

Fundamentals of Artificial Intelligence
Prof. Shyamanta M Hazarika
Department of Mechanical Engineering
Indian Institute of Technology- Guwahati

Lecture-26
Making Complex Decisions

Welcome to fundamentals of artificial intelligence, we are addressing computational issues in making decisions particularly making complex decisions. In the last lecture we had looked at sequential decision problems where the utility is based on a sequence of decisions, your solution is not a sequence of actions, but rather a policy, a set of situation action pair for each state. We had focused on accessible environments and looked at a Markov decision problem.

The problem of computing an optimal policy in an accessible stochastic environment with a known transition model, we have looked at value iteration and algorithm for computing the optimal policy and also looked at policy iteration and alternate formulation to arrive at the policy in an inaccessible environment the percepts do not give enough information to determine the state and the transition probabilities such a problem is referred to as partially observable Markov decision problem.

Exact solution for partially observable Markov decision problems is difficult and it is possible to arrive at approximate solutions through the technology of decision networks. This is the focus of the lecture today.

(Refer Slide Time: 02:37)

POMDP



- In an inaccessible environment, the percept does not provide enough information to determine the state or the associated transition probabilities; Such problems are called Partially Observable Markov Decision Problems, or POMDP.
- Methods used for Markov Decision Problems are not directly applicable to POMDPs.
- The standard method for solving a POMDP is to construct a new MDP in which the current probability distribution plays the role of the state variable.

2

© Sipser and M. Thrun, MIT, 6.034

So methods used for Markov decision problems are not directly applicable to partially observable Markov decision problems. The standard method for solving a POMDP is to actually construct a new Markov decision problem in which the current probability distribution plays the role of the state variable.

(Refer Slide Time: 03:04)

POMDP



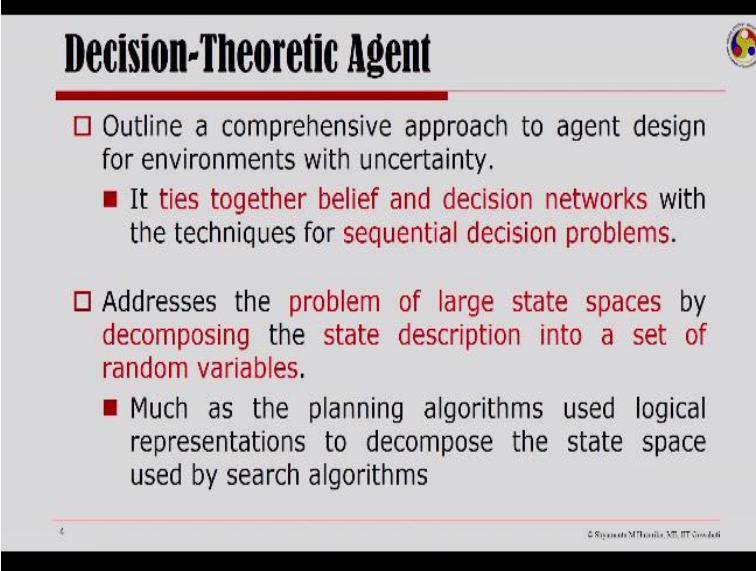
- Resulting MDP is not easy to solve! State space is characterized by real-valued probabilities; therefore infinite.
- Exact solution methods for POMDPs require some fairly advanced tools; Beyond the scope of this course.
- Instead of exact solutions for POMDPs, often obtain a good approximation using a limited lookahead.
 - This can be realized for POMDPs using Dynamic Decision Networks.

3

© Sipser and M. Thrun, MIT, 6.034

The resulting Markov decision problem is not easy to solve, this is because now the state space is characterized by real-valued probabilities and therefore is infinite. Exact solution methods for partially observable Markov decision problems require some fairly advanced tools and therefore we often obtain approximate solutions using a limited look ahead. Now this can be realized for partially observable Markov decision problems using dynamic decision networks.

(Refer Slide Time: 03:48)



Decision-Theoretic Agent

- Outline a comprehensive approach to agent design for environments with uncertainty.
 - It ties together belief and decision networks with the techniques for sequential decision problems.
- Addresses the problem of large state spaces by decomposing the state description into a set of random variables.
 - Much as the planning algorithms used logical representations to decompose the state space used by search algorithms

© Stuart Russell, 2011. MIT OpenCourseWare

Before we move on to discuss dynamic decision networks let us look first on the formulation of a decision theoretic agent, we outline a comprehensive approach to agent design for environments with uncertainty. Now for such an agent it needs to tie together belief and decision networks together with the techniques for sequential decision problems that we have looked at in the last lecture. This idea of putting together belief and decision networks address the problem of large state spaces by decomposing the state description into a set of random variables.

This is something like the planning algorithm where logical representations are used to decompose the state space used by the search algorithms.

