Reinforcement Learning Algorithms in Markov Decision Processes
AAAI-10 Tutorial

Part III: Learning to control

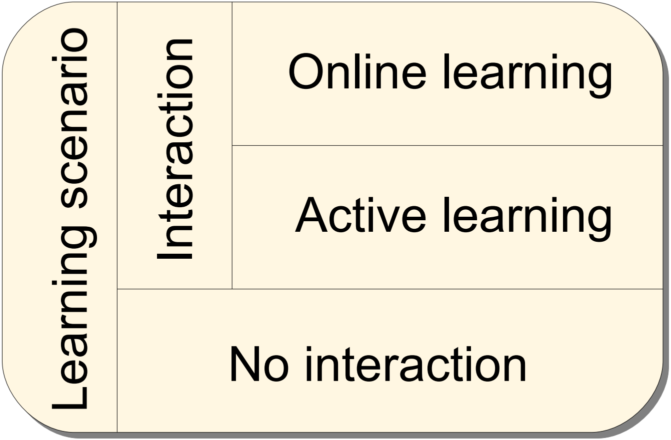
Csaba Szepesvári   Richard S. Sutton

University of Alberta
E-mails: {**szepesva,rsutton**}**@.ualberta.ca**

Atlanta, July 11, 2010

Outline

The landscape



Bandit problems: How to gamble if you must? Part II

Bandit problem

- MDP with single state
- Unknown distribution of rewards
- Which action to choose so as to minimize the regret,

$$L_T = T \max_{a \in \mathcal{A}} r(a) - \sum_{t=1}^{T} R_t.$$

- Lai and Robbins (1985): optimism in the face of uncertainty (OFU) principle:

  > *Choose the action with the best potential where the uncertainty of the available information is taken into account*

- They "solved" the parametric case: $\log$ regret, matching upper and lower bounds

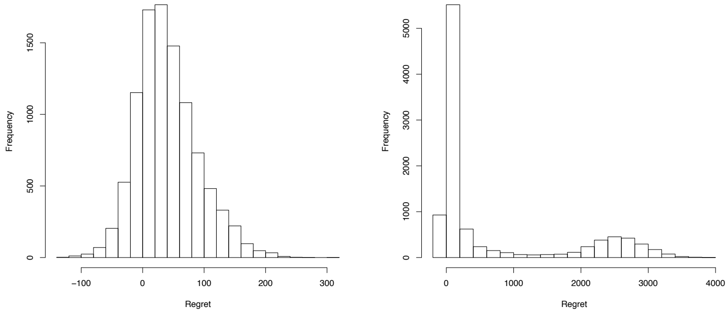
Bandit problems: Nonparametrics

- Auer et al. (2002): When the distributions can be arbitrary ($R_t \in [0, 1]$), play the action maximizing

$$U_t(a) = r_t(a) + \mathcal{R}\sqrt{\frac{2\log t}{n_t(a)}}.$$

- Upper Confidence Bound: UCB $\Rightarrow$ UCB1 algorithm
- Main result: $L_T = O(\log(T))$
- The minimax regret is $O(\sqrt{T})$.
- By estimating the variance the expected regret can be improved, but there is a bias-variance tradeoff

Beware the risk!



Distribution of the regret for UCB-V at times $T_1 = 16,384$ (l.h.s. figure) and $T_2 = 524,288$ (r.h.s. figure) on a two-armed bandit, where the payoff of the optimal arm is Ber(0.5), and the payoff of the suboptimal arm is 0.495.

Online learning: Epilogue

- Bayesian bandits
  - The issue is not conceptual, but computational
  - Gittins (1989): "Gittins index" (cheap computation)
  - The Bayesian setting applies e.g. in poker (we know the distribution of cards)
- Active learning in bandits
  - "Action elimination"
  - These algorithms are unimprovable (Even-Dar et al., 2002; Tsitsiklis and Mannor, 2004; Mnih et al., 2008).
- Online learning in MDPs
  - UCRL2 by Auer et al. (2010) implements the OFU principle
  - Individual rate: $O(\log T)$, minimax: $O(\sqrt{T})$
- PAC-MDP algorithms
  - "Mistake bounds"
  - R-MAX, MBIE, OI, MorMax, Delayed-Q, ..
  - (Kearns and Singh, 1998; Brafman and Tennenholtz, 2002; Kakade, 2003; Strehl and Littman, 2005; Strehl et al., 2006; Szita and Lőrincz, 2008; Szita and Szepesvári, 2010)

Goal

Idea/Goal

Learn Q^* directly.

