## Lecture 17

# Policy Learning and Markov Decision Processes

**Tuesday, October 26, 1999**

**William H. Hsu**
**Department of Computing and Information Sciences, KSU**
http://www.cis.ksu.edu/~bhsu

Readings:
Chapter 17, Russell and Norvig
Sections 13.1-13.2, Mitchell



## Lecture Outline

- Readings: Chapter 17, Russell and Norvig; Sections 13.1-13.2, Mitchell
- Suggested Exercises: 17.2, Russell and Norvig; 13.1, Mitchell
- This Week's Paper Review: Temporal Differences [Sutton 1988]
- Making Decisions in Uncertain Environments
  - Problem definition and framework (MDPs)
  - Performance element: computing optimal policies given *stepwise reward*
    - Value iteration
    - Policy iteration
  - Decision-theoretic agent design
    - Decision cycle
    - Kalman filtering
    - Sensor fusion *aka* data fusion
  - Dynamic Bayesian networks (DBNs) and dynamic decision networks (DDNs)
- Learning Problem: Acquiring Decision Models from Rewards
- Next Lecture: Reinforcement Learning



## In-Class Exercise: Elicitation of Numerical Estimates [1]

- Almanac Game [Heckerman and Geiger, 1994; Russell and Norvig, 1995]
  - Used by decision analysts to calibrate numerical estimates
  - Numerical estimates: include subjective probabilities, other forms of knowledge
- Question Set 1 (Read Out Your Answers)

| | |
|---|---|
| – Number of passengers who flew between NYC and LA in 1989 | 3 million |
| – Population of Warsaw in 1992 | 1.6 million |
| – Year in which Coronado discovered the Mississippi River | 1541 |
| – Number of votes received by Carter in the 1976 presidential election | 41 million |
| – Number of newspapers in the U.S. in 1990 | 1611 |
| – Height of Hoover Dam in feet | 221 |
| – Number of eggs produced in Oregon in 1985 | 649 million |
| – Number of Buddhists in the world in 1992 | 295 million |
| – Number of deaths due to AIDS in the U.S. in 1981 | 132 |
| – Number of U.S. patents granted in 1901 | 25546 |



## In-Class Exercise: Elicitation of Numerical Estimates [2]

- Calibration of Numerical Estimates
  - Try to revise your bounds based on results from first question set
  - Assess your own penalty for having too wide a CI versus guessing low, high
- Question Set 2 (Write Down Your Answers)

| | |
|---|---|
| – Year of birth of Zsa Zsa Gabor | 1917 |
| – Maximum distance from Mars to the sun in miles | 155 million |
| – Value in dollars of exports of wheat from the U.S. in 1992 | 4.5 billion |
| – Tons handled by the port of Honolulu in 1991 | 11 million |
| – Annual salary in dollars of the governor of California in 1993 | 120000 |
| – Population of San Diego in 1990 | 1.1 million |
| – Year in which Roger Williams founded Providence, RI | 1636 |
| – Height of Mt. Kilimanjaro in feet | 19340 |
| – Length of the Brooklyn Bridge in feet | 1595 |
| – Number of deaths due to auto accidents in the U.S. in 1992 | 41710 |



## In-Class Exercise: Elicitation of Numerical Estimates [3]

- Descriptive Statistics
  - 50%, 25%, 75% guesses (median, first-second quartiles, third-fourth quartiles)
  - Box plots [Tukey, 1977]: actual frequency of data within 25-75% bounds
  - *What kind of descriptive statistics do you think might be informative?*
  - *What kind of descriptive graphics do you think might be informative?*
- Common Effects
  - Typically about half (50%) in first set
  - Usually, see some improvement in second set
  - Bounds also widen from first to second set (second system effect [Brooks, 1975])
  - *Why do you think this is?*
  - *What do you think the ramifications are for interactive elicitation?*
  - *What do you think the ramifications are for learning?*
- Prescriptive (Normative) Conclusions
  - Order-of-magnitude ("back of the envelope") calculations [Bentley, 1985]
  - Value-of-information (VOI): framework for selecting questions, precision



