On the Interplay between Optimization and Machine Learning, and the Role of Sensitivities

Dinesh Krishnamoorthy

dinesh.krishnamoorthy@ntnu.no

09 Nov 2020

Guest Talk at DAIS Lab, University of British Columbia

Introduction

Machine learning for Optimization

- Complex decision-making problems - formulated as optimization problem
- Function approximators to handle computational complexity

Introduction

Machine learning for Optimization

- Complex decision-making problems - formulated as optimization problem
- Function approximators to handle computational complexity

Optimization for Machine learning

- Optimization at the heart of machine learning
- Find best parameters that maximize some utility function

Introduction

Machine learning for Optimization

- Complex decision-making problems - formulated as optimization problem
- Function approximators to handle computational complexity

Optimization for Machine learning

- Optimization at the heart of machine learning
- Find best parameters that maximize some utility function

Parametric sensitivity

- What happens to a solution of a problem when there are changes in the data

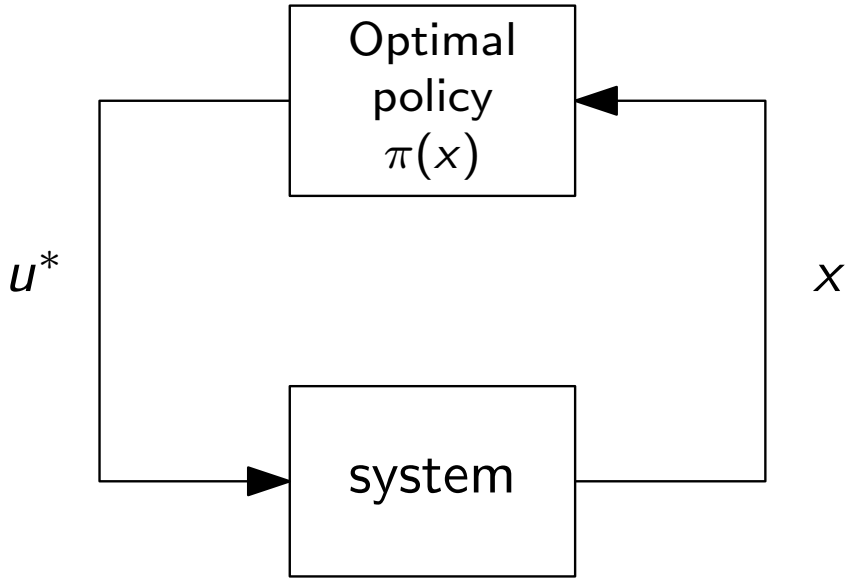
<u>This talk</u>: What role does sensitivity play in the interplay between machine learning and optimization?

Table of Contents

Machine learning for optimization

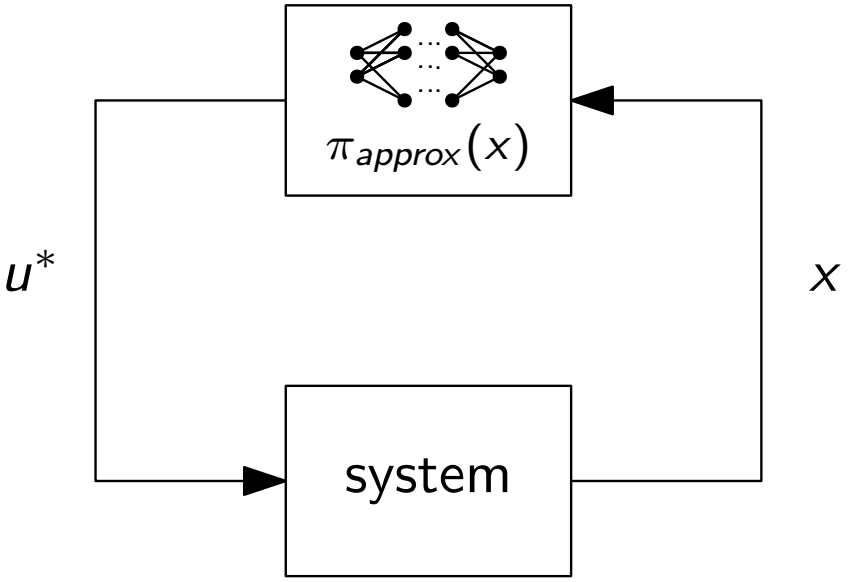
Optimization for Machine Learning

ML for optimization: Policy Approximation



E.g. Model Predictive Control – implicit policy

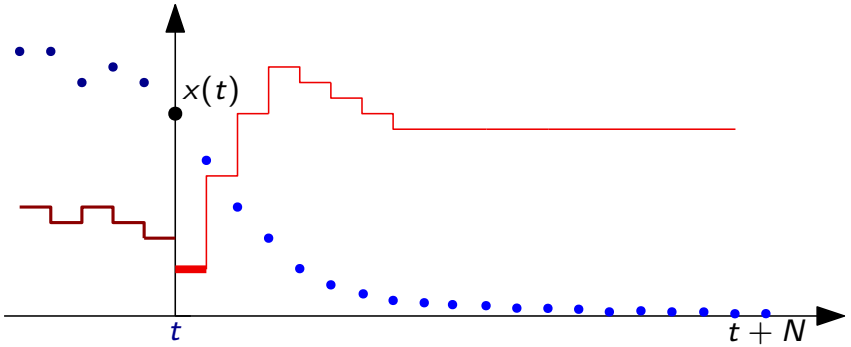
ML for optimization: Policy Approximation



E.g. Model Predictive Control – implicit policy

Model predictive control

$$
\begin{aligned}
V_N(x(t)) = \min_{x(\cdot|t),u(\cdot|t)} \quad & \sum_{k=0}^{N-1} \ell(x(k|t), u(k|t)) \\
\text{s.t. } & x(k+1|t) = f(x(k|t), u(k|t)) \\
& x(k|t) \in \mathcal{X}, \quad u(k|t) \in \mathcal{U} \\
& x(N|t) \in \mathcal{X}_f \\
& x(0|t) = x(t)
\end{aligned}
$$



