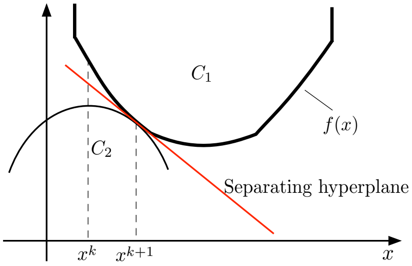
Proximal Algorithms and Temporal Difference Methods

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

January 2017

A Bridge Between Convex Analysis and Approximate Dynamic Programming



Convex Analysis

Deterministic Problems
Proximal Algorithms
Iterative Regularization
Hyperplane Separation
Iterative Descent

Approximate DP

Stochastic Problems
Policy Iteration
Large Linear Systems of Equations
Simulation-Temporal Differences
AlphaGo

Problem Formulation: Find a Fixed Point of a Nonexpansive Mapping

Problem: Solve $x = T(x)$

where 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 Euclidean norm.

Primary focus: The linear case

$$x = Ax + b$$

where $I - A$ is nearly singular and/or has huge dimension.

- “Nearly singular" suggests the use of regularization and the proximal algorithm.
- “Huge dimension" suggests projection over a low-dimensional subspace and simulation.

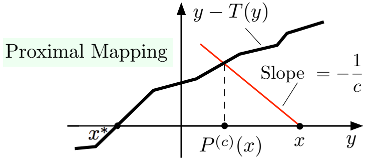
Proximal Algorithm - Convex Analysis (Martinet, 1970)

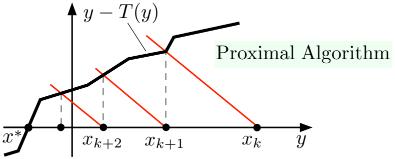
The proximal mapping $P^{(c)} : \Re^n \mapsto \Re^n$ for $x - T(x) = 0$, where $c > 0$

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

The proximal algorithm is

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





Special case: Convex minimization $\min_{x \in \Re^n} f(x)$, or $\nabla f(x) = 0$

$$T(x) = x - \nabla f(x), \qquad f: \text{ Convex differentiable function}$$

$$P^{(c)}(x) = \arg \min_{y \in \Re^n} \left\{ f(y) + \frac{1}{2c} \|y - x\|^2 \right\}$$

Multistep Mappings - Temporal Differences - DP (1990s)

Consider the special case of a linear system $T(x) = Ax + b$

For $\lambda \in (0,1)$, introduce the multistep mapping

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

- $T^{(\lambda)}$ is linear: $T^{(\lambda)}(x) = A^{(\lambda)}x + b^{(\lambda)}$, where

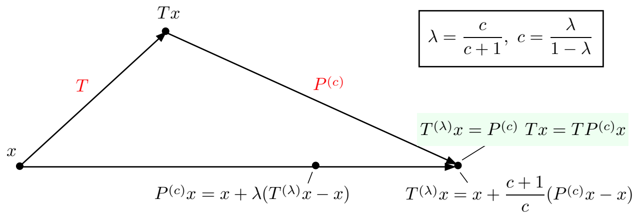
$$A^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell+1}, \qquad b^{(\lambda)} = \sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell} b$$

- $T^{(\lambda)}$ has the same fixed point as T

Algorithms (central in approximate DP/policy iteration/policy evaluation, where T is the Bellman equation mapping of a policy)

- $x_{k+1} = T(x_k)$ (value iteration) or $x_{k+1} = T^{(\lambda)}(x_k)$
- $x_{k+1} = x_k + \gamma_k\big(\text{sample}\,T^{(\lambda)}(x_k) - x_k\big)$ with $\gamma_k \downarrow 0$ (TD(λ) algorithm)
- Simulation-based with intermediate projection onto a subspace of basis functions

Key Fact: The Multistep Mapping is an Extrapolated and Faster Version of the Proximal Mapping

