Machine Intelligence 1

4.2 Reinforcement Learning – Improvement

Prof. Dr. Klaus Obermayer

Fachgebiet Neuronale Informationsverarbeitung (NI)

WS 2016/2017

4.2.1 Policy Improvement

Valuation of state-action pairs: Q-values

- so far: agents learns where to *expect* rewards
- now: agents learn how to *act* to maximize rewards
- **Q-value** is the value of state $\underline{\mathbf{x}}_i$ **after** choosing action $\underline{\mathbf{a}}_k$

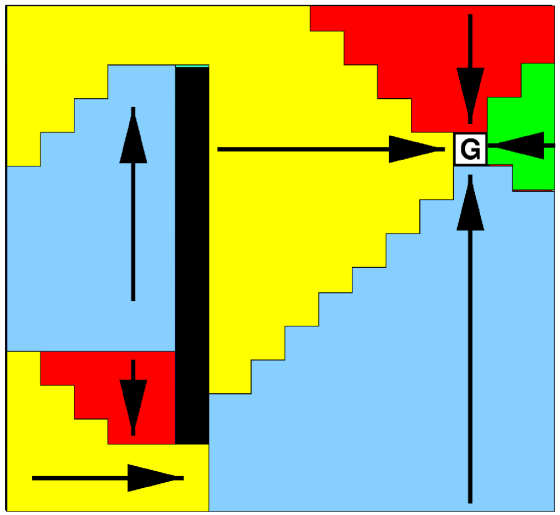
$$Q^\pi(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, r(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}) \;\middle|\; \begin{array}{l} \underline{\mathbf{x}}^{(0)} = \underline{\mathbf{x}}_i \,, \quad \underline{\mathbf{a}}^{(0)} = \underline{\mathbf{a}}_k \\ \underline{\mathbf{x}}^{(t+1)} \sim P(\cdot|\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}) \\ \underline{\mathbf{a}}^{(t+1)} \sim \pi(\cdot|\underline{\mathbf{x}}^{(t+1)}) \end{array} \right]$$

Bellman equation for Q-values

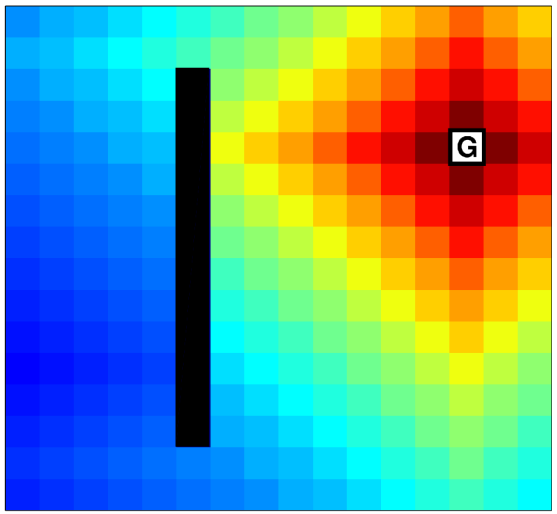
$$Q^\pi(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) = r(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) + \gamma \sum_{j=1}^{S} P(\underline{\mathbf{x}}_j|\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) \underbrace{\sum_{l=1}^{A} \pi(\underline{\mathbf{a}}_l|\underline{\mathbf{x}}_j) \, Q^\pi(\underline{\mathbf{x}}_j, \underline{\mathbf{a}}_l)}_{V^\pi(\underline{\mathbf{x}}_j)}$$

Example

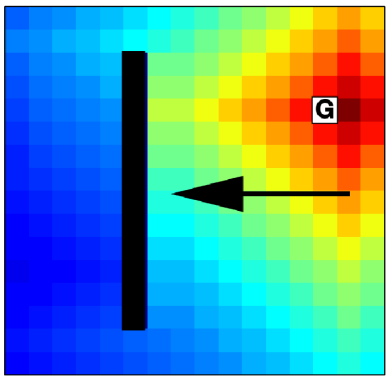
policy π



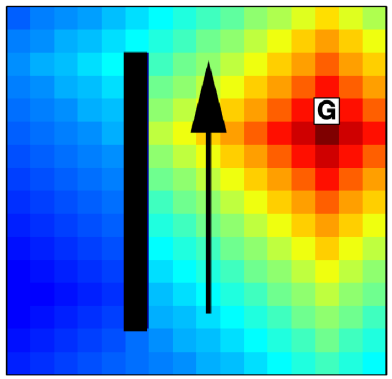
value V^π



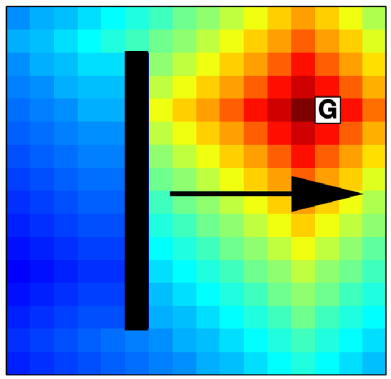
Q-value $Q^\pi(\cdot, \text{left})$



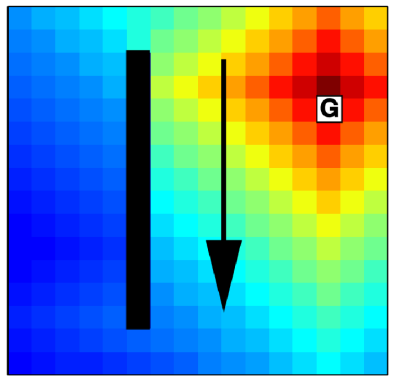
Q-value $Q^\pi(\cdot, \text{up})$



Q-value $Q^\pi(\cdot, \text{right})$



Q-value $Q^\pi(\cdot, \text{down})$



(optimal policy π for navigation to rewarded goal **G**, value V^π and Q-values Q^π for $\gamma = 0.9$)

Q-value estimation

$$V^\pi(\underline{\mathbf{x}}_i) = \overbrace{\sum_{k=1}^{A} \pi(\underline{\mathbf{a}}_k|\underline{\mathbf{x}}_i)\, r(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)}^{r^\pi(\underline{\mathbf{x}}_i)} + \gamma \sum_{j=1}^{S} \overbrace{\sum_{k=1}^{A} \pi(\underline{\mathbf{a}}_k|\underline{\mathbf{x}}_i) P(\underline{\mathbf{x}}_j|\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)}^{P^\pi(\underline{\mathbf{x}}_j|\underline{\mathbf{x}}_i)} V^\pi(\underline{\mathbf{x}}_j)$$

$$Q^\pi(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) = r(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) + \gamma \sum_{j=1}^{S} \sum_{l=1}^{A} \underbrace{\pi(\underline{\mathbf{a}}_l|\underline{\mathbf{x}}_j)\, P(\underline{\mathbf{x}}_j|\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)}_{P'^\pi(\underline{\mathbf{x}}_j, \underline{\mathbf{a}}_l|\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)} Q^\pi(\underline{\mathbf{x}}_j, \underline{\mathbf{a}}_l)$$

- P'^π is a transition model in the space of states and actions

$$\sum_{j=1}^{S} \sum_{l=1}^{A} P'^\pi(\underline{\mathbf{x}}_j, \underline{\mathbf{a}}_l|\underline{\mathbf{x}}, \underline{\mathbf{a}}) = 1, \qquad \forall \underline{\mathbf{x}} \in \mathcal{X}, \quad \forall \underline{\mathbf{a}} \in \mathcal{A}$$

- **Q-values are value functions** of state-action pairs
  - with reward function r and transition model P'^π
  - all techniques for value estimation apply to Q-values
  - much larger space $\rightsquigarrow$ requires more samples to estimate

SARSA

- inductive TD(0) learning for Q-values using TD-errors
  - with $\underline{\mathbf{x}}^{(t+1)} \sim P(\cdot|\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})$ and $\underline{\mathbf{a}}^{(t+1)} \sim \pi(\cdot|\underline{\mathbf{x}}^{(t+1)})$

$$\Delta Q_t \quad = \quad r_t \; + \; \gamma \, Q(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)}) \; - \; Q(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})$$

- **on-policy**: SARSA estimates the Q-values Q^π of sampling policy π
  - named after the type of training data:
    State$_t$-Action$_t$-Reward$_t$-State$_{t+1}$-Action$_{t+1}$

- a SARSA(λ) variant with eligibility traces exists

SARSA algorithm for Q-value estimation

for $t \in \{0, \ldots, p-1\}$ **do**

$$Q_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} \quad \leftarrow \quad Q_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} + \eta \Big(\underbrace{r_t + \gamma \, Q_{(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)})} - Q_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})}}_{\Delta Q_t} \Big)$$

end

Off-policy value estimation

- sampling policy affect SARSA updates through $\underline{\mathbf{a}}^{(t+1)} \sim \pi(\cdot|\underline{\mathbf{x}}^{(t+1)})$

$$\Delta Q_t \quad = \quad r_t \; + \; \gamma \, Q(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)}) \; - \; Q(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})$$

- why not estimate Q-values of a *different* policy π' (**off-policy**[1])?

$$\Delta Q'_t \quad = \quad r_t \; + \; \gamma \sum_{k=1}^{A} \pi'(\underline{\mathbf{a}}_k \,|\, \underline{\mathbf{x}}^{(t+1)}) \, Q(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}_k) \; - \; Q(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})$$

- contracts for an ergodic Markov chain: $\lim_{p \to \infty} \mathbb{E}\left[\sum_{t=0}^{p} \Delta Q'_t\right] = Q^{\pi'}$

- Markov chain $\{x_t, a_t, r_t\}_{t=1}^{p}$ used for training can be reused when π' changes!

[1] Note that the concepts underlying TD(λ) cannot be applied to off-policy learning.

Optimal policy

- the optimal policy π^* should maximize value function V^{π^*}
- maximizing values:

$$\begin{aligned} \max_{\pi^*} V^{\pi^*}(\underline{\mathbf{x}}_i) &= \max_{\pi^*} \sum_{k=1}^{A} \pi^*(\underline{\mathbf{a}}_k|\underline{\mathbf{x}}_i)\, Q^{\pi^*}(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) \\ &= \max_{1\le k\le A} Q^{\pi^*}(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) \end{aligned}$$

- π^* chooses the action $\underline{\mathbf{a}}_k$ with the largest associated Q-value

Greedy policy π' on a Q-value function Q^π

$$\pi'(\underline{\mathbf{a}}_k \,|\, \underline{\mathbf{x}}_i) = \begin{cases} 1 & , \text{ if } k = \underset{1\le l\le A}{\operatorname{argmax}}\, Q^\pi(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_l) \\ 0 & , \text{ otherwise} \end{cases}$$

Policy iteration

- any policy π_i is *improved* by the greedy policy π_{i+1} of its Q-value Q^{π_i}:

$$Q^{\pi_{i+1}}(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k) \quad \geq \quad Q^{\pi_i}(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)$$

Policy iteration (PI)

initialize first policy π_0 randomly
while *Q-values not converged* **do**
  estimate $\tilde{Q}^{\pi_i} :\approx Q^{\pi_i}$ of policy π_i
  choose π_{i+1} to be the greedy policy on $\tilde{Q}^{\pi_i}$
end

(see Bertsekas, 2007, for convergence with estimated Q-values)

On-line policy improvement: Q-learning

$$\tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}) \leftarrow \tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}) + \eta \Big(\underbrace{r_t + \gamma \overbrace{\max_{1 \le k \le A} \tilde{Q}^*(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}_k)}^{\text{greedy policy improvement}} - \tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})}_{\text{TD-error } \Delta Q_t} \Big)$$

- Q-learning estimates the Q-value Q^{π^*} of the optimal policy π^*
- Q-learning is a contraction mapping for *any* ergodic Markov chain

$$\lim_{p \to \infty} \mathbb{E}\Big[\sum_{t=0}^{p} \Delta Q_t\Big] = Q^{\pi^*}$$

(Watkins and Dayan, 1992)

