Introduction to Reinforcement Learning

Prof. Javier Ruiz del Solar

amtc
ADVANCED MINING TECHNOLOGY CENTER

Faculty of Mathematics and Physical Sciences
University of Chile
www.amtc.cl

References/disclaimer

Somes slides "taken" from David Silver's course (highly recommended)

https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver

Recommended references

- An Introduction to Reinforcement Learning, Sutton and Barto, 1998
  - MIT Press, 1998
  - ~ 40 pounds
  - Available free online!
  - http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html
- Algorithms for Reinforcement Learning, Szepesvari
  - Morgan and Claypool, 2010
  - ~ 20 pounds
  - Available free online!
  - http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

Agenda

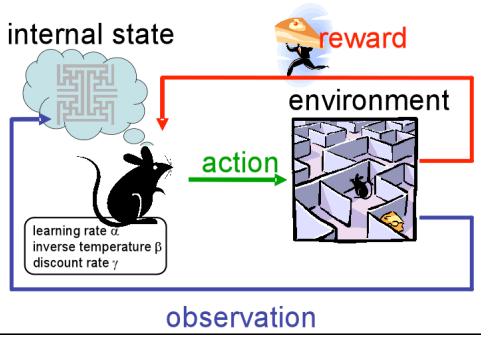
- Motivation
- Reinforcement Learning
  - Markov Decision Process
  - Classical Tabular Methods
  - Modeling Issues
- Function Approximation Methods
  - Deep Reinforcement Learning
  - RL & Imitation Learning & Human Feedback
  - Challenges

www.amtc.cl

RL in a sentence

"RL is learning by trial and error"

"Based on the framework of Markov Decision Processes (MDP), RL addresses the problem that how an autonomous active agent learns the optimal policies while interacting with an initially unknown environment."



Motivation

How can we learn to drive a bycicle?

Image taken from http://hackedgadgets.com/wp-content/MURATA_BOY_bicycle_riding_robot_2.jpg

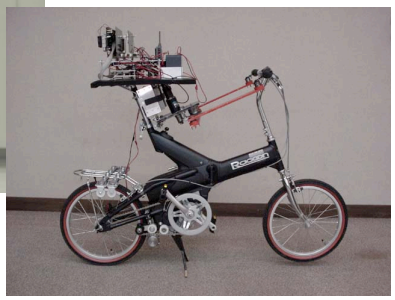
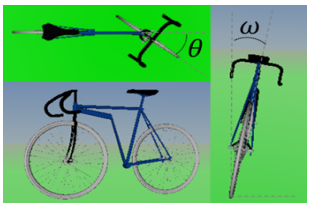
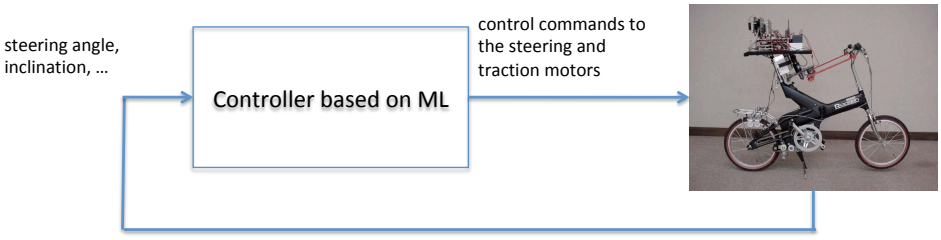


Image taken from https://oldster.files.wordpress.com/2006/03/robotbike1.jpg



www.amtc.cl

Motivation



- Controller training by supervised learning requires knowledge of the commands to be given to the motors for each situation (for each possible observation).
- This is not possible unless imitation learning is used, which requires that there is a solution to imitate.
- Alternatively, this can be addressed by Reinforcement Learning, which requires defining a *reward*, which quantifies how good the control is being performed

www.amtc.cl

Agenda

- Motivation
- Reinforcement Learning
  - Markov Decision Process
  - Classical Tabular Methods
  - Modeling Issues
- Function Approximation Methods
- Advanced Topics
  - Deep Reinforcement Learning
  - RL & Imitation Learning & Human Feedback
  - Challenges

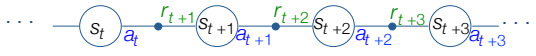
www.amtc.cl

Markov Decision Processes (MDPs)

An MDP is defined by:

S – set of states of the environment
$A(s)$ – set of actions possible in state s
$P(s,s',a)$ – probability of transition from s to s' given a
$R(s,s',a)$ – expected reward on transition s to s' given a
γ – discount rate for delayed reward discrete time, $t = 0, 1, 2, \ldots$

π - the policy, the set of actions that the agent takes. It maps each state to an action.



