COMP9414: Artificial Intelligence

Lecture 10: Review

Wayne Wobcke

e-mail:w.wobcke@unsw.edu.au

Lectures

- Artificial Intelligence and Agents
- Problem Solving and Search
- Constraint Satisfaction Problems
- Logic and Knowledge Representation
- Reasoning with Uncertainty
- Machine Learning
- Natural Language Processing
- Knowledge Based Systems
- Neural Networks and Reinforcement Learning

What is an Agent?

An entity

- situated: operates in a dynamically changing environment
- reactive: responds to changes in a timely manner
- autonomous: can control its own behaviour
- proactive: exhibits goal-oriented behaviour
- communicating: coordinate with other agents??

Examples: humans, dogs, ..., insects, sea creatures, ..., thermostats?

Where do current robots sit on the scale?

Environment Types

Fully Observable vs Partially Observable
: Agent's sensors give access to complete state of environment (no internal state required)

Deterministic vs Stochastic
: Next state of environment determined only by current state and agent's choice of action

Episodic vs Sequential
: Agent's experience divided into "episodes"; agent doesn't need to think ahead in episodic environment

Static vs Dynamic
: Environment changes while agent deliberates

Discrete vs Continuous
: Limited number of distinct, clearly defined percepts and actions

Specifying Agents

- percepts: inputs to the agent via sensors
- actions: outputs available to the agent via effectors
- goals: objectives or performance measure of the agent
- environment: world in which the agent operates

Most generally, a function from percept sequences to actions

Ideally rational agent does whatever action is expected to maximize some performance measure – the agent may not know the performance measure (Russell and Norvig 2010)

Resource bounded agent must make "good enough" decisions based on its perceptual, computational and memory limitations (design tradeoffs)

Example Agents

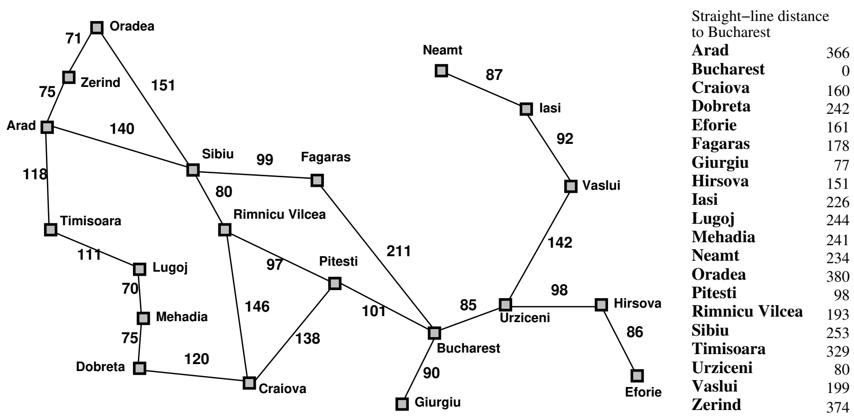
Agent Type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient responses	Questions, tests, treatments	Healthy patient, minimise costs	Patient, hospital
Satellite image system	Pixels of varying intensity, colour	Print categorisation of scene	Correct categorisation	Images from orbiting satellite
Automated taxi driver	Cameras, speedometer, GPS, sonar, microphone	Steer, accelerate, brake, talk to passenger	Safe, fast, legal, comfortable trip, maximise profits	Roads, other traffic, pedestrians, customers
Robocup robot	Camera images, laser range finder readings, sonar readings	Move motors, "kick" ball	Score goals	Playing field with ball and other robots

Based on Russell and Norvig (2010) Figure 2.5.

State Space Search Problems

- State space — set of all states reachable from initial state(s) by any action sequence
- Initial state(s) — element(s) of the state space
- Transitions
  - Operators — set of possible actions at agent's disposal; describe state reached after performing action in current state, or
  - Successor function — $s(x)$= set of states reachable from state x by performing a single action
- Goal state(s) — element(s) of the state space
- Path cost — cost of a sequence of transitions used to evaluate solutions (apply to optimization problems)

Example Problem — Romania Map



Summary — Blind Search

Criterion	Breadth First	Uniform Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional
Time	b^d	b^d	b^m	b^l	b^d	$b^{\frac{d}{2}}$
Space	b^d	b^d	bm	bl	bd	$b^{\frac{d}{2}}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

b — branching factor
d — depth of shallowest solution
m — maximum depth of tree
l — depth limit