On-policy vs. off-policy value estimation

- Q-value estimation requires many samples
  - on-policy estimation discards these after policy improvement
- policy iteration typically uses off-policy *batch* estimation
  - batch and model-based estimates are very sample efficient
- Q-learning is the only *online* off-policy algorithm

4.2.2 Exploration-Exploitation

The exploration-exploitation dilemma

- policy iteration and Q-learning learn the *optimal* Q-values Q^{π^*}
  - as long as all state-action pairs are sufficiently well **explored**

  $\Rightarrow$ agent should transition all state-action pairs to acquire knowledge

- an agent has the task to maximize the overall reward
  - this tasks requires to **exploit** learned knowledge

  $\Rightarrow$ agent should follow the greedy policy to collect reward

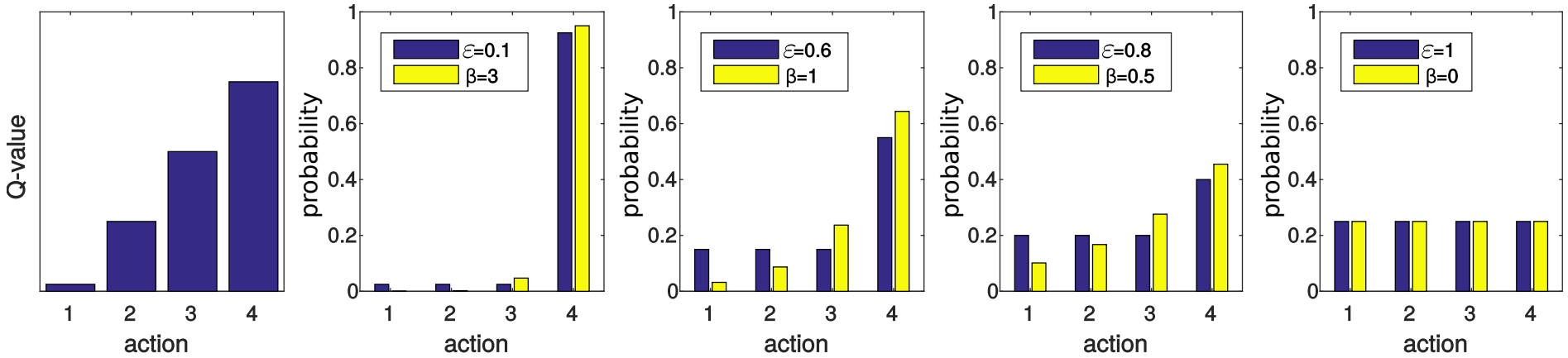
- a good sampling policy π should balance both requirements
  - find *expected* reward through *exploitation* of actions with high Q-values
  - find *potential* reward through *exploration* of alternative actions

ε-greedy and softmax

- ε-**greedy policy** controls exploration with parameter ε
  - random actions with probability ε
  - otherwise (probability $1 - \varepsilon$) greedy
- **softmax policy** controls exploration with noise parameter β

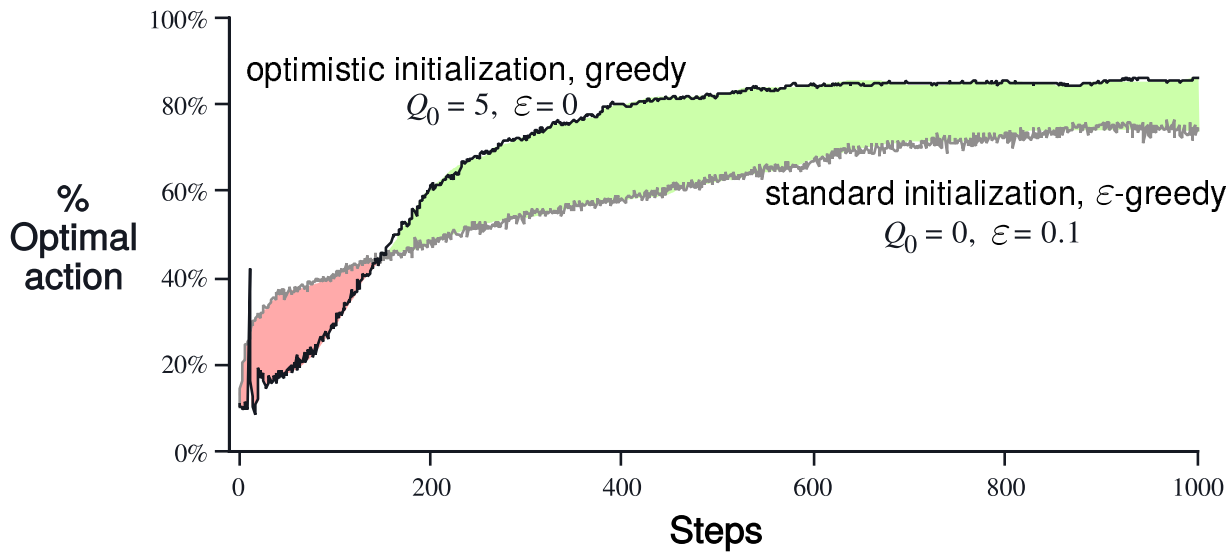
$$\pi(\underline{\mathbf{a}}_k \,|\, \underline{\mathbf{x}}_i) \quad = \quad \frac{\exp\left(\beta\, Q(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_k)\right)}{\sum_{l=1}^{A} \exp\left(\beta\, Q(\underline{\mathbf{x}}_i, \underline{\mathbf{a}}_l)\right)}$$

- good parameters ε and β depend on the agent's envisioned lifetime
  - often first slowly annealed and then fixed at almost-greedy level



Optimistic initialization

- initializing all Q-values with high $Q_0 \geq \frac{r_{\max}}{1-\gamma}$ directs exploration
- unexplored actions are initially very attractive
- Q-values are reduced until *exploitation* takes over
- initially slower, but converges faster to the limit
- can be applied to SARSA(λ) and Q-learning



(2-armed bandit task from Sutton and Barto, 1998)

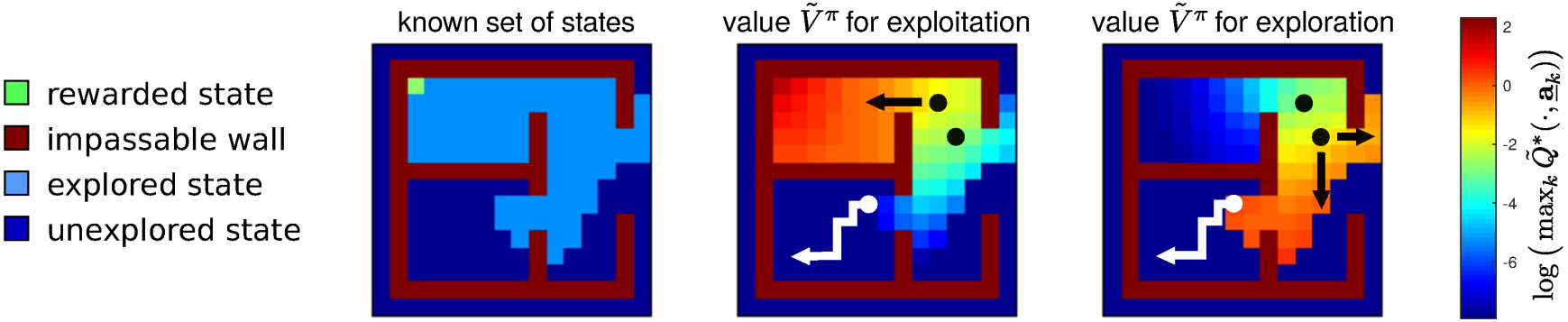
Model-based exploration: E^3 and R-MAX

1. explore (e.g. random-walk) until some states are visited often enough
   - those states constitute the "known set" $\bar{\mathcal{S}} \subset \mathcal{X}$
2. estimate models of transitions and rewards in $\bar{\mathcal{S}}$
   - by counting observed transitions and rewards
3. use these models to calculate two Q-value functions in $\bar{\mathcal{S}}$:

"exploit": Q-value of *observed* rewards, transitions leaving $\bar{\mathcal{S}}$ yield no reward

"explore": Q-value where *only* transitions leaving $\bar{\mathcal{S}}$ yield a reward of $\frac{r_{\max}}{1-\gamma}$

4. in $\bar{\mathcal{S}}$, choose actions according to the larger Q-value function
   - outside of $\bar{\mathcal{S}}$, continue random-walk and add explored states to the known set $\bar{\mathcal{S}}$ when visited sufficiently often

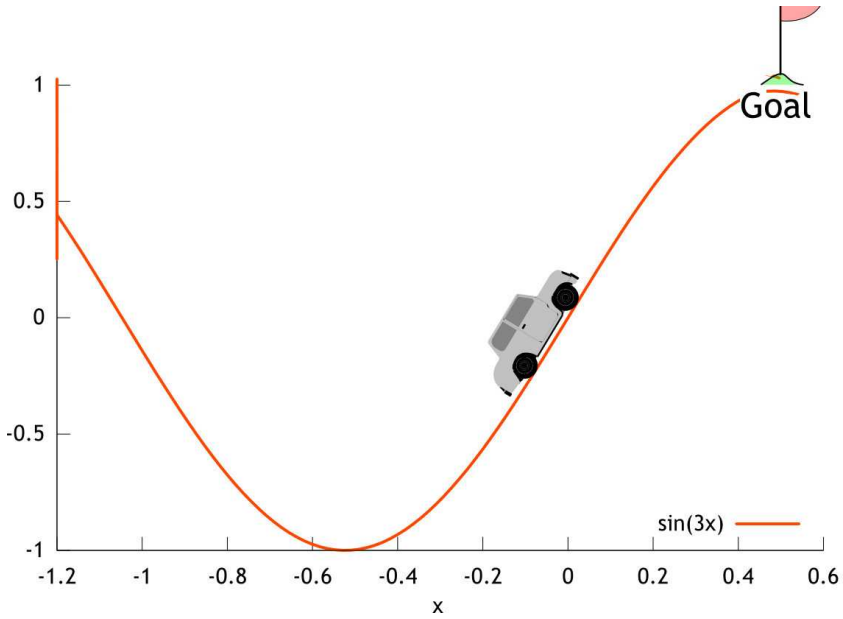


(E^3, Kearns and Singh, 2002) (R-MAX, Brafman and Tennenholtz, 2002)