Repeatedly solving the receding horizon optimization problem and implementing the first control input implicitly give the policy

$$u^*(t) = \pi_{mpc}(x(t))$$

Expert Assisted Supervised Learning

- Comparison Learning using examples from human experts (Tesauro, 1989)
- Expert can also be an optimization problem or a control policy (Bertsekas, 2019)

[1] Parisini & Zoppoli (1995) *Automatica*, Chakrabarty et al. (2017) *IEEE TAC*, Karg & Lucia (2018), *IFAC NMPC*, Chen et al. (2018) *ACC*, Paulson & Mesbah (2020) *IEEE L-CSS*, Hertneck et al. (2018) *IEEE L-CSS*, Drgona et al. (2018) *Applied Energy*, Ceskö et al. (2015) *IEEE-TCST*, Zhang et al. (2019) *ACC*, Spielberg et al. (2019) *AIChE J.*

Expert Assisted Supervised Learning

- Comparison Learning using examples from human experts (Tesauro, 1989)
- Expert can also be an optimization problem or a control policy (Bertsekas, 2019)

Approximate Explicit MPC [1]

- Sample $\mathcal{X}$ offline to generate N_s discrete states $\{x_i\}_{i=1}^{N_s}$
- Solve the optimization problem for each state x_i to obtain $u_i^* = \pi_{mpc}(x_i)$
- Training data set $\mathcal{D} := \{x_i, u_i^*\}_{i=1}^{N_s}$
- Train a parametric function $\pi_{approx}(x_i; \theta)$ such that

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\pi_{\text{approx}}(x_i; \theta) - u_i^*\|^2$$

[1] Parisini & Zoppoli (1995) *Automatica*, Chakrabarty et al. (2017) *IEEE TAC*, Karg & Lucia (2018), *IFAC NMPC*, Chen et al. (2018) *ACC*, Paulson & Mesbah (2020) *IEEE L-CSS*, Hertneck et al. (2018) *IEEE L-CSS*, Drgona et al. (2018) *Applied Energy*, Ceskö et al. (2015) *IEEE-TCST*, Zhang et al. (2019) *ACC*, Spielberg et al. (2019) *AIChE J.*

Training is expensive

- Need training data set that covers the state space $\mathcal{X}$
- No. of training samples increases exponentially for higher dimensional systems
- Any changes in the system model, the training data must be generated all over again

How can we address this issue?

Data Augmentation

A strategy to artificially increase the number of training samples using computationally inexpensive transformations, e.g. photometric or geometric transformations

Ho et al. *ICML* 2019

nice video illustrating down sampling

Parametric Sensitivity

Parametric NLP

$$
\begin{aligned}
V_N(p) = \min_{\mathbf{w}} \quad & J(\mathbf{w}, p) \\
\text{s.t.} \quad & c(\mathbf{w}, p) = 0 \\
& g(\mathbf{w}, p) \leq 0
\end{aligned}
$$

Assume:

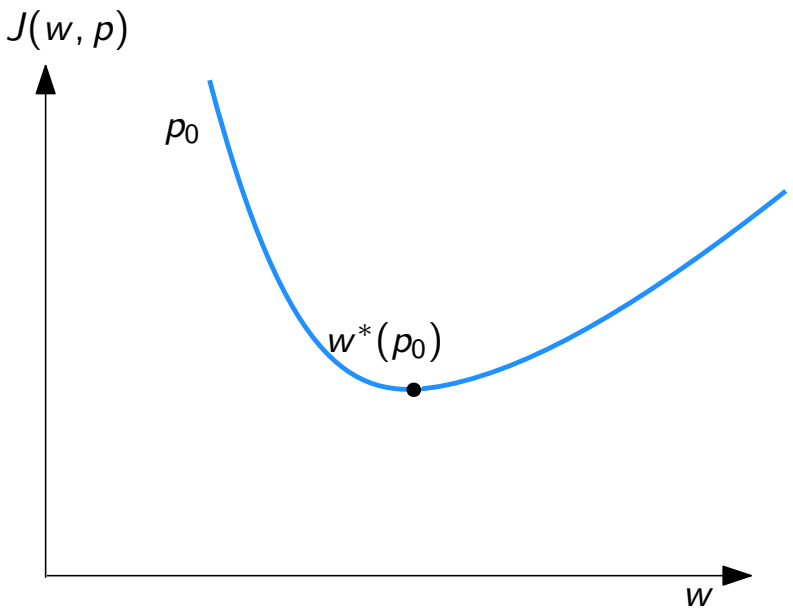
- Constraint qualification, e.g. LICQ
- Second order sufficient condition
- Strict Complimentarity

Fiacco (1976) *Math. Prog.*

Parametric Sensitivity

Parametric Sensitivity

- $\mathbf{s}^*(p_0)$ is a unique local minimizer of $\mathcal{P}(p_0)$.
- For parametric perturbations Δp in the neighborhood of p_0, $\exists$ a unique, continuous, and differentiable vector function $\mathbf{s}^*(p_0 + \Delta p)$ which is a KKT point satisfying LICQ and SSOSC for $\mathcal{P}(p_0 + \Delta p)$.

