Action Refinement in Reinforcement Learning by Probability Smoothing

By Thomas G. Dietterich
& Didac Busquets
Speaker: Kai Xu

Presentation Overview

- *Background*
- The Probability Smoothing Method
- Experimental Study of Action Refinement
- Conclusion

Background -- Model Based Reinforcement Learning (MBRL)

- Experience gained during exploring is employed to learn the models of the state-action transition function and the reward function
- From the learned model, the optimal policy can be computed by many good algorithms
- MBRL is appropriate when the state and action space and relatively small and finite, and each exploring action is expensive.

Background -- Methods To Reduce the Need For Training Data

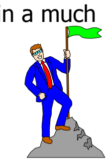
- By incorporate some kind of prior knowledge
- Previous Study: Abstraction knowledge across the states
  - So that the RL can generalize across states
- Abstraction knowledge across the actions (in this paper)
  - The RL assumes similar actions will have similar transition effects and rewards

Background -- Action Refinement

- Recall how human learns
  - Bad Kongfu Masters teach the students all the tricks at the beginning.
  - The students have to spend a long time to grasp all of them

Action Refinement

- Good Kongfu Masters teach the students only the basic actions at the beginning.
- After the students grasp the basic skills, he teach them the subtleties among different similar actions.
- The students grasp all the tricks in a much shorter time.



Action Refinement

- An RL algorithm initially treats a set of similar actions as a single abstraction
- Later, refines that abstraction action into individual actions.

The Probability Smoothing Method

- Background
- *The Probability Smoothing Method*
- Experimental Study of Action Refinement
- Conclusion

The Probability Smoothing Model

- Context:
  - The agent is interacting with an unknown but observable Markovian environment.
  - The environment contains a finite state set S, and a finite action set A.
  - The programmer groups set A into L disjoint action sets $A_1, A_2, \ldots, A_L$. Actions in the same subsets are 'similar'.

The Probability Smoothing Model

- Let $N_t(s,a)$ denote the # of times action a has been executed in state s.
- Let $N_t(s,a,s')$ denote the # of times this results in a transition to state s'.
- Let $W_t(s,a,s')$ denote the total rewards received when a caused a transition from s to s'.
- Define the probability smoothing model M_t such that

$$P_t(s'|s,a) = \frac{\sum_{a'\in A_l} \lambda_{a'} N_t(s,a',s')}{\sum_{a'\in A_l} \lambda_{a'} N_t(s,a')}$$

$$R_t(s,a,s') = \frac{\sum_{a'\in A_l} \lambda_{a'} W_t(s,a',s')}{\sum_{a'\in A_l} \lambda_{a'} N_t(s,a')}$$

Determine Smoothing Parameter λ

- Suppose the true transition probability from s to s' after executing action a is $P(s'|s,a)$, and the estimate to this probability is $P_t(s'|s,a)$
- We want to find a proper λ such that $P_t(s'|s,a)$ would be a consistent estimator for the true probability.
- To determine which estimator is more appropriate, we need to define the error measure as the following

$$J(s,a) = \sum_{s'} [P(s'|s,a) - P_t(s'|s,a)]^2$$

- So the problem is to find a λ which minimizes J(s,a)

Derivation of Optimal Smoothing Parameters in the simplest case

- Let's suppose there are only two similar actions, a_1 and a_2
- The current state is s
- There are only two possible resulting states, s' and s''
- Action a_1 has been applied on state s for N1 times.
- For H1 times it transit to state s'
- Action a_1 has been applied on state s for N2 times.
- For H2 times it transit to state s'

Derivation of Optimal Smoothing Parameters in the simplest case

- Suppose the true transition probability from s to s' after exe a_1 is p_1.
- Although H1/N1 is an estimator for p_1, it requires large number of trials.
- So we should use the smoothing model:

$$\hat{p}_1 = \frac{H_1 + \lambda H_2}{N_1 + \lambda N_2}$$

Derivation of Optimal Smoothing Parameters in the simplest case

- After calculation, we find the most appropriate smoothing parameter

$$\lambda = \frac{V_1}{N_2 \varepsilon^2 + V_2} \quad \text{where}$$

$$V_1 = p_1(1-p_1)\ ,\ V_2 = p_2(1-p_2)$$

$$\varepsilon = |p_1 - p_2|$$

- Properties of using this λ:

$$\lim_{N_1 \to \infty} \hat{p}_1 = p_1 \quad \text{and} \quad \lim_{N_1 \to \infty} \lim_{N_2 \to \infty} \hat{p}_1 = p_1$$

Derivation of Optimal Smoothing Parameters in the simplest case

- Therefore, the probability smoothing will converge to the optimal policy.
- This model can be expand to cases such as
  - there are more than 2 similar actions
  - there are more than 2 possible resulting states
- We can use the resulting λ to build good estimator for the reward.

