Control-boosting Algorithm

Phillip Yam
The Chinese University of Hong Kong
Department of Statistics

Nov 29, 2018 Big Data Challenges for Predictive Modeling of Complex Systems

Joint work with Alain Bensoussan (UTD & CityU of HK), Yiqun Li(CUHK), and Xiang Zhou(CityU of HK).

Parts of the work are supported by HKRGC: HKSAR-GRF 14301015 and HKSAR-GRF 14300717. Also acknowledge to the financial support and hospitality by Columbia University in the City of New York.

Outline

General Perception

- As a part of static and dynamic optimization, machine learning is also:
  - Model free
  - Facing at large amount and new types of data
  - Robustness fails and too sensitive to data
- Simple algorithms, big data, together with large computing facilities find wide applications in speech recognition and image processing
- Is mathematics still needed to be educated to young students?

General Perception

- As a part of static and dynamic optimization, machine learning is also:
  - Model free
  - Facing at large amount and new types of data
  - Robustness fails and too sensitive to data
- Simple algorithms, big data, together with large computing facilities find wide applications in speech recognition and image processing
- Is mathematics still needed to be educated to young students?
  - Yes, particularly much to be explored in "Intelligent dynamically controlled systems".

Outline

Supervised Learning

Supervised learning concerns approximating a function $F(x) : \mathbb{R}^d \to \mathbb{R}$ such that $y^m = F(x^m), m = 1, 2, \dots, M$.

- Non-parametric, in particular the kernel method: find a functional space $\mathcal{H}$, to which the approximation $f(x)$ of $F(x)$ belongs where

$$\gamma||f||^2 + \sum_{m=1}^{M}(f(x^m) - y^m)^2. \tag{1}$$

- Parametric method: fix a function $f(x;\theta)$, where $\theta \in \mathbb{R}^q$ and one defines it to minimize

$$\gamma|\theta|^2 + \sum_{m=1}^{M}(f(x^m;\theta) - y^m)^2. \tag{2}$$

Example: Shadow Neural Network for Binary Classification

The approximating function $f(x;\theta)$ is defined as follows:

- Let $W \in \mathbb{R}^{d\times n}$, $b \in \mathbb{R}^n$. (W, b) represents the parameter θ.
- Let σ be a real-valued function, called the **activation (transfer) function**.
- For the hidden layer with n units, let $Z := \chi(x)$, where $\chi : \mathbb{R}^d \to \mathbb{R}^n$ such that $\chi(x) = W^* x + b$, then define the vector X by $X_i := \sigma(Z_i)$, $i = 1, 2, ..., n$.
- For the output layer,

$$f(x;\theta) := \sigma_{output}(W'^* X + b'), \tag{3}$$

  where $\sigma_{output} : \mathbb{R} \to \mathbb{R}^{n_{output}}$.

The minimization in (2) is performed by a backward propagation via stochastic steepest decent method.

Outline

Deep Learning Example: AlexNet in Image Recognition (2012)

- Five CNN (Convolutional Nueral Network) layers with ReLU function
- Three FC (fully-connected tradiontal ANN) layers with ReLU function or softmax function

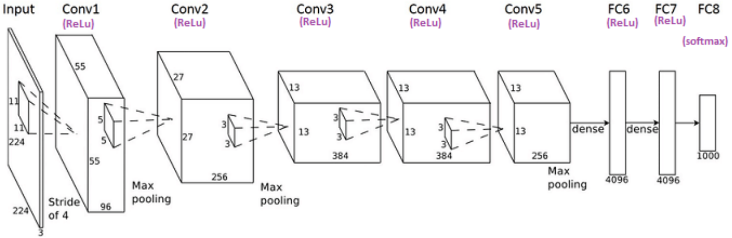

Figure: Architecture of AlexNet

It can be regarded as a regime switching dynamical systems.