(Refer Slide Time: 04:52)

Decision-Theoretic Agent



□ Decision Theory = Probability theory
+
Utility theory

The fundamental idea of decision theory is that an agent is rational if and only if it **chooses the action that yields the highest expected utility, averaged over all possible outcomes** of the action.

5

© Raymond M. Turner, MIT, 6.034

Now kindly recall that we have looked at the very definition of what we meant by a decision theory in one of our previous lectures where we have seen that the coming together of utility theory and the probability theory is referred to as the decision theory. The fundamental idea of decision theory is that a agent is rational if and only if it chooses the action that is the highest expected utility.

So given some actions on which the utility is to be computed we would love to go via the process of decision theory where we look at the probability theory and utility theory and we choose the action that yields the highest expected utility averaged over all possible outcomes of the action.

(Refer Slide Time: 05:53)

Decision Theoretic Agent: Decision Cycle



function DECISION-THEORETIC-AGENT(*percept*) **returns** *action*

- ✓ calculate updated probabilities for current state based on available evidence including current percept and previous action
 - ✓ calculate outcome probabilities for actions given action descriptions and probabilities of current states
 - ✓ select *action* with highest expected utility given probabilities of outcomes and utility information
- return** *action*

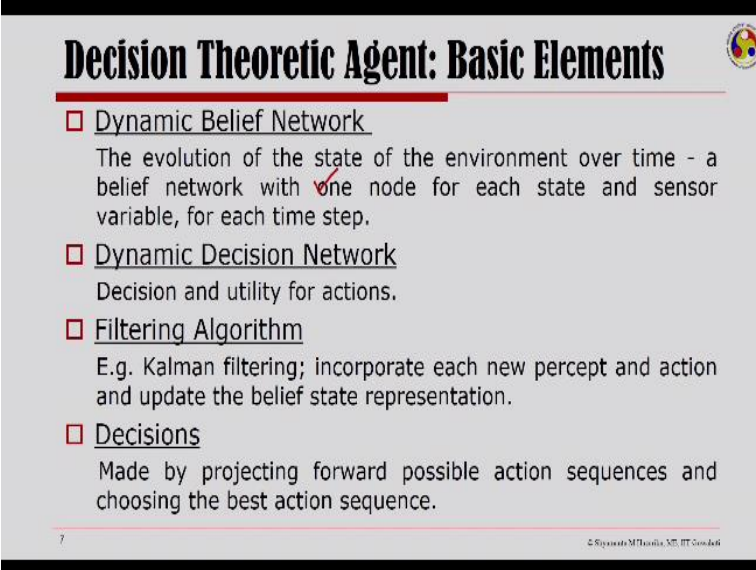
Schematic agent design for rational agents; The processing done by the agent at each step is the decision cycle.

9

© Raymond M. Turner, MIT, 6.034

The schematic agent design for rational agents is what is given here we calculate the updated probabilities for the current state and this is done based on available evidence which is including the current percept and the previous action, we then compute the probabilities for actions given action descriptions and probabilities of the current state and then we select action with the highest expected utility given probabilities of outcomes and the utility information. Now the processing that is done by an agent at each step is called a decision cycle.

(Refer Slide Time: 06:42)



Decision Theoretic Agent: Basic Elements

- Dynamic Belief Network
The evolution of the state of the environment over time - a belief network with one node for each state and sensor variable, for each time step.
- Dynamic Decision Network
Decision and utility for actions.
- Filtering Algorithm
E.g. Kalman filtering; incorporate each new percept and action and update the belief state representation.
- Decisions
Made by projecting forward possible action sequences and choosing the best action sequence.

7 © Sipser and MIT course, MIT, BT Condell

The decision theoretic agent has to have a number of basic elements which are the dynamic belief network, the dynamic decision network, our filtering algorithm and a couple of decisions. The dynamic belief network is about the evolution of the state of the environment over time, a belief network with nodes for each state and sensor variable for each time step. The dynamic decision network brings together decision and utility for actions.

We have filtering algorithms which incorporate each new percept and action and update the belief state representation. Now decisions are made by projecting forward possible action sequences and choosing the best action sequence.

(Refer Slide Time: 07:39)

Current State of the World



□ State Variables

Set of random variables X , that refer to the current state of the world.

For example, if the agent is a robot moving in the X-Y plane, then we might use (X_t, Y_t) to refer to the robot's position at time t .

□ Belief

The belief about the state at time t is the probability distribution over the state given all available evidence:

$$Bel(X_t) = P(X_t | E_1 \dots E_t, A_1 \dots A_{t-1})$$

where X_t is the state variable; E_t is the evidence variable

Complicated expression, and direct evaluation is out of the question because it requires conditioning on a large number of variables.

8

© Stuart M. Russell, MIT, 6.034

The current state of the world under such a scenario is described by the state variables as well as a belief. The state variable is a set of random variables that refer to the current state of the world. For example if I have an agent which is a robot let us say moving in the X-Y plane, then we might use X_t, Y_t to refer to the robots position at time T and the belief about the state at time T is the probability distribution over the state given all available evidence.

