An Optimal Control Framework for First Order Methods

Robert Bassett

UC Davis Department of Mathematics

March 2015
Introduction

Problem: Analyzing first order methods is hard!

Question: Is there a unified framework for investigating these algorithms?
A Linear Dynamical System is a set of recursive linear equations

\[ \xi_{k+1} = A\xi_k + Bu_k \]

\[ y_k = C\xi_k + Du_k. \]

\( u_k \) in the input, \( y_k \) is the output, and \( \xi_k \) is the state at time \( k \).

We can connect this linear system in feedback with a nonlinearity \( \Delta \) by including

\[ u_k = \Delta(y_k) \]

in the rules above.

For our purposes, the nonlinearity has the form \( \Delta(y) = \nabla f(y) \).
Some Basics

Definition

\( S(m, L) \) is the set of continuously differentiable, strongly convex with parameter \( m \) and have Lipschitz gradients with parameter \( L \). In other words,

\[
m \| x - y \|^2 \leq (\nabla f(x) - \nabla f(y))^T (x - y) \leq L \| x - y \|^2.
\]

All the functions we consider will be in this class.
First Order Methods

Gradient Descent

\[ x_{k+1} = x_k - \alpha \nabla f(x_k) \]

Nesterov’s Accelerated Gradient Descent

\[ x_{k+1} = y_k - \alpha \nabla f(y_k) \]
\[ y_k = (1 + \beta)x_k - \beta x_{k-1} \]

Heavy-Ball Method

\[ x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \]

Each of the above methods can be written as a linear dynamical system in feedback for choice of \( A, B, C, D \).
Cast a first order method as a linear dynamical system in feedback.

Use *Integral Quadratic Constraints* to overcome non-linear feedback.

Convergence rates can be found by establishing feasiblity of a certain *Semi-Definite Program*. 
Quadratic Problems

Assume that \( f \) is a convex, quadratic function

\[
f(x) = \frac{1}{2} x^T Q x - p^T x + r.
\]

- \( \nabla f(x) = Q x - p \)
- The optimal solution is \( x^* = Q^{-1} p \).

We assume that \( D = 0 \), as is the case in GD, NAGD, and HBM.

\[
\begin{align*}
\xi_{k+1} &= A \xi_k + B u_k \\
y_k &= C \xi_k + D u_k \\
u_k &= Q y_k - p
\end{align*}
\]

\[
\xi_{k+1} = A \xi_k + B Q y_k - p \rightarrow \xi_{k+1} = (A + B Q C) \xi_k - c
\]

If \( \xi^* \) is a fixed point of the dyn. sys. then \( \xi^* = (A + B Q C) \xi^* - c \).
\[ \xi_{k+1} - \xi_* = (A + BQC)(\xi_k - \xi_*) \]

A necessary and sufficient condition for \( \xi_k \to \xi_* \) is that \( T := A + BQC \) has spectral radius strictly less than one.

**FACTS:**

- \( \rho(M) \leq \|M^K\|^{1/k} \) for all \( k \)
- \( \rho(M) = \lim_{k \to \infty} \|M^K\|^{1/k} \)

So for any \( \epsilon \) and \( k \) large enough, we can bound the convergence rate

\[ \|\xi_k - \xi_*\| = \|T^K(\xi_0 - \xi_*)\| \leq \|T^k\| \|\xi_0 - \xi_*\| \leq (\rho(T) + \epsilon)^k \|\xi_0 - \xi_*\|. \]
The following theorem connects the spectral radius to the feasibility of an SDP.

**Theorem**

\[ \rho(T) < \rho \text{ if and only if there exists a } P > 0 \text{ satisfying} \]

\[ T^T P T - \rho^2 P < 0. \]
Integral Quadratic Constraints cope with the nonlinearity of the gradient in the non-quadratic case.

Idea: Replace nonlinear component by a quadratic constraint on its inputs and outputs that is known to be satisfied by all possible instances of the component.

There are different types of IQCs

\[ \{ \text{Pointwise IQCs} \} \subset \{ \rho - \text{Hard IQCs} \} \subset \{ \text{Hard IQCs} \} \subset \{ \text{all soft IQCs} \}. \]
Main Theorem

Suppose $\phi$ satisfies a certain $\rho$-hard IQC and $0 \leq \rho \leq 1$. If

$$
\begin{bmatrix}
\hat{A}^T P \hat{T} - \rho^2 P & \hat{A}^T P \hat{B} \\
\hat{B}^T P \hat{A} & \hat{B}^T P \hat{B}
\end{bmatrix} + \lambda \begin{bmatrix}
\hat{C} & \hat{D}
\end{bmatrix}^T M \begin{bmatrix}
\hat{C} & \hat{D}
\end{bmatrix} \preceq 0
$$

is feasible for some $P \succ 0$ and $\lambda \geq 0$, then for any $\xi_0$

$$
\|\xi_k - \xi_*\| \leq \sqrt{\text{cond}(P)} \rho^k \|\xi_0 - \xi_*\|
$$

for all $k$, where $\text{cond}(P)$ is the condition number of $P$

$\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$, and $M$ all come from the IQC.
My Experiments

I confirmed the results for Gradient Descent myself.

- Pointwise IQC (suffices for the GD case)
- Used Convex in Julia, which is a frontend for solving convex problems in julia (open source).
- Solver: SCS = splitting conic solver (open source, developed by Stanford Univeristy Convex Optimization Group).
- I computed the best $\rho$ and compared it with the theoretical rate of $\frac{L-m}{L+m}$. 
Convergence Rates for G.D.

julialang.org