

Chapter 3. Reinforcement Learning

in RL for Adaptive Dialogue Systems, V. Rieser & O. Lemon

Course: Autonomous Machine Learning

Patrick Emaase

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

<http://bi.snu.ac.kr>

How can we develop an Autonomous Learning Dialogue Systems?

What concepts, behavior representation do we need to consider?

Summary

- Learning – Interaction between agent and world
 - Percepts received by an agent acts and improves agent's ability to behave optimally in the future to achieve the goal
- Reinforcement Learning – Achieve goal successfully
 - Learn how to behave successfully to achieve a goal while interacting with external environment, Learn via experience
 - Game playing – know when its win or loss
 - Well suited for dialogue strategy development – as dialogue is learned by evaluative feedback with delayed rewards and exploration

Summary

■ Scope:

- RL in dialogue development – Skills {↑, ↓}
- Empirical Justification – Why, Hence..
- Dialogue Simulation – How to generalize

■ Peter Rabbit – Rewards known

- Mischievous and disobedient - Exploratory
- Chased, escapes, rests, regrets
- Obedience: Sumptuous meal, mother's love
- Disobedience: Losses clothes, stomach ache



Peter Rabbit: An agent receiving rewards

■ RL: Objectives

- Model dialogue as a sequence of action (in global point-wise estimates)
- Mimic behaviour observed in non-stationary corpus and explore new strategies

Contents

- Nature of Dialogue Interactions
 - Dialogue is Temporal
 - Dialogue is Dynamic
- Reinforcement Learning Based Dialogue Strategy Learning
 - Dialogue as a Markov Decision Process
 - Reinforcement in Learning Problem
 - Model based vs Simulation based Strategy Learning
- Dialogue Simulation
 - Wizard-of-Oz Studies
 - Computer-based Simulations
 - Discussion
- Application Domains
- Conclusion

Nature of Dialogue Interaction

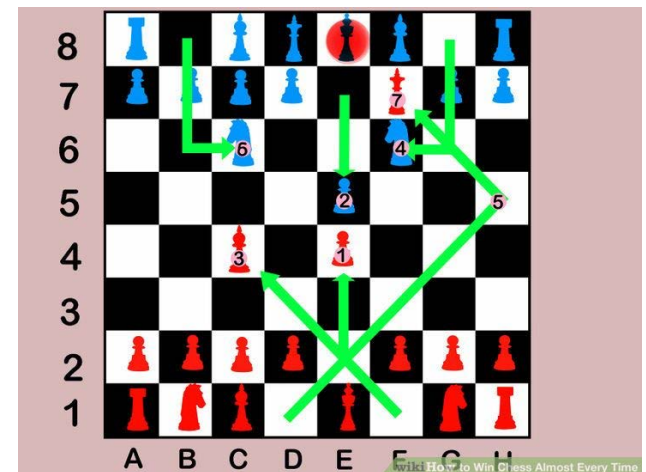
■ Dialogue is Temporal

- Goodness of action – depends on dialogue progress; planning nec.
 - Actions affects the state, options and opportunities
- ∴ SL not suitable for dialogue strategy; potential for “multi-expert” learning

■ RL – Sequential decisions process

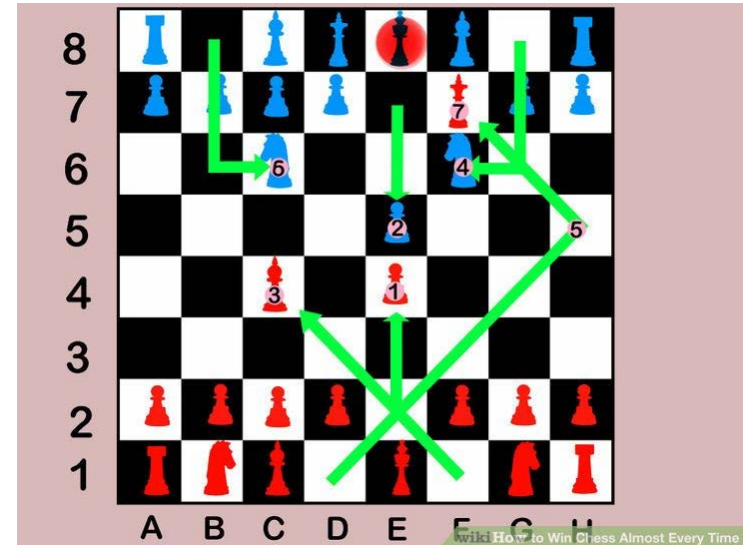
- Based on delayed rewards (benefits apparent at the end of the dialogue; avoids local minima)

■ RL – Chess player may sacrifice pawn for promising strategy at the long-run



Nature of Dialogue Interaction

- Dialogue is Dynamic
 - Interaction in stochastic environment – Dynamic {conditions change, differential reaction by agents - unpredictable}. Need robust strategy
 - ∴ chess players strategize – think ahead {sequence of action choices}
 - Good player dynamically explores; winning – rewarded; losing – punished
 - Language Learners – improve communicative skills over time {encouragement, correction}



