Quasi-Newton methods for solving nonsmooth optimization problems in learning and control

Adeyemi Damilare Adeoye

Supervisor: Prof. Alberto Bemporad

PhD Thesis Defence
IMT School for Advanced Studies Lucca, Italy

Jury: Prof. Mikael Johansson (KTH Royal Institute of Technology, Stockholm, Sweden), Prof. Panagiotis Patrinos (KU Leuven, Leuven, Belgium), Dr. Puya Latafat (IMT School for Advanced Studies Lucca, Italy)

June 3, 2025

Outline

Introduction: Neural networks for function approximation

RNN for Markovian state-space dynamics

DCRNN: An RNN for Markovian state-space approximation

iSQPRL: An inexact SQP approach for recurrent learning

Recurrent-Control Neural Network

Generalized Gauss-Newton with self-concordant regularization

Conclusions and future directions

Some challenges in modeling, optimizing, and controlling the physical entities of a nonlinear dynamical system from first principles

As an example, some questions that concern arterial blood flow with no known straightforward answers:

- is a Newtonian or non-Newtonian model more appropriate?
- is there any universally accepted model for blood flow?
- how to choose parameters for any given model?
- efficient numerical methods for solving the governing equations under any given circumstance?
- are there efficient (numerical) methods to handle the high-complexity of realistic geometries?

Some challenges in modeling, optimizing, and controlling the physical entities of a nonlinear dynamical system from first principles

Notable remarks:

- **known fact:** blood is a non-Newtonian fluid
  (its viscosity changes with the wall shear rate)
- no known non-Newtonian model accurately describes the rheological behaviour of blood [1]
- governing equations computationally intensive to solve
  (a new problem needs to be solved for each new configuration)
- studies suggest that a Newtonian model can be a good approximation (especially for large arteries) [2][1]

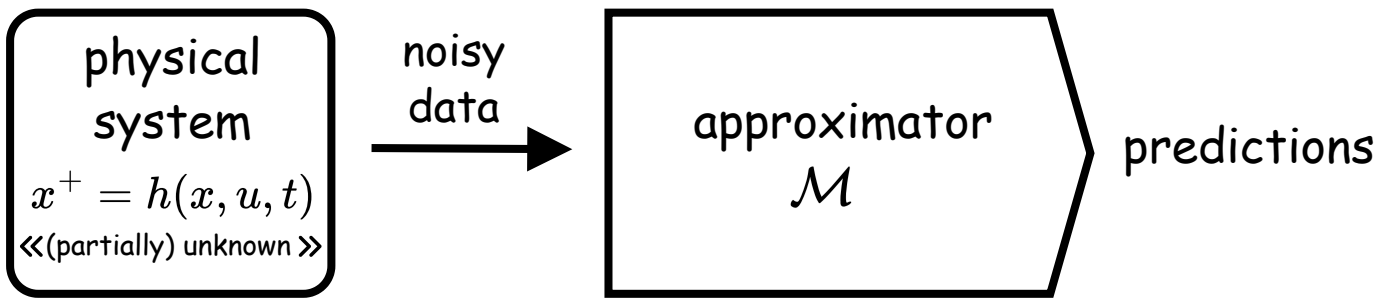
...still... there is no universally accepted model for blood flow!

[1] Johnston et al. 2006
[2] Johnston et al. 2004

A search for fast and accurate approximations of complex dynamical models

now, on the modeling aspect:

from (noisy) experimental data, can we "learn" models that provide real-time approximations to explicit (constitutive) models while accurately capturing the high-complexities in them?

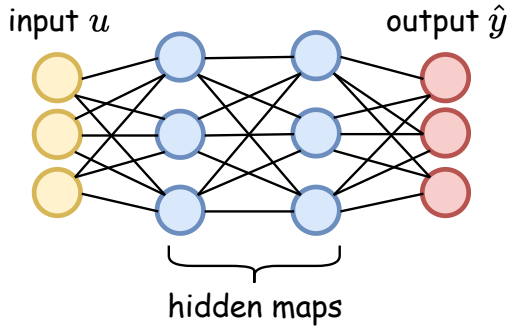


Neural networks for function approximation

Neural network:

