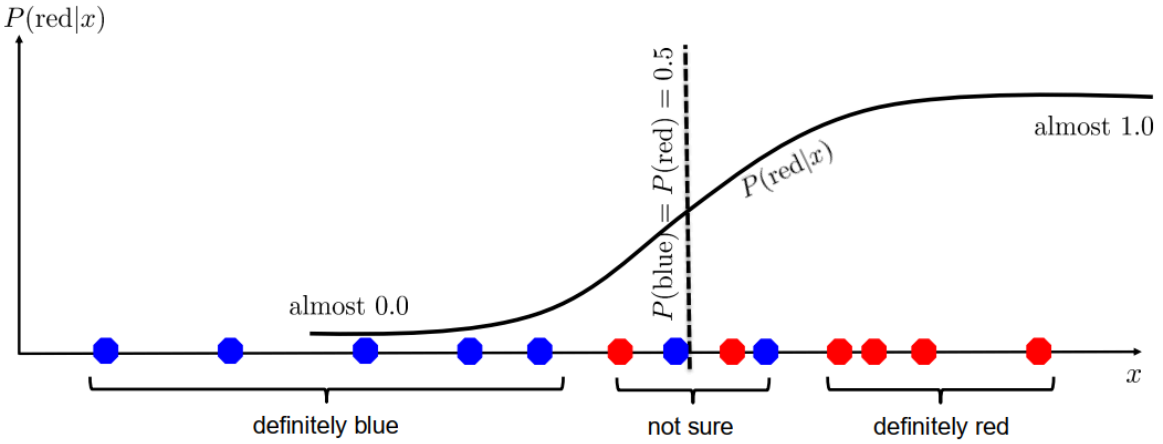
- 可以证明，对于线性可分的情况，PLA算法在有限时间内可以收敛
- 对于多分类的PLA，每个类别有一个权重 w_y，预测分类为 $y = argmax_y w_y \cdot f(x)$；权重更新为 $w_y = w_y - f(x)$；$w_{y^*} = w_{y^*} + f(x)$

感知机的问题

- 如果数据集线性不可分，权重可能会跳变 Thrash
- 对于在决策边界上的点无法处理
- 会过拟合 Overtraining：测试集准确率先上升后下降

逻辑回归 Logistic Regression

- 在感知机模型中，无法处理非线性可分的数据
- 在感知机中，只通过 z 的正负性判断结果，而不论其绝对值大小；这会忽略模型对自信程度的输出
- 激活值 Activation $z = w \cdot f(x) \in (-\infty, +\infty)$
- 逻辑回归使用了概率视角（概率决策 Probabilistic decisions），随着激活值 z 从正到负，输出结果为1的概率逐渐递减，输出结果为0的概率逐渐递增
- 设计一个激活函数 $\phi(z) = \frac{1}{1+e^{-z}}$，这个函数叫做 Sigmoid 函数



- 逻辑回归 Logistic Regression：找到 $max_w ll(w) = max_w \sum_i log P(y^{(i)}|x^{(i)}; w)$，其中 $P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1+e^{-w \cdot f(x^{(i)})}}$；$P(y^{(i)} = -1|x^{(i)}; w) = 1 - \frac{1}{1+e^{-w \cdot f(x^{(i)})}}$

- 对于多类逻辑回归，使用 softmax 作为新的激活值：
  $z_1, z_2, z_3 \rightarrow \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$

梯度上升/下降 Gradient Ascent / Descent

- 在权重参数 w_i 所形成的坐标系上，试图提高 $g(w_1, \cdots, w_i)$，采用梯度上升的做法，即：
  - $w_1 \leftarrow w_1 + \alpha \cdot \frac{\partial g}{\partial w_1}(w_1, w_2)$
  - $w_2 \leftarrow w_2 + \alpha \cdot \frac{\partial g}{\partial w_2}(w_1, w_2)$
  - 写成向量形式，有 $w \leftarrow w + \alpha \cdot \nabla_w g(w)$
- 对于多分类问题，梯度上升解法为：$w \leftarrow w + \alpha \times \sum_i \nabla log P(y^{(i)}|x^{(i)}; w)$
  化简可得 $w \leftarrow w + \alpha \cdot \sum_i (y^{(i)} - P(y=1|x^{(i)})) \cdot x^{(i)}$