In conclusion

- Reinforcement Learning:

- Learns by exploration – learning robust strategies, appropriate for unseen states
- Learns by experience
- Simulation Based RL – Ensures enough exploration
- Explores most strategies at low cost

RL – based Dialogue Strategy Learning

■ Dialogue as a Markov Decision Process

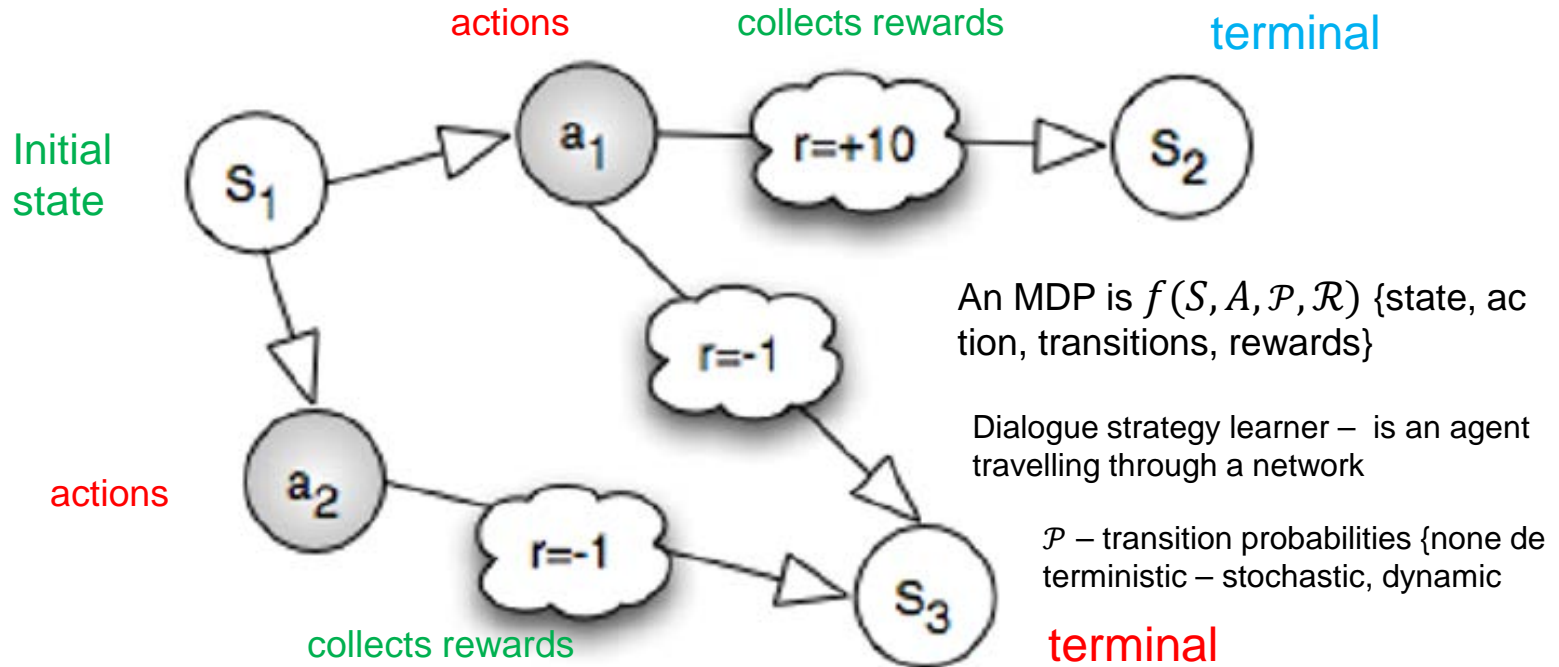


Fig. 3.2 RL with Markov Decision Process (MDP): The learning agent travels through a network of interconnected states. At time t , the agent is in state s_t , takes an action a_t , transitions into state s_{t+1} according to the transition probability $p(s_{t+1}|s_t, a_t)$, and receives rewards r_{t+1}

RL – based Dialogue Strategy Learning

- Specialized MDP – accounts temporal nature of dialogue
 - **Markov Property** requires that the **state** and **reward** at time $t + 1$ only depends on the **state** and **action** at time t .
 - State at step t information available to agent about its environment; Summarize past sensations, retains all “essential” info
 - **Markov Property:**

$$P(s_{t+1}, r_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_0, a_0) \approx P(s_{t+1}, r_{t+1} | s_t, a_t) \quad (3.1)$$

