Introduction

- Reinforcement learning to set up the system state
- Ask parameters
- System needs to determine when and how to confirm and when order to
- Might be some speech recognition errors
- Small number of slots than system needs to find out what value user wants
- Mainly been used for system-initiated form-filling dialogs
- RL is becoming a popular tool for building dialogue managers

Conclusion

- Experiments
- Modeling strategy
- Encoding in RL
- Encoding in IS
- Furniture layout task
- Related work
- Reinforcement learning

Overview
Overview

• Introduction
  ⇒ Reinforcement Learning
• Related Work
  ← Reinforcement Learning
• Furniture Layout Task
• Encoding in IS
• Encoding in RL
• Modeling Strategy
• Experiments
• Conclusion

Goal of Talk

- Use lessons learned from artificial task to help with real tasks
- To explore this, we will use an artificial task with artificial users
  + How do we avoid MDP violations?
  - How do we formulate the state so that RL can learn an optimal policy?
- But how do we set up the RL state?
- Expand use of RL to more complex dialogue tasks, such as exchange information in order to decide on a good solution
  negotiation dialogues, in which the system and the user need to
In order to ensure that RL is able to find an optimal solution, states must satisfy the two MDP assumptions:

1. Markov Decision Process Assumption

- Probability of the next state depends only on the previous state and on what action was taken in that state:
  \[ P(s_{i+1}|s_i, a_i) = P(s_{i+1}|s_1, a_1, \ldots, s_i, a_i) \]

How Should RL State be Crafted?

- Ensure the number of state-action pairs does not explode
  - Keep the number of RL states as small as possible
  - Fewer states means there will be fewer Q values to estimate, which should allow the optimal policy to be learned with less training

- RL state space cannot be too small
  - RL needs to determine the best action for each state
  - Each state can only specify a single action, so each state needs to be described enough so that the same action is appropriate for the entire state
  - We refer to these as action-decision variables
  - Dialogue designers can often identify these variables when setting up a dialogue problem for RL

We feel that dialogue designers can often identify these variables when setting up a dialogue problem for RL.
Introduction

Reinforcement Learning

Furniture Layout Task

Encoding in IS

Encoding in RL

Modeling Strategy

Experiments

Conclusions

Overview

2nd Markov Decision Process Assumption

\[ \Pr(c_i | s_i a_i) = \Pr(c_i | s_1 a_1 \ldots s_i a_i) \]

- The cost of transitioning to the next state only depends on the previous state and action, and not on any of the previous states or actions.

- We refer to any variable that is needed to meet the second MDP assumption, other than the action-decision variables, as a bookkeeping variable.

- We refer to any variable that is needed to meet the second MDP assumption.

- Must include these variables to make sure RL converges, but these variables are very difficult for dialogue designers especially for negotiation dialogues.

Thus, if two sequences of actions, say \( a_1 \ldots a_i \) and \( a'_1 \ldots a'_i \), have different expected costs to get to the finish state, the two action sequences should result in two different states.

\[ \Pr(c_i | s_i a_i) = \Pr(c_i | s_1 a_1 \ldots s_i a_i) \]

- Cost of transitioning to the next state only depends on the previous state and action.
Learning Strategy

Walkcr (2000) learned policy for system for reading email

- **Read**
  - Read-first email, only read summaries, ask user whether to use system-initiative (requesting specific pieces of information one at a time) or mixed-initiative (asking an open-ended question to the user)
  - System learns how to interact with user over a number of states

- **Summarization**
  - Summarize both sender and subject, summarize just one based on context
  - System learns how to interact with user

- **Initiative**
  - Whether to use system-initiative to use specific pieces of information or mixed-initiative

- **Walker (2000)** learned policy for system for reading email

Form-Filling Dialogues

- Number of researchers have applied RL to form-filling
  - Levin, Pieraccini, Eckert (2000); Scheffler and Young, 2002; Georgila, Henderson and Lemon, 2005

- For each slot in the form, the cost at the end of the dialogue only depends on the value of the slot. Is the user adamant about its value?
  - Does system know the value?
  - Has value been confirmed?

