TD(λ) and the Proximal Algorithm

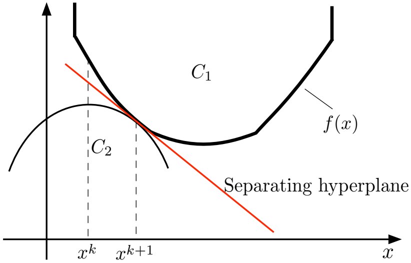
Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Optimization Methods Workshop

NIPS 2017

A Bridge Between Convex Analysis and Approximate Dynamic Programming



Convex Analysis

- Deterministic Problems
- Geometric Ideas
- Iterative Descent
- Proximal Algorithms

Approximate DP

- Stochastic Problems
- Simulation Ideas
- Value and Policy Iteration
- Temporal Differences

Fixed Point Problem Formulation

Problem: Solve $x = T(x)$

We assume that $T : \Re^n \mapsto \Re^n$ has a unique fixed point and is nonexpansive,

$$\big\|T(x_1) - T(x_2)\big\| \leq \gamma \|x_1 - x_2\|, \qquad \forall\, x_1, x_2 \in \Re^n,$$

where $0 \leq \gamma \leq 1$ and $\|\cdot\|$ is some norm

Special focus for this talk: The linear case

$$x = Ax + b$$

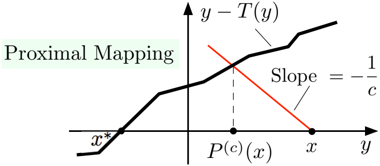
where $I - A$ is invertible and A has spectral radius ≤ 1

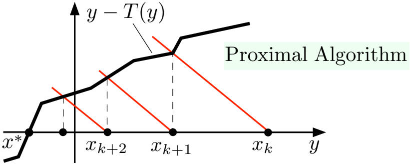
The proximal mapping $P^{(c)} : \Re^n \mapsto \Re^n$ for $x - T(x) = 0$, where $c > 0$

$$P^{(c)}(x) = \text{Unique solution of} \quad y - T(y) = \frac{1}{c}(x - y)$$

The proximal algorithm is

$$x_{k+1} = P^{(c)}(x_k)$$





Policy Iteration in DP - Bellman's Equation

Solve $x = T(x) = \min_\mu T_\mu(x)$ where μ: a policy, $T_\mu(x) = A_\mu x + b_\mu$

Policy iteration alternates between policy evaluation and policy improvement

$$x_k = T_{\mu_k}(x_k), \quad \text{(linear)}, \qquad \mu_{k+1} \in \arg\min_\mu T_\mu(x_k), \quad \text{(componentwise)}$$

Alternative policy evaluation based on the TD (Temporal Differences) mapping

For $\lambda \in (0,1)$ solve $x = T_\mu^{(\lambda)}(x)$ where $T_\mu^{(\lambda)}$ is the multistep linear mapping

$$T_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T_\mu^{\ell+1}$$

that has the same fixed point as T_μ

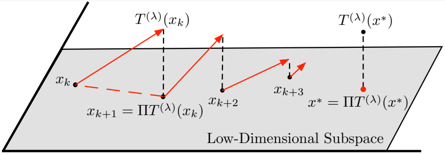
Iterative policy evaluation: Iterate one or more times with T_{μ_k} or $T_{\mu_k}^{(\lambda)}$

- $x_{k+1} = T_{\mu_k}(x_k)$ (value iteration) or $x_{k+1} = T_{\mu_k}^{(\lambda)}(x_k)$ (λ-policy evaluation)
- $x_{k+1} = x_k + \gamma_k\big(\text{sample}\, T_{\mu_k}^{(\lambda)}(x_k) - x_k\big)$ with $\gamma_k \downarrow 0$ (TD(λ) algorithm)

Policy Evaluation with Subspace Projection for Very Large Problems

Use intermediate projection onto a subspace of basis functions

For a fixed policy, solve the projected equation $x = \Pi T^{(\lambda)}(x)$ (Galerkin approximation)



$$x_{k+1} = \Pi T^{(\lambda)}(x_k), \qquad \text{Projected } \lambda\text{-policy evaluation}$$

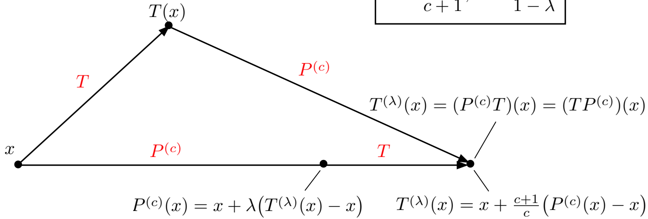
$$x_{k+1} = x_k + \gamma_k \Pi\big(\text{sample } T^{(\lambda)}(x_k) - x_k\big), \qquad \text{Projected TD}(\lambda)$$