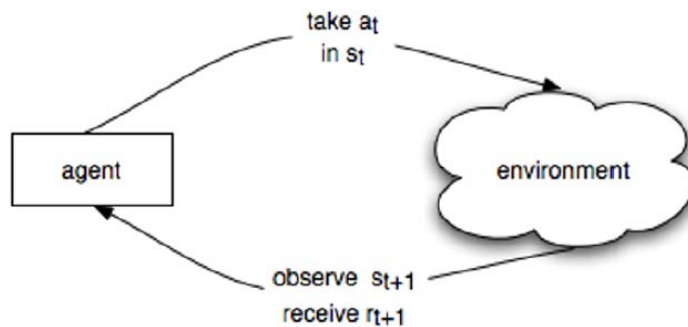


Fig. 3.2 The agent interacting with a stochastic environment and actively influencing its behavior by taking action a_t in state s_t . The changes in the environment are observed (o_{t+1}) and a reward is received r_{t+1}

Dialogue as a MDP

- **State space (S) – reachable states for agent**
 - Dialogue features – knowledge about dialogue history (\check{s}_d), user input action (\check{a}_u) [e.g. confidence values], and task level features (\check{s}_u)
- **Action set (A) – All actions available to agent**
 - Dialogue actions to be learned represented as abstract semantic Speech Acts on the intentional level
- Reward critical for **learning**

Dialogue as a MDP (2)

■ State transition function (\mathcal{P}) – dynamics of environment

- Next state $s' \in S$ is likely to follow when taking action $a \in A$ in states $s \in S$.
 \mathcal{P} is defined over $\mathcal{P}: S \times A \times S \rightarrow [0,1]$ where:

$$\mathcal{P}_{ss'}^a = P(s_{t+1} = s' | a_t = a, s_t = s); \quad (3.2)$$

■ Reward function – Reward for a decision

- For state s_t and action a_t expected reward value:

$$\mathcal{R}_{ss'}^a = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s); \quad (3.3)$$

Partially Observable MDP (POMDP)

■ History and State

- **History** – Sequence of observations, actions, rewards
 - $H_t = O_1, R_1, A_1, \dots, O_{t-1}, R_t, A_t$ {all observable variables up to time t }
- State – information used to determine what happens next
$$S_t = f(H_t)$$

■ Information state – (Markov state) contains useful info

- **Environment State** S_t^e – Environment's private representation
 - Data for picking next observation/reward
- **Agent State** S_t^a - agent's internal representation
 - Info agent uses to pick next action, used by RL algorithm
- Can be any function of history
$$S_t^a = f(H_t)$$

Partially Observable MDP (POMDP) (2)

■ Definition

- A state S_t is **Markov** if and only if

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

“The future is independent of the past given the present”

- Once the state is known, the history may be thrown away – The state is a sufficient statistic of the future
- The environment state S_t^e is Markov, the history H_t is Markov

Partially Observable MDP (POMDP) (3)

- MDP – entire space is fully observable
 - Uncertainty represented as state feature encoding [low, high; or confirmed or unconfirmed]
- Fully observability: Agent directly observes the environment state;
 $O_t = S_t^a = S_t^e$; Agent state = environment state = information state
- Partial Observability: agent indirectly observes environment
 - Observes current state, agent state \neq environment state
- In POMDP - Agent constructs its own state representation S_t^a providing
 - Complete History: $S_t^a = H_t$
 - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], \dots, \mathbb{P}[S_t^e = s^n])$
 - Recurrent neural network: $S_t^a = \sigma(S_{t+1}^a W_s + O_t W_0)$

POMDP – Belief Monitoring

■ Belief Monitoring –

- POMDP encodes uncertainty by representing current dialogue state s as a **belief state $b(s)$** – **distribution of the possible states**
- Belief Monitoring - Update belief state based current observation o

■ Belief state update:

$$b'(s') = P(s'|o', a, b(s)) = k \times P(o'|s', a_s) \sum_{s \in \mathcal{S}} P(s'|a, s) b(s); \quad (3.4)$$

- $b'(s')$ = estimated belief state, $P(s'|o', a, b)$ = probability of being in a state s' given observation o' , the user action a and the current belief state $b(s)$.
- Re-written as probability of observing o' in state s' and given a system action a_s , given transition probability for current belief state to the new state s , k is normalization constant
- **POMDP**: Can track **multiple hypotheses simultaneously**; fast **backtrack** and **correct** errors; User's goal info **accumulates** over dialogue turns; **scaling up problem – computationally very expensive and intractable for dialogue system**
- **MDP** – Loses alternative hypothesis info; complex in error discovery and correction
 - Approximation possible

Reinforcement Learning problem

- MDP – allows dialogue management strategy (policy) {agent's behavior} to map state to action $\pi: S \rightarrow A$
- Elements of RL – {Policy, action, Reward, discount factor}
 - **Policy π – Selections action of highest rewards during a dialogue**
 - Deterministic policy - $a = \pi(s)$
 - Stochastic policy - $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
 - The final Reward \mathcal{R} - total discounted return received from time t . Discount factor γ – weights rewards {immediate rewards 0; further future -1}; $\gamma=0$ RL maximizes the immediate utility; $\gamma=1$ takes into consideration future rewards.