Q-learning in finite MDPs

- Bellman equation for the action-value function of a policy π:

$$Q^\pi(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} \mathcal{P}(x,a,y) \sum_{a'\in\mathcal{A}} \pi(a'|y) Q^\pi(y,a').$$

- TD-learning for the action-value function of π:

$$Q(X,A) \leftarrow Q(X,A) + \alpha \left\{ R + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|Y) Q(Y,a') - Q(X,A) \right\}$$

- Bellman optimality equation for Q^*:

$$Q^*(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} \mathcal{P}(x,a,y) \max_{a'\in\mathcal{A}} Q^*(y,a'), \quad x \in \mathcal{X}, a \in \mathcal{A}.$$

(or, in short, $Q^* = T^* Q^*$).

- Watkins (1989) Q-learning algorithm:

$$Q(X,A) \leftarrow Q(X,A) + \alpha \left\{ R + \gamma \max_{a'\in\mathcal{A}} Q(Y,a') - Q(X,A) \right\}$$

Q-learning in finite MDPs

function QLEARNING(X, A, R, Y, Q)
Input: X is the last state, A is the last action, R is the immediate reward received, Y is the next state, Q is the array storing the current action-value function estimate

1: $\delta \leftarrow R + \gamma \cdot \max_{a' \in \mathcal{A}} Q[Y, a'] - Q[X, A]$
2: $Q[X, A] \leftarrow Q[X, A] + \alpha \cdot \delta$
3: **return** Q

Theorem (Watkins and Dayan 1992; Tsitsiklis 1994; Jaakkola et al. 1994)

Consider a finite MDP. If all state-action pairs are visited infinitely often and "appropriate" local learning rates are used then the sequence of iterates $(Q_t; t \geq 0)$ computed with Q-learning converges to Q^* w.p.1.

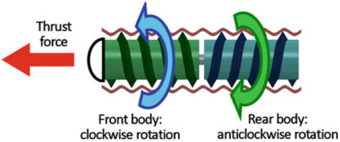
Q-learning with linear function approximation

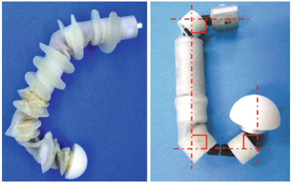
function QLEARNINGLINFAPP(X, A, R, Y, θ)

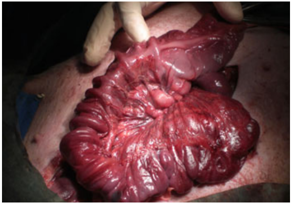
Input: X is the last state, Y is the next state, R is the immediate reward associated with this transition, $\theta \in \mathbb{R}^d$ parameter vector

1: $\delta \leftarrow R + \gamma \cdot \max_{a' \in \mathcal{A}} \theta^\top \varphi[Y, a'] - \theta^\top \varphi[X, A]$
2: $\theta \leftarrow \theta + \alpha \cdot \delta \cdot \varphi[X, A]$
3: **return** θ

Application: colon endoscope robot (Ukawa et al., 2010)