Deep Learning

- Deep learning can be regarded as a generalization of the shadow neural networks with a sequence of layers.
- Generalize (3) up to $K+1$ layers as follows: for any $k = 0, \cdots K-1$,

$$Z^{k+1} = (W^{(k+1)})^* X^k + b^{(k+1)}, \;\; X_i^{k+1} = \sigma(Z_i^{k+1}), \;\; i = 1, \cdots n, \qquad (4)$$

and

$$X^0 = x, \qquad (5)$$

$$f(x, \theta) = X^K = \sigma_{output}(Z_\cdot^K). \qquad (6)$$

- By setting $\bar{f} := (\bar{f}_1, \ldots, \bar{f}_n)$ as a multi-valued composed function such that $\bar{f}_i(x) := \sigma(\chi_i(x; W, b))$, where $\chi_i(x; W, b)$ denotes the i-th component of $\chi(x; W, b)$, the above can be simplified as follows:

$$X^{k+1} = \bar{f}(X^k; W^{(k+1)}, b^{(k+1)}), \;\; k = 0, \cdots K-1, \qquad (7)$$

- **The parameter θ is the collection of $(W^{(k)}, b^{(k)})$.**

Outline

1. General Perception
2. Inter-play between Control Theory and Deep Learning
   - Supervised Learning
   - Deep Learning
   - Seeing Machine Learning as a Control Problem
3. Machine Learning Approach of Control Problems
   - General Theory
   - Dynamic Programming Approach
   - Convergence in Functional Sense

Seeing Machine Learning as a Control Problem

(LI, CHEN, TAI, E (2018, JMLR)) Recast the deep learning of supervised learning as a control problem.

- Discrete: seeing (7) as

$$X^{k+1} - X^k = f(X^k, \theta^k, k),$$

  where $\theta^k = (W^{k+1}, b^{k+1})$, and $f(x, \theta^k, k) := \bar{f}(x; \theta^k) - x$.

- Continuous: for any $x \in \mathbb{R}^d, y \in \mathbb{R}^{n_{output}}$, set $X \in \mathbb{R}^n$ satisfying the dynamics

$$\frac{dX}{dt} = f(X_t, \theta_t, t), \qquad X_0 = x. \tag{8}$$

- The approximation of $F(x)$ is then $X_T(x)$.
- The error $\Phi(X_T) := \sum \Phi_m(X_T)$ with $\Phi_m(X_T) := (y^m - X_T(x^m))^2$, recalling that $y = F(x)$ is given as they are labeled.

θ_t is now regarded as a control and wants to minimize the analogue of (1) expressed as

$$J(\theta) = \sum_{m=1}^{M} \Phi_m(X_T) + \int_0^T L(\theta_t)dt, \tag{9}$$

where $L(\theta)$ is a loss function, for instance, $\gamma|\theta|^2$.

Pontryagin Maximum Principle: *LI, CHEN, TAI, E (2018, JMLR)*

- Give a necessary condition of the optimality:
  - The optimal state and the adjoint state, $\hat{X}_t$ and $\hat{p}_t$ solve the following:

$$\frac{d\hat{X}_t}{dt} = f(\hat{X}_t, \hat{\theta}_t, t), \qquad \hat{X}_0 = x, \tag{10}$$

and

$$-\frac{d\hat{p}_t}{dt} = (D_x f)^*(\hat{X}_t, \hat{\theta}_t, t)\hat{p}_t, \quad \hat{p}_T = D_x\Phi(\hat{X}_T). \tag{11}$$

  - The optimal control $\hat{\theta}$ satisfies the optimality condition

$$\hat{\theta}_t \text{ minimizes } H(\hat{X}_t, \hat{p}_t, \theta, t), \text{ a.e. } t,$$

where

$$H(\hat{X}_t, \hat{p}_t, \theta, t) = \hat{p}_t^* f(\hat{X}_t, \theta, t) + L(\theta).$$