$$\mathcal{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^T \gamma^k r_{t+k+1} \quad (3.5)$$

- Reward \mathcal{R}_t - intrinsic desirability of a state or action; policy discovered via trial-and-error search through interaction btw. learning agent and its dynamic environment

Reinforcement Learning problem (2)

■ Value Function $V^\pi(s)$

- Long term desirability of a state considering all likely subsequent states
- V – value of a state; is future expected reward for visiting states s following policy π subsequently

$$V^\pi(s) = E_\pi(\mathcal{R} | s_t = s) \quad (3.6)$$

- Value function V is a prediction of future reward; evaluates goodness/badness of a state; selects actions

■ Q-function $Q^\pi(s, a)$ - Expected return

- The value function can be re-written as the Q-function $Q^\pi(s, a)$ - is expected return for taking action a in a given state s and following policy π thereafter

$$Q^\pi(s, a) = E_\pi(\mathcal{R} | s_t = s, a_t = a) \quad (3.7)$$

- $V^\pi(s)$ and $Q^\pi(s, a)$ can be formulated recursively using Bellman equations

Bellman Equations

- Using equation 3.2 and 3.3 the resulting equations are Bellman's equations

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} \mathcal{T}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')] \quad (3.8)$$

$$Q^\pi(s, a) = \sum_{s'} \mathcal{T}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^\pi(s')] \quad (3.9)$$

- Bellman equations describe the expected reward for taking action prescribed by policy π . The equations for the optimal policy π^* are referred to as Bellman optimality equations:

$$V^*(s) = \max_a \sum_{s'} \mathcal{T}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^*(s')] \quad (3.10)$$

$$Q^*(s, a) = \sum_{s'} \mathcal{T}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma \max_a Q^*(s', a)] \quad (3.11)$$

- Finding an optimal policy by solving the Bellman Optimality Equations requires accurate knowledge of the environment dynamics, time and space

Summary

- The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State - value function for policy π :

$$V^\pi(s) = E_\pi \left\{ R_t \mid s_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}$$

- The **value of taking an action in a state under policy π** is the expected return starting from that state, taking that action, and thereafter following π :

Action - value function for policy π :

$$Q^\pi(s, a) = E_\pi \left\{ R_t \mid s_t = s, a_t = a \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}$$

Bellman Equation for a Policy π

The basic idea:

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \square \\ &= r_{t+1} + \gamma (r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \square) \\ &\square\square\square = r_{t+1} + \gamma R_{t+1} \end{aligned}$$

So:

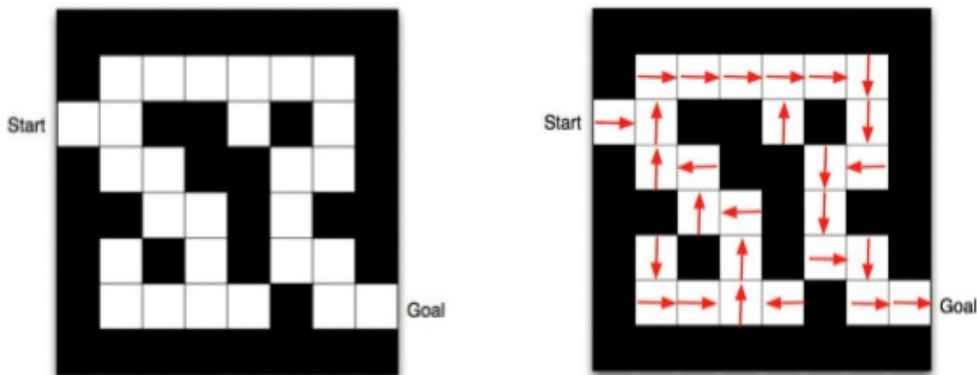
$$\begin{aligned} V^\pi(s) &= E_\pi \{ R_t \mid s_t = s \} \\ &= E_\pi \{ r_{t+1} + \gamma V(s_{t+1}) \mid s_t = s \} \end{aligned}$$

Or, without the expectation operator:

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]]$$

RL Algorithms (1)

Maze example: $r = -1$ per time-step and policy



Actions: N, E, S, W; States: Agents location;
Arrows: policy $\pi(s)$ for each state s .

