COS402- Artificial Intelligence
Fall 2015

Lecture 24: AI Wrap-up

Outline

- **Review of the algorithms you have learned**
- **Information on the final exam**
- **Humans and Robots (RoboCup 1997--2015)**

Problems and applications

1	8 puzzle	2	Software verification
3	Route planning	4	Theorem proving
5	N-queen problem	6	Medical diagnosis
7	Logistic planning	8	Insurance policy
9	Face detection	10	Speech recognition
11	Optical character recognition	12	Weather forecast
13	Spam detection	14	Stock price prediction
15	Travelling salesman problem		

Problems and applications

1	8 puzzle	2	Software verification
3	Route planning	4	Theorem proving
5	N-queen problem	6	Medical diagnosis
7	Logistic planning	8	Insurance policy
9	Face detection	10	Speech recognition
11	Optical character recognition	12	Weather forecast
13	Spam detection	14	Stock price prediction
15	Travelling salesman problem		

Problems and applications

1	8 puzzle	2	Software verification
3	Route planning	4	Theorem proving
5	N-queen problem	6	Medical diagnosis
7	Logistic planning	8	Insurance policy
9	Face detection	10	Speech recognition
11	Optical character recognition	12	Weather forecast
13	Spam detection	14	Stock price prediction
15	Travelling salesman problem		

Search

Logic

Bayes Network

Supervised Learning

HMM

Search

Logic:SAT

Theme of problem solving in AI

- **Develop general algorithms that can be applied to a whole class of problems.**
- **Start with simpler problems**
- **Use knowledge to help model a problem and/or develop more efficient algorithms.**

Search techniques

- **Formulate/define a search problem**
  - **States (including initial state), actions, transition model, goal test, path cost**
- **Search approaches**
  - **tree search vs graph search**
- **Performance measures**
  - *completeness*
  - *optimality*
  - *Time complexity*
  - *Space complexity*

Search: Blind search

- **Formulate/define a search problem**
- **Search strategies/algorithms (tree search vs graph search)**
  - *Breadth First Search*
  - *Depth First Search*
  - *Depth Limited Search*
  - *Iterative Deepening Search*
  - *Bidirectional Search*

Search: Heuristic search

- **Best First Search (f(n): evaluation function)**
  - **Choose a node n with minimum f(n) from frontier**
- **Greedy Best-First Search**
  - **f(n) = h(n)**
  - **h(n) : An estimate of cost from n to goal**
- **A* Search**
  - **f(n) = g(n) + h(n)**
  - **g(n) : cost from initial state to n**

BFS, DFS and Uniform-cost Search

- **Breadth First Search**
  - **f(n) = depth of node n**
- **Depth First Search**
  - **f(n) = -(depth of node n)**
- **Uniform-cost Search**
  - **f(n) = g(n)**
  - **g(n) : cost from initial state to n**

A* and Heuristics

- **Heuristics**
  - **Admissible vs consistent**
- **Optimality of A***
  - **If h(n) is admissible, A* using tree search is optimal**
  - **If h(n) is consistent, A* using graph search is optimal**
- **Constructing heuristic**
  - **Relaxed versions of the original problem**
  - **Combine multiple heuristics**

Search in games

- **Games we are looking at**
  - **2-player game**
  - **Zero-sum game**
- **The Minimax algorithm**
- **Alpha-beta pruning**
- **The Minimax algorithm extends to multiplayer game**

Some points

- **The Minimax value of a node**
  - **The utility (for Max) of being in the corresponding state if both players play optimally from there to the end of the game.**
- **Alpha-beta pruning**
  - **Alpha: the value of the best choice we have found so far at any choice point along the path for MAX. (i.e. highest-value)**
  - **Beta: the value of the best choice we have found so far at any choice point along the path for MIN. (i.e. lowest value)**

Some points--more

- **Evaluation function**
  - **Needed when building/searching a complete game tree is impossible**
  - **An estimate of the utility of nodes at the cutoff level**
  - **Usually a functions of features of the state**
- **When to cut off**
  - **Go to fixed depth?**
  - **Iteratively increase depth until time runs out?**
  - **Other strategies?**

Propositional Logic

- **Syntax and Semantics**
- **Entailment**
- **Model checking**
- **Concepts needed for theorem proving**
  - **Logical equivalence**
  - **Validity**
  - **Satisfiability**

Satisfiability and Validity

- **A sentence is valid if it is true in all models.**
- **A sentence is satisfiable if it is true in some model.**
- **A sentence P is valid if and only if ¬P is unsatisfiable**
- **A valid sentence is always satisfiable**

Theorem proving

- **Logical equivalence, validity and satisfiability**
- **Inference rules**
  - **Modus Ponens:** $\dfrac{P \Rightarrow Q, P}{Q}$
  - **And elimination:** $\dfrac{P \wedge Q}{P}$
  - **Reverse And elimination:** $\dfrac{P\ ,\ Q}{P \wedge Q}$
  - **All equivalences**

Resolution algorithm

- **Resolution rule**
  - **Takes 2 clauses and produce a new clause containing all the literals of the two original clauses except the two complementary literals.**
- **Conjunction Normal Form(CNF) : A conjunction of clauses**
- **Resolution algorithm (show KB |= α by prove KB ∧ ¬α is unsatisfiable.)**
  - **Convert KB ∧ ¬α to CNF**
  - **Repeatedly apply resolution rule to add new clauses**
  - **Stops when**
    - **(1) Generating the empty clause (KB entails α) or**
    - **(2)no new clause can be added. (KB does not entail α)**

Some points

- **A clause is a disjunction of literals.**
- **A CNF is a conjunction of clauses.**
- **Resolution algorithm is both complete and sound.**
- **Theorem proving does not need to consult models.**
- **Every sentence can be written in CNF.**
- **KB |= α if and only if (KB => α) is valid. (Deduction Theorem)**
- **KB |= α if and only if KB ∧ ¬α is unsatisfiable.**

