Partially Observable Markov Decision Process in Reinforcement Learning

Shvechikov Pavel

National Research University Higher School of Economics,
Yandex School of Data Analysis

April 6, 2017

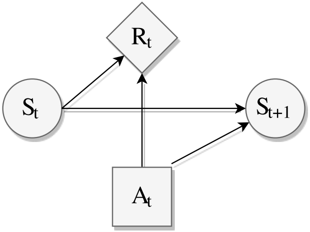
Overview

What is MDP?

Definition of Markov Decision Process

MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where

1. $\mathcal{S}$ – set of states of the world
2. $\mathcal{A}$ – set of actions
3. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \triangle(\mathcal{S})$ – state-transition function, giving us $p(s_{t+1} \mid s_t, a_t)$
4. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ – reward function, giving us $\mathbb{E}_R \left[R(s_t, a_t) \mid s_t, a_t \right]$.



Markov property

$$p(r_t, s_{t+1} \mid s_0, a_0, r_0, \ldots, s_t, a_t) = p(r_t, s_{t+1} \mid s_t, a_t)$$

(next state, expected reward) depend on (previous state, action)

What could be a problem?

Robot-cleaner in ideal world

1. have precise vison sensors
2. have precise map of a building
3. have perfect mechanics
4. should clean all floors

What could be a problem?

Robot-cleaner in ideal world

1. have precise vison sensors
2. have precise map of a building
3. have perfect mechanics
4. should clean all floors

How could this robot be modelled with MDP?

Sources of uncertainty

Typically autonomous agent's state is composed of

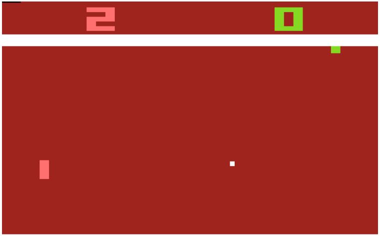
- measurement of environment
- measurment of agent itself

In real system there is even more uncertainty:

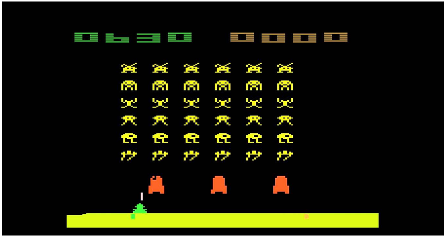
1. imperfect self-sensing (position, torque, velocity, etc.)
2. imperfect environment perception
3. incomplete observation of (nonstationary?) environment

How to incorporate uncertainty into decision making?

MDP problems are closer than they seem



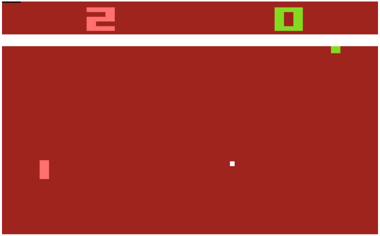
Pong



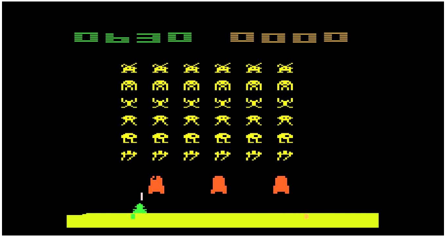
Space invaders

What is a state here?

MDP problems are closer than they seem



Pong



Space invaders

What is a state here?

128 bytes of unobserved Atari simulator RAM

POMDP is a powerful mathematical abstraction

- Industrial applications
  - Machine Maintenance (Shani et al., 2009)
  - Wireless networking (Pajarinen, 2013)
  - Wind Farms managing (Memarzadeh et al., 2014)
  - Aircraft Collision avoidance (Bai et al., 2012)
  - Choosing sellers in E-marketplaces (Irissappane et al., 2016)
- Assistive care
  - Assistant for patients with dementia (Hoey et al., 2010)
  - Home assistants (Pineau et al., 2003)
- Robotics
  - Grasping with a robotic arm (Hsiao et al., 2007)
  - Navigating an office (Spaan et al., 2005)
- Spoken dialog systems
  - Uncertainty in voice recognition (Young et al., 2013)

Outline

POMDP's place in a model world

Markov Models		Do we have control over the state transitions?	
		NO	**YES**
Are the states completely observable?	**YES**	**Markov Chain**	**MDP** Markov Decision Process
	NO	**HMM** Hidden Markov Model	**POMDP** Partially Observable Markov Decision Process

