COMP 424 - Artificial Intelligence
Lecture 12: Uncertainty

Instructor: Jackie CK Cheung (jcheung@cs.mcgill.ca)

Readings: R&N Ch 13

Recap: planning

- A plan is a collection of actions for performing some task.

  E.g. Assembling your new IKEA desk.

- Planning allows us to think about problems in a structured way
  - Express complex actions
  - Express preconditions and effects of actions

- **Today**: solving planning problems, uncertainty

STRIPS (Stanford Research Institute Planning System)

- **Domain**: set of typed objects represented as propositions.

- **States**: represented as **first-order predicates over objects**.

  Closed-world assumption:

  - everything not stated is **false**
  - only objects in the world are the ones defined.

- **Operators**: defined in terms of:
  - **Preconditions**: when can the action be applied?
  - **Effects**: what happens after the action?

  (No explicit description of how the action should be executed.)

STRIPS representations

- **States** are represented as conjunctions of predicates:

  $In(robot, room) \wedge Closed(door) \wedge \ldots$

- **Goals** are represented as conjunctions of predicates:

  $In(robot, r) \wedge In(Charger, r)$

- **Operators**:
  - Name: $Go(x, y)$
  - Preconditions are represented as conjunctions:

    $At(robot, x) \wedge Path(x, y)$

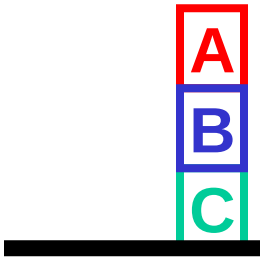
  - Postconditions are represented as conjunctions:

    $At(robot, y) \wedge \neg At(robot, x)$

- Variables (e.g. *x, y, r*) can be instantiated only with objects of correct type.

Welcome to the Blocks World!

Initial state



Goal state

Initial state = On(A,table) ∧ On(B,table) ∧ On(C,table) ∧ Clear(A) ∧ Clear(B) ∧ Clear(C)

Goal state = On(A,B) ∧ On (B,C)

Exercise: Fill in the preconditions and effects of these actions

Action = Move(b,x,y) Move b from x to y, where y is not the table.

Precondition =

Effect =

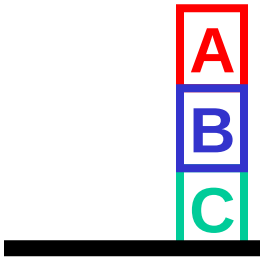
Action = MoveToTable(b,x) Move b from x to the table.

Preconditions =

Effect =

Welcome to the Blocks World!

Initial state



Goal state

Initial state = On(A,table) ∧ On(B,table) ∧ On(C,table) ∧ Clear(A) ∧ Clear(B) ∧ Clear(C)

Goal state = On(A,B) ∧ On (B,C)

Action = Move(b,x,y)

Precondition = On(b,x) ∧ Clear(b) ∧ Clear(y)

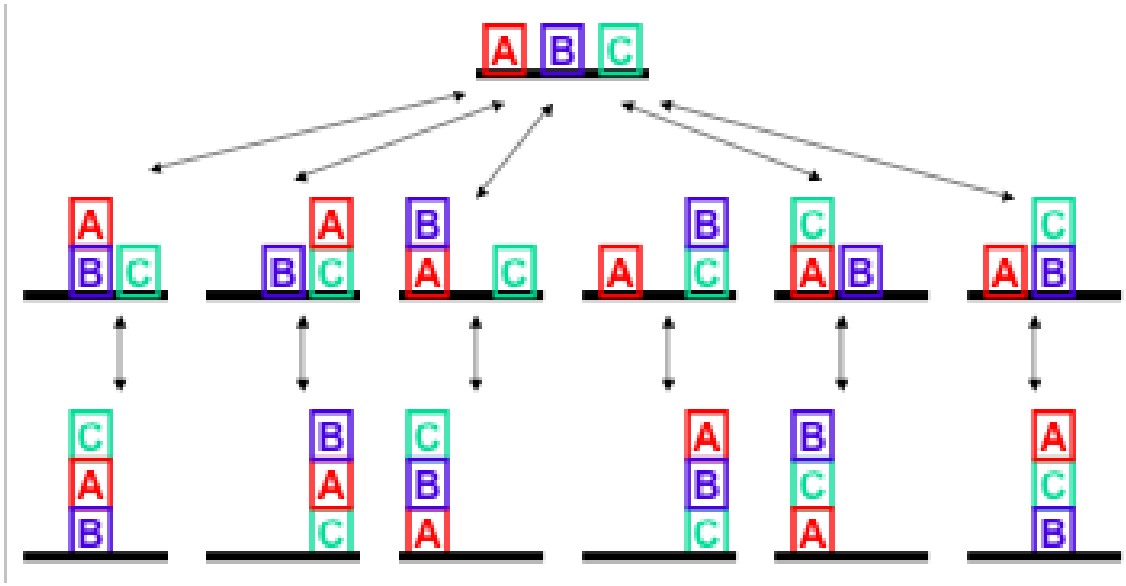
Effect = On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y)