Practical methods of solving CNFs

- **Faster inference in special cases**
  - **Forward chaining**
  - **Backward chaining**
- **Algorithms based-on model checking**
  - **DPLL**
  - **WALKSAT**

Some points

- **A Horn clause has at most one positive literal.**
- **A definite clause has exactly one positive literal.**
- **DPLL does recursive exhaustive search of all models for the given CNF.**
- **WALKSAT uses random and greedy search to find a model that may satisfy the given CNF.**

Forward chaining

- **Initially set all symbols false**
- **Start with symbols that are true in KB**
- **When all premises of a horn clause are true, make its head true.**
- **Repeat until you can't do more.**

Backward chaining

- **Start at goal and work backwards**
- **Takes linear time.**

DPLL

- **Do recursive exhaustive search of all models**
- **Set P_1 = T**
- **Recursively try all settings of remaining symbols.**
- **If no model found**
  - **Set P_1 = F**
  - **Recursively try all settings of remaining symbols**

Additional tricks for DPLL

- **Early termination**
- **Pure symbols**
- **Unit clauses**
- **Component analysis**
- **And more …**

WALKSAT

- **Set all symbols to T/F randomly**
- **Repeat MAX times**
  - **If all clauses are satisfied, then return model**
  - **Choose an unsatisfied clause randomly**
  - **Flip a coin**
    - **If head**
      - **flip a symbol in the clause that maximizes # if satisfied clauses**
    - **Else**
      - **flip a symbol selected randomly from the clause.**

DPLL and WALKSAT

- **DPLL**
  - **Complete and sound**
  - **Determine KB |= α**
  - **Check satisfiability of a cnf + find a model if it is satisfiable**
- **WALKSAT**
  - **Sound, but not complete**
  - **Mostly used for finding a model when a cnf is satisfiable**

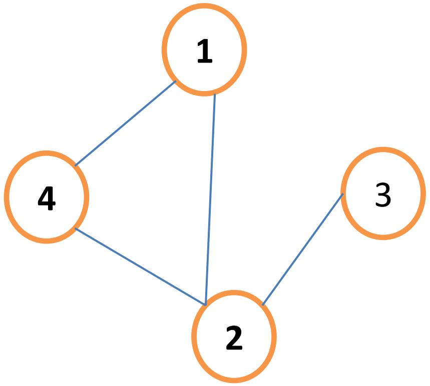
Applications of solving CNF

- **SAT is used in problems other than logical inference**
  - **N-queen problem**
  - **3-coloring graph**
  - **Hamiltonian path**
  - **Planning**
  - **Jigsaw puzzle**
  - **Sudoku**

  …

Reduce 3-coloring graph to SAT

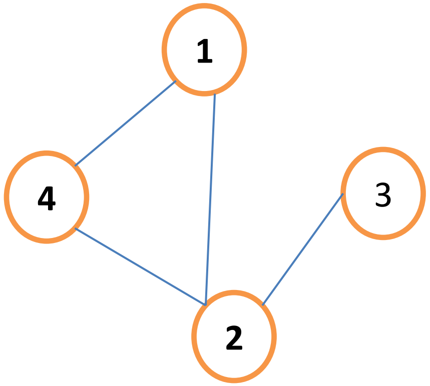
- **Define Symbols:**
  - **P_{ij} : node i is colored in color j**
  - **i = 1,2,3 or 4**
  - **j = r, g or b**



- **Express facts/rules in clauses**
  1. **Each node gets one color**
  2. **Two nodes sharing a common edge can't be colored the same**

Reduce 3-coloring graph to SAT

1. Each node gets one color



(1) Each node gets at least one color

$P_{1r} \vee P_{1g} \vee P_{1b}$

$P_{2r} \vee P_{2g} \vee P_{2b}$

$P_{3r} \vee P_{3g} \vee P_{3b}$

$P_{4r} \vee P_{4g} \vee P_{4b}$

(2) Each node gets only one color

$(\sim P_{1r} \vee \sim P_{1g}) \wedge (\sim P_{1r} \vee \sim P_{1b}) \wedge (\sim P_{1g} \vee \sim P_{1b})$

$(\sim P_{2r} \vee \sim P_{2g}) \wedge (\sim P_{2r} \vee \sim P_{2b}) \wedge (\sim P_{2g} \vee \sim P_{2b})$

$(\sim P_{3r} \vee \sim P_{3g}) \wedge (\sim P_{3r} \vee \sim P_{3b}) \wedge (\sim P_{3g} \vee \sim P_{3b})$

$(\sim P_{4r} \vee \sim P_{4g}) \wedge (\sim P_{4r} \vee \sim P_{4b}) \wedge (\sim P_{4g} \vee \sim P_{4b})$

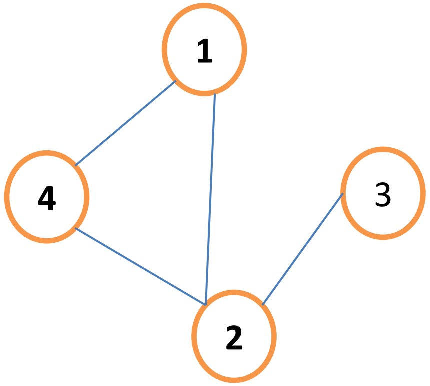
Reduce 3-coloring graph to SAT(cnt'd)

2. Two nodes sharing a common edge can't be colored the same

- **For edge 1-4**

  $(\sim P_{1r} \vee \sim P_{4r}) \wedge (\sim P_{1g} \vee \sim P_{4g}) \wedge (\sim P_{1b} \vee \sim P_{4b})$

- **For edge 2-4**
  - **$(\sim P_{2r} \vee \sim P_{4r}) \wedge (\sim P_{2g} \vee \sim P_{4g}) \wedge (\sim P_{2b} \vee \sim P_{4b})$**
- **For edge 1-2**
  - **$(\sim P_{1r} \vee \sim P_{2r}) \wedge (\sim P_{1g} \vee \sim P_{2g}) \wedge (\sim P_{1b} \vee \sim P_{2b})$**
- **For dege 2-3**
  - **$(\sim P_{2r} \vee \sim P_{3r}) \wedge (\sim P_{2g} \vee \sim P_{3g}) \wedge (\sim P_{2b} \vee \sim P_{3b})$**



---Put all clauses in a cnf and pass to a sat-solver.