4.2.3 A Thorough Example: Mountain Car

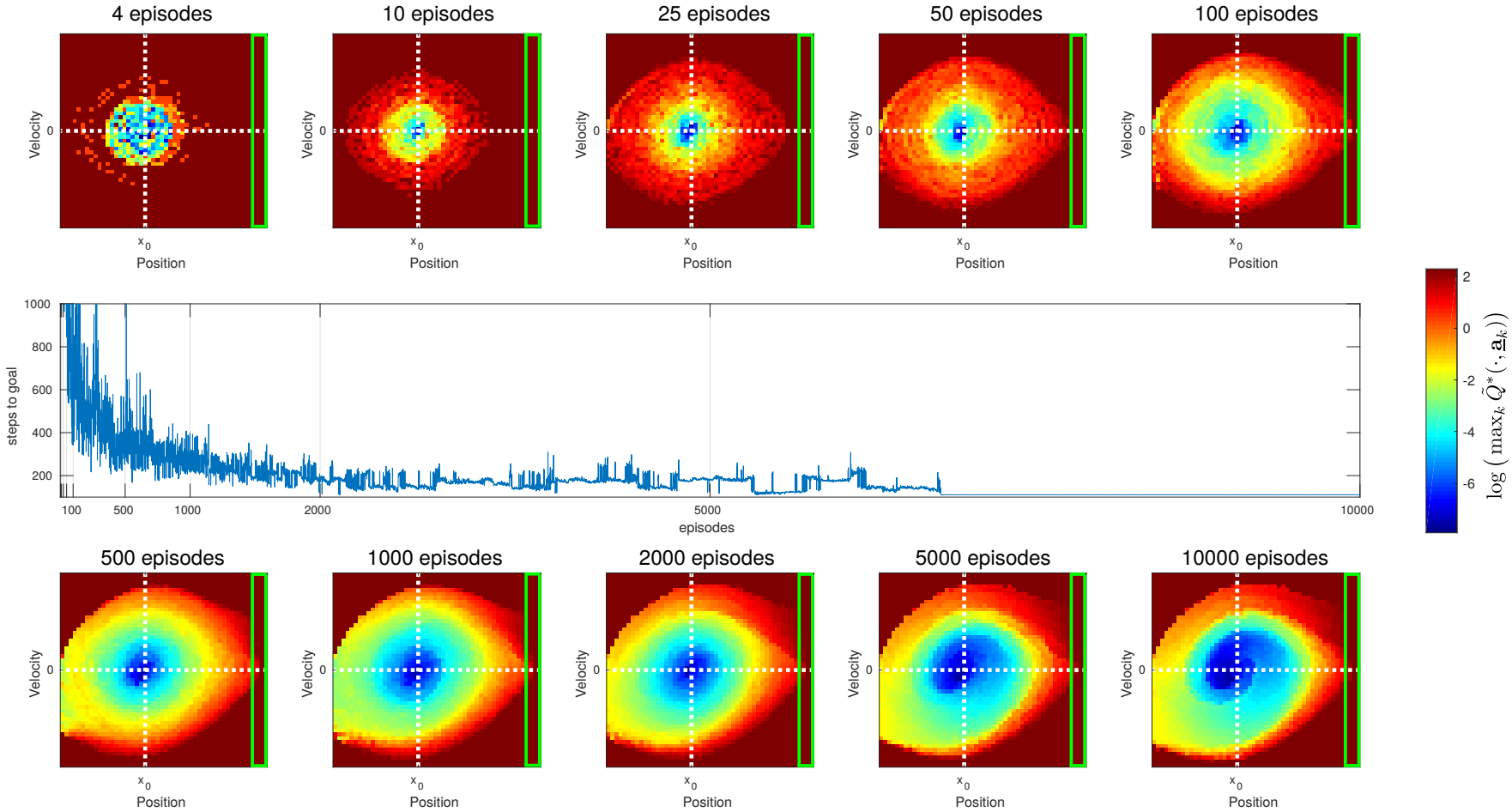
Mountain car

- a car in a valley between mountains
  - $\mathcal{X}$: position and velocity
- agent drives the car
  - $\mathcal{A}$: forward, backward, nothing
    (i.e., accelerate the car by $+a$, $-a$ and 0)
- dynamics are given by physics
  - deterministic transition model P
  - gravitation but no friction
- goal: reach right hilltop
  - reward $r=0$, except $r=1$ at goal
- exploration/exploitation is performed in **episodes**
  - start motionless at the bottom
  - end after 1000 steps or when the goal is reached
  - very few episodes with random control reach the goal



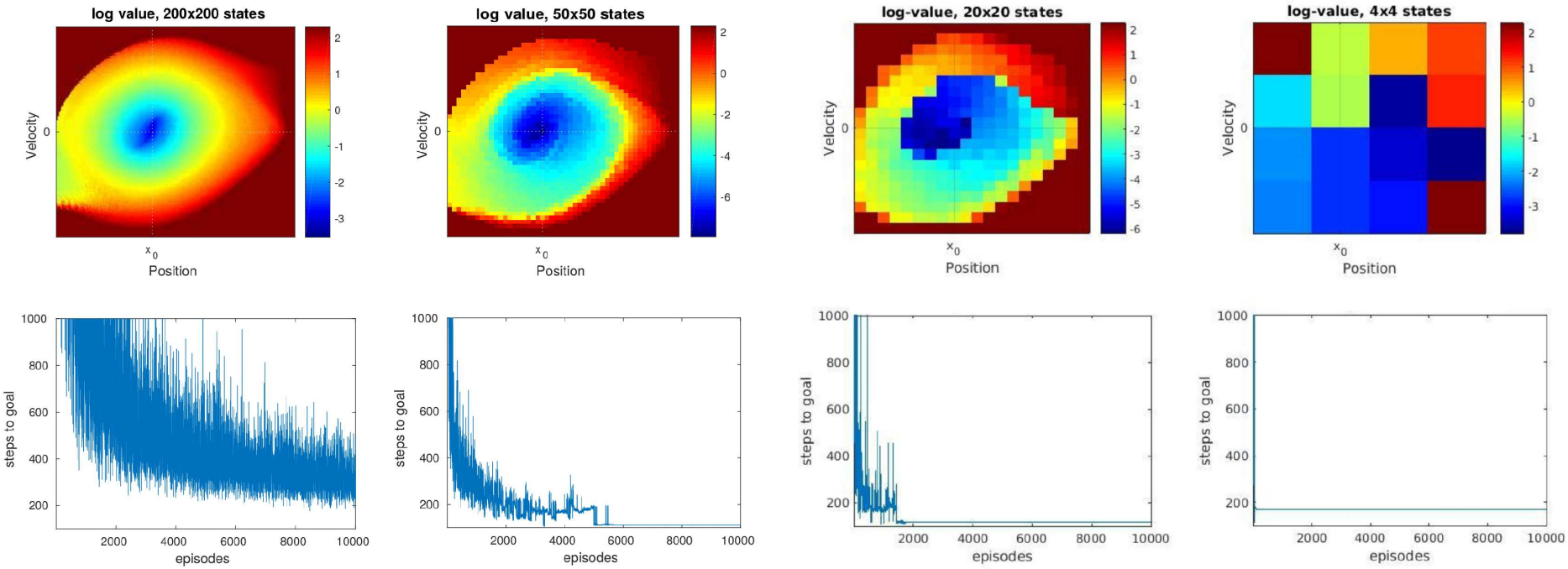
Q-learning: Development of the value function

- Q-learning with optimistic initialization and ϵ-greedy sampling policy
  - $\epsilon = 0, Q_0 = 10, \eta = 0.5, \gamma = 0.9$, discretization in 50×50 states



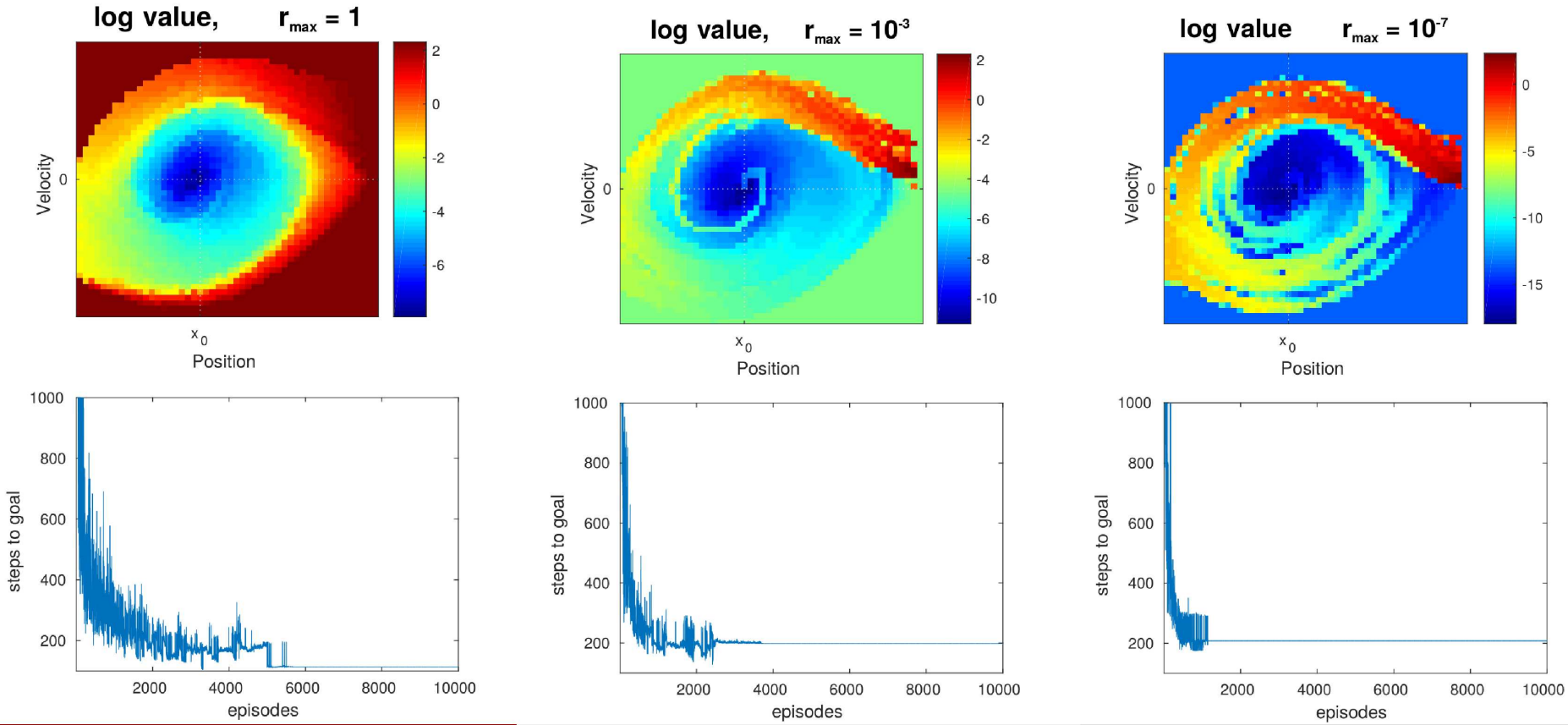
Parcellation of state space

- finer grids lead to better policies, but learning requires more episodes
- coarse discretizations may fail to learn a good policy