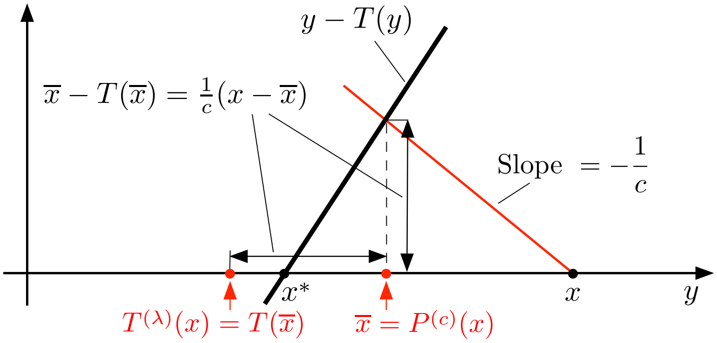

Tx
T
$P^{(c)}$
x
$T^{(\lambda)}x = P^{(c)}\,Tx = TP^{(c)}x$
$P^{(c)}x = x + \lambda(T^{(\lambda)}x - x)$
$T^{(\lambda)}x = x + \frac{c+1}{c}(P^{(c)}x - x)$

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

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

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

Extrapolated Iteration



$y - T(y)$
$\overline{x} - T(\overline{x}) = \frac{1}{c}(x - \overline{x})$
Slope $= -\frac{1}{c}$
x^*
x
y
$T^{(\lambda)}(x) = T(\overline{x})$
$\overline{x} = P^{(c)}(x)$

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

A FREE LUNCH

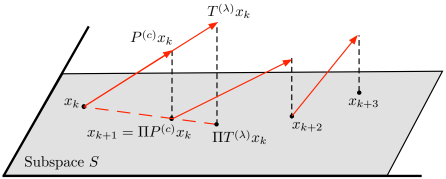
Another Possibility: Introduce Projection into the Proximal Algorithm

Consider Π: Projection Onto a Low-Dimensional Subspace

- Solve the projected proximal equation $x = \Pi P^{(c)}(x)$ [has the same solution as the multistep equation $x = \Pi T^{(\lambda)}(x)$]
- Use the projected proximal algorithm

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

modeled after the TD algorithms with projection.



The simulation-based TD methodology can be used in the proximal context

The sampled version of the projected proximal algorithm is identical to TD(λ)

$$x_{k+1} = x_k + \gamma_k\big(\text{sample } \Pi P^{(c)}(x_k) - x_k\big), \qquad \gamma_k \downarrow 0$$

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.
- D. P. Bertsekas, “Projected Proximal Algorithms for Large Linear Systems," in preparation.

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.

General textbook references:

- Convex Optimization Algorithms, 2015 (DPB).
- Dynamic Programming and Optimal Control: 4th edition, 2017 (DPB).

Outline

Properties of Multistep Mappings for Linear Systems

Consider the linear system $x = Ax + b$ under the following assumption:

The system has a unique solution x^* and spectral radius $\sigma(A) \leq 1$.

Some basic results (Bertsekas and Yu, 2009)

- The mapping $T^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^{\ell} T^{\ell+1}$ has the form

$$T^{(\lambda)}(x) = A^{(\lambda)}x + b^{(\lambda)},$$

  where

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

- The eigenvalues of $A^{(\lambda)}$ have the form

$$\theta_i = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^{\ell} \zeta_i^{\ell+1} = \frac{\zeta_i(1-\lambda)}{1-\zeta_i\lambda}, \qquad i = 1, \ldots, n,$$

  where ζ_i, $i = 1, \ldots, n$, are the eigenvalues of A. Furthermore,

$$\sigma(A^{(\lambda)}) < 1, \qquad \lim_{\lambda \to 1} \sigma(A^{(\lambda)}) = 0$$

The Extrapolation Formula

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

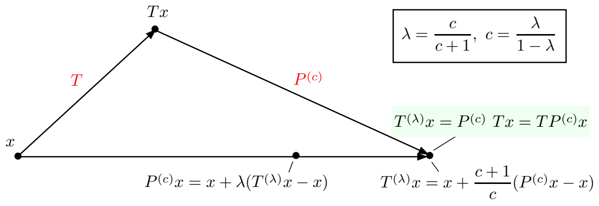
$$P^{(c)} : \; x \;\; \mapsto \;\; \text{Unique solution of} \quad y - T(y) = \frac{1}{c}(x - y)$$

