Reinforcement Learning Algorithms in Markov Decision Processes
AAAI-10 Tutorial

Part II: Learning to predict values

Csaba Szepesvári   Richard S. Sutton

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

Atlanta, July 11, 2010

Outline

1. Introduction
2. Simple learning techniques
   - Learning the mean
   - Learning values with Monte-Carlo
   - Learning values with temporal differences
   - Monte-Carlo or TD?
   - Resolution
3. Function approximation
4. Methods
   - Stochastic gradient descent
   - TD(λ) with linear function approximation
   - Gradient temporal difference learning
   - LSTD and friends
   - Comparing least-squares and TD-like methods
5. How to choose the function approximation method?
6. Bibliography

The problem

How to learn the value function of a policy over a large state space?

Why learn?

- Avoid the “curses of modeling”
  - Complex models are hard to deal with
  - Avoids modelling errors
  - Adaptation to changes

Why learn value functions?

- Applications:
  - Failure probabilities in a large power grid
  - Taxi-out times of flights on airports
  - Generally: Estimating a long term expected value associated with a Markov process
- Building block

Learning the mean of a distribution

- Assume $R_1, R_2, \ldots$ are i.i.d., $\exists V = \mathbb{E}[R_t]$.
- Estimating the expected value by the sample mean:

$$V_t = \frac{1}{t}\sum_{s=0}^{t-1} R_{s+1}.$$

- Recursive update:

$$V_t = V_{t-1} + \frac{1}{t}(\underbrace{R_t}_{\text{“target”}} - V_{t-1}).$$

- More general update:

$$V_t = V_{t-1} + \alpha_t (R_t - V_{t-1}),$$

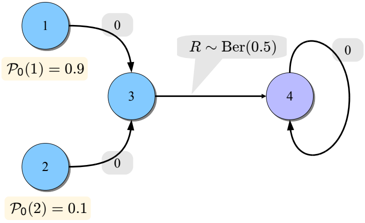
- “Robbins-Monro” conditions:

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

Application to learning a value: the Monte-Carlo method

Setup:

- Finite, episodic MDP
- Policy π, which is proper from x_0
- Goal: Estimate $V^\pi(x_0)$!
- Trajectories:



$$X_0^{(0)}, R_1^{(0)}, X_1^{(0)}, R_2^{(0)}, \ldots, X_{T_0}^{(0)},$$
$$X_0^{(1)}, R_1^{(1)}, X_1^{(1)}, R_2^{(1)}, \ldots, X_{T_1}^{(0)},$$
$$\vdots$$

where $X_0^{(i)} = x_0$

First-visit Monte-Carlo

function FIRSTVISITMC($\mathcal{T}, V, n$)

$\mathcal{T} = (X_0, R_1, \ldots, R_T, X_T)$ is a trajectory with $X_0 = x$ and X_T being an absorbing state, n is the number of times V was updated

$\text{sum} \leftarrow 0$

for $t = 0$ **to** $T - 1$ **do**

  $\text{sum} \leftarrow \text{sum} + \gamma^t R_{t+1}$

end for

$V \leftarrow V + \frac{1}{n}(\text{sum} - V)$

return V

Every-visit Monte-Carlo – learning a value function

function EVERYVISITMC($X_0, R_1, X_1, R_2, \ldots, X_{T-1}, R_T, V$)

Input: X_t is the state at time t, R_{t+1} is the reward associated with the t^{th} transition, T is the length of the episode, V is the array storing the current value function estimate

$\text{sum} \leftarrow 0$

for $t \leftarrow T-1$ **downto** 0 **do**

  $\text{sum} \leftarrow R_{t+1} + \gamma \cdot \text{sum}$

  $\text{target}[X_t] \leftarrow \text{sum}$

  $V[X_t] \leftarrow V[X_t] + \alpha \cdot (\text{target}[X_t] - V[X_t])$

end for

return V

Learning from snippets of data

Goals

- Learn from elementary transitions of the form (X_t, R_{t+1}, X_{t+1})
- Learn a full value function
- Increase convergence rate (if possible)