## Overview: Making Decisions in Uncertain Environments

- Problem Definition
  - Given: stochastic environment, outcome *P*(*Result* (*action*) | *Do*(*action*), *state*)
  - Return: a policy *f* : *state* → *action*
- Foundations of Sequential Decision Problems and Policy Learning
  - Utility function: *U* : *state* → *value*
  - *U*(*State*): analogy with *P*(*State*) ≡ agent's belief as distributed over event space
  - Expresses *desirability* of state according to decision-making agent
- Constraints and Rational Preferences
  - Definition: a lottery is defined by the set of outcomes of a random scenario and a probability distribution over them (e.g., denoted [*p*, *A*; 1 - *p*, *B*] for outcomes *A*, *B*)
  - Properties of rational preference (ordering on utility values)
    - Total ordering: antisymmetric, transitive, and
    - Continuity:
    - Substitutability:
    - Monotonicity:
    - Decomposability:

## Markov Decision Processes and Markov Decision Problems

- Maximum Expected Utility (MEU)
  - $E[U(action \mid D)] = \sum_i P(Result_i(action) \mid Do(action), D) \cdot U(Result_i(action))$
  - $D$ denotes agent's available evidence about world
  - Principle: *rational agent should choose actions to maximize expected utility*
- Markov Decision Processes (MDPs)
  - Model: probabilistic state transition diagram, associated actions $A$: *state* → *state*
  - Markov property: transition probabilities from any given state depend only on the state (not previous history)
  - Observability
    - Totally observable (MDP, TOMDP), *aka* accessible
    - Partially observable (POMDP), *aka* inaccessible, hidden
- Markov Decision Problems
  - Also called MDPs
  - Given: a stochastic environment (process model, utility function, and $D$)
  - Return: an optimal policy $f$ : *state* → *action*



## Value Iteration

- Value Iteration: Computing Optimal Policies by Dynamic Programming
  - Given: transition model $M$, reward function $R$: *state* → *value*
  - $M_{ij}(a)$ denotes probability of moving from state $i$ to state $j$ via action $a$
  - Additive utility function *on state sequences*: $U[s_0, s_1, \ldots, s_n] = R(s_0) + U[s_1, \ldots, s_n]$
- Function *Value-Iteration* ($M$, $R$)
  - Local variables $U$, $U'$: "current" and "new" utility functions, initially identical to $R$
  - REPEAT
    - $U \leftarrow U'$
    - FOR each state $i$ DO // dynamic programming update

      $U'[i] \leftarrow R[i] + max_a \sum_j M_{ij}(a) \cdot U[j]$

    UNTIL *Close-Enough* ($U$, $U'$)
  - RETURN $U$ // approximate utility function on all states
- Result: Provably Optimal Policy [Bellman and Dreyfus, 1962]
  - Use computed $U$ by maximizing utility $U$(*next action* | $s_i$)
  - Evaluation: RMS error of $U$ or expected difference $U^* - U$ (policy loss)



## Policy Iteration

- Policy Iteration: Another Algorithm for Calculating Optimal Policies
  - Given: transition model $M$, reward function $R$: *state* → *value*
  - Value determination function: estimates current $U$ (e.g., by solving linear system)
- Function *Policy-Iteration* ($M$, $R$)
  - Local variables $U$: initially identical to $R$; $P$: policy, initially optimal under $U$
  - REPEAT
    - $U \leftarrow$ *Value-Determination* ($P$, $U$, $M$, $R$); *unchanged?* ← true
    - FOR each state $i$ DO // dynamic programming update

      IF $max_a \sum_j M_{ij}(a) \cdot U[j] > \sum_j M_{ij}(P[i]) \cdot U[j]$ THEN

      $P[i] \leftarrow R[i] + \arg max_a \sum_j M_{ij}(a) \cdot U[j]$; *unchanged?* ← false