神经网络 Neural Networks

- 人工设计特征 feature 是有偏且代价高昂的，而神经网络可以从数据中学习特征
- $z_i^{(k)} = g(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)})$，其中 g 为非线性激活函数
- 常用的 g 函数：
  - Sigmoid Function $g(z) = \frac{1}{1+e^{-z}}$，$g'(z) = g(z)(1-g(z))$
  - Hyperbolic Tangent $g(z) = \frac{e^z - e^{-z}}{e^z + z^{-z}}$，$g'(z) = 1 - g(z)^2$
  - Rectified Linear Unit (ReLU) $g(z) = max(0, z)$，$g'(z) = \begin{cases} 1 & \text{, z>0} \\ 0 & \text{, otherwise} \end{cases}$
- 深度学习 Deep learning：使用深层神经网络（如1000层），使用更复杂的层间连接关系（本课程中不涉及）

回归问题 Regression

回归问题定义

- 学习一个未知的函数 f
- 输入数据集，包括一系列带标签的数据点 (x_i, y_i)，满足 $y_i = f(x_i)$
- 输出一个接近 f 的函数 h

最小均方误差 Minimizing squared error, MSE

- L2 loss function：$L(w) = \sum_i (y_i - h_w(x_i))^2 = \sum_i (y_i - w^T x_i)^2$
- 本质上，最小化 L2 loss function 就是最小化高斯分布噪声的最大似然估计误差，即最小二乘法
- 存在闭式解 $w^* = (X^T X)^{-1} X^T y$

平均绝对误差 Mean Absolute error, MAE

- 最小化平均绝对误差 MAE 本质上是最小化拉普拉斯分布噪声的最大似然估计误差

正则回归 Regularized Regression

- Least Absolute Shrinkage and Selection Operator, LASSO / L1 Regularization
  - $L(w) = \sum_i (y_i - w^T x_i)^2 + \lambda \sum_k |w_k|$
  - 本质上是假设噪声服从拉普拉斯先验
- Ridge Regression / L2 Regularization

- $L(w) = \sum_i (y_i - w^T x_i)^2 + \lambda \sum_k w_k^2$
- 本质上是假设噪声服从高斯先验

非线性回归的过拟合问题 Overfitting in non-linear regression

- “奥卡姆剃刀” Ockham's razor：用最简单的假设来拟合数据/解决问题
- 下图中，E_{in} 表示样本（训练集）平均损失函数，E_{out} 表示样本外（测试集）平均损失函数

Empirical risk (training error, in-sample error)
Empirical Risk Minimization (ERM)

- The expected loss of a predictor h on a particular observed sample data set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ could also be referred to as the empirical risk $E_{in}(h)$

$$E_{in}(h) = \frac{1}{m} \sum_{i=1}^{m} [\text{loss}(h(x_i), y_i)] = \underbrace{\frac{1}{m} \sum_{i=1}^{m} \{h(x_i) = y_i\}}_{\text{0-1 loss}}$$

- Usually, the target predictor is found by ERM

$$\hat{h} = \arg\min_{h} \; E_{in}(h), \qquad h \in \mathcal{H}$$

- Parameterize $h(\cdot\,; \theta) \iff \theta$

$$\hat{\theta} = \arg\min_{\theta} \; E_{in}(\theta), \qquad \theta \in \Theta$$

- If $E_{in}(h) = 0$ on $\mathcal{D}$, we say h agrees with $\mathcal{D}$ (与$\mathcal{D}$一致)