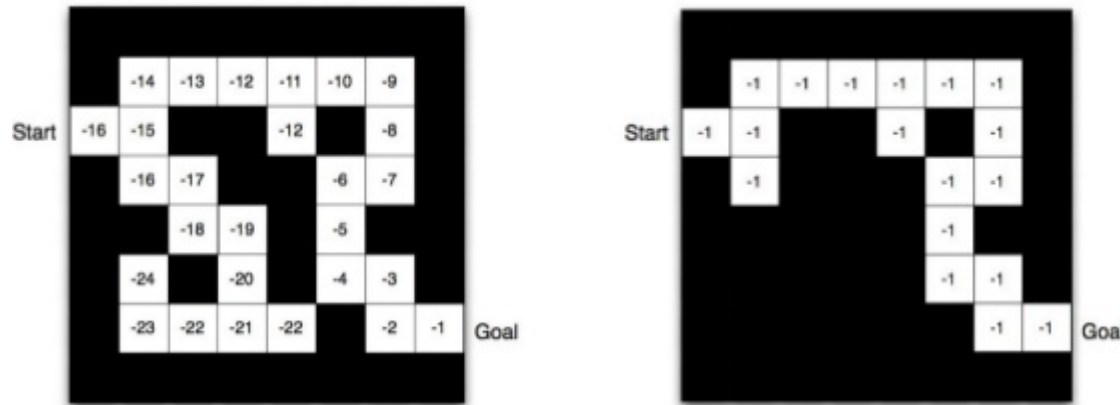
- A **Model** – predicts what the environment will do next
- \mathcal{P} predicts the next states
- \mathcal{R} predicts the next (intermediate) rewards
 - Agent may have an internal model of the environment
 - Dynamics: how actions change the state
 - Rewards: How much reward from each state

■ Categorizing RL agents

- Value Based
 - No policy (implicit)
 - Value function
- Policy Based
 - Policy
 - No Value Function
- Actor Critic model
 - Policy
 - Value Function

RL Algorithms (3)

Maze example: Value function and Model



- Grid layout represents the transition model $\mathcal{P}_{SS'}^a$
- Numbers represents immediate reward \mathcal{R}_S^a

■ Categorizing RL agents

- **Model Based**
 - Policy and/or Value Function; model; Example: **Dynamic Programming** (DP)
- **Model Free**
 - Policy and or Value Function; Model; ex. **Temporal Difference** and **Monte Carlo**

Algorithms for RL (4)

- **Dynamic Programming**
 - Model based approach;
- **Temporal Difference and Monte Carlos**
 - Model-free (simulation) – explicitly models the dynamics of the environment
- **Mechanisms of RL algorithms**
 - Learn by incrementally updating the expected Q-values for each action pairs, estimating the Bellman optimality equation
 - Initialize Q-values to arbitrary value
 - Visualize process as a matrix of states vs actions
 - Update state-action pair Q_k for each iteration k
 - Equation form

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate] \quad (3.12)$$

Where Step-size – Learning rate (α); $[Target - OldEstimate]$ – error to achieve Q^*

Algorithms for RL (5)

	s_1	s_2	s_3	s_4
a_1	0.0	0.0	0.0	0.0
a_2	0.0	0.0	0.0	0.0
a_3	0.0	0.0	0.0	0.0

Initialized state

π^*



Optimal policy selects a with highest expected value in each state s

	s_1	s_2	s_3	s_4
a_1	2.40	1.80	4.05	0.46
a_2	3.67	1.38	2.01	2.78
a_3	0.9	2.90	2.52	1.24

Terminal state

- Stopping criterion: Q-value convergence [difference \leq some threshold]

DP vs TD – Differences

■ DP - Difference based on update of Q-values

- Updates Q-value *off-line for every* possible state action pair in a **single iteration**
- Requires **explicitly model** of the dynamics of the environment
 - Transition function fully defines the probability of moving from state $s \rightarrow s'$
 - Given transition probability and reward functions: $P(s'|s, a); R(s, a)$ we can:

$$Q_t(s, a) \leftarrow R(s, a) + \sum_{s'} P(s'|s, a) \max_a Q_t(s', a') \quad (3.13)$$

■ TP – Model-free

- No need for full model of the transition function
- Requires **some sample episodes** of state transitions; not all
 - Requires online exploration of large state-action pairs
 - Only sampled transitions contribute to improved Q^* ; requires **online exploration**

$$Q_t(s, a) \leftarrow Q_t(s, a) + \alpha \left[\underbrace{R(s, a) + Q_t(s', a')}_{\text{Target}} - \underbrace{Q_t(s, a)}_{\text{OldEstimate}} \right] \quad (3.14)$$

Advantages of TD Learning

- TD methods do not require a model of the environment, only experience
- TD, but not MC, methods can be fully incremental
 - You can learn **before** knowing the final outcome
 - Less memory
 - Less peak computation
 - You can learn **without** the final outcome
 - From incomplete sequences
- Both MC and TD converge (under certain assumptions), but which is faster?