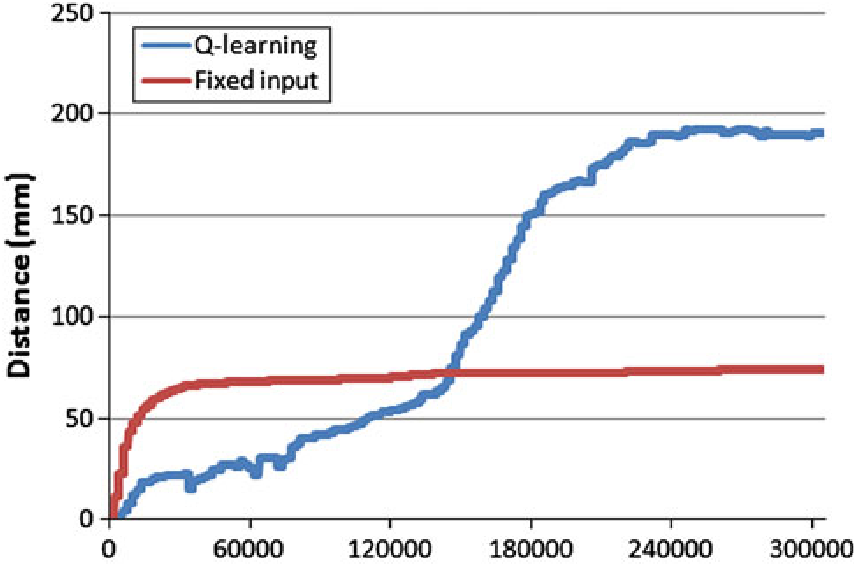






- State-discretization: torque (9), movement (5)
- Actions: voltage discretized to 5 levels
- Reward: 1 upon reaching waypoints (almost)
- Using ε-greedy with adaptive ε

Results



Q-learning
Fixed input
Distance (mm)
250
200
150
100
50
0
0
60000
120000
180000
240000
300000

Fitted Q-iteration

function FITTEDQ(D, θ)

Input: $D = ((X_i, A_i, R_{i+1}, Y_{i+1}); i = 1, \ldots, n)$ is a list of transitions, θ are the regressor parameters

1: $S \leftarrow []$ ▷ Create empty list
2: **for** $i = 1 \to n$ **do**
3: $T \leftarrow R_{i+1} + \max_{a' \in \mathcal{A}}$ PREDICT$((Y_{i+1}, a'), \theta)$ ▷ Target at (X_i, A_i)
4: $S \leftarrow$ APPEND$(S, \langle (X_i, A_i), T \rangle)$
5: **end for**
6: $\theta \leftarrow$ REGRESS(S)
7: **return** θ

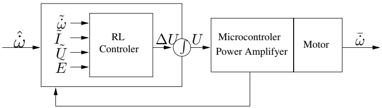
Caveat

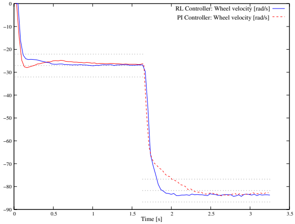
The algorithm might diverge/become unstable To prevent this

- one might use a special regressor ("averager")
- one could use a powerful regressor such that $\sup_{Q \in \mathcal{F}} \|\Pi_{\mathcal{F}} T^* Q - T^* Q\|$ is small

Application: Controlling the speed of a DC motor

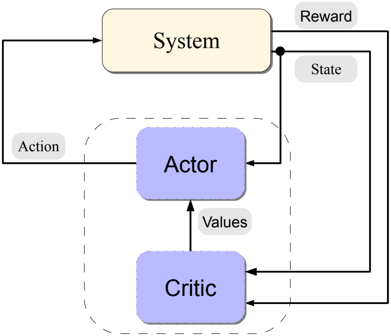
(Hafner and Riedmiller, 2007)





- Goal is to track a reference signal $\dot{\omega}_r = \dot{\omega}_r(t)$
- Inputs:
  - I – armature current
  - $\dot{\omega}$ – current motor speed
  - U – actual voltage
  - $E = \dot{\omega}_r - \dot{\omega}$ – tracking error
- Action: $\Delta U \in \{-0.3, -0.1, -0.01, 0.0, 0.01, 0.1, 0.3\}$
- Reward: -1 if E is big
- Less than 5 minutes of data is needed, $\Delta t = 33\text{ms}$

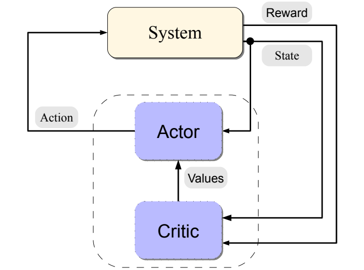
The actor-critic architecture



The actor-critic architecture

Implementation choices

- Critic:
  - Action-value functions or value functions?
  - What method?
- Actor:
  - With function approximation
    - What method?
  - Without function approximation
- How to explore?