Action = MoveToTable(b,x)

Preconditions = On(b,x) ∧ Clear(b)

Effect = On(b,table) ∧ Clear(x) ∧ ¬On(b,x)

STRIPS state transitions



Exercise

Write out a plan that actually achieves the goal state

Pros and cons of STRIPS?

- **Pros**:
  - Since it is restricted, inference can be done efficiently.
  - All operators can be viewed as simple *deletions* and *additions* of propositions to the knowledge base.

- **Cons**:
  - Assumes that a small number of propositions will change for each action (otherwise operators are hard to write down, and reasoning becomes expensive.)
  - Limited language (preconditions and effects are expressed as conjunctions), so not applicable to all domains of interest.

Two basic approaches to planning

1. **State-space planning** works at the level of states and operators.
   - Finding a plan is formulated as a search through state space for a path from the start state to the goal state(s).
   - Most similar to constructive search.

2. **Plan-space planning** works at the level of plans.
   - Partial order planning – read R&N 10.4.4

Progression (forward) planning

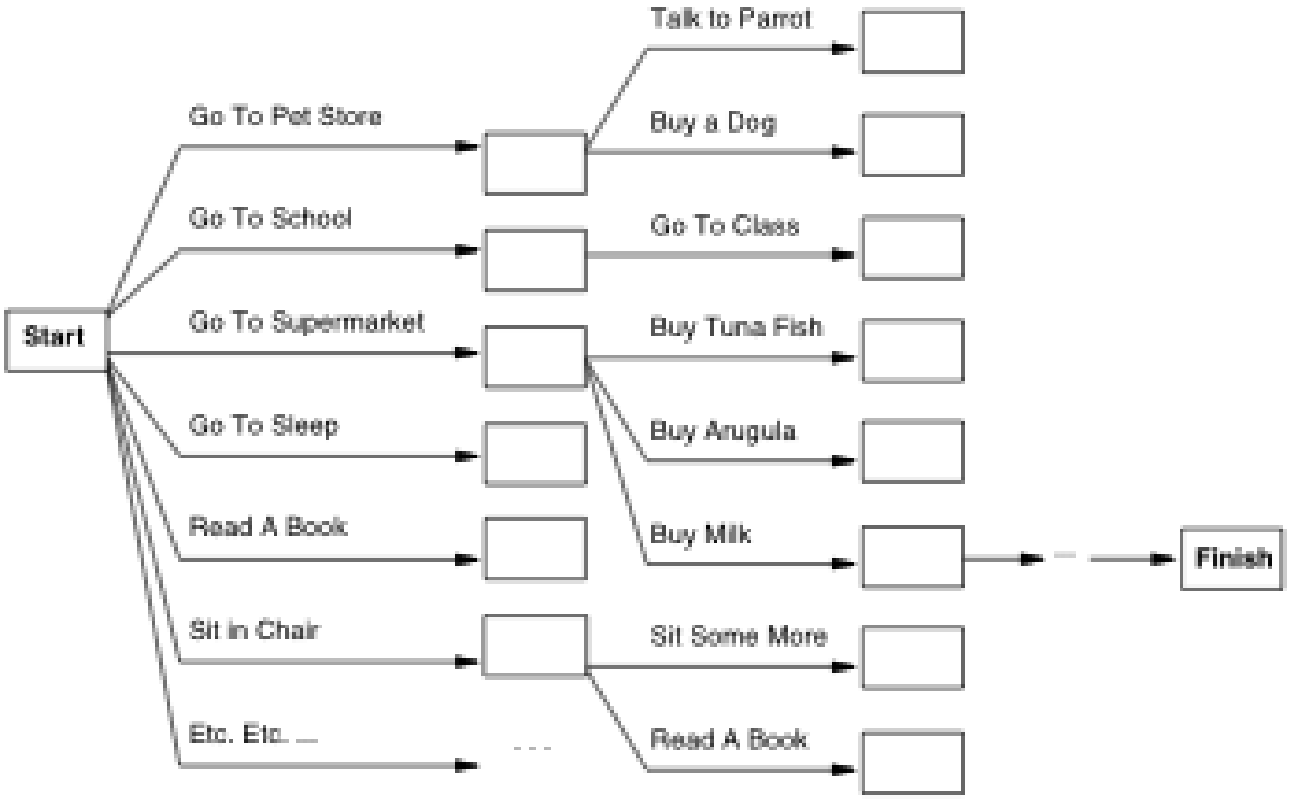
Starting from the initial state, do:

1. Determine all operators that are applicable in the start state by examining preconditions.
2. Ground the operators, by replacing any variables with constants.
3. Choose an operator to apply.
4. Determine the new content of the knowledge base, based on the operator description (apply the effects from the add- and delete lists).

Repeat until goal state is reached.

Example: Supermarket domain

- We're at home, but we need to buy milk, bananas, and a cordless drill (don't ask).



Example: Supermarket domain

- In the start state we have *At(Home)*, which allows us to apply operators of the type *Go(x,y)*.
- The operator can be instantiated as *Go(Home, HardwareStore)*, *Go(Home, GroceryStore)*, *Go(Home, School)*, …
- If we choose to apply *Go(Home, HardwareStore)*, we will delete from the KB *At(Home)* and add *At(HardwareStore)*.
- The new proposition enables new actions, e.g. *Buy*

- Note that now there are a lot of possible operators to consider!

Regression planning

- Pick actions that satisfy (some of) the goal propositions.
- Make a new goal, containing the preconditions of these actions, as well as any unsolved goal propositions.
- Repeat until the goal set is satisfied by the start state.

Example: Supermarket domain

- In the goal state we have *At(Home)* $\wedge$ *Have(Milk)* $\wedge$ *Have(Bananas)* $\wedge$ *Have(Drill)*
- The action *Buy(Milk)* would allow us to achieve *Have(Milk)*.
- To apply this action we need to have the precondition *At(GroceryStore)*, so we add it to the set of propositions we want to achieve.
- Similarly, we want to achieve *At(HardwareStore)*.

Note that in this case, the order in which we try to achieve these propositions matters!

State-space planning

- **Progressive planners** reason from the start state, trying to find the operators that can be applied. (-> match preconditions)
  - Analogy: forward search!

- **Regression planners** reason from the goal state, trying to find the actions that will lead to the goal. (-> match effects)
  - Analogy: Backward search!

In both cases, the planners work with **sets of states**, instead of using individual states as in straightforward search.

Analysis of STRIPS planning

- STRIPS planning is **SOUND**.
  - Only legal plans will be found.
- STRIPS planning is **NOT COMPLETE**.
  - Once a subgoal ordering is selected, no backtracking is allowed.
- STRIPS planning is **NOT OPTIMAL**.
  - No guarantee of finding shortest possible plan.
- STRIPS planning is **EXPENSIVE**. Typically NP-hard or worse.

Some state-of-the-art classical planners

- SATPlan: Planning as satisfiability (Kautz & Selman, 1996)
- Heuristic-search planning (Bonet & Geffner, 1998)
- GraphPlan (Blum & Furst, 1995)
- Fast Forward planning (Hoffman & Nebel, 2001)
  - Use hill-climbing to get near a good solution, then best-first-search.
- Hierarchical task network planning
  - Decompose complex planning problem into a hierarchy of smaller planning tasks.
- Many more!

Planning as logic

- Planning is very much like searching over sets of states, but the problem representation is more structured.

	Search	*Planning*
States	Data structures	Logical sentences
Actions	Code	Preconditions/outcomes
Goal	Goal test	Logical sentence (conjunction)
Plan	Sequence from S_0	Constraints on actions

Key idea: describe states and actions in propositional logic and use forward/backward chaining to find a plan.

Planning as satisfiability: SatPlan

- Introduced by Kautz and Selman, 1990s, very successful method over the years.
- Take a description of a planning problem and generate all possible literals at all time slices.
- Generate a humongous SAT problem.
- Use a state-of-the-art SAT solver (e.g WalkSAT) to get a plan.
- Randomized SAT solvers can be used as well.

