No.

Date

CS181 期末复习 BN Exact Inference

BN is directed acyclic graph. It implicitly encode 联合分布.

$$P(x_1, \cdots, x_n) = \prod_{i=1}^{n} P(x_i \mid \text{parents}(X_i))$$

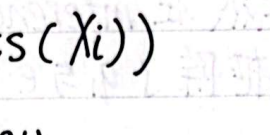


fig1

Eg. $P(+b, +e, +a, +j, +m)$ (fig1)

$= P(+b)\, P(+e)\, P(+a \mid +b, +e)\, P(+j \mid +a)\, P(+m \mid +a)$

Outline: Enumeration. Variable Elimination. NPC. Sampling

① By enumeration: evidence: $E_1, \cdots, E_k = e_1, \cdots, e_k$

Query: Q Hidden variables: $H_1 \cdots H_r$ and Evidence

欲求 $P(Q \mid e_1, \cdots, e_k)$, 实际求: sum out H to get joint of Query

$$= \sum_{h_1, \cdots, h_r} P(Q, h_1, \cdots, h_r, e_1, \cdots, e_k) \times \frac{1}{\sum} \quad \text{normalize}$$

如 fig1 中: $P(B \mid +j, +m) = \frac{1}{z} \sum_{e,a} P(B)\, P(e)\, P(a \mid b, e)\, P(+j \mid a)\, P(+m \mid a)$

这个式子完全正确, 但计算 $\sum$ 中大量重复项, 不高效

② Variable Elimination: 两步:

a. 把求和操作尽可能 move inward

b. 之后, 从内至外作计算

$$P(B \mid j, m) = \alpha \sum_{e,a} P(B)\, P(e) P(a \mid B, e)\, P(j \mid a)\, P(m \mid a)$$

$$= \alpha\, P(B) \sum_e P(e) \sum_a P(a \mid B, e)\, P(j \mid a)\, P(m \mid a)$$

思路: 固定其余的 (e), 让一个 sum (a)

之后计算流程: sum over a, then sum over e

注意到 $P(a \mid B, e)$, 启示: 不应以数为单位, 而应以 Factor

Def: Factor: multi-dimensional array to represent $P(Y_1 \cdots Y_N \mid X_1 \cdots X_m)$

KOKUYO

No.

Date

R → T

Operation 1: Join Factors: $P(R) \times P(T|R) \Rightarrow P(R,T)$

2: Eliminate: $P(R,T) \xrightarrow{\text{sum R}} P(T)$

消除变量法依然在 inference 问题上 NP-hard，但比 enum 快多了

在 inference 中，把除了 q_1 与 e 的其它 Hidden variables

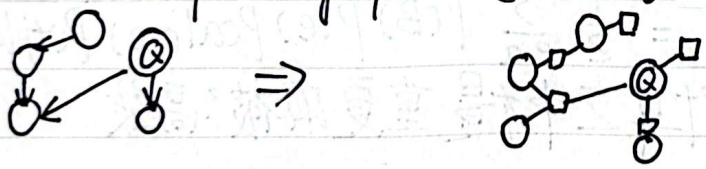
VE 消除变量顺序也十分重要，can greatly affect the size of the largest factor

△：是否有算法总能返回产生最小 factors 的 order？不存在！

这是个 NP-Hard 问题，可从 3SAT 归约至此问题

但是对于 poly-tree* BN 这种特殊 BN，VE 复杂度 is linear in the BN size if 采取如下策略：

① convert to factor graph ② take Q as root ③ Eliminate from leaves towards root



* Poly-tree: A directed graph with no undirected cycles.

No.

Date

Approximate Inference

Sampling: Goal: P Basic idea: 在采样分布S下抽N个sample，进而计算 quantity，其收敛至P
此方法快而简单，且内存开销小（O(n)）

Prior Sampling: For $i=1,2,\cdots,n$ (topological order!)
Sample X_i from $P(X_i \mid parents(X_i))$
最后样本的频率代表 BN's 联合分布概率！
i.e., the sampling procedure is consistent

Reject Sampling: 在 prior 基础上微改：即：与evidence不一致的采样
e.g., 欲近似 $P(C \mid +a,+b)$，则在 prior 采样过程中，若 [A≠+a or B≠+b]，立即拒绝」该样本
It is consistent for conditional probabilities !!