Approximation-generalization tradeoff

- Small E_{out}: good approximation of f out of sample
- More complex — better chance of approximating f
- Less complex — better chance of generalizing out of sample
- Ideal $\mathcal{H} = \{f\}$
- Bias-variance analysis: decomposing E_{out} into
    A. How well $\mathcal{H}$ can approximate f
    B. How well we can zoom in on a good $h \in \mathcal{H}$
- Applies to real-valued targets and uses squared error

Decomposing E_{out}

$$E_{\text{out}}(g^{(\mathcal{D})}) = \mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

$g^{(D)}$ is derived from repeatedly sampled dataset D

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right]\end{aligned}$$

Now, let us focus on:

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

$\mathbb{E}_D\left[\left(g^{(D)}(x) - f(x)\right)^2\right]$ is the average loss on x over all the sampled datasets.

The average hypothesis

To evaluate $\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$

we define the 'average' hypothesis $\bar{g}(\mathbf{x})$:

$$\bar{g}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})\right]$$

Imagine **many** data sets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K$

$$\bar{g}(\mathbf{x}) \approx \frac{1}{K}\sum_{k=1}^{K} g^{(\mathcal{D}_k)}(\mathbf{x})$$

$\mathbb{E}_D\left[\left(g^{(D)}(x) - f(x)\right)^2\right]$ is the average loss on x over all the sampled datasets, and $\mathbb{E}_D[g^{(D)}(x)]$ is the average prediction on x over all the sampled datasets

Using the average hypothesis

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-f(\mathbf{x})\right)^2\right]=\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})+\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2\right]$$

$$=\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2+\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2\right.$$

$$\left.+\ 2\ \left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)\ \left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)\right]$$

Where is this term? $\bar{g}(\mathbf{x})=\mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})\right]$

$$=\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2\right]+\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2$$

Bias and Variance

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-f(\mathbf{x})\right)^2\right]=\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}(\mathbf{x})}+\underbrace{\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2}_{\text{bias}(\mathbf{x})}$$

$$\text{Therefore,}\quad \mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right]=\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-f(\mathbf{x})\right)^2\right]\right]$$

$$=\mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x})+\text{var}(\mathbf{x})]$$

$$=\quad \text{bias}\quad +\quad \text{var}$$

Lec14 无监督学习 Unsupervised Learning

聚类问题 Clustering

- 给一个数据集进行分类的任务

K-means 算法

- 算法流程：
  - 给定分成 k 类（超参数），随机（或按照规则）初始化 k 个聚类中心
  - 循环：对于每个点，把它归类到距离最近的聚类中心所在的类中 $a_i = argmin_k dist(x_i, c_k)$
  - 把每个聚类中心移动到该类的重心处 $c_k = \frac{1}{|\{i:a_i=k\}|}\sum_{i:a_i=k} x_i$
  - 循环以上过程，直到某一轮迭代，所有的点都不再改变所在的类

- 初始化相关，不同的初始化可能导致不同的结果
- 归纳偏置 Inductive Bias：修改距离函数 dist，如减少水平距离来偏向于归类横向条带状点，或映射到极坐标系来归类环状点带

凝聚层次聚类 Agglomerative Clustering

- 算法流程：
  - 总共 m 个样本点，每个点初始化为一个聚类 Cluster
  - 每次选两个最近的聚类 Cluster 进行合并，直到所有点合并为一个聚类 Cluster
  - 取第 m-k 次聚类后的状态，作为最终分类
- 对于两个聚类之间的距离，可以取以下的任何标准之一：
  - 最近对之间的距离（单连接聚类）
  - 最远对之间的距离（全连接聚类）
  - 所有点对的距离均值
  - 使得新聚类内方差增幅最小 Ward's method
- 在生物信息学中较为常用