Sutton, R., Reinforcement Learning: A Tutorial

www.amtc.cl

Reward

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

Reinforcement learning is based on the reward hypothesis

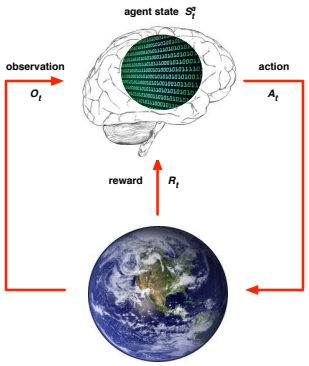
Definition (Reward Hypothesis)

All goals can be described by the maximisation of expected cumulative reward

Do you agree with this statement?

(taken from Silver course)

State



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

the environmenr can be fully observable or not

(taken from Silver course)

Policy

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent), $A_t \sim \pi(\cdot|S_t), \forall t > 0$

(taken from Silver course)

Value Function

A key concept is the value function, which captures the long term expected return (i.e. expected sum of the discounted rewards) of a policy for every possible state:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s, a_0 = a\right]$$ state-action value function

$$\begin{aligned} V^{\pi}(s) &\triangleq Q^{\pi}(s,\pi(s)) \\ &= E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s\right] \end{aligned}$$ (state) value function

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), \forall s \in \mathcal{S}$$ optimal policy

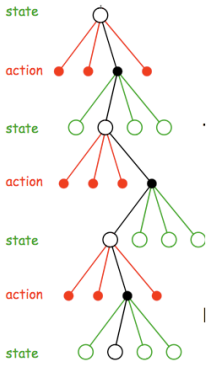
The goal of solving a MDP is to find the optimal policy.
The use of a state-action value functions eliminates the need for storing policies explicitly or knowing the model (state transition probabilities).

Bellman Equation

The optimal value function satisfies the Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \right] \quad s \in \mathcal{S}$$

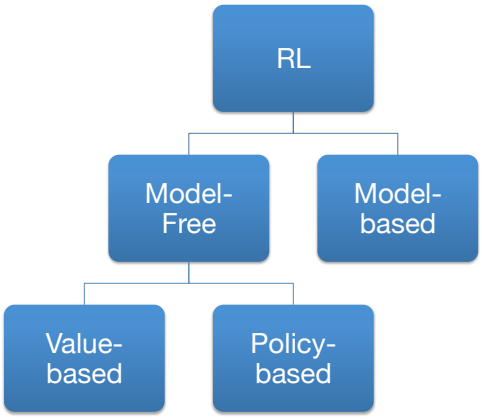
(Bellman, 1957)



(recursive derivation of Bellman Equ.)

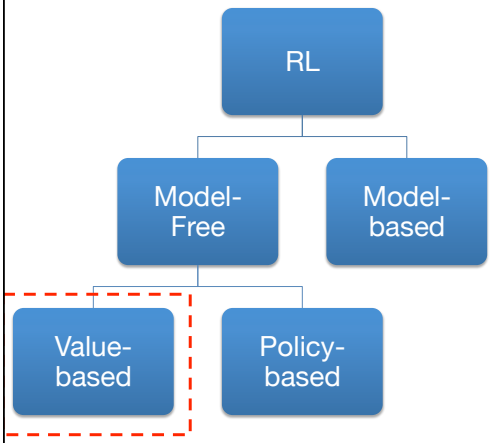
$$\begin{aligned} V^\pi(s) &= E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s \right] \\ &= E_\pi \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0 = s \right] \\ &= E_\pi \left[\mathcal{R}^{\pi(s)}_{ss'} + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_0 = s \right] \\ &= \sum_{s' \in \mathcal{S}} \mathcal{P}^{\pi(s)}_{ss'} \left[\mathcal{R}^{\pi(s)}_{ss'} + \gamma V^\pi(s') \right]. \end{aligned}$$