So belief at X_t where X_t is the state variable is the probability of X_t given the evidences E_1, E_2, E_3 so on and so forth up to E_t .

(Refer Slide Time: 08:40)

Calculation of Belief - Assumptions



1. The main assumption is that the problem is Markovian — the probability distribution for the current state of the world depends only on the previous state and the action taken in it.

$$P(X_t | X_1 \dots X_{t-1}, A_1 \dots A_{t-1}) = P(X_t | X_{t-1}, A_{t-1})$$

2. Each percept depends only on the state at the time. Percepts (E_t) are causally determined by the state of the world:

$$P(E_t | X_1 \dots X_t, A_1 \dots A_{t-1}, E_1 \dots E_{t-1}) = P(E_t | X_t)$$

9

© Stuart M. Russell, MIT, 6.034

The main assumptions that are required for calculation of the belief are number 1 that the problem is Markovian, the problem being Markovian means that the probability distribution for the current state of the world depends only on the previous state and action in it. So if I am talking of the probability of X_t given X_1 up to X_{t-1} and actions A_1 to A_{t-1} , then the Markovian assumption means that the probability of X_t would only depend on X_{t-1} and A_{t-1} , the state and the action just in the previous time point.

Each percept depends only on the state at the time, so percepts are causally determined by the state of the world. So if I have a percept E_t then the probability of E_t would just depend on the state X_t , this is the second assumption on way to calculation of beliefs.

(Refer Slide Time: 09:56)

Calculation of Belief - Assumptions

3. A similar equation holds for actions. The **action taken depends only on the percepts the agent has received to date**

$$P(A_{t-1} | A_1 \dots A_{t-2}, E_1 \dots E_{t-1}) = P(A_{t-1} | E_1 \dots E_{t-1})$$

This final assertion is valid because of the structure of the agent itself: its only input from the outside is the percept at each time step.

□ Taken together, **above equations allow us to simplify the calculation of the current state estimate .**

19 © Raymond M. Thomas, MS, ET, Consider

The third assumption is that the action taken depends only on the percepts and the agent has received today. So if I am talking of an action A_{t-1} , so all that it will depend on would be the percept $E_1 E_2 E_3$ so on and so forth up to E_{t-1} . Now this final assertion is valid because of the structure of the agent itself it is only input form from the outside is the percept at each time step. Now taken together the above equations allow us to simplify the calculation of the current state estimate.

(Refer Slide Time: 10:41)

Calculation of Belief - Phases



□ Prediction phase:

1. Predict the **probability distribution over states** we would have expected, given our knowledge of the previous state and how actions affect states.
2. Calculate it by adding up the probabilities of arriving in a given state at time t for each of the states we could have been in at time $t-1$.

$$\hat{Bel}(X_t) = \sum_{X_{t-1}} P(X_t | X_{t-1} = x_{t-1}, A_{t-1}) Bel(X_{t-1} = x_{t-1})$$

X_{t-1} ranges over all possible values of X_{t-1}

11

© Stewart M. Edelkamp, M.Sc., Ph.D. 2006

The calculation of belief have 2 phases, 1 the prediction phase in which we predict the probability distribution over states we would have expected given our knowledge of the previous state and how actions affect states and calculate it by adding up the probabilities of arriving in a given state at time T for each of the state. So the belief at X_t would be the probability of summations of all state time points. So we add up all the probabilities at time T for each of the states we could have been in at time $t - 1$. And here the X_{t-1} ranges over all the state variables during $t - 1$.

(Refer Slide Time: 11:35)

Calculation of Belief - Phases



□ Estimation phase:

1. Now we have a distribution over the current state variables, given everything but the most recent observation.
2. The estimation phase updates this using the percept E_t . Because both the state variables and the percept refer to the same time, this is a simple matter of Bayesian updating

$$Bel(X_t) = \alpha P(E_t | X_t) \hat{Bel}(X_t)$$

α is a normalization constant

12

© Stewart M. Edelkamp, M.Sc., Ph.D. 2006

In the estimation phase once we have a distribution over the current state variables we are given everything but the most recent observation. So the estimation phase updates this using the

percept at time t because both the state variables and the percept refer to the same time this is just a Bayesian updating, that is the belief at X t is the probability of Et and Xt and the probability distribution of the belief at X t. Now here alpha is a normalization constant.

(Refer Slide Time: 12:16)

Decision Theoretic Agent: Detailed Design

Function DECISION-THEORETIC-AGENT(E_t) returns an action

inputs: E_t , the percept at time t

static: BN , a belief network with nodes X

$Bel(X)$, a vector of probabilities, updated over time

$Bel(X_t) \leftarrow \sum_{x_{t-1}} P(X_t | X_{t-1} = x_{t-1}, A_{t-1}) Bel(X_{t-1} = x_{t-1})$ The action model describes the effect of actions.

$Bel(X_t) \leftarrow \alpha P(E_t | X_t) Bel(X_t)$ The sensor model describes how the environment generates the sensor data.