A* Search

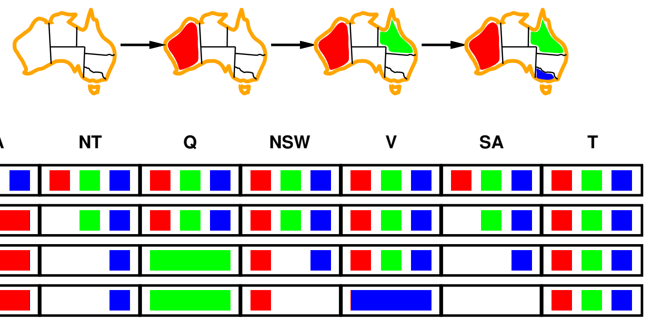
- **Idea:** Use both cost of path generated and estimate to goal to order nodes on the frontier
- $g(n)$ = cost of path from start to n; $h(n)$ = estimate from n to goal
- Order priority queue using function $f(n) = g(n) + h(n)$
- $f(n)$ is the estimated cost of the cheapest solution extending this path
- Expand node from frontier with smallest f-value
- Essentially combines uniform-cost search and greedy search

Constraint Satisfaction Problems

- Constraint Satisfaction Problems are defined by a set of variables X_i, each with a domain D_i of possible values, and a set of constraints C
- Aim is to find an assignment to each the variables X_i (a value from the domain D_i) such that all of the constraints C are satisfied

Forward Checking

Idea: Keep track of remaining legal values for unassigned variables
Terminate search when any variable has no legal values

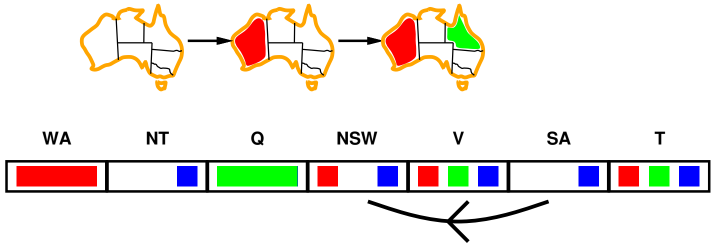


Arc Consistency

Simplest form of constraint propagation is arc consistency

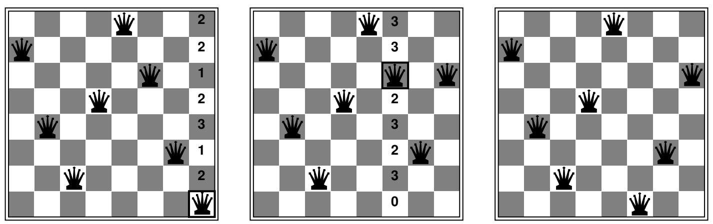
Arc (constraint) $X \rightarrow Y$ is arc consistent if

for every value x in $dom(X)$ there is some allowed y in $dom(Y)$



Make $X \rightarrow Y$ arc consistent by removing any such x from $dom(X)$

Hill Climbing by Min-Conflicts



- Variable selection: randomly select any conflicted variable
- Value selection by min-conflicts heuristic
  - Choose value that violates fewest constraints
  - Can (often) solve n-Queens for $n \approx 10,000,000$

Propositional Logic

- Use letters to stand for "basic" propositions; combine them into more complex sentences using operators for not, and, or, implies, iff
- Propositional connectives:

$\neg$	negation	$\neg P$	"not P"
$\wedge$	conjunction	$P \wedge Q$	"P and Q"
$\vee$	disjunction	$P \vee Q$	"P or Q"
$\rightarrow$	implication	$P \rightarrow Q$	"If P then Q"
$\leftrightarrow$	bi-implication	$P \leftrightarrow Q$	"P if and only if Q"

Truth Table Semantics

- The semantics of the connectives can be given by truth tables

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
True	True	False	True	True	True	True
True	False	False	False	True	False	False
False	True	True	False	True	True	False
False	False	True	False	False	True	True

- One row for each possible assignment of True/False to variables
- **Important:** P and Q are **any** sentences, including complex sentences

Definitions

- A sentence is valid if it is True under all possible assignments of True/False to its variables (e.g. $P \vee \neg P$)
- A tautology is a valid sentence
- Two sentences are equivalent if they have the same truth table, e.g. $P \wedge Q$ and $Q \wedge P$
  - So P is equivalent to Q if and only if $P \leftrightarrow Q$ is valid
- A sentence is satisfiable if there is some assignment of True/False to its variables for which the sentence is True
- A sentence is unsatisfiable if it is not satisfiable (e.g. $P \wedge \neg P$)
  - Sentence is False for all assignments of True/False to its variables
  - So P is a tautology if and only if $\neg P$ is unsatisfiable

