Deep Q-learning for Active Recognition of GERMS: Baseline performance on a standardized dataset for active learning

Mohsen Malmir, Karan Sikka, Deborah Forster, Javier Movellan, and Garrison W. Cottrell

Presented by Ruohan Zhang
The University of Texas at Austin

April 13, 2016

Outline

The Active Object Recognition (AOR) Problem

- The **recognition** module: what is this?
- The **control** module: where to look?
- Goal: find a sequence of sensor control commands that maximizes recognition accuracy and speed.

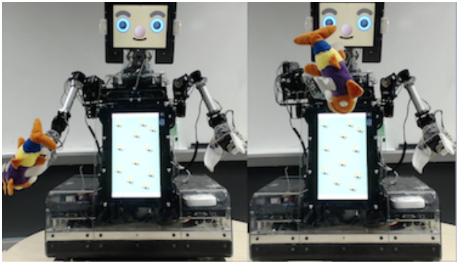


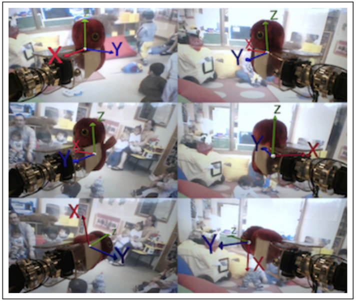
Figure : The AOR problem for the RUBI robot [Malmir et al.,].

Motivation

- A benchmark dataset for the AOR research
  - more difficult than previous ones, e.g. [Nayar et al., 1996].
  - without the need to have access to a physical robot.
- A baseline method and its performance
  - combines deep learning and reinforcement learning: deep Q-learning.

Data Collection

- The RUBI project at UCSD Machine Perception Lab.
- Six configurations for each object, two arms and three axes.
- RUBI brings the object to its center of view, rotate object by 180°.



Data Statistics

- Data format: [image][capture time][joint angles].
- Joint angles: 2-DOF head , 7-DOF arms X 2.
- 136 objects, 1365 videos, 30fps, 8.9s on average.
- Bound boxes are annotated manually.

Examples

Figure : Left: the collage of all 136 objects. Right: some ambiguous objects that require rotation to disambiguate.

Example Videos

The videos for the left arm and for the right arm.

The goal: what to do in a state?

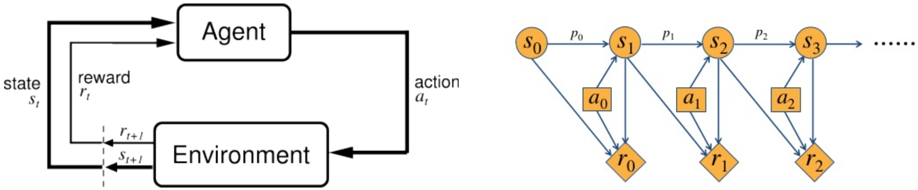


Figure : The agent-environment interaction and Markov decision process (MDP).

Markov Decision Process (MDP)

Definition

A tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is a finite set of states.
- $\mathcal{A}$ is a finite set of actions.
- $\mathcal{P}$ is a state transition probability matrix. $\mathcal{P}^a_{ss'} = \mathbb{P}[s'|s, a]$.
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[r|s, a]$.
- γ is a discount factor, $\gamma \in [0, 1)$.

Policy and Value Function

Policy

Agent behavior is fully specified by $\pi(s, a) = \mathbb{P}[a|s]$, one can directly optimize this by trying to maximize expected reward.

Policy and Value Function

Policy

Agent behavior is fully specified by $\pi(s, a) = \mathbb{P}[a|s]$, one can directly optimize this by trying to maximize expected reward.

Action-value function

$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[v_t|s_t = s, a_t = a]$, expected return starting from state s, taking action a, and then following policy π.

Policy and Value Function

Policy

Agent behavior is fully specified by $\pi(s, a) = \mathbb{P}[a|s]$, one can directly optimize this by trying to maximize expected reward.

Action-value function

$Q^\pi(s, a) = \mathbb{E}_\pi[v_t|s_t = s, a_t = a]$, expected return starting from state s, taking action a, and then following policy π.

Goal of reinforcement learning

Find optimal policy:

$$\pi^*(s, a) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Therefore, if we know $Q(s, a)$, we find the optimal policy.

Bellman Equations

Action-value function recursive decomposition

$Q^\pi(s,a) = \mathbb{E}_\pi[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]$

Dynamic programming to solve MDP

Assumption: environment model $\mathcal{P}, \mathcal{R}$ is fully known.

Model-free Reinforcement Learning: Q-learning

The Q-learning algorithm [Sutton and Barto, 1998]

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step):
    Choose a from s
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$
  until s is terminal

Remark

$r + \gamma \max_{a'} Q(s', a')$ can be seen as a supervised learning target, but it is changing.

Deep Reinforcement Learning?

The basic Q-learning

Assumptions: discrete states and actions (lookup Q-table); manually defined state space.

The deep Q-learning

Using a deep neural network to approximate the Q function.