期望最大化 Expectation-Maximization, EM

- 对于高斯分布 $P(x|\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
- 取 $p(x)=\sum_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$
- 其中 $\forall k,\pi_k\geq 0,\sum_{k=1}^{K}\pi_k=1$
- 对于如上的若干个高斯分布生成的数据，如果已知每个数据点由哪个高斯生成，则可以直接使用统计方法进行聚类并估计出分布
- 算法流程：
  - 随机初始化若干个高斯分布
  - [E step] 按照高斯分布，给第 i 个样本计算属于第 j 个高斯分布的概率 $Q_{ij}\propto e^{-\frac{d_{ij}^2}{2}}$
  - [M step] 基于上一步，每个点按照其概率分配权重，计算新的高斯分布参数 $\mu_j=\frac{\sum_i Q_{ij}x_i}{\sum_i Q_{ij}}$ $\sigma_j^2=\frac{\sum_i Q_{ij}d_{ij}^2}{\sum_i Q_{ij}}$
  - 重复以上过程，直到收敛

EM for GMM

Iterate: On the *t*'th iteration let our estimates be

$$\theta^{(t)} = \{ \mu_1^{(t)}, \mu_2^{(t)} \dots \mu_k^{(t)}, \Sigma_1^{(t)}, \Sigma_2^{(t)} \dots \Sigma_k^{(t)}, \pi_1^{(t)}, \pi_2^{(t)} \dots \pi_k^{(t)} \}$$

E-step

Compute label distribution of each data point

$$P\left(y_j = i \left| x_j, \theta^{(t)}\right.\right) \propto \pi_i^{(t)} N\left(x_j \left| \mu_i^{(t)}, \Sigma_i^{(t)}\right.\right)$$

Just evaluate a Gaussian at x_j

M-step

Compute weighted MLE of parameters given label distributions

$$\mu_i^{(t+1)} = \frac{\sum_j P\left(y_j = i \left| x_j, \theta^{(t)}\right.\right) x_j}{\sum_{j'} P\left(y_{j'} = i \left| x_{j'}, \theta^{(t)}\right.\right)} \quad \Sigma_i^{(t+1)} = \frac{\sum_j P\left(y_j = i \left| x_j, \theta^{(t)}\right.\right) [x_j - \mu_i^{(t+1)}][x_j - \mu_i^{(t+1)}]^T}{\sum_{j'} P\left(y_{j'} = i \left| x_{j'}, \theta^{(t)}\right.\right)} \quad \pi_i^{(t+1)} = \frac{\sum_j P\left(y_j = i \left| x_j, \theta^{(t)}\right.\right)}{m}$$

m = #training examples

- 当 EM 的高斯都取球形和相同的权重、方差，且计算样本属于每个高斯分布的概率仅取最大值时，EM 退化为 K-means
- Coordinate Ascent
  - 找到一组参数 θ，使得 $l(\theta : \mathcal{D}) = \sum_{j=1}^{m} logP(x_j|\theta)$ 取最大值
  - 对于隐变量 z，边缘化有 $l(\theta : \mathcal{D}) = \sum_{j=1}^{m} log \sum_z P(x_j, z|\theta)$
  - 引入任意分布 Q，有 $l(\theta : \mathcal{D}) = \sum_{j=1}^{m} log \sum_z Q(z|x_j) \cdot \frac{P(x_j,z|\theta)}{Q(z|x_j)}$
  - 构造期望表达，有 $l(\theta : \mathcal{D}) = \sum_{j=1}^{m} logE_Q \left[\frac{P(x_j,z|\theta)}{Q(z|x_j)} \right]$
  - 根据 Jensen 不等式，有 $l(\theta : \mathcal{D}) \geq \sum_{j=1}^{m} E_Q \left[log \frac{P(x_j,z|\theta)}{Q(z|x_j)} \right]$
  - 即有对数似然下界 $l(\theta : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_z Q(z|x_j) log \frac{P(z,x_j|\theta)}{Q(z|x_j)} := F(\theta, Q)$
  - E-step 固定 θ 优化 Q，此时 Jensen 不等式取等：$Q(z|x_j) = P(z|x_j, \theta)$
  - M-step 固定 Q 优化 θ，此时即求 $\theta^{(t+1)} = argmax_\theta \sum_j E_Q[logP(x_j, z|\theta)]$