Likelihood Weighting: reject 采样的问题在于：若 evidence 发生概率很低，那么便会拒掉很多 sample.
Algorithm: Input: evidence $e_1,\cdots,e_k$

```
w = 1.0
for i = 1, ···, n
    if Xi is an evidence variable
        xi = ei (observed)
        w = w * P(xi | parents(Xi))
    else: Sample xi from P(Xi | parents(Xi))
return (x1, x2, ···, xn), w
```

下游 Var 受上游影响，但上游不受下游影响！
It is consistent for conditional probabilities !!

KOKUYO

No.

Date

Gibbs Sampling: Generate each sample by making a random change to preceding sample

固定 evidence，对剩下的 non-evidence var，sample its value conditioned on all other var, i.e., var in Markov Blanket

流程：fix evidence → initialize (Randomly 随机选)

→ 选一个非 evidence var，在 Markov Blanket 分布下采样它

→ 不断重复上一个操作

△: Theorem: Gibbs sampling is consistent

综上：prior sampling 针对任意 probability，而 Reject、Likelihood 与 Gibbs sampling 是针对固定的一个条件概率的

Probabilistic Temporal Models

Value of X: state

Markov Models: $X_0 \to X_1 \to X_2 \to X_3 \to \cdots$

Stationary Assumption: $P(X_t|X_{t-1})$ 不变! $P(X_0)$ $P(X_t|X_{t-1})$ → Transition Model

则 $P(X_0, \cdots, X_T) = P(X_0) \prod_t P(X_t|X_{t-1})$

△: "可也"不可"视之为BN! "可"是因为它是DAG, "不可"是因为它无限多变量

在Model中, X_{t+1} 与 $X_0, \cdots, X_{t-1}$ 是条件独立的 given X_t! 即只与上一时间步有关, 故称 first-order Markov Model

(k-order allows dependency on k earlier steps)

Forward Algorithm: 推理t时Xt分布 (given $P(X_0)$)

$$P(X_t) = \sum_{x_{t-1}} P(X_t, X_{t-1} = x_{t-1}) = \sum_{x_{t-1}} P(x_{t-1}) P(X_t | X_{t-1} = x_{t-1})$$

probability of previous iteration; Transition model

考场上灵机应变作 vectorization

观察到: 对于大多数chains: initial分布影响 → end up with a distribution independent of initial distribution ⇒ Stationary分布

$$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x) P_\infty(x)$$

Hidden Markov Models: $X \to E$ (Evidence)

New! Emission Model: $P(E_t|X_t)$ $X_0 \to X_1 \to X_2 \to \cdots$, $X_1 \to E_1$, $X_2 \to E_2$

$P(X_0, X_1, E_1, \cdots, X_T, E_T)$

$= P(X_0) \prod_{t=1:T} P(X_t|X_{t-1}) \cdot P(E_t|X_t)$

先 transit 再 emit

Inductive Bias: E_t 与其它所有条件独立 given X_t

Notation: $X_{a:b} = X_a, X_{a+1}, \cdots, X_b$

KOKUYO

No.

Date

Task: Filter: $P(X_t|e_{1:t})$: 在给定 $e_{1:t}$, 推 X_t 后验分布

如何求它? 注意到:

$$P(X_{t+1}|e_{1:t+1}) = P(X_{t+1}|e_{1:t}, e_{t+1}) \quad \text{Bayes' Rule}$$

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

$$= \alpha\ P(e_{t+1}|X_{t+1})\ [P(X_{t+1}|e_{1:t})] \rightarrow \text{predict}$$

normalize ↑

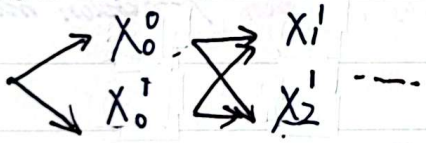
$$= \alpha\ P(e_{t+1}|X_{t+1}) \sum_{x_t} P(x_t|e_{1:t})\ P(X_{t+1}|x_t)$$

emit; ↪ $P(X_t|e_{1:t})$; transit

Cost Per Time Step: $O(|X|^2)$ where $|X|$ is the number of states

Trick: αs are constant! 只需最后一步时normalize即可。

Another view:



每个arc代表 $x_{t-1} \rightarrow x_t$, 权重视为: $P(x_t|x_{t-1}) \cdot P(e_t|x_t)$

Each path → a sequence of states

⇒ Path上 weight 的乘积 is proportional to that state sequence's probability。故all path便是在filter:

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

可用dynamic programming。除了Filter, 有上述view还可:

Task: Most Likely Explanation: $\text{argmax}_{x_{0:t}} P(x_{0:t}|e_{1:t})$

↪ viterbi中顶

Viterbi算法: $\text{argmax}_{x_{0:t}} P(x_{0:t}|e_{1:t}) = \text{argmax}_{x_{0:t-1}} \{\text{argmax}_{x_t} P(x_{0:t}|e_{1:t})\}$

$$P(x_t|x_{t-1}) \cdot P(e_t|x_t)]$$

⇒知 Π_{t-1} 情况, 则在 $\sum_{x_{t-1}} [P(x_t|x_{t-1})\ P(e_t|x_t)]$

找最大项即其对应 x_t。注意过程中, 推 X_{t+1} 时, 要用到:

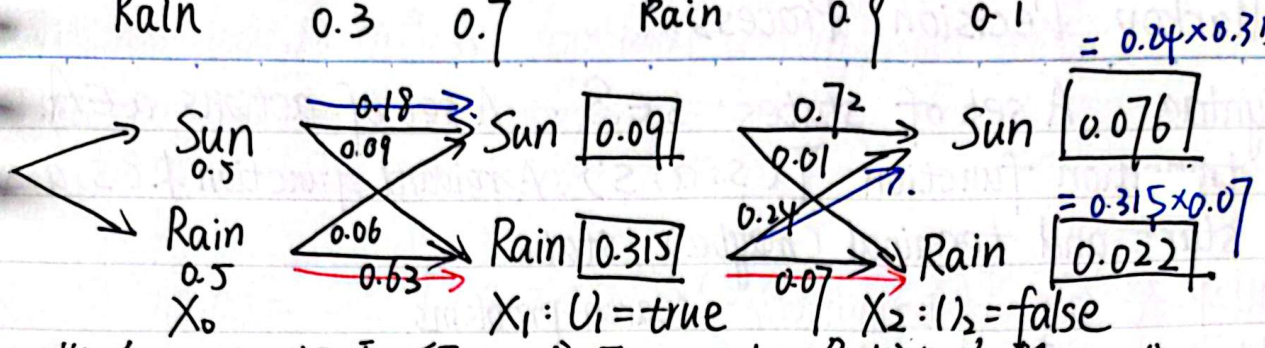
Π_i 情况 & $x_i \rightarrow x_{i+1}$ arc's weight (而此weight要用 e_i), 并记录路径.

Campus

Eg.

W_{t-1}	$P(W_t \mid W_{t-1})$ Sun	Rain
Sun	0.9	0.1
Rain	0.3	0.7

W_t	$P(U_t \mid W_t)$ true	false
Sun	0.2	0.8
Rain	0.9	0.1

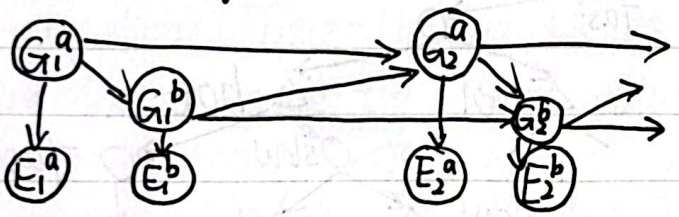


蓝红 path都需留！推至 e_t 时，谁代表的数大，选哪个 x_t，然后按照 path 倒推回 $x_{0:t-1}$

Time & Space Complexity: $O(|X|^2 T)$ & $O(|X|T)$

Dynamic Bayes Net (DBN)

Track Multiple Hidden Variables via Multiple Evidence



→ Observation：算每个particle的权重从1变为多少；(归一化)