SARSA(λ) with linear function approximation

function SARSALAMBDALINFAPP($X, A, R, Y, A', \theta, z$)

Input: X is the last state, A is the last action chosen, R is the immediate reward received when transitioning to Y, where action A' is chosen. $\theta \in \mathbb{R}^d$ is the parameter vector of the linear function approximation, $z \in \mathbb{R}^d$ is the vector of eligibility traces

$\delta \leftarrow R + \gamma \cdot \theta^\top \varphi[Y, A'] - \theta^\top \varphi[X, A]$

$z \leftarrow \varphi[X, A] + \gamma \cdot \lambda \cdot z$

$\theta \leftarrow \theta + \alpha \cdot \delta \cdot z$

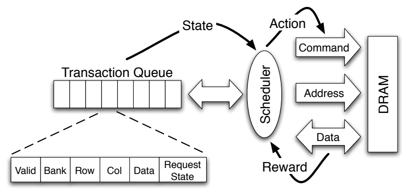
return (θ, z)

SARSA $\equiv$
current **S**tate, current **A**ction, next **R**eward, next **S**tate, and next **A**ction
(Rummery and Niranjan, 1994; Rummery, 1995)

Application: DRAM command scheduling

Problem (Ipek et al., 2008)

- Goal: Optimize DRAM command scheduling policy to optimize performance
- Tool: SARSA(0) with CMAC (tile coding)
- Observations: Transaction queue
- Actions: Candidate scheduling commands
- Reward: 1 for read/write, 0 for others (e.g. precharge, activate,..)



Application: DRAM command scheduling

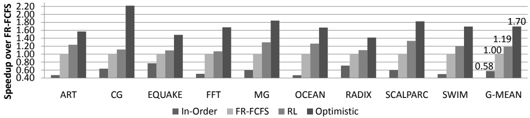
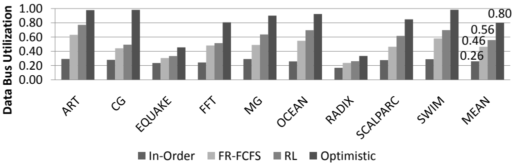
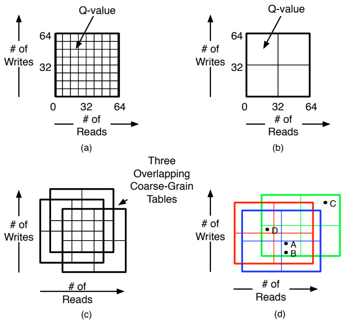


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers





Policy gradient

- Fix $\Pi = (\pi_\omega; \omega \in \mathbb{R}^{d_\omega})$
- Goal:

$$\operatorname*{argmax}_{\omega} \rho_\omega = ?$$

- Choices for ρ_ω:
  - $\rho_\omega = \mathbb{E}\left[V^{\pi_\omega}(X_0)\right]$, $X_0 \sim \mu$
  - When μ is the stationary distribution of π ($\mu = \mu_\pi$), the two performance measures become the same, apart from a constant factor

Policy gradient theorem

Assumption

The Markov chain resulting from following any policy π_ω is ergodic, regardless of the choice of ω.

- How to estimate the gradient of ρ_ω?
- Let $\psi_\omega : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^{d_\omega}$ be the score function underlying π_ω:

$$\psi_\omega(x,a) = \frac{\partial}{\partial \omega} \log \pi_\omega(a|x), \qquad (x,a) \in \mathcal{X} \times \mathcal{A}.$$

- Define

$$G(\omega) = \left(Q^{\pi_\omega}(X,A) - h(X)\right) \, \psi_\omega(X,A),$$

  where $(X,A) \sim \mu_{\pi_\omega}$.
- Let Q^{π_ω} be the action-value function of π_ω and h is an arbitrary bounded function.