Lec15 大语言模型 Large Language Models

背景与定义

- 大语言模型 LLM 的三个核心维度
  - 规模 Scaling: 如何利用基础设施 Infrastructure 和分布式系统 Distributed System 实现模型扩展
  - 任务形式 Language: 将各类任务转化为语言处理，涉及数据 Data, 提示词 Prompt 和应用 Application
  - 模型设计 Models: 涵盖机器学习建模 ML Modeling 和优化 Optimization

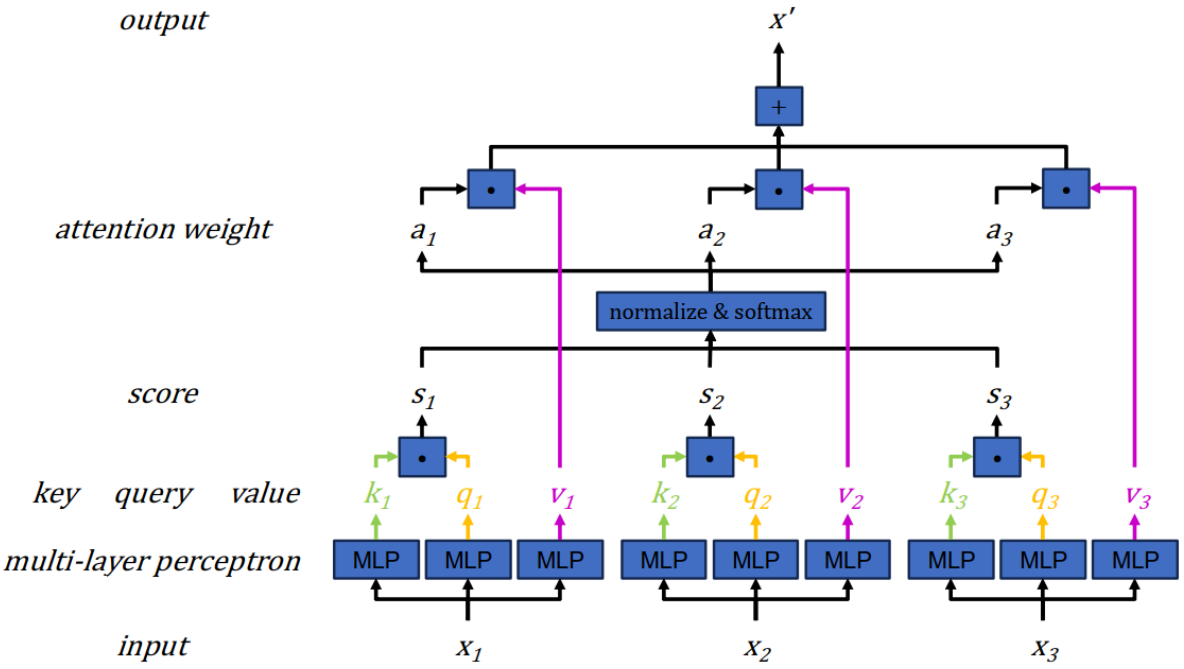
特征工程 Feature Engineering