Simulation-based implementations (key characteristic of RL)

- For large dimension (e.g., $n > 10^{10}$) there is no alternative to simulation (because of high-dimensional inner products)
- How simulation is implemented makes a big difference; e.g., sample collection, correlations, bias, choice of λ, etc. (We will not deal with that in this lecture.)
- A lot of know-how has been accumulated over the last 30 years

TD(λ) Map for Policy Evaluation $\approx$ Proximal Map for the Lin. Bellman Eq

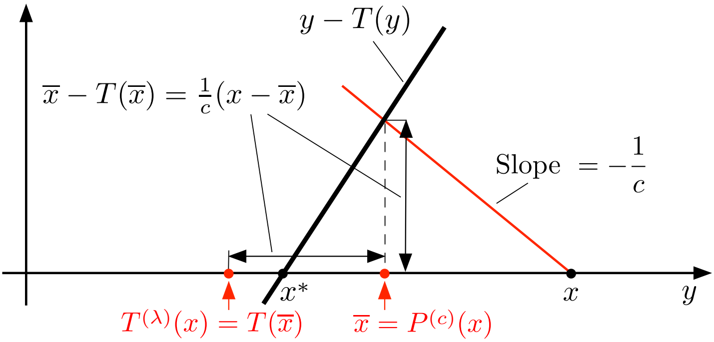
$$\boxed{\lambda = \frac{c}{c+1}, \;\; c = \frac{\lambda}{1-\lambda}}$$



Extrapolation Formula $T^{(\lambda)} = T \cdot P^{(c)} = P^{(c)} \cdot T$

$T^{(\lambda)}$ IS FASTER

TD(λ) IS A STOCHASTIC PROXIMAL ALGORITHM FOR LINEAR FIXED POINTS



The extrapolated iterate $T(\overline{x})$ is closer to x^* than the proximal iterate $\overline{x}$

A FREE LUNCH

Potential Implications of the TD-Proximal Relation

Benefit to the TD context

- Clarify the nature of TD(λ) and other TD methods
- Bring proximal methodology and insights to bear on exact and approximate DP

Benefit to the proximal context

- Bring large scale DP/RL methodology to bear on the proximal mainstream
- Develop new convex analysis algorithms based on DP/RL ideas

References for this Talk

- D. P. Bertsekas, “Proximal Algorithms and Temporal Differences for Large Linear Systems: Extrapolation, Approximation, and Simulation," Report LIDS-P-3205, MIT, Oct. 2016 (rev. Nov. 2017)

Related book references:

- D. P. Bertsekas, Abstract Dynamic Programming, 2nd Edition, in press
- D. P. Bertsekas, Convex Optimization Algorithms, 2015
- D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, 1996

Related works on Monte Carlo solution methods for linear systems:

- D. P. Bertsekas and H. Yu, “Projected Equation Methods for Approximate Solution of Large Linear Systems," J. of Comp. and Applied Mathematics, Vol. 227, 2009
- M. Wang and D. P. Bertsekas, “Convergence of Iterative Simulation-Based Methods for Singular Linear Systems", Stoch. Systems, Vol. 3, 2013
- M. Wang and D. P. Bertsekas, “Stabilization of Stochastic Iterative Methods for Singular and Nearly Singular Linear Systems", Math. of Op. Res., Vol. 39, 2013

Outline

The Extrapolation Formula

Let $c > 0$ and $\lambda = \frac{c}{c+1}$. Consider the proximal mapping

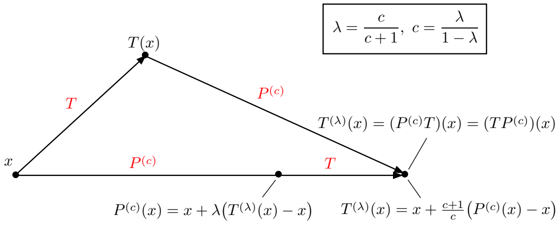
$$P^{(c)}(x) = \text{ Unique solution of } \quad y - T(y) = \frac{1}{c}(x - y)$$

Then:

$$T^{(\lambda)} = T \cdot P^{(c)} = P^{(c)} \cdot T$$

and x, $P^{(c)}(x)$, and $T^{(\lambda)}(x)$ are colinear:

$$T^{(\lambda)}(x) = P^{(c)}(x) + \frac{1}{c}\big(P^{(c)}(x) - x\big)$$



Extrapolation Formula $T^{(\lambda)} = T \cdot P^{(c)} = P^{(c)} \cdot T$

Proof outline

Main idea: Express the proximal mapping in terms of a power series

We have

$$P^{(c)}(x) = \left(\frac{c+1}{c}I - A\right)^{-1}\left(b + \frac{1}{c}x\right)$$

and by a series expansion

$$\left(\frac{c+1}{c}I - A\right)^{-1} = \left(\frac{1}{\lambda}I - A\right)^{-1} = \lambda(I - \lambda A)^{-1} = \lambda\sum_{\ell=0}^{\infty}(\lambda A)^{\ell}$$