Resample：从N个样中以weight为概率，放回地抽N次

每个HMM都是DBN，且每个离散DBN也能用一个HMM表达

Inference in DBN: Particle Filtering* 近似解

(Track samples of X, not all values; Samples are called particles)

则 $P(X)$ 实则是用 a list of N particles 表示的

一般，$N << |X|$。N↑，越精确

X_0 时，particles 在 $P(X_0)$ 下分布。之后，每个 sample 在 $P(X_{t+1} \mid x_t)$ 下采样，并予以(*) Weight = $P(e_t \mid x_t)$。t步后，有N个样本们以及 weight，加和便是 X_t 分布 (normalize)，便可用于 t+1 时 inference. 很像 likelihood weighting！| 每个sample在 $P(X_{t+1} \mid x_t)$ 下 propogate 采样，weight = $P(e_{t+1} \mid x_t)$。用此时的加权样本 ~~$P(X_{t+1})$~~ Resample N个样本且 weight 为1，然后 propogate…

△ 很像 likelihood weighting，但它对于HMM不适合，因为 sample 权重掉的太快；第t步resample后，一个state中样本数量占比代表 t时该 state 的概率分布

No.

Date

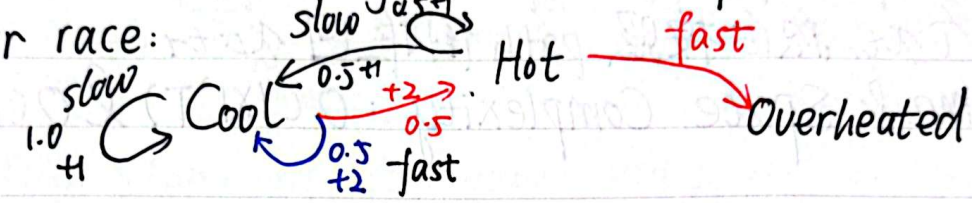
Markov Decision Process

Syntax: A set of states $s \in S$. A set of actions $a \in A$.
A transition function $T(s, a, s')$. A reward function $R(s, a, s')$
A start and terminal (maybe) state

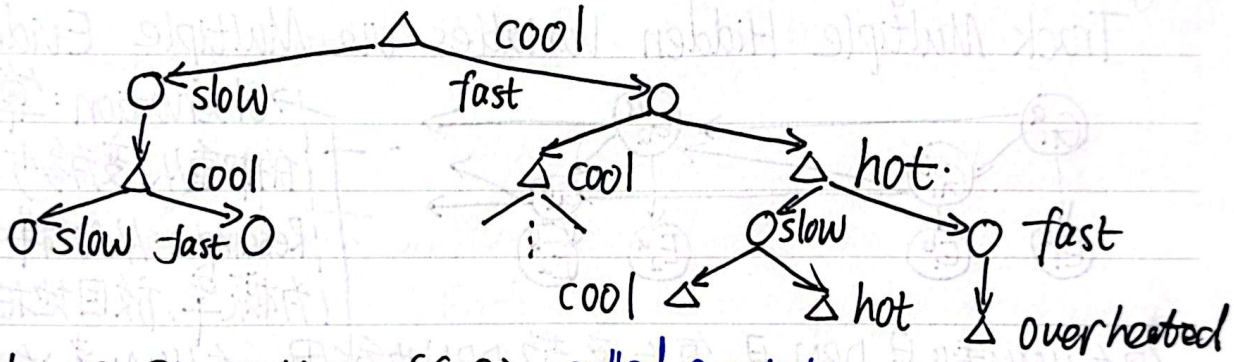
MDP 是 non-deterministic search problems.

Policy: $\pi^*: S \to A$: π gives an action for each state

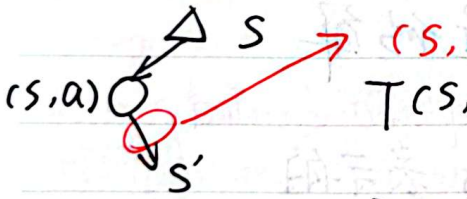
Example: car race:



故: MDP state projects an expectimax-like search tree



△: state, s ○: action, (s,a), called q_n-state



(s,a,s') is called a transition

$T(s,a,s') = P(s'|s,a)$, 且有 $R(s,a,s')$

对于每个s时, 看眼前与未来的收益都很重要! 一种权衡法是 Discounting, How? Level↑, 就给那个收益乘上 γ^i 因子。本level, $i=0$, 再下一层, $i=1$, --

Solving DMP:

value of s: $V^*(s)$ = expected utility starting from s and act optimally.

value of q_n-state (s,a): $Q^*(s,a)$ = expected utility starting out having taken action a from s and thereafter acting optimally

No.