$action \leftarrow \arg \max_A \sum_{x_t} [Bel(X_t = x_t) \sum_{x_{t+1}} P(X_{t+1} = x_{t+1} | X_t = x_t, A_t) U(x_{t+1})]$

return action

The action model generalizes the transition model used earlier for sequential decision problems. The sensor model was not used for sequential decision problem, of course, because we assumed an accessible environment in which the percept and the state can be equated.

13 © Sipkowitz, MIT, 6.034

The decision theoretical that we have discussed with its 3 steps, now can be looked at in a more detailed way. So here we would take the percept, we would have a belief network and we will have a vector of probabilities updated over time from that we could create the probability distribution and then the belief itself at X t and then we would take an action based on the maximum expected utility.

The action model if you see generalizes the transition model used earlier for sequential decision problems. In the sequential decision problems we did not use the sensor model of course we did not do that because we assumed an accessible environment in which the percept and the state can be equated, here we take help of the sensor model. So the action model here it describes the effects of the action and the sensor model describes how the environment generates the sensor data.

(Refer Slide Time: 13:40)

Sensing in Uncertain Worlds



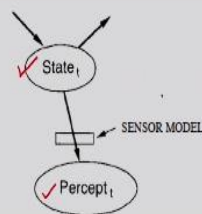
- Sensor model: $P(E_t | X_t)$
Describes how the environment generates the sensor data.
- Action model: $P(X_t | X_{t-1}, A_{t-1})$
Describes the effects of actions
- Stationary sensor model: $\forall t \quad P(E_t | X_t) = P(E | X)$
where E and X are random variables ranging over percepts and states
Advantage: $P(E | X)$ can be used at each time step.

Now sensing in the uncertain world we are talking of a sensor model the probability of E_t given X_t the state variable, this describes how the environment generates the sensor data, the action model probability of X_t given X_{t-1} and A_{t-1} describes the effects of action and a stationary sensor model is assumed here, that is for all t 's we assume that the probability of E_t given X_t is the probability of E given X where E and X are random variables ranging over percepts and states.

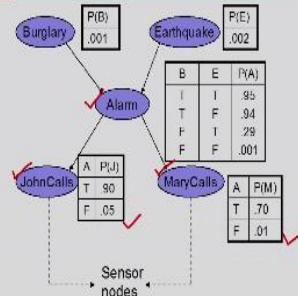
And this could be used at each time step, so we assume a stationary sensor model in a belief network.

(Refer Slide Time: 14:42)

Sensor Model in a Belief Network



Belief network fragment showing the general relationship between state variables and sensor variables.

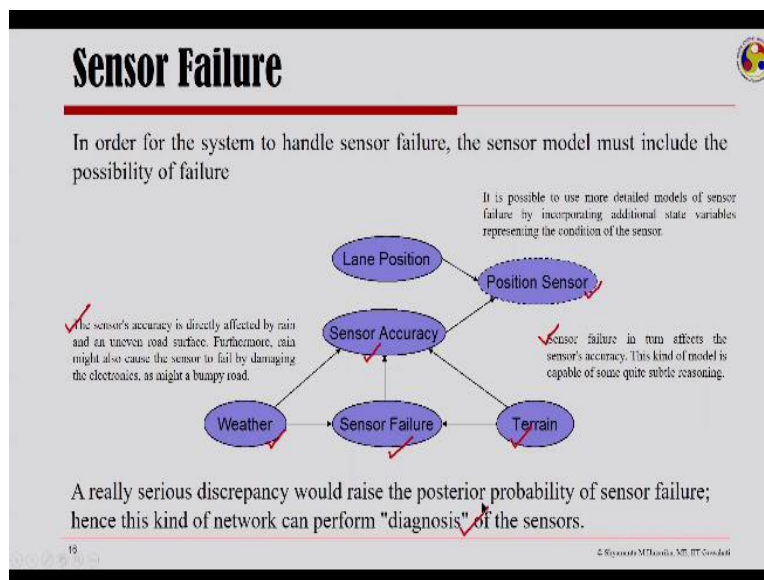


The sensor model itself is the CPT associated with the percept node. If the sensor gives a perfect report of the state, then the sensor model (the CPT) will be purely deterministic. In the burglar-alarm network, both *Johncalls* and *Marycalls* can be viewed as sensor nodes for the *Alarm* state variable. Their conditional probability tables shown in the figure above show how reliable they are as sensors.

So let us look at a little bit more detail to understand what we mean by a sensor model in a belief network. So here is a belief network fragment showing the general relationship between the state variable and the sensor variables. So from the state to the percept we come via the sensor model. The sensor model actually is the conditional probability distribution table associated with the percept node. If the sensor gives a perfect report of the state then the sensor model will be purely deterministic.