Recall that

$$T^{(\lambda)}(x) = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell+1}x + \sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}b$$

Using these relations and the fact $\frac{1}{c} = \frac{1-\lambda}{\lambda}$, it follows that

$$T^{(\lambda)} = T \cdot P^{(c)} = P^{(c)} \cdot T$$

The eigenvalues of $T^{(\lambda)}$ and $P^{(c)}$ are simply related:

$$\theta_i = \zeta_i \cdot \overline{\theta}_i$$

where

$$\theta_i = i\text{th Eig}(T^{(\lambda)}), \qquad \overline{\theta}_i = i\text{th Eig}(P^{(c)}), \qquad \zeta_i = i\text{th Eig}(A)$$

Moreover, $T^{(\lambda)}$ and $P^{(c)}$ have the same eigenvectors

Spectral radius of $T^{(\lambda)}$ $\leq$ Spectral radius of $P^{(c)}$

Nonlinear System $x = T(x)$ - Proximal Extrapolation

- Assume that the system has a unique solution x^*, and T is nonexpansive:

$$\big\|T(x_1) - T(x_2)\big\| \le \gamma\|x_1 - x_2\|, \qquad \forall\, x_1, x_2 \in \Re^n$$

where $\|\cdot\|$ is some Euclidean norm and γ is a scalar with $0 \le \gamma \le 1$

- Consider the proximal mapping $P^{(c)}$:

$$P^{(c)}(x) = \text{ Unique solution of } \quad y - T(y) = \frac{1}{c}(x - y)$$

- Define the extrapolated proximal mapping

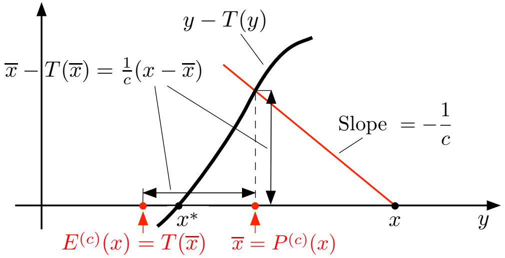
$$E^{(c)}(x) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big)$$

- Important difference: $P^{(c)}(x)$ and $E^{(c)}(x)$ cannot be easily computed by simulation

Similar to the linear case, we have

$$E^{(c)}(x) = T\big(P^{(c)}(x)\big), \qquad \big\|E^{(c)}(x) - x^*\big\| \le \gamma\big\|P^{(c)}(x) - x^*\big\|$$

Geometric Interpretation and Proof



From the definition of $P^{(c)}$, we have

$$T\big(P^{(c)}(x)\big) = P^{(c)}(x) + \frac{1}{c}\big(P^{(c)}(x) - x\big)$$

so that

$$T\big(P^{(c)}(x)\big) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big) \stackrel{\text{def}}{=} E^{(c)}(x)$$

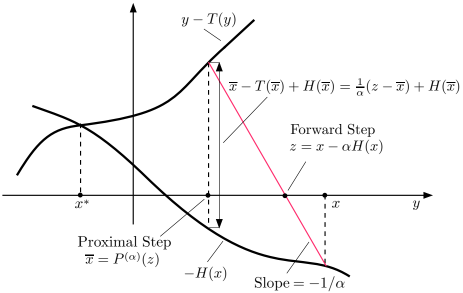
Hence, using the assumption,

$$\big\|E^{(c)}(x) - x^*\big\| = \Big\|T\big(P^{(c)}(x)\big) - x^*\Big\| = \Big\|T\big(P^{(c)}(x)\big) - T(x^*)\Big\| \leq \gamma\big\|P^{(c)}(x) - x^*\big\|$$

1. Acceleration of the Proximal Algorithm for Linear Systems
2. Acceleration of the Proximal Algorithm for Nonlinear Systems
3. Acceleration of Forward-Backward and Proximal Gradient Algorithms
4. Linearized Proximal Algorithms for Nonlinear Systems

Forward-Backward Splitting Algorithm for Fixed Point Problem $x = T(x) - H(x)$

$$x_{k+1} = P^{(\alpha)}\big(x_k - \alpha H(x_k)\big), \qquad \alpha > 0$$



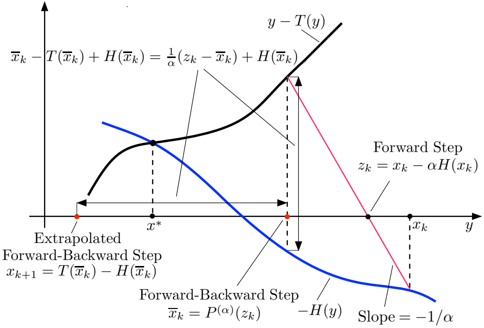
Properties (Lions and Mercier, 1979, Gabay, 1983, Tseng, 1991):