Date

Optimal Policy: $\pi^*(s)$ = optimal action from state s

How to compute? Recursive Definition:

$V^*(s) = \max_a Q^*(s,a)$, 而:

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

故: $V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

这被称为 Bellman Equation

但这种计算方式有2个问题: ① States are repeated

② Tree goes on forever (1种解法: depth-limited)

Value-Iteration: Time-Limited Values

Define $V_k(s)$ to be the optimal value of s if the game ends in k more steps. 等价地: depth-k expectimax

从 $V_0(s)=0$ 开始, 递归时: 若有 $V_k(s)$, 推 $V_{k+1}(s)$:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

重复直至收敛。每一次递归: complexity 为 $O(S^2A)$

△: Theorem: will converge to unique optimal values

($[V_{k+1}(s)] \Rightarrow V_{k+1}(s)$)

Policy Extraction: 假设 $V^*(s)$ 已有。那么, 如何act呢?

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

来以此行动。上述便是 value-iteration.

Q-value Iteration: (流程与原理上与value iteration近似)

从 $Q_0(s,a)=0$ 开始, 若知 $Q_k(s,a)$ (即全部信息)

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

Repeat till converge. (通常比Value Iteration更快收敛)

KOKUYO

No.

Date

Policy Iteration: 一种 value iter 的 alternative approach

Policy Evaluation → Policy Improvement → Repeat till converge!

Step1: 评估: Define $V^{\pi}(s)$ = expected utility starting in s and following fixed π:

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')\left[R(s, \pi(s), s') + \gamma V^{\pi}(s')\right]$$

* 有fixed policy，算utility 巨快!

显然此式要 iterate 以 converge，引入 $V_k^{\pi}(s)$:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')\left[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')\right]$$

Or: $$\overrightarrow{V^{\pi}(s)} = \sum_{s'} T(s, \pi(s), s')\left[R(s, \pi(s), s') + \gamma \overrightarrow{V^{\pi}(s')}\right]$$

Step2: 改进: $\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^{\pi_i}(s')\right]$

Repeat the two steps until policy converges (此式子)

Δ: 要做 policy extraction ☆☆

Summary: Value Iteration: update state value, don't track policy. Finally use state value to extract policy

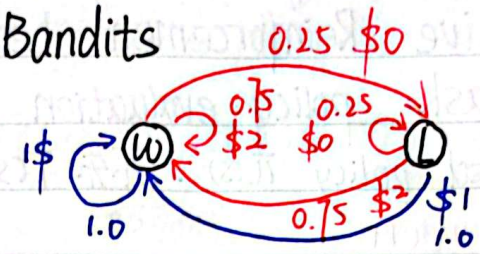
Policy Iteration: May converge faster. After the policy is evaluated, a new policy is chosen

Reinforcement Learning

Intro Example: Double Bandits

Action: Blue, Red

States: Win, Lose



解MDP是offline planning. 但这意味着：你需要知道所有的信息，包括 s take a 后 s'抵达概率分布

若概率不知，只知reward呢？甚至reward也不知呢？

那么在play的时候，实则在'learning'(RL)。这的确是MDP问题，但仅compute是绝不了的。Need to act, to figure it out.

Key ideas:

Exploration vs. Exploitation. ⇒ try unknown actions to gain info & eventually have to use what we know

强化学习中，假设是MDP：有 states $s \in S$，有actions.

有model $T(s,a,s')$，有reward function $R(s,a,s')$

我们 seek for $\pi(s)$，但不知T or R

Basic idea: 行动，然后观察结果（new states, rewards）。nb之学习，然后学着最优化期望收益

Model-based Learning

先用experience学近似model，然后运用它as if the learned model is correct

Step 1: Learn Empirical MDP Model

数对于 s,a 下的 s' 分布，归一化后以代表 $\hat{T}(s,a,s')$；并且发现每个 $\hat{R}(s,a,s')$（当从s采取a抵达s'）

Step 2: Solve the MDP

KOKUYO

No.