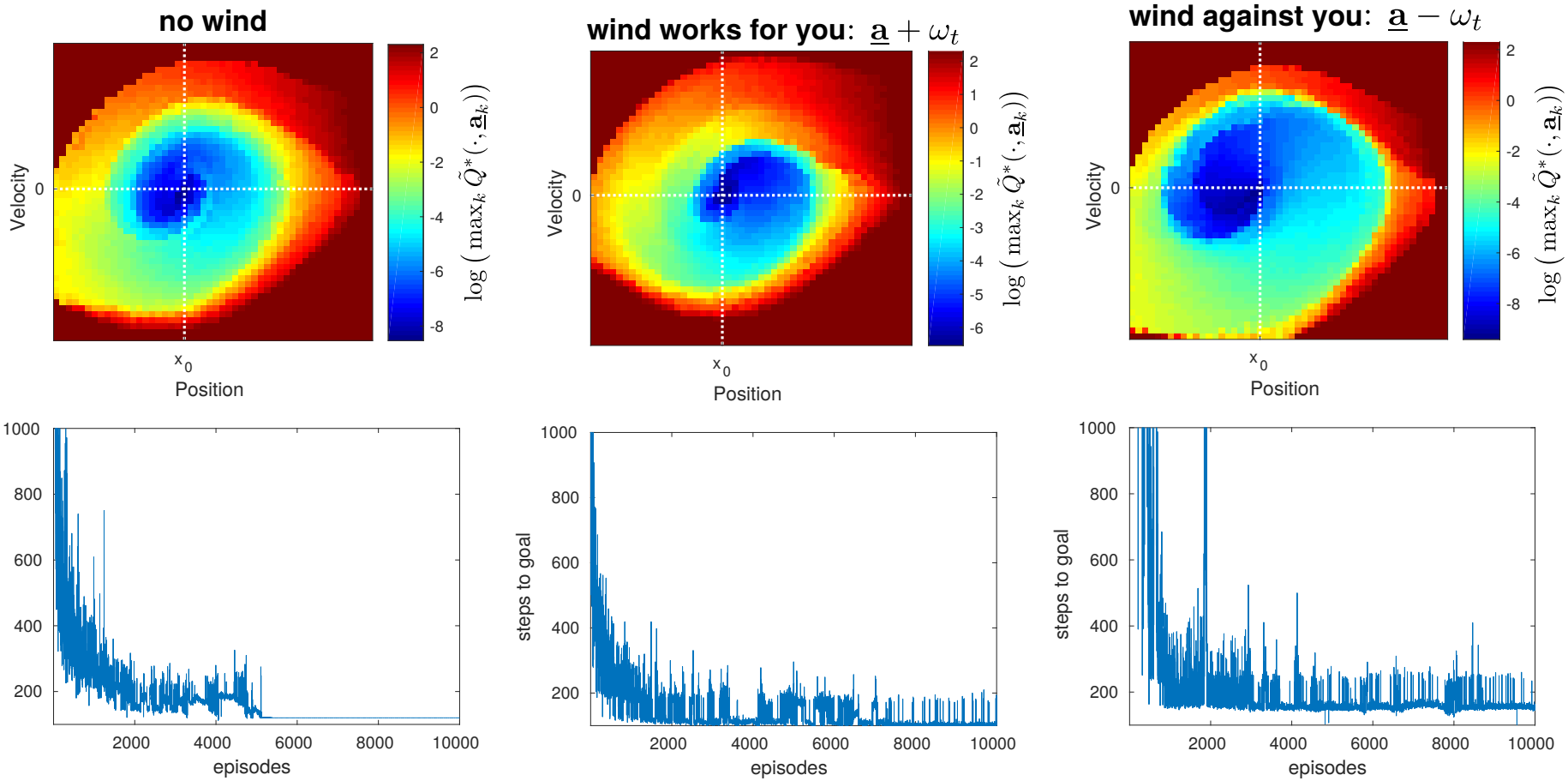
Optimistic initialization

- without optimistic initialization, ε-greedy Q-learning fails
  - observed reward must be propagated over 100+ time steps
  - ε-greedy does not reach the goal often enough (for any ε)
- small initial $Q_0 = \frac{r_{\max}}{1-\gamma}$ converges faster, but yields suboptimal policies



Non-deterministic transitions

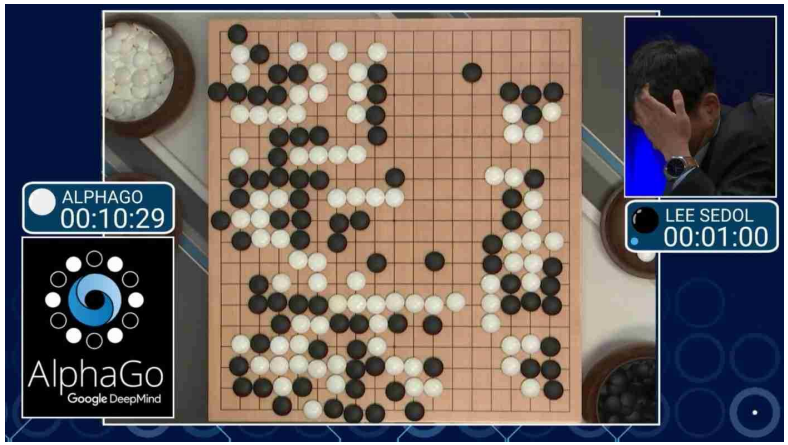
- i.i.d. wind $\underline{\mathbf{a}} \pm |\omega_t|$ with $\omega_t \sim \mathcal{N}(0, \sigma^2)$, $\sigma = \frac{2}{3}$



4.2.4 Approximate RL

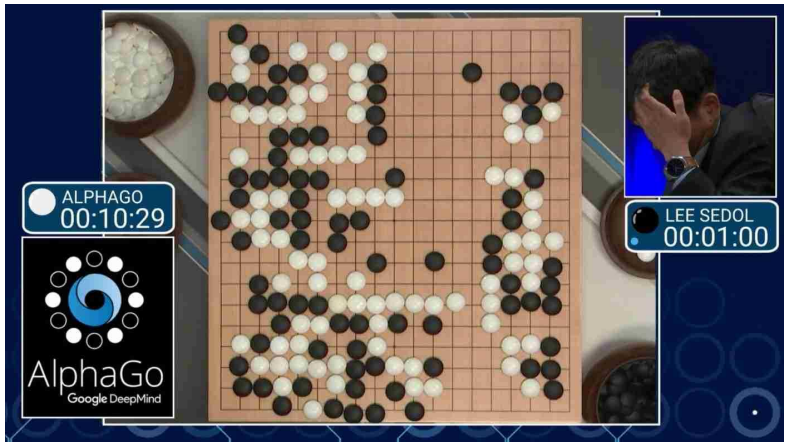
Real-world application of RL

- realistic state spaces can be huge
  - e.g. game of GO has $3^{19\times 19} \approx 10^{172}$ states (Silver et al., 2016)



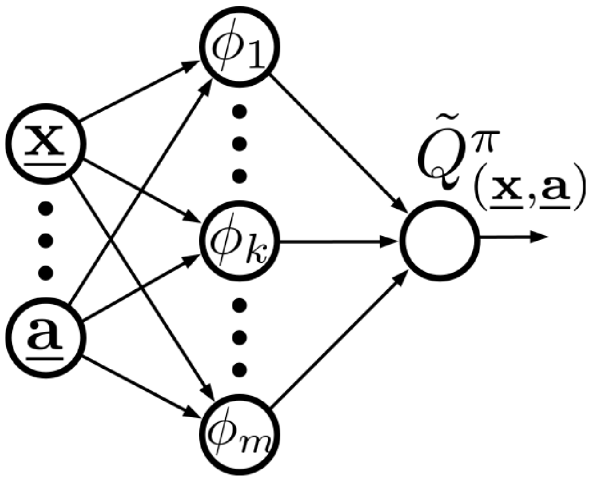
Real-world application of RL

- realistic state spaces can be huge
  - e.g. game of GO has $3^{19\times19} \approx 10^{172}$ states (Silver et al., 2016)
- real world states/actions are often continuous
  - e.g. autonomous helicopter control (Abbeel et al., 2007)

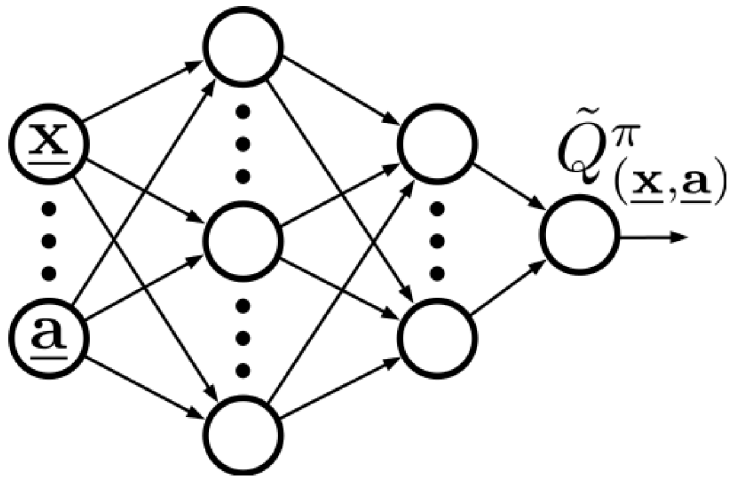


Value function approximation

- previous methods require values for all states
  - continuous state spaces must be discretized
- often impossible to visit every state-action pair
- use function approximation to generalize Q-values
  - training set $\{\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}, r_t\}_{t=0}^{p}$ drawn by an ergodic Markov chain
  - model classes: *linear functions*, *multi-layer perceptrons*, ...

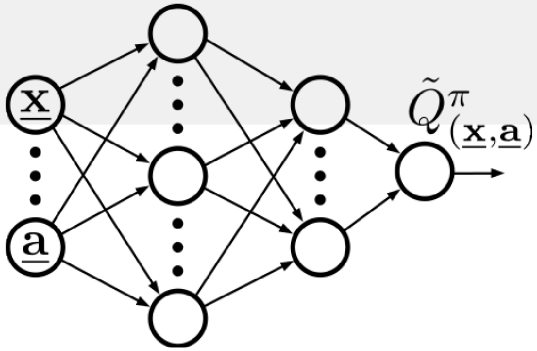


basis-function approach



deep neural network

Minimization of Bellman residuals

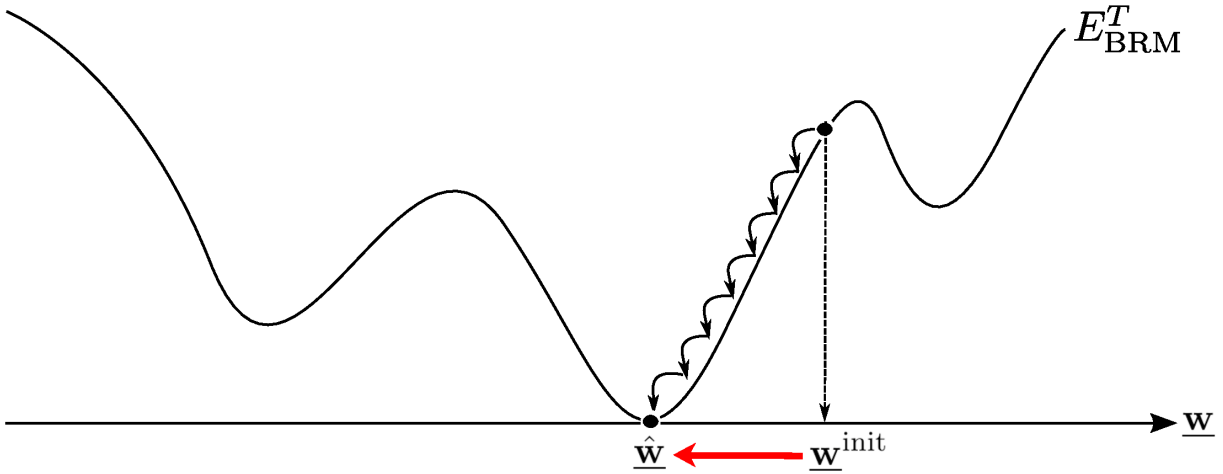


- quadratic error with parameterized function $\tilde{Q}^{\pi}(\underline{\mathbf{x}}, \underline{\mathbf{a}}; \underline{\mathbf{w}})$

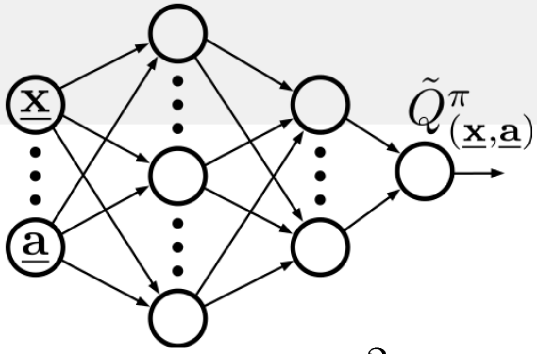
$$E^T_{\text{BRM}} = \frac{1}{2p} \sum_{t=0}^{p-1} \Big(\overbrace{r_t + \gamma \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)}; \underline{\mathbf{w}})}^{\text{target value for } \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})} - \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}) \Big)^2$$

TD-error ΔQ_t

- minimize E^T_{BRM} e.g. with gradient descend



Gradients



$$E^T_{\text{BRM}} = \frac{1}{2p}\sum_{t=0}^{p-1}\Big(\overbrace{r_t + \gamma\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})}^{\text{target value for } \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})} - \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})\Big)^2$$

TD-error ΔQ_t

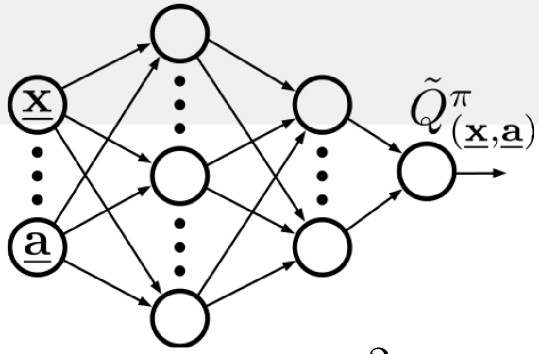
- two possible gradients for *online learning*
- **residual gradient** for online gradient descend

$$\frac{\partial(\Delta Q_t)^2}{\partial \underline{\mathbf{w}}} = -2\Delta Q_t\left(\frac{\partial\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}} - \gamma\frac{\partial\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}\right)$$

