Search Problem

State Space [location (dots booleans)] / Successor function / Start & Goal state

World state: Agent pos × $2^{\text{food counts}}$ × Ghost pos$^{\#\text{ of ghosts}}$ × actions

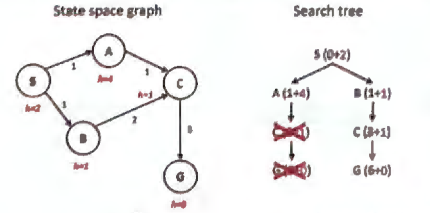
Tree Search

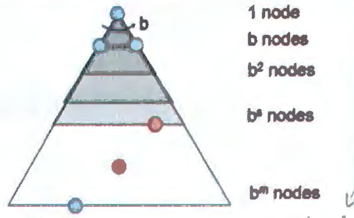
		Strategy	Fringe	is Complete	is Optimal	Time complexity	Space
Uninformed	DFS	a deepest node	LIFO stack	Yes | no cycle	No | leftmost	$O(b^m)$	$O(bm)$
	BFS	a shallowest node	FIFO queue	Yes	Yes | same costs	$O(b^s)$	$O(b^m)$
	UCS	a cheapest node (path cost / backward cost g(n); goal proximity / forward cost h(n))	priority queue	Yes	Yes	$O(b^{C^*/\epsilon})$	
Informed	Greedy	a node that is closest to a goal		badly guide DFS			
	A*	orders by $f(n) = g(n) + h(n)$	Admissible (optimistic): $0 \le h(n) \le h^*(n)$		Yes		

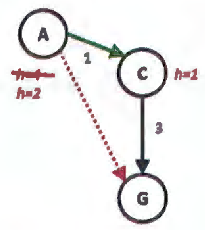
C^*: solution costs C^*, arcs cost at least ϵ

\# Heuristic
A function that estimates how close a state is to a goal

Graph Search never expand a node twice A* with consistent heuristic







- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal
    h(A) ≤ actual cost from A to G
  - Consistency: heuristic "arc" cost ≤ actual cost for each arc
    h(A) – h(C) ≤ cost(A to C)
  - Consistency implies admissibility
- Consequences of consistency:
  - The f value along a path never decreases, because:
    h(A) ≤ cost(A to C) + h(C)

Adversial Search

Resource Limit: ① Depth-limited ② Limiting branching factor

Evaluation function: $\text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots$

Minimax

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

α: MAX's best option on path to root
β: MIN's best option on path to root

Alpha-beta

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Expectimax
no pruning!

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```

Varible Elimination (VE) on Polytree BN = Sum-product message passing algorithm or belief propagation algorithm. (from leaf to root)

General: group nodes to have a polytree (junction/join) tree, Intractable (不易实现) with large cliques (large tree width)
Approximate. tractable but not guaranteed to work well.

Constraint Satisfication Problems

d: Domain Size

Original $O(d^n)$
Suppose a graph of n varibles can be broken into subproblems of only c varibles → $O[(n/c)(d^c)]$

Variables / Domains / Constraints

Basic solution: Backtracking ① One variable at a time ② check constraints as you go
= DFS + variable-ordering + Fail-on-violation

Filtering: Forward Checking: Cross off values that violate a constraint when added to assignment; whenever any variables has no value left, backtrack.

Arc Consistency: An arc X→Y is consistent iff for every x in the tail there is some y in the head could be assigned without violating a constraint.
Run time: $O(n^2d^3)$, can be reduced to $O(n^2d^2)$ to check

* Delete from the tail: If Y loses a value, arc X→Y needs to be re-checked.