Approximation: *LI, CHEN, TAI, E (2018, JMLR)*

- To solve (10) and (11), one may use the method of successive approximations:
  - Given θ_t^k, define X_t^k, p_t^k by

$$\frac{d\hat{X}_t^{\,k}}{dt} = f(X_t^k, \theta_t^k, t), \qquad X_0^k = x, \tag{12}$$

    and

$$-\frac{dp_t^k}{dt} = (D_x f)^*(X_t^k, \theta_t^k, t)p_t^k, \quad p_T^k = D_x\Phi(X_T^k). \tag{13}$$

  - Then find θ_t^{k+1} to minimize

$$H(\hat{X}_t, \hat{p}_t, \theta, t), \text{ a.e. } t. \tag{14}$$

- This approximation may fail to converge. Some improvements have been proposed in *LI, CHEN, TAI, E (2018, JMLR)*.

Outline

1. General Perception
2. Inter-play between Control Theory and Deep Learning
   - Supervised Learning
   - Deep Learning
   - Seeing Machine Learning as a Control Problem
3. Machine Learning Approach of Control Problems
   - General Theory
   - Dynamic Programming Approach
   - Convergence in Functional Sense

Generic Control Problem

The state equation is a controlled dynamical system:

$$dx = g(x, a)dt, \qquad x(0) = x \tag{15}$$

The payoff is given by

$$J_x(a(.)) = \int_0^{+\infty} \exp(-\alpha t)\, f(x(t), a(t))dt \tag{16}$$

There are two approaches: Dynamic Programming and PMP. The optimal control is described by a feedback. The value function is defined by

$$u(x) = \inf_{a(\cdot)} J_x(a(\cdot)). \tag{17}$$

Standard Approaches

- (Bensoussan (2018)) Using Bellman equation:
  - Value iteration:

$$\alpha u^{k+1}(x) = \inf_a \left[g^*(x,a) D u^k(x) + f(x,a) \right].$$

  - Policy iteration:

$$\begin{aligned} \alpha u^{k+1}(x) &= g^*(x, a^k(x)) D u^{k+1}(x) + f(x, a^k(x)), \\ a^{k+1} &= \arg\inf_a \left[g^*(x,a) D u^{k+1}(x) + f(x,a) \right]. \end{aligned}$$

- (Powell (2007)) Approximate dynamic programming for the discrete case with noise, such as the Markov decision process problem, based on the Bellman equation:

$$V_t(x) = \max_{a_t} \left[C(X_t, a_t) + \gamma \mathbb{E}\left(V_{t+1}(X_{t+1}) | X_t = x \right) \right],$$

  where $X_{t+1} = S^M(Xt, a_t, W_{t+1})$ with S^M, W, C, and γ denoting the transition function of the state, the information arriving between t and $t+1$, the cost function, and the discount factor respectively; approximate value iteration and approximate policy iteration are developed correspondingly, on purpose.
- (Li, Chen, Tai, and E (2018), Rao (2009), Chernousko and Lyubushin (1982)) Methods of successive approximations with Pontryagin Maximum Principle and its variants.

Where Does Machine Learning Come in?

- 3 functions of interest: the value function $u(x)$, the optimal feedback $\hat{a}(x)$, the gradient $\lambda(x) = Du(x)$.
- Numerically, approximating the gradient of $u(x)$ by the gradient of its approximation induces much error.
- Interestingly, the gradient is a solution of a self-contained system of equations.
- The gradient has a very interesting interpretation: the shadow price in economics.

We first introduce a control boosting algorithm, and then we may think of parametric and non-parametric approximations for these functions. We shall discuss a parametric approach for the optimal feedback, and a non-parametric for the value function and its gradient.