$\Longrightarrow$ Temporal Difference (TD) Learning

Learning from guesses: TD learning

- Idealistic Monte-Carlo update:

$$V(X_t) \leftarrow V(X_t) + \frac{1}{t}(\mathcal{R}_t - V(X_t)),$$

  where $\mathcal{R}_t$ is the return from state X_t.
- However, $\mathcal{R}_t$ is not available!
- Idea: Replace it with something computable:

$$\begin{aligned} \mathcal{R}_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \\ &= R_{t+1} + \gamma \{R_{t+2} + \gamma R_{t+3} + \ldots\} \\ &\approx R_{t+1} + \gamma V(X_{t+1}). \end{aligned}$$

- Update:

$$V(X_t) \leftarrow V(X_t) + \frac{1}{t} \overbrace{\{R_{t+1} + \gamma V(X_{t+1}) - V(X_t)\}}^{\delta_{t+1}(V)}.$$

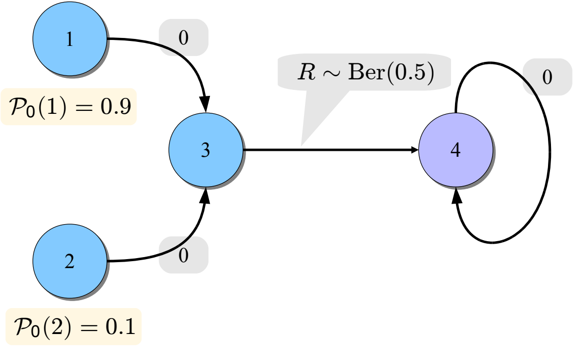
The TD(0) algorithm

function TD0(X, R, Y, V)

Input: X is the last state, Y is the next state, R is the immediate reward associated with this transition, V is the array storing the current value estimates

1: $\delta \leftarrow R + \gamma \cdot V[Y] - V[X]$
2: $V[X] \leftarrow V[X] + \alpha \cdot \delta$
3: **return** V

Which one to love? Part I



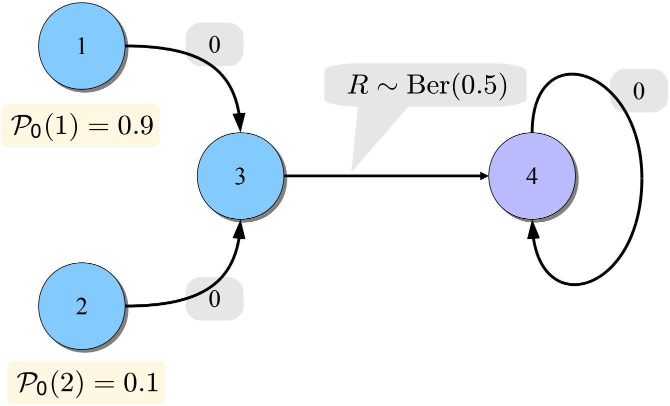
TD(0) at state 2:

- By the k^{th} visit to state 2, state 3 has already been visited $\approx 10\,k$ times!
- $\mathrm{Var}\left[\hat{V}_t(3)\right] \approx 1/(10\,k)$!

MC at state 2:

- $\mathrm{Var}\left[\mathcal{R}_t | X_t = 2\right] = 0.25$, does not decrease with k!

Which one to love? Part II



- Replace the stochastic reward by a deterministic one of value 1
- TD has to wait until the value of 3 converges
- MC updates towards the correct value in every step (no variance!)

The happy compromise: TD(λ)

- Choose $0 \le \lambda \le 1$
- Consider the k-step return estimate:

$$\mathcal{R}_{t:k} = \sum_{s=t}^{t+k} \gamma^{s-t} R_{s+1} \;+\; \gamma^{k+1}\, \hat{V}_t(X_{t+k+1}),$$

- Consider updating the values toward the so-called λ-return estimate:

$$\mathcal{R}_t^{(\lambda)} = \sum_{k=0}^{\infty} (1-\lambda)\lambda^k\, \mathcal{R}_{t:k}.$$