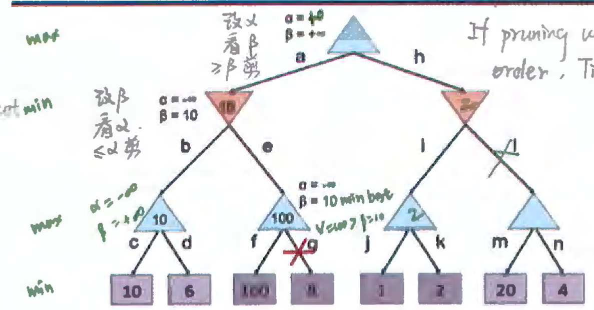
Ordering: Minimum Remaing Value: the variable with the fewest legal left values in domain
Least Constraining Value: the one rules out the fewest values in the remaing vars.

Structure: Tree structure: Choose a root variable, order varible that parents precede childs
$O(nd^2)$
Remove backward: n→2, remove inconsistent + Assign forward: 1→n assign x

Cutset structure: a set of variables s.t. the remaining constraint is a tree.
c: cutset size
$O((d^c)(n-c)d^2)$
NP hard to find smallest cutset

Alpha-Beta Example 2

Time $O(b^m)$
Space $O(bm)$



If pruning with perfect order, Time: $O(b^{\frac{m}{2}})$

Propositional Logic

语法 syntax & 语义 semantic

{ conjunctive: ∧
{ disjunctive: ∨

Valid / satisfiable / un-

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of ∧
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of ∨
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of ∧
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of ∨
$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of ∧ over ∨
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of ∨ over ∧

{ Entailment: $\alpha \vDash \beta$ model(α) ⊆ model(β)
{ Proof: $\alpha \vdash \beta$ { model checking ×
                              { rule of inference

{ Sound: $\alpha \vdash \beta \rightarrow \alpha \vDash \beta$
{ complete: $\alpha \vDash \beta \rightarrow \alpha \vdash \beta$

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ Convert to CNF

1. Eliminate ⇔, replacing α ⇔ β with (α ⇒ β)∧(β ⇒ α).
$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate ⇒, replacing α ⇒ β with ¬α∨ β.
$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move ¬ inwards using de Morgan's rules and double-negation:
$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law (∧ over ∨) and flatten:
$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

sound & complete

Prove $KB \vDash \alpha$ → show $KB \wedge \neg\alpha$ is unsatisfiable ① to_cnf$(KB \wedge \neg\alpha)$ ② resolve to empty clause

Horn Logic 1° $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$, 2° $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$

⇒ Forward chain: Add new clauses into KB until Q is added ⇒ data-driving

Backward chain: goal-driving * avoid loop (check if new subgoal is on the goal stack)
\# avoid repeated work (has already be proved or fail)

Resolution:
$$\frac{A \vee X, \quad B \vee \neg X}{A \vee B}$$

Modus Popens $\dfrac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$

only sound and complete for Horn Logic
run in linear time

Complete search algorithm can be used to produce a complete inference algorithm

Constraint graph: MN with 0/1 potentials
PL: BN/MN with deterministic CPTs / potentials
CRF (Conditional Random Field): everything conditioned on an input in MN.

First-Order Logic

Syntax Logical symbols: Connectives ($\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$) Quantifiers ($\forall, \exists$) Variable ($x, y$) Equality ($=$)

Non-logical symbols: Constants (King, 2) Predicates (Brother, >) Function (Sqrt, LeftLegOf)

Term: constant / variable / function (term1, term2, ...)

Atomatic sentence: predicate (term1, term2, ...) / term1 = term2.

Complex sentence: automatic sentences using connectives ($\neg$ S, S1 $\Rightarrow$ S2, ...) $>(1,2) \vee \leq (1,2)$

Sibling(A,B) ⇒ ...

Semantic model = objects + relations + interpretation (specify symbols)

$\forall x$ At(x, STU) ⇒ Smart(x) $\exists x$ At(x, STU) $\wedge$ Smart(x)

$\forall x \forall y$ same as $\forall y \forall x$, $\exists x \forall y$ not $\forall y \exists x$.
$\exists x \exists y$ same as $\exists y \exists x$

$\forall x P(x) \equiv \neg \exists x \neg P(x)$
$\exists x P(x) \equiv \neg \forall x \neg P(x)$

entailment of FOL is semi-decidable (can't say no to every non-entailed sentence).