Date

Model-Free Learning

这属于 Passive Reinforcement Learning.

Simplified task: policy evaluation　　learn the state values

Input: a fixed policy $\pi(s)$. 不知 $T(s,a,s')$ & $R(s,a,s')$ ⇒ Goal:

Direct Evaluation

在这种范式下，按照 policy 行动。但这不是 offline learning!

Act according to π. 每次到一个 s，记录下：

s 之后的 the sum of discounted rewards；然后求和

如：

Episode 1　B, →, C, -1　　2. B, →, C, -1

C, →, D, -1　　C, →, A, -1

D, exit, x, 10　　A, exit, x, -10

Direct Evaluation Example

则此时 State Value 记算：Epi 1 中：B 之后：8，C 之后 9，D 10

Ep 2 中. B 之后 -12，C 之后 -11，A 之后 -10

则 B: $\frac{-12+8}{2} = -2$，　C: $\frac{9-11}{2} = -1$，A: -10. D: 10，E: ?

Direct Evaluation 有一些缺点：浪费了 state 之间的联系. 每个 state 单独学

新方法：Policy Evaluation

Simplified Bellman updates calculate V for a fixed policy.

$V_0^{\pi}(s) = 0$，$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$

But don't know T & R! How can we update V without T & R?

Idea: Take samples of outcome s' (by doing the action!)* and 平均

$sample_i = R(s, \pi(s), s_i') + \gamma V_k^{\pi}(s_i')$，$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$

*: 注意：can't rewind 故此 idea X!

New idea: Update $V(s)$ each time 经历一个 (s, a, s', r)

① $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

② $V^{\pi}(s) = (1-\alpha)V^{\pi}(s) + \alpha\, sample = V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

这个更新下：make recent samples more important

此法称为 Temporal Difference Learning

TD learning 是 model-free way to do policy evaluation. 但若想利用评估的 values 转化为 new policy:

$$\pi(s) = \arg\max_a \left[\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V(s')]\right]$$

$T(s,a,s')$ 仍未知　$R(s,a,s')$ 仍未知　$Q(s,a)$

Idea: Learn Q-values, not values!

⇒ Q-learning: $Q_{k+1}(s,a)$

Q-value iteration: $\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$

每一次 receive sample (s,a,s',r), 考虑到旧的 $Q(s,a)$:

$$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

更新 $Q(s,a)$: $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[\text{sample}]$

Amazingly: 即使 act suboptimally, ☆ policy converge to optimal one.

这被称为 off-policy learning.

上述介绍完 learning paradigm, 现介绍探索及利用:

How to explore? Simplest: random (ε-greedy)

即小 prob ε 会 act randomly, 大 prob ε 会 act on current policy

但有问题: 学好了之后, 依然会 act randomly! 一种解法:

lower ε over time.

另一种: Exploration Function:

考虑 Q-value estimate 为 u, $s' \to a'$ count of visit 为 n:

Optimistic Utility: $f(u,n) = u + k/n$

则 Q update: original $Q(s,a) \leftarrow \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$

后来: $Q(s,a) \leftarrow \alpha[R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))]$

↓ 这变向于: 给 under-explored states 加 bonus!

No.

Date

Approximate Q-learning: 原来Q learning有 Q(s,a) table.
但现在考虑 Feature-Based Representation 衡量 Q(s,a)
Linear form: $V(s) = \sum_i w_i f_i(s)$

$Q(s,a) = \sum_i w_i f_i(s,a)$, 则更新逻辑为:

transition = (s, a, r, s')

difference = $[r + \gamma \max_{a'} Q(s',a')] - Q(s,a)$

$Q(s,a) \leftarrow Q(s,a) + \alpha$ [different] Exact Q's

$w_i \leftarrow w_i + \alpha$ [difference] $f_i(s,a)$ Approximate Q's

↓ 调整 feature 权重

Regret 是衡量 your rewards 与最优 rewards 间的差距。

⚠ Minimizing regret going beyond learning to be optimal:
It requires optimally learning to be optimal!