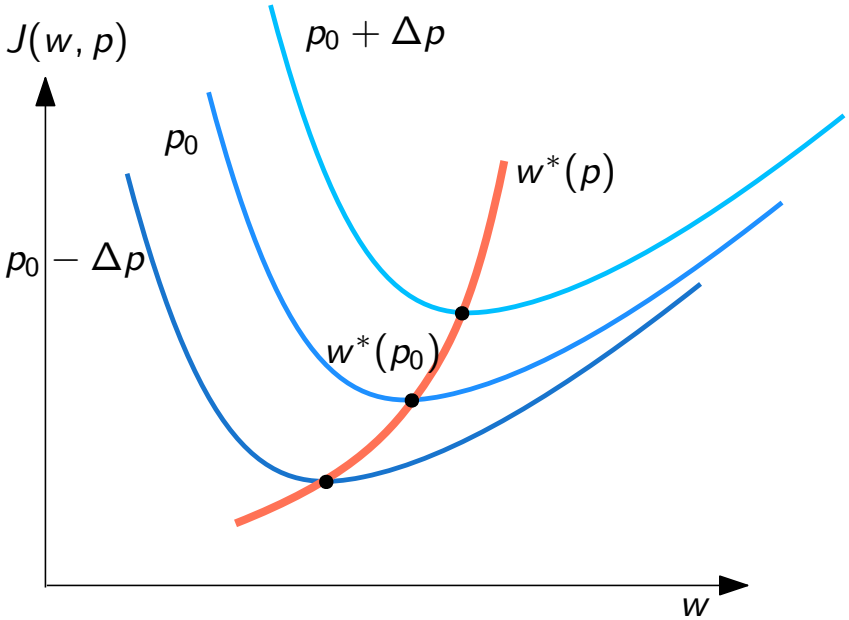


Fiacco (1976) *Math. Prog.*

Parametric Sensitivity

Parametric Sensitivity

- $\mathbf{s}^*(p_0)$ is a unique local minimizer of $\mathcal{P}(p_0)$.
- For parametric perturbations Δp in the neighborhood of p_0, $\exists$ a unique, continuous, and differentiable vector function $\mathbf{s}^*(p_0 + \Delta p)$ which is a KKT point satisfying LICQ and SSOSC for $\mathcal{P}(p_0 + \Delta p)$.

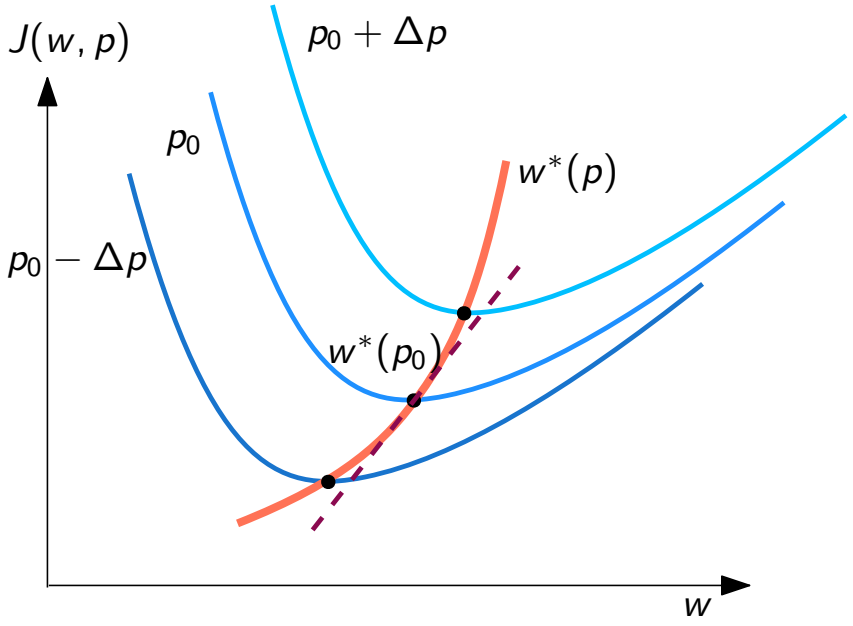


Fiacco (1976) *Math. Prog.*

Parametric Sensitivity

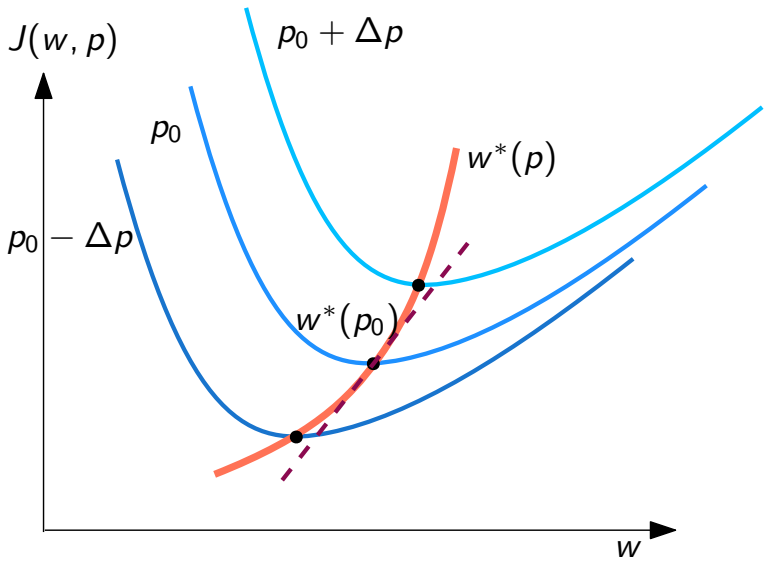
Parametric Sensitivity

- $\mathbf{s}^*(p_0)$ is a unique local minimizer of $\mathcal{P}(p_0)$.
- For parametric perturbations Δp in the neighborhood of p_0, $\exists$ a unique, continuous, and differentiable vector function $\mathbf{s}^*(p_0 + \Delta p)$ which is a KKT point satisfying LICQ and SSOSC for $\mathcal{P}(p_0 + \Delta p)$.