In FOL, every var must be bound. $\forall x P(x,y)$ is not valid.

Inference. I) Propositionalization (UI / EI) (命题化, 即对 UI: $\forall x$ 的替换成所有常量; EI: $\exists x$ 的替换成 C_1: Skolem constant)

II) Unification substitude, MGU = {y/John, x/z}

III) FC/BC (Horn Logic) ← universally quantified.

FC: sound & complete; infinite loop when α is not entailed
BC: avoid infinite loop

IV) Resolution

1. Eliminate biconditionals and implications
∀x [¬∀y ¬*Animal(y)* ∨ *Loves(x,y)*] ∨ [∃y *Loves(y,x)*]

2. Move ¬ inwards: ¬∀x p ≡ ∃x ¬p, ¬∃x p ≡ ∀x ¬p
∀x [∃y ¬(¬Animal(y) ∨ Loves(x,y))] ∨ [∃y Loves(y,x)]
∀x [∃y ¬¬Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]
∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃y Loves(y,x)]

3. Standardize variables: each quantifier should use different variable
∀x [∃y Animal(y) ∧ ¬Loves(x,y)] ∨ [∃z Loves(z,x)]

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
∀x [Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)

5. Drop universal quantifiers:
[Animal(F(x)) ∧ ¬Loves(x,F(x))] ∨ Loves(G(x),x)

6. Distribute ∨ over ∧:
[Animal(F(x)) ∨ Loves(G(x),x)] ∧ [¬Loves(x,F(x)) ∨ Loves(G(x),x)]

MN: $P(x) = \frac{1}{Z} \prod_c \psi_c(x_c)$, $Z = \sum_x \prod_c \psi_c(x_c)$

BN and MN encode the same distribution
The set of distributions whose conditional independence can be exactly represented by a directed / undirected graph.

D   U   P

Exact Inference

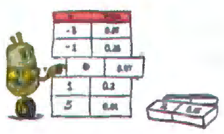
Enumeration exponential   Worst Time $O(d^n)$ Space $O(d^n)$ n vars, d domain size

- General case:
  - Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
  - Query variable: Q
  - Hidden variables: $H_1 \dots H_r$

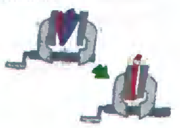
  $X_1, X_2, \dots X_n$ All variables

- We want: $P(Q|e_1 \dots e_k)$

- Step 1: Select the entries consistent with the evidence



- Step 2: Sum out H to get joint of Query and evidence



$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

- Step 3: Normalize

$$\times \frac{1}{Z}$$

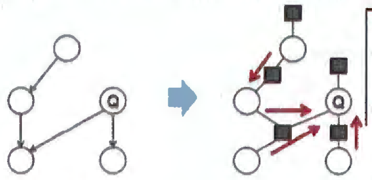
$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Variable Elimination   exponential often better   Choose one to join and eliminate

$\sum_e \sum_r P(L|t) P(r) P(t|r) \Rightarrow \sum_t P(L|t) \sum_r P(r) P(t|r)$ NP hard.

Polytree: Directed graph without undirected cycles ⇒ linear complexity in size of BN



Bayesian Network   Factor Graph

- Input: evidence $e_1, \dots, e_k$
- For i=1, 2, ..., n
  - Sample X_i from P(X_i | parents(X_i))
  - If x_i not consistent with evidence
    - Reject: Return, and no sample is generated in this cycle
- Return ($x_1, x_2, \dots, x_n$)

reject

Sampling memory: $O(n)$

- For i=1, 2, ..., n (in topological order)
  - Sample X_i from P(X_i | parents(X_i))
- Return ($x_1, x_2, \dots, x_n$)

prior

- Input: evidence $e_1, \dots, e_k$
- w = 1.0
- for i=1, 2, ..., n
  - if X_i is an evidence variable
    - x_i = observed value, for X_i
    - Set w = w * P(x_i | Parents(X_i))
  - else
    - Sample x_i from P(X_i | Parents(X_i))
- return ($x_1, x_2, \dots, x_n$), w

likelihood

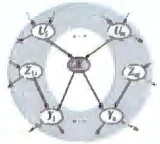
All consistent.