⇔ 为了optimal policy, 不必要最小化 regret!!

No.

Date

:即均值：即使方值最小的点

Unsupervised ML

Clustering: Group together similar instances

'similar': One option: Euclidean Squared: $dist(x,y) = \sum_i (x_i - y_i)^2$

(也有其它option in specific domain)

① K-means Approach: K指分K类 ← Squared Euclidean

$\phi(\{x_i\}, \{a_i\}, \{C_k\}) = \sum_i dist(x_i, C_{a_i})$

Two stage 循环：更新 assignment ① → 更新 means（中心点）②

① fix means c, change assignment

② 在该 assignment 下改 means c

Phase 1: $a_i = \underset{k}{\operatorname{argmin}}\ dist(x_i, C_k)$ 每个点找最近 C_k

此步不会使 ϕ 函数上升

Phase 2: $C_k = \frac{1}{|\{i: a_i = k\}|} \sum_{i:a_i=k} x_i$ *. 亦不使 ϕ 上升

② 对于K-means，它 non-deterministic！选择的initials很重要！

且可能 converge to 局部最优

GMM: Gaussian Mixture Model Approach

Recall Gaussian: $P(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Given 观察到的 data:

$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$ $\quad$ $\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$

多元高斯分布定义：n维样本空间 χ 中随机向量 x，PDF为：

$$p(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

其中 Σ 是 $n \times n$ 协方差矩阵

⇒ 高斯分布完全由均值向量 μ 与 Σ 确定！

KOKUYO

我们记：$p(x|\mu,\Sigma)$

则 GMM 中，对于 k 类，则定义高斯混合分布：

$$p_M(x) = \sum_{i=1}^{k} \alpha_i \cdot p(x|\mu_i,\Sigma_i)$$

每一类算 μ_i, Σ_i，且 $\alpha_i > 0$ 为混合系数，$\sum_{i=1}^{k}\alpha_i = 1$

Inductive Bias：Sample 生成由 GMM 给出：且有 $\alpha_1,\cdots,\alpha_k$ 先验

则依 Bayes：$p_M(z_j=i|x_j) = \frac{P(z_j=i)\cdot p_M(x_j|z_j=i)}{=\alpha_i\,(\text{prior})\; p_M(x_j)}$

$$= \frac{\alpha_i \cdot p(x_j|\mu_i,\Sigma_i)}{\sum_{l=1}^{k}\alpha_l p(x_j|\mu_l,\Sigma_l)}$$

其中 $z_j \in \{1,\cdots,k\}$ 表示生成 x_j 的高斯

上在讨论：在 Inductive Bias 下，x_j 是由 $z_j=i$ 个 Gaussian 生成的概率。式子显示：$p_M(z_j=i|x_j)$ 给出了样本 x_j 由第 i 个 Gaussian 生成的后验概率，记为 γ_{ji}，$i \in \{1,\cdots,k\}$

根据此逻辑，每个样本 x_j 可依后验概率取 max 来判定 x_j 所属的簇标记：(λ_j)

$$\lambda_j = \underset{i\in\{1,\cdots,k\}}{\operatorname{argmax}} \gamma_{ji}$$

一言以蔽之：高斯混合聚类是采用高斯分布对原型进行刻画

簇划分则由原型对应后验概率确定

欲求解 $\{(\alpha_i,\mu_i,\Sigma_i)\,|\,1\le i\le k\}$，最大化似然：

$$LL(D) = \sum_{j=1}^{m} \ln\left(\sum_{i=1}^{k}\alpha_i\cdot p(x_j|\mu_i,\Sigma_i)\right)$$

（即 $\ln\left(\prod_{j=1}^{m} p_M(x_j)\right)$）

常用 EM 算法优化

$$\mu_i = \frac{\sum_{j=1}^{m}\gamma_{ji}x_j}{\sum_{j=1}^{m}\gamma_{ji}} \qquad \Sigma_i = \frac{\sum_{j=1}^{m}\gamma_{ji}(x_j-\mu_i)(x_j-\mu_i)^T}{\sum_{j=1}^{m}\gamma_{ji}} \qquad \alpha_i = \frac{1}{m}\sum_{j=1}^{m}\gamma_{ji}$$

No.

