Overview

- Task
  - RL
  - IS
  - Combine IS and RL

Introduction

- Dialogue Manager is integral part of next-generation spoken dialogue systems
- Decides when the system should do next: make a suggestion, ask a question, answer a question, make an acknowledgment
- Two different approaches have become popular for building dialogue managers
- Symbolic Reasoning, especially Information State approach
- Hand-craft the conditions under which each action can be performed
- Learn for each state what action to perform
- Reinforcement Learning

Evaluation

Combine these two approaches
Example Dialogue

Flight Information Task

User actions:
- constrain p v: answer query that p has value v
- relax p a: answer whether p can be relaxed

System actions:
- display: display results to user
- determine whether any flights match the current parameters
- ask constraint: ask whether user is flexible on the values for parameter p
- ask value of parameter: ask value of parameter p

Goal of system:
- Display a short list of flights that meet the user's requirements
- Which is flexible on some of its parameters
- From to, airline, departure, number of stops
- User is assumed to have a flight in mind

User is assumed to have a flight in mind
- From, to, airline, departure, number of stops
- Might be flexible on some of the parameters
Reinforcement Learning (RL)

• RL determines the optimal action for the system to perform in each state according to a cost function.
Reinforcement Learning (RL)

• Reinforcement Learning (RL) determines the optimal action for the system to perform in each state, according to a cost function.

Cost function assesses how good a dialogue is:

- Whether preferred flight was found
- Number of flights displayed to user
- Database queries
- Length of dialogue

High Information Task: sum of four components:

- Lower the cost, the better the dialogue.

Set of actions captures what system can perform and level of granularity that RL will reason about:

- 13 actions:
  - Flight Information Task: 13 actions
  - 5 queries for each parameter (to, from, airline, departure time, number of stops)
  - 5 queries for each parameter (to, from, airline, departure time, number of stops)
  - Database query to determine which flights match the current parameters
  - Output action
  - Finish action

Reinforcement Learning (RL)
Policy Exploration

- RL works by estimating the cost $Q(s,a)$ for each $s,a$ where $Q(s,a)$ is the minimal cost to get to the end state from state $s$.
- RL interleaves:
  - Estimating $Q$ for the current policy: by running sample dialogues, and using the scores to update the values.
  - Improving the current policy: by choosing the optimal action in each state, given the current estimate of $Q$.
- RL explores the space of policies by using an $\epsilon$-greedy approach:
  - Usually follow the current policy.
  - But with probability $\epsilon$, choose some other action.

Reinforcement Learning (RL)

- RL determines the optimal action for the system to perform in each state, according to a cost function.
- RL works by estimating the cost $Q(s,a)$ for each $s,a$.

Flight Information Task: 14 variables:

- outputP: whether any flights have been given to the user
- Npeople: number of people has returned from the database, quantized into 5 groups
- controll: whether there is still control
- database: whether the database has been queried
- known: whether we know if parameter can be relaxed, and what the answer is
- parameter: whether parameter has been given by the user etc.

Keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
- keeping the state space small
- reducing the number of state variables
- reducing the numbers of state variables
- keeping the state space integral, small is possible by:
Overview

- Evaluation
- Combine IS and RL
- IS \Rightarrow RL
- TASK

Problems with RL

- RL used to learn all preconditions (e.g. Levin et al., 2000)
  - Including "Do not ask a question if you already know the answer"
  - Lumps together effects of system action, user's understanding of action, user's response, system's understanding of user's response
- RL used to choose among a subset of the actions in certain states
  - No formalism for decomposing state transitions
- No formalism for updating RL state variables after each action
- No formalism for specifying which actions to choose from
- Will make it difficult to apply RL to more difficult tasks
Update Rules

Pre: whether departure time can be relaxed

Ask whether departure time can be relaxed.

Pre. departure != ''
departureR == ''

Eff: set nextMove 'askRelax from'
set keepTurn false

Information State: State Variables

Similar to the RL variables, but includes all information

- includes a number of variables to interface to other modules
- include indices when flights were displayed to user
- results indicate actual database results (not just how many)
- from/to, etc., have actual values of parameters if known
- similar to RL variables, but ...
Problem with IS

CSE560 Class RLIS

© P. Heeman, 2003

Control Strategy

- Which sets of rules should be applied when
- Which rules in each set should be applied
- Which sets or rules should be applied when
- Understanding rules are processed first
- All applicable rules are applied
- If system has turn, action rules are processed
- First applicable action rule is applied

Preconditions for each system action?