Determine the Level of Smoothing in Practice

- Big problem: In most practical cases, we will never know the true value for p_1, p_2, or ε
- A naive approach for choosing λ would be estimate p_1 by H1/N1, estimate p_2 by H2/N2
- But when the trial number is small, the variance to these estimates are very high. The result is poor.
- So the paper proposed to use "default smoothing", in which we assume the default values of p_1, p_2, and ε, and plug in the value of N2 from the real data.

Determine the Level of Smoothing in Practice

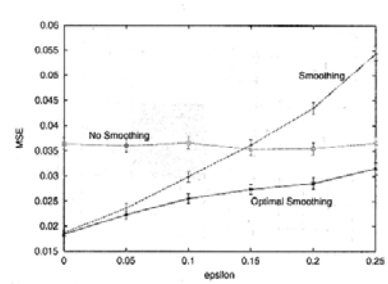
- The author proposed to use default values

$$p_1 = 0.1,\quad p_2 = 0.15,\quad \varepsilon = |p_1 - p_2| = 0.05$$

  for the simplest case.
- They work well when $\varepsilon < 0.15$ for all values of p_1
- For cases that there are more than 2 possible resulting states, the author proposed to use default values

$$V_1 = 0.09\ ,\quad V_2 = 0.1275\ ,\quad \varepsilon^2 = 0.0025$$

Determine the Level of Smoothing in Practice

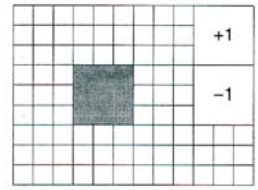


Experimental Study of Action Refinement

- Background
- The Probability Smoothing Method
- *Experimental Study of Action Refinement*
- Conclusion

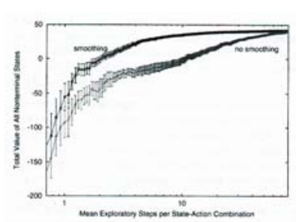
Experimental Study of Action Refinement -- Context

- A toy maze with 81 non-terminal states and 2 terminal states.
- 16 actions from the cross-product of the 4 compass directions with 4 modifiers.
- Actions are grouped into 4 sets.
- To measure the performance of a policy, we compute the value function V^{π} and sum the value of all 81 non-terminal states
- The optimal policy has total value of 43.37



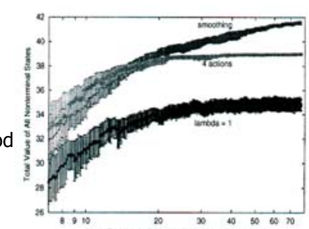
Experimental Study of Action Refinement -- Compare With No Smoothing Method

- Comparison of
  - probability smoothing, and
  - no smoothing ($\lambda = 0$)
- The probability smoothing model is much better



Experimental Study of Action Refinement -- Compare with fixed smoothing & four-action

- Comparison of
  - fixed smoothing ($\lambda = 1$)
  - four-action method
  - probability smoothing
- After 9.3 exploration steps, four-action method and probability smoothing method beat fixed smoothing.
- After 23 steps, probability smoothing method wins.

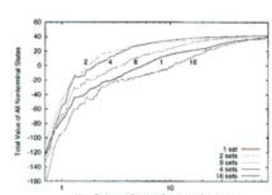


Experimental Study of Action Refinement -- Conclusion

- Conclusion for the previous experiment
  - The probability smoothing method is vastly superior to no-smoothing method.
  - If large training set is available, probability smoothing method is better than fix-smoothing and four-action method.