Analysis of SatPlan

- Optimality: Yes! (Assuming exact SAT solution.)
- Completeness: Yes!
- Complexity:
  - Clearly NP-hard (as it can be seen as SAT in finite-length plan case).
  - But actually worse (PSPACE) if we let plan duration vary.

Heuristic-search planning

- Don’t want domain-specific heuristics.
- Define heuristics based on planning problem itself.
  - Can we derive an **admissible** heuristic for search in planning domains?
  - Technique we've seen before: solve **relaxed** problem!

- Solving the problem becomes much easier in the relaxed case; number of steps to goal in this heuristic can be used as part of A* search
  - Historical note: A* search came from the same research project responsible for STRIPS and Shakey!

Two Heuristics

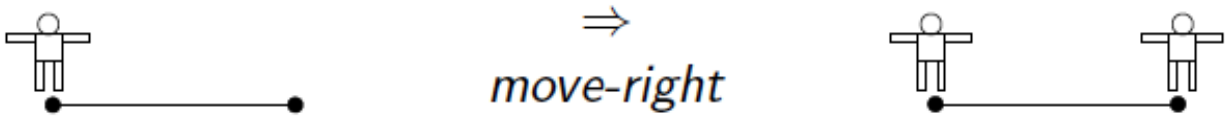
1. Ignore Preconditions

- Assume that we can also apply an operator

e.g., you can always move to point B, no matter where you are

2. Ignore Delete lists (in Effects)

- Assumes that when something is achieved, it stays achieved.



- Solve this relaxed version and use this as the heuristic.

Image from: *http://icaps09.uom.gr/tutorials/tut1.pdf*

Problems in the real world

- **Incomplete information**
  - Unknown predications, e.g., *Intact(Spare)*
  - Disjunctive effects, e.g. *Inflate(x)* causes *Inflated(x)* according to the KB, but in reality it actually causes *Inflated(x) v SlowHiss(x) v Burst(x) v BrokenPump v …*
- **Incorrect information**
  - Current state it incorrect, e.g., spare NOT intact.
  - Unanticipated outcomes (missing / incorrect postconditions) of operators, that lead to failure.
- **Qualification problem**
  - Can never finish listing all the required preconditions and possible conditional outcomes of actions.

The real world: Some solutions

- **Conditional (contingency) planning:**
  - Plans include observation actions which obtain information.
  - Sub-plans are created for each contingency (eash possible outcome of the observation actions).

  E.g. Check the tire. If it is intact, then we're ok, otherwise there are several possible solutions: Inflate, Call-CAA, …
  - Expensive because it plans for many unlikely cases.

- **Monitoring / Replanning:**
  - Assume normal states, outcomes.
  - Check progress during execution, replay if necessary.

  In general, some monitoring / replanning is highly useful.

Study up to here for midterm

How do we *deal with* uncertainty?

- **Implicit methods**
  - Ignore uncertainty as much as possible.
  - Build procedures that are robust to uncertainty.
  - This is the approach of the planning methods studied so far.
- **Explicit methods**
  - Build a **model** of the world that *describes the uncertainty* (about the system's state, dynamics, sensors, model).
  - Reason about the effect of actions given the model.

How do we *represent* uncertainty?

- What language should we use? What are the semantics of our representations?
- What queries can we answer with our representations? How do we answer them?
- How do we construct a representation? Do we need to ask an expert or can we learn it from data?

Why not use logic?

A purely logical approach has two problems:

1. Risks falsehood:
   "Check the tire => Tire is ok." (Noisy sensors, unexpected effects.)

2. Leads to conclusions that are too weak:
   "Check the tire after every other action" (Too slow!)

Methods for handling uncertainty

Logic: make assumptions unless contradicted by evidence.

- E.g. "Assume my car doesn't have a flat tire."

Issue: What assumptions are reasonable? How to handle contradictions?

Reasoning under uncertainty: consider all possible states.

- E.g. "Assume my car does or doesn't have a flat tire."

Issue: All possible states are equally likely. How do we draw conclusions?

Methods for handling uncertainty

Logic: make assumptions unless contradicted by evidence.

- E.g. "Assume my car doesn't have a flat tire."

Issue: What assumptions are reasonable? How to handle contradictions?

Reasoning under uncertainty: consider all possible states.

- E.g. "Assume my car does or doesn't have a flat tire."