---A model for the constructed cnf is a solution to the original problem.

---Legal coloring is guaranteed by the rules in 1 and 2.

Bayesian Networks

- **Logical inference and probabilistic inference**
- **Independence and conditional independence**
- **Bayes Nets**
  - **Semantics of Bayes Nets**
  - **How to construct a Bayes net**
  - **Conditional Independence in Bayes nets**
- **Variable elimination algorithm**
- **Naïve Bayes**

Logical inference vs. probabilistic inference

- Problem: KB |= α ?
- Model checking can determine entailment
-

P1	P2	P3	KB	α
T	T	T		
T	T	F	T	?
…	…			
F	F	T	T	?
F	F	F	T	?

- Is M(KB) a subset of M(α)?
- # of models: 2^n, n=3 here.

- Problem: P(X,Y)=? Or P(X|Y)=?
- Full joint probability distribution can be used to answer any query.

X	Y	Z	P(X,Y,Z)
x_1	y_1	z_1	0.3
x_1	x_1	Z_2	0.25
…	…		
X_h	y_m	z_{k-1}	0.1
x_h	y_m	Z_k	0.05

- # of parameters: hmk > 2^n
- How to answer the query?

Inference given full joint probability distribution

- **Joint probability**
  - P(x,y) = $\sum_z P(x, y, z)$ (Marginalization)
- **Conditional probability**
  - P(x|y) = $\frac{P(x,y\}}{P(y)} = \frac{\sum_z P(x,y,z)}{\sum_{x,z} P(x,y,z)}$ (definition + marginalization)
  - Or P(x|y) = $\alpha \sum_z P(x, y, z)$ (normalization)
    - $\alpha = \frac{1}{\sum_{x,z} P(x,y,z)}$
- **Time and space: $O(2^n)$**

Independence and conditional independence

- **Independence of two events**
  - **Events a and b are independent if knowing b tells us nothing about a**
  - **$P(a|b) = P(a)$ or $P(a|b) = P(a)P(b)$**
- **Independence of two random variables**
  - **Random variable X and Y are independent if for all x,y, $P(X=x, Y=y) = P(X=x)P(Y=y)$**
  - **Shorthand: $P(X,Y)=P(X)P(Y)$**
- **Conditional independence**
  - **X and Y are conditionally independent given Z if $P(X,Y|Z)=P(X|Z)P(Y|Z)$**

Bayesian Network/Bayes Net (1)

- **Semantics**
  - Nodes are random variables
  - Edges are directed. Edge X --> Y indicates x has a direct influence on Y
  - There is no cycles
  - Each node is associated with a conditional probability distribution: P(x|Parents(x))
- **How to construct a Bayes Net?**
  - Topology comes from human expert
  - Conditional probabilities: learned from data

Bayesian Network/Bayes Net(2)

- **Conditional independence in Bayes Nets**
  - **A node is conditionally independent of non-descendants given its parents.**
  - **A node conditionally independent of all other nodes given its Markov blanket.**
    - **A Markov blanked of a node is composed of its parents, its children, and its children's other parents.**

Bayesian Network/Bayes Net(3)

- **Bayes nets represent the full joint probability**

P(X_1, X_2 , … , X_n**) =** $\prod_{\boldsymbol{i}}^{\boldsymbol{n}} \boldsymbol{P}(X_i|\boldsymbol{Parents}(X_i))$

- **Exact inference (P(b|j,m) = ? example in the textbook)**

P(b,|j,m) = α $\mathbf{P(b, j, m)} = \boldsymbol{\alpha} \sum_{\boldsymbol{e,a}} \boldsymbol{P(b, j, m, e, a)}$

= α $\sum_{\boldsymbol{e,a}} \boldsymbol{P(b)P(e)P(a|e, b)P(j|a)P(m|a)}$

= α $\mathbf{P(b)} \sum_{\boldsymbol{e}} \boldsymbol{P(e)} \sum_{\boldsymbol{a}} \boldsymbol{P(a|e.b)P\,(j|a)P(m|a)}$

Variable Elimination Algorithm (1)

- **Variable elimination algorithm**
  - **P(b,|j,m) =** $\boldsymbol{\alpha}\ \mathbf{P(b)} \sum_e \boldsymbol{P(e)} \sum_a \boldsymbol{P(a|e.b)P\ (j|a)P(m|a)}$
    - **g_1(e,b) =** $\sum_a \boldsymbol{P(a|e.b)P\ (j|a)P(m|a)}$
    - **g_2(b) =** $\sum_e \boldsymbol{P(e)}$**g_1(e,b)**
    - **g_3(b) = P(b) g_2(b)**
  - **Define and evaluate function for each summation from right to left.**
  - **Evaluate once and store the values to be used later.**
  - **Normalize.**

Variable elimination algorithm (2)

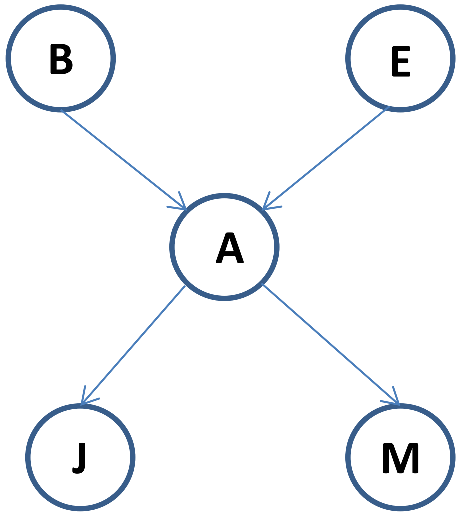
- **Time and space:**
  - linear in terms of the size of Bayes net for singly connected networks.
  - Exponential for multiply connected networks.
- **Singly-connected networks vs Multiply-connected networks**
  - In singly-connected networks, also called polytrees, there is at most one undirected path between any two nodes.
  - In mutliply-connected networks, there could be 2 or more undirected paths between 2 nodes.