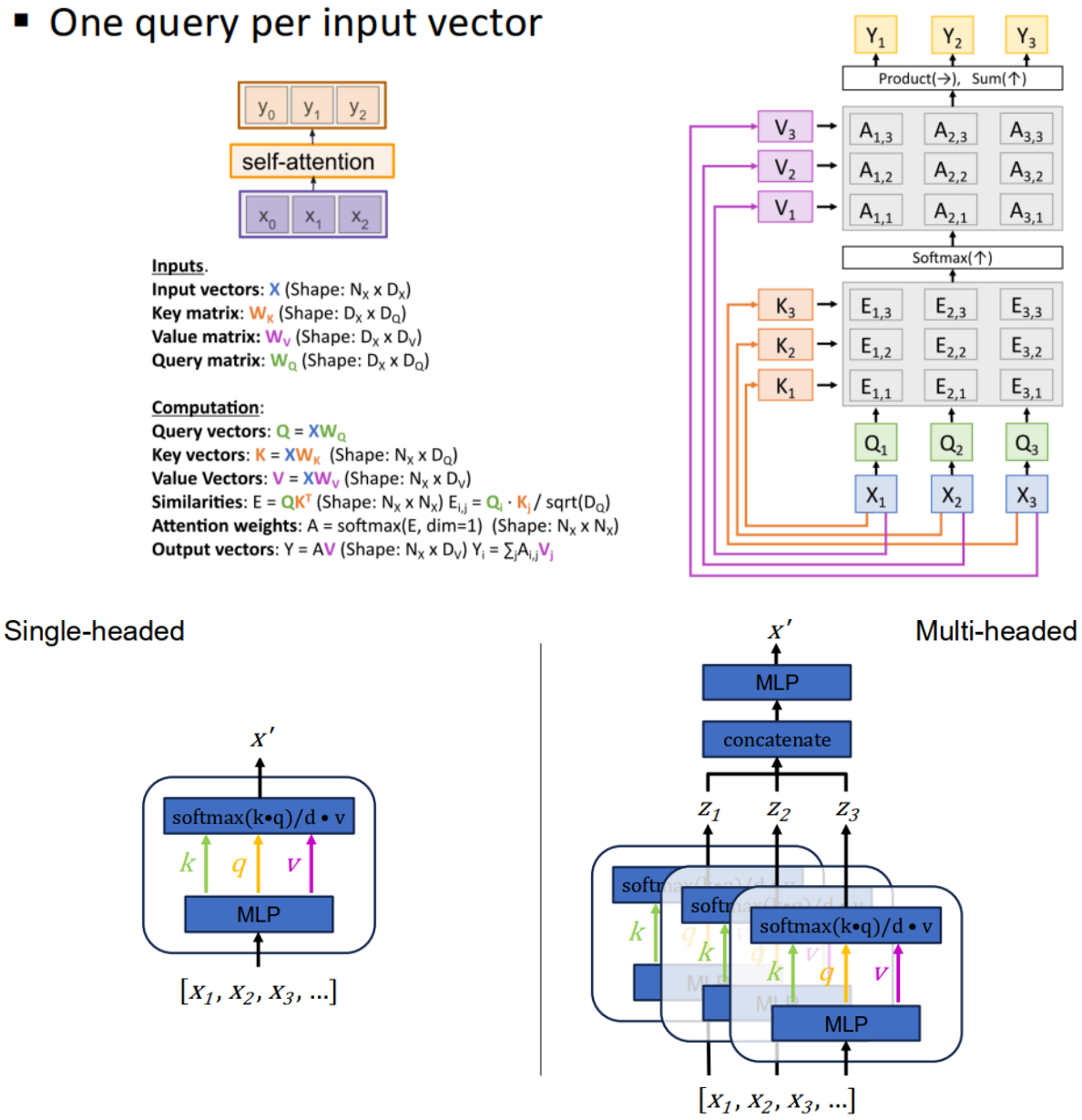
- 文本分词 Text Tokenization
  - 将文本输入映射为离散标记 Tokens，进而转化为标记标识符 Token IDs

    - 常用优化算法包括字节对编码 BPE 和 WordPiece
    - 复杂的单词或特殊符号可能被拆分为多个标记
- 词向量 Word Embeddings
    - 通过 Word2Vec 等模型将标记映射到连续的嵌入空间 Embeddings
    - 具有相似语义的单词在向量空间中会聚类 Cluster
    - 嵌入空间展现出向量代数特性 Algebra: 例如国王 King - 男人 Man + 女人 Woman ≈ 女王 Queen

