CS 540

1.1 2.1 2.2 2.3

deterministic

stochastic

episodic

sequential

static

semi-dynamic

discrete

multi-agent

<u>why AI is hard?</u>-

- Large data set (high res. images)
- enviroment
- Many solns
- Inaccurate as a data ("noise")
- Domain ~~[illegible]~~ knowledge

<u>Uninformed SEARCH</u> 3.1-3.4

HW1

* 5 components

1. States → initial states, Goal states
2. Action set
3. Transition model
4. goal test
5. PATH COST

FL→

←F

FC→

←FS

FD→

←F FS→

Basic Search Task
What knowledge ~~he~~ does the Agent need?

WATER JUG PROBLEMS.

(0,0) // START STATE

(4,0) (0,+3)

(3,0)

(3,3)

(4,2)

(0,2)

(2,0)

node

(0,0)

$\xrightarrow{\text{edge}}$ (0,3)

- Expanding

action ((0,0))

The generated, but not yet expended states define the Frontier (aka Open or Fringe) list.

Directed graph to formalize the search

G = (V, E)

Frontier

TREE SEARCH Alg:

Frontier = {S}, where S is the start state.

Uninformed Search on Trees

BFS - Queue used for the Frontier list

DFS - stack

O—$\boxed{\text{ABC}}$

DFS
UCS — Priority Queue

Evaluating Search Strategies
Completeness
Optimality / Admissibility
Time Complexity
SPACE Complexity

Breath is finite

Informed SEARCH
domain knowledge to guide selection of the best path

All domain knowledge used in the search is encoded in the heuristic function, h.
weak method.
Best First Search
Greedy Best First Serch
Beam Search

$h(n) > h^*(n)$

☆ $h^*(n) \leq 2$, $= 2$

$h^*(n)$

local search & informed search
solution ✓ we don't care about path.

① Random start
② Stochatic HC
→ Simulated annealing

$$P = e^{(f(\text{newnode}) - f(\text{currentnode}))/T}$$

$$f(S) > f(t)$$

Boltzman's equation

Genetic Algorithms

1 0 1 1

0 1 1 1

2 operations { crossover
mutute

Selection: Finding the Fittest

Game PLAYING

Types of Games

~~goal~~ Utility Function

Minimax Principle

Assume both players

The computer assumes after it moves

MINMax ALGORITHM

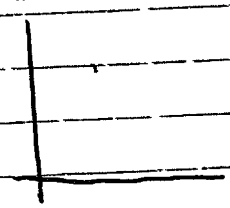
STATIC Board Evaluator

① KNN

Store all straining example in memory

- Take in last example

② SVM



Decision TREES

OFFICE HOUR: T/R 2-3:00 pm

Exam

- Multiple choice matching 40

  T/F

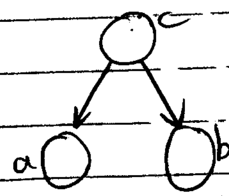
- Problem solving : 60

  Open-ended answer

X

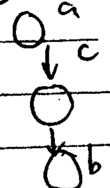
Neural Network

					W_1	W_P	W_0
1	1	0	< 0				
1	0	1	> 0		1	0	
0	1	1	> 0		0	1	
0	0	0	< 0				

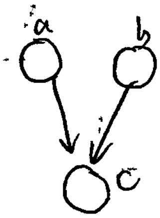
① 

a & b are cond. indepand given C

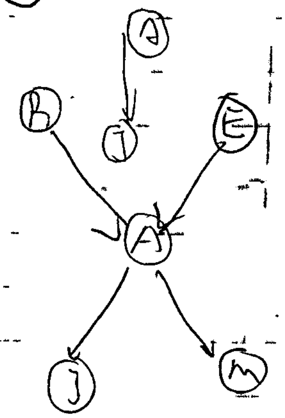
a & b are cond indep given C

② 

(3)



a $\perp\!\!\!\perp$ b cond dependent given c



a. state space — a directed graph   cost funct

actions

transition model

goal test   determine if it is a goal node.

path cost

b. BFS   Queue

DFS   Stack

UCS   Priority Queue.

IDS   ~~DFS~~ depth DFS to depth. { has advantage of BFS: complete / optimal; DFS: limited space

	BFS	DFS	UCS	IDS
TIME complexity	$O(b^d)$	$O(b^d)$	$O(b^{c^*/\epsilon})$	$O(b^d)$
Space complexity	$O(b^d)$	$O(bd)$	$O(b^{c^*/\epsilon})$	$O(bd)$
Complete	✓	✗	✓	✓
Optimal	✓	✗	✓	✓

2) Informed Search. uses domain-specific information in some way

a. heuristic: is computable from the current state description

it estimates { the "goodness" of node n / how close node n is to goal

b. Greedy Best-First — not optimal

A search

A* search.