So let us recall the burglar alarm network that we have discussed while we were talking of Bayesian networks, here John calls and Mary calls can both be seen as sensor nodes for the alarm state variable. Their conditional probability tables shown in the figure here show how reliable these 2 sensors are.

(Refer Slide Time: 16:05)



So in order for a system to handle sensor failure the sensor model must include the possibility of failure. Now it is possible to use more detailed models of sensor failure by incorporating additional state variables representing the condition of the sensor anyone with hands-on experience of robotics computerized process control or other forms of automatic sensing and control will readily testify to the fact that sensors fail.

When a sensor fails it does not necessarily send a signal announcing its failure instead it simply starts sending garbage, this can be dangerous if taken literally. For example a robot's shown on a

distance sensor might start sending infinity meaning that no object is near its vicinity, this could be because the sonar's detector is broken. In this case the robot could start crashing into us and this is why we need sensor models that allow failure.

A sensor model for the sonar sensor that says that the sensor is accurate within some stated distance explicitly disallows the possibility of failure and therefore forces the robot to take the sensor reading literally, for any given actual distance the sonar model on the other hand should allow the possibility that the observed distance could be infinity. Then the robot can handle sensor failure more appropriately.

For example if the robot is in a room and reports infinity then the most likely conclusion would be that the sensor has failed, furthermore if the robot has more than one distance sensor the sensor fusion process will automatically discount the reading of the failed sensor. The figure here on your screen shows a model of a reason based lane position sensor, such sensors are used in autonomous vehicles to keep them in the centre of their lane.

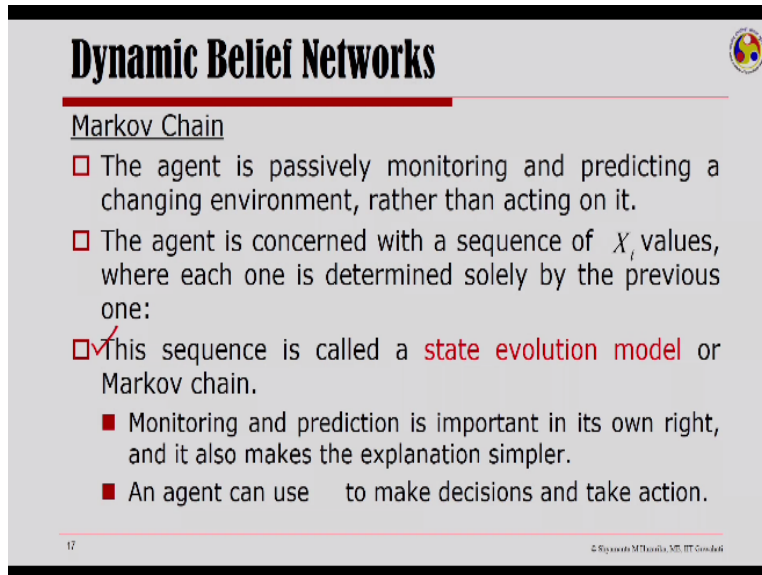
They also could be used to sound a warning in a human driven car when it starts to move away from its desired course. So the sensor's accuracy in this case that is being shown here is directly affected by an uneven road surface furthermore rain might also cause the sensor to fail by damaging the electronics as might a bumpy road. So we are talking of sensor failure which would depend on whether or not it is raining.

And all of this would somehow impact sensor accuracy. The sensor's failure affects the sensor's accuracy, we know that the sensor's accuracy would impact the position sensor and the lane position would be dependent on the position sensor. This kind of model is capable of quite certain reasoning. For example if the system believes that it is raining then it will alter the sensor accuracy variable raising the likelihood of larger error in the lane position sensor.

When an unexpected reading occurs the system will be less likely to assume that the car is out of position conversely a large discrepancy between the expected and observed position can on the other hand increase the system's belief that it is raining without perhaps any other sensors

involvement. Now a really serious discrepancy would raise the posterior probability of sensor failure. Hence this kind of networks can perform some diagnosis of the sensor.

(Refer Slide Time: 20:46)



Dynamic Belief Networks

Markov Chain

- The agent is passively monitoring and predicting a changing environment, rather than acting on it.
- The agent is concerned with a sequence of X_t values, where each one is determined solely by the previous one:
- This sequence is called a **state evolution model** or Markov chain.
 - Monitoring and prediction is important in its own right, and it also makes the explanation simpler.
 - An agent can use to make decisions and take action.

17 © Srijayanta M. Thakuria, IIT Guwahati

Now let us look at what we mean by dynamic belief networks, first we shall talk of a Markov chain. Now an agent can be passively monitoring and predicting a changing environment rather than acting on it. So the agent is concerned with a sequence of state variable X_t values where each one is determined solely by the previous one. Now this sequence is called a state evolution model or a Markov chain. Monitoring and prediction is important in its own right and it also makes the explanation simpler. The agent can use such a model to make decisions and take actions.