Simple Taxonomy of RL



- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

Simple Taxonomy of RL



- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

www.amtc.cl

Online RL

- **Monte-Carlo** value estimate: Use the empirical return, $U(s_t)$ as a target estimate for the actual value function:

$$V(s_t) \leftarrow V(s_t) + \alpha\left(U(s_t) - V(s_t)\right)$$

** Not a Bellman equation. More like a gradient equation.*

- TD-learning:

$$V(s_t) \leftarrow V(s_t) + \alpha\,(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))\;\forall t = 0, 1, 2, \ldots$$

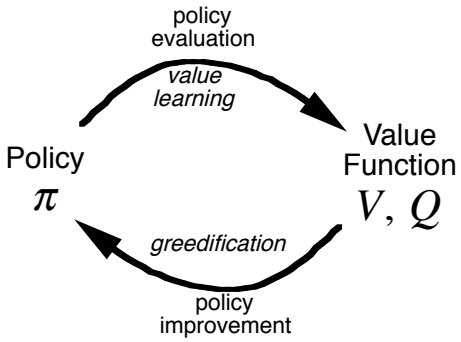
TD-error

learning rate

Temporal-Difference learning = TD learning

www.amtc.cl

Q-Learning



(exploration vs. exploitation dilemma)

$$\pi^{\epsilon}(s) \triangleq \begin{cases} \operatorname{argmax}_a Q^{\pi}(s,a), & \text{with probability } 1-\epsilon; \\ \texttt{UniformRandom}(\mathcal{A}), & \text{with probability } \epsilon. \end{cases}$$

The use of a state-action value function eliminates the need for storing policies explicitly.

Sutton & Barto, 1998

Q-Learning

➢Q(*s*,*a*) can be estimated incrementally:

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\right)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Q(s,a)	s_1	s_2	…	s_n
a_1				
a_2				
…				
a_m				

Watkins, C. J. C. H. (1989). "*Learning from delayed rewards". PhD thesis, King's College, University of* Cambridge.

Q-Learning
Off-Policy TD Control

One-step Q-learning :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s' (interaction with the environment!)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until s is terminal

off-policy: chosen action *a* is not necesarilly the one used previoulsy in Q(s,a) as *a'*

www.amtc.cl

Sarsa
On-Policy TD Control

S A R S A: State Action Reward State Action

Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s' (interaction with the environment!)
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

www.amtc.cl

Q-Learning vs. Sarsa

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Figure: SARSA

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure: Q-Learning

Update formula SARSA: "on-policy" TD

$$Q^{(k+1)}(s,a) = Q^{(k)}(s,a) + \alpha \cdot \left[r(s,a) + \gamma \cdot Q^{(k)}(s',a') - Q^{(k)}(s,a)\right]$$

Update formula Q-Learning: "off-policy" TD

$$Q^{(k+1)}(s,a) = Q^{(k)}(s,a) + \alpha \cdot \left[r(s,a) + \gamma \cdot \max_{a'} Q^{(k)}(s',a') - Q^{(k)}(s,a)\right]$$

www.amtc.cl

(taken from Jan-Hendrik & Bertold, 2008)

on-policy versus off-policy

Algorithms which concern about the policy which yielded past state-action decisions are referred to as on-policy algorithms, while those ignoring it are known as off-policy.

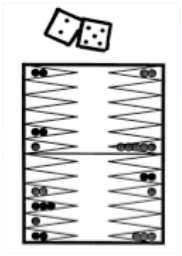
Off-Policy learning algorithms evaluate and improve a policy that is different from Policy that is used for action selection.

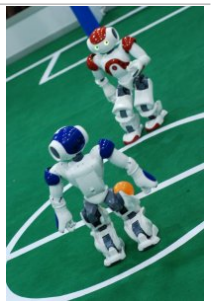
On-policy methods attempt to evaluate or improve the policy that is used to make decisions. In contrast, off-policy methods evaluate or improve a policy different from that used to generate the data.