Outline

1. General Perception
2. Inter-play between Control Theory and Deep Learning
   - Supervised Learning
   - Deep Learning
   - Seeing Machine Learning as a Control Problem
3. Machine Learning Approach of Control Problems
   - General Theory
   - Dynamic Programming Approach
   - Convergence in Functional Sense

Dynamic Programming: Equations and Algorithm

- The Hamilton-Bellman equation (HB equation for short) for the control problem (17) can be derived as follows:

$$\alpha u(x) = g^*(x, \hat{a}(x)) Du(x) + f(x, \hat{a}(x)),$$
$$\hat{a}(x) \text{ minimizes in } a, \; g^*(x, a) Du(x) + f(x, a),$$

which links the value function $u(x)$, its gradient $\lambda(x) = Du(x)$ and the optimal feedback $\hat{a}(x)$ together.

Dynamic Programming: Equations and Algorithm

- Substitute Du and D^2u by λ and $D\lambda$ respectively,

$$\alpha u(x) = g^*(x, \hat{a}(x))\lambda(x) + f(x, \hat{a}(x)),$$
$$\hat{a}(x) \text{ minimizes in } a,\ g^*(x, a)\lambda(x) + f(x, a).$$

- Differentiating in x on both sides yields a new problem:

$$\alpha\lambda(x) = D\lambda(x)g(x, \hat{a}(x)) + D_x^* g(x, \hat{a}(x))\lambda(x) + D_x f(x, \hat{a}(x)), \tag{18}$$

with

$$\hat{a}(x) \text{ minimizes in } a,\ g^*(x, a)\lambda(x) + f(x, a). \tag{19}$$

- The error arising from the gradient of the approximation of u is avoided in the optimization step (19), as we can now directly deal with $\lambda(x) = D\Phi(x)$.

Dynamic Programming: Equations and Algorithm

- Define the following iteration:
  - Given a^k,λ^k, find λ^{k+1} and a^{k+1} by solving

$$\alpha\lambda^{k+1}(x) - D\lambda^{k+1}(x)g(x, a^k(x)) = D_x^* g(x, a^k(x))\lambda^k(x) + D_x f(x, a^k(x)), \tag{20}$$

and

$$a^{k+1}(x) \text{ minimizes in } a,\ g^*(x, a)\lambda^{k+1}(x) + f(x, a) \tag{21}$$

- The equations for the components of $\lambda^{k+1}(x)$ are completely decoupled, and can be solved in parallel.
- Specifically, for each $i = 1, 2, \ldots, d$, the PDE of λ_i^{k+1} is totally independent of other components of λ^{k+1}, $\lambda_j^{k+1}, j = 1, 2, \ldots, d,\ j \neq i$, even though it still depends on all the components of λ^k.
- One possibility is to use simulation to define $\lambda^{k+1}(x)$ in a finite number of points and to use an extrapolation by using kernels from a Hilbert space.

Dynamic Programming: Equations and Algorithm

To avoid the optimization step in (21), we update the control by the following approximation, with the idea of boosting:

$$a^{k+1}(x) = a^k(x) - \rho^k(x)\left((D_a g)^*(x, a^k(x))\lambda^{k+1}(x) + D_a f(x, a^k(x))\right), \tag{22}$$

where ideally, $\rho^k(x)$ could be selected by the line search

$$\begin{aligned} \rho^k(x) &:= \arg\min_{\rho}\left\{g^*(x, w^k(x;\rho))\lambda^{k+1}(x) + f(x, w^k(x;\rho))\right\}, \\ w^k(x;\rho) &:= a^k(x) - \rho\left((D_a g)^*(x, a^k(x))\lambda^{k+1}(x) + D_a f(x, a^k(x))\right). \end{aligned} \tag{23}$$

- But is it workable?

Dynamic Programming: Equations and Algorithm

To avoid the optimization step in (21), we update the control by the following approximation, with the idea of boosting:

$$a^{k+1}(x) = a^k(x) - \rho^k(x)\left((D_a g)^*(x, a^k)\lambda^{k+1}(x) + D_a f(x, a^k)\right),$$

where ideally, $\rho^k(x)$ could be selected by the line search

$$\rho^k(x) := \arg\min_{\rho}\left\{g^*(x, w^k(x;\rho))\lambda^{k+1}(x) + f(x, w^k(x;\rho))\right\},$$
$$w^k(x;\rho) := a^k(x) - \rho\left((D_a g)^*(x, a^k)\lambda^{k+1}(x) + D_a f(x, a^k)\right).$$

- But is it workable?
- Generally, NO!
- So we choose $\rho^k(x)$ so as to ensure the convergence.