- With such a simple representation, good policies can be learned.
  - Variables needed for the RL state are simply action-decision variables
  - No bookkeeping variables are needed
Overview

Introduction

Reinforcement Learning

Related Work

Modeling Strategy

Experiments

Conclusion

What is interesting about Walker's work?
- System learns choices for components of the dialogue strategy at the first utterance that the component would effect
- Choice remembered in RL state and constrains system's future actions
- System learns choices for components of the dialogue strategy at the first utterance that the component would effect
- System learns choices for components of the dialogue strategy at the first utterance that the component would effect

Our work will generalize this approach without having to hand-code so much of the system's behavior.
How can the agents solve the task?

- The agents can do the following:
- If no item has been proposed, either agent can PROPOSE an item, which makes that item the proposal.
- If there is a proposal, either agent can ACCEPT it. Accepting an item includes it in the solution and removes it as a proposal. Preference becomes mutually known by each agent.
- Either agent can REJECT proposal.
- If there is no proposal, either agent can PROPOSE an item, which makes that item the proposal.
- If no item has been proposed, either agent can PROPOSE an item, which makes that item the proposal.
- Allows agents to do multiple actions in a row, e.g., inform followed by a reject.
- Agent who has the turn can RELEASE TURN, allowing agents to do multiple actions in a row, e.g., inform followed by a reject.
- Access to intervening items in the solution, and removing items from the proposal, can be done by the non-proposer.
- E.g., if there is a red couch, there should also be a lamp.
- Each has private preferences about which items to place in room.
- Both know all items, which differ by color, type, and value (1-20).
- Two agents must agree on 5 furniture items to place in room, which makes that item the proposal.
- Complex task that requires agents to share information and preferences of other agents.

Furniture Layout Task
Domain Reasoning

- Which item can be proposed?
  - Let's just hard-code this rather than trying to learn it.
  - Agents use a greedy algorithm.
- Picks item that results in lowest score when added to current plan.
- Score is computed as the sum of the items' values less any penalties from any violated preferences.
- Agent determines which item it would have proposed instead.
- Let's be solution so far with the proposed item.
- Used to argue against current proposal.
- Which preference to inform?
  - Used to argue against current proposal.
  - Let $S$ be solution so far with the proposed item.
  - Agent determines which item it would have proposed instead.
  - Let $S'$ be solution so far with the proposed item.
  - Checks each preference unknown to the other agent, ranked by penalty.
- Preference violation by $S$ but not by $S'$?
- Size of the penalty can be used to decide whether to inform or not.
- Is preference violated by $S$ but not by $S'$?
- Which preference to inform?

Collaborative Framework

- Goal has subgoals.
- Each has knowledge about the goals/subgoals.
- Once a subgoal is accepted, can not be re-negotiated.
- Reason plans become part of the common knowledge.
- Reason can accept it or give a reason against it.
- One proposes a solution to a subgoal.
- Each has knowledge about the goals/subgoals.
- Indicative of how natural negotiation happens.
- Where user knows where they want to travel to.
- Where user knows how much they want to travel to.
- Where system knows where they are located, type of food, reviews, etc.
- Finding a restaurant.
- Where system knows what type of food, cost, reviews.
Possible System Strategies

- Use a weighted average of dialogue length (with weight 0.15) and solution quality (weight -1.0)

Which is best?
- Randomly pick an action
- If first violated preference has penalty less than 5, then accept, otherwise reject
- Only inform of the first violated preference, which will have the greatest penalty
- If any violated preferences inform all of them, and then reject
- Accept everything

Additional Restrictions

- The user always goes first
- Only the system proposes and accepts
- Only the system informs
- System can only reject if it has made at least one inform
- So as to avoid infinite loops

We purposely made the behavior of the user be simple so as to focus on learning the system behavior.
Information State Variables

• Information state of each agent also includes:
  + WVPREF: used by non-proposed to indicate worst violated preference.
  + BETTER: If other agent proposed, indicates which item agent thinks is better.
  + BEST: If no proposed item, indicates which item agent thinks is best to add.