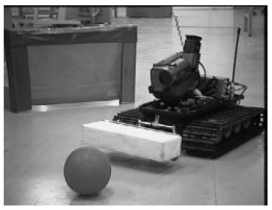
www.amtc.cl

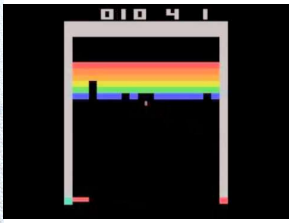
(taken from Jan-Hendrik & Bertold, 2008)

Examples of Problems Addressed using RL









www.amtc.cl

Modeling Issues

- The selection of the reward is a key issue. The reward defines what the agent learns (e.g. chess).

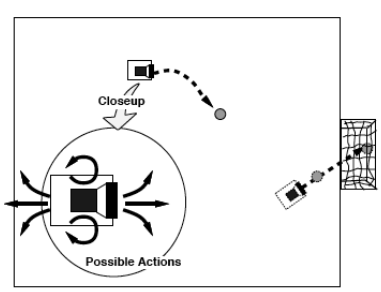
  - sparse versus dense rewards issue (how much a priori knowledge to introduce)

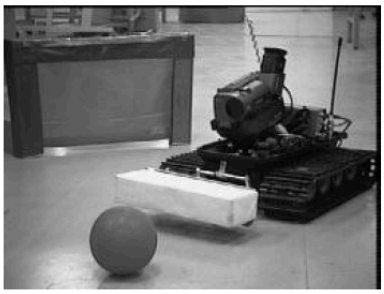
- The agent must be able to observe the reward and the states (observability). This is obvious, but often forgotten!

www.amtc.cl

Modeling – Example 1: Learning to kick goals

states? … actions? … reward?

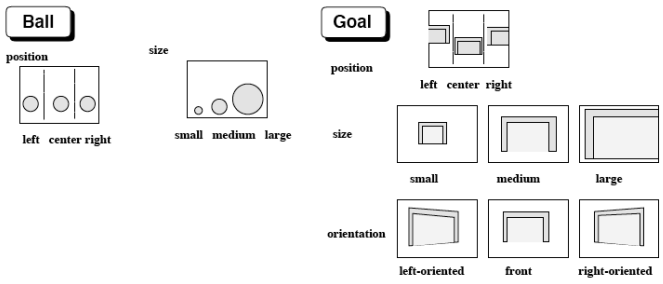




M. Asada , S. Noda , S. Tawaratsumida , K. Hosoda, Purposive behavior acquisition for a real robot by vision-based reinforcement learning, Machine Learning, v.23 n.2-3, p. 279-303, May/June 1996.

Modeling – Example 1: Learning to kick goals

States

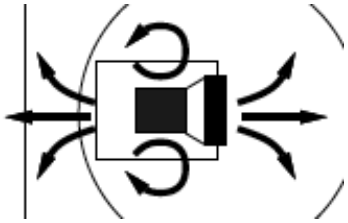


Number of states: 280 (27x9+ 9+27+1)
(it considers the cases in which just the ball, just the goal, both or none object is observed)

www.amtc.cl

Modeling – Example 1: Learning to kick goals

Actions



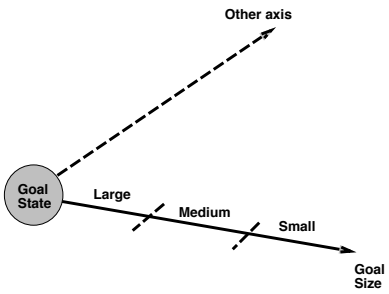
Number of actions: 9 (3x3)
(each motor forward, backwards or stop)

Reward

1 when the ball is kicked into the goal and 0 otherwise

Modeling – Example 1: Learning to kick goals

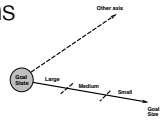
LEM: Learning from Easy Missions



www.amtc.cl

Modeling – Example 1: Learning to kick goals

LEM: Learning from Easy Missions



This idea has been generalized and know today as curriculum learning.

"Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones."

https://medium.com/@pprocks/curriculum-learning-654aa6423abd

www.amtc.cl

Modeling – Example 2: Learning to give passes

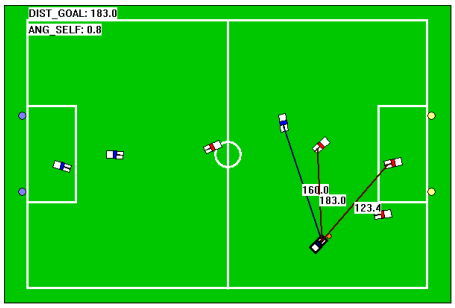
In a football game with *n* players, how many states do we have?

Pablo Recabal, Memoria Ingeniero Civil Electricista, Universidad de Chile.

www.amtc.cl

Modeling – Example 2: Learning to give passes

In a football game with n players, how many states do we have?
observability is a key issue



State
- positions of the goals
- position of the ball
- position of the "best" teammate
- position of the two "worst" players of the other team

Pablo Recabal, Memoria Ingeniero Civil Electricista, Universidad de Chile.

www.amtc.cl

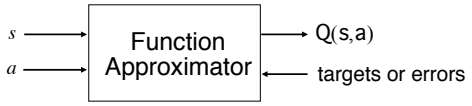
Function Approximation Methods

- Value functions of large-scale problems (large state and/or action spaces) are difficult to store in a table.
- An important factor in creating solutions for such large-scale problems is the use of linear function approximation.
- This approximation technique allows the long-term utility (value) of policies to be represented in a low-dimensional form, dramatically decreasing the number of parameters that need to be learned or stored.
- $Q(s,a)$ is approximated using a parametric function, with a number of parameters smaller than the number of states.

[Geramifard et al., 2013]

www.amtc.cl

Function Approximation Methods



Could be:

- table
- Backprop Neural Network
- Radial-Basis-Function Network
- Tile Coding (CMAC)
- Nearest Neighbor, Memory Based
- Decision Tree

gradient-descent methods

[Geramifard et al., 2013]

Remember: Gradient descent

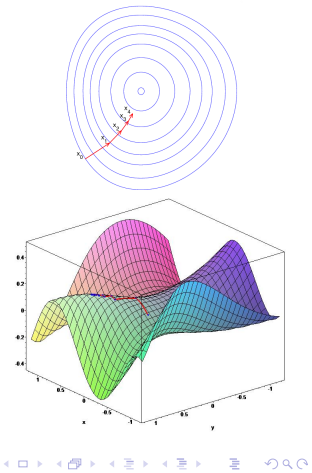
- Let $J(\mathbf{w})$ be a differentiable function of parameter vector $\mathbf{w}$
- Define the *gradient* of $J(\mathbf{w})$ to be

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{pmatrix} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_n} \end{pmatrix}$$

- To find a local minimum of $J(\mathbf{w})$
- Adjust $\mathbf{w}$ in direction of -ve gradient

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

where α is a step-size parameter



(taken from Silver course)

Action-Value Function Approximation

- Approximate the action-value function

$$\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A)$$

- Minimise mean-squared error between approximate action-value fn $\hat{q}(S, A, \mathbf{w})$ and true action-value fn $q_\pi(S, A)$

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))^2 \right]$$

- Use stochastic gradient descent to find a local minimum

$$-\frac{1}{2} \nabla_{\mathbf{w}} J(\mathbf{w}) = (q_\pi(S, A) - \hat{q}(S, A, \mathbf{w})) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w})$$
$$\Delta \mathbf{w} = \alpha (q_\pi(S, A) - \hat{q}(S, A, \mathbf{w})) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w})$$

(taken from Silver course)

www.amtc.cl

Q-Learning with Function Approximation

Algorithm 5: Q-Learning

Input: MDP$\setminus\{\mathcal{P}, \mathcal{R}\}, \alpha, \epsilon$
Output: π

$\boldsymbol{\theta} \leftarrow$ Initialize arbitrarily
$\langle s, a \rangle \leftarrow \langle s_0, \pi^\epsilon(s_0) \rangle$
while *time left* **do**
  Take action a and receive reward r and next state s'
  $Q^+(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$
  $\delta \leftarrow Q^+(s, a) - Q(s, a)$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta \phi(s, a)$
  $\langle s, a \rangle \leftarrow \langle s', \pi^\epsilon(s') \rangle$
return π greedy w.r.t. Q