Issue: All possible states are equally likely. How do we draw conclusions?

Probability: associate a probability of occurrence with facts.

- E.g. "Driving $->_{0.01}$ FlatTire", "FlatTire $->_{0.9}$ Lateness"

Let's explore this!

Bayesian Probability

- A well-known and well-understood framework for dealing with uncertainty.
- Has a clear semantics.
- Provides principled answers for:
  - Combining evidence.
  - Incorporating new evidence.
  - Performing predictive and diagnostic reasoning.
- Can be learned from data
- Intuitive to human experts

Probabilistic beliefs

- We use probabilities to describe the world and the existing uncertainties.
- **Beliefs** relate logical propositions to current state of knowledge.
- Beliefs are **subjective** assertions, given one's state of knowledge.

  e.g., P(FlatTire | driving on dry pavement) = 0.05

  - Different agents may hold different beliefs.

Probabilistic beliefs

- **Probabilities of propositions change** with **new evidence**.

  e.g. P(FlatTire | driving on dry pavement, broken glass on road) = 0.45

- This is analogous to logical entailment status:
  - KB $\vdash \alpha$ means that α is true given the KB, not necessarily true in general.
- **Prior (unconditional) beliefs** denote belief before the arrival of any new evidence.

Making decisions under uncertainty

Suppose I believe the following:

P (FlatTire | …) = 0.05

P (FlatTire | NewTire, …) = 0.02

P (FlatTire | NewTire, SlowDriving…) = 0.01

Which action should I choose?

Making decisions under uncertainty

Suppose I believe the following:

P (FlatTire | …) = 0.05

P (FlatTire | NewTire, …) = 0.02

P (FlatTire | NewTire, SlowDriving…) = 0.01

Which action should I choose?

- Depends on my *preferences* for replacing tires, spending time in garage, etc.
- **Utility theory** is used to represent and infer preferences.
- **Decision theory** = utility theory + probability theory.
- Let's start with probability theory.

What you should know

- Basic axioms of probability.
- Joint probabilities.
- Conditional probabilities.
- Chain rule and Bayes rule.
- Conditional independence.

Defining probabilistic models

- Define the world as a set of **random variables**:

  $\Omega = \{X_1, .., X_n\}$.
- The world is divided into a set of elementary, mutually exclusive **events** (also called **states**).

- A **probabilistic model** is an encoding of probabilistic information that allows us to compute the probability of any event in the world.

- A **joint probability distribution function** assigns non-negative weights to each event (such that the weights sum to 1).

Inference using joint distributions

E.g. Suppose *Happy* and *Rested* are two **random variables**:

	Happy = true	*Happy = false*
Rested = true	0.05	0.10
Rested = false	0.60	0.25

The **unconditional probability** of any proposition is computable as the sum of entries from the full joint distribution.

Inference using joint distributions

E.g. Suppose *Happy* and *Rested* are two **random variables**:

	Happy = true	*Happy = false*
Rested = true	0.05	0.10
Rested = false	0.60	0.25

The **unconditional probability** of any proposition is computable as the sum of entries from the full joint distribution.

P(Happy) = P(Happy, Rested) + P(Happy, ~Rested) = 0.65

Using Bayes rule for inference

- Let's say we want to form a hypothesis about the world, based on observable variables.
- **Bayes rule** tells us how to calculate the belief in a hypothesis *H*, given evidence ***e***:

$$P(H \mid e) = P(e \mid H)\, P(H) / P(e)$$

$P(H \mid e)$	is the **posterior probability**
$P(H)$	is the **prior probability**
$P(e \mid H)$	is the **likelihood**
$P(e)$	is a **normalizing constant**,

which can be computed as

$$P(e) = P(e|H)P(H) + P(e|\sim H)P(\sim H)$$

Example: Medical Diagnosis

- You go to the doctor complaining of having a fever (F).
- A doctor knows that bird flu (B) causes a fever 95% of the time.
- The doctor knows that if a person is selected randomly from the population, there is a 10^{-7} chance of the person having bird flu.
- In general, 1 in 100 people in the population suffer from fever.

What is the probability that you have bird flu?

Example: Medical Diagnosis (cont'd)

- What are the relevant facts?

  F = fever B = bird flu

- What are the probabilities of these facts?

  $P(F) = 0.01$ $P(B) = 10^{-7}$

- Anything else we know about the world?