Naïve Bayes

- **Naïve Bayes:**
  - **A special case of Bayes net: one parent node and the rest are its children.**
  - **Random variables: One cause and multiple effects.**
  - **Assume that all effects are conditionally independent given the cause.**
  - **Very tractable.**
  - **Can be used for classification: Naïve Bayes classifier.**

Approximate inference in BN

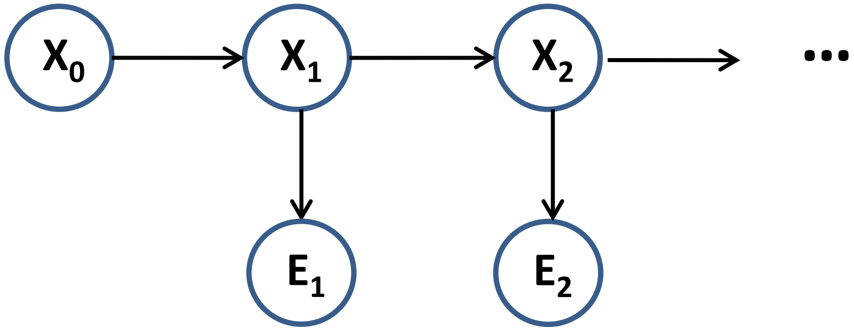
- **Direct sampling**
  - **Prior sample algorithm: for joint probability**
  - **Rejection sampling: for conditional probability**
  - **Likelihood sampling: for conditional probability**
- **How to sample the variables?**
- **P(J=t, M=t) = ?**
- **P(J=t, M=t|B=t) = ?**
- **P(J=t|E=t)= ?**



Approximate inference in BN

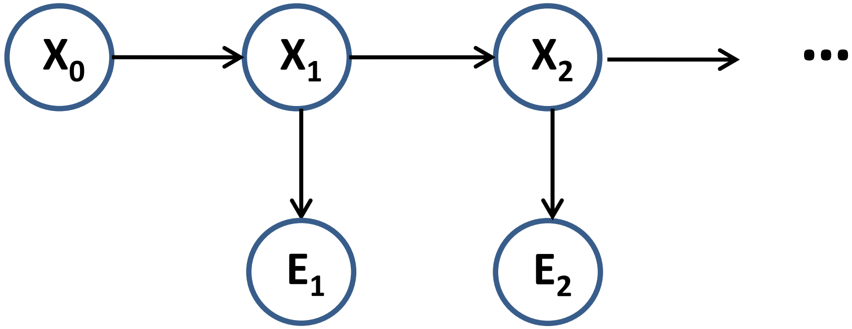
- **MCMC**
  - **A state in MCMC specifies a value for every variable in the BN.**
  - **Initialize the state with random values for all the non-evidence variable, and copy the evidence for the evidence variables**
  - **Repeat N times (long enough to assume convergence: stationary distribution.)**
    - **Randomly choose a non-evidence variable z, set the value of z by sampling from P(z|mb(z))**
  - **Estimate P(X|e)**

Hidden Markov Models



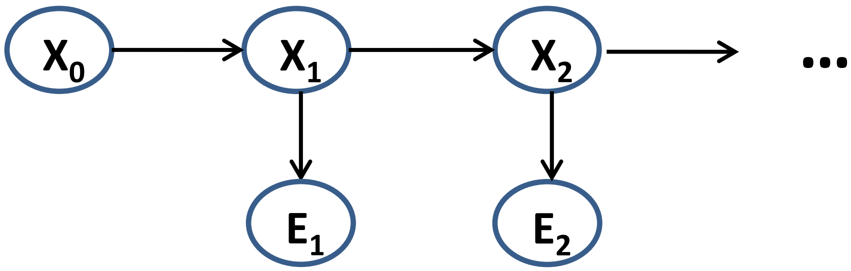
- **X_t: random variable**
  - **State at time t**
  - **Discrete, finite number of values**
  - **Single variable representing a single state, can be decomposed into several variables**
  - **Hidden, invisible**
- **E_t: random variable, evidence at time t**

Hidden Markov Models(parameters)



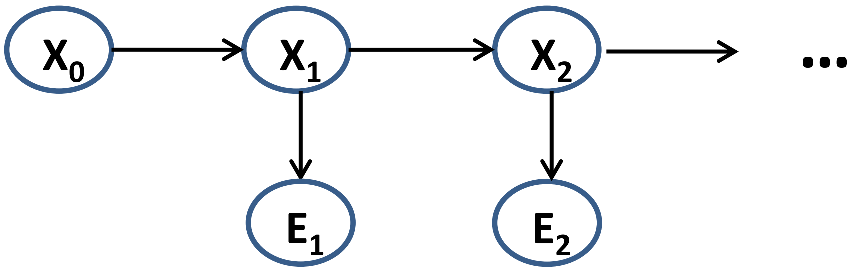
- **$P(X_0)$: the initial state model**
- **$P(X_t|X_{t-1})$: Transition model (usually assume stationary, same for all t)**
- **$P(E_t|X_t)$: sensor/observation model (usually assume stationary.)**