Date

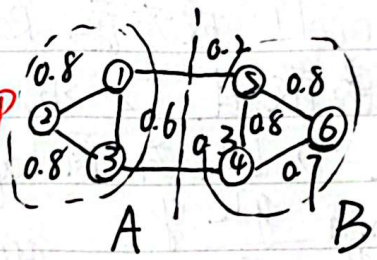
依式可推出EM流程：

(E) 根据当前参数计算每个样本属于每个高斯成分的后验 γ_{ji}，再依三个式子更新参数 $\{(\alpha_i, \mu_i, \Sigma_i) \mid 1 \le i \le k\}$ (M)

③. 谱聚类 Spectral Clustering

用顶点 V 代表数据点，点间用边 E 相连。E有weight W. 权重越大，说明两点间越 'similar'. 最后cluster可转化为 partitioning a similarity graph

则要 minimize weight of between-group connections



即找 groups with the minimal cut：

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

只看 inter-cluster connection 不够，还有 intra-cluster！

⇒也要 maximize the weights of connections within groups

$$\max (assoc(A, A) + assoc(B, B)), \quad assoc(A, A) = \sum_{i,j \in A} w_{ij}$$

综合上述：Criterion: Normalized Cuts

The connectivity between groups：

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

$$assoc(A, V) = \sum_{i \in A, j \in V} w_{ij}$$

$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)}$$

则式子上 $Ncut(A, B) = 2 - Nassoc(A, B)$

既然是个图了，那么就有 Adjacency Matrix (Symmetric)

定义属于哪一簇：$x \in [-1, 1]^n$，$x_i = \begin{cases} 1, & i \in A \\ -1, & i \in B \end{cases}$ $d_i = \sum_j w_{ij}$

记录哪个数据点属于哪个簇

则 $Ncut(A, B) = \frac{\sum_{x_i > 0, x_j < 0} -w_{ij} x_i x_j}{\sum_{x_i > 0} d_i} + \frac{\sum_{x_i < 0, x_j > 0} -w_{ij} x_i x_j}{\sum_{x_i < 0} d_i}$

$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$，簇A的度占总度比例

定义矩阵 $W \in \mathbb{R}^{n\times n}$ (adjacency)　$D \in \mathbb{R}^{n\times n}$ (度，$d_{ii} = \sum_j w_{ij}$，其余为0)

则 $4Ncut(A,B) = \frac{(\vec{1}+\vec{x})^T(D-W)(\vec{1}+\vec{x})}{k\vec{1}^T D\vec{1}} + \frac{(\vec{1}-\vec{x})^T(D-W)(\vec{1}-\vec{x})}{(1-k)\vec{1}^T D\vec{1}}$

$b = \frac{k}{1-k}$ $\frac{[(\vec{1}+\vec{x})-b(1-x)]^T(D-W)[(\vec{1}+\vec{x})-b(1-x)]}{b\vec{1}^T b\vec{1}}$

$$\min_x Ncut(\vec{x}) = \min_y \frac{y^T(D-W)y}{y^T D y} \quad s.t.\ \vec{y} \in [2-2b, -2b]^n,\ \vec{y}^T D\vec{1} = 0$$

NP-hard! 故考虑 Rayleigh Quotient 的放宽约束，简化问题

$\vec{y}$ 从离散值 $[2-2b, -2b]^n$ 变为 $\vec{y} \in \mathbb{R}^n$

定义 $L = D - W$，为 图 Laplacian Matrix

则：$\min_y \frac{y^T L y}{y^T D y}$，$y \in \mathbb{R}^n$，$y^T D\vec{1} = 0$

$\Leftrightarrow \max_x x^T A x$ s.t $x^T B x = 1$ $\xleftarrow{\text{Laplace}}$ $Ax = \kappa Bx$，$\kappa = -\lambda$ 广义特征值

i.e., $(D-W)y = \lambda D y$，令 $z = D^{1/2} y$，则

$D^{-1/2}(D-W)D^{-1/2} z = \lambda z$，求解特征值与其对应特征方程 之 ！

↓ 第二小的!!!