SARSA Algorithm

- Commonly used is SARSA (λ). λ - eligibility trace factors to ensure rapid converge
- Reflects updating Q-values based on $(a_t, s_t, r_{t+1}, a_{t+1}, s_{t+1})$.
- Is a greedy method updating the policy be greedy w.r.t current estimate

Algorithm 1 SARSA

```
1:  $Q(s, a) \leftarrow$  arbitrarily
2: repeat {for each episode:}
3:   Initialise  $s$ 
4:   Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.  $\epsilon$ -greedy)
5:   for all steps in the episode do
6:     Take action  $a$ , observe  $r, s'$ 
7:     Choose  $a'$  from  $s'$  using policy derived from  $Q$  (e.g.  $\epsilon$ -greedy)
8:      $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
9:      $s \leftarrow s'; a \leftarrow a'$ ;
10:  end for
11: until  $s$  is terminal
```

Basic facts – So far

- What are do these exploration and exploitation strategies mean?
 - Offline – Data is fixed, agent learns from previous interaction
 - Online – agents interacts with environment including previously explored states
 - On-policy – Learns about policy currently executing
- Exploration – Find more information about the environment
- Exploitation – maximize/exploit know information to maximize reward
- Prediction: Evaluates the future for a given policy
- Control: Optimize the future, finding the best policy
- Learning: Using the history, predict (determine future) state while exploring the best policy [What, how, why, where]

Curse of Dimensionality

- Is exponential growth of policies with state and action spaces
 - Four binary features and 3 action states = 3^{2^4} → 43,046,721 policies
- To reduce state space for learning proposed [many approaches]
 - Feature reduction; Reduction of possible state-action combinations; Summarizing similar states

Dialogue - application

- Dialogue is represented as vector of real valued features $f(s)$ – learns function approximation; $f(s)$ is mapped to vector of estimate $Q(s, a)$
- Given the weight weights trained on data, Q-function is re-written as the inner product of state vector $f(s)$ and weighted vector w_a :

$$Q^\pi(s, a) = f(s)^T w_a = \sum_t f(s) w_{at} \quad (3.15)$$

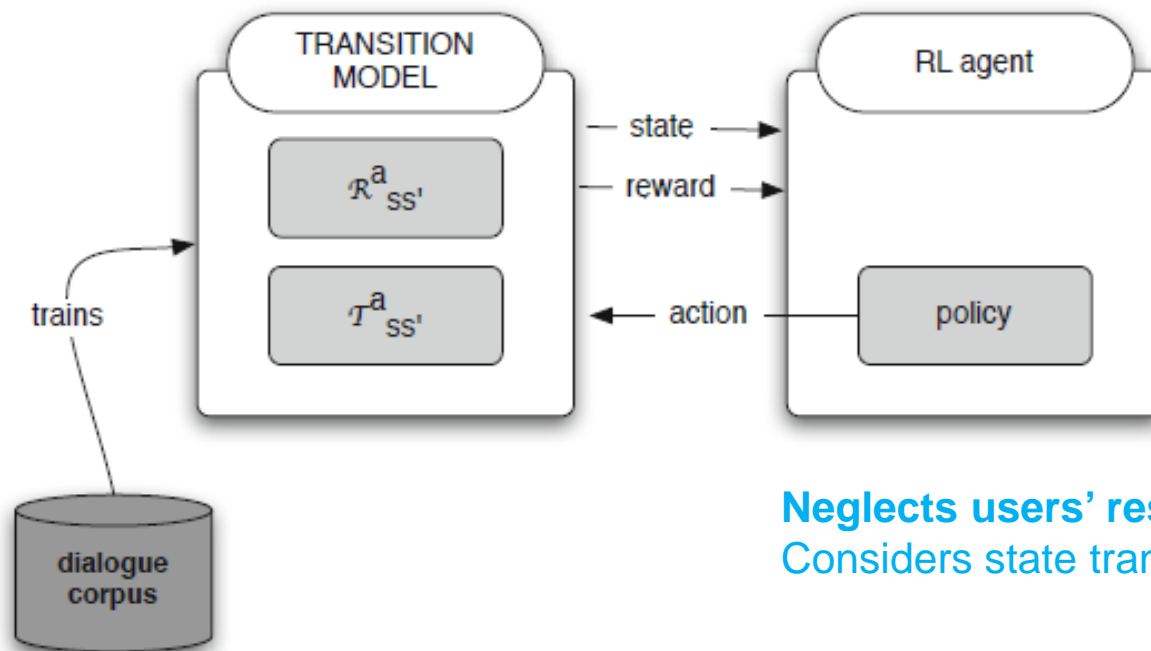
- Two are treated as similar if they share features – affected in training
- Similarity measure is called **Linear Kernel**

- Simulation-based RL offers **cheap** learning by training the policy

So far

- Agent-environment interaction
 - States
 - Actions
 - Rewards
- Policy: stochastic rule for selecting actions
- Return: the function of future rewards the agent tries to maximize
- Episodic and continuing tasks
- Markov Property
- Markov Decision Process
 - Transition probabilities
 - Expected rewards
- Value functions
 - State-value function for a policy
 - Action-value function for a policy
 - Optimal state-value function
 - Optimal action-value function
- Optimal value functions
- Optimal policies
- Bellman Equations
- The need for approximation