(Refer Slide Time: 21:45)

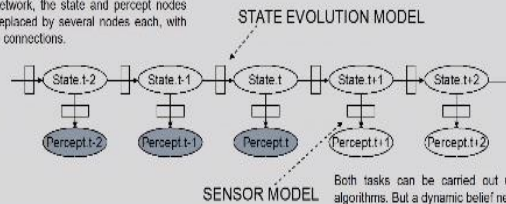
Dynamic Belief Network



Dynamic Belief Network

- A belief network with one node for each state and sensor variable for each time step.

In a real network, the state and percept nodes would be replaced by several nodes each, with appropriate connections.



Both tasks can be carried out using the standard algorithms. But a dynamic belief network shown above could be extremely large, so the belief net algorithms could be extremely inefficient.

Two tasks of the network:

- Calculate the probability distribution for state at time t
- Probabilistic projection: concern with how the state will evolve into the future

18

© Stuart Russell, 2000. MIT Course 6.034

A dynamic belief network is a belief network with one node for each state and sensor variable for each time step. So in a real network the state and the percept nodes would be replaced by several nodes each with appropriate connections and there would be 2 tasks for the n network one you need to calculate the probability distribution for state at time T and next it would be concerned with how the state will evolve into the future that is called the probabilistic projection.

Now both this task can be carried out using standard algorithms but the dynamic belief network that we have shown here could be extremely large. So the belief that algorithms could be extremely inefficient.

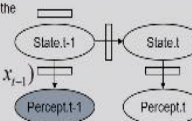
(Refer Slide Time: 22:50)

Prediction, Roll-Up and Estimation



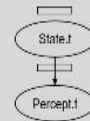
- Prediction: Calculate the belief vector. This is actually the standard belief network updating process.

$$\hat{Bel}(X_t) = \sum_{X_{t-1}} P(X_t | X_{t-1} = x_{t-1}, A_{t-1}) Bel(X_{t-1} = x_{t-1})$$



- Rollup: remove slice t-1

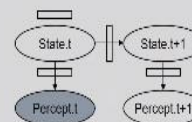
Requires adding a prior probability table for the state variables at time t .



- Estimation:

$$Bel(X_t) = \alpha P(E_t | X_t) \hat{Bel}(X_t)$$

Applying standard belief network updating to calculate $Bel(X_t)$, the probability distribution over the current state. We then add the slice for $t+1$.



Now we will see the benefit of all the work that went into the equations. Implement the prediction and estimation phases as operations on the belief network. This process implements the formal algorithm specified in the detailed design for a decision theoretic agent, using the belief network inference machinery for all calculations.

19

© Stuart Russell, 2000. MIT Course 6.034

This is where the idea of using beliefs to estimate and come to the best possible sequence of actions is what is important. So we have prediction roll up an estimation and now we will see the benefit of all the work that went into these equations. We implement the prediction and estimation phase as operations on the belief network. So this process actually implements the formal algorithms specified in the detail design for a decision theoretic agent using the belief network inference machinery.

So first you predict and thereafter you remove some slices at $t - 1$ and then formally arrive at the estimation. So you calculate the belief vector and this is actually the standard belief network updating process, thereafter at a prior probability table for the state variable at time T and applying standard belief network updating calculate the belief at X_t . The probability distribution over the current state, we then add the slice for $T + 1$.

Let us repeat the process of prediction roll up an estimation one more time, I will particularly emphasize that we implement the prediction an estimation phases as operations on the belief network, further we need only structure to represent the 2 time steps referred to here as the 2 slices, on the right of your screen you can see the prediction estimation process in operation each cycle of the process go through the steps of prediction, ready roll up an estimation.

We begin with prediction, we have a 2 slice network the slices are $t - 1$ and t we have already computed the belief of $X_{t - 1}$ incorporating all evidence up to and including the precept $E_{t - 1}$. The slice $t - 1$ has no connection to previous slices, the state variables in $t - 1$ have prior probabilities associated with them we then calculate the belief vector at t , this is actually the standard belief network updating process applied to evidence $t - 1$.

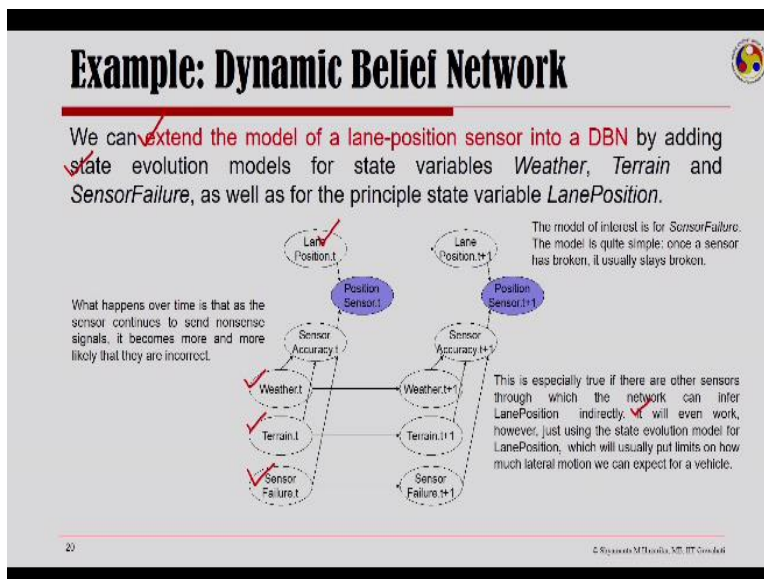
The next stage is of roll-up where we remove the slice of $t - 1$, this requires now adding a prior probability table for the state variables at time t . Now this prior is just what we have calculated in the previous step. Finally we arrive at estimation when we add the new percept U of T applying standard belief network updating to calculate the belief involving X_t , the probability distribution over the current state.