Domain Reasoning updates the following:
  + INFORMED: Tracks whether any violated preferences have been communicated.
  + PROPOSED UNDER WHO PROPOSED it.
  + PROPOSED TRACKS WHICH ITEM HAS BEEN PROPOSED.
  + SHARED PREFERENCES TO TRACK THE SHARED PREFERENCES.
  + MYPREFS TO TRACK THE AGENT'S PRIVATE PREFERENCES.
  + SOLUTION TO TRACK THE ITEMS THAT HAVE BEEN AGREED UPON.

Information state of each agent also includes:

Overview

• Conclusion
• Experiments
• Modeling Strategy
• Encoding in RL
• Encoding in IS
  = Furniture Layout Task
• Related Work
• Reinforcement Learning
• Introduction
### Hand-Crafted Policies

- Hand-crafted several system policies
  - Uncollaborative: the system always accepts what the user proposed
  - Random: the system randomly chooses an applicable action
  - Inform if \( \geq 9 \): inform if worst violated preference is \( \geq 9 \).
  
  Otherwise, if we have informed, reject. Otherwise accept

- Tested them with 1,000,000 different task setups
  - Created 15 furniture items, randomly assigning type, color, and value
  - Created 15 preferences for each agent, involving the 15 items

<table>
<thead>
<tr>
<th>Dialogue Length</th>
<th>Solution Quality</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncollaborative</td>
<td>3.30</td>
<td>27.38</td>
</tr>
<tr>
<td>Random</td>
<td>8.72</td>
<td>19.61</td>
</tr>
<tr>
<td>Inform if ( \geq 9 )</td>
<td>7.85</td>
<td>3.95</td>
</tr>
</tbody>
</table>

### Update Rules

- 6 action selection rules
  - \( \text{PROPOSE} \), \( \text{ACCEPT} \), \( \text{REJECT} \), \( \text{INFORM} \), \( \text{RELEASE} \), and \( \text{FINISH} \)
  - Each has preconditions to ensure only used in correct conditions
  - \( \text{PROPOSE} \) can only be done if there is not an item already proposed
  - \( \text{INFORM} \) can only be done if there is an item

- Also have several deliberation rules
  - Computed the values of \( \text{BEST} \), \( \text{BETTER} \), and \( \text{WORST} \)
  - Invoked after the understanding rules

- Also have understanding rules
  - Update the information state variables based on what was last said

- Each has effects that indicate how the information state changes

- Each has preconditions to ensure only used in correct conditions

### Dialogue

- Created 25 preferences for each agent, involving the 15 items
- Created 15 furniture items, randomly assigning type, color, and value
- Tested them with 1,000,000 different task setups

- Random: the system randomly chooses an applicable action
- Inform if \( \geq 9 \): the system always rejects whenever the worst violated preference is \( \geq 9 \).

- Otherwise, if we have informed, reject. Otherwise accept
Use a subset of the IS variables as our RL state.

Encoding in RL

These variables are all potential action-decision variables:

- Include DONE, which indicates whether the agents have completed the task (by accepting the required number of items).
- Include PROPOSED and BETTER: PROPOSED indicates if anything has been proposed, and BETTER indicates if a better item has been found.
- Include boolean versions of PROPOSED and BETTER: PROPOSED.
- Include INFORMED and PROPOSED.
- Include a variant of WVP: WVPREF: WVPREF Q indicates the penalty of the worst violated preference quantized into 4 groups: 0, 1-3, 4-8, 9-20.
- Include D DONE, which indicates whether the agents have completed the task (by accepting the required number of items).