$f(n)=g(n)+h(n)$ → the estimate cost of minimal cost path from node n to a goal state

$g(n)$: cost from start to current node n. $h(n)$: cost from node n to goal

never over-estimated $h(n) \le h^*(n)$ (actual) ⇒ h is a admissible heuristic function

c. TIME complexity

space

complete / optimal / Admissible

d.

e. admissible / consistent ⇒   A heur h is consistent.   Consistency is a stronger condition than admissibility

$h(n) \le c(n, n') + h(n')$

$h(n)$: for every node n, the estimated cost of reaching the goal from n

$c(n, n')$: step cost of getting to n'

$h(n')$: estimated cost of reaching the goal from n'

~~f. local searching~~

~~hill climbing~~

~~g. simulated annealing~~

$h(n)=h^*(n)$ ⇒ no necessary work is performed

$h(n)=0$ same as UCS

h is close to h^* ⇒ the fewer extra nodes that will be expanded

$h_1(n) \le h_2(n) \le h^*(n)$ A_2^* is better informed

f. Local searching: every node is a solution

action go from one solution to another

can stop at any time and have a valid solution

goal: to find a better/best solution

evaluation function

hill-climbing: Pick initial s

Pick t in neigh(s) with the largest f(t)

f(t) ≤ f(s) then stop and returns s   s=t

⇒ stop at a local maximum

stochastic restart

Pick initial state, s
-- Randomly pick state t from neighbors of s
if f(t) better than f(s)
then s=t else with small probability s=t
$P = e^{\Delta E/T}$ can performed multiple backward steps in a row to espace a local max
different from local serch!

h. genetic algorithms
proportional selection
fitness = sum Fitness(individual) / Sum Fitness for all individuals

3) a.

	zero-sum Deterministic	Discrete Finite: Stochastic
Fully Observable (perfect info)	checkers chess, Go Othello	Backgammon Monopoly
Partially observed (imperfect info)	stratego Battleship	Bridge, Poker Scrabble

b. utility function, to map each terminal state of the board. to a score indicating the value of that outcome to the computer.

Greedy search: Expand the search tree to the terminal states on each branch.
Evaluate the utility of each terminal board configuration
make the initial move that results in the board config with the maximum value.

c. Minimax complexity. Space: DFS $O(bd)$ time BFS $O(b^d)$

e: because utility function is defined only for terminal states
static Evaluation function use heuristics to estimate the value of non-terminal states
is used to estimate how good the current board configuration is for the computer
should agree with utility function when calculated at terminal node.

f. g. Pruning can be used to ignore some branches.
Max value $\alpha \to$ come from children
$\beta \to$ come from parents
min value $\alpha \to$ parents
$\beta \to$ children
$O(b^{d/2})$

h. how to deal with limited time
IDS iterative deepening search

Horizon Effect: the computer has a limited horizon, it cannot see that this significant event could happen
sol: quiescence search
secondary search

4. Unsupervised machine learning
a. solve classification problems / learn models of data (data fitting) / understand and improve efficiency of human learning / discover new things or structures that are unknown to humans ("data mining")
b. Inductive learning: generalize from a given set of (training) examples so that accurate predictions can be made about future examples. learn unknown function $f(x)=y$
only memorizes training examples
completely unbiased inductive ALGO
c. $x \to$ represent a specific object. $y \to$ lable.
h (hypothesis) function is learned that approximates f
difference between unsupervised/supervised. given a set of $(\vec{x}, y)$ pairs and only the $\vec{x}$'s are given
f. clustering: group training sample into clusters such that examples in the same cluster are similar, and examples in different clustering are different.
g h i. HAC k-Means mean shift clustering: j. distortion: determine how good our clusters are.