The Network Architecture

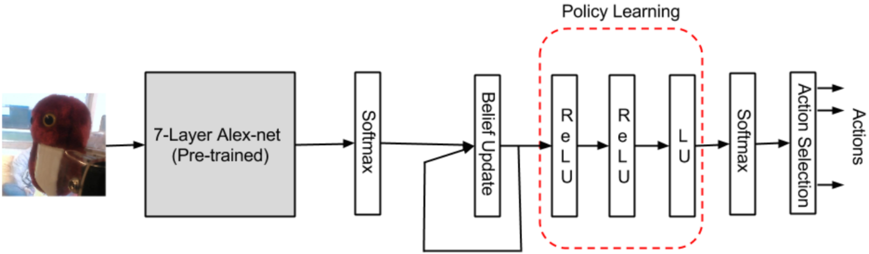


Figure : The deep network architecture in [Malmir et al.,].

The MDP in this Paper

MDP

- The state B_t : the output of softmax layer of the CNN at time t, i.e., the belief vector over object labels.
  - not the input image at time step t, as in [Mnih et al., 2013].
  - use Naive Bayes to accumulate belief from history.

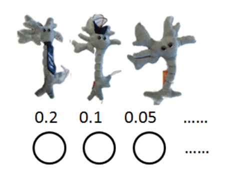


Figure : The state space representation in [Malmir et al.,].

The MDP in this Paper

MDP

- a_t : ten rotation commands $\{\pm\pi/64, \pm\pi/32, \pm\pi/16, \pm\pi/8, \pm\pi/4\}$.
- $\mathcal{P}$: transition matrix **unknown** (The reason they used Q-learning).
- $\mathcal{R}$: $+10$ for correct classification, -10 ow.
- γ : unknown.

The Training Algorithm

- Exactly the Q-learning algorithm.

$$Q(B_t, a_t) \leftarrow Q(B_t, a_t) + \alpha[r_t + \gamma \max_a Q(B_{t+1}, a) - Q(B_t, a_t)]$$

- For network weights update, use stochastic gradient descent:

$$W \leftarrow W - \lambda[r_t + \gamma \max_a Q(B_{t+1}, a) - Q(B_t, a_t)] \frac{\partial}{\partial W} Q(B_t, a_t)$$

- mini-batch update. This is a key trick to stabilize deep RL network. Otherwise, the learning target is changing rapidly and it will not converge.

Results

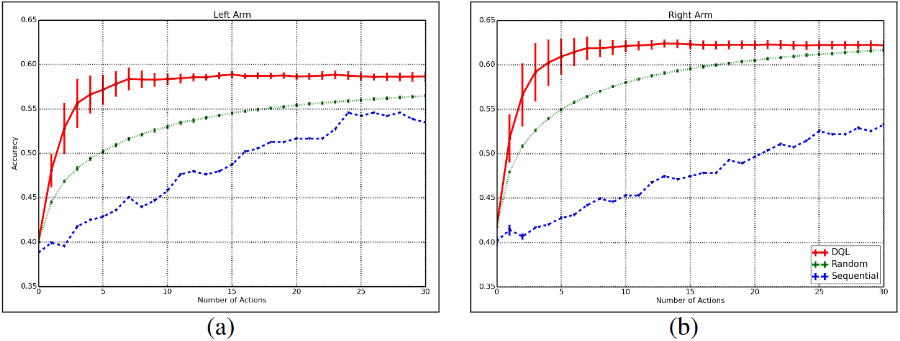


Figure : The experiment results on classification accuracy [Malmir et al.,].

Results

Method \ Prediction Accuracy(%)	48	53	55	58	62	
Sequential	18	30	-	-	-	**Right Arm**
Random	2	4	6	10	-	
DQL	**1**	**2**	**2**	**3**	**10**	
Sequential	15	24	-	-	-	**Left Arm**
Random	3	10	18	-	-	
DQL	**1**	**3**	**3**	**7**	-	

Figure : The number of steps required to achieve certain classification accuracy by different algorithms [Malmir et al.,].

Conclusions

- The GERMS dataset.
- The deep Q-learning for AOR, however, much space left for improvement:
    - performance-wise.
    - very basic version of deep Q-learning.

Discussions

- Right arm outperforms left arm.
- "Uncommon" objects for robotic tasks.
- Manual bounding box annotations is labor intensive.
- State representation (belief vector).
- The most representative frame?
- Any other similar datasets?
- Extension: using RNN to combine the two modules (control and recognition), e.g., **Recurrent models of visual attention** [Mnih et al., 2014].

Malmir, M., Sikka, K., Forster, D., Movellan, J., and Cottrell, G. W.
Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning.
In *Proceedings of the British Machine Vision Conference (BMVC), pages*, pages 161–1.

Mnih, V., Heess, N., Graves, A., et al. (2014).
Recurrent models of visual attention.
In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013).
Playing atari with deep reinforcement learning.
arXiv preprint arXiv:1312.5602.

Nayar, S., Nene, S., and Murase, H. (1996).
Columbia object image library (coil 100).
Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96.

Sutton, R. S. and Barto, A. G. (1998).
Reinforcement learning: An introduction.
MIT press.