$$Q^\pi(s, a) = \phi(s, a)^\top \boldsymbol{\theta}$$

(taken from Geramifard et al., 2013)

www.amtc.cl

SARSA with Function Approximation

Algorithm 6: SARSA

Input: MDP$\setminus\{\mathcal{P}, \mathcal{R}\}, \alpha, \epsilon$
Output: π

$\boldsymbol{\theta} \leftarrow$ Initialize arbitrarily
$\langle s, a \rangle \leftarrow \langle s_0, \pi^{\epsilon}(s_0) \rangle$
while *time left* **do**
  Take action a and receive reward r and next state s'
  $a' \leftarrow \pi^{\epsilon}(s')$
  $Q^{+}(s, a) \leftarrow r + \gamma Q(s', a')$
  $\delta \leftarrow Q^{+}(s, a) - Q(s, a)$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta \phi(s, a)$
  $\langle s, a \rangle \leftarrow \langle s', a' \rangle$
return π greedy w.r.t. Q

$$Q^{\pi}(s, a) \quad = \quad \phi(s, a)^{\top} \boldsymbol{\theta}$$

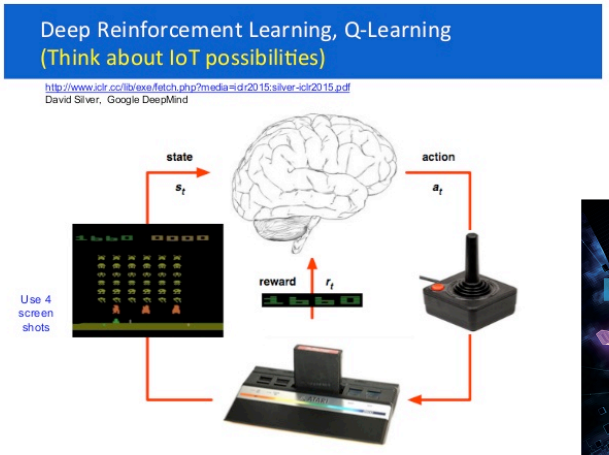
(taken from Geramifard et al., 2013)

Deep Reinforcement Learning

The basic idea is to approximate the value function using a deep network.

The use of a deep network is required when the mapping from states to actions is complex.

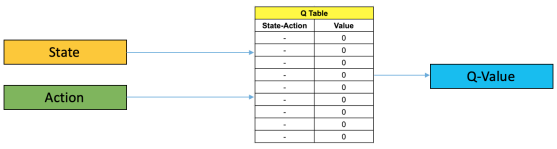
Deep Reinforcement Learning



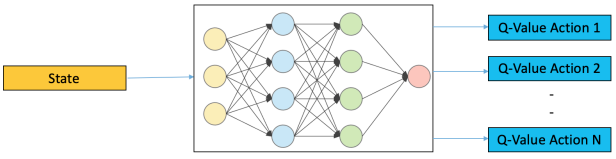
David Silver, Google DeepMind

www.amtc.cl

DQN – Deep Q-Learning



Q Learning



Deep Q Learning

https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

www.amtc.cl

DQN Training

- Learns to play video games simply by playing
- Can learn Q function by Q-learning

$$\Delta \boldsymbol{w} = \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{w}) - Q(S_t, A_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} Q(S_t, A_t; \boldsymbol{w})$$

- Core components of DQN include:
  - Target networks (Mnih et al. 2015)

$$\Delta \boldsymbol{w} = \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{w}^-) - Q(S_t, A_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} Q(S_t, A_t; \boldsymbol{w})$$

  - Experience replay (Lin 1992): replay previous tuples (s, a, r, s')