Outline

Assumptions

Suppose that $g(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ and $f(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}$ are continuously differentiable up to the second order w.r.t. both variables, and they satisfy that:

A1. There exist constants $\bar{g}, \bar{g}' > 0$ such that

$$\begin{aligned} |g(x,a)| &\le \bar{g}(1+|x|+|a|), \\ 0 \le g_x, |g_a|, |g_{xx}|, |g_{xa}| &\le \bar{g}, \text{ and } 0 \le g_{aa} \le \bar{g}'; \end{aligned} \tag{24}$$

A2. There exist constants $\bar{f}, \bar{f}' > 0$ such that

$$\begin{aligned} 0 \le f_x(x,a), |f_a(x,a)| &\le \bar{f}(1+|x|+|a|), \\ |f_{xx}|, |f_{xa}| &\le \bar{f}, \text{ and } 0 < f_{aa} \le \bar{f}'; \end{aligned} \tag{25}$$

Convergence Results

Lemma 1. Under Assumptions A1 and A2, by choosing α large and $\rho^{(k)}(x)$ small, for any k,

(i) $\lambda^{(k)}, \lambda_x^{(k)}$ and $a^{(k)}$ are of linear growth;

(ii) $a_x^{(k)}$ is uniformly bounded.

Theorem 1. Under Assumptions A1 and A2, $\{\lambda^{(k)}\}_k$ and $\{a^{(k)}\}_k$ are Cauchy sequences in L^2_ν-sense.

Convergence Results: Kernel Method for Sub-Optimum

- From Theorem 1, both $\{\lambda^{(k)}\}_k$ and $\{a^{(k)}\}_k$ generated by the control-boosting Algorithm 1 converge in L^2_ν-sense, i.e. $\exists\ \lambda^*$ and a^* such that $\lim_{k\to\infty} \lambda^{(k)} = \lambda^*$ and $\lim_{k\to\infty} a^{(k)} = a^*$ respectively.
- But, generally speaking, is (λ^*, a^*) our targeted limit we want?

Convergence Results: Kernel Method for Sub-Optimum

- From Theorem 1, both $\{\lambda^{(k)}\}_k$ and $\{a^{(k)}\}_k$ generated by the control-boosting Algorithm 1 converge in L^2_ν-sense, i.e. $\exists\ \lambda^*$ and a^* such that $\lim_{k\to\infty} \lambda^{(k)} = \lambda^*$ and $\lim_{k\to\infty} a^{(k)} = a^*$ respectively.
- But, generally speaking, is (λ^*, a^*) our targeted limit we want?
- Essentially YES! But how to solve it in practical sense? We propose a parametric approach for a sub-optimum.

Convergence Results: Kernel Method for Sub-Optimum

Let S be a set of real-valued functions, and define a linear spanning function space as

$$\mathrm{span}(S) = \left\{ \sum_{j=1}^{N} \omega^j \varphi^j : \varphi^j \in S, \omega^j \in \mathbb{R}, N \in Z^+ \right\}. \tag{26}$$

- We want to find a function $\Phi \in \mathrm{span}(S)$ that approximately solves the systems:

$$\alpha\lambda(x) = D\lambda(x)g(x, \Phi(x)) + D_x^* g(x, \Phi(x))\lambda(x) + D_x f(x, \Phi(x)), \tag{27}$$

with

$$\Phi = \arg \inf_{\varphi \in \mathrm{span}(S)} \mathcal{M}\left(g^*(\cdot, \varphi(\cdot))\lambda(\cdot) + f(\cdot, \varphi(\cdot))\right), \tag{28}$$

where $\mathcal{M}$ is a real-valued function on $\mathrm{span}(S)$ representing a loss/penalty function, for instance, the expectation of the square of certain deviation in decision theory.