Overview

- Conclusion
- Experiments
- Modeling Strategy
- Encoding in IS
- Encoding in RL
- Furniture Layout Task
- Related Work
- Reinforcement Learning
- Introduction
Introduction

Reinforcement Learning

Related Work

Furniture Layout Task

Encoding in IS

Encoding in RL

Modeling Strategy

Experiments

Conclusion

Number of Agreed Items

The number of items that are still remaining to decide is an important factor of the cost to the final state.

Hence, we also include in our RL state the number of items agreed to so far, $N_{AGREED}$.

How many items have been agreed to so far should not affect how we agree to the next one.

Hence, we also include in our RL state the number of items agreed to so far, $N_{AGREED}$.

This is not an action-decision variable.

It is a bookkeeping variable.

Agree to the next one.

The number of items that are still remaining to decide is an important factor of the cost to the final state.

Hence, we also include in our RL state the number of items agreed to so far, $N_{AGREED}$.

How many items have been agreed to so far should not affect how we agree to the next one.

Hence, we also include in our RL state the number of items agreed to so far, $N_{AGREED}$.
Possible Answers

Option 1: Keep track of all actions that were done, and from which each action originated.

- But this would explode the number of states.
- Future actions would be chosen in agreement with selected strategies.
- Enumerating strategies.
- But, as we apply RL to more complex domains, it might not be feasible to enumerate all strategies.
- As the number of states increases, even if we use a completely separate subject to forget the initial state, the number of states will explode.
- Also, each strategy will use a completely separate subject of states. This means all strategies will be independent.

Option 2: Enumerate all possible strategies for completing the task.

- At the beginning of the dialogue, have an action that sets an RL variable to one of the enumerated strategies.
- Future actions are chosen in accordance with the selected strategy.

But, RL does not keep track of how the solution is formed.

In the furniture domain, does RL have the appropriate bookkeeping variables?

- Immediate state to the final state.
- Which of these two strategies is used will result in different average solution qualities, and so will affect the cost from any previous solution qualities.
- Proposal, inform the user, and reject the proposal.
- (a) If there are any uncommunicated violated preferences for the current proposal, inform the user, and reject the proposal.
- (b) If there are any uncommunicated violated preferences for the current proposal, inform the user, and reject the proposal.
- Consider the following strategies.
- Do they capture the process by which the system arrives at a solution?
- Does the RL state have the appropriate bookkeeping variables?
result in a bad solution. Our use of the STRATEGY variable allows RL to reward action sequences that result in a good solution and punish those that do not. The order that actions were performed is not recorded, as the STRATEGY variable is an unordered set. The lack of order is important because the RL state is part of the RL state. Each time the system makes a choice, keep track of a condensed version of the current situation. This version is helpful for understanding the consequences of each action. Our approach is to record the past.
Implications of Constraining Future Actions

- Our approach will preclude it from finding such a solution.
- We are restricting the number of policies that are considered.
- As we are restricting what actions the system can take, the number of decisions that we need to track is also much less, which means fewer decisions.
- Process by which it accepts one item should be the same as for the others.
- Potential limitation is that the optimal system policy might require it to behave differently for accepting one item than it does for another item.
- Our approach of constraining the future actions forces the system to behave in a consistent manner.
- A lot of states will only have one single action that is applicable, so fewer.
- We are restricting number of policies that are considered.

Example:

- System must inform the user of the violated preference.
- User proposes item with a worst violated preference of 4
- Strategy is ACCEPT (1,0) REJECT (1,1) INFORM (4,0)

In the dialogue, it will do the action that it previously did.
If the system sees a situation that it already encountered earlier, it does so too. In the dialogue, it also constrains what the system has done so far in the dialogue.
STRATEGY variable not just records a history of what the system
For the furniture task, we use 4 rules for deducing additional constraints for the STRATEGY variable:

- If INFORM(W, I) is in STRATEGY, add INFORM(X, I) for all x > \( W \).
- If ACCEPT(W, I) is in STRATEGY, add ACCEPT(X, I) for all x < \( W \).
- If REJECT(W, I) is in STRATEGY, add REJECT(Y, I) for all X s.t. \( W \leq X < Y \).
- If INFORM(W, 1) is in STRATEGY, add INFORM(W, 0).