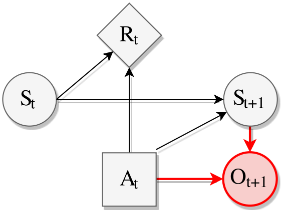
POMDP siblings

POMDP model

Definition

Partially Observed Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$

1. $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ are the same as in MDP
2. Ω – finite *set of observations*
3. $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \triangle(\Omega)$ – *observation function*, which gives, for each state and action, a probability distribution over Ω, i.e. $p(o \,|\, s_{t+1}, a_t) \quad \forall o \in \Omega$

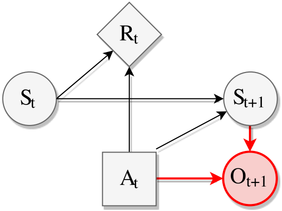


POMDP model

Definition

Partially Observed Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$

1. $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ are the same as in MDP
2. Ω – finite *set of observations*
3. $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \triangle(\Omega)$ – *observation function*, which gives, for each state and action, a probability distribution over Ω, i.e. $p(o \mid s_{t+1}, a_t) \quad \forall o \in \Omega$



But how to reason about what state are we currently in?

Reasoning about state uncertainty

Belief state

Distribution over state space, i.e $\sum_{s \in \mathcal{S}} b(s) = 1, \quad 0 \leq b(s) \leq 1$

$$\mathcal{A} = \{left, right\}, \quad p(\overline{A} \mid do(A)) = 0.1$$

$b(s_1)$	$b(s_2)$	$b(s_3)$	$b(s_4)$
0.333	0.333		0.333
0.1	0.450		0.450
0.1	0.164		0.736

1. **Belief** state **is sufficient statistic**: contains all of the information required for decision making (Striebel, 1965)
2. **POMDP is MDP** over properly updated belief states (Astrom, 1965)

Belief updating (Bayes filter)

Good news: belief updating is rather straightforward (Bayes Rule)

$$
\begin{aligned}
b'(s') = p(s' \mid o', a, b) &= \frac{p(o' \mid s', a) \cdot p(s' \mid a, b)}{\sum_o p(o' \mid s', a) \cdot p(s' \mid a, b)} \\
&\propto p(o' \mid s', a) \cdot p(s' \mid a, b) \\
&\propto p(o' \mid s', a) \sum_s p(s' \mid a, b, s) \cdot p(s \mid a, b) \\
&\propto p(o' \mid s', a) \sum_s p(s' \mid a, s) \cdot b(s)
\end{aligned}
$$

Bad news: belief updating can be computed exactly only for

1. discrete low-demensional state-spaces
2. linear-Gaussian dynamics (leading to Kalman filter), i.e.
   - $s' \sim \mathcal{N}(s' \mid T_s s + T_a a,\ \Sigma_s), \quad o' \sim \mathcal{N}(o' \mid O_s s', \Sigma_o)$
   - $R(s, a) = s^\top R_s s + a^\top R_a a$

Taxonomy of POMDP tasks

1. **Learning** / planning task
2. **Finite** / infinite horizon
3. **Online** / offline approach
4. **Approximate** / exact algorithm
5. **Discrete** / continous states
6. **Discrete** / continous action
7. **Discrete** / continous time
8. **Stationary** / non-stationary environment
9. **One** / many agents

Outline

1. What is wrong with MDP?
   - MDP remainder
2. POMDP details
   - Bayes filters
3. Approximate Learning
   - Deep Recurrent Q-Learning
   - Learning to Act by Predicting the Future
4. Explicit memory
   - Neural Map

Deep Recurrent Q-Learning (DRQN)