Fiacco (1976) *Math. Prog.*

Data Augmentation for Policy Approximation

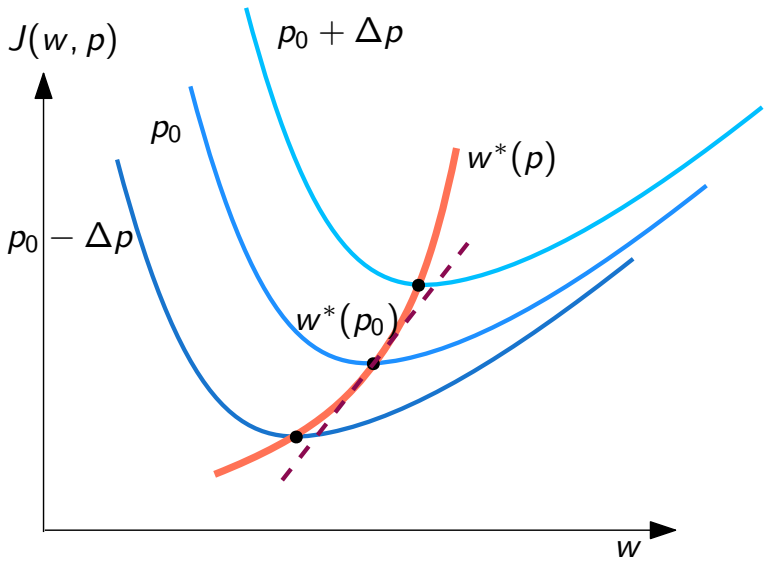


Tangential Approximator

$$\hat{\mathbf{s}}^*(p_0 + \Delta p) = \mathbf{s}^*(p) + \Delta \mathbf{s}^*$$

$$\mathcal{M} \Delta \mathbf{s}^* = -\mathcal{N} \Delta p$$

Data Augmentation for Policy Approximation



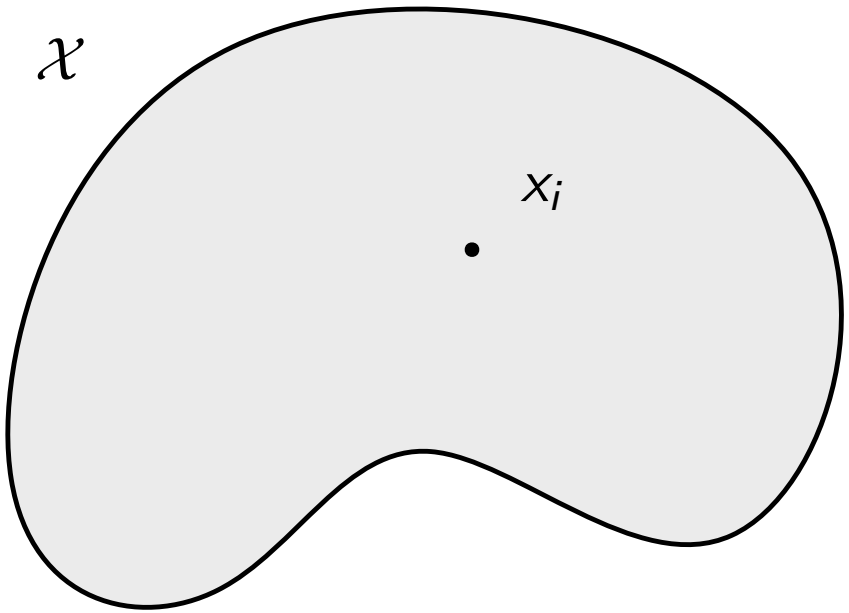
Tangential Approximator

$$\hat{\mathbf{s}}^*(p_0 + \Delta p) = \mathbf{s}^*(p) + \Delta \mathbf{s}^*$$

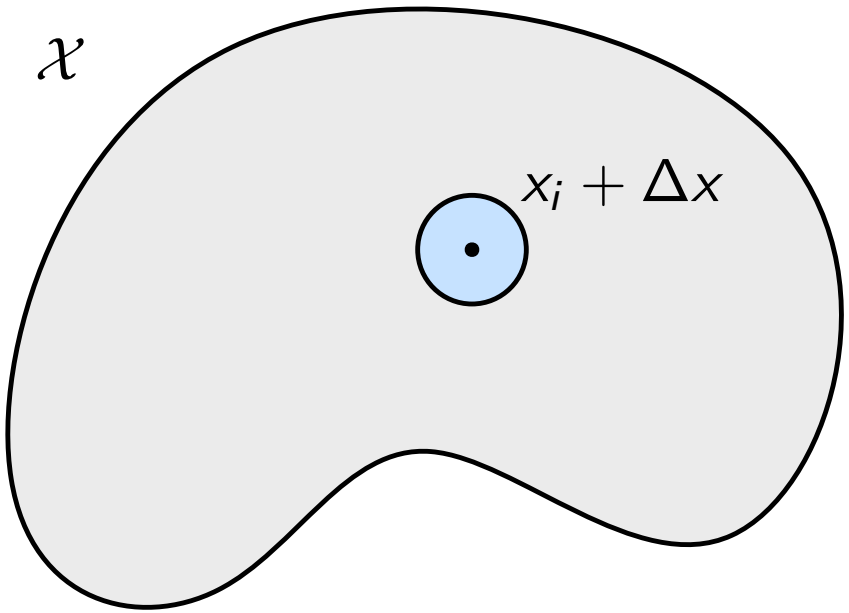
$$\mathcal{M} \Delta \mathbf{s}^* = -\mathcal{N} \Delta p$$

$$\mathcal{M} := \begin{bmatrix} \nabla^2_{\mathbf{ww}} \mathcal{L}(\mathbf{s}^*(p_0)) & \nabla_{\mathbf{w}} c(\mathbf{w}^*(p_0)) & \nabla_{\mathbf{w}} g_{\mathbb{A}}(\mathbf{w}^*(p_0)) \\ \nabla_{\mathbf{w}} c(\mathbf{w}^*(p_0))^\mathsf{T} & 0 & 0 \\ \nabla_{\mathbf{w}} g_{\mathbb{A}}(\mathbf{w}^*(p_0))^\mathsf{T} & 0 & 0 \end{bmatrix} \qquad \mathcal{N} := \begin{bmatrix} \nabla_{\mathbf{w}p} \mathcal{L}(\mathbf{s}^*(p_0)) \\ \nabla_p c(\mathbf{w}^*(p_0))^\mathsf{T} \\ \nabla_p g_{\mathbb{A}}(\mathbf{w}^*(p_0))^\mathsf{T} \end{bmatrix}$$