BNs need not actually be causal (因果关系的)
Topology (arrows) really encodes conditional independence.

Every varible is conditionally independent of its non-descendants given its parents.

n vars,
d max (Domain Size),
k max (# of parents).
Full Joint Distribution: $O(d^n)$
Bayes Net Size $O(n \cdot d^{k+1})$.

Bayesian Network

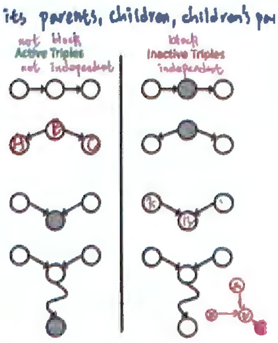


Syntax DAG + CPTs
Semantic

Markov blanket: indepent of all other vars given its parents, children, children's parents

- Question: X, Y, Z are non-intersecting subsets of nodes. Are X and Y conditionally independent given Z?
- A triple is active in the following three cases
  - Causal chain A → B → C where B is unobserved (either direction)
  - Common cause A ← B → C where B is unobserved
  - Common effect (aka v-structure) A → B ← C where B or one of its descendents is observed
- A path is active if each triple along the path is active
- A path is blocked if it contains a single inactive triple
- If all paths from X to Y are blocked, then X is said to be "d-separated" from Y by Z
- If d-separated, then X and Y are conditionally independent given Z

undirected



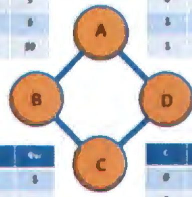
Markov Networks   Undirected graph + potentials

A	B	C	D	[illegible]
0	0	0	0	250
0	0	0	1	37000
0	0	1	0	50000
0	0	1	1	825000
0	1	0	0	1125
0	1	0	1	168750
0	1	1	0	90000
0	1	1	1	825000
1	0	0	0	250
1	0	0	1	375
1	0	1	0	50000
1	0	1	1	6250
1	1	0	0	112000
1	1	0	1	168750
1	1	1	0	5000000
1	1	1	1	825000

Z = 7520750

A	B	ϕ_{AB}
0	0	30
0	1	5
1	0	5
1	1	10

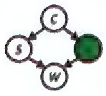
A	D	ϕ_{AD}
0	0	5
0	1	10
1	0	50
1	1	5



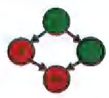
B	C	ϕ_{BC}
0	0	1
0	1	5
1	0	40
1	1	10

C	D	ϕ_{CD}
0	0	1
0	1	10
1	0	40
1	1	30

- Step 1: Fix evidence
  - *R* = true

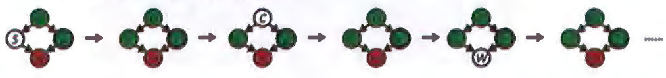


- Step 2: Initialize other variables
  - Randomly

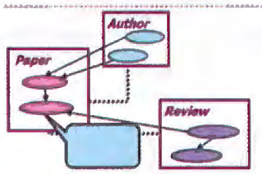


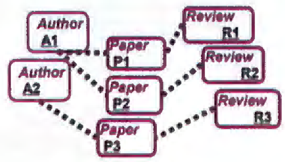
- Step 3: Repeat
  - Choose an arbitrary non-evidence variable *X*
  - Resample *X* from *P*(*X* | *markov_blanket*(*X*))

Gibbs



Sample *S* ~ *P*(*S* | *c*, *r*, ¬*w*)   Sample *C* ~ *P*(*C* | *s*, *r*)   Sample *W* ~ *P*(*W* | *s*, *r*)