How does dialogue designer determine the complete set of
Combine IS and RL

- Use RL to learn best action for each state
- Use IS to specify effects of system actions, user simulation, and hand-crafted preconditions
Transitions

- Transitions happen at a coarser granularity than IS
- Group together everything between successive system actions

Implementing the Simulated User

- With IS, system is normally run against actual user
- To enable combining IS and RL, need to produce dialogues
- IS is very general, use it for the simulated user
- Two separate IS instantiations used
  + Run system and user's understanding rules in lock-step
  + User's rules update the user's state variables
  + System's rules update the system's state variables
  + Two parallel IS instantiations used

Re-construct the system's utterance and a simulated user's speech act seen
+ Check who has the turn and run that agent's action rules
+ Check who has the turn and run that agent's action rules
Example

proc Derivatives {agent} {
upvar ::is$agent is
if {$agent == "A"} {
  if {is(From) != ""} {
    set is(FromP) 1
  } else {
    set is(FromP) 0
  }

  if {is(To) = ""} {
    set is(ToP) 1
  } else {
    set is(ToP) 0
  }

  if {is(genre) = ""} {
    set is(genreP) 1
  } else {
    set is(genreP) 0
  }

  upvar ::isSagent isSagent
  proc Derivatives (agent)
}

...
Cost Function

• All other actions use a cost of 1
  - Database query sets the cost of doing the database lookup
  - Output sets the cost of displaying the results
  - Database query sets the cost of doing the database lookup
  - Parameters can keep track of the sum of the costs
• Effects of each system action rule specifies its cost
  - Include IS variables to track the components of the cost
  - RL needs to track the costs incurred in the dialogue

Variables in Preconditions

• RL variables are a subset of the IS variables, so the IS variables are a subset of the RL variables, so the RL variables are

This restriction does not apply to the user rules, as those are not subject to RL.

- Only RL variables can be used in hand-crafted preconditions
- Actions only applicable in part of an RL state
- Actions not applicable in part of an RL state
- Distinct actions not captured in RL
- When happens if hand-crafted preconditions use an IS parameter than the IS states
- This restriction does not apply to the system's understanding

CSE560 Class RLIS: 23 c
P. Heeman, 2003

CSE560 Class RLIS: 24 c
P. Heeman, 2003
Demonstrate advantage of including hand-crafted preconditions

• High information task

Demonstrate that methodology works by applying it to the flight information task

Overview

Evaluation

⇒

Combine IS and RL

IS •

RL •

Task •
Training

- For each set of preconditions
  - Trained 30 dialogue policies
    - Used Q-Learning
    - Used epoch size of 100
    - After certain epochs, ran the learned policy 2500 times strictly according to the policy in order to measure its performance
- Used \( \epsilon \)-greedy method to explore policy space with \( \epsilon \) set at 0.15

Five Sets of Preconditions

- None (Typical RL)
  - Capture conditions under which action can be performed felicitously
  - + e.g. Only ask the value of a parameter if you do not know its value

- Speech Act
  - + e.g. Only output data once: once data is output, end conversation

- Application Restrictions
  - + e.g. ask the 'to', 'from', and 'departure' parameters first

- Partial Strategy
  - + e.g. Add additional constraints that seem reasonable

- Baseline (Typical IS)
  - + e.g. Add additional constraints to enforce the specification of the application
More constrained precondition conditions dramatically reduce the number of states, and the size of the policy search space.

- Large search space in 'speech acts' and 'none' - Search strategy or task formulation was not adequate to deal with the large search space in 'speech acts' and 'none'.
- Partially probably included some constraints that were not optimal.
- Best result achieved by the 'application' preconditions, which was 17.17.

All four conditions achieved average cost less than the baseline.

<table>
<thead>
<tr>
<th>Policies (log$^{10}$)</th>
<th>States Explored</th>
<th>Dialogue Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.58, 1.95</td>
<td>16.95</td>
<td>5.00</td>
</tr>
<tr>
<td>1.56, 1.94</td>
<td>16.95</td>
<td>5.00</td>
</tr>
<tr>
<td>1.58, 1.96</td>
<td>16.95</td>
<td>5.00</td>
</tr>
<tr>
<td>1.58, 1.95</td>
<td>16.95</td>
<td>5.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dialogue Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.95</td>
</tr>
</tbody>
</table>

Cost & Search Size Table
Conclusion

- Demonstrated how RL and IS can be combined
  - Use IS variables to track action costs
  - Only use RL variables in system action preconditions
  - Introduced derived variables to simplify writing action effects
  - Merge state variables
  - RL uses coarser transitions than IS
  - Formulate user simulation in IS

Average dialogue cost versus epochs

- For 'none', it takes 3700 epochs to achieve a cost less than 17.0
- For 'partial', takes 10 epochs to achieve a cost less than 17.0
- More constrained preconditions result in RL finding a good solution more quickly

More constrained preconditions result in RL finding a good solution more quickly
Conclusion II

• For RL, allows rich formalism to specify effects of system actions to specify user simulation to include hand-crafted preconditions.

• For IS, not all preconditions need to be hand-crafted.
  - RL can determine remaining preconditions that are optimal.
  - This will allow us to use RL to solve very difficult problems.
  - To include hand-crafted preconditions.
  - To specify user simulation.
  - To specify effects of system actions.

• For RL, allows rich formalism of IS.