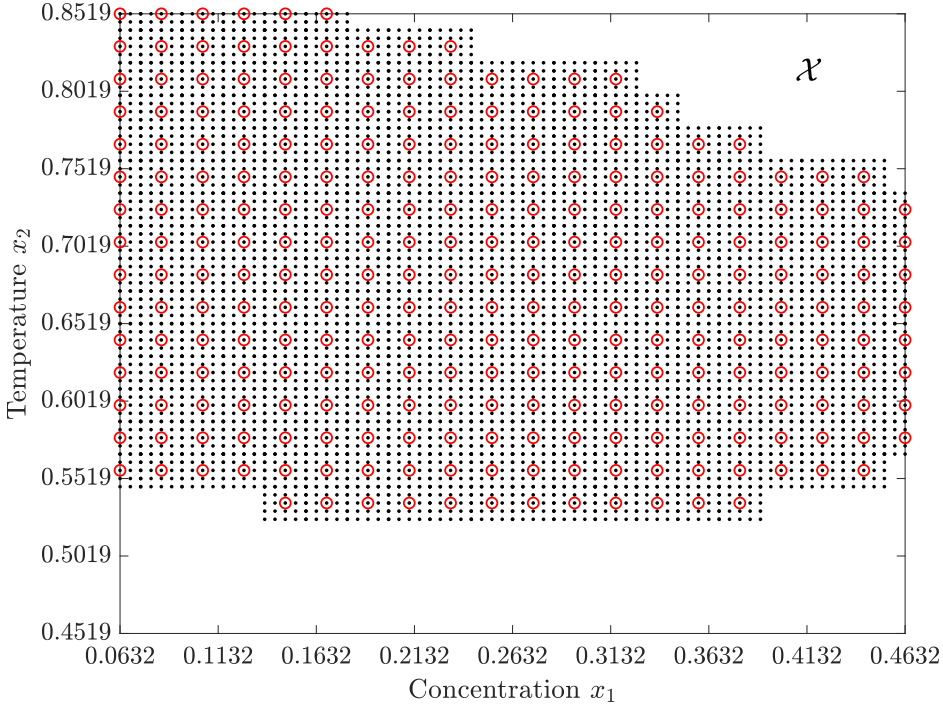
Sensitivity-based Data Augmentation for Policy Approximation



Sensitivity-based Data Augmentation for Policy Approximation



Illustrative Example - CSTR



Hertneck et al. (2018) *IEEE-LCSS*

Illustrative Example - CSTR

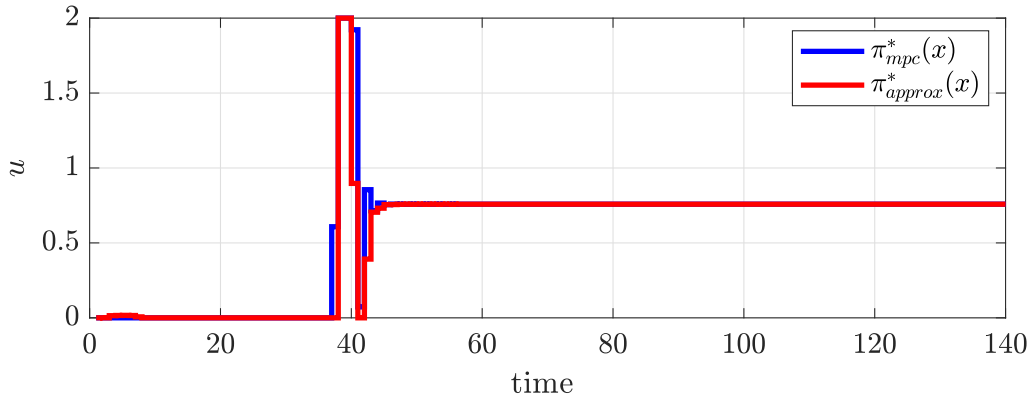


Table: Example A: CPU time in [s] for generating the training samples.

	min	avg	max	# of samples
Full NLP	0.2837	0.5980	2.7697	268
Sensitivity update	0.0038	0.0055	0.0222	6700