  $P(F|B) = 0.95$

- Given these facts, we can answer the question using Bayes rule:

  $P(B|F) = P(F|B)\,P(B) / P(F)$

  $= 0.95 \times 10^{-7} / 0.01$

  $= 0.95 \times 10^{-5}$

Computing conditional probabilities

- Consider a world described by a set of variables X.
- Typically, we are interested in the **posterior joint distribution** of some **query variables** Y, given specific values e for some **evidence variables** E.
- Often, there are some hidden variables $Z = X - Y - E$
- If we know the joint probability distribution, we compute the answer by "summing out" the hidden variables:

$$P(Y, e) = \sum_z P(Y, e, z) \quad \text{<= "summing out"}$$

Problem: the joint distribution is too big to handle!

Example

- Consider medical diagnosis, where there are 100 different symptoms and test results that the doctor could consider.
- A patient comes in complaining of fever, cough and chest pains.
- The doctor wants to compute the probability of pneumonia:
  - Probability table has >= 2^{100} entries!
  - To compute probability of pneumonia, we have to sum out over 97 hidden variables (=2^{97} entries).

Independence of random variables

- Two random variables *X* and *Y* are **independent** if knowledge about *X* does not change the uncertainty about *Y* (and vice versa.)

$$P(x \mid y) = P(x) \qquad \forall x \in S_X,\ \forall y \in S_Y$$

$$P(y \mid x) = P(y) \qquad \forall x \in S_X,\ \forall y \in S_Y$$

Independence of random variables

- Two random variables *X* and *Y* are **independent** if knowledge about *X* does not change the uncertainty about *Y* (and vice versa.)

$$P(x \mid y) = P(x) \qquad \forall x \in S_X,\ \forall y \in S_Y$$

$$P(y \mid x) = P(y) \qquad \forall x \in S_X,\ \forall y \in S_Y$$

- If *n* Boolean variables are independent, the whole joint distribution can be computed:

$$P(x_1, \ldots, x_n) = \prod_i P(x_i)$$

- Only *n* numbers are needed to specify the joint, instead of 2^n.

Problem: Absolute independence is a very strong requirement!

Conditional independence

- Two variables *X* and *Y* are **conditionally independent** given *Z* if:

$$P(x \mid y, z) = P(x \mid z) \qquad \forall x, y, z$$

- This means that knowing the value of *Y* does not change the prediction about *X* if the value of *Z* is known.

This is much more common!

Example

- Consider a patient with three random variables:

  B (patient has bronchitis), ***F*** (patient has fever), ***C*** (patient has cough)

  The full joint distribution has 2^3-1 = 7 independent entries.

Example

- Consider a patient with three random variables:

  B (patient has bronchitis), ***F*** (patient has fever), ***C*** (patient has cough)

  The full joint distribution has 2^3-1 = 7 independent entries.

- If someone has bronchitis, we can assume that the probability of a cough does *not* depend on whether they have a fever:

  $$P(C \mid B, F) = P(C \mid B)$$

- The same independence holds if the patient does not have bronchitis: $P(C \mid \sim B, F) = P(C \mid \sim B)$

  therefore ***C*** and ***F*** are conditionally independent given *B*.

Example (cont'd)

- Full joint distribution can now be written as:

  $P(C, F, B) = P(C, F \mid B)\, P(B)$

  $= P(C \mid B)\, P(F \mid B)\, P(B)$

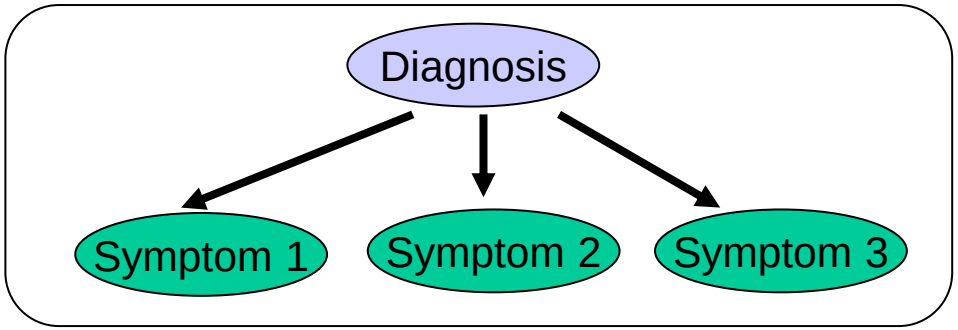
  I.e. 2+2+1 = 5 independent numbers.

