Quantifying uncertainty

- partial observability. - noisy sensors -
- uncertainty in action outcomes - complex modelling

⇒ risk falsehood - too weak to decide

Probability

Decision = probability + utility

atomic events: mutually exclusive - exhaustive

Axioms
- $0 \leq P(A) \leq 1$
- $P(true) = 1$ $P(false) = 0$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A|B) = P(A \cap B) / P(B)$

product: $P(A \wedge B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$

chain: - $P(x_1, \dots, x_n) = P(x_1, \dots, x_{n-1}) P(x_n / x_1, \dots, x_{n-1})$

$= P(x_1, \dots, x_{n-2}) P(x_{n-1} / x_1, \dots, x_{n-2}) P(x_n / x_1, \dots, x_{n-1})$

independ.: $P(A|B) = P(A)$. $P(A \wedge B) = P(A) P(B)$

cond. indep: $P(catch / toothache, cavity) = P(catch / cavity)$

$P(toothache / catch, cavity) = P(toothach / cavity)$

$P(toothache, catch / cavity) = P(toothach / cavity) \, P(catch / cavity)$

$= P(T, Cat, Cav) / P(Cav) = \frac{P(T / cat, cav) P(cat, cav) \cdot P(cav)}{P(cav)}$

$= P(T/cav) \cdot P(cat/cav)$. (5 indep numbers).

Bayes' Rule $P(a/b) = \frac{P(b/a) \cdot P(a)}{P(b)}$

$P(b) = P(b/a) \cdot P(a) + P(b/c) \cdot P(c) + \dots$

$= P(a \wedge B) + P(B \wedge C) + P(B \wedge D) + \dots$

$P(cause/effect) = \frac{P(effect / cause) \cdot P(cause)}{P(effect)}$ ← sum over causes.

$P(A/B) = \alpha \; P(A \wedge B)$ $\alpha = P(B)$: normalization

$P(Y | E=e) = \alpha \sum_h P(Y, E=e, H=h)$.

Time & space complexity: indep. removes redundancy.

$O(2^n) \rightarrow O(n)$.

Bayesian Networks

systematic to rep. indep.

- set of nodes & directed influence arrows.
- each node X_i has conditional distribution (CPT) for each node given its parent $P(X_i / Parent(X_i))$ (for each combination of parents rows $= 2^{\# parents}$)

indep: no effect on one another, no common parent.

cond. indep:
- a common neighbor
- have no effect on one another given that neighbor.

Note entries of node's CPT must not sum to 1.

DAG: directed acyclic

e.g. $P(JC, MC, A, !B, !E)$ = "from Bayesian graph"

$P(JC/A) \cdot P(MC/A) \cdot P(A/!B, !E) \cdot P(!B) \cdot P(!E)$.

CPT: - X_i has 2^k rows : k: # parents.
- each row has one number $X_i = T \rightarrow p$.
- while $X_i = F \rightarrow 1-p$.

compactness $O(n \cdot 2^k)$ numbers not $O(2^n)$ if nodes has no more than k parents.

changing order of nodes → changes map.

since conditionally indep. nodes require placing the parent before them, else not cond. indep.

Learning KNN.

k: # nearest neighbors.

1) Find dist from query to each neighbor
2) Sort & find nearest k-th neighbors (k-th min dist)
3) Find category for nearest neighbors.
4) Predict every category based on majority for neigh.

Distances to sample elements:

use Euclidean distance "normalized over the max value of the criterion" -

$$D(X, Y) = \sqrt{\sum_{i=1}^{} \left(\frac{x_i}{\tilde{x}} - \frac{y_i}{\tilde{y}} \right)^2}$$

Nominal values: (Yes/No) assign numerics (0,1)

variation can use ε-approximate nearest neigh.

Learning: Decision Trees

2^{2^n} possible without $IG(A)$

given examples with attribute values (Boolean - discrete - continuous)

classification of example is (T or F)

choosing root:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - rem(A)$$

$$rem(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

$$I(V) = I(P(v_1), \dots P(v_n)) = \sum_{i=1}^{v} -P(v_i) \log_2 P(v_i)$$

choose var with highest info entropy $IG(A)$ in DTL algorithm (Decision Tree learning).

learning curve = % correct on test set = fn. of training set size - (performance measure)

"Try it on new test set of examples"

Intro to Basic Image Processing

spatial image filter: center at $w(0,0)$

w(-1,1)	w(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
-1,-1	0,-1	1,-1

$$f * w = \sum_{\substack{(a,b) \in w \\ (x-a, y-b) \in f}} f(x-a, y-b) \, w(a,b)$$

replace each pixel by linear combination of neigh.

Convolution
1. position the center of filter on 1st pixel and flip the filter.
2. find the inner product b/w the filter & sub-image covered by it.
3. Slide to next pixel....