Krishnamoorthy (2020) *arXiv:2009.07398*

Illustrative Example - Building Climate Control

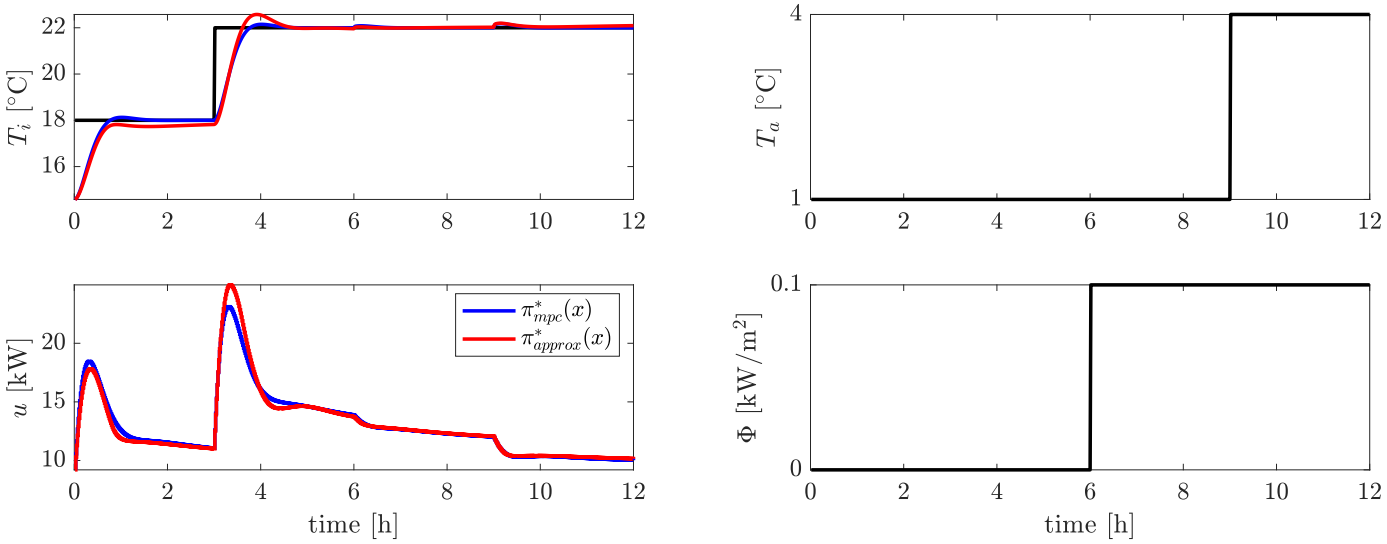


Table: Example B: CPU time in [s] for generating the training samples.

	min	avg	max	# of samples
Full NLP	0.3925	0.55	1.0683	330
Sensitivity update	0.029	0.0344	0.1034	6600

Krishnamoorthy (2020) *arXiv:2009.07398*

Table of Contents

Machine learning for optimization

Optimization for Machine Learning

Model-fitting and statistical learning

- Given a labeled training data set $\mathcal{D} := \{(u_j, y_j)\}_{j=1}^{M}$
- we want to fit a parametric architecture $y = f(u, x)$

Learning problem

$$x^* = \arg\min_{x \in \mathbb{R}^n} J(x, \mathcal{D}) + r(x) \tag{1}$$

- Loss function $J(x, \mathcal{D}) := \sum_{j=1}^{M} \ell(f(u_j, x) - y_j)^2$
- Regularization function $r : \mathbb{R}^n \to \mathbb{R}$

Model-fitting and statistical learning

- Given a labeled training data set $\mathcal{D} := \{(u_j, y_j)\}_{j=1}^{M}$
- we want to fit a parametric architecture $y = f(u, x)$

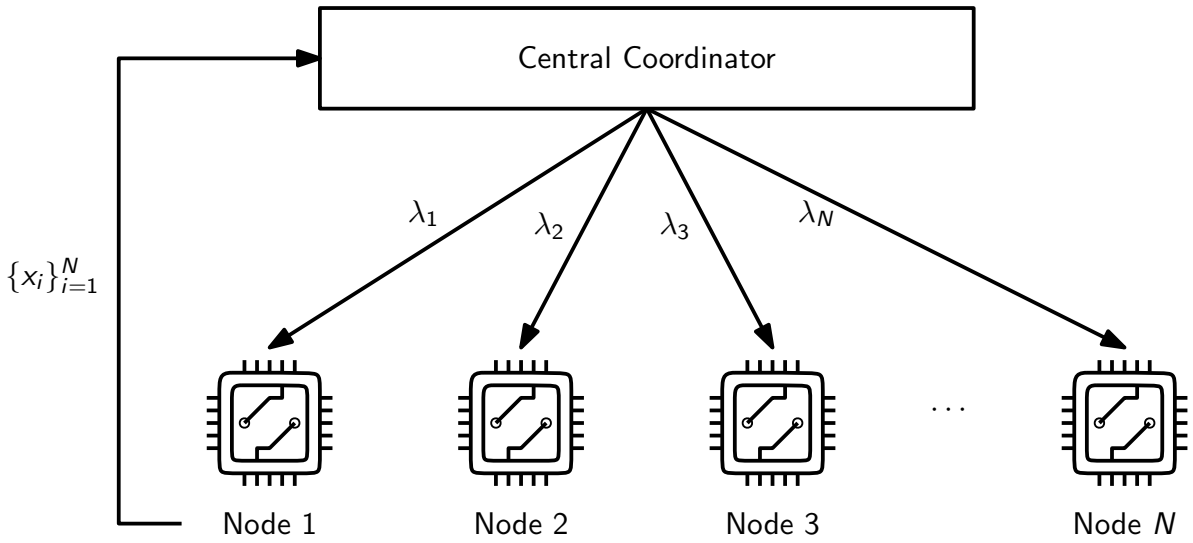
Learning problem

$$x^* = \arg\min_{x \in \mathbb{R}^n} J(x, \mathcal{D}) + r(x) \tag{1}$$

- Loss function $J(x, \mathcal{D}) := \sum_{j=1}^{M} \ell(f(u_j, x) - y_j)^2$
- Regularization function $r : \mathbb{R}^n \to \mathbb{R}$