PRM (S, Θ) + relational skeleton (σ) =

probability distribution over instantiations of attributes I:

$$P(I|\sigma, S, \Theta) = \prod_{x \in \sigma} \prod_{x.A} P(x.A \,|\, parents_{S,\sigma}(x.A))$$

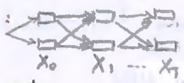
Objects   Attributes

MLN. $P(x) = \frac{1}{Z} \exp(\sum_i w_i n_i(x))$
w_i: Weight of formula i, $n_i(x)$: # of true groundings of formula i in x.

Probabilistic Relational Model (PRM)
Logic Language: Frame (sub class of FOL)
Probabilistic Language: Bayes Nets
Bayes net template for object classes
Object's attr. can depend on attr. of related objs.

Markov Logic (Network, MLN)
a set of pairs (F, W), F: a formula in FOL, w: a real number
Logic Language: First-order logic
Probabilistic Language: Markov networks
Syntax: First-order formulas with weights.
Semantics: Templates for Markov net cliques

I. Probabilistic Temporal Model.

1. **Stationarity** assumption: same transition probabilities at all time steps.
   Infinitely many variables.
   Markov assumption: past and future independent given present. **first-order**. k-th order: dependent k earlier steps
2. Forward Algorithm: $P(X_t) = \sum_{x_{t-1}} P(X_{t-1}=x_{t-1}) P(X_t | X_{t-1}=x_{t-1})$
   For most chains, $P_\infty \perp P_0$.
   Stationary distribution. $P_\infty(X) = \sum_x P(X|x) P_\infty(x)$.
   Application: Web Link Analysis, PageRank, Gibbs Sampling.
3. HMM. Initial distribution $P(X_0)$; Transition Model $P(X_t | X_{t-1})$,
   Emission model $P(E_t | X_t)$.
   evidence is independent of everything else give current state.
4. **Filtering**. $P(X_t | e_{1:t})$. (belief state) recursive
   $P(X_{t+1} | e_{1:t+1}) = \underbrace{\alpha}_{\text{normalize}} \underbrace{P(e_{t+1} | X_{t+1})}_{\text{update}} \underbrace{\sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t)}_{\text{Predict}}$
   $\alpha = \frac{1}{P(e_{t+1} | e_{1:t})}$
   $f_{1:t+1} = \text{Forward}(f_{1:t}, e_{t+1})$. start with $f_{1:0} = P(X_0)$
   Cost $O(|X|^2)$. $|X|$. # of states.
   State trellis
   

   each arc: $P(x_t | x_{t-1}) P(e_t | x_t)$
   init arc: $P(X_0)$
   product of weight on a path: state sequence probability
   Forward algorithm,
   sum over all possible paths: $P(X_{t+1} | e_{1:t+1}) = \sum_{x_{1:t}} P(x_{1:t+1} | e_{1:t+1})$
   DP. $f_{1:t+1} = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) f_{1:t}(x_t)$
   keep track of the total probability of all path to it.
5. Most likely explanation. $\arg\max_{x_{1:t}} P(x_{1:t} | e_{1:t})$
   Viterbi Algorithm:
   keep track of the maximum probability of any path to it.
   $m_{1:t+1} = \text{Viterbi}(m_{1:t}, e_{t+1})$
   $= P(e_{t+1} | X_{t+1}) \max_{x_t} P(X_{t+1} | x_t) m_{1:t}(x_t)$.
   Time: $O(|X|^2 T)$, Space: $O(|X| T)$.
6. Dynamic Bayes Net (DBN).
   Every HMM is a DBN, Every discrete DBN can be represented HMM.
   exponentially fewer parameters.
   Exact Infer: unroll as we go, eliminate all variables from prev time step to find $P(X_T | e_{1:T})$.
7. Particle Filtering.
   N particles. Generally, $N \ll |X|$. Many x: $P(x) = 0$
   propagate forward: $x_{t+1} \sim P(X_{t+1} | x_t)$
   observe (weight): $w = P(e_t | x_t)$
   resample.