Problem:

we could not estimate $Q(s_t, a_t)$, since we don't know s_t

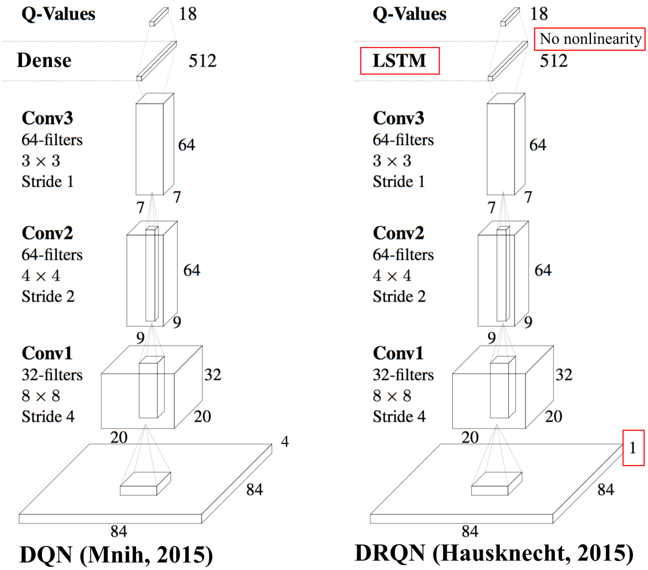
DRQN solution: (Hausknecht et al., 2015)

1. equip agent with memory h_t
2. approximate $Q(s_t, a_t)$ with $Q(o_t, h_{t-1}, a_t)$
3. eliminate dependence on o_t by modelling $h_t = LSTM(o_t, h_{t-1})$

Benefits:

1. simple approximate POMDP solver with *one frame* input
2. need only to model $Q(h_t, a_t)$
3. minor changes to vanilla DQN architecture

DRQN: architecture



DRQN with experience replay (ER) – I

No ER is needed if unlimited env access is granted

- train mutliple agents *asynchronoulsy*
- share all parameters (including LSTM's)
- do not share LSTM cell states
- leads to fast, almost iid training

Two original way-to-go's with ER (Hausknecht et al., 2015):

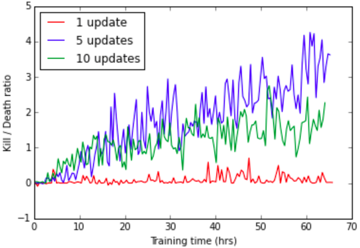
1. **Sequential Updates:** sample at random *full episode* from ER and perform sequential update
   - Random sampling from ER is violated
   - Updates are correlated
2. **Random Updates:** sample random time point in random episode from ER and train on k subsequent frames
   - LSTM hidden state must be zeroed at session start
   - First few updates are potentially erroneous

DRQN with experience replay (ER) – II

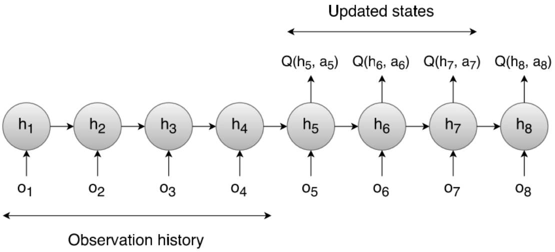
History based:
(Lample et al., 2016)

same as Random Updates, but update only last frames, i.e frames with indicies

$t + \color{red}{k}, \; \ldots, \; t + \tau - 1$



- Sample efficiency / correlation tradeoff



Deep Attention Recurrent Q-Network (DARQN)

Soft attention module: (Sorokin et al., 2015)

1. Flatten CNN output tensor into m^2 of D-demensional vectors $\{v_t^i\}_{i=1}^{m^2}$

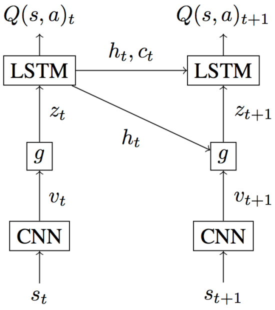
$$m \times m \times D \quad \Rightarrow \quad m^2 \times D$$

2. Apply attention module g:

$$\tilde{g}_t^i = \mathrm{Linear}(\mathrm{Tanh}(\mathrm{Linear}(v_t^i) + Wh_{t-1})))$$