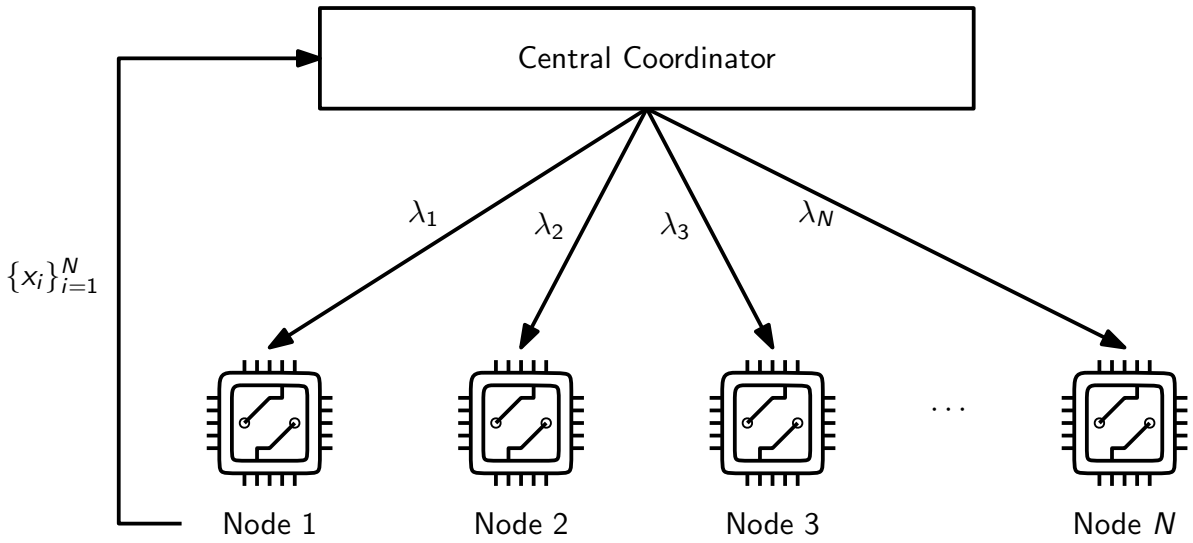
- Large-scale optimization
- Distributed optimization: Decompose into smaller problems

Federated Learning



Boyd et al. (2009) Foundations and Trend in Machine Learning

Federated Learning



Distributed Model Fitting

- Decompose across dataset $\mathcal{D}$ - Consensus problem
- Decompose across features u - Optimal exchange problem

Solved using Alternating Directions Method of Multipliers (ADMM)

Boyd et al. (2009) Foundations and Trend in Machine Learning

Federated Learning

Split the training data set into N chunks

$$\mathcal{D}_i := \{(u_j, y_j)\}_{j=1}^{M_i} \qquad M = \sum_{i=1}^{N} M_i \quad \text{and} \quad \mathcal{D} = \bigcup_{i=1}^{N} \mathcal{D}_i$$

Learning problem

$$\min_x \sum_{i=1}^{N} J(x, \mathcal{D}_i) + r(x)$$

Distributed Learning problem

Decompose the problem using a local copy $x_i \in \mathbb{R}^n$ of x

$$\begin{aligned} \min_{x_1,\ldots,x_N} \sum_{i=1}^{N} J_i(x_i, \mathcal{D}_i) + r(x_0) \\ \text{s.t.} \quad x_i = x_0 \quad \forall i = 1, \ldots, N \end{aligned}$$

Federated Learning

Augmented Lagrangian function:

$$\min_{x_1,\ldots,x_N} \mathcal{L}(\{x_i\}, x_0; \{\lambda_i\}) := \sum_{i=1}^{N} J(x_i, \mathcal{D}_i) + r(x_0) + \sum_{i=1}^{N} \lambda_i^{\mathsf{T}}(x_i - x_0) + \sum_{i=1}^{N} \frac{\rho}{2}\|x_i - x_0\|^2$$

Federated Learning

Augmented Lagrangian function:

$$\min_{x_1,\ldots,x_N} \mathcal{L}(\{x_i\}, x_0; \{\lambda_i\}) := \sum_{i=1}^{N} J(x_i, \mathcal{D}_i) + r(x_0) + \sum_{i=1}^{N} \lambda_i^{\mathsf{T}}(x_i - x_0) + \sum_{i=1}^{N} \frac{\rho}{2}\|x_i - x_0\|^2$$

ADMM iterations

$$x_i^{k+1} = \arg\min_{x_i} \mathcal{L}_i(x_i, x_0^k; \lambda_i^k) := J_i(x_i, \mathcal{D}_i) + \lambda_i^{k\mathsf{T}}(x_i - x_0^k) + \frac{\rho}{2}\left\|x_i - x_0^k\right\|^2 \qquad \forall i = 1, \ldots, N$$

$$x_0^{k+1} = \arg\min_{x_0} r(x_0) + \sum_{i=1}^{N} \lambda_i^{k\mathsf{T}}(x_i^{k+1} - x_0) + \sum_{i=1}^{N} \frac{\rho}{2}\left\|x_i^{k+1} - x_0\right\|^2$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho(x_i^{k+1} - x_0^{k+1}) \qquad \forall i = 1, \ldots, N$$

Sensitivity-assisted ADMM (sADMM)