Toward TD(λ)

$$\mathcal{R}_{t:k} = \sum_{s=t}^{t+k} \gamma^{s-t} R_{s+1} + \gamma^{k+1} \hat{V}_t(X_{t+k+1}), \quad \mathcal{R}_t^{(\lambda)} = \sum_{k=0}^{\infty} (1-\lambda)\lambda^k \mathcal{R}_{t:k}.$$

$$\begin{aligned}
\mathcal{R}_t^{(\lambda)} - \hat{V}_t(X_t) &= (1-\lambda) \left\{ R_{t+1} + \gamma \hat{V}_t(X_{t+1}) - \hat{V}_t(X_t) \right\} + \\
&\quad (1-\lambda)\lambda \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 \hat{V}_t(X_{t+2}) - \hat{V}_t(X_t) \right\} + \\
&\quad (1-\lambda)\lambda^2 \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 \hat{V}_t(X_{t+3}) - \hat{V}_t(X_t) \right\} + \\
&\vdots \\
&= \left[R_{t+1} + \gamma \hat{V}_t(X_{t+1}) - \hat{V}_t(X_t) \right] \\
&\quad \gamma\lambda \left[R_{t+2} + \gamma \hat{V}_t(X_{t+2}) - \hat{V}_t(X_{t+1}) \right] + \\
&\quad \gamma^2\lambda^2 \left[R_{t+3} + \gamma \hat{V}_t(X_{t+3}) - \hat{V}_t(X_{t+2}) \right] + \\
&\vdots
\end{aligned}$$

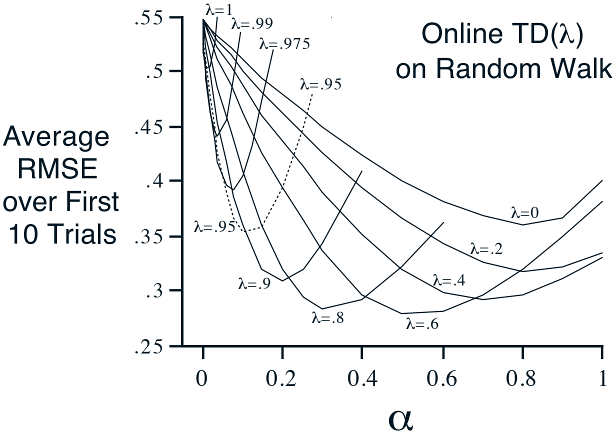
The TD(λ) algorithm

function TDLAMBDA(X, R, Y, V, z)

Input: X is the last state, Y is the next state, R is the immediate reward associated with this transition, V is the array storing the current value function estimate, z is the array storing the eligibility traces

$\delta \leftarrow R + \gamma \cdot V[Y] - V[X]$
for all $x \in \mathcal{X}$ **do**
    $z[x] \leftarrow \gamma \cdot \lambda \cdot z[x]$
    if $X = x$ **then**
        $z[x] \leftarrow 1$
    end if
    $V[x] \leftarrow V[x] + \alpha \cdot \delta \cdot z[x]$
end for
return (V, z)

Experimental results



Problem: 19-state random walk on a chain. Reward of 1 at the left end. Both ends are absorbing. The goal is to predict the values of states.

Too many states! What to do?

- The state space is too large
  - Cannot store all the values
  - Cannot visit all the states!
- What to do???
- Idea: Use compressed representations!
- Examples
  - Discretization
  - Linear function approximation
  - Nearest neighbor methods
  - Kernel methods
  - Decision trees
  - $\vdots$
- How to use them?

Regression with stochastic gradient descent

- Assume $(X_1, R_1), (X_2, R_2), \ldots$ are i.i.d., $\exists V(x) = \mathbb{E}[R_t | X_t = x]$.
- Goal:
  - Estimate V!
  - With a function of the form $V_\theta(x) = \theta^\top \varphi(x)$
- This is called regression in statistics/machine learning
- More precise goal: Minimize the expected squared prediction error:

$$J(\theta) = \frac{1}{2} \mathbb{E}\left[(R_t - V_\theta(X_t))^2 \right].$$

- Stochastic gradient descent:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha_t \frac{1}{2} \nabla_\theta (R_t - V_{\theta_t}(X_t))^2 \\ &= \theta_t + \alpha_t (R_t - V_{\theta_t}(X_t)) \nabla_t V_{\theta_t}(X_t) \\ &= \theta_t + \alpha_t (R_t - V_{\theta_t}(X_t)) \varphi(X_t). \end{aligned}$$

- "Robbins-Monro" conditions: $\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty.$

Limit theory

- Stochastic gradient descent:

$$\theta_{t+1} - \theta_t \quad = \quad \alpha_t \left(R_t - V_{\theta_t}(X_t) \right) \varphi(X_t).$$