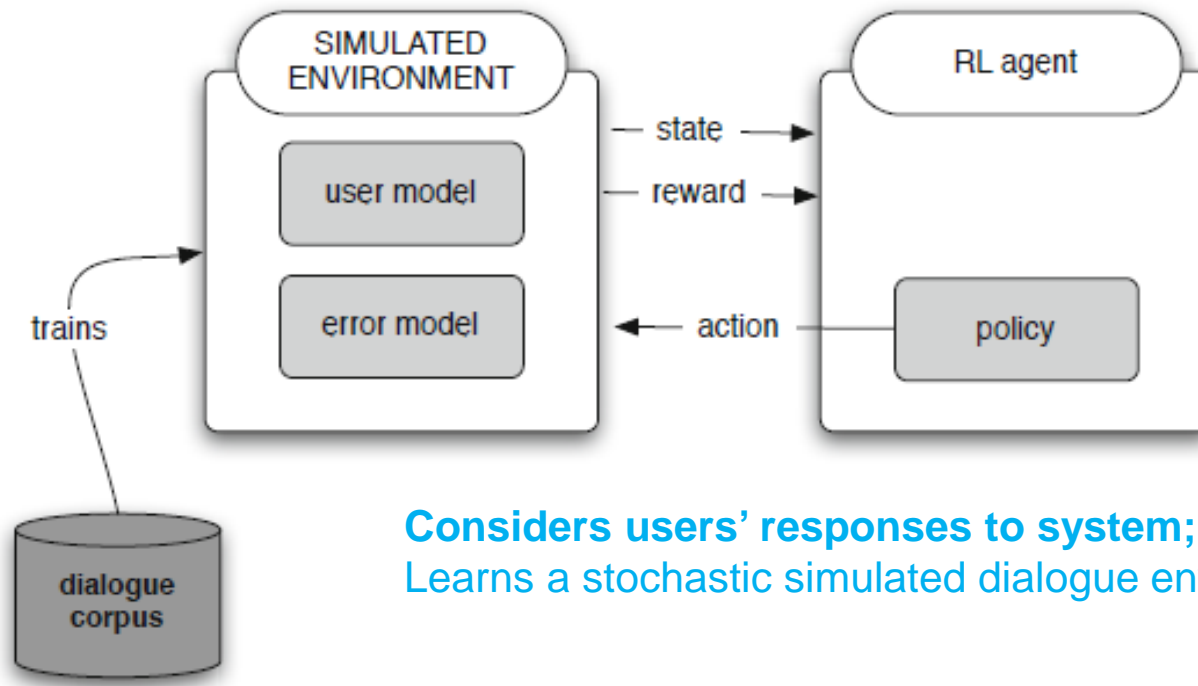
Model-based RL



Neglects users' responses to system
Considers state transitions and rewards only

- Approaches explicitly models dynamics of environment; learning via off-line [limited agent-environment interaction]
- Limitation: Corpora **not large enough** all transition probabilities; learning **limited to explored a-s combinations** [inflexibility]; learning from fixed data – working systems already exist – **What system do we need?**

Simulation-based RL



Considers users' responses to system; model the future
Learns a stochastic simulated dialogue environment from data

- Involves two phases: Simulated environment and RL agent (DS)
 - SE including components – trained via SL; includes user and error model [Dual architecture, CLS -> differential training];
 - Dialogue strategy training – trained by simulated MC or TD for systematic exploration, near optimal solution exploration; **generalize unseen dialogue states**

Simulation-Based RL

- Model-free approach that directly approximates value function via **online interaction**;
- Advantages
 - Large training episodes generated – **exhaustive strategies exploration**
 - Exploration of strategies not in the training data [**Unseen strategies exploration**]
 - No prior fixing of state space and actions – **dynamic scaling & modeling of dialogue**
- Challenges
 - **Quality** of learned strategy depends on quality of simulated environment
 - **Reward signal not readable** from data; yet reward function must explicitly constructed
 - Simulation results **inability to replicate** real user-dialogue performance
 - Simulated components need **in-domain data training** – expensive to collect & train
- Hence generalizability and automatic dialogue learning feasible