Then:

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

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

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



Tx
T
$P^{(c)}$
x
$\lambda = \frac{c}{c+1}, \; c = \frac{\lambda}{1-\lambda}$
$T^{(\lambda)}x = P^{(c)}\,Tx = TP^{(c)}x$
$P^{(c)}x = x + \lambda(T^{(\lambda)}x - x)$
$T^{(\lambda)}x = x + \frac{c+1}{c}(P^{(c)}x - x)$

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

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

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

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

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

Acceleration

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

Convergence rate improvement: We have

$$\frac{\sigma(A^{(\lambda)})}{\sigma(A)} \leq \sigma(\overline{A}^{(\lambda)}) < 1$$

so $\sigma(A^{(\lambda)}) < \sigma(\overline{A}^{(\lambda)})$ if $\sigma(A) < 1$.

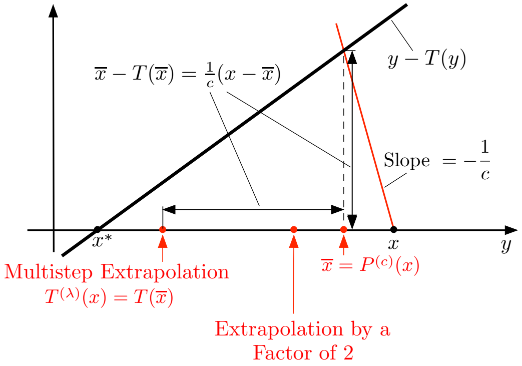
Optimal extrapolation

The eigenvalues of the extrapolated iteration

$$x_{k+1} = \big((1-\gamma)P^{(c)} + \gamma T^{(\lambda)}\big)x_k, \qquad \gamma > 0$$

are $\theta_i(\gamma) = (1-\gamma)\overline{\theta}_i + \gamma\theta_i$, and for some $\hat{\gamma} \geq 1$, we have acceleration for all $\gamma \in (0, \hat{\gamma})$.

A Note on Extrapolation in the Proximal Algorithm



$\overline{x} - T(\overline{x}) = \frac{1}{c}(x - \overline{x})$
$y - T(y)$
Slope $= -\frac{1}{c}$
x^*
x
y
$\overline{x} = P^{(c)}(x)$
Multistep Extrapolation
$T^{(\lambda)}(x) = T(\overline{x})$
Extrapolation by a Factor of 2

- It is well-known that extrapolation by any factor less than 2 preserves the convergence of the proximal algorithm, but does not guarantee acceleration.
- This is a different and unrelated old result (Bertsekas, 1975, for the convex minimization case, Eckstein and Bertsekas, 1992, for the general case).
- The acceleration result of this talk holds only for the fixed point/nonexpansive case $x = T(x)$.

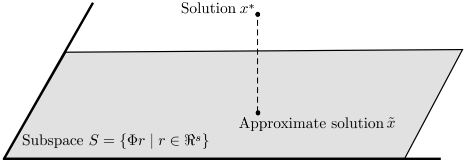
Approximation in a Subspace of Basis Functions

Approximate the solution x^* of $x = Ax + b$ within a low-dimensional subspace

Consider the subspace

$$S = \{\Phi r \mid r \in \Re^s\}$$

spanned by the columns $\phi_1, \dots, \phi_s$ of an $n \times s$ matrix Φ ($s << n$)

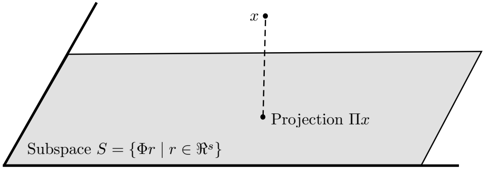

Solution x^*
Approximate solution $\tilde{x}$
Subspace $S = \{\Phi r \mid r \in \Re^s\}$

Examples

- Standard bases: Polynomials, radial basis functions, wavelets, etc
- Throw away some components of x interpolate for the rest
- Aggregation (e.g., form a smaller system using linear combinations of rows and columns of A)
- Feature-based approximation (features of the components of x are the rows of Φ - generated “manually" or “automatically", e.g., by a neural network)

How to Approximate Vectors x within S?