- "Robbins-Monro" conditions: $\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty.$
- If converges, it must converge to θ^* satisfying

$$\mathbb{E}\left[\left(R_t - V_\theta(X_t) \right) \varphi(X_t) \right] = 0.$$

- Explicit form:

$$\theta^* = \mathbb{E}\left[\varphi_t \varphi_t^\top \right]^{-1} \mathbb{E}\left[\varphi_t R_t \right],$$

  where $\varphi_t = \varphi(X_t)$.
- Indeed a minimizer of J.
- "LMS rule", "Widrow-Hoff" rule, "delta-rule", "ADALINE"

Learning from guesses: TD learning with function approximation

- Replace reward with return!
- Idealistic Monte-Carlo based update:

$$\theta_{t+1} = \theta_t + \alpha_t \left(\mathcal{R}_t - V_{\theta_t}(X_t) \right) \nabla_\theta V_{\theta_t}(X_t)$$

  where $\mathcal{R}_t$ is the return from state X_t.
- However, $\mathcal{R}_t$ is not available!
- Idea: Replace it with an estimate:

$$\mathcal{R}_t \quad \approx \quad R_{t+1} + \gamma V_{\theta_t}(X_{t+1}).$$

- Update:

$$\theta_{t+1} = \theta_t + \alpha_t \overbrace{\left\{ R_{t+1} + \gamma V_{\theta_t}(X_{t+1}) - V_{\theta_t}(X_t) \right\}}^{\delta_{t+1}(V_{\theta_t})} \nabla_\theta V_{\theta_t}(X_t)$$

TD(λ) with linear function approximation

function TDLAMBDALINFAPP(X, R, Y, θ, z)

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

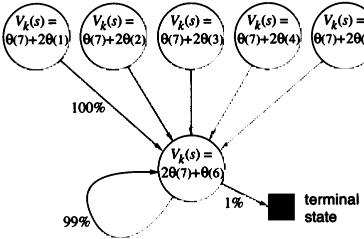
1: $\delta \leftarrow R + \gamma \cdot \theta^\top \varphi[Y] - \theta^\top \varphi[X]$

2: $z \leftarrow \varphi[X] + \gamma \cdot \lambda \cdot z$

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

4: **return** (θ, z)

Issues with off-policy learning



5-star example

Behavior of TD(0) with expected backups on the 5-star example

Defining the objective function

- Let $\delta_{t+1}(\theta) = R_{t+1} + \gamma V_\theta(Y_{t+1}) - V_\theta(X_t)$ be the TD-error at time t, $\varphi_t = \varphi(X_t)$.
- TD(0) update:
$$\theta_{t+1} - \theta_t = \alpha_t \, \delta_{t+1}(\theta_t)\varphi_t.$$
- When TD(0) converges, it converges to a unique vector θ^* that satisfies
$$\mathbb{E}\left[\delta_{t+1}(\theta^*)\varphi_t\right] = 0. \tag{TDEQ}$$
- Goal: Come up with an objective function such that its optima satisfy (TDEQ).
- Solution:
$$J(\theta) = \mathbb{E}\left[\delta_{t+1}(\theta)\varphi_t\right]^\top \mathbb{E}\left[\varphi_t\varphi_t^\top\right]^{-1} \mathbb{E}\left[\delta_{t+1}(\theta)\varphi_t\right].$$

Deriving the algorithm

$$J(\theta) = \mathbb{E}\left[\delta_{t+1}(\theta)\varphi_t\right]^\top \mathbb{E}\left[\varphi_t\varphi_t^\top\right]^{-1} \mathbb{E}\left[\delta_{t+1}(\theta)\varphi_t\right].$$

- Take the gradient!

$$\nabla_\theta J(\theta) = -2\mathbb{E}\left[(\varphi_t - \gamma\varphi'_{t+1})\varphi_t^\top\right] w(\theta),$$

where

$$w(\theta) = \mathbb{E}\left[\varphi_t\varphi_t^\top\right]^{-1} \mathbb{E}\left[\delta_{t+1}(\theta)\varphi_t\right].$$

- Idea: introduce two sets of weights!