Dialogue Simulation – WOZ

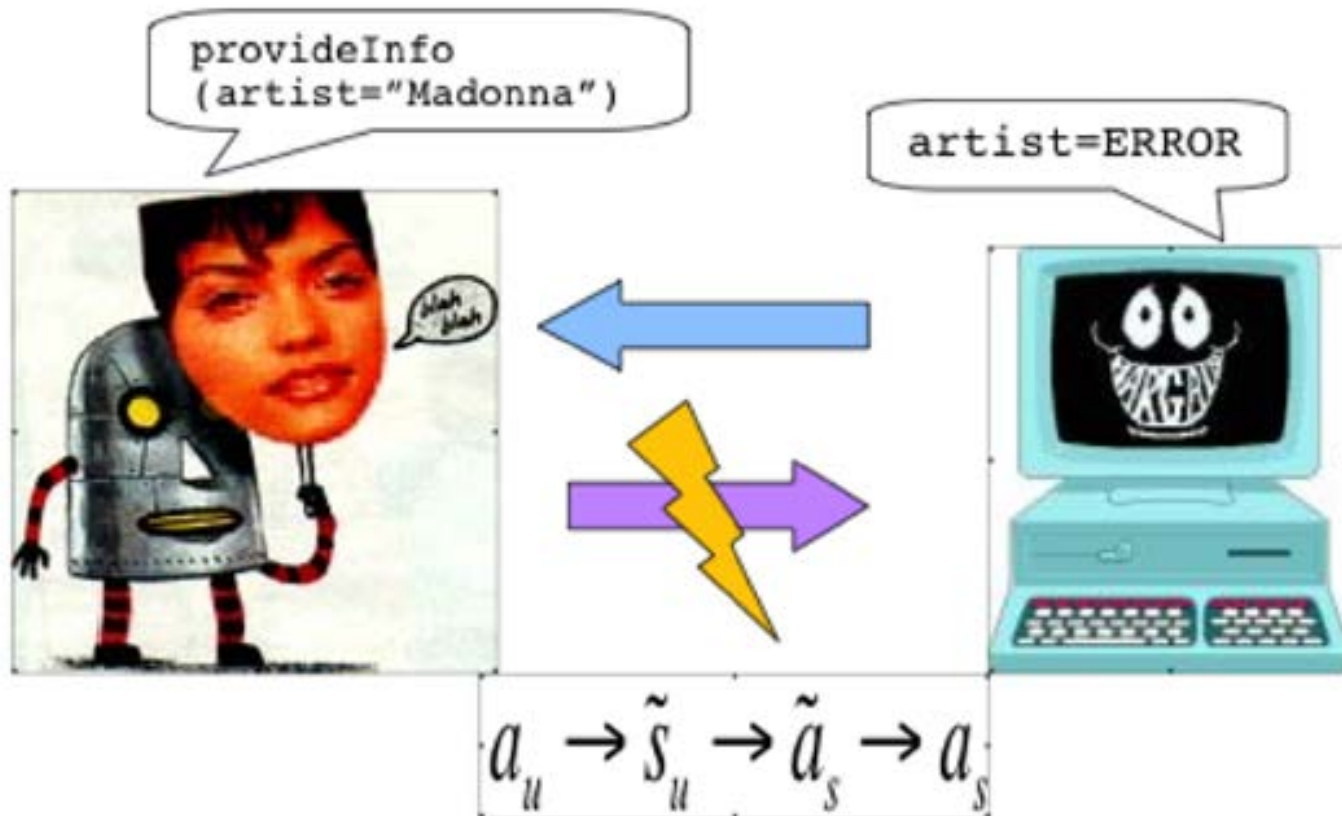
- Goal of Wizard-of- Oz: Characterize human behavior and user preferences; develop a language model and/or acoustic model for prototype system
- Procedure: Simulate, evaluate, generalize – development system
 - Hidden operator (wizard) simulates some aspects of human behavior in dialogue [ensure real HCI illusion considering human dynamics in communication]
 - Subjects evaluate by filling out questionnaires
 - Generalize human behavior, develop models
- Dialogue simulation need to be able to approximate real HCI in order to facilitate dialogue system development and testing

Wizard of Oz simulation



Wizard simulates dialogue behavior; user interacts in they are talking to the belief that they are talking to a machine, rates a machine and rates the dialogue behavior

DS: Computer Based Simulation



Dialogue manager interacts with a simulated user over a noisy channel

DS: Computer-based Simulation

- Goals: Testing and debugging prototype systems; automatic strategy development [RL, SL]
- Error model: Simulates error prone ASR;
- Variation from real HCI due to:
 - Quality of simulated components
 - Simulated users cannot rate the system according to their preferences
- Discussion: Learnt Framework
 - Simulated Environment for RL from data collected in WOZ experiment
 - Enables automatic strategy learning

Application Domains

- Information-Seeking Dialogue Systems
- Multimodal Output Planning and Information Presentation
- Multimodal Dialogue Systems for In-car digital Music Player

Conclusion

- Learning – Interaction between agent and world
 - Percepts received by an agent acts and improves agent's ability to behave optimally in the future to achieve the goal
- Reinforcement Learning – Achieve goal successfully
 - Learn how to behave successfully to achieve a goal while interacting with external environment, Learn via experience
 - Game playing – know when its win or loss
 - Well suited for dialogue strategy development – as dialogue is learned by evaluative feedback with delayed rewards and exploration