Introduce a “projection" operation $\Pi : \Re^n \to S$ (Π is linear and $\Pi x = x$ for all $x \in S$)



General form: Oblique Projection

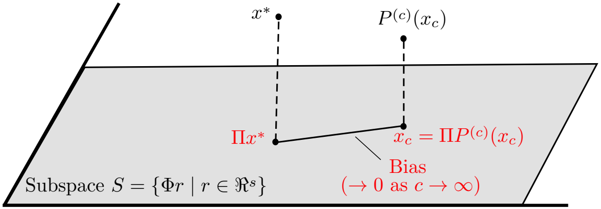
$$\Pi = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi,$$

where Ξ is a diagonal $n \times n$ positive semidefinite, and Ψ is an $n \times s$ matrix such that $\Psi'\Xi\Phi$ is invertible

Examples

- Orthogonal projection ($\Psi = \Phi$ and Ξ is positive definite)
- Seminorm projection (Ξ may have some 0 diagonal components)
- Aggregation ($\Pi = \Phi D$, where the rows of Φ and D are probability distributions)

Projected Proximal Equation: $x = \Pi P^{(c)}(x)$


x^*
$P^{(c)}(x_c)$
Πx^*
$x_c = \Pi P^{(c)}(x_c)$
Bias
$(\to 0$ as $c \to \infty)$
Subspace $S = \{\Phi r \mid r \in \Re^s\}$

Galerkin approximation approach: Project the equation not the solution

Recall the proximal equation

$$x = P^{(c)}(x) = \overline{A}^{(\lambda)} x + \overline{b}^{(\lambda)}$$

We solve the projected version $x = \Pi P^{(c)}(x)$ at the expense of "bias" $(x_c - \Pi x^*)$

Important Point: Large c diminishes the bias

$$x^* - x_c = \left(I - \Pi \overline{A}^{(\lambda)}\right)^{-1} (x^* - \Pi x^*)$$

We have $\lim_{\lambda \to 1} \overline{A}^{(\lambda)} = 0$, so the bias $(x_c - \Pi x^*) \to 0$ as $c \to \infty$

Algebraic Form of the Projected Proximal Equations

Recall the proximal equation

$$x = P^{(c)}(x) = \overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}$$

where

$$\overline{A}^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}, \qquad \overline{b}^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^{\ell}b, \qquad \lambda = \frac{c}{c+1}$$

For the oblique projection case $\Pi = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi$

The projected equation is the (low-dimensional linear equation) $r = Q^{(\lambda)}r + d^{(\lambda)}$ where

$$Q^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{A}^{(\lambda)}\Phi, \qquad d^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{b}^{(\lambda)}$$

Important point:

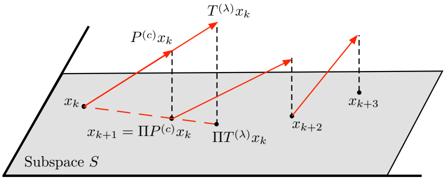
- $\overline{A}^{(\lambda)}$ and $\overline{b}^{(\lambda)}$ involve powers of matrices (which facilitates simulation)
- For any value of λ, $Q^{(\lambda)}$ and $d^{(\lambda)}$ can be evaluated by simulation, just as conveniently as for $\lambda = 0$

Projected Proximal and Proximal Projected Algorithms

Projected proximal: Fixed point algorithm for the projected proximal equation

$$x_{k+1} = \Pi P^{(c)}(x_k) \text{ or equivalently } r_{k+1} = Q^{(\lambda)} r_k + d^{(\lambda)}$$

Can also use its extrapolated version $x_{k+1} = \Pi T^{(\lambda)}(x_k)$. Converges if $\Pi P^{(c)}$ is a contraction (true if c is sufficiently large or if Π is properly chosen)