5) Supervised Machine Learning
b) Concept learning: Determine from a given set of examples if a given example is or is not an instance of the concept/class/category. If it is $\to$ positive / negative
c KNN d. Inductive bias. $\Rightarrow$ learning can be viewed as searching the Hypothesis Space H of possible h functions. Inductive Bias is used when one h is chosen over another / is needed to generalize beyond the specific training examples beyond the specific training examples
e. SVMs.
slack variables: $\min_{w,b,\varepsilon} 1/2\, W^T W + C \sum_i \varepsilon_i$
trade off parameter
slack variables.
kernels.
Map data to high dimensional space

1) Decision trees a. feature vector usually discreted ; output discontinuous classification ; regression

b. In supervised learning { Training set: n pairs of (example, label)
A predictor $f: x \to y$
Hypothesis space: space of predictors
Find the "best function in the hypothesis space that generalizes well

c. Performance measure: MSE for regression
c. Accuracy $acc = \frac{1}{m}\sum_{i=n+1}^{n+m} 1_{\hat{y}_i = y_i}$

d. ID3 buildtree (examples, questions, defaults)
IF empty(examples) then return the attribute of parent node
IF (examples have same label y) then return(y)
IF empty(questions) then return (majority vote in examples)
q = best_question(examples, questions) { create and return an internal node with n children
let there be n answers to q | The ith child is built by calling
buildtree {examples | q = ith answer}, question \ {q}, default}

i. info gain $I(Y:X) = H(Y) - H(Y|X)$ $H(Y|X=v) = \sum_{i=1}^{k} -Pr(Y=y_i|X=v)\log_2 Pr(Y=y_i|X=v)$ $= \sum_{v \text{ values of } X} Pr(X=v)H(Y|X=v)$

e. $H(Y) = \sum_{i=1}^{k} -Pr(Y=y_i)\log_2 Pr(Y=y_i) = \sum_{i=1}^{k} -p_i \log_2 p_i$ ii. see purity node   iii root node is also internal node

j. overfitting: why: too many features
how to overcome: pruning: 1. Randomly split data into train and tune 2. build a full tree using only TRAIN
3. Prune the tree down on the TUNE set

ALGO: Prune (Tree, T, TUNE set)
1. Compute T's accuracy on TUNE, call it A(T)
2. For every internal node N in T
a) New tree T_N = copy of T, but prune (delete) the subtree under N
b) N becomes a leaf node in T_N. The label is the majority vote of TRAIN examples reaching N
c) $A(T_N)$ = T_N's accuracy on TUNE
3. Let T^* be the tree (among the T_N's and T) with the largest A(). set $T \leftarrow T^*$ /* prune */

h. real-valued features: use threshold
missing data: { replace with most likely value
use unknown as a value
follow arcs for all values and weight each by the frequency of the examples along the arc

i) Evaluation of Performance
Cross-Validation: k-fold cross-validation estimate of performance = mean accuracy = $(PA_1 + PA_2 + \dots PA_k)/K$
leave-one-out Cross Validation

j ensembles: Aggregation of predictions of multiple classifiers with the goal of improving accuracy by reducing the variance of an estimated prediction function

boosting: sequential production of classifiers, each is dependent on the previous one. make examples misclassified by current classifier more important

k. bagging: bootstrap samples ⇒ Construct a classifier ⇒ calculate error ⇒ Repeat
l. random forests: ① bagging ② Randomized choose attributes. X overfitting

Conditional independence in Bayes Nets
• A node is conditionally independent of its non-descendants, given its parents.
• A node is conditionally independent of all other nodes, given its "Markov blanket" i.e. its parents, children, and children's parents)

logical Equivalence
$(\alpha\wedge\beta) \equiv (\beta\wedge\alpha)$ | $\alpha\vee\beta \equiv \beta\vee\alpha$ | $(\alpha\wedge\beta)\wedge\gamma \equiv \alpha\wedge(\beta\wedge\gamma)$ | $\alpha\vee\beta\vee\gamma \equiv \alpha\vee(\beta\vee\gamma)$
$\neg(\neg\alpha) \equiv \alpha$ | $\alpha\Rightarrow\beta \equiv \neg\beta\Rightarrow\neg\alpha$ | $\alpha\Rightarrow\beta \equiv \neg\alpha\vee\beta$ | $\alpha\Leftrightarrow\beta \equiv ((\alpha\Rightarrow\beta)\wedge(\beta\Rightarrow\alpha))$
$\neg(\alpha\wedge\beta) \equiv \neg\alpha\vee\neg\beta$ | $\neg(\alpha\vee\beta) \equiv \neg\alpha\wedge\neg\beta$
$(\alpha\wedge(\beta\vee\gamma)) \equiv ((\alpha\wedge\beta)\vee(\alpha\wedge\gamma))$
$(\alpha\vee(\beta\wedge\gamma)) \equiv ((\alpha\vee\beta)\wedge(\alpha\vee\gamma))$