Hidden Markov Models (2 Markov assumptions)



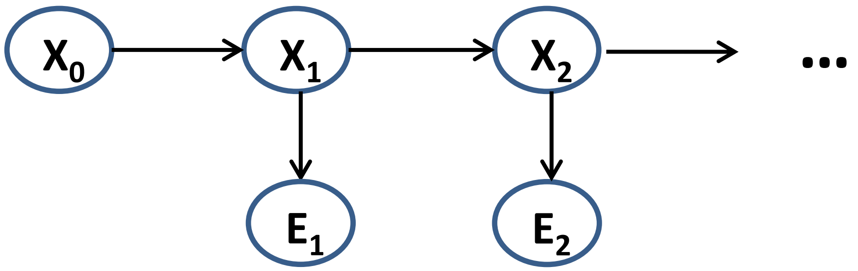
- $P(X_{t+1}|X_{0:t}) = P(X_{t+1}|X_t)$
  - The future is independent of the past given the present.
- $P(E_t|X_{0:t,}E_{1:t-1}) = P(E_t|X_t)$
  - Current evidence only depends on current state.
- **Note:**
  - given the 2 Markov assumptions, you can draw the Bayes Net; Given the Bayes net, the 2 Markov assumptions are implied.
  - HMMs are special cases of BNs, what is the full joint probability? $P(X_{0:t,}E_{1:t}) = ?$

Hidden Markov Models (4 basic tasks)



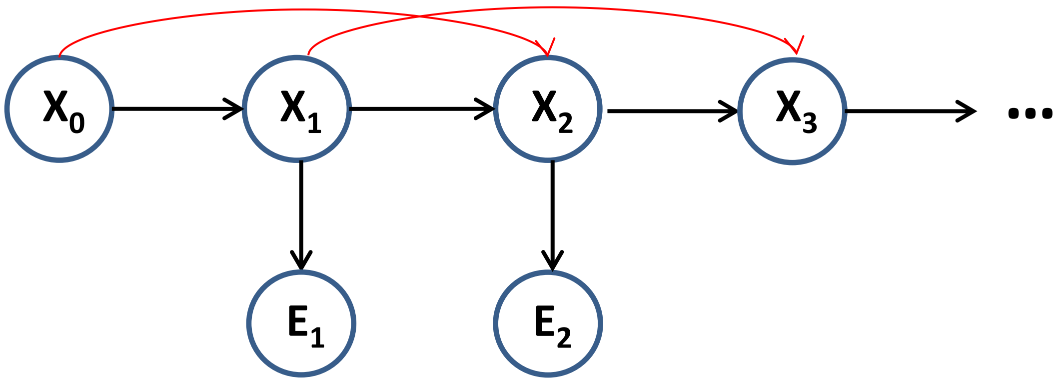
- **Filtering: Where am I now?**
  - $P(X_{t+1} | e_{1:t+1}) = ?$
- **Prediction : where will I be in k steps?**
  - $P(X_{t+k} | e_{1:t}) = ?$
- **Smoothing: Where was I in the past?**
  - $P(X_k | e_{1:t}) = ?$ $(k<t)$
- **Finding the most likely sequence**
  - $\text{Max } P(X_{0:t} | e_{1:t}) = ?$

Hidden Markov Models (4 basic tasks)



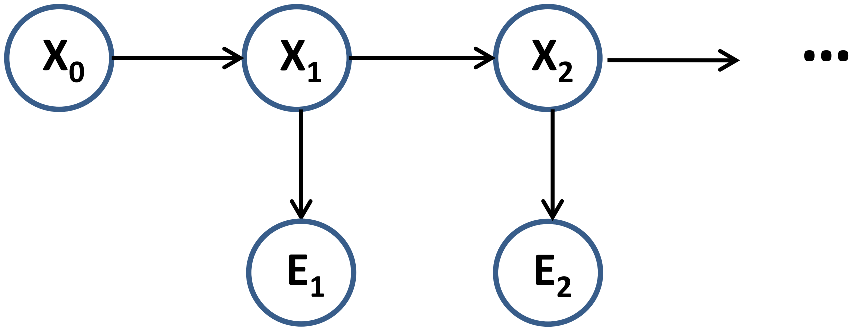
- **Filtering:** $P(X_{t+1} | e_{1:t+1}) = ?$
- **Prediction :** $P(X_{t+k} | e_{1:t}) = ?$
- **Smoothing:** $P(X_k | e_{1:t}) = ?$ $(k<t)$
- **Finding the most likely sequence:** Max $P(X_{0:t} | e_{1:t}) = ?$
- **Question?**
  - **Time complexity for the 4 basic tasks? $O(t \bullet (\#states)^2)$**
  - **Can we do other inference in HMM? $P(E_2 | X_1, X_3) = ?$, time complexity?**

Kth order Hidden Markov Models



- **First order HMM**
  - **$P(X_{t+1}|X_{0:t}) = P(X_{t+1}|X_t)$**
- **Second order HMM**
  - **$P(X_{t+1}|X_{0:t}) = P(X_{t+1}|X_t,X_{t-1})$**
- **Kth order HMM?**
  - **The future is dependent on the last k states.**

Kalman Filters



- **$P(X_0)$: Gaussian distribution**
- **$P(X_{t+1}|X_t)$: Linear Gaussian distribution**
  - **The next state X_{t+1} is a linear function of the current state X_t, plus some Gaussian noise.**
- **$P(E_t|X_t)$: Linear Gaussian distribution**
- **Filtering: $P(X_{t+1}|e_{1:t+1})$ is also a Gaussian distribution.**

Particle Filtering—When to use it?