Conversion to Conjunctive Normal Form

- Eliminate $\leftrightarrow$ rewriting $P \leftrightarrow Q$ as $(P \rightarrow Q) \wedge (Q \rightarrow P)$
- Eliminate $\rightarrow$ rewriting $P \rightarrow Q$ as $\neg P \vee Q$
- Use De Morgan's laws to push $\neg$ inwards (repeatedly)
  - Rewrite $\neg(P \wedge Q)$ as $\neg P \vee \neg Q$
  - Rewrite $\neg(P \vee Q)$ as $\neg P \wedge \neg Q$
- Eliminate double negations: rewrite $\neg\neg P$ as P
- Use the distributive laws to get CNF [or DNF] – if necessary
  - Rewrite $(P \wedge Q) \vee R$ as $(P \vee R) \wedge (Q \vee R)$ [for CNF]
  - Rewrite $(P \vee Q) \wedge R$ as $(P \wedge R) \vee (Q \wedge R)$ [for DNF]

Resolution Rule of Inference

$$A_1 \vee \cdots \vee A_m \vee B \qquad\qquad \neg B \vee C_1 \vee \cdots \vee C_n$$

$$A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n$$

where B is a propositional variable and A_i and C_j are literals

- B and $\neg B$ are complementary literals
- $A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n$ is the resolvent of the two clauses
- Special case: If no A_i and C_j, resolvent is empty clause, denoted $\Box$

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated query into CNF
- Repeatedly apply resolution until either the empty clause (contradiction) is derived or no more clauses can be derived
- If the empty clause is derived, answer 'yes' (query follows from knowledge base), otherwise answer 'no' (query does not follow from knowledge base)

Random Variables

- Propositions are random variables that can take on several values

  $P(Weather = Sunny) = 0.8$
  $P(Weather = Rain) = 0.1$
  $P(Weather = Cloudy) = 0.09$
  $P(Weather = Snow) = 0.01$

- Every random variable X has a domain of possible values $\langle x_1, x_2, \cdots, x_n \rangle$
- Probabilities of all possible values $\mathbf{P}(Weather) = \langle 0.8, 0.1, 0.09, 0.01 \rangle$ is a probability distribution
- $\mathbf{P}(Weather, Appendicitis)$ is a combination of random variables represented by cross product (can also use logical connectives $P(A \wedge B)$ to represent compound events)

Conditional Probability by Enumeration

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

$$P(\neg cavity | toothache) = \frac{P(\neg cavity \wedge toothache)}{P(toothache)}$$
$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

Bayes' Rule

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

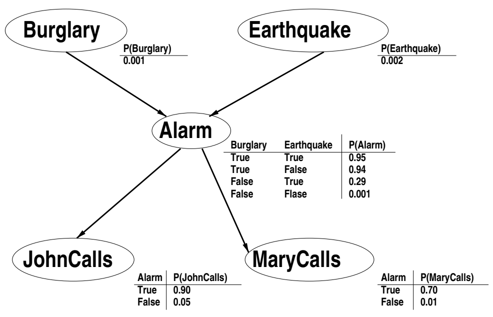
- AI systems abandon joint probabilities and work directly with conditional probabilities using Bayes' Rule
- Deriving Bayes' Rule:

  $P(A \wedge B) = P(A|B)P(B)$ (Definition)
  $P(B \wedge A) = P(B|A)P(A)$ (Definition)
  So $P(A|B)P(B) = P(B|A)P(A)$ since $P(A \wedge B) = P(B \wedge A)$
  Hence $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$ if $P(A) \neq 0$

- Note: If $P(A) = 0$, $P(B|A)$ is undefined

Bayesian Networks

- Example (Pearl, 1988)