Much more important savings happen if the system has lots of variables!

Naïve Bayes Model

- A common assumption in early diagnosis is that the symptoms are independent of each other given the disease.
- Let $s_1, .., s_n$ be the symptoms exhibited by a patient (e.g. fever, headache, etc.). Let D be the patient's disease.
- Using the Naive Bayes assumption:

$$P(D, s_1, \ldots, s_n) = P(D)\, P(s_1 \mid D) \ldots P(s_n \mid D)$$



Exercises

- How many parameters are there in the joint probability distribution, assuming all diagnoses and symptoms are binary:

$$P(D, s_1, \ldots, s_n)$$

- How many parameters are there assuming the Naïve Bayes assumption?

$$P(D, s_1, \ldots, s_n) = P(D)\, P(s_1 \mid D) \ldots P(s_n \mid D)$$

- What is the conditional probability that you would use to actually diagnose a patient, given their symptom?

Exercises

- How many parameters are there in the joint probability distribution, assuming all diagnoses and symptoms are binary:

$P(D, s_1, \ldots, s_n)$ $\quad 2^{n+1} - 1$

- How many parameters are there assuming the Naïve Bayes assumption?

$P(D, s_1, \ldots, s_n) = P(D)\, P(s_1 \mid D) \ldots P(s_n \mid D)$ $\quad 2n + 1$

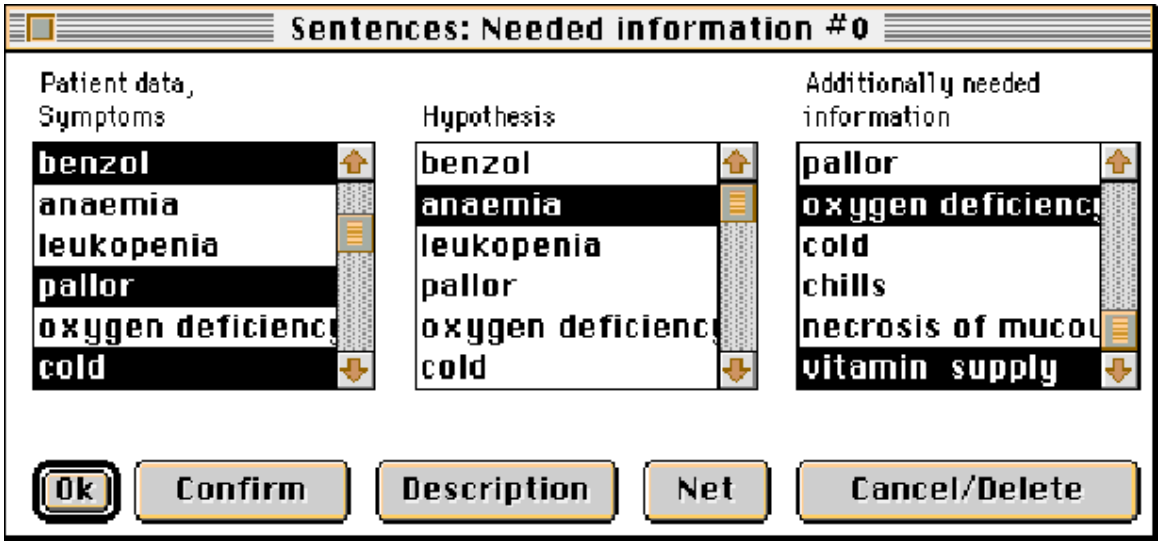
- What is the conditional probability that you would use to actually diagnose a patient, given their symptom?

$$P(D|s_1, \ldots, s_n)$$

Example AI system: Medical diagnosis

Pathfinder (D. Heckerman, Microsoft Research, 1992)

- Perception: symptoms, test results.
- Actions: suggest tests, make diagnosis.
- Reasoning: Bayesian inference, machine learning, Monte-Carlo simulation.



http://lls.informatik.uni-oldenburg.de/dokumente/1996/96_icls/icls_96_5.gif

Next Week's Lectures

- **Monday**:
  - Midterm review!
  - Send me any topic you'd like for me to cover again (by Friday, 11:59pm).
  - If not, I'll cancel lecture and turn it into extra office hours for the midterm.
- **Wednesday**:
  - Midterm in the evening!
  - No lecture during regular class time