2) Neural Networks
i. linear perceptron $a = w_0 \cdot x_0 + w_1 \cdot x_1 + \dots + w_D \cdot x_D$ b. biased term.   c. linear perceptron unique global minimum $E(w) = 1/2\sum_{i=1}^{n}(a_i - y_i)^2$ $\frac{\partial E}{\partial w_d} = \sum_{i=1}^{n}(a_i - y_i)x_{id}$ $w_d \leftarrow w_d - \alpha\sum_{i=1}^{n}(a_i - y_i)x_{id}$
d. step function $g(h) = 0$, if $h<0$; $g(h) = 1$ if $h \geq 0$
e. AND: $w_1 = w_2 = 1$, $w_0 = -1.5$ OR: $w_1 = w_2 = 1$, $w_0 = -0.5$ NOT: $w_1 = -1$, $w_0 = 0.5$
f. step function is discontinuous, cannot use gradient descent
sigmoid function $g(h) = 1/(1+\exp(-h))$ $g'(h) = g(h)(1-g(h))$
$E(w) = 1/2\sum_{i=1}^{n}(a_i - y_i)^2$ $\partial E/\partial w_d = \sum_{i=1}^{n}(a_i - y_i)a_i(1-a_i)x_{id}$ $w_d \leftarrow w_d - \alpha\sum_{i=1}^{n}(a_i - y_i)a_i(1-a_i)x_{id}$
Even with a non-linear sigmoid function, the decision boundary a perceptron can produce is still linear

h. Calculate output h1 out(h1) = sigmoid(h1) | out = w · out(h) = sigmoid(...)
② calculate delta terms
① outer layers: $E_{total} = \frac{1}{2}(out - y_i)^2$
$\frac{\partial E_{rror}}{\partial w} = (o - target)\,o(1-o) * out(h1) = w$

i | TUNING SET: for setting parameters & evaluate errors.
TRAINING SET: GENERATE Decision Tree
TEST SET: Compute performance accuracy

② hidden layers: $\frac{\partial E_{rror}}{\partial w_1} = i1 \cdot out(h1) * (1 - out(h1)) * w5 * (o - target) * o * (1-o) =$
j. Deep learning: multiple layer NNs.

3) Uncertainty
a. Event space / Mutually exclusive events / Random variables
b. Probability Distribution: P(A) the set of values $\langle P(a_1), P(a_2) \dots P(a_n)\rangle$ sum = 1
c. The axioms of Probability: ① $0 \leq P(A) \leq 1$ ② $P(true) = 1$ $P(false) = 0$ ③ $P(A\vee B) = P(A) + P(B) - P(A\wedge B)$
d. joint probability: P(A, B) full joint probability distribution table n random variables, each taking k values has k^n entries e. marginalization. marginal probabilities f. prior (unconditional) probability
g. Conditional Probability $0 \leq P(a|e) \leq 1$ $P(\neg a|e) = 1 - P(a|e)$
$P(a_1|e) + \dots P(a_k|e) = 1$ $P(\neg a|e) = 1 - P(a|e)$

Summary

Conditional Probability: $P(A|B) = P(A,B)/P(B)$

Product Rule: $P(A,B) = P(A|B)P(B)$

Chain Rule: $P(A,B,C,D) = P(A|B,C,D)P(B|C,D)P(C|D)P(D)$

Conditionalized version of chain Rule:
$P(A,B|C) = P(A|B,C)P(B,C)$

Bayes' Rule: $P(A|B) = P(B|A)P(A)/P(B)$

Conditionalized version of Bayes's Rule $P(A|B,C) = \frac{P(B|A,C)\,P(A|C)}{P(B|C)}$

Addition/conditioning rule: $P(A) = P(A,B) + P(A,\neg B)$

$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B)$

independence: ~~...~~ $P(A,B) = P(A) * P(B)$ $P(A|B) = P(A)$

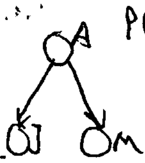
conditional independence.

$P(A|B,C) = P(A|C)$ $P(B|A,C) = P(B|C)$. $P(A,B|C) = P(A|C)P(B|C)$

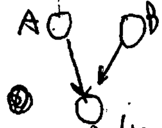
4. Bayesian Networks table $k^n \to$ combination

CPT: Conditional Probability Table

$P(J,M|A) = P(J|A)P(M|A)$



e. d-separation:



A and B are independent given nothing else, but are dependent given C

d. Bayes Net must be acyclic $O(Nk^m)$!