  - moves $\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})$ towards $r_t + \gamma\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})$
  - but also $\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1};\underline{\mathbf{w}})$ towards $\frac{1}{\gamma}\left(\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}}) - r_t\right)$
  - converges very slowly (Baird, 1995)

(see Sutton and Barto, 1998)

Gradients



$$E^T_{\text{BRM}} = \frac{1}{2p}\sum_{t=0}^{p-1}\Big(\overbrace{r_t + \gamma\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})}^{\text{target value for } \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})} - \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})\Big)^2$$

TD-error ΔQ_t

- two possible gradients for *online learning*
- **residual gradient** for online gradient descend

$$\frac{\partial(\Delta Q_t)^2}{\partial \underline{\mathbf{w}}} = -2\Delta Q_t\left(\frac{\partial\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}} - \gamma\frac{\partial\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}\right)$$

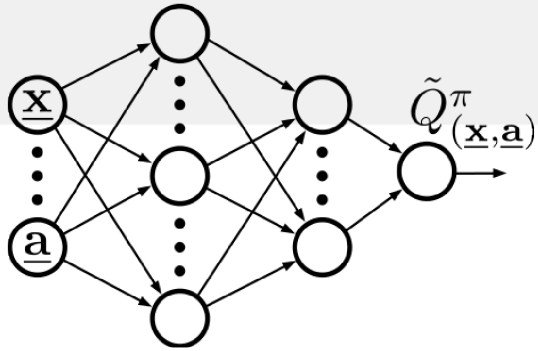
- **semi-gradient** approximates online gradient descend

$$\frac{\partial(\Delta Q_t)^2}{\partial \underline{\mathbf{w}}} \approx -2\Delta Q_t\frac{\partial\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}$$

  - assume target $r_t + \gamma\,\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}})$ is independent of $\underline{\mathbf{w}}$
  - converges faster than residual gradients (Gordon, 1995)

(see Sutton and Barto, 1998)

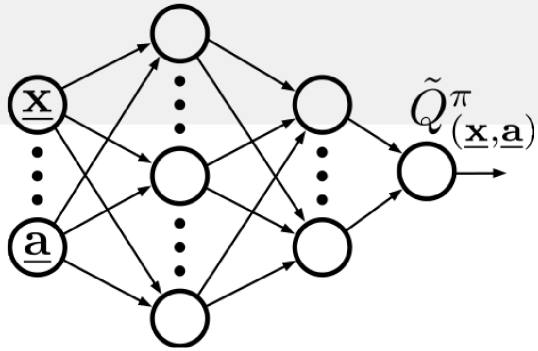
Semi-gradient descent



$$\underline{\mathbf{w}}^{(i+1)} = \underline{\mathbf{w}}^{(i)} + \eta \Delta Q_t \underbrace{\frac{\partial \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}}_{\text{backpropagation}}$$

- semi-gradient $\frac{\partial (\Delta Q_t)^2}{\partial \underline{\mathbf{w}}} \approx -2\Delta Q_t \frac{\partial \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}$
- online *on-policy* value estimation using semi-gradients *converges*
  - for basis function networks (Sutton and Barto, 1998)
  - but not always for non-linear functions (like MLP)
- online *off-policy* value estimation using semi-gradients may not
  - for all known model classes (Baird, 1995; Tsitsiklis and Van Roy, 1997)

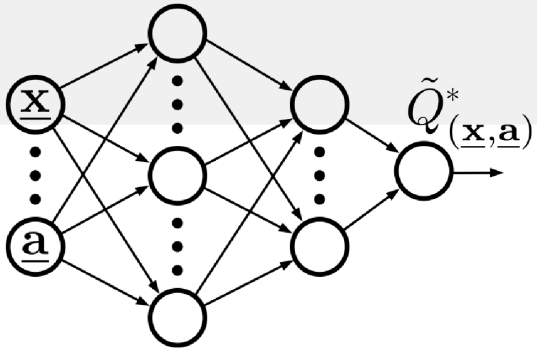
Gradient descend on Markov chains



- online gradient descend on Markov chains often fails:
  - gradient updates can influence value predictions for all states
  - the data is not drawn i.i.d. from a stationary distribution

- solution: **experience replay**:
  - sampling state-action pairs from long ergodic Markov chains is equivalent to sampling state-action pairs from the underlying stationary distribution (Section 4.1, p. 26)

Semi-gradient Q-learning



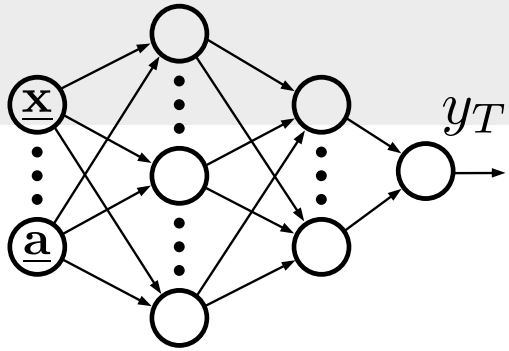
- model class: multi-layer perceptrons $\tilde{Q}^*(\underline{\mathbf{x}}, \underline{\mathbf{a}}; \underline{\mathbf{w}})$

$$E^T_{\text{BRM}} = \frac{1}{2p}\sum_{t=0}^{p-1}\Big(\overbrace{r_t + \gamma \max_{\underline{\mathbf{a}}' \in \mathcal{A}} \tilde{Q}^*(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}'; \underline{\mathbf{w}})}^{\text{target of } \tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})} - \tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})\Big)^2$$

TD-error ΔQ_t

- semi-gradient $\frac{\partial (\Delta Q_t)^2}{\partial \underline{\mathbf{w}}} = -2\Delta Q_t \underbrace{\frac{\partial \tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}})}{\partial \underline{\mathbf{w}}}}_{\text{backpropagation}}$
- gradient descend with i.i.d. samples from *experience replay* buffer
- however: semi-gradient Q-learning is *off-policy* and may not converge

Neural fitted Q-learning (NFQ)



- approximate Q-learning operator $\hat{B}^*$ with MLP
  - off-policy *batch learning* using *experience replay*
  - determine $\underline{\mathbf{w}}^{(i+1)}$ by regression of targets $y_T^{(t)}$

$$y_T^{(t)} := \hat{B}^*[\tilde{Q}^*(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i)})] = r_t + \gamma \max_{\underline{\mathbf{a}}' \in \mathcal{A}} \tilde{Q}^*(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}'; \underline{\mathbf{w}}^{(i)})$$

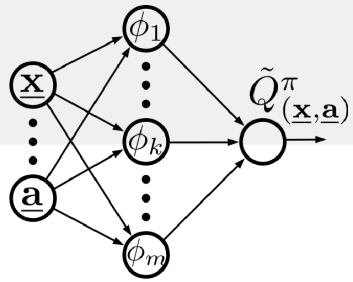
- any MLP regression algorithm can be used

NFQ algorithm (Riedmiller, 2005)

initialize $\underline{\mathbf{w}}^{(0)}$ randomly
while $\underline{\mathbf{w}}^{(i)}$ *not converged* **do**
  for $t \in \{1, \ldots, p-1\}$ **do**
    $y_T^{(t)} \leftarrow r_t + \gamma \max_{\underline{\mathbf{a}}' \in \mathcal{A}} \tilde{Q}^*(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}'; \underline{\mathbf{w}}^{(i)})$
  end
  $\underline{\mathbf{w}}^{(i+1)} \leftarrow \texttt{regression}\left(\left\{ \begin{bmatrix} \underline{\mathbf{x}}^{(t)} \\ \underline{\mathbf{a}}^{(t)} \end{bmatrix}, y_T^{(t)} \right\}_{t=0}^{p-1}\right)$
end

4.2.5 Least-Squares RL

Projected Bellman operator

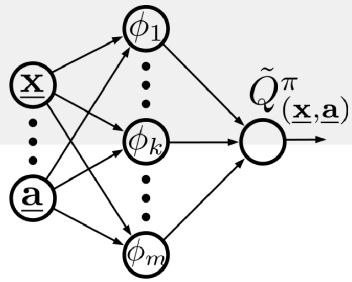


- linear model $\tilde{Q}^{\pi}(\underline{\mathbf{x}}, \underline{\mathbf{a}}; \underline{\mathbf{w}}) = \underline{\mathbf{w}}^{\top} \underline{\phi}(\underline{\mathbf{x}}, \underline{\mathbf{a}})$ with $\phi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
- approximate Bellman operator $B^{\pi}[\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i)})]$

$$\min_{\underline{\mathbf{w}}^{(i+1)}} E^{T}_{\text{BRM}} = \min_{\underline{\mathbf{w}}^{(i+1)}} \frac{1}{2p} \sum_{t=0}^{p-1} \Bigg(\overbrace{r_t + \gamma \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)}; \underline{\mathbf{w}}^{(i)})}^{\text{target } B^{\pi}[\tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i)})]} - \tilde{Q}^{\pi}(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i+1)}) \Bigg)^2$$

independent of $\underline{\mathbf{w}}^{(i+1)}$

Projected Bellman operator

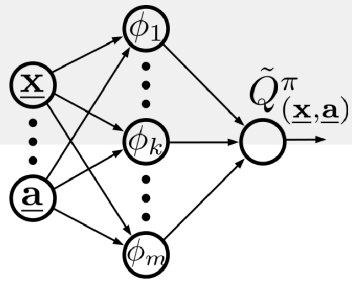


- linear model $\tilde{Q}^\pi(\underline{\mathbf{x}}, \underline{\mathbf{a}}; \underline{\mathbf{w}}) = \underline{\mathbf{w}}^\top \underline{\phi}(\underline{\mathbf{x}}, \underline{\mathbf{a}})$ with $\phi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
- approximate Bellman operator $B^\pi[\tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i)})]$

$$\min_{\underline{\mathbf{w}}^{(i+1)}} E^T_{\text{BRM}} = \min_{\underline{\mathbf{w}}^{(i+1)}} \frac{1}{2p} \sum_{t=0}^{p-1} \Bigg(\overbrace{\underbrace{r_t + \gamma \tilde{Q}^\pi(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)}; \underline{\mathbf{w}}^{(i)})}_{\text{independent of } \underline{\mathbf{w}}^{(i+1)}}}^{\text{target } B^\pi[\tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i)})]} - \tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}; \underline{\mathbf{w}}^{(i+1)}) \Bigg)^2$$

$$\frac{\partial E^T_{\text{BRM}}}{\partial \underline{\mathbf{w}}} = \overbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} \underline{\phi}^\top_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})}}^{\mathbf{C}} \underline{\mathbf{w}}^{(i+1)} - \overbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} r_t}^{\mathbf{b}}$$
$$- \gamma \underbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} \underline{\phi}^\top_{(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}^{(t+1)})}}_{\underline{\mathbf{D}}^\pi} \underline{\mathbf{w}}^{(i)}$$

Projected Bellman operator



- linear model $\tilde{Q}^\pi(\underline{\mathbf{x}},\underline{\mathbf{a}};\underline{\mathbf{w}}) = \underline{\mathbf{w}}^\top \underline{\phi}(\underline{\mathbf{x}},\underline{\mathbf{a}})$ with $\phi_i : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$
- approximate Bellman operator $B^\pi[\tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}}^{(i)})]$

$$\min_{\underline{\mathbf{w}}^{(i+1)}} E^T_{\text{BRM}} = \min_{\underline{\mathbf{w}}^{(i+1)}} \frac{1}{2p} \sum_{t=0}^{p-1} \Big(\overbrace{\underbrace{r_t + \gamma \tilde{Q}^\pi(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)};\underline{\mathbf{w}}^{(i)})}_{\text{independent of } \underline{\mathbf{w}}^{(i+1)}}}^{\text{target } B^\pi[\tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}}^{(i)})]} - \tilde{Q}^\pi(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)};\underline{\mathbf{w}}^{(i+1)}) \Big)^2$$