Proximal projected: Proximal algorithm for the low-dimensional proximal equation $r = Q^{(\lambda)} r + d^{(\lambda)}$

$$r_{k+1} = \hat{P}^{(\hat{c})}(r_k), \qquad (\hat{c} > 0\text{: unrelated to } c \text{ and } \lambda)$$

where $\hat{P}^{(\hat{c})} : \Re^s \mapsto \Re^s$ is the mapping

$$r \mapsto \text{ Unique solution of } y - Q^{(\lambda)} y - d^{(\lambda)} = \frac{1}{\hat{c}}(r - y)$$

The Need for Simulation for Large Systems

Recall the projected equation $r = Q^{(\lambda)} r + d^{(\lambda)}$

$$Q^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{A}^{(\lambda)}\Phi, \qquad d^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{b}^{(\lambda)}$$

$$\overline{A}^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}, \qquad \overline{b}^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^{\ell}b, \qquad \lambda = \frac{c}{c+1}$$

Need for simulation

- $Q^{(\lambda)}$ and $d^{(\lambda)}$ have low dimension but cannot be explicitly computed
- Reason: They involve HUGE-dimensional inner products
- Monte Carlo simulation can approximate HUGE-dimensional inner products
- Connection with Monte Carlo integration

Simulation-Based Methodology

Simulation Analytics

- Key idea: Interpret linear algebra operations (matrix products, inner products) as computing expected values with suitable distributions (matrix Ξ)
- Approximate the expected values by using sampling and laws of large numbers
- Generate samples of powers of A by using a suitable Markov chain

Important issues

- Contraction properties of $\Pi P^{(c)}$
- Choice of projection (norm mismatch issue)
- Near singularity of projected proximal equation (sensitivity to sampling error)
- Bias-variance tradeoff (as $\lambda \uparrow 1$, less bias, greater simulation error, more sampling needed)
- Issues of importance sampling

Simulation-Based Experience

- A happy union of research in AI (low-dimensional representations, deep neural networks, BIG data) and in control/OR (DP, optimization, aggregation, etc)
- Many algorithmic variations at the interface of DP, iterative stochastic optimization, Monte Carlo methods
- Challenging implementation, but very difficult problems can be addressed
- A long history of successful implementation in approximate DP
- Recent success story of AlphaGo program

Proximal Extrapolation for the Nonlinear System $x = T(x)$

- Assume that the system has a unique solution x^*, and T 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 $\|\cdot\|$ is some Euclidean norm and γ is a scalar with $0 \leq \gamma \leq 1$.
- Define the proximal mapping $P^{(c)}$:

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

- Consider the extrapolated proximal mapping

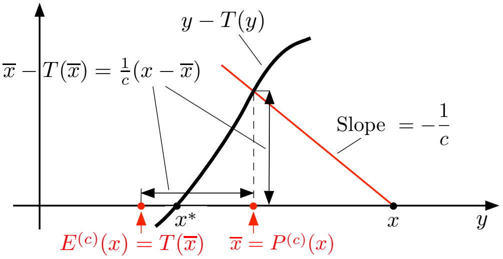
$$E^{(c)}(x) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big)$$

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

Acceleration Result: We have $E^{(c)}(x) = T\big(P^{(c)}(x)\big)$ and hence

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

Geometric Interpretation and Proof


$y - T(y)$
$\overline{x} - T(\overline{x}) = \frac{1}{c}(x - \overline{x})$
Slope $= -\frac{1}{c}$
x^*
x
y
$E^{(c)}(x) = T(\overline{x})$
$\overline{x} = P^{(c)}(x)$