DRL, Hado van Hasselt

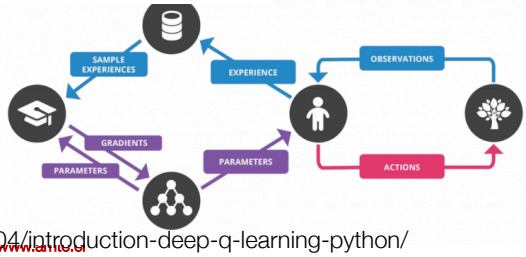
DQN Training

1. Preprocess and feed the game screen (state s) to our DQN, which will return the Q-values of all possible actions in the state
2. Select an action using the epsilon-greedy policy. With the probability epsilon, we select a random action *a* and with probability 1-epsilon, we select an action that has a maximum Q-value, such as a = argmax(Q(s,a,w))
3. Perform this action in a state *s* and move to a new state *s'* to receive a reward. This state s' is the preprocessed image of the next game screen. We store this transition in our replay buffer as <s,a,r,s'>
4. Next, sample some random batches of transitions from the replay buffer and calculate the loss
5. It is known that:

$$Loss = (r + \gamma max_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2$$

   which is just the squared difference between target Q and predicted Q
6. Perform gradient descent with respect to our actual network parameters in order to minimize this loss
7. After every C iterations, copy our actual network weights to the target network weights
8. Repeat these steps for *M* number of episodes

Key: replay buffer, target network

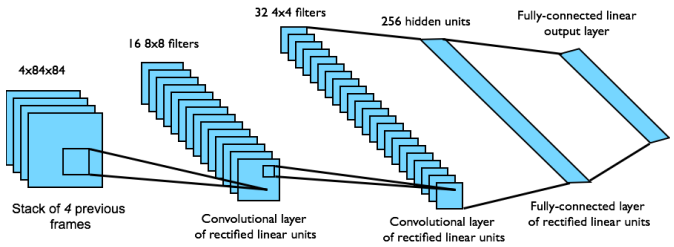


https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

DQN

DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



Network architecture and hyperparameters fixed across all games

David Silver, Google DeepMind

DQN Improvements

- Many later improvements to DQN
  - Double Q-learning (van Hasselt 2010, van Hasselt et al. 2015)
  - Prioritized replay (Schaul et al. 2016)
  - Dueling networks (Wang et al. 2016)
  - Asynchronous learning (Mnih et al. 2016)
  - Adaptive normalization of values (van Hasselt et al. 2016)
  - Better exploration (Bellemare et al. 2016, Ostrovski et al., 2017, Fortunato, Azar, Piot et al. 2017)
  - … many more …

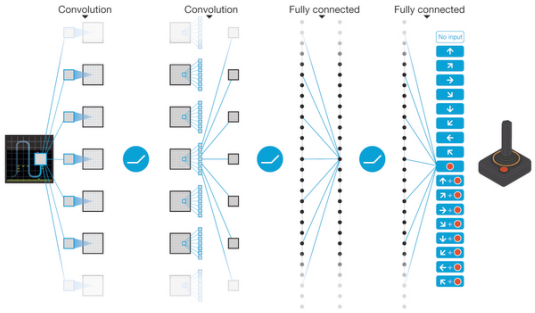
DRL, Hado van Hasselt

Double DQN

Improvements since Nature DQN

- Double DQN: Remove upward bias caused by $\max_a Q(s, a, \mathbf{w})$
  - Current Q-network $\mathbf{w}$ is used to select actions
  - Older Q-network $\mathbf{w}^-$ is used to evaluate actions

$$I = \left(r + \gamma Q(s', \underset{a'}{\text{argmax}}\ Q(s', a', \mathbf{w}), \mathbf{w}^-) - Q(s, a, \mathbf{w})\right)^2$$



David Silver, Google DeepMind

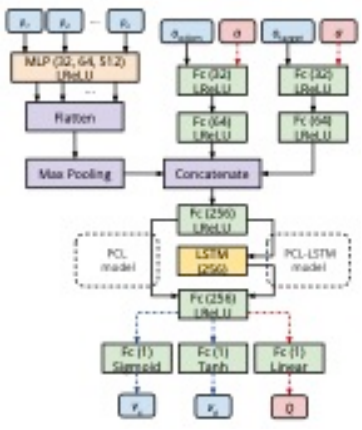
(Classical) Challenges of RL

- One of the main limitations of the use of RL approaches in real-world problems is the large number of learning trials required to learn complex behaviors. This can make its use non-viable in problems such as autonomous driving, x-copter flight control, or soccer robotics, in which the implementation of learning trials with real robots, in the real-world, may have a high cost.
- Another limitation of RL is the combinatorial explosion of state and action spaces of large-scale problems. This leads to intractable problems in terms of memory requirements or learning time.