模型架构 Model Architecture

- 演进路径
    - 深度神经网络 DNN -> 循环神经网络 RNN -> 变换器 Transformer
    - 任务层级从编码器 Encoder, 解码器 Decoder 演变为编码器-解码器 Encoder-Decoder，现代 LLM 多采用纯解码器架构 Decoder-only
- 自回归模型 Autoregressive Models
    - 在每个步骤预测一个标记，并将其作为下一步的输入
    - 模型捕捉历史上下文 Context 下的条件分布 $p(o_t|o_1, \ldots, o_{t-1})$
    - 整个句子的联合分布 Joint Distribution 是每步条件概率的连乘
- 注意力机制 Attention Mechanisms
    - 核心思想是为输入标记分配不同的权重，关注相关标记 Relevant tokens
    - 通过查询 Query, 键 Key, 值 Value 进行计算
    - 变体包括多头注意力 MHA, 分组查询注意力 GQA 和 多查询注意力 MQA





- 因果注意力 Causal Attention
  - 在自回归生成时使用掩码 Masking，确保当前标记只能关注到位置在前的标记，防止信息泄露
- 变换器架构 Transformer Architecture
  - 由层归一化 LayerNorm, 多头注意力 MHA, 残差连接和前馈神经网络 FFN 组成
  - 引入位置嵌入 Positional Embeddings 来捕捉序列顺序，包括绝对位置编码, 相对位置编码和旋转位置编码 RoPE

训练与学习阶段 Learning

- 预训练 Pre-training
  - 在海量数据上通过自监督学习 Self-supervised Learning 进行下一个标记预测 Next token prediction
  - 取 $argmax_\theta log \sum_{t=1}^{T} logp(o_t = y_t | o_1 = y_1, \cdots, o_{t-1} = y_{t-1}; \theta)$
  - 目标是学习通用语言特征
- 指令微调 Instruction Tuning
  - 在人类对话示例上进行监督学习 Supervised Learning
  - 使模型从模仿人类文本 Mimic human-written text 转向生成有用文本 Generate helpful text

- 基于人类反馈的强化学习 RLHF
    - 建立奖励模型 Reward Model 来评估回答质量
    - 通过策略搜索 Policy search 和策略梯度方法 Policy gradient methods（如 PPO 或 GRPO）优化策略

推理与解码 Decoding

- 贪心搜索 Greedy Search: 每步仅选择概率最高的标记
- 束搜索 Beam Search: 保留得分最高的多个候选束 Beam，通过束大小 Beam size 控制搜索范围，计算成本高于贪心搜索，但是能搜索一段一定长度序列的总体最优解
- 其他方法包括采样 Sampling 和投机解码 Speculative decoding

推理增强与前沿技术

- 推理模型 Reasoning LLM
    - 思维链 CoT: 通过引导步骤 Trigger step by step 让模型在给出答案前先生成推理过程
    - 自我演化 Self-evolution: 模型在强化学习过程中学会 反思 Reflect 并探索替代解决方案
- 深度求索 DeepSeek-R1
    - 采用组相对策略优化 GRPO，通过组内得分估计基线以节省计算成本
    - 模型展现出顿悟时刻 Aha moment，能够自主纠正推理错误并生成更长的推理路径
- 多模态大模型 MLLM
    - 目标是构建能处理图像、音频、视频等所有数据类型的统一模型
    - 代表架构如 LLaVA，使用投影矩阵 Projection W 连接视觉编码器 Vision Encoder 和 语言模型 Language Model
- 智能体能力 Agentic abilities
    - LLM 作为大脑解决复杂的现实任务
    - 涉及规划 Planning, 工具调用 Tools 和外部知识检索 RAG 等闭环交互流程