We then add the slice for $t + 1$ and now the network is ready for the next cycle, one needs to realize at this point that this process that we have highlighted here actually implements the formal algorithm that we had specified in the detailed design of the decision theoretic agent using the belief network inference mechanism for all the calculations. Notice that as in the formal algorithm the percent history is summarized in the belief vector for the current state.

Now probabilistic projection is also straightforward we take the network and we add slices for the future times and we then apply a belief network inference algorithm to calculate the posterior probability distribution for the future states given the current percept. Now one thing to note here is that unlike the update cycle this might be expensive because here it involves inference in a temporarily extended network.

However a very interesting property of this network that one needs to know is that none of the future nodes has any evidence associated with it, this means that a simple stochastic simulation technique will work well because every run can be consistent with the evidence.

(Refer Slide Time: 28:28)



Now as an example of the application of dynamic belief networks consider again the sense of failure model that we have seen earlier, this can be extended into a dynamic belief network, the figure here shows a 2 slice fragment of a dynamic belief network for continuous monitoring of

the lane positioning of an automated vehicle. Focus on the model of the lane position sensor that we have shown while we were talking of sensor failure.

And extend the model into a dynamic belief network, in order to extend that into a dynamic belief network we need to have state evaluation models for static variables which were the weather, the terrain and the sensor failure as well as for the principle state variable which was lane position. Now the model of interest for us here is sensor failure and the model is quite simple, we say once a sensor is broken it usually stays broken.

What happens over time is that as the sensor continues to send nonsense signals it becomes more and more likely that they are incorrect. This is especially true if there are other senses through which the network can infer lane position indirectly. Now it will even work however just using the state evaluation model for lane position which will easily put limits on how much lateral motion we can expect for a vehicle. And this is how we could take decisions on what is happening to lane position vis-a-vis our percepts.

(Refer Slide Time: 30:45)

Prediction, Roll-Up and Estimation

Prediction: Calculate the belief vector. This is actually the standard belief network updating process.

$$\hat{Bel}(X_t) = \sum_{X_{t-1}} P(X_t | X_{t-1} = x_{t-1}, A_{t-1}) Bel(X_{t-1} = x_{t-1})$$

Rollup:
remove slice t-1

Requires adding a prior probability table for the state variables at time t.

Estimation:

$$Bel(X_t) = \alpha P(E_t | X_t) \hat{Bel}(X_t)$$

Now we will see the benefit of all the work that went into the equations. Implement the prediction and estimation phases as operations on the belief network. This process implements the formal algorithm specified in the detailed design for a decision theoretic agent, using the belief network inference machinery for all calculations.

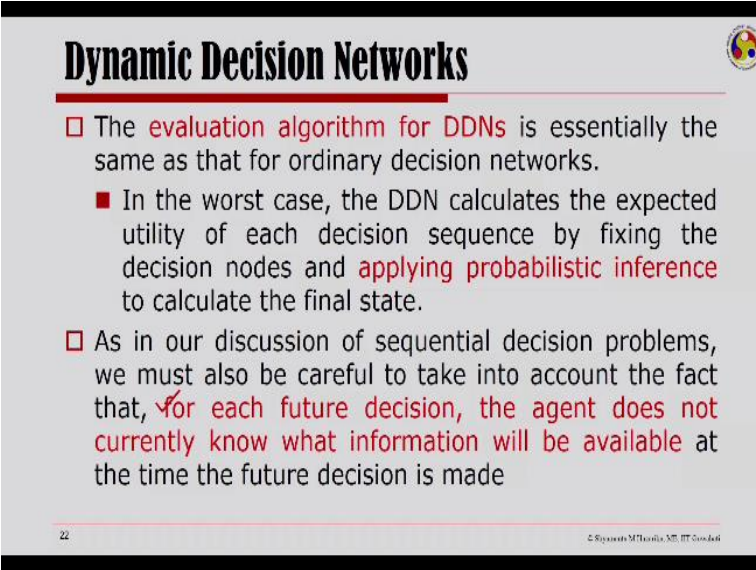
19
© Wataru M. Tanaka, M.S., Ph.D., 2004

Now dynamic decision networks add utility nodes and decision nodes for action into the dynamic belief network. So the general structure of a dynamic decision network for a sequential decision problem where the terminal states are 3 steps ahead is shown here. The decision problem involves calculating the values of E_t that maximizes the utility over the remaining state

sequence, that is at $t + 1$, $t + 2$ and $t + 3$. So E_{t-1} is treated as evidence because it has already happened.