$$g_t^i = \mathrm{Softmax}(\tilde{g}_t^1, ..., \tilde{g}_t^{m^2})$$

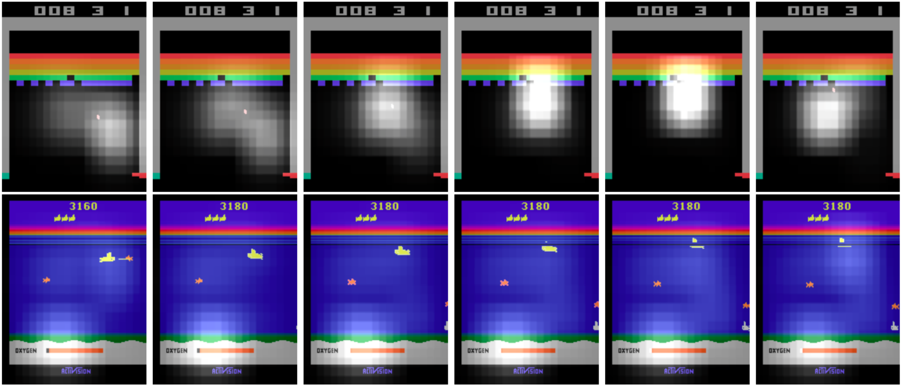
where $\mathrm{Linear}(x) = Ax + b$



3. Get context vector z_t as average of v_t^i weighted by g_t^i

$$z_t = \sum_{i=1}^{m^2} g_t^i v_t^i$$

DARQN: soft attention



Reinforcement Learning (RL) vs Supervized Learning (SL)

General leitmotif in RL: RL problem $\rightarrow$ SL problem

SL is used in RL problems in different ways, i.e. we could learn:

1. policy by regressing good actions on states (CEM)
2. policy by regressing Q-values on observed rewards (DQN)
3. policy as a product of reward weighted regression (A2C)
4. system dynamics, reward function and do planning (qNAF)
5. to predict full sensory input (improve exploration) (Oh, Guo, et al., 2015)
6. to predict subset of sensory input related to reward
    - rich low dimensionanal supervision signal $\rightarrow$ stabilization, learning acceleration
    - supports training without a fixed goal $\rightarrow$ robust to goal change at test time
    - not general

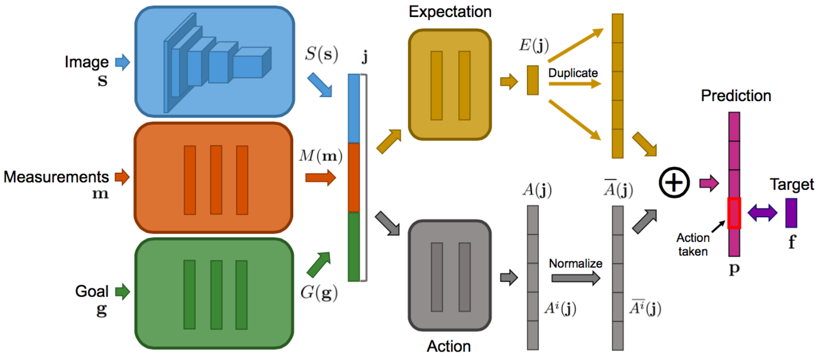
Direct Future Prediction (DFP)

(Dosovitskiy et al., 2016) :

1. Measurements, $\mathbf{m}_t$ – subset of sensory input (observation)
   - health, ammunition, number of opponents killed
2. Temporal offsets, $\tau_1, \ldots, \tau_n$ – horizons of prediction
3. Differences of measurement $\mathbf{f} = \langle \mathbf{m}_{t+\tau_1} - \mathbf{m}_t, \ldots, \mathbf{m}_{t+\tau_n} - \mathbf{m}_t \rangle$
4. Agent pursue goal defined by utility $u(\mathbf{f}, \mathbf{g}) = \mathbf{g}^\top \mathbf{f}$
   - goal vector $\mathbf{g}$ is hyperparameter
5. Estimate $\widehat{\mathbf{f}}_t^a = F(o_t, a, \mathbf{g} \mid \theta)$ with any parametric model
6. Train model with regression loss $\mathcal{L}(\theta) = ||\widehat{\mathbf{f}}_t^a - \mathbf{f}||_2^2$
   - classification loss can be used in case of discrete measurments
   - replay memory could be used but is not vital
7. At test time choose actions by $a_t = \arg\max_a \ \mathbf{g}^\top \widehat{\mathbf{f}}_t^a$

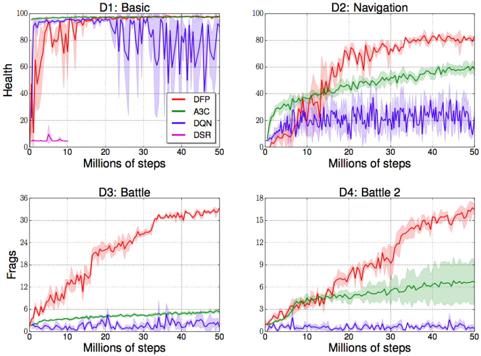
DFP: architecture (Dosovitskiy et al., 2016)

$$\widehat{\mathbf{f}} = \langle \widehat{\mathbf{f}}^{a_1}, \dots, \widehat{\mathbf{f}}^{a_n} \rangle = \langle \overline{A_1}(\mathbf{j}) + E(\mathbf{j}), \dots, \overline{A_n}(\mathbf{j}) + E(\mathbf{j}) \rangle$$