II. Markov Decision Process. (Offline)

1. non-deterministic search problems.
   discounting helps to converge.
2. The bellman equation.
   $V^*(s) = \max_a Q^*(s,a)$
   $Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$.
   $V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$.
3. Time-Limited values.
   $V_k(s)$: optimal value of s if game ends in k more steps.
   (DP) Value Iteration: $V_0(s) = 0$
   $V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$.
   until convergence. each iteration $O(S^2 A)$. Total $O(S^2 AH)$
   will converge to unique optimal values. Compared to Expectimax: $O((AS)^H)$
   Policy extraction:
   $\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
   $= \arg\max_a Q^*(s,a)$.
4. Q-value Iteration: $Q_0(s,a) = 0$
   $Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$.
3. Policy Iteration (DP)
   Optimal. converge much faster under some conditions.
   ① Policy evaluation
   a. Iterative updates. $V_0^\pi(s) = 0$
   $V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^\pi(s')]$.
   $O(S^2)$ per iteration.
   b. Solve Linear system. $O(S^3)$ per step
   $V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$.
   ② Policy improvement.
   $\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$

III. Reinforcement Learning (Online)

1. Model-Based Learning.
   Learn empirical MDP Model.
   Count s' for each s,a $\xrightarrow{\text{normalize}} \hat{T}(s,a,s')$
   Solve the learnt MDP.
2. Model-Free Learning
   2-1. Passive Reinforcement Learning. (Direct Evaluation).
   act according to π, average the samples.
   easy to understand | waste info about state
   don't need T, R | connection. state learnt
   correct a.v.g. value using sample transitions | separately, takes a long time
   2-2. Temporal Difference Learning. (Evaluates a policy π)
   sample $= R(s,\pi(s),s') + \gamma V^\pi(s')$
   update: $V^\pi(s) \leftarrow (1-\alpha) V^\pi(s) + \alpha \cdot \text{sample}$.
   $V^\pi(s) \leftarrow V^\pi(s) + \alpha(\text{sample} - V^\pi(s))$.
   Decreasing α can give converging averages.
   2-3 Q-Learning (Active Reinforcement Learning) (Full RL)
   sample $= R(s,a,s') + \gamma \max_{a'} Q(s',a')$
   $Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \cdot \text{sample}$.
   converges to optimal policy even if acts suboptimally.
   * off-policy learning
   have to explore enough, eventually $\alpha \downarrow$ small enough and not too quickly.
3. Explore.
   ε-greedy: randomly / exploration function.
   n: visit count.
   $f(u,n) = u + k/n$.
   $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$
4. Regret.
   a measure of total mistake cost.
   minimizing regret > learning to be optimal.
5. Approximate Q-learning
   $Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \dots + w_n f_n(s,a)$
   difference $= [r + \gamma \max_{a'} Q(s',a')] - Q(s,a)$
   $w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$ $\left(\frac{\partial Q}{\partial w_i}(s,a)\right)$
6. Policy Search.
   Real priority, get the ordering of Q-values right (action prediction)
   start with an OK solution,
   change weight up and down to see if it's better.
   problem: may need to run many episodes.
   better method: exploit lookahead structure, sample wisely, change multiple parameters.

IV. Supervised learning.

1. Classification

learning f with discrete output value.

1-1 Model-Based Classification.

Naïve Bayes. Assume all features are independent effects of label

# of parameters is linear in n

tree structure. linear inference time.

$P(Y|w_1, w_2, \cdots, w_n) \propto P(Y)\prod_i P(w_i|Y)$

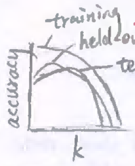
1-2 Training

MLE. $P_{ML}(x) = \frac{count(x)}{total\ samples}$

1-3 Generalization and Overfitting.

why: too few training samples / noisy training data / too many attributes / too expressive. (store all spams)

Smooth.