$P(x_1,\dots,x_N) = \prod_i P(x_i|parents(X_i))$ → Inference by Enumeration → $P(d) = \sum\sum\sum\sum\sum\sum P(v,s,t,l,a,b,x,d)$

g. variable elimination – bookkeeping algo to be more efficiency → $P(d) = f_v(t)\,P(s)\,P(l|s)\,P(b|s)\,P(a|t,l)\,P(x|a)\,P(d|a,b)$

$P(t) = f_v(t)$

h. Parameter learning problem: unseen event smoothing

① add-one smoothing $P(A|B,E) = [\#(A)+1]/[\#(A)+1+\#(\neg A)+1]$

② add-delta smoothing $P = (delta + token)/(delta * V + sum token)$

j Naive Bayes Net. Maxim. Likelihood Estimate: $V = \mathrm{argmax}_v\, P(Y=v)\prod_{i=1}^{n} P(X_i=u_i|Y=v)$ classifier.

Assumption: all evidence variables are conditionally independent of each other given the class variable.

Input (evidence): C, Z, H $P(J|C,Z,H) = P(J,C,Z,H)/P(C,Z,H)$ $P(J|C,Z,H) = P(J,C,Z,H)/P(C,Z,H)$

$= P(J,C,Z,H)/[P(J,C,Z,H) + P(\neg J,C,Z,H)]$

Output (query): J $P(J,C,Z,H) = P(J)P(C|J)P(Z|J)P(H|J)$ $P(\neg J,C,Z,H) = P(\neg J)P(C|\neg J)P(Z|\neg J)P(H|\neg J)$

5. Speech Recognition. a. mapping an acoustic signal into a string of words

Task: Analog speech → Signal Processor → Discrete Observation ($o_1 \dots$) → Match Search → word sequence ($w_1 w_2 \dots$)

a. Model: $P(words|signal) = \frac{P(signal|Words)\,P(words)}{P(Signal)}$

b. Language: $P(words)$

c first-order Markov Assumption $P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-1})$

$P(w_i|w_1 \dots w_{i-1}) = P(w_i|w_{i-1},w_{i-2})$ ~~trigram model~~

e. LM: $P(w_1,w_2 \dots w_n) = P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$ bigram model

trigram

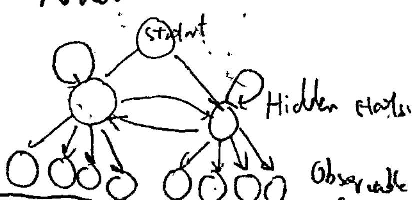
$P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-1}w_{i-2})$

Acoustic Model: $P(signal|Words) \Rightarrow P(Phones|word)$: (MM) $P(Signal|Phones)$ HMM.

MM model: State q_{t+1} is conditionally independent of $\{q_{t-1}, q_{t-2}, \dots q_1\}$ given q_t $P(q_{t+1}=S_j|q_t=S_i) =$

HMM model:

$P(q_{t+1}=S_j|q_t=S_i, q_{t-1}=S_k, \dots q_1=S_l)$



$P(q_1,q_2) = P(q_2|q_1)P(q_1)$

$P(q_2=H) = P(q_2=H|q_1=H)P(q_1=H) + P(q_2=H|q_1=S)P(q_1=S)$

$P(q_3=H) = P(q_3=H|q_1=H,q_2=H)P(q_1=H,q_2=H) + P(q_3=H|q_1=H,q_2=S)P(q_1=H,q_2=S)$

$+\dots = P(q_3=H|q_2=H)P(q_1=H,q_2=H) + \dots P$

6. Propositional logic

Syntax Precedence: highest to lowest $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

a. Interpretation: is a complete T/F assignment to all proposition symbols P.Q.R $\Rightarrow$ 8 interpretation

The semantics (meaning) of a sentence is the set of interpretations in which the sentence evaluates to T

the semantics of the sentence $P \vee Q$ is the set of 6 interpretations

P=T, Q=T, R=True or F

P=T, Q=F, R=T or F

P=F, Q=T, R=T/F

c. A model of a set of sentences is an interpretation in which all the sentences are true

e.g. a sentence is T, how many models?

P	Q	¬P	P∧Q	P∨Q	P⇒Q	P⇔Q
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

b. Knowledge Base (KB): a set of sentences.

A model of a KB is an interpretation in which all sentences in KB True

d. Entailment is the relation of a sentence β logically following from other sentences α, $\alpha \models \beta$ iff α is true, β is true

iff $\alpha \Rightarrow \beta$ is valid / $\alpha \wedge \neg\beta$ is not satisfiable.

Soundness & Completeness.

Soundness: any wff that follows deductively from a set of axioms, KB, is valid (i.e., true in all models).

Completeness: all valid sentences (i.e. true in all models of KB) can be proved from KB and hence are theorems

f. Inference by Enumeration: complete. slow: takes exponential time.

g. Resolution Sound

Complete