- epsilon greedy, train on 1,2,4,8,16,32 offsets,
- arg max is based on 8, 16, 32 offsets with weights (0.5, 0.5, 1)

DFP: results (Dosovitskiy et al., 2016)



test goal	(a) fixed goal (0.5, 0.5, 1) ammo	health	frags	(b) random goals [0, 1] ammo	health	frags	(c) random goals [−1, 1] ammo	health	frags
$(0.5, 0.5, 1)$	83.4	97.0	33.6	92.3	96.9	31.5	49.3	94.3	28.9
$(0, 0, 1)$	0.3	−3.7	11.5	4.3	30.0	20.6	21.8	70.9	24.6
$(1, 1, -1)$	28.6	−2.0	0.0	22.1	4.4	0.2	89.4	83.6	0.0
$(-1, 0, 0)$	1.0	−8.3	1.7	1.9	−7.5	1.2	0.9	−8.6	1.7
$(0, 1, 0)$	0.7	2.7	2.6	9.0	77.8	6.6	3.0	69.6	7.9

Outline

Memory is essential

Memory in RL have different flavours:

1. Temporal convolution memory (k last frames in DQN)
   - dead simple but very restrictive
2. RNN-like memory (LSTM layer in DRQN, DARQN)
   - capacious but suitable only for simple tasks
3. Bank-like memory of embeddings with deliberate read access (Oh, Chockalingam, et al., 2016)
   - is more intelligent but may be redundant, requires expert
4. Human-like spatial memory with deliberate read access (Parisotto et al., 2017)
   - has great potential but needs structural assumptions

Neural map (NM): (Parisotto et al., 2017)

Key features:

- structured memory designed specifically for RL agents in 3D
- size and comp. cost doesn't grow with env. time horizon
- training alogrithm: A2C with synchronous updates

$$
\boxed{\begin{aligned}
r_t &= \mathrm{read}(M_t) \\
c_t &= \mathrm{context}(s_t, M_t, r_t) \\
w_{t+1}^{(x_t, y_t)} &= \mathrm{write}(s_t, r_t, c_t, M_t^{(x_t, y_t)}) \\
M_{t+1} &= \mathrm{update}(M_t, w_{t+1}^{(x_t, y_t)}) \\
o_t &= [r_t, c_t, w_{t+1}^{(x_t, y_t)}] \\
\pi(a \mid s_t) &= \mathrm{Softmax}(f(o_t))
\end{aligned}}
$$

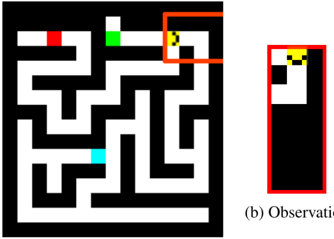
$[x_1, \ldots, x_n]$ – concat operator

1. M_t – tensor $C \times H \times W$
2. r_t – global info about NM
3. s_t – state embedding
4. $x_t \in \{1, ..W\}, y_t \in \{1, ..H\}$ agent position on map
5. $w_{t+1}^{(x_t, y_t)}$ – features to write
6. o_t – NM output at tick t
7. $f(\cdot)$ – policy network

Neural map (NM): operation details

1. Global read, $r_t = \mathrm{CNN}(M_t)$ – 3-layer convolutional network (3x3n8), 256fc, 32fc
2. Context, c_t – targeted extract of memory:
   1. $q_t = W[s_t, r_t]$ – query, relevant to current state
   2. $\alpha_t^{(x,y)} \propto \exp(q_t^\top M_t^{(x,y)})$ – normalized (over (x, y) axis) similarities between NM features and query
   3. $c_t = \sum_{(x,y)} \alpha_t^{(x,y)} M_t^{(x,y)}$ – weighted average of NM features
3. Local write computes new $M_t^{(x,y)}$ features
   1. $w_{t+1}^{(x_t,y_t)} = g([s_t, r_t, c_t, M_t^{(x,y)}])$, where $g(\cdot)$ is another neural network (e.g. GRU-based) with inner state equal to $M_t^{(x,y)}$
4. Update is straightforward – rewrite only features corresponding to current location x_t, y_t $M_{t+1}^{(x,y)} = w_{t+1}^{(x_t,y_t)}$