- Probabilities summarize potentially infinite set of possible circumstances

Example – Causal Inference

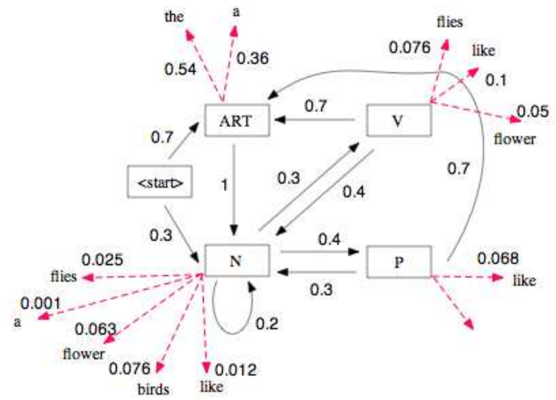
- $P(JohnCalls|Burglary)$
- $P(J|B) = P(J|A \land B).P(A|B) + P(J|\neg A \land B).P(\neg A|B)$
  $= P(J|A).P(A|B) + P(J|\neg A).P(\neg A|B)$
  $= P(J|A).P(A|B) + P(J|\neg A).(1 - P(A|B))$
- Now $P(A|B) = P(A|B \land E).P(E|B) + P(A|B \land \neg E).P(\neg E|B)$
  $= P(A|B \land E).P(E) + P(A|B \land \neg E).P(\neg E)$
  $= 0.95 \times 0.002 + 0.94 \times 0.998 = 0.94002$
- Therefore $P(J|B) = 0.90 \times 0.94002 + 0.05 \times 0.05998 = 0.849017$
- **Fact 3**: $P(X|Z) = P(X|Y \land Z).P(Y|Z) + P(X|\neg Y \land Z).P(\neg Y|Z)$, since $X \land Z \Leftrightarrow (X \land Y \land Z) \lor (X \land \neg Y \land Z)$ (conditional version of Fact 2)

Bigram Model

Maximize $P(w_1, \cdots, w_n|t_1, \cdots, t_n).P(t_1, \cdots, t_n)$

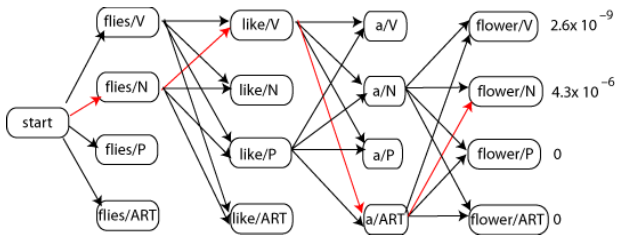
- Apply independence assumptions (Markov assumptions)
  - $P(w_1, \cdots, w_n|t_1, \cdots, t_n) = \Pi P(w_i|t_i)$
  - Observations (words) depend only on states (tags)
  - $P(t_1, \cdots, t_n) = P(t_n|t_{n-1}). \cdots .P(t_0|\phi)$, where ϕ = start
  - Bigram model: state (tag) depends only on previous state (tag)
- Estimate probabilities
  - $P(t_i|t_j)$ = #((t_j,t_i occurs)/#(t_j starts a bigram)
  - Choose tag sequence that maximizes $\Pi P(w_i|t_i).P(t_i|t_{i-1})$
  - Parts of speech generated by finite state machine

Hidden Markov Model for POS Tagging



Viterbi Algorithm

1. Sweep forward (one word at a time) saving only the most likely sequence (and its probability) for each tag t_i of w_i
2. Select highest probability final state
3. Follow chain backwards to extract tag sequence



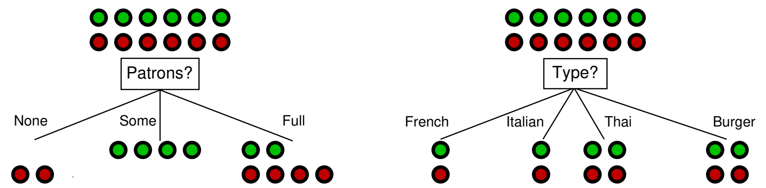
Supervised Learning

- Given a training set and a test set, each consisting of a set of items for each item in the training set, a set of features and a target output
- Learner must learn a model that can predict the target output for any given item (characterized by its set of features)
- Learner is given the input features and target output for each item in the training set
  - Items may be presented all at once (batch) or in sequence (online)
  - Items may be presented at random or in time order (stream)
  - Learner **cannot** use the test set **at all** in defining the model
- Model is evaluated by its performance on predicting the output for each item in the test set

Restaurant Training Data

	Alt	Bar	F/S	Hun	Pat	Price	Rain	Res	Type	Est	Wait?
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

Choosing an Attribute to Split

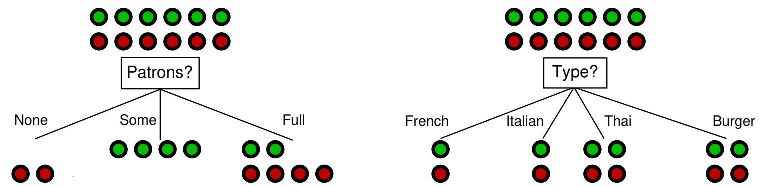


Patrons is a “more informative” attribute than Type, because it splits the examples more nearly into sets that are “all positive” or “all negative”

This notion of “informativeness” can be quantified using the mathematical concept of “entropy”