- **In DBNs where state variables are continuous, but both the initial state distribution and transitional model are not Gaussian.**
- **In DBNs where state variables are discrete, but the state space is huge.**
- **HMMs with huge state space.**

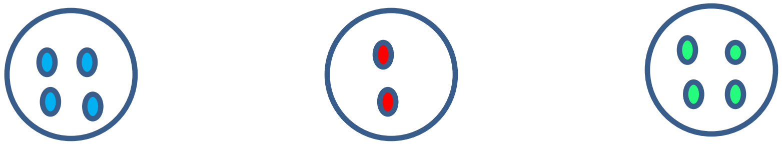
Particle Filtering—How does it work?

- **First, a population of N samples is created by sampling from the prior distribution $P(X_0)$.**
- **Repeat the update cycle for t= 0,1,…**
  - **1. each sample is propagated forward by sampling the next state value X_{t+1} based on the transitional model $P(X_{t+1} | x_t)$.**
  - **2. each sample is weighted by the likelihood it assigns to the new evidence. $P(e_{t+1} | x_{t+1})$**
  - **3. Resample to generate a new population of N samples: The probability that a sample is selected is proportional to its weight. The new samples are un-weighted.**

Particle Filtering—Example

$P(X_0)$=(0.4, 0.2, 0.4) , e={T,F}, , x={A,B,C}, N=10

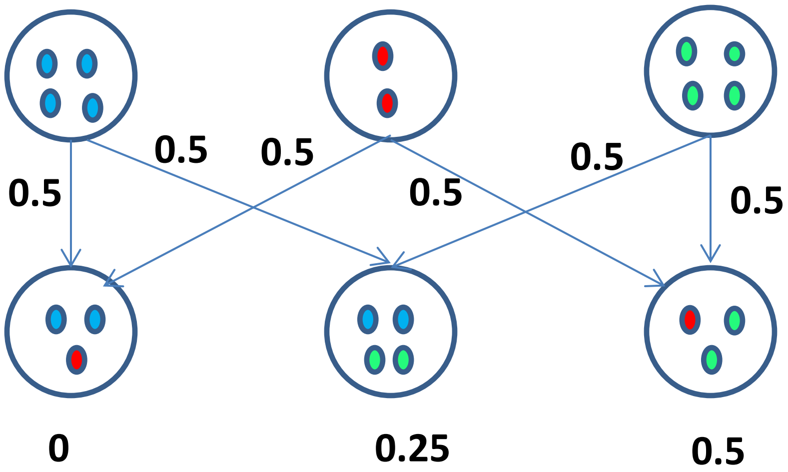
- **t=0, $P(X_0)$**



Particle Filtering—Example

$P(X_0)=(0.4, 0.2, 0.4)$, e={T,F}, , x={A,B,C}, N=10

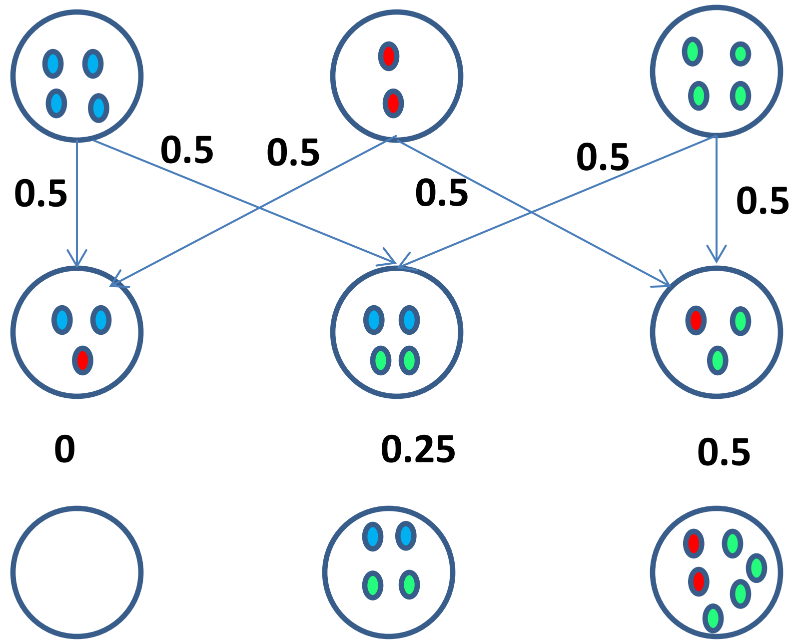
- **t=0, $P(X_0)$**
- **t=1,**
  - **$P(X_1 | x_0)$**
  - **e_1= T**
  - **$P(e_1 | x_1)$**



Particle Filtering—Example

$P(X_0)$=(0.4, 0.2, 0.4) , e={T,F}, , x={A,B,C}, N=10

- **t = 0**
  - **$P(X_0)$**
- **t = 1**
  - **$P(X_1 | x_0)$**
  - **e_1= T**
  - **$P(e_1 | x_1)$**
  - **$P(X_1|e_1)$=(0. 0.4, 0.6)**
- **t = 2 …**



Particle Filtering—Demo?