Nerual map: empirical results



(a) Maze (b) Observation

1. Test on 1000 unseen mazes
2. Episode ends in train/test – 100/500 ticks
3. Positive reward for finding torch with proper light
4. Negative if found wrong light

Agent	Goal-Search					
	Train			Test		
	7-11	13-15	Total	7-11	13-15	Total
Random	41.9%	25.7%	38.1%	46.0%	29.6%	38.8%
LSTM	60.6%	41.8%	59.3%	65.5%	47.5%	57.4%
MemNN-32	85.1%	58.2%	77.8%	92.6%	69.7%	83.4%
Neural Map	92.4%	80.5%	89.2%	93.5%	87.9%	91.7%
Neural Map (GRU)	**97.0%**	**89.2%**	**94.9%**	**97.7%**	**94.0%**	**96.4%**

Thank you!

References I

- Astrom, Karl J (1965). "Optimal control of Markov processes with incomplete state information". In: *Journal of mathematical analysis and applications* 10.1, pp. 174–205.
- Bai, Haoyu et al. (2012). "Unmanned aircraft collision avoidance using continuous-state POMDPs". In: *Robotics: Science and Systems VII* 1, pp. 1–8.
- Dosovitskiy, Alexey et al. (2016). "Learning to act by predicting the future". In: *arXiv preprint arXiv:1611.01779*.
- Hausknecht, Matthew et al. (2015). "Deep recurrent q-learning for partially observable mdps". In: *arXiv preprint arXiv:1507.06527*.
- Hoey, Jesse et al. (2010). "Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process". In: *Computer Vision and Image Understanding* 114.5, pp. 503–519.

References II

- Hsiao, Kaijen et al. (2007). "Grasping pomdps". In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, pp. 4685–4692.
- Irissappane, Athirai Aravazhi et al. (2016). "A Scalable Framework to Choose Sellers in E-Marketplaces Using POMDPs." In: *AAAI*, pp. 158–164.
- Lample, Guillaume et al. (2016). "Playing FPS games with deep reinforcement learning". In: *arXiv preprint arXiv:1609.05521*.
- Memarzadeh, Milad et al. (2014). "Optimal planning and learning in uncertain environments for the management of wind farms". In: *Journal of Computing in Civil Engineering* 29.5, p. 04014076.
- Oh, Junhyuk, Valliappa Chockalingam, et al. (2016). "Control of memory, active perception, and action in minecraft". In: *arXiv preprint arXiv:1605.09128*.

References III

- Oh, Junhyuk, Xiaoxiao Guo, et al. (2015). "Action-conditional video prediction using deep networks in atari games". In: *Advances in Neural Information Processing Systems*, pp. 2863–2871.
- Pajarinen, Joni et al. (2013). "Planning under uncertainty for large-scale problems with applications to wireless networking". In:
- Parisotto, Emilio et al. (2017). "Neural Map: Structured Memory for Deep Reinforcement Learning". In: *arXiv preprint arXiv:1702.08360*.
- Pineau, Joelle et al. (2003). "Towards robotic assistants in nursing homes: Challenges and results". In: *Robotics and autonomous systems* 42.3, pp. 271–281.
- Shani, Guy et al. (2009). "Improving existing fault recovery policies". In: *Advances in Neural Information Processing Systems*, pp. 1642–1650.

References IV

- Sorokin, Ivan et al. (2015). "Deep attention recurrent Q-network". In: *arXiv preprint arXiv:1512.01693*.
- Spaan, Matthijs TJ et al. (2005). "Perseus: Randomized point-based value iteration for POMDPs". In: *Journal of artificial intelligence research* 24, pp. 195–220.
- Striebel, Charlotte (1965). "Sufficient statistics in the optimum control of stochastic systems". In: *Journal of Mathematical Analysis and Applications* 12.3, pp. 576–592.
- Young, Steve et al. (2013). "Pomdp-based statistical spoken dialog systems: A review". In: *Proceedings of the IEEE* 101.5, pp. 1160–1179.