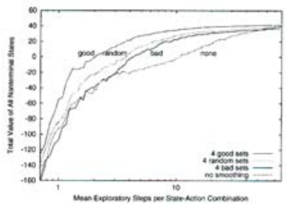
Experimental Study of Action Refinement -- Sensitivity To The Size of Action Sets

- Vary the # of action sets from 1, 2, 4, 8, and 16
- With similar actions be grouped together to the extent possible.
- 16 separate action sets gives high variance.
- One single action set gives high bias.
- 4 sets and 2 sets gave the best performance during the early part of the curve.



Experimental Study of Action Refinement -- Sensitivity To The Action Set Correctness

- At intermediate sample size (4), even random groupings give better performance than no smoothing.
- At large sample sizes, the bias in the random and bad groupings leads to worse performance than either no smoothing or well-chosen action sets.



Conclusions

- Probability Smoothing Method is introduced to action refinement to
  - speed up RL applications
  - by partition actions into sets of similar actions.
- It significantly eases the designing of a set of good actions in RL.
- Probability smoothing parameter is determined by “default smoothing” and the corresponding # of trials.
- Good prior action set partition is critical to the performance.

Action Refinement in Reinforcement Learning by Probability Smoothing

Thomas G. Dietterich, Didac Busquets
Ramon Lopez de Mantaras,
Carles Sierra

Comments by : Sameer Apte

Action Refinement applied to the Robot Navigation Problem

- Robot navigates by finding visual landmarks
- Robot's camera has a viewing angle of 60 degrees
- The space around the robot was partitioned into six 60-degree sectors

Action Refinement applied to the Robot Navigation Problem

- An action called "Move While looking for Landmarks (MLL)" was defined
- Robot moves forward while aiming its camera in one of the six sectors to search for new visual landmarks
- Can define six MLL actions, one for each sector and let robot decide which sector to examine with the camera

Action Refinement applied to the Robot Navigation Problem

- Designers do not know which of these actions would be most useful
- Include all these actions in the MDP and let RL system determine which actions are useful
  - Problem : Large amount of exploration required to learn a good policy
- Train the robot several times ,each time with a different set of actions
  - Problem : Even more training experiences required

Action Refinement applied to the Robot Navigation Problem

- Solution : Action Refinement
- We know that different variants of the MLL action have similar behavior
- Initially treat these similar actions as a single abstract action
- Later allow the learning algorithm to refine abstract action into individual actions

Direct vs. Model-Based Reinforcement Learning

-- Commentary on Kai Xu's presentation

-- Commented by Ruinan Lu

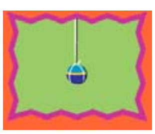
-- Reference: paper by C. Atkeson, et al.

Criteria

- Data efficiency
- Computing efficiency

Problem for comparison of the two approaches: single pendulum swing-up

- Make it swing!



Model-based RL

- Known reward function:

$$r(\theta,\tau) = ((\theta-\theta_d)^2 + \tau^2)\Delta$$

  - θ : angle of the pendulum
  - θ_d : desired angle for the inverted vertical state
  - τ : motor torque
  - Δ : time step

- Equation of motion: $\theta'' = \tau - 9.81\cos(\theta)$

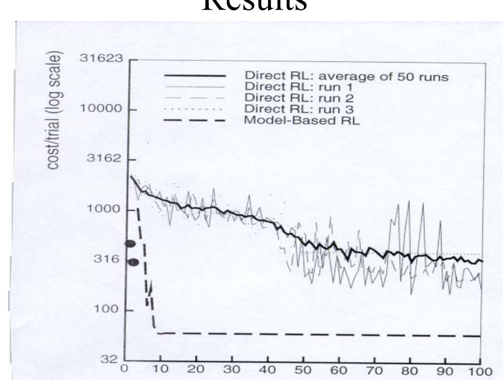
Q-learning

$$Q(x_k,u_k) = Q(x_k,u_k) + \alpha[r(x_k,u_k) + \gamma Q(x_{k+1},u_{k+1}) - Q(x_k,u_k)] * e(x_k,u_k)$$

- α : learning rate
- γ : discount factor
- x : state vector
- u : control vector

- Optimal action: $\arg\min_u Q(x_1,u)$

Results



Conclusions

- Simple Dynamics favor MRL
  - Exploratory action is expensive
  - Exploration is performed on a physical system
- Cases favor Direct RL
  - More training experiences
  - Learner interacts with an inexpensive simulator