And the final utility would be computed as the sum of the expected rewards that would come out from 1, 2 and 3. Now here we have not shown the reward nodes just to make the diagram simple.

(Refer Slide Time: 32:02)



Dynamic Decision Networks

- The **evaluation algorithm for DDNs** is essentially the same as that for ordinary decision networks.
 - In the worst case, the DDN calculates the expected utility of each decision sequence by fixing the decision nodes and **applying probabilistic inference** to calculate the final state.
- As in our discussion of sequential decision problems, we must also be careful to take into account the fact that, **for each future decision, the agent does not currently know what information will be available** at the time the future decision is made

22 © Stephen M. Edelkamp, 2011, MIT OpenCourseWare

Dynamic decision networks the evaluation algorithm for dynamic decision networks is essentially the same as that for ordinary decision networks, in the worst case the dynamic decision network calculates the expected utility of each decision sequence by fixing the decision nodes and applying probabilistic inference to calculate the final same. As in our discussion of sequential decision problems we must also be careful to take into account the fact that for each future decision the agent does not currently know what information will be available at the time the future decision is made.

(Refer Slide Time: 32:54)

Dynamic Decision Networks



- In our earlier discussion, we handled this by iteratively computing a policy that associates a decision with each state.
- With DDNs, we do not have this option because the states are represented implicitly by the set of state variables. Furthermore, in inaccessible environments, the agent will not know what state it is in anyway.
- What we must do instead is consider each possible instantiation of the future sensor variables as well as each possible instantiation of the future decision variables.

23

© Raymond M. Jensen, MIT, 1998

In our earlier discussion we handle this by iteratively computing a policy that associates a decision with each state, with dynamic decision networks we do not have this option because the states are represented implicitly by the set of state variables and in inaccessible environments the agent will not know what state it is in anyway, what we must do in sit there for is to consider each possible instantiation of the future sensor variable as well as possible instantiation of the future decision variable.

(Refer Slide Time: 33:42)

Dynamic Decision Networks



- The expected utility of each decision sequence is then the weighted sum of the utilities computed using each possible percept sequence, where the weight is the probability of the percept sequence given the decision sequence.
- The **DDN provides approximate solutions for Partially Observable Markov Decision Problems**, where the degree of approximation depends on the amount of lookahead.
 - Just as we used a limited horizon in game playing and with value iteration and policy iteration, we can limit the extent of forward projection in the DON in order to reduce complexity.

24

© Raymond M. Jensen, MIT, 1998

And the expected utility of each decision sequence is therefore the weighted sum of the utilities computed using each possible percept sequence, in a way the dynamic decision networks provide approximate solutions for partially observable Markov decision problems, where the degree of

approximation as we have seen in our illustration depends on the amount of look ahead. So just as we use their limited horizon in game playing and with value iteration and policy iteration we can limit the extent of forward projection in dynamic decision networks in order to reduce the complexity.

(Refer Slide Time: 34:35)

Dynamic Decision Networks

- In evaluating an action, one must consider not only its effect on the environment, but also its effect on the internal state of the agent via the percepts it generates

- ✓ Limited horizon combined with a heuristic estimate for the utility of the remaining steps, can provide a reasonable approximation to rational action.

25 © Stanford M. Thrun, M. BT, 2004

In evaluating an action one must consider not only its effect on the environment, but also it affects on the internal state of the agent via the percepts it generates. So limited horizon combined with a heuristic estimate for the utility of the remaining steps can provide a reasonable approximation to rational action.

(Refer Slide Time: 35:02)

Final Comments

- Dynamic Decision Networks promises potential solutions to many of the problems that arise as AI systems are moved from static, accessible, and above all simple environments to dynamic, inaccessible, complex environments that are closer to the real world.
- Dynamic Decision Networks provide a general, concise representation for large POMDP, so they can be used as inputs for any POMDP algorithm including value and policy iteration methods.
- Overall, the potential payoff of combining DDN-like techniques with planning methods is enormous.
 - The technical and mathematical problems involved in getting it right are difficult, but it is an important area within AI.

25 © Stanford M. Thrun, M. BT, 2004

And thus dynamic decision networks promise potential solutions to many of the problems that arise in AI systems as we move from static accessible and above all simple environments to dynamic inaccessible complex environments that are closer to the real world. Dynamic decision networks does provide a general concise representation for large partially observable Markov decision problems.

Overall the potential payoff of combining dynamic decision network like techniques with planning methods is enormous. The technical and mathematical problems involved in getting it right are difficult, but it is a very important area with an AI, in our next module we should be looking at machine learning, thank you.