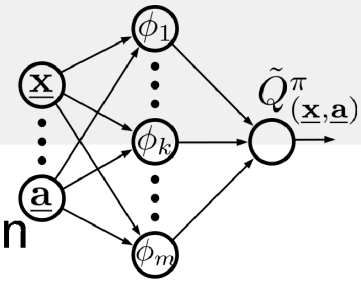
$$\frac{\partial E^T_{\text{BRM}}}{\partial \underline{\mathbf{w}}} = \overbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)})} \underline{\phi}^\top_{(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)})}}^{\mathbf{C}} \underline{\mathbf{w}}^{(i+1)} - \overbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)})} r_t}^{\mathbf{b}} - \gamma \underbrace{\sum_{t=1}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)},\underline{\mathbf{a}}^{(t)})} \underline{\phi}^\top_{(\underline{\mathbf{x}}^{(t+1)},\underline{\mathbf{a}}^{(t+1)})}}_{\underline{\mathbf{D}}^\pi} \underline{\mathbf{w}}^{(i)}$$

$$\underline{\mathbf{w}}^{(i+1)} = \underline{\mathbf{C}}^{-1}\big(\mathbf{b} + \gamma \underline{\mathbf{D}}^\pi \underline{\mathbf{w}}^{(i)}\big) =: \tilde{B}^\pi\big[\underline{\mathbf{w}}^{(i)}\big]$$

- **projected Bellman operator** $\tilde{B}^\pi$ is affine transformation of $\underline{\mathbf{w}}^{(i)}$

(see Bertsekas, 2007)

Least-squares temporal difference learning (LSTD)



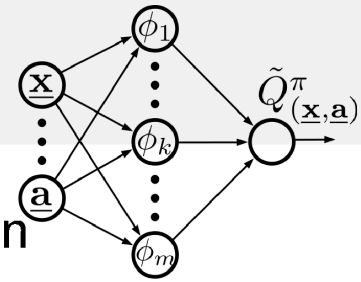
- fixed point $V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*) = \tilde{B}^{\pi}[V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*)]$ is the LSTD solution

$$\underline{\mathbf{w}}^* \quad = \quad \left(\underline{\mathbf{C}} - \gamma \underline{\mathbf{D}}^{\pi}\right)^{-1} \underline{\mathbf{b}}$$

(defined by Bradtke and Barto, 1996)

[2] in a L_2 norm weighted with the *steady state distribution* P_{ssd}

Least-squares temporal difference learning (LSTD)



- fixed point $V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*) = \tilde{B}^{\pi}[V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*)]$ is the LSTD solution

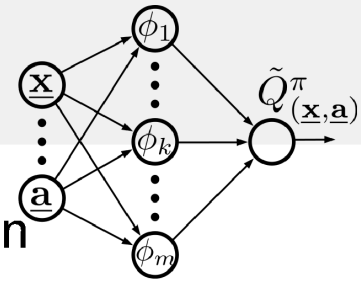
$$\underline{\mathbf{w}}^* = \left(\underline{\mathbf{C}} - \gamma \underline{\mathbf{D}}^{\pi}\right)^{-1} \underline{\mathbf{b}} = \left(\underline{\mathbf{I}} - \gamma \underbrace{\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}^{\pi}}_{\tilde{\mathbf{P}}^{\pi}}\right)^{-1} \underbrace{\underline{\mathbf{C}}^{-1}\underline{\mathbf{b}}}_{\tilde{\mathbf{r}}}$$

- LSTD solution is fix-point of least-squares models $\tilde{\mathbf{P}}^{\pi}$ and $\tilde{\mathbf{r}}$
  - discrete batch value estimation is special case of LSTD

(see Parr et al., 2008)

[2] in a L_2 norm weighted with the *steady state distribution* P_{ssd}

Least-squares temporal difference learning (LSTD)



- fixed point $V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*) = \tilde{B}^{\pi}[V(\underline{\mathbf{x}};\underline{\mathbf{w}}^*)]$ is the LSTD solution

$$\underline{\mathbf{w}}^* = \left(\underline{\mathbf{C}} - \gamma \underline{\mathbf{D}}^{\pi}\right)^{-1} \underline{\mathbf{b}} = \left(\underline{\mathbf{I}} - \gamma \underbrace{\underline{\mathbf{C}}^{-1}\underline{\mathbf{D}}^{\pi}}_{\tilde{\underline{\mathbf{P}}}^{\pi}}\right)^{-1} \underbrace{\underline{\mathbf{C}}^{-1}\underline{\mathbf{b}}}_{\tilde{\underline{\mathbf{r}}}}$$

- LSTD solution is fix-point of least-squares models $\tilde{\underline{\mathbf{P}}}^{\pi}$ and $\tilde{\underline{\mathbf{r}}}$
  - discrete batch value estimation is special case of LSTD

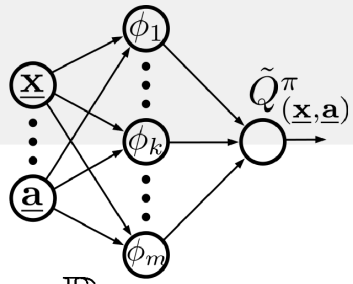
- on-policy LSTD finds a fixed-point close to V^{π}
  - $\tilde{B}^{\pi}$ is a contraction mapping and the approximation error is bounded[2]

$$\left\| V^{\pi} - \underline{\phi}^{\top}\underline{\mathbf{w}}^* \right\|_{P_{\text{ssd}}} \leq \frac{1}{\sqrt{1-\gamma^2}} \left\| V^{\pi} - \underbrace{\underline{\phi}^{\top}\mathbb{E}\left[\underline{\phi}\,\underline{\phi}^{\top}\right]\mathbb{E}\left[\underline{\phi}V^{\pi}\right]}_{\text{value projected on } \underline{\phi}} \right\|_{P_{\text{ssd}}}$$

(proven by Tsitsiklis and Van Roy, 1997)

[2] in a L_2 norm weighted with the *steady state distribution* P_{ssd}

Least-squares policy iteration (LSPI)



- policy iteration with LSTD as *off-policy* Q-value estimator
  - given training set $\{\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)}, r_t\}_{t=0}^{p}$ and bases $\phi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
  - greedy policy chooses action $\underline{\mathbf{a}}'^{(t+1)}$ for each sample $\underline{\mathbf{x}}^{(t+1)}$

LSPI algorithm

(see Lagoudakis and Parr, 2003)

initialize Q-value parameters $\underline{\mathbf{w}}$ randomly

$\underline{\mathbf{C}} := \sum_{t=0}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} \underline{\phi}^{\top}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})}$ and $\underline{\mathbf{b}} := \sum_{t=0}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} r_t$ // constant

while *Q-values not converged* **do**

  for $t \in \{0, \ldots, p-1\}$ **do**

    $\underline{\mathbf{a}}'^{(t+1)} := \underset{\underline{\mathbf{a}} \in \mathcal{A}}{\operatorname{argmax}} \{\underline{\mathbf{w}}^{\top} \underline{\phi}_{(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}})}\}$ // policy improvement

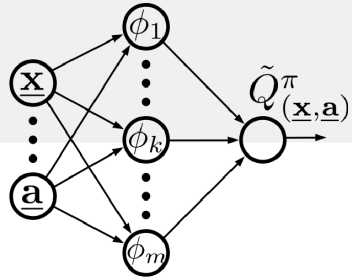
  end

  $\underline{\mathbf{D}}^{\pi} := \sum_{t=0}^{p-1} \underline{\phi}_{(\underline{\mathbf{x}}^{(t)}, \underline{\mathbf{a}}^{(t)})} \underline{\phi}^{\top}_{(\underline{\mathbf{x}}^{(t+1)}, \underline{\mathbf{a}}'^{(t+1)})}$

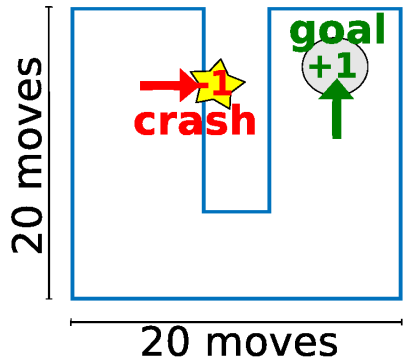
  $\underline{\mathbf{w}} := \left(\underline{\mathbf{C}} - \gamma \underline{\mathbf{D}}^{\pi}\right)^{-1} \underline{\mathbf{b}}$ // value estimation

end

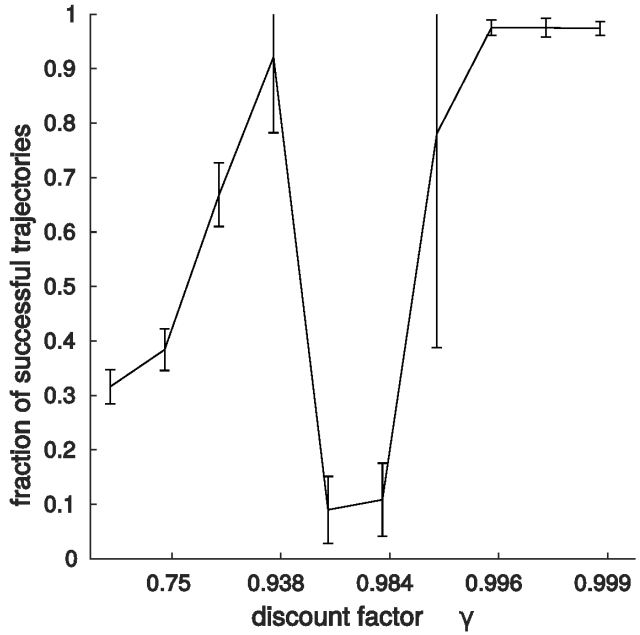
An example for LSPI



- no convergence guarantee for *off-policy* LSTD
- dynamics can become instable after policy improvement
  - example: navigation in a U-shaped room

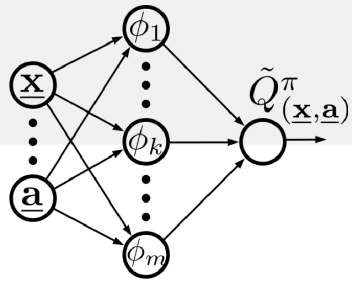


- 3 continuous state dimensions: x/y position, orientation
- 3 discrete actions: move forward, turn left/right
- rewards: +1 in goal area, -1 for crashes
- 1500 Fourier state-action bases
- validated on trajectories from greedy policy



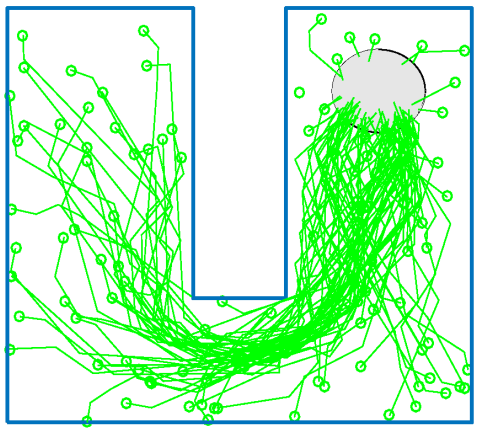
(see Böhmer et al., 2016, for details)