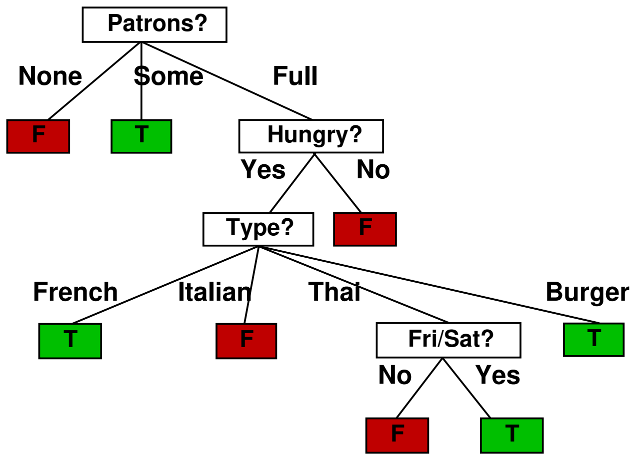
A parsimonious tree can be built by minimizing the entropy at each step

Information Gain



$$\begin{aligned}
\text{For Patrons, Entropy} &= \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})\right] \\
&= 0+0+\frac{1}{2}\left[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)\right]=0.459 \\
\text{For Type, Entropy} &= \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1)=1
\end{aligned}$$

Induced Decision Tree



Laplace Error and Pruning

Following Ockham's Razor, prune branches that do not provide much benefit in classifying the items (aids generalization, avoids overfitting)

For a leaf node, all items will assigned the majority class at that node. Estimate error rate on the (unseen) test items using the Laplace error

$$E = 1 - \frac{n+1}{N+k}$$

N = total number of (training) items at the node
n = number of (training) items in the majority class
k = number of classes

If the average Laplace error of the children exceeds that of the parent node, prune off the children

Text Classification

- Input: A document (e-mail, news article, review, **tweet**)
- Output: One class drawn from a fixed set of classes
  - So text classification is a multi-class classification problem
  - ... and sometimes a multi-label classification problem
- Learning Problem
  - Input: Training set of labelled documents $\{(d_1, c_1), \cdots\}$
  - Output: Learned classifier that maps d to predicted class c

Bernoulli Model

Maximize $P(x_1, \cdots, x_n|c).P(c)$

- Features are presence or absence of word w_i in document
- Apply independence assumptions
  - $P(x_1, \cdots, x_n|c) = P(x_1|c). \cdots .P(x_n|c)$
  - Probability of word w (not) in class c independent of context
- Estimate probabilities
  - $P(w|c)$ = #(w in document in class c)/#(documents in class c)
  - $P(\neg w|c) = 1 - P(w|c)$
  - $P(c)$ = #(documents in class c)/#documents

Naive Bayes Classification

w_1	w_2	w_3	w_4	$Class$
1	0	0	1	1
0	0	0	1	0
1	1	0	1	0
1	0	1	1	1
0	1	1	0	0
1	0	0	0	0
1	0	1	0	1
0	1	0	0	1
0	1	0	1	0
1	1	1	0	0

	$Class = 1$	$Class = 0$
$P(Class)$	0.40	0.60
$P(w_1\|Class)$	0.75	0.50
$P(w_2\|Class)$	0.25	0.67
$P(w_3\|Class)$	0.50	0.33
$P(w_4\|Class)$	0.50	0.50

To classify document with w_2, w_3, w_4

- $P(Class = 1|\neg w_1, w_2, w_3, w_4)$
  $\approx ((1-0.75)*0.25*0.5*0.5)*0.4$
  $= 0.00625$
- $P(Class = 0|\neg w_1, w_2, w_3, w_4)$
  $\approx ((1-0.5)*0.5*0.67*0.33)*0.6$
  $= 0.03333$

Bag of Words Model

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1

MNB Example

	Words	$Class$
d_1	Chinese Beijing Chinese	c
d_2	Chinese Chinese Shanghai	c
d_3	Chinese Macao	c
d_4	Tokyo Japan Chinese	j
d_5	Chinese Chinese Chinese Tokyo Japan	?

$P(\text{Chinese}|c)$ = (5+1)/(8+6) = 3/7
$P(\text{Tokyo}|c)$ = (0+1)/(8+6) = 1/14
$P(\text{Japan}|c)$ = (0+1)/(8+6) = 1/14
$P(\text{Chinese}|j)$ = (1+1)/(3+6) = 2/9
$P(\text{Tokyo}|j)$ = (1+1)/(3+6) = 2/9
$P(\text{Japan}|j)$ = (1+1)/(3+6) = 2/9

To classify document d_5

- $P(c|d_5) \propto [(3/7)^3 \,.\, 1/14 \,.\, 1/14] \,.\, 3/4$
  ≈ 0.0003
- $P(j|d_5) \propto [(2/9)^3 \,.\, 2/9 \,.\, 2/9] \,.\, 1/4$
  ≈ 0.0001
- Choose $Class\ c$