Policy gradient theorem II

Theorem (Policy gradient theorem)

$$\nabla_\omega \rho_\omega = \mathbb{E}\left[G(\omega)\right].$$

Corollary

Let $(X_t, A_t) \sim \mu_{\pi_{\omega_t}}$, *and assume*

$$\mathbb{E}\left[\hat{Q}_t(X_t, A_t)\psi_{\omega_t}(X_t, A_t)\right] = \mathbb{E}\left[Q^{\pi_{\omega_t}}(X, A)\psi_{\omega_t}(X_t, A_t)\right]. \tag{Q-PG}$$

Then

$$\omega_{t+1} = \omega_t + \beta_t \left(\hat{Q}_t(X_t, A_t) - h(X_t)\right) \psi_\omega(X_t, A_t) \tag{1}$$

implements stochastic gradient ascent.

Compatible function approximation

Compatible function approximation

Choose the feature-extraction function to be the score function underlying the policy class:

$$Q_\theta(x,a) = \theta^\top \psi_\omega(x,a), \qquad (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Note

The basis functions change when ω changes!

Theorem

Let $\theta_*(\omega) = \mathrm{argmin}_\theta \, \mathbb{E}\left[(Q_\theta(X,A) - Q^{\pi_\omega}(X,A))^2\right]$*. Then* $Q_{\theta_*(\omega)}$ *satisfies* (Q-PG) *and*

$$\omega_{t+1} = \omega_t + \beta_t \left(\hat{Q}_{\theta_*(\omega_t)}(X_t,A_t) - h(X_t)\right) \psi_\omega(X_t,A_t)$$

implements stochastic gradient ascent.

Actor-critic with SARSA(1)

function SARSAACTORCRITIC(X)
Input: X is the current state

$\omega, \theta, z \leftarrow 0$
$A \leftarrow a_1$ ▷ Pick any action
repeat
    $(R, Y) \leftarrow \text{EXECUTEINWORLD}(A)$
    $A' \leftarrow \text{DRAW}(\pi_\omega(Y, \cdot))$
    $(\theta, z) \leftarrow \text{SARSALAMBDALINFAPP}(X, A, R, Y, A', \theta, z)$
    ▷ Use $\lambda = 1$ and $\alpha \gg \beta$
    $\psi \leftarrow \frac{\partial}{\partial \omega} \log \pi_\omega(X, A)$
    $v \leftarrow \text{SUM}(\pi_\omega(Y, \cdot) \cdot \theta^\top \varphi[X, \cdot])$
    $\omega \leftarrow \omega + \beta \cdot \left(\theta^\top \varphi[X, A] - v\right) \cdot \psi$
    $X \leftarrow Y$
    $A \leftarrow A'$
until `True`

Natural actor-critic

function NAC(X)

Input: X is the current state

$\omega, \theta, z \leftarrow 0$, $A \leftarrow a_1$ ▷ Pick any action

repeat

    $(R, Y) \leftarrow$ EXECUTEINWORLD(A)

    $A' \leftarrow$ DRAW($\pi_\omega(Y, \cdot)$)

    $(\theta, z) \leftarrow$ SARSALAMBDALINFAPP($X, A, R, Y, A', \theta, z$)

▷ Use $\lambda = 1$ and $\alpha \gg \beta$

    $\psi \leftarrow \frac{\partial}{\partial \omega} \log \pi_\omega(X, A)$

    $v \leftarrow$ SUM($\pi_\omega(Y, \cdot) \cdot \theta^\top \varphi[X, \cdot]$)