Example:

- If the system makes sense to accept when the worse violated preference is 1, it should also make sense to inform when the worse violated preference is 9.
- If the system makes sense to accept when the worse violated preference is 4, it should also make sense to inform of a penalty of 4.
- If the system has an uncommunicated violated preference against the item of penalty 4, it should also propose an item.
- If the system makes a certain action in one type of situation, it makes sense in constructing the value of STRATEGY.
Implications of Inferences

- Inferences are knowledge that the system should be able to discover with RL
- Reduces the number of states
  - Some values of STRATEGY no longer possible
- Reduces the number of applicable actions for some states
- Due to inconsistency reasoning
- Use of inferences highlights that the STRATEGY variable does not just have to be a record of what occurred in the dialogue, but can capture the strategy in a compositional way from individual system actions

Reasoning about Inferences

- Does not slow simulation as we cache the applicable actions for a given state
- We incorporate sophisticated checks in the preconditions to make sure this does not happen
- As this would imply it should have informed for worst violated preferences
- Example: what is already in the strategy
  - If you do an action, inferences that it allows might not be consistent with action is applicable
  - Inferences complicate how we determine whether a system

- Inferences complicate how we determine whether a system
Experiments

- For training:
  - We use Q-learning with $\epsilon$-greedy to explore the state-action pair space, with $\epsilon$ at 20%.
  - New training experiences are given a weight of $1/\sqrt{c}$, where $c$ is the number of times the state-action pair has been visited.
  - We group training into epochs of 100 dialogue runs; after each epoch, a new policy is determined.
  - Learn 12 policies for 200,000 epochs (way more than is actually needed).
  - After every 1000 epochs (and after 10, 100, and 500 epochs), we test the policy with 5,000 dialogue runs.
  - We use 12 training weights $c$-greedy to explore the state-action pair space.
  - Average the results over the 12 learned policies.
  - For testing:
    - After every 1000 epochs (and after 10, 100, and 500 epochs), we test the policy with 5,000 dialogue runs.
    - We always follow the action in the policy, with no exploration.
    - Average the results over the 12 learned policies.

Conclusion

Experimental Strategy

Encoding in RL

Encoding in IS

Furniture Layout Task

Related Work

Reinforcement Learning

Introduction
How Does State-Space Grow with Problem Size?

- Altered number problem size from 1 to 5 items

States and State-Space both grow linearly

- Initial part of the curve is not linear because some values of STRATEGY do not occur when the number of items that has been agreed to so far

- States and State-Space both grow linearly

Altered number problem size from 1 to 5 items

Can We Learn a Good Policy?

- Use Full RL State

Does as well as best hand-crafted solution

<table>
<thead>
<tr>
<th>States</th>
<th>NumAgreed</th>
<th>Strategy</th>
<th>State-Actions</th>
<th>Dialogue Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>34,838</td>
<td>51</td>
<td>With</td>
<td>30.716</td>
<td>11.825</td>
</tr>
<tr>
<td>81,883</td>
<td>101</td>
<td>Without</td>
<td>12.086</td>
<td>11.822</td>
</tr>
<tr>
<td>14,087</td>
<td>35</td>
<td>With</td>
<td>40.699</td>
<td>34.438</td>
</tr>
<tr>
<td>19,833</td>
<td>75</td>
<td>Without</td>
<td>8.388</td>
<td>12.067</td>
</tr>
</tbody>
</table>

- Include STRATEGY and NumAgreed variables and inferences

- Proposes

- Use Full RL State
Removing Inferences (continued)

- Takes longer to converge, as expected

- How does learning progress?