Natural Languages – Ambiguity

- Natural languages exhibit ambiguity

  "The fisherman went to the bank" (lexical)
  "The boy saw a girl with a telescope" (structural)
  "Every student took an exam" (semantic)
  "The table won't fit through the doorway because it is too [wide/narrow]" (pragmatic)

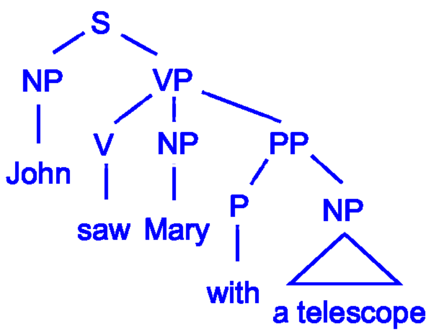
- Ambiguity makes it difficult to interpret meaning of phrases/sentences
  - But also makes inference harder to define and compute
- Resolve ambiguity by mapping to unambiguous representation

Typical (Small) Grammar

S → NP VP
NP → [Det] Adj* N [AP | PP | Rel Clause]*
VP → V [NP] [NP] PP*
AP → Adj PP
PP → P NP
Det → a | an | the | . . .
N → John | park | telescope | . . .
V → saw | likes | believes | . . .
Adj → hot | hotter | . . .
P → in | . . .

Special notation: * is "0 or more"; [. .] is "optional"

Syntactic Structure

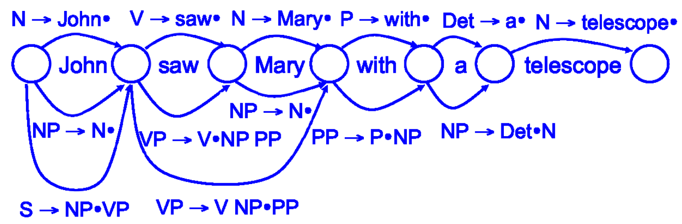


Syntactically ambiguous = more than one parse tree

Chart Parsing

- Use a chart to record parsed fragments and hypotheses
- Hypotheses N → α • β where N → αβ is a grammar rule means "trying to parse N as αβ and have so far parsed α"
- One node in chart for each word gap, start and end
- One arc in chart for each hypothesis
- At each step, apply fundamental rule
  - If chart has N → α • Bβ from n_1 to n_2 and B → γ• from n_2 to n_3 add N → αB • β from n_1 to n_3
- Accept sentence when S → α• is added from start to end
- Can produce any sort of derivation

Example Chart



First-Order Logic

- **Terms:** constants, variables, functions applied to terms (refer to objects)
  - e.g. a, $f(a)$, $mother\_of(Mary)$, …
- **Atomic formulae:** predicates applied to tuples of terms
  - e.g. $likes(Mary, mother\_of(Mary))$, $likes(x, a)$
- **Quantified formulae:**
  - e.g. $\forall x\, likes(x, a)$, $\exists x\, likes(x, mother\_of(y))$
  - here the second occurrences of x are bound by the quantifier ($\forall$ in the first case, $\exists$ in the second) and y in the second formula is free

Converting English into First-Order Logic

- Everyone likes lying on the beach — $\forall x\, likes\_lying\_on\_beach(x)$
- Someone likes Fido — $\exists x\, likes(x, Fido)$
- No one likes Fido — $\neg\exists x\, likes(x, Fido)$ (or $\forall x \neg likes(x, Fido)$)
- Fido doesn't like everyone — $\neg\forall x\, likes(Fido, x)$
- All cats are mammals — $\forall x\,(cat(x) \rightarrow mammal(x))$
- Some mammals are carnivorous — $\exists x\,(mammal(x) \wedge carnivorous(x))$
- Note: $\forall x A(x) \Leftrightarrow \neg\exists x \neg A(x)$, $\exists x A(x) \Leftrightarrow \neg\forall x \neg A(x)$

Defining Semantic Properties

Brothers are siblings

$\forall x \forall y\,(brother(x, y) \rightarrow sibling(x, y))$

"Sibling" is symmetric

$\forall x \forall y\,(sibling(x, y) \leftrightarrow sibling(y, x))$

One's mother is one's female parent

$\forall x \forall y\,(mother(x, y) \leftrightarrow (female(x) \wedge parent(x, y))$

A first cousin is a child of a parent's sibling

$\forall x \forall y\,(firstcousin(x, y) \leftrightarrow \exists p \exists s\, parent(p, x) \wedge sibling(p, s) \wedge parent(s, y)$