- http://robots.stanford.edu/movies/sca80a0.avi
- **A robot is wandering around in some cluster of rooms**
- **Modeled as HMM**
  - **States: locations**
  - **Observations: sonar readings**
  - **Task: Determining current state**
  - **Particle filtering: the green dot is the robot's actual location; the little red dots are the particles(samples.)**

Decision theory: Utility and expected value

- **Expected value (expectation) of a discrete random variable**
  - **Weighted average of all possible values**
  - **E[X] =** $\sum_x \boldsymbol{P(X = x) * x}$
- **Expected value of a discrete random variable**
  - **Replace the sum with an integral and the probabilities with probability densities.**
- **Conditional expectation**
  - **E[X|y=a] =** $\sum_x \boldsymbol{P(X = x|y = a) * x}$
- **Expectation of a real-valued function**
  - **E[f(x)] =** $\sum_x \boldsymbol{P(X = x) * f(x)}$

Decision theory: Utility and expected value

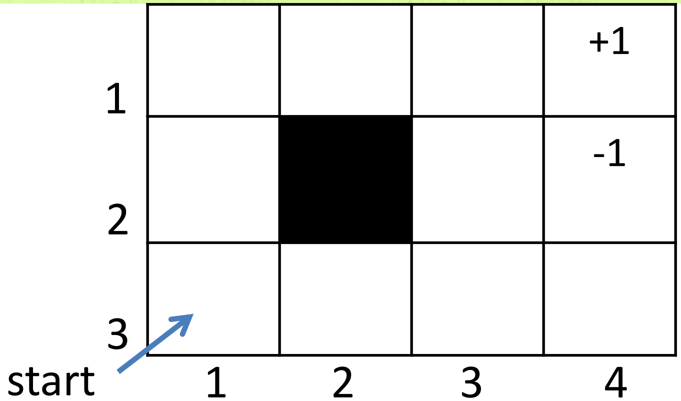
- **Linearity of expectations**
  - **(1) $E[X + c] = E[X] + c$**
  - **(2) $E[c * X] = c * E[X]$**
  - **(3) $E[X + Y] = E[X] + E[Y]$**
  - **Note: X and Y do not need to be independent.**
- **Examples:**
  - **E[X] = ? If X is the result of throwing a die.**
  - **E[X] = ? If X is the number of heads when throw two fair coins.**

Decision theory: MDP

- **General principle:**
  - **Assign utilities to states**
  - **Take actions that yields highest expected utility**
  - **Rational decision vs. human decision**
- **Simple decision vs complex decision**
  - **Simple decision: make a single decision, achieve short-term goal.**
  - **Complex decision: make a sequence of decisions, achieve long-term goal.**
    - **We will look at problems of making complex decisions**
    - **Markov assumption: The next state only depends on current state and action.**

MDP: Example

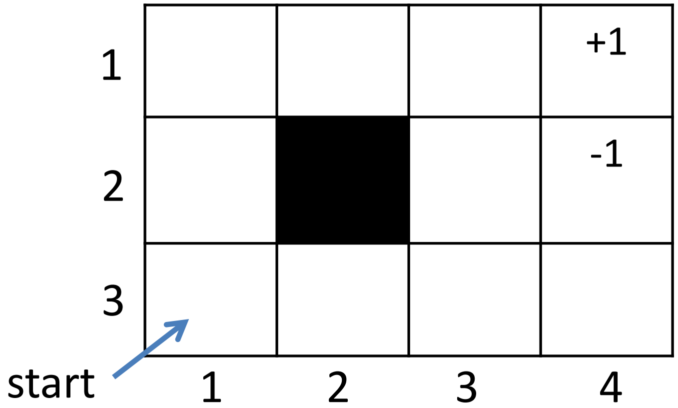
- **A robot in a grid**



- **MDP:**
  - **Initial state and states: locations/squares,**
  - **Actions: can move in 4 directions: up, down, left and right**
    - **No available actions at terminal states.**
  - **Transition model: P(s'|s, a)**
    - **80% of time moves in desired direction; 20% of time moves at right angle to the desired direction; no movement if bumps to a wall/barrier.**
  - **Rewards: +1 at [1,4], -1 at [2,4], and -0.04 elsewhere**
  - **Solution?**

MDP: Example

- **A robot in a grid**



- **MDP:**
  - **Initial state and states: fully observable**
  - **Actions:**
  - **Transition model: P(s'|s,a)**
    - **Markov assumption: The next state only depends on current state and action.**
  - **Rewards: R(s), additive**
  - **Solution: A policy maps from states to actions. An optimal policy yields the highest expected utility/sum of rewards.**

MDP: More Examples

- **Driving cars**
- **Controlling elevators**
  - **states: locations of the elevator, buttons pushed**
  - **Actions: send the elevator to particular floor**
  - **Rewards: measure of how long people wait**
- **Game playing(backgammon)**
- **Searching the web**
  - **states: urls**
  - **Actions: choose a link to expand**
  - **Rewards: find what is looking for**

MDP: More Examples

- **Animals deciding how to act/live**
  - **Must figure out what to do to get food, get mate, avoid predators, etc.**
  - **Cat and mouse in P5.**
    - **states:**
    - **Actions:**
    - **Rewards**:

Optimal policies and the utilities of states

- $U^{\pi}(s)$: **The expected utility obtained by executing π staring in s.**
  - $U^{\pi}(s) = E[\sum_{t=0}^{\infty} r^t R(S_t)]$
- **π^* : an optimal policy**
  - $\Pi^* = \underset{\pi}{argmax}\, U^{\pi}(s)$
- **π^* is independent of the starting state**
  - **When using discounted utilities with no fixed time limit.**
- $U(s) = U^{\pi^*}(s)$
  - **The true utility of a state is the expected sum of discounted rewards if an agent executes an optimal policy.**

Optimal policies and the utilities of states

- **π^* : an optimal policy**
  - **$\Pi^*(s) = \underset{a}{\arg\max} \sum_{s'} P(s'|s,a)\, U(S')$**
  - **Choose an action that maximizes the expected utility of the subsequent state.**
- **How to calculate U(s)?**

Value iteration: Does it work?

- **A contraction is a function of one argument. When applied to two different inputs in turn, the output values are getting “closer together”.**
  - **A contraction has one fixed point.**
  - **Ex. “divided by 2” is a contraction. The fixed point is 0.**
- **Bellmen update is a contraction. Its fixed point it the vector/point of the true utilities of the states.**
- **The estimate of utility at each iteration is getting closer to the true utility.**

Policy iteration: Algorithm

- **Start with any policy Π_0,**
- **For i = 0,1,2, …**
  - **Evaluate: compute $U^{\Pi_i}(s)$**
  - **Greedify: $\Pi_{i+1}(s) = \underset{a}{arg\max} \sum_{s'} P(s'|s,a) U^{\Pi_i}(S')$**
  - **Stop when $\Pi_{i+1} = \Pi_i$.**

Policy iteration: how to evaluate **Π** ?