An example for LSPI

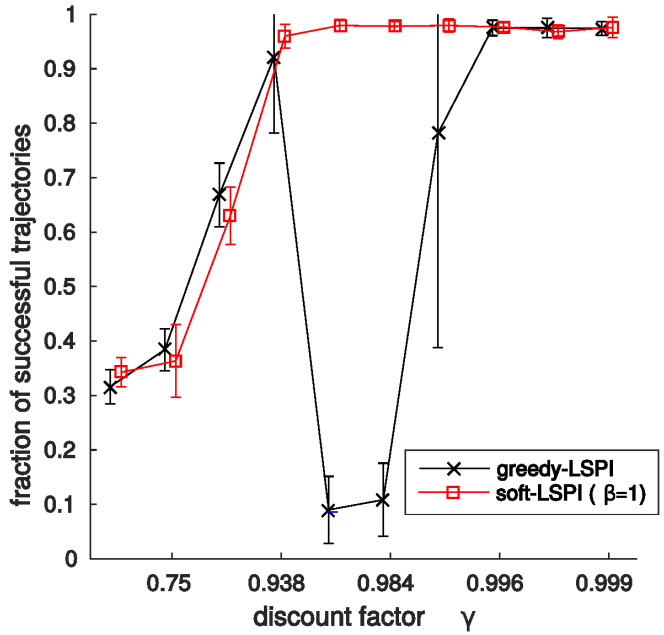


- no convergence guarantee for *off-policy* LSTD
- dynamics can become instable after policy improvement
  - example: navigation in a U-shaped room
- dynamics can often be stabilized to allow convergence
  - e.g. by using a softmax instead of a greedy policy improvement

near optimal greedy policies



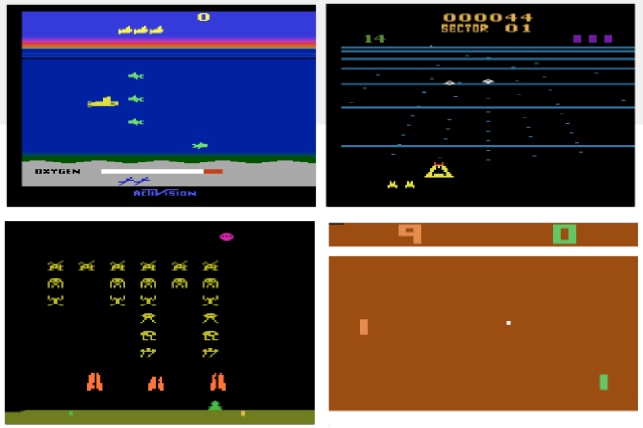
learned for $\gamma > 0.975$



(see Böhmer et al., 2016, for details)

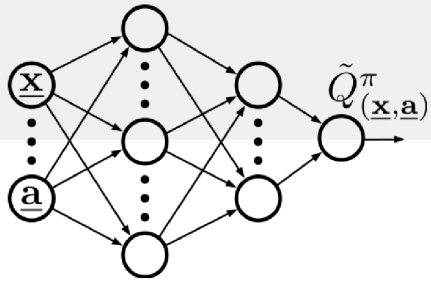
4.2.6 End-to-End RL

End-to-end reinforcement learning



- real world observations are often high-dimensional
  - e.g. images of ATARI games (Mnih et al., 2015)
- autonomous agents learn direct mapping from sensors to actors
  - sensory images as the state of the system
- non-Markovian observations must be *temporally embedded*
  - the current state is a combination of the last n observations
  - alternatively one can include a *recurrent* neural network
- a deep convolutional neural network estimates Q-values (DQN)
  - or learns the policy directly (Levine et al., 2016)
  - or learns a *deep actor-critic* (A3C, Mnih et al., 2016)

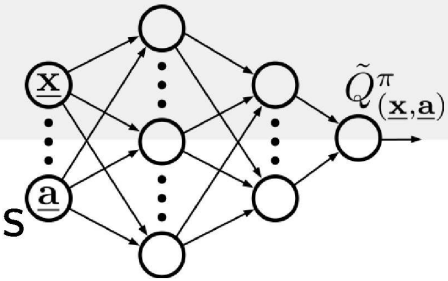
Deep Q-learning (DQN)



- *deep MLP* architectures with *rectified linear* transfer functions
  - the first layers are *convolutional*, later *fully connected*
  - image preprocessing and temporal embedding (of ~ 4 observations)
- long training sequences ($\sim 50.000.000$ samples)
  - ϵ-*greedy* exploration (annealed for $\sim 1.000.000$ samples, then $\epsilon \sim 0.1$)
- *regression* of Q-learning operator $\hat{B}^*$ (like NFQ, p. 33)
  - target Q-value function $Q(\cdot, \cdot; \underline{\mathbf{w}}^{(i)})$ changes ~ 10.000 updates
  - updates use average gradient from *mini-batches* (~ 32 samples), drawn i.i.d. from *experience replay buffer* ($\sim 1.000.000$ samples)
  - uses `RMSProp` with gradient momentum and adjusting learning rate

(Mnih et al., 2015)

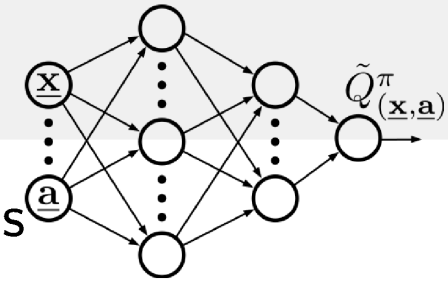
Example: DQN plays ATARI games



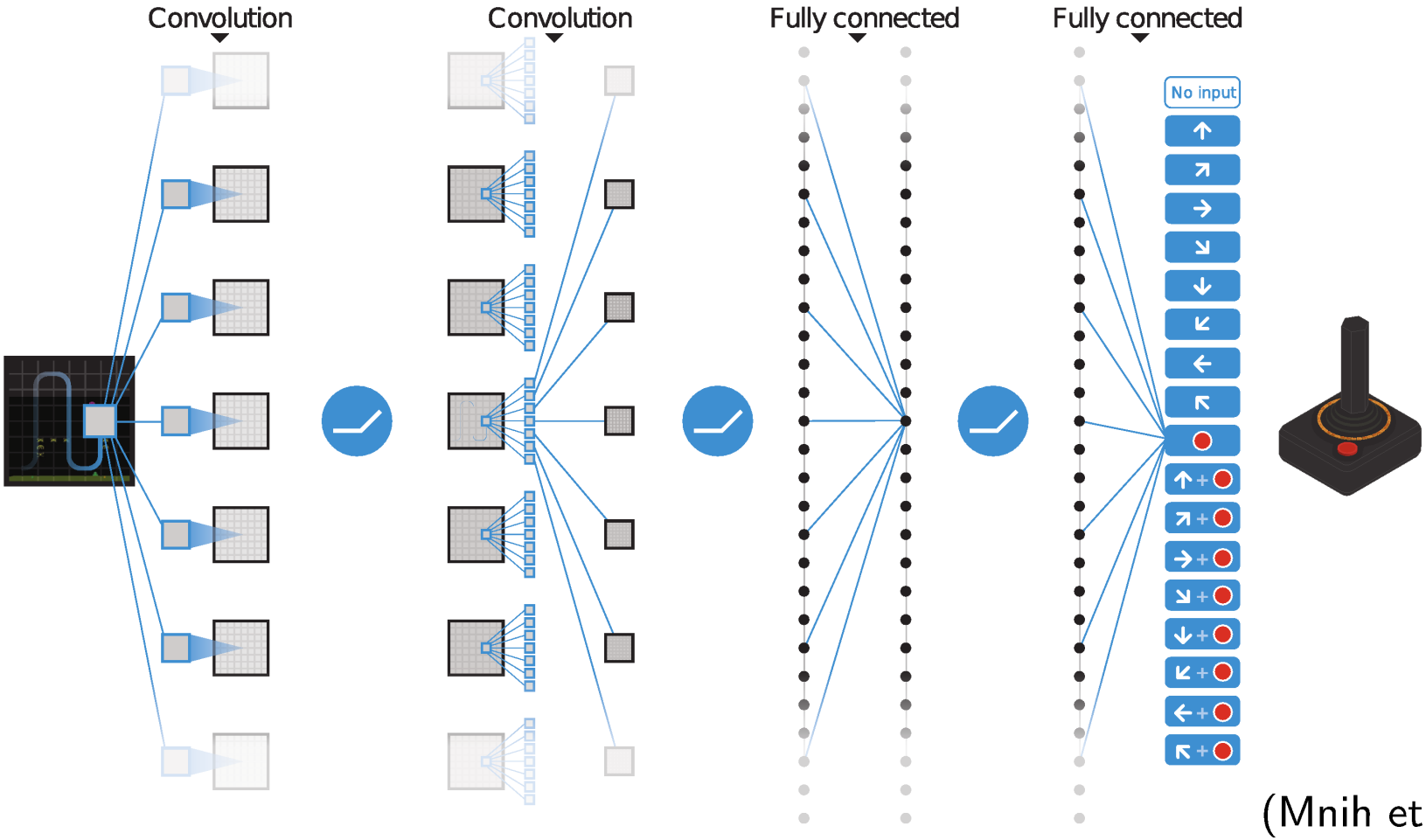
- mapping game-images "end-to-end" to controller-actions
  - challenging task for human players

(Mnih et al., 2015)

Example: DQN plays ATARI games

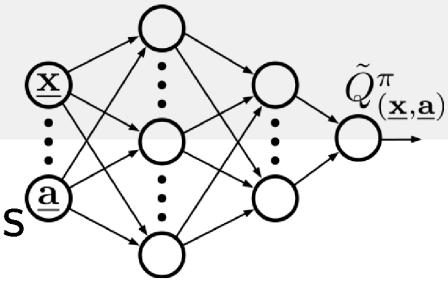


- mapping game-images "end-to-end" to controller-actions
  - MLP with 3 convolutional, and 2 fully connected layers

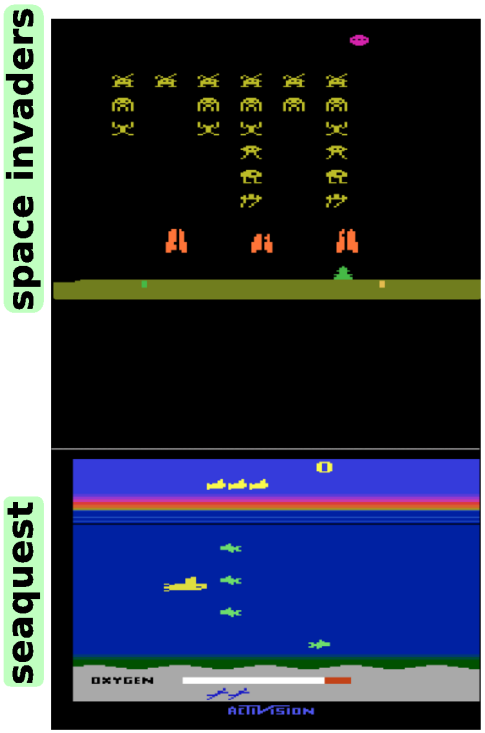


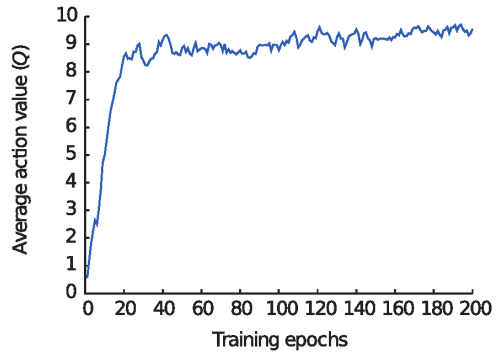
(Mnih et al., 2015)