First-Order Resolution

$$A_1 \vee \cdots \vee A_m \vee B \qquad\qquad \neg B' \vee C_1 \vee \cdots \vee C_n$$

$$(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)\theta$$

where B, B' are positive literals, A_i, C_j are literals, θ is an mgu of B and B'

- B and $\neg B'$ are complementary literals
- $(A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n)\theta$ is the resolvent of the two clauses
- Special case: If no A_i and C_j, resolvent is empty clause, denoted $\Box$

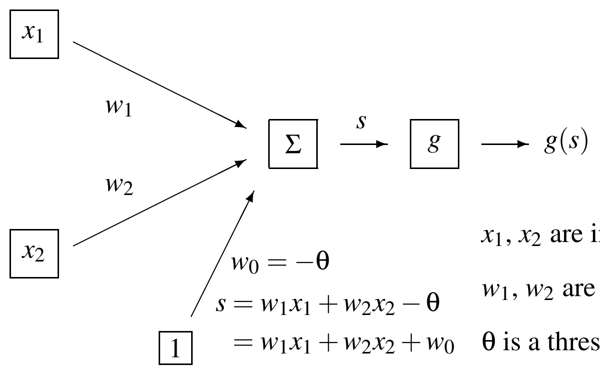
Unification

- A unifier of two atomic formulae is a substitution of terms **for variables** that makes them identical
  - Each variable has at most one associated term
  - Substitutions are applied simultaneously
- Unifier of $P(x,\ f(a),\ z)$ and $P(z,\ z,\ u)$: $\{x/f(a),\ z/f(a),\ u/f(a)\}$
- Substitution σ_1 is a more general unifier than a substitution σ_2 if for some substitution τ, $\sigma_2 = \sigma_1\tau$ (i.e. σ_1 followed by τ)
- **Theorem.** If two atomic formulae are unifiable, they have a most general unifier (mgu).

Examples

- $\{P(x,a), P(b,c)\}$ is not unifiable
- $\{P(f(x),y), P(a,w)\}$ is not unifiable
- $\{P(x,c), P(b,c)\}$ is unifiable by $\{x/b\}$
- $\{P(f(x),y), P(f(a),w)\}$ is unifiable by
  $\sigma = \{x/a, y/w\}$, $\tau = \{x/a, y/a, w/a\}$, $\upsilon = \{x/a, y/b, w/b\}$
  Note that σ is an mgu and $\tau = \sigma\theta$ where $\theta = \ldots$?
- $\{P(x), P(f(x))\}$ is not unifiable (c.f. occur check!)

McCulloch & Pitts Model of a Single Neuron



x_1, x_2 are inputs

w_1, w_2 are synaptic weights

θ is a threshold

w_0 is a **bias** weight

g is transfer function

Perceptron Learning Rule

Adjust the weights as each input is presented

Recall $s = w_1x_1 + w_2x_2 + w_0$

if $g(s) = 0$ but should be 1,

$$w_k \leftarrow w_k + \eta x_k$$
$$w_0 \leftarrow w_0 + \eta$$
$$\text{so}\quad s \leftarrow s + \eta(1 + \sum_k x_k^2)$$

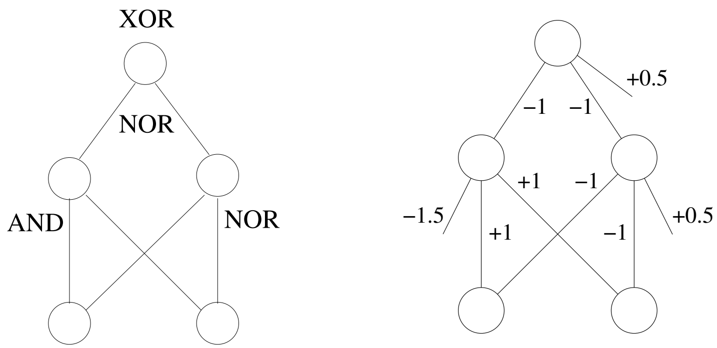
if $g(s) = 1$ but should be 0,

$$w_k \leftarrow w_k - \eta x_k$$
$$w_0 \leftarrow w_0 - \eta$$
$$\text{so}\quad s \leftarrow s - \eta(1 + \sum_k x_k^2)$$

otherwise weights are unchanged ($\eta > 0$ is called the **learning rate**)

Theorem: This will eventually learn to classify the data correctly, as long as they are linearly separable