    UNTIL *unchanged?*
  - RETURN $P$ // optimized policy
- Guiding Principle: Value Determination Simpler than Value Iteration
  - Reason: action in each state is fixed by the policy
  - Solutions: use value iteration without *max*; solve linear system



## Applying Policies: Decision Support, Planning, and Automation

- Decision Support
  - Learn an action-value function (to be discussed soon)
  - Calculate MEU action in current state
  - Open loop mode: *recommend* MEU action to agent (e.g., user)
- Planning
  - Problem specification
    - Initial state $s_0$, goal state $s_G$
    - Operators (actions, preconditions ≡ applicable states, effects ≡ transitions)
  - Process: computing policy to achieve goal state
  - Traditional: symbolic; first-order logic (FOL), subsets thereof
  - "Modern": abstraction, conditionals, temporal constraints, uncertainty, etc.
- Automation
  - Direct application of policy
  - Caveats: partially observable state, uncertainty (measurement error, etc.)



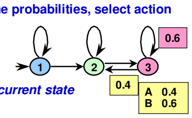
## Decision-Theoretic Agents

- Function *Decision-Theoretic-Agent* (*Percept*)
  - Percept: agent's input; collected evidence about world (from sensors)
  - COMPUTE updated probabilities for current state based on available evidence, including current percept and previous action
  - COMPUTE outcome probabilities for actions, given action descriptions and probabilities of current state
  - SELECT *action* with highest expected utility, given probabilities of outcomes and utility functions
  - RETURN *action*
- Decision Cycle
  - Processing done by rational agent at each step of action
  - Decomposable into prediction and estimation phases
- Prediction and Estimation
  - Prediction: compute pdf over expected states, given knowledge of previous state, effects of actions
  - Estimation: *revise belief* over current state, given prediction, new percept



## Kalman Filtering

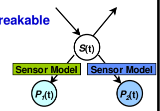
- Intuitive Idea
  - *Infer* "where we are" in order to compute outcome probabilities, select action
  - Inference problem: estimate $Bel(X(t))$
- Problem Definition
  - Given: action history, new percept
  - Return: estimate of *probability distribution over current state*




- Assumptions
  - State variables: real-valued, normal (Gaussian) distribution
  - Sensors: unbiased (mean = 0), normally distributed (Gaussian) noise
  - Actions: can be described as vector of real values, one for each state variable
  - New state: linear function of previous state, action
- Interpretation as Bayesian Parameter Estimation
  - Technique from classical control theory [Kalman, 1960]
  - Good success even when not all assumptions are satisfied
  - Prediction: $\hat{Bel}(X(t)) = \sum P(X(t) / X(t-1) = x(t-1), action(t-1)) \cdot Bel(X(t-1) = x(t-1))$
  - Estimation:

## Sensor and Data Fusion

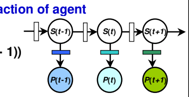
- Intuitive Idea
  - Sensing in uncertain worlds
  - Compute estimates of conditional probability tables (CPTs)
    - Sensor model (how environment generates sensor data): $P(percept(t) \mid X(t))$
    - Action model (how actuators affect environment): $P(X(t) \mid X(t-1), action(t-1))$
  - Use estimates to implement *Decision-Theoretic-Agent : percept → action*
- Assumption: Stationary Sensor Model
  - Stationary sensor model: $\forall t \,.\, P(percept(t) \mid X(t)) = P(percept(t) \mid X)$
    - Circumscribe (exhaustively describe) percept influents (variables that affect sensor performance)
    - *NB*: this does *not* mean sensors are immutable or unbreakable
  - Conditional independence of sensors given true value
- Problem Definition
  - Given: multiple sensor values for same state variables
  - Return: *combined* sensor value
  - Inferential process: sensor fusion, *aka* sensor integration, *aka* data fusion