$P_{LAP,k}(x) = \frac{c(x)+k}{N+k|X|}$. k: strength of piror

$P_{LAP,k}(x|y) = \frac{c(x,y)+k}{c(y)+k|X|}$

poor performance when $|X|$ or $|Y|$ is large.

Linear interpolation ($P(X|Y)$ isn't too diff from $P(X)$)

$P_{LIN}(X|Y) = \alpha\hat{P}(x|y) + (1-\alpha)\hat{P}(x)$.

1-4 Perceptron

activation$_w(x) = w \cdot f(x)$ $>0 \to 1$, $<0 \to -1$

learn: $w = w + y^* \cdot f$ $y^* = \pm 1$

If training set separable, perceptron converges.

2. Regression

learning f with real-valued output value.

$L(w) = \sum_i (y_i - w^T x_i)^2$, $w^* = (X^TX)^{-1}X^Ty$

Regularization: alleviate overfitting.

LASSO: $L(w) = \sum_i (y_i - w^T x_i)^2 + \lambda\sum_k |w_k|$

Ridge Regression: $L(w) = \sum_i (y_i - w^T x_i)^2 + \lambda\sum_k w_k^2$

"Ockham's razor": prefer the simplest hypothesis consistent with the data.

V. Unsupervised Machine Learning.

1. Clustering what does similar mean?

one option: $dist(x,y) = \sum_i (x_i - y_i)^2$ is small.

2. K-means: $\varphi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i dist(x_i, c_{a_i})$ point assign mean (center)

pick k means

assign data to closest mean $a_i = \arg\min_k dist(x_i, c_k)$

assign mean to average assigned points $c_k = \frac{1}{|\{i: a_i = k\}|}\sum_{i: a_i = k} x_i$

stop when no points' assignments change.

Problem: Local optima; Equally Sized Clusters; Circular Clusters.

3. Expectation-Maximization (EM).

pick k random cluster models (Gaussian)

assign data proportionately to different models

revise each cluster model on its assigned points

stop when no changes.

EM degrades to K-means if

1. All Gaussians are spherical and have identical weights and covariances μ is only parameter
2. Label distribution in E step are point-estimations $\sigma^2 \to 0$

$$l(\theta; D) \geq F(\theta, Q) = \sum_{j=1}^{m}\sum_z Q(z|x_j)\log\frac{P(z, x_j|\theta)}{Q(z|x_j)}$$

EM for HMM.

E Step. compute the distribution of hidden states given each training instance.

Infeasible to enumerate. But can compute expected counts of transitions and emissions using the forward and backward algorithms.

M Step. Update the parameters to maximize expected log likelihood based on distributions over hidden states. Closed-form solution: simply normalize the expected counts of transitions and emissions

Known as Baum-Welch algorithm.

VI. Natural Language Parsing.

1. CNF. $A \to BC$, $A \to w$

conversion to CNF:

$A \to B, B \to C \Rightarrow A \to C$

$S \to A\ B\ C \Rightarrow S \to X\ C,\ X \to A\ B$

2. Regular Grammar

$A \to a\ B$ or $A \to a$; Probabilistic RG = HMM.

3. Dependency Grammar. siblings. grandparent

The tree score is the sum of arc scores.

4. Parsing (DP)

Probabilistic Parsing

$P_{A,i,j} = \max_{B,C,k} P(A \to BC) \times P_{B,i,k} \times P_{C,k,j}$.

Run CYK on a HMM

CYK on HMMs ≈ Viterbi

5. Learning Grammas

Supervise Methods. (treebank)

MLE × poor performance

增加额外标记.

Unsupervise Methods

EM.

6. Context-free Grammar

① a set Σ of terminals (words)

② a set N of nonterminals (phrases)

③ a start symbol $S \in N$

④ a set R of production rules specifies how a nonterminal can produce a string of terminals and/or nonterminals.

7. CFG focus on constituents.