Convergence Results: Kernel Method for Sub-Optimum

Consider the iteration:

- Given Φ^k,λ^k, find λ^{k+1} by solving

$$\alpha\lambda^{k+1}(x) - D\lambda^{k+1}(x)g(x,\Phi^k(x)) = D_x^* g(x,\Phi^k(x))\lambda^k(x) + D_x f(x,\Phi^k(x)). \tag{29}$$

- Select a closed subset $\Lambda^k \subset \mathbb{R}$ such that $0 \in \Lambda^k$ and $\Lambda^k = -\Lambda^k$.
- Find $\bar{\rho}^k \in \Lambda^k$ and $\bar{\varphi}^k \in S$ to approximately minimize:

$$(\rho^k, \varphi^k) \to \mathcal{M}(\Phi^k + \rho^k\varphi^k); \tag{30}$$

 any standard greedy algorithm ([4],[7]).

- Update Φ^k by

$$\Phi^{k+1} = \Phi^k + \bar{\rho}^k\bar{\varphi}^k. \tag{31}$$

Theorem 2. Under Assumptions A1 and A2, the above iteration produces a convergence sequence $\{(\lambda^k, \Phi^k)\}_k$ such that its limit solves the systems (27), (28), which is also a sub-optimum for the original problem with span(S) as the admissible set of controls.

Example: Linear-Quadratic Case

Consider a deterministic control problem of Linear-Quadratic case:

- The state $x \in \mathbb{R}^d$ satisfies that

$$dx(t) = (Ax(t) + Ba(t))dt, \qquad x(0) = x. \tag{32}$$

- The payoff is given by

$$J_x(a(\cdot)) = E\int_{-\infty}^{\infty} \exp(-\alpha t)\frac{1}{2}\left(x^*(t)Mx(t) + a^*(t)Na(t)\right)dt. \tag{33}$$

- We aim to solve the minimization problem

$$u(x) := \inf_{a(\cdot)} J_x(a(\cdot)). \tag{34}$$

Accordingly, the specific control-boosting algorithm for the LQ case as:

Generic Algorithm	LQ Algorithm
$\alpha\lambda^{k+1}(x) - D\lambda^{k+1}(x)g(x, a^k(x))$ $= D_x^* g(x, a^k(x))\lambda^k(x) + D_x f(x, a^k(x))$	$\alpha\lambda^{k+1}(x) - D\lambda^{k+1}(x)(Ax + Ba^k(x))$ $= A\lambda^k(x) + Mx$
$a^{k+1}(x) = a^k(x)$ $\rho^k(x)\left((D_a g)^*(x, a^k(x))\lambda^{k+1}(x) + D_a f(x, a^k(x))\right)$	$a^{k+1}(x) = a^k(x)$ $\rho^k(x)\left(B^*\lambda^{k+1}(x) + Na^k(x)\right)$

Example: Linear-Quadratic Case

Particularly, from (21), by the first order principle, as $a^{(k+1)}$ can be explicitly solved as $a^{(k+1)}(x) = -N^{-1}B^*\lambda^{(k+1)}(x)$, so that one can skip the "boosting" step, and (20) immediately yields

$$\alpha\lambda^{(k+1)}(x) = Mx + D\lambda^{(k+1)}(x)(Ax - BN^{-1}B^*\lambda^{(k)}(x)) + A^*\lambda^{(k)}(x), \qquad (35)$$

and this has the solution

$$\lambda^{(k+1)}(x) = P^{(k+1)}x, \qquad (36)$$

where $P^{(k+1)}$ is the solution of the algebraic Riccati equation

$$\alpha P^{(k+1)} = M + A^*P^{(k)} + P^{(k+1)}(A - BN^{-1}B^*P^{(k)}). \qquad (37)$$