Each subproblem can be written as

$$x_i^{k+1}(p^k) = \arg\min_{x_i} \mathcal{L}_i(x_i, p^k)$$

where $p^k = [x_0^k, \lambda_i^k]^\mathsf{T}$

For iterations $k > 1$

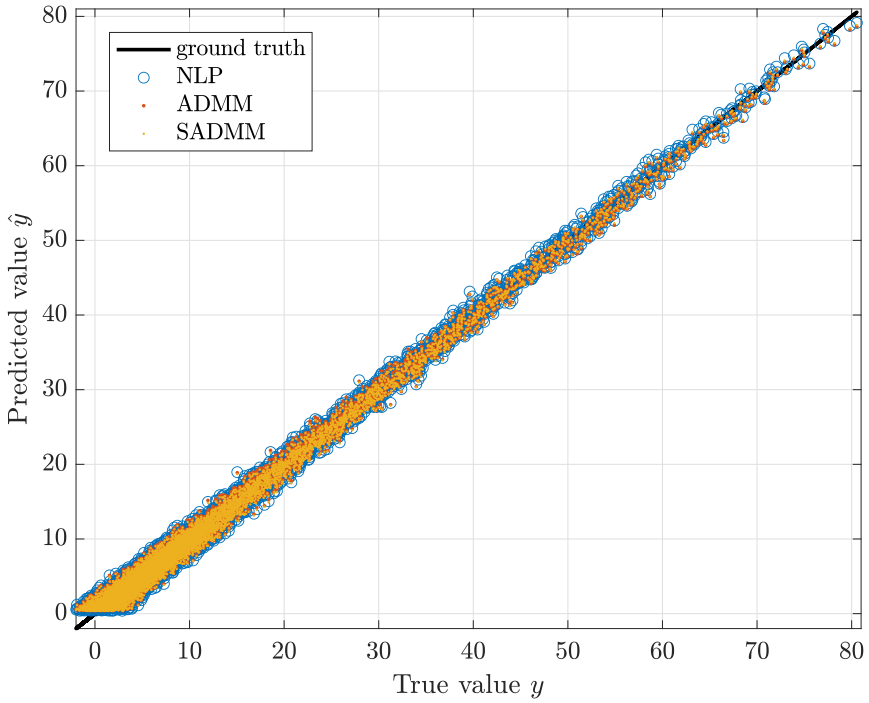
$$\underbrace{\nabla^2_{x_i,x_i}\mathcal{L}_i(x_i^*(p^k))}_{=\mathcal{M}}(x_i^*(p^{k+1}) - x_i^*(p^k)) \approx -\underbrace{\nabla^2_{x_i,p}\mathcal{L}_i(x_i^*(p^k))}_{\mathcal{N}}(p^{k+1} - p^k)$$

$$\Rightarrow \qquad \tilde{x}_i^*(p^{k+1}) = x_i^*(p^k) - \mathcal{M}^{-1}\mathcal{N}(p^{k+1} - p^k)$$

Krishnamoorthy & Kungurtsev (2020) *arXiv:2009.05845*

Example 1: Inverse Model fitting

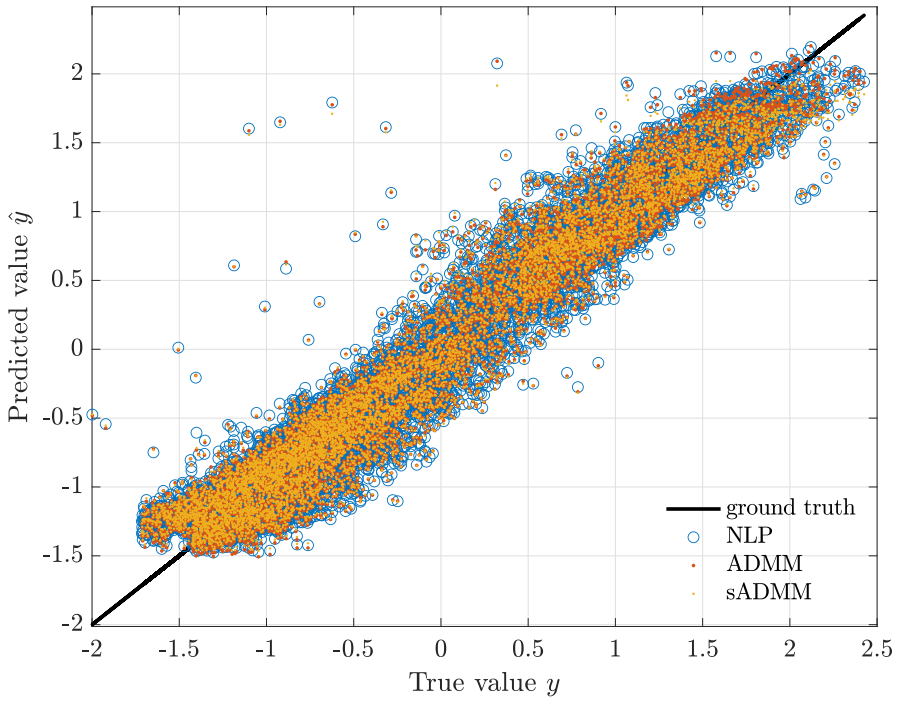
Fit $f(u,x) = x_1 e^{x_2 u_1} + \frac{0.01}{x_4}((x_3 - u_1)^2) + x_5(u_2 - (x_6 u_1)^2)^2$ using 10201 noisy samples of data using 4 server nodes.



	NLP	ADMM	sADMM
MSE	1.005	1.022	1.0318
R^2	0.9945	0.9944	0.9944
CPU/iteration		0.045s	0.5E-4s
cumulative CPU time 21 iter		0.74s	0.05s

Example 2: Multi-Layer Perceptron

Predict net hourly electrical energy output from a combined cycle power plant using 9568 noisy samples of data using 4 server nodes.



	NLP	ADMM	sADMM
MSE	0.0606	0.0614	0.0617
R^2	0.9394	0.9386	0.9383
CPU/iteration		3.5s	1.44E-4s
cumulative CPU time 30 iter		104.4s	5.05s

Data Source: Tufekci, P.; and Kaya, H. 2014. Combined cycle power plant dataset [UCI Machine Learning Repository].

Summary

- Machine Learning for optimization

  Use sensitivities to improve training data generation

Krishnamoorthy (2020) *arXiv:2009.07398*
Krishnamoorthy & Kungurtsev (2020) *arXiv:2009.05845*

Summary

- Machine Learning for optimization

  Use sensitivities to improve training data generation

- Distributed Optimization for Machine learning

  Use sensitivities to cheaply evaluate optimal solution from one iteration to another

---

Krishnamoorthy (2020) *arXiv:2009.07398*
Krishnamoorthy & Kungurtsev (2020) *arXiv:2009.05845*