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t \cdot (\varphi_t - \gamma \cdot \varphi'_{t+1}) \cdot \varphi_t^\top w_t \\ w_{t+1} &= w_t + \beta_t \cdot (\delta_{t+1}(\theta_t) - \varphi_t^\top w_t) \cdot \varphi_t. \end{aligned}$$

GTD2 with linear function approximation

function GTD2(X, R, Y, θ, w)

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

$f \leftarrow \varphi[X]$

$f' \leftarrow \varphi[Y]$

$\delta \leftarrow R + \gamma \cdot \theta^\top f' - \theta^\top f$

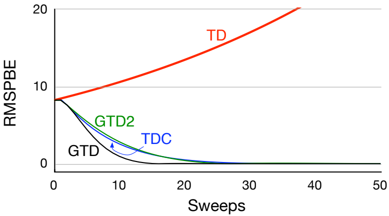
$a \leftarrow f^\top w$

$\theta \leftarrow \theta + \alpha \cdot (f - \gamma \cdot f') \cdot a$

$w \leftarrow w + \beta \cdot (\delta - a) \cdot f$

return (θ, w)

Experimental results



Behavior on 7-star example

Bibliographic notes and subsequent developments

- GTD – the original idea (Sutton et al., 2009b)
- GTD2, a two-timescale version (TDC) (Sutton et al., 2009a).

  Just replace the update in line 5 by

$$\theta \leftarrow \theta + \alpha \cdot (\delta \cdot f - \gamma \cdot a \cdot f').$$

- Extension to nonlinear function approximation (Maei et al., 2010)

  Addresses the issue that TD is unstable when used with nonlinear function approximation
- Extension to eligibility traces, action-values (Maei and Sutton, 2010)
- Extension to control (next part!)

The problem

- The methods are “gradient”-like, or “first-order methods”
- Make small steps in the weight space
- They are sensitive to:
  - choice of the step-size
  - initial values of weights
  - eigenvalue spread of the underlying matrix determining the dynamics
- Solution proposals:
  - Use of adaptive step-sizes (Sutton, 1992; George and Powell, 2006)
  - Normalizing the updates (Bradtke, 1994)
  - Reusing previous samples (Lin, 1992)
- Each of them have their own weaknesses

The LSTD algorithm

- In the limit, if TD(0) converges it finds the solution to
  $(*) \quad \mathbb{E}\left[\, \varphi_t \, \delta_{t+1}(\theta) \,\right] = 0.$
- Assume the sample so far is

$$\mathcal{D}_n = ((X_0, R_1, Y_1), (X_1, R_2, Y_2), \ldots, (X_{n-1}, R_n, Y_n)),$$

- Idea: Approximate (*) by $\quad (**) \quad \frac{1}{n}\sum_{t=0}^{n-1} \varphi_t \, \delta_{t+1}(\theta) = 0.$
  - Stochastic programming: *sample average approximation* (Shapiro, 2003)
  - Statistics: Z-estimation (e.g., Kosorok, 2008, Section 2.2.5)
- Note: (**) is equivalent to

$$-\hat{A}_n\theta + \hat{b}_n = 0,$$

  where $\hat{b}_n = \frac{1}{n}\sum_{t=0}^{n-1} R_{t+1}\varphi_t$ and $\hat{A}_n = \frac{1}{n}\sum_{t=0}^{n-1} \varphi_t(\varphi_t - \gamma\varphi'_{t+1})^\top$.
- Solution: $\theta_n = \hat{A}_n^{-1}\hat{b}_n$, provided the inverse exists.
- Least-squares temporal difference learning or LSTD (Bradtke and Barto, 1996).

RLSTD(0) with linear function approximation

function RLSTD(X, R, Y, C, θ)

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

$f \leftarrow \varphi[X]$
$f' \leftarrow \varphi[Y]$
$g \leftarrow (f - \gamma f')^\top C$ $\triangleright$ g is a $1 \times d$ row vector
$a \leftarrow 1 + gf$
$v \leftarrow Cf$
$\delta \leftarrow R + \gamma \cdot \theta^\top f' - \theta^\top f$
$\theta \leftarrow \theta + \delta \,/\, a \cdot v$
$C \leftarrow C - v\, g \,/\, a$
return (C, θ)

Which one to love?

Assumptions

- Time for computation T is fixed
- Samples are cheap to obtain

Some facts

How many samples (n) can be processed?

- Least-squares: $n \approx T/d^2$
- First-order methods: $n' \approx T/d = nd$

Precision after t samples?