Lowpass filtering smoothing; (avg - gaussian).
- reduce noise
- image may become blurred.

average filter: all ones $m \times n$ filter

$\frac{1}{m \times n} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots \end{bmatrix}$ multiplied by $1/m \times n$ for averaging.

gaussian

$\frac{1}{\sum w(i,j)} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ (1/16) weighted importance for neighbors.

Median
- get the $n \times m$ subimage centred at a pixel
- sort them & find the median.
- replace the pixel value by median ...

Highpass filters sharpen details & edges.

$\sum w(i,j) = 0$. same procedure

Note edges can be calculated by; $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$
- padding with zeroes
- or replicating ~~image~~ sub-image borders.

Roberts $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Prewitts $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Sobel's $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ Rotate better use 90

2nd order edge detection

Laplacian: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

2D edge detection: canny

$\nabla J = (J_x, J_y) = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right)$

$||\nabla J|| = \sqrt{J_x^2 + J_y^2}$.

$J = I \otimes G$.

CSP

variables (X_i's) $x_i \in D_i$

Domains (D_i's)

Constraints (C_i's)

sol: assign x_i's that are valid in D_i's that are restricted by C_j's

(hard optimality: good temporal & non limited problems)

e.g. Map Coloring - Sheet Problem - Queens

Backtracking

- commutative var assignment.
- single var. assignment to each node
- Depth first for CSP's with single var. assign.

improved choosing the root.

which var? most constrained variable (MRV): min. remain. values. detect early failures.

which val? Least constraining value rules out fewest values in other var's. avoid conflicts.

Forward Checking
- Keep track of remaining values for unassigned var's.
- Terminate when any var has no valid val.

"doesn't detect early failures"

Constraint Propagation: Arc Consistency

$X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y for Y.

Before assigning a value for a variable, check it's arc consistency with dependent variables

→ used as preprocessing procedure.

Local Search for CSPs

work with complete "states"
- allow "initially" states with unsatisfied constraints,
- reassign vars later.

select constrained var: min-conflict heuristic (violates min # of constraints.

most constraining var: most const. on remaining var.s.

Most constrained var: has fewest legal values.

least constraining val: rules out fewest values in remain. var.

Simulated Annealing

1) set initial temp, cooling rate, random initial solution.
2) loop until (reach good-enough sol or system cooled)
3) Select neighbor sol. by making a small change to current solution.
4) decide whether to accept new sol. based on acceptance function.
5) decrease temp & continue looping.

GA

1) encoding: $T \in [a_i, b_i]$ - y precision digits
→ bits $2^{m_i - 1} < (b_i - a_i) * 10^y < 2^{m_i} - 1$

$T_i = a_i + \text{decimal}(\text{substring}_i) * \frac{b_i - a_i}{2^{m_i} - 1}$

2) Initialize: randomly.
3) Evaluate fitness: "maximization problem"
4) Selection:
- $F = \sum_{k=1}^{\#chromo} eval(C_k)$
- find $P_k = \frac{eval(C_k)}{F}$.
- find $q_k = \sum_{j=1}^{k} P_j$.
- Generate N random num.

if $r <= q_1$ → C_1 selected.
else $q_{k-1} < r \leq q_k$ → select C_k
might have repeated chromosomes -
- Get new N chromosomes.

5) Crossover: rate P_c.
- Generate N random #'s.
- Select $C(k)$ if $R(k) < P_c$.
- Place chrom. in order of selection.

C_1 $C[1] = C_1 \times C_3$
C_3 $C[3] = C_3 \times C_5$
C_5 $C[5] = C_5 \times C_1$

- Determine position of crossover.
generate random #'s for each crossover
$R_k \in [1,$ ~~length~~ length of chromo $-1]$
for crossover$_k$ cut after gene R_k.
- New set: old with chrom. from crossover stage

6) Mutation P_m %
- Find total #genes = # chromo × gene/chromo.
- find # Mutants(M) $P_m \times$ (#genes)
- Generate M random #'s $\in$ [1, #genes] flip the gene (binary bit)
- get resultant population

7) Re-evaluate fitness & Iterate

A* tree

h(n) admissible → Tree search optimal
h(n) consistent → Graph " "

Consistent $h(n) \leq c(n, a, n') + h(n')$
$f(n') \geq f(n)$. non-decreasing along path

admissible $h(n) \leq h^*(n)$
true cost to reach goal from n.

Greedy: $f(n) = h(n)$
A* Search: $f(n) = h(n) + g(n)$

	FIFO Breadth first	Uniform cost	LIFO Depth first	Depth limited	LIFO Iterative deepening
complete	Yes	Y	~~Y~~ N	N	Y
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	~~$O(b^l)$~~	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b \times m)$	$O(bl)$	$O(bd)$
optimal	Y	Y	N	N	Y
Descrip.	expand level at a time $f(n) = depth(n)$	expand least cost unexpanded node $f(n) = g(n)$	expand deepest unexpanded node $f(n) = -depth(n)$	same as depth first but limited depth	same as depth-first but iterat on level limit.