Answers to (Classical) Challenges

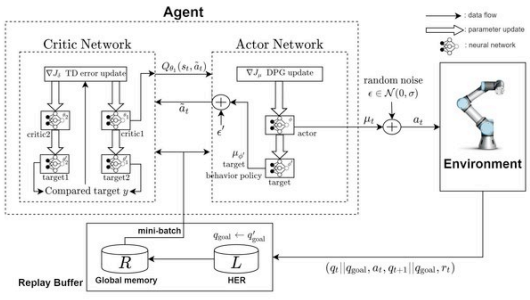
- Deep reinforcement learning
- Use of simulation tools
- Use of human knowledge: RL+IL (Imitation Learning), Interactive RL
- Decentralized RL
- Hierarchical RL
- Offline RL
- …

Example of Current DRL Architectures



Actor & critic networks (DDPG based). blue/red are only for the actor/critic LSTM is a RNN

Leiva & Ruiz –del-Solar, 2020



TD3 with RAMDP; kim et al. 2021

TD3. Twin-delayed DDPG
2 critic+1 actor network

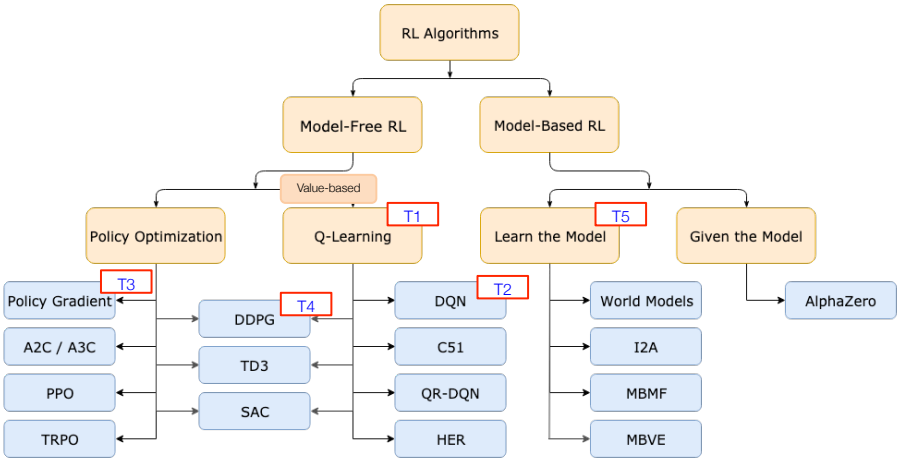
Fujimoto et al, 2018

(Current) Challenges of RL

- *Sim-to-real*
- Adaptation to new environments
- Transfer learning
- Training time
- Rewards design
- End-to-end versus “parts”
- “Explainability”
- Uncertainty & Safe Systems
- Hierarchical RL
- Real-world applications (autonomous vehicles, robots, etc.)

www.amtc.cl

Taxonomy & coursework



Tomado de https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

www.amtc.cl

Main References

- R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
- A. Geramifard et al., A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning, *Foundations and Trends in Machine Learning*, Vol. 6, No. 4 (2013) 375–454.
- J Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks*, 2015 – Elsevier.
- Leottau, L., Ruiz-del-Solar, Babuska, R. (2018). Decentralized Reinforcement Learning of Robot Behaviors, *Artificial Intelligence*, Vol. 256, March 2018, pp. 130-159.
- Perez-Dattari, R., Celemin, C., Franzese, G., Ruiz-del-Solar, J., Kober, J. (2020). Interactive Learning of Temporal Features for Control: Shaping Policies and State Representations From Human Feedback, *IEEE Robotics and Automation Magazine*, Vol. 27, Number 2, pp. 46-54, June 2020
- Celemin, C., Ruiz-del-Solar, J., Kober, J. (2019). A Fast Hybrid Reinforcement Learning Framework with Human Corrective Feedback, *Autonomous Robots*, Vol. 43, Issue 5, pp. 1173–1186, June 2019.

www.amtc.cl

Advanced Mining Technology Center

ADVANCED MINING TECHNOLOGY CENTER

Faculty of Mathematics and Physical Sciences
University of Chile

www.amtc.cl