Multi-Layer Neural Networks

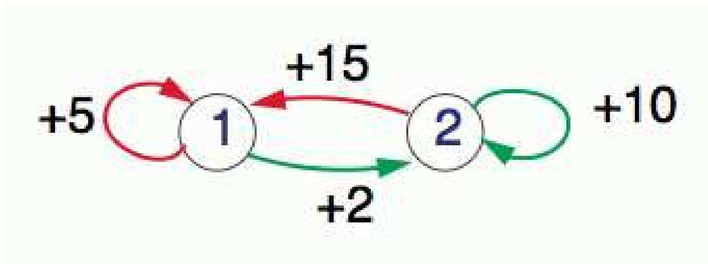


Question: Given an explicit logical function, we can design a multi-layer neural network by hand to compute that function – but if we are just given a set of training data, can we train a multi-layer network to fit this data?

Reinforcement Learning Framework

- Agent interacts with its environment
- There is a set S of *states* and a set A of *actions*
- At each time step t, the agent is in some state s_t and must choose an action a_t, whereupon it goes into state $s_{t+1} = \delta(s_t, a_t)$ and receives reward $r(s_t, a_t)$
- In general, $r()$ and $\delta()$ can be multi-valued, with a random element
- The aim is to find an optimal *policy* $\pi : S \rightarrow A$ which maximizes the cumulative reward

Example: Delayed Rewards



Calculation

Theorem: In a deterministic environment, for an optimal policy, the value function V^* satisfies the Bellman equations: $V^*(s) = r(s,a) + \gamma V^*\delta(s,a)$ where $a = \pi^*(s)$ is the optimal action at state s.

Let $\delta^*(s)$ be the transition function for $\pi^*(s)$ and suppose $\gamma = 0.9$

1. Suppose $\delta^*(s_1) = s_1$. Then $V^*(s_1) = 5 + 0.9V^*(s_1)$ so $V^*(s_1) = 50$
   Suppose $\delta^*(s_2) = s_2$. Then $V^*(s_2) = 10 + 0.9V^*(s_2)$ so $V^*(s_2) = 100$
2. Suppose $\delta^*(s_1) = s_2$. Then $V^*(s_1) = 2 + 0.9V^*(s_2)$ so $V^*(s_1) = 92$
   Suppose $\delta^*(s_2) = s_2$. Then $V^*(s_2) = 10 + 0.9V^*(s_2)$ so $V^*(s_2) = 100$
3. Suppose $\delta^*(s_1) = s_2$. Then $V^*(s_1) = 2 + 0.9V^*(s_2)$ so $V^*(s_1) = 81.6$
   Suppose $\delta^*(s_2) = s_1$. Then $V^*(s_2) = 15 + 0.9V^*(s_1)$ so $V^*(s_2) = 88.4$

So 2 is the optimal policy

Examination Instructions

(1) READING TIME – 10 MINUTES

(2) TIME ALLOWED – 2 HOURS

(3) THIS EXAMINATION COUNTS FOR 50% OF THE FINAL MARK

(4) TOTAL NUMBER OF QUESTIONS – 35

(5) ANSWER **ALL** QUESTIONS

(6) ALL QUESTIONS ARE OF EQUAL WEIGHT

(7) CHOOSE **ONE** ANSWER PER QUESTION

For any queries during the exam, contact the Course Convenor (w.wobcke@unsw.edu.au) or Course Admin (alfredk@unsw.edu.au). Any announcements during the exam will be sent to students using the course e-mail alias.

Examination Rules

Fit to Sit Rule: By sitting this exam, you are declaring that you are fit to do so and cannot later apply for Special Consideration. If, during the exam, you feel unwell to the point that you cannot continue with the exam, you should take the following steps:

1. Stop working on the exam and take note of the time;
2. Contact the Course Convenor (w.wobcke@unsw.edu.au) or Course Admin (alfredk@unsw.edu.au) immediately by e-mail or chat and advise them that you are unwell;
3. Immediately submit a Special Consideration application saying that you felt ill during the exam and were unable to continue;
4. Obtain a doctor's certificate within 24 hours and attach it to the Special Consideration application;
5. If you were unable to advise the Course Convenor or Course Admin of the illness during the exam, attach screenshots of this conversation to the Special Consideration application.

Examination Rules

Technical Issues: If you experience a technical issue during the exam, take the following steps:

1. Take screenshots of as many of the following as possible:
   - error messages
   - screen(s) not loading
   - timestamped speed tests
   - power outage maps
2. Contact the Course Convenor (w.wobcke@unsw.edu.au) or Course Admin (alfredk@unsw.edu.au) by e-mail or chat as soon as possible to advise them of the issue;
3. Submit a Special Consideration application immediately after the exam, including all appropriate screenshots.