- **Iterative approach – simplified value iteration.**
  - **Like value iteration, except now action at state S is fixed to be Π(S).**
  - $\mathbf{U_{i+1}^{\Pi}}(s) = \mathbf{R}(s) + r.\sum_{s'} \mathbf{P(s'|\Pi(s),a)}\, U_i^{\Pi}(S')$
- **Direct approach.**
  - $\mathbf{U^{\Pi}}(s) = \mathbf{R}(s) + r.\sum_{s'} \mathbf{P(s'|\Pi(s),a)}\, U^{\Pi}(S')$
  - **A system of linear equations, can be solved directly in $O(n^3)$.**
  - **Efficient for small state spaces.**

Policy iteration: why does it work ?

- **Can prove (Policy improvement theorem)**
  - **$U^{\Pi_{i+1}}(s) \geq U^{\Pi_i}(s)$, with strict inequality for some s unless $\Pi_i = \Pi^*$**
- **Means policies getting better and better Π_{i+1}**
  - **Will never visit same policy Π twice**
  - **Will only terminate when reach Π***
- **#iterations <= #policies**
  - **In practice, no case found where more than O(n) iterations are needed.**
  - **Open question: does policy iteration converge in O(n)? (n is the number of that states in the MDP)**

Machine Learning

- Supervised learning
  - Given a train set of N example input-output pairs, (x_i, y_i), discover a function h(called a hypothesis) that approximates the true function f, where $f(x_i) = y_i$.
- The theory of Learning
  - A PAC Learning algorithm: any learning algorithm that returns hypotheses that are probably approximately correct.
  - Provides bounds on the performance of learning algorithms.
  - $N \geq \frac{1}{\varepsilon}(\ln \frac{1}{\delta} + \ln |H|)$, a learning algorithm returns a hypothesis that is consistent with N examples, then with probability at least 1- δ , it has error at most ε.

Machine Learning Algorithms

- **Decision Trees**
- **AdaBoost**
- **Neural Networks**
- **Support Vector Machines**
- Naïve Bayes
- Nearest neighbors
- Random forest
- Voted perceptron algorithm

Support Vector Machines

- **SVMs construct a maximum margin separator – a linear decision boundary(hyperplane) with the largest possible distance to closest example points.**
- **A hyperplane is one dimension less than the input space and splits the space into two half-spaces.**
- **Support vectors: all points that are closest to the separating hyperplane.**
- **The separating hyperplane is a linear combination of all the support vectors.**

Lagrange multipliers with inequality constraints

- Minimize $\frac{1}{2}||w||^2$, st. $y_i(wx_i+b)-1 \geq 0$ for all i
- The Lagrangian is

$$L = \frac{1}{2}||w||^2 - \sum_i \alpha_i (y_i(wx_i+b)-1)$$

Can find solutions when $\alpha_i (y_i(wx_i+b)-1)=0$. (*Karush-Kuhn-Tucker* conditions)

- Solution: $W = \sum_i \alpha_i y_i x_i$ ($\alpha_i > 0$, if x_i is a support vector)

(Reference: http://mat.gsia.cmu.edu/classes/QUANT/NOTES/chap4/node6.html)

Reinforcement Learning

- **Learn how to behave through experience (rewards)**
- **Learning in MDPs**
  - **Model-based methods**
    - **ADP (Adaptive dynamic programming)**
  - **Model-free methods**
    - **TD learning (temporal-difference learning)**
      - **Adjusting the utility estimate with the difference between the utilities in successive states.**
    - **Q-Learning: learns an action-utility representation instead of learning utilities.**

Final exam

- **When:** 1:30pm — 4:00pm, Friday, Jan 15.
- **Where:** McCosh Hall 10
- **What: materials covered in class and in the assigned reading**
- **What to bring: (**The exam will be **closed book**.**)**
  - **may bring a one-page "cheat-sheet“** consisting of a single, ordinary 8.5"x11" blank sheet of paper with whatever notes you wish written upon it. You may write on both the front and the back.
  - **bring a calculator** However, you may only use the basic math functions on the calculator
  - **You may not use your cell phone or similar device as a calculator.**

Final exam : format (1)

- **A: True/false questions:**
  - **Ex. Policy iteration is guaranteed to terminate and find an optimal policy. (True/False)**
- **B: Modified True/false questions:**
  - **(write "correct" if the statement is correct as is, or cross the part that is underlined and write in the correct word or phrase)**
  - **Ex. The graph-search version of A will be optimal if an <u>admissible</u> heuristic function is used.**

Final exam : format (2)

- **C: Multiple choice questions (Circle all right answers)**
  - Which of the following are used in typical chess programs such as Deep Blue?
    - (a) alpha-beta pruning
    - (b) MCMC
    - (c) forward chaining
    - (d) genetic algorithms
    - (e) evaluation functions
- **D: problems: similar to problems in written exercises.**
  - **To obtain full credit, be sure to show your work, and justify your answers.**

Humans vs. Robots

- RoboCup: "Robot Soccer World Cup“ (1997)
  - **https://www.youtube.com/watch?v=u4iN-DtPyK8 (2005)**
  - **https://www.youtube.com/watch?v=4wMSiKHPKX4 (2010)**
  - **https://www.youtube.com/watch?v=iNLcGqbhGcc (2015)**
- Things that are easy for humans are difficult for robots.
- AI is not about building robots to do what humans do. Rather it should aim to help humans perform specific tasks.

Thank you!