- Least-squares: $C_1 t^{-\frac{1}{2}}$
- First-order: $C_2 t^{-\frac{1}{2}}$
- $C_2 > C_1$

Conclusion

Ratio of precisions:

$$\frac{\|\theta'_{n'} - \theta_*\|}{\|\theta_n - \theta_*\|} \approx \frac{C_2}{C_1} d^{-\frac{1}{2}},$$

Hence: *If $C_2/C_1 < d^{1/2}$ then the first-order method wins, in the other case the least-squares method wins.*

The choice of the function approximation method

Factors to consider

- Quality of the solution in the limit of infinitely many samples
- Overfitting/underfitting
- “Eigenvalue spread” (decorrelated features) when using first-order methods

Error bounds

Consider TD(λ) estimating the value function V. Let $V_{\theta(\lambda)}$ be the limiting solution. Then

$$\|V_{\theta(\lambda)} - V\|_\mu \le \frac{1}{\sqrt{1-\gamma_\lambda}} \|\Pi_{\mathcal{F},\mu} V - V\|_\mu .$$

Here $\gamma_\lambda = \gamma(1-\lambda)/(1-\lambda\gamma)$ is the contraction modulus of $\Pi_{\mathcal{F},\mu} T^{(\lambda)}$ (Tsitsiklis and Van Roy, 1999; Bertsekas, 2007).

Error analysis II

- Define the Bellman error $\Delta^{(\lambda)}(\hat{V}) = T^{(\lambda)}\hat{V} - \hat{V}$, $\hat{V} : \mathcal{X} \to \mathbb{R}$ under $T^{(\lambda)} = (1-\lambda)\sum_{m=0}^{\infty} \lambda^m T^{[m]}$, where $T^{[m]}$ is the m-step lookahead Bellman operator.
- Contraction argument: $\left\| V - \hat{V} \right\|_\infty \le \frac{1}{1-\gamma} \left\| \Delta^{(\lambda)}(\hat{V}) \right\|_\infty$.
- What makes $\Delta^{(\lambda)}(\hat{V})$ small?
- Error decomposition:

$$\Delta^{(\lambda)}(V_{\theta^{(\lambda)}}) = (1-\lambda)\sum_{m\ge 0} \lambda^m \Delta_m^{[r]} + \gamma \left\{ (1-\lambda) \sum_{m\ge 0} \lambda^m \Delta_m^{[\varphi]} \right\} \theta^{(\lambda)},$$

where

  - $\Delta_m^{[r]} = \bar{r}_m - \Pi_{\mathcal{F},\mu}\bar{r}_m$
  - $\Delta_m^{[\varphi]} = P^{m+1}\varphi^\top - \Pi_{\mathcal{F},\mu}P^{m+1}\varphi^\top$
  - $\bar{r}_m(x) = \mathbb{E}\left[R_{m+1} \,|\, X_0 = x\right]$,
  - $P^{m+1}\varphi^\top(x) = (P^{m+1}\varphi_1(x), \ldots, P^{m+1}\varphi_d(x))$,
  - $P^m\varphi_i(x) = \mathbb{E}\left[\varphi_i(X_m) \,|\, X_0 = x\right]$.

For Further Reading

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, MA, 3 edition.

Bradtke, S. J. (1994). *Incremental Dynamic Programming for On-line Adaptive Optimal Control*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts.

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

George, A. P. and Powell, W. B. (2006). Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65:167–198.

Kosorok, M. R. (2008). *Introduction to Empirical Processes and Semiparametric Inference*. Springer.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 9:293–321.

Maei, H., Szepesvári, C., Bhatnagar, S., Silver, D., Precup, D., and Sutton, R. (2010). Convergent temporal-difference learning with arbitrary smooth function approximation. In *NIPS-22*, pages 1204–1212.

Maei, H. R. and Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In Baum, E., Hutter, M., and Kitzelmann, E., editors, *AGI 2010*, pages 91–96. Atlantis Press.

Shapiro, A. (2003). Monte Carlo sampling methods. In *Stochastic Programming, Handbooks in OR & MS*, volume 10. North-Holland Publishing Company, Amsterdam.

Sutton, R. S. (1992). Gain adaptation beats least squares. In *Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems*, pages 161—166.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009a). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In Bottou, L. and Littman, M., editors, *ICML 2009*, pages 993—1000. ACM.

Sutton, R. S., Szepesvári, C., and Maei, H. R. (2009b). A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *NIPS-21*, pages 1609–1616. MIT Press.

Tsitsiklis, J. N. and Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica*, 35(11):1799–1808.