- Modified STRATEGY variable to remove inferences
  - So must separately learn what to do for each preference value
  - Thus, more states and state-action pairs are needed
  - But more can still learn optimal policy

- Without inferences:
  - RL can still learn optimal policy

<table>
<thead>
<tr>
<th>State Agreed</th>
<th>States</th>
<th>State-Actions</th>
<th>Dialogue Cost</th>
<th>State-Actions</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Inferences</td>
<td>34,838</td>
<td>40,629</td>
<td>11.825</td>
<td>99.46</td>
<td>34.838</td>
</tr>
<tr>
<td>Without Inferences</td>
<td>81,883</td>
<td>99,946</td>
<td>11.822</td>
<td>99.46</td>
<td>34.838</td>
</tr>
<tr>
<td>Proposed</td>
<td>14,087</td>
<td>16,086</td>
<td>12.067</td>
<td>81.883</td>
<td>34.838</td>
</tr>
</tbody>
</table>

- More policies are searched over
- So must separately learn what to do for each preference value

Removing Inferences
Removing the NumAgreed Variable (continued)

- How does learning progress?

Without NumAgreed variable, RL cannot estimate the number of utterances to finish the problem, which prohibits it from learning a good policy.

- Substantial reduction in number of states and state-action pairs:
  - They no longer increase as the problem size increases.
  - This should make it easier to learn a good solution.

<table>
<thead>
<tr>
<th>Dialogue Cost</th>
<th>State-Actions</th>
<th>State Inferences</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.837</td>
<td>14,087</td>
<td>14,087</td>
<td>Without</td>
</tr>
<tr>
<td>11.825</td>
<td>34,838</td>
<td>51</td>
<td>Without</td>
</tr>
<tr>
<td>11.822</td>
<td>81,883</td>
<td>51</td>
<td>With</td>
</tr>
<tr>
<td>12.067</td>
<td>40,629</td>
<td>101</td>
<td>With</td>
</tr>
</tbody>
</table>

- Including STRATEGY and inferences, but exclude NumAgreed variable.
Removing the Strategy Variable (continued)

• Policies start off similar to random policies, and get worse!

• Without the STRATEGY variable, RL does not take into account the process by which the conversants agree to items in the plan.

• Learns the uncollaborative hand-coded policy.

How does learning progress?

Removing the Strategy Variable

How does learning progress?

- Faster learning
- Fewer probability distributions to estimate, which usually leads to faster learning

- Reduces number of states and state-action pairs dramatically

- But, we are unable to learn a good policy

Table:

<table>
<thead>
<tr>
<th>STRATEGY</th>
<th>NumAgreed</th>
<th>State-Actions</th>
<th>Dialogue Cost</th>
<th>State-Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without</td>
<td>34,838</td>
<td>40,629</td>
<td>11.825</td>
<td>34,838</td>
</tr>
<tr>
<td>With</td>
<td>81,883</td>
<td>99,946</td>
<td>11.822</td>
<td>40,629</td>
</tr>
<tr>
<td></td>
<td>14,087</td>
<td>16,086</td>
<td>12.067</td>
<td>16,086</td>
</tr>
</tbody>
</table>

- Removed STRATEGY variable from the RL state, but included

- NUMAGREED
Conclusion

• Showed how to construct the RL state for a negotiation task with actual users. Although this work was on an artificial negotiation task, we feel that the manner in which the RL state tracks how negotiation occurs can be applied to real negotiation tasks with actual users.

- Also included NUMAVERED to track how far we are in the negotiation process by which the system negotiates with the user.
- Used STRATEGIC to track how far we are in the negotiation process by which the key decisions were made.
- Also included NUMAVERED to track how far we are in the negotiation process by which the key decisions were made.
- Also included NUMAVERED to track how far we are in the negotiation process by which the key decisions were made.

- Also included NUMAVERED to track how far we are in the negotiation process by which the key decisions were made.

- Also included NUMAVERED to track how far we are in the negotiation process by which the key decisions were made.

Overview

Introduction

Reinforcement Learning

Modeling Strategy

Encoding in RL

Experiments

Encoding in IRA

Furniture Layout Task

Related Work

Conclusion