- If T is nonexpansive, and H is single-valued and strongly monotone, the F-B algorithm converges to x^* if α is sufficiently small
- For a minimization problem where H is the gradient of a strongly convex function, FB = the proximal gradient algorithm

Extrapolated forward-backward algorithm

$$z_k = x_k - \alpha H(x_k), \qquad \overline{x}_k = P^{(\alpha)}(z_k) \qquad \text{(Forward-Backward Iteration)}$$

$$x_{k+1} = \overline{x}_k + \frac{1}{\alpha}(\overline{x}_k - z_k) - H(\overline{x}_k) \qquad \text{(Extrapolation)}$$



We have

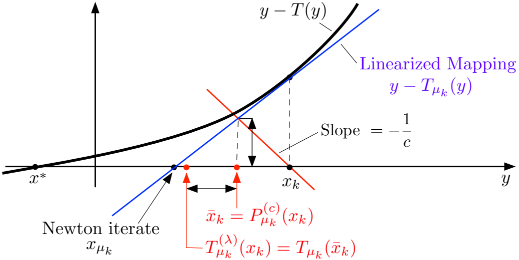
$$x_{k+1} = T(\overline{x}_k) - H(\overline{x}_k)$$

so there is acceleration if $T - H$ is contractive

Connect with TD: Apply Policy Iteration Ideas to the Proximal Context

Problem: Solve "concave" fixed point problem $x = T(x) = \min_\mu T_\mu(x)$

$$i\text{th component of } T_\mu(x) = a\big(i,\mu(i)\big)'x + b\big(i,\mu(i)\big), \qquad \mu(i) \in \mathcal{M}(i) \quad \text{(Linear)}$$



The policy iteration algorithm

$$x_{\mu_k} = T_{\mu_k}(x_{\mu_k}), \qquad \mu_{k+1} \in \arg\min_\mu T_\mu(x_{\mu_k}), \qquad \text{("Newton" method)}$$

The λ-policy iteration algorithm

$$x_{k+1} = T^{(\lambda)}_{\mu_k}(x_k), \qquad \mu_{k+1} \in \arg\min_\mu T_\mu(x_{k+1}), \qquad \text{("prox-linear" method)}$$

A Fundamental Difficulty

- The algorithm chases a moving target
- The TD(λ) mapping $T_{\mu_k}^{(\lambda)}$ “targets" the fixed point of T_{μ_k}, but as μ_k changes so does the target ...
- This is why some policy iteration algorithms may not converge (particularly with cost function approximation) ...

Convergence Under Monotonicity Assumptions

Assume the following:

- For all μ, the matrix A_μ has nonnegative components
- The mappings T_μ are all contractions with respect to a common sup-norm (this can be relaxed ...)

Then:

- T is a contraction and its fixed point is $x^* = \min_\mu x_\mu$
- A sequence $\{x_k\}$ generated from an initial condition x_0 such that $x_0 \geq T(x_0)$ is monotonically nonincreasing and converges to x^* (this can be improved ...)

Proof idea:

- Based on DP/policy iteration arguments
- Monotonicity is critical
- Once projection is introduced in policy iteration, monotonicity may be lost

Convergence of Randomized Version Without Monotonicity

Assume the following:

- The set $\mathcal{M}$ is finite
- The mappings T_μ and T are all contractions with respect to a common norm
- We use a randomized form of the linearized iteration:

$$x_{k+1} = T_{\mu_k}(x_k), \qquad \text{with probability } p,$$

$$x_{k+1} = T_{\mu_k}^{(\lambda)}(x_k), \qquad \text{with probability } 1 - p,$$

followed by $\mu_{k+1} \in \arg\min_\mu T_\mu(x_k)$

Then:

For any starting point x_0, a sequence $\{x_k\}$ generated by the algorithm converges to the fixed point of T with probability one

Randomization resolves the “moving target" problem

Concluding Remarks

- Proximal and multistep/TD iterations for fixed point problems are closely connected
- x, $P^{(c)}(x)$, and $T^{(\lambda)}(x)$ are colinear and simply related (no line search needed)
- TD(λ) mapping is “faster" than proximal
- A free lunch: Acceleration of the proximal algorithm. It can be substantial, particularly for small c
- Extrapolation formula provides new insight and justification for TD-type methods
  - TD(λ) is stochastic version of the proximal algorithm
  - TD(λ) with subspace approximation is stochastic version of the projected proximal
- The ideas extend to the forward-backward algorithm and potentially other algorithmic contexts that involve fixed points and proximal operators
- The relation between proximal and TD methods extends to classes of nonlinear fixed point problems using linearization/policy iteration ideas

Thank you!