Proposition 1. Assume that

$$\alpha > 2\|A\| + 2\sqrt{\|M\|\|BN^{-1}B^*\|}, \qquad (38)$$

and $\|P^0\| \le \omega$ in the interval such that

$$\frac{\alpha - \|A\| - \sqrt{(\alpha - 2\|A\|)^2 - 4\|M\|\|BN^{-1}B^*\|}}{2\|BN^{-1}B^*\|} < \omega < \frac{\alpha - 2\|A\|}{2\|BN^{-1}B^*\|},$$

then $\|P^{(k)} - P\| \to 0$, as $k \to \infty$, with P unique solution of

$$\alpha P = M + A^*P + P(A - BN^{-1}B^*P),$$

with $\|P\| \le \omega$.

Numerical Example: Linear Quadratic Case

$$A = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ -35 & -28 & -9 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, M = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & -1 \end{bmatrix}, N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \alpha = 50.$$

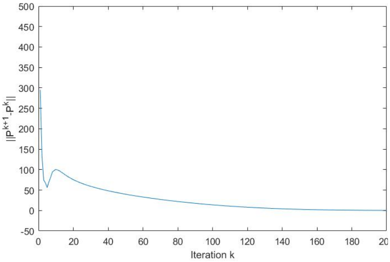

Figure: $\|P^{(k+1)} - P^{(k)}\|$

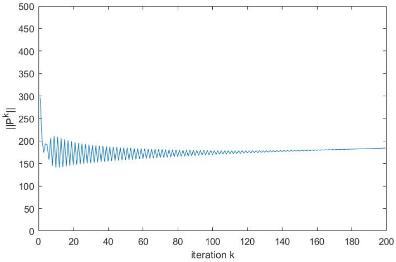

Figure: $\|P^{(k)}\|$

Thank you!

In memory of my father (1948 - 2018)

References

Bensoussan, A.
Estimation and Control of Dynamical Systems,
Springer Vol.18 (2018).

Bouvrie, J. and Hamzi, B.
Kernel methods for the approximation of nonlinear systems,
SIAM Journal on Control and Optimization (2017), **55**: 2460–2492.

Chernousko, F.L. and Lyubushin, A.A.
Method of successive approximations for solution of optimal control problems,
Optimal Control Applications and Methods (1982), **3**: 101–114.

Friedman, J.H.
Greedy function approximation: a gradient boosting machine,
Annals of statistics (2001), 1189–1232.

Friedman, J.H., Hastie, T. and Tibshirani, R.
The elements of statistical learning,
Springer series in statistics New York, NY, USA Vol.1 (2001).

Li, Q., Chen, L., Tai, C. and E, W.
Maximum principle based algorithms for deep learning,
The Journal of Machine Learning Research (2017), **18**: 5998–6026.

Lu, T., Neittaanmaki, P. and Tai, X-C.
A parallel splitting-up method for partial differential equations and its applications to Navier-Stokes equations,
ESAIM: Mathematical Modelling and Numerical Analysis (1992), **26**: 673–708.

References

Mason, L., Baxter, J. Bartlett, P. L., Frean, M. and others.
Functional gradient techniques for combining hypotheses,
Advances in Neural Information Processing Systems (1999), 221–246.

Powell, W.B.
Approximate Dynamic Programming: Solving the curses of dimensionality,
John Wiley & Sons Vol.703 (2007).

Powell, W.B.
What you should know about approximate dynamic programming,
Naval Research Logistics (NRL) (2009), **56**: 239–249.

Rao, A.V.
A survey of numerical methods for optimal control,
Advances in the Astronautical Sciences (2009), **135**: 497–528.

Smola, A.J., Vidal, R. and Vishwanathan, S,V.N.
Kernels and dynamical systems,
Automatica (2004).

E, Weinan and Yu, Bing
The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems,
Communications in Mathematics and Statistics (2018), **6**: 1–12.

Zhang, T. and Yu, B.
On the convergence of boosting procedures,
Proceedings of the 20th International Conference on Machine Learning (ICML-03) (2003), 904–911.