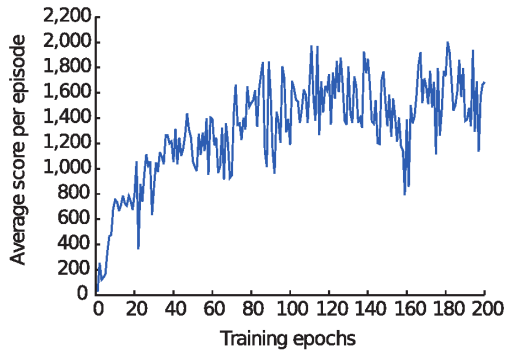
Example: DQN plays ATARI games

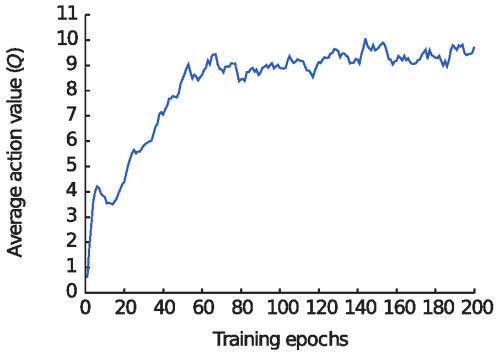


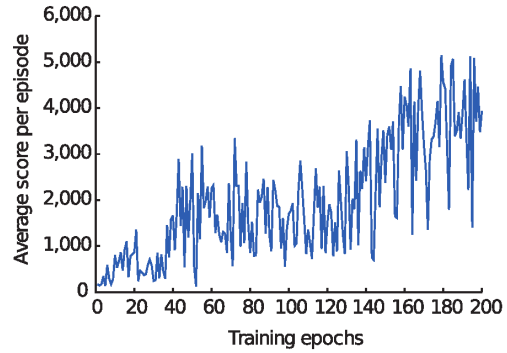
- mapping game-images "end-to-end" to controller-actions
- network learns policy that plays the game well





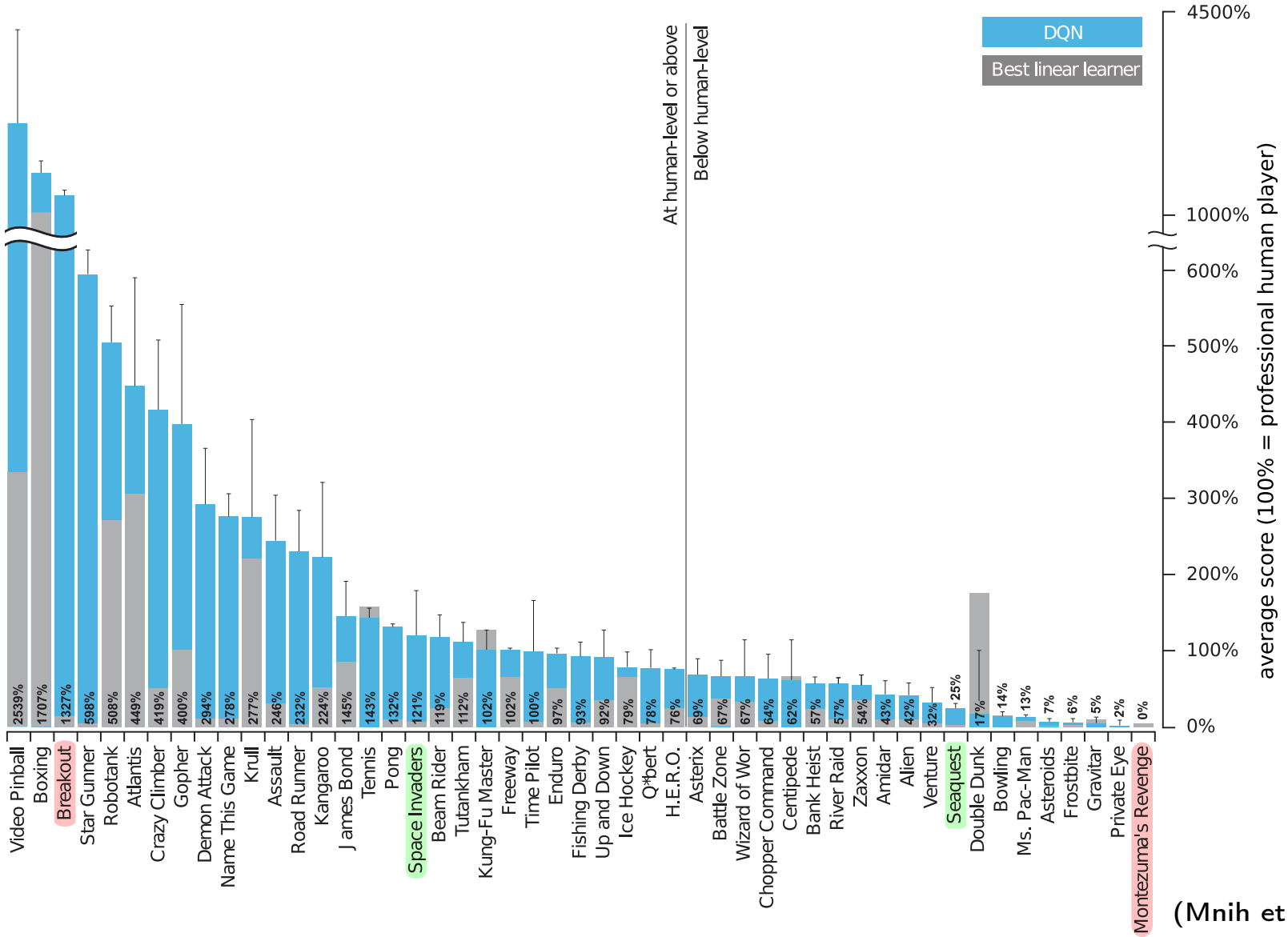






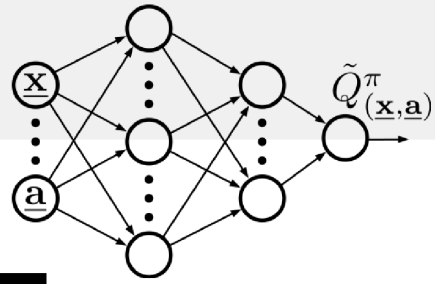
(Mnih et al., 2015)

Analysis: limits of end-to-end learning (1)



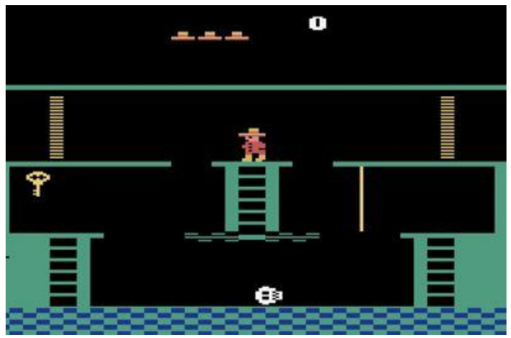
(Mnih et al., 2015)

Analysis: limits of end-to-end learning (2)



Breakout

VS.

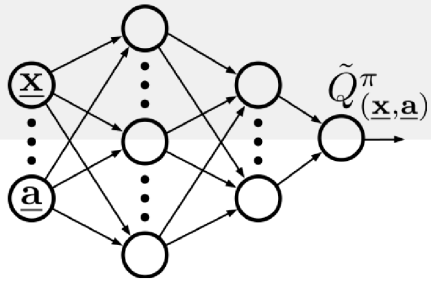


Montezuma's Revenge

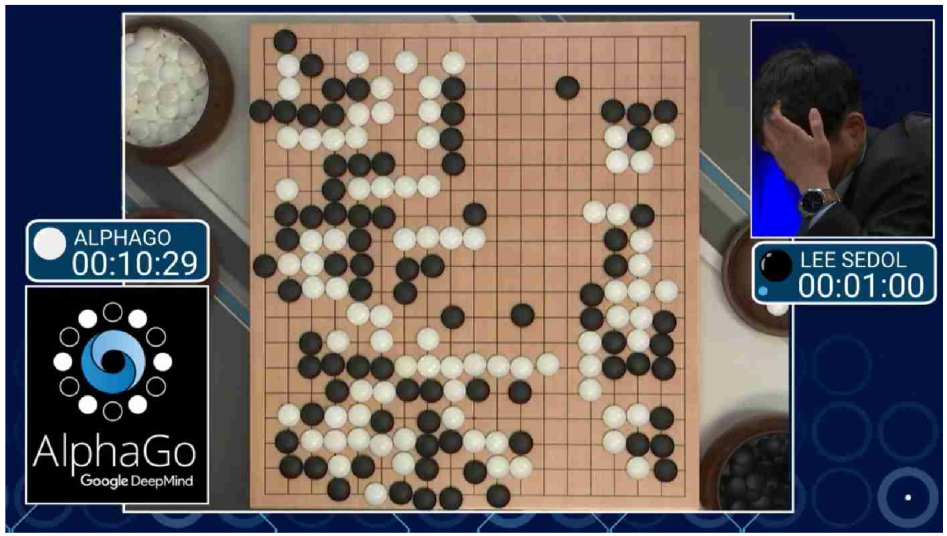
- 1327% of human performance
- **objective** is pixel encoded
  - clear all blocks/pixels
- **winning strategy** reacts well
  - bring the platform to the ball

- 0% of human performance
- **objective** is goal-oriented
  - reach key, ...
- **winning strategy** thinks ahead
  1. first go to the rope
  2. then jump over enemy
  3. then climb latter

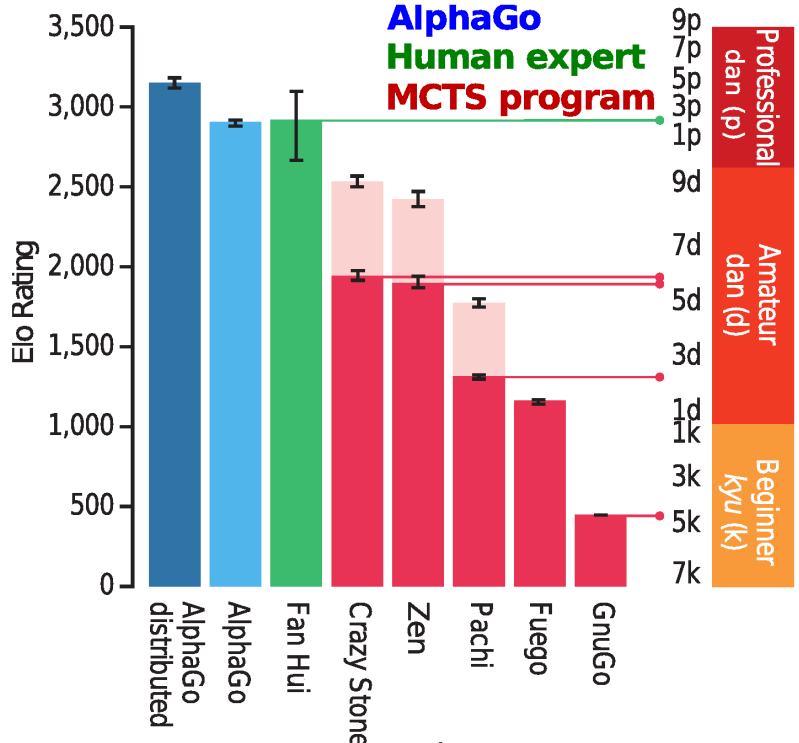
State of the art reinforcement learning



- deep end-to-end RL fails if agent must plan ahead
- AlphaGo combines deep RL with *Monte Carlo tree search* (MCTS)
  - MCTS plans ahead of current position
  - deep network generalizes values



AlphaGo beat world champion Lee Sedol in March 2016



(Silver et al., 2016)

End of Section 4.2

the following slides contain

OPTIONAL MATERIAL

References I

Peter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems*, pages 1–8, 2007.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3rd edition, 2007.

Wendelin Böhmer, Rong Guo, and Klaus Obermayer. Non-deterministic policy improvement stabilizes approximate reinforcement learning. In *European Workshop of Reinforcement Learning (EWRL)*. 2016. URL https://ewrl.files.wordpress.com/2016/11/ewrl13-2016-submission_2.pdf.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1/2/3):33–57, 1996.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX- a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

G. Gordon. Stable function approximation in dynamic programming. In *International Conference on Machine Learning*, 1995.

References II

Micheal Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 1928–1937, 2016. URL http://jmlr.org/proceedings/papers/v48/mniha16.html.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefiled, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*, 2008.

References III

Martin Riedmiller. Neural fitted Q-iteration - first experiences with a data efficient neural reinforcement learning method. In *16th European Conference on Machine Learning*, pages 317–328. Springer, 2005.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, January 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.