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

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

so that

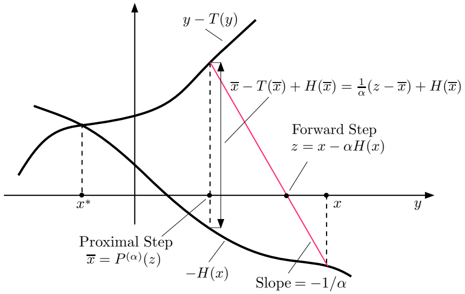
$$E^{(c)}(x) = x + \frac{c+1}{c}\big(P^{(c)}(x) - x\big) = P^{(c)}(x) + \frac{1}{c}\big(P^{(c)}(x) - x\big) = T\big(P^{(c)}(x)\big)$$

Hence, using the assumption,

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

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

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



$y - T(y)$
$\overline{x} - T(\overline{x}) + H(\overline{x}) = \frac{1}{\alpha}(z - \overline{x}) + H(\overline{x})$
Forward Step
$z = x - \alpha H(x)$
x^*
x
y
Proximal Step
$\overline{x} = P^{(\alpha)}(z)$
$-H(x)$
Slope $= -1/\alpha$

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

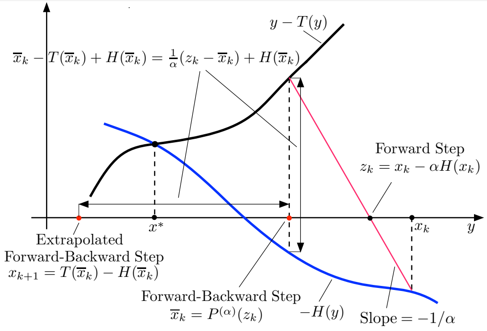
- If T is nonexpansive, and H is single-valued and strongly monotone, the F-B algorithm converges to x^* if α is sufficiently small
- For a minimization problem where H is the gradient of a strongly convex function, it becomes the proximal gradient algorithm

Extrapolation and Acceleration

Extrapolated forward-backward algorithm

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

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



$y - T(y)$
$\overline{x}_k - T(\overline{x}_k) + H(\overline{x}_k) = \frac{1}{\alpha}(z_k - \overline{x}_k) + H(\overline{x}_k)$
Forward Step
$z_k = x_k - \alpha H(x_k)$
x^*
x_k
y
Extrapolated
Forward-Backward Step
$x_{k+1} = T(\overline{x}_k) - H(\overline{x}_k)$
Forward-Backward Step
$\overline{x}_k = P^{(\alpha)}(z_k)$
$-H(y)$
Slope $= -1/\alpha$

We have

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

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

Using Simulation in the Linear Case: $T(x) = Ax + b$, $H(x) = Bx$

Oblique projection $\Pi = \Phi(\Psi'\Xi\Phi)^{-1}\Psi'\Xi$ onto a subspace $S = \{\Phi r \mid r \in \Re^s\}$

$$z_k = x_k - \alpha B x_k, \qquad \overline{x}_k = \Pi P^{(\alpha)}(z_k) \qquad \text{(Projected F-B Iteration)}$$

The projected F-B equation is the (low-dimensional linear equation)

$$r = Q^{(\lambda)} r + d^{(\lambda)}$$

where

$$Q^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{A}^{(\lambda)}(I - \alpha B)\Phi, \qquad d^{(\lambda)} = (\Psi'\Xi\Phi)^{-1}\Psi'\Xi\overline{b}^{(\lambda)}$$

$$\overline{A}^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty}\lambda^{\ell}A^{\ell}, \qquad \overline{b}^{(\lambda)} = \sum_{\ell=0}^{\infty}\lambda^{\ell+1}A^{\ell}b, \qquad \lambda = \frac{\alpha}{\alpha+1}$$

Similar to the proximal case, it can be implemented by simulation.

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)
- Multistep iteration is faster than proximal
- Cost-free acceleration of the proximal algorithm. It can be very substantial, particularly for small c
- Extrapolation formula provides new insight and justification for multistep methods
  - TD(λ) is the stochastic version of the proximal algorithm
  - TD(λ) with subspace approximation is stochastic version of the projected proximal
- Bring the use of subspace approximation and simulation into the proximal context (for linear problems)
- The ideas extend to the forward-backward algorithm and potentially other algorithmic contexts that involve fixed points and proximal operators

Thank you!