    $\omega \leftarrow \omega + \beta \cdot \theta$

    $X \leftarrow Y$

    $A \leftarrow A'$

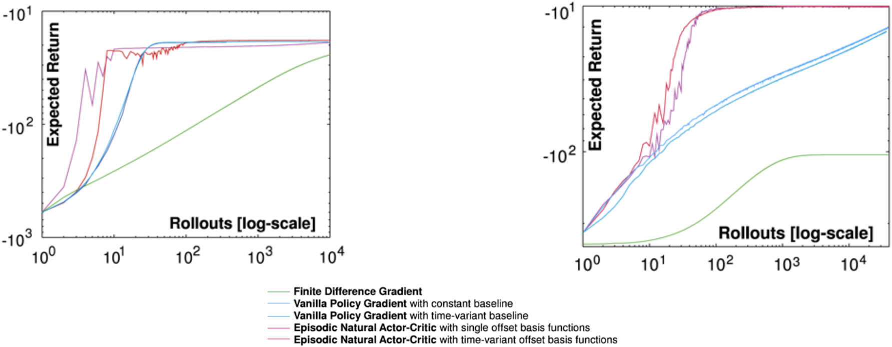
until `True`

Natural actor-critic

- We have $\theta_*(\omega) = F_\omega^{-1} \nabla_\omega \rho_\omega$ for a suitable F_ω.

  $\Longrightarrow$ this is a stochastic pseudo-gradient algorithm
- Better: This algorithm follows a (more) natural gradient
  - Gradient in the space of policies: Avoiding plateaus
  - Covariant trajectories (insensensitive to reparameterizing π_ω)

Learning motor primitives with NAC – Toy problems



Performance on problems (a) minimum motor command learning and (b) passing through a point.

Source: (Peters and Schaal, 2008)

Learning motor primitives with NAC

(a) Performance. (b) Imitation learning. (c) Initial reproduction. (d) After reinforcement learning.

Source: (Peters and Schaal, 2008)

For Further Reading I

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Auer, P., Jaksch, T., and Ortner, R. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563—1600.

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

Cohen, W. W., McCallum, A., and Roweis, S. T., editors (2008). *ICML 2008*. ACM.

Even-Dar, E., Mannor, S., and Mansour, Y. (2002). PAC bounds for multi-armed bandit and Markov decision processes. In Kivinen, J. and Sloan, R. H., editors, *COLT 2002*, pages 255–270. Springer.

Gittins, J. C. (1989). *Multi-armed Bandit Allocation Indices*. Wiley-Interscience series in systems and optimization. Wiley, Chichester, NY.

Hafner, R. and Riedmiller, M. (2007). Neural reinforcement learning controllers for a real robot application. In *ADPRL*, pages 2098–2103.

Ipek, E., Mutlu, O., Martinéz, J., and Caruana, R. (2008). Self-optimizing memory controllers: A reinforcement learning approach. In *Intl. Symp. on Computer Architecture (ISCA)*, Beijing, China.

Jaakkola, T., Jordan, M., and Singh, S. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201.

Kakade, S. (2003). *On the sample complexity of reinforcement learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London.

Kearns, M. and Singh, S. P. (1998). Near-optimal performance for reinforcement learning in polynomial time. In Shavlik, J. W., editor, *ICML 1998*, pages 260–268. Morgan Kauffmann.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22.

Mnih, V., Szepesvári, C., and Audibert, J.-Y. (2008). Empirical Bernstein stopping. In Cohen et al. (2008), pages 672–679.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks 21 (2008)*, 21:682—697.

Rummery, G. A. (1995). *Problem solving with reinforcement learning*. PhD thesis, Cambridge University.

For Further Reading II

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. (2006). PAC model-free reinforcement learning. In Cohen, W. W. and Moore, A., editors, *ICML 2006*, pages 881–888. ACM.

Strehl, A. L. and Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. In De Raedt, L. and Wrobel, S., editors, *ICML 2005*, pages 857–864. ACM.

Szita, I. and Lőrincz, A. (2008). The many faces of optimism: a unifying approach. In Cohen et al. (2008), pages 1048–1055.

Szita, I. and Szepesvári, C. (2010). Model-based reinforcement learning with nearly tight exploration complexity bounds. In *ICML 2010*.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202.

Tsitsiklis, J. N. and Mannor, S. (2004). The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648.

Ukawa, G. T. . M. S. . G., Kinoshita, J., Murai, N., Lee, J. W., Ishii, H., Takanishi, A., Tanoue, K., Ieiri, S., Konishi, K., and Hashizume, M. (2010). Development of a colon endoscope robot that adjusts its locomotion through the use of reinforcement learning. *Int J CARS*, 5:317–325.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 3(8):279–292.

DONE!