## Dynamic Bayesian Networks (DBNs)

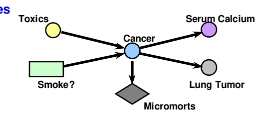
- Intuitive Idea
  - State of environment evolves over time
    - Evolution modeled by conditional pdf: $P(X(t) \mid X(t-1), action(i-1))$
    - Describes how state depends on previous state, action of agent
  - Monitoring scenario
    - *Agent can only observe (and predict)*: $P(X(t) \mid X(t-1))$
    - State evolution model, *aka* Markov chain
  - Probabilistic projection
    - Predicting continuation of observed $X(t)$ values (see last lecture)
    - Goal: *use results of prediction and monitoring* to make decisions, take action
- Dynamic Bayesian Network (*aka* Dynamic Belief Network)
  - Bayesian network *unfolded through time* (one note for each state and sensor variable, at each step)
  - Decomposable into prediction, rollup, and estimation phases
  - Prediction: as before; rollup: compute $\hat{Bel}(X(t))$; estimation: unroll $X(t+1)$






## Dynamic Decision Networks (DDNs)

- Augmented Bayesian Network [Howard and Matheson, 1984]
  - Chance nodes (ovals): denote random variables as in BBNs
  - Decision nodes (rectangles): denote points where agent has choice of actions
  - Utility nodes (diamonds): denote agent's utility function (e.g., in chance of death)
- Properties
  - Chance nodes: related as in BBNs (CI assumed among nodes not connected)
  - Decision nodes: choices can influence chance nodes, utility nodes (directly)
  - Utility nodes: conditionally dependent on joint pdf of parent chance nodes and decision values at parent decision nodes
  - See Section 16.5, Russell and Norvig
- Dynamic Decision Network
  - *aka* dynamic influence diagram
  - DDN : DBN :: DN : BBN
  - Inference: over predicted (unfolded) sensor, decision variables






## Learning to Make Decisions in Uncertain Environments

- Learning Problem
  - Given: interactive environment
    - No notion of *examples* as assumed in supervised, unsupervised learning
    - Feedback from environment in form of rewards, penalties (reinforcements)
  - Return: utility function for decision-theoretic inference and planning
    - Design 1: utility function on states, $U$ : *state → value*
    - Design 2: action-value function, $Q$ : *state × action → value* (expected utility)
  - Process
    - Build predictive model of the environment
    - Assign credit to components of decisions based on (current) predictive model
- Issues
  - How to explore environment to acquire feedback?
  - Credit assignment: how to propagate positive credit and negative credit (blame) back through decision model *in proportion to importance*?



## Terminology

- Making Decisions in Uncertain Environments
  - Policy learning
    - Performance *element*: decision support system, planner, automated system
    - Performance *criterion*: utility function
    - Training signal: reward function
  - MDPs
    - Markov Decision Process (MDP): model for decision-theoretic planning (DTP)
    - Markov Decision Problem (MDP): problem specification for DTP
    - Value iteration: iteration over actions; decomposition of utilities into rewards
    - Policy iteration: iteration over policy steps; value determination at each step
  - Decision cycle: processing (inference) done by a rational agent at each step
  - Kalman filtering: estimate belief function (pdf) over state by iterative refinement
  - Sensor and data fusion: combining multiple sensors for same state variables
  - Dynamic Bayesian network (DBN): temporal BBN (*unfolded through time*)
  - Dynamic decision network (DDN): temporal decision network
- Learning Problem: Based upon Reinforcements (Rewards, Penalties)



## Summary Points

- Making Decisions in Uncertain Environments
  - Framework: Markov Decision Processes, Markov Decision Problems (MDPs)
  - Computing policies
    - Solving MDPs by dynamic programming given a *stepwise reward*
    - Methods: value iteration, policy iteration
  - Decision-theoretic agents
    - Decision cycle, Kalman filtering
    - Sensor fusion (*aka* data fusion)
  - Dynamic Bayesian networks (DBNs) and dynamic decision networks (DDNs)
- Learning Problem
  - Mapping from observed actions and rewards to decision models
  - Rewards/penalties: reinforcements
- Next Lecture: Reinforcement Learning
  - Basic model: *passive learning in a known environment*
  - *Q* learning: policy learning by adaptive dynamic programming (ADP)