$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

- powerful tool for approximating complex functions from data



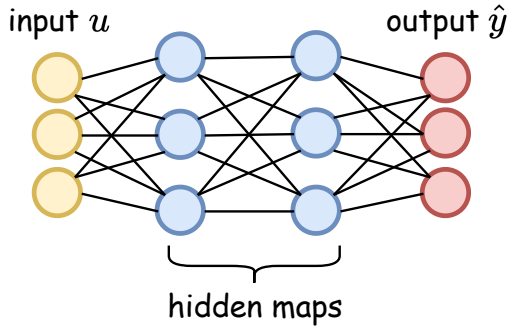
[3] LeCun et al. 1998
[4] Krizhevsky, Hinton, et al. 2009
[5] Deng et al. 2009
[6] DeVore 1998

Neural networks for function approximation

Neural network:

$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

- data $\{(u_i, y_i)\}_{i=1}^m$
  model $\mathcal{M}(u;\theta)$, $u = [u_1, \ldots, u_m]^\top$
  predictions $\hat{y}_i = \mathcal{M}(u_i;\theta)$



[3] LeCun et al. 1998
[4] Krizhevsky, Hinton, et al. 2009
[5] Deng et al. 2009
[6] DeVore 1998

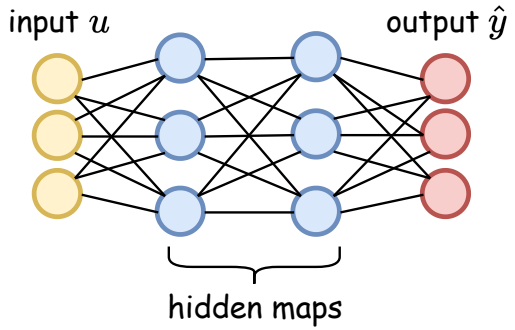
Neural networks for function approximation

Neural network:

$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

- $\theta \triangleq \{(W_l, b_l)\}_{l=1}^{L}$ is learned by empirical risk minimization:

$$\min_{\theta\in\mathcal{C}\subseteq\mathbb{R}^n} f(\theta) \triangleq \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i,\hat{y}_i\right)$$



[3] LeCun et al. 1998
[4] Krizhevsky, Hinton, et al. 2009
[5] Deng et al. 2009
[6] DeVore 1998

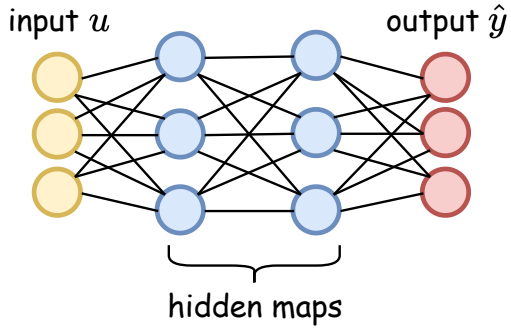
Neural networks for function approximation

$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

Neural network:

- $\theta \triangleq \{(W_l, b_l)\}_{l=1}^{L}$ is learned by empirical risk minimization:

$$\min_{\theta\in\mathcal{C}\subseteq\mathbb{R}^n} f(\theta) \triangleq \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i, \hat{y}_i\right)$$



[3] Bellman 1952; Novak and Woźniakowski 2009
[4] LeCun et al. 1998
[5] Krizhevsky, Hinton, et al. 2009
[6] Deng et al. 2009
[7] DeVore 1998

Neural networks for function approximation

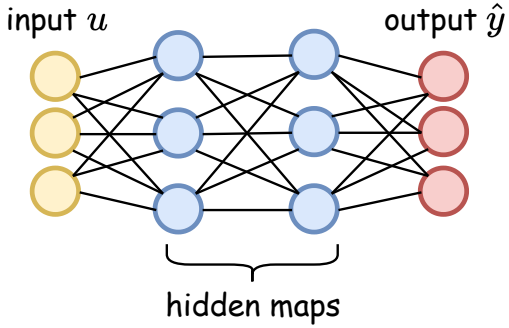
$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

Neural network:

- $\theta \triangleq \{(W_l, b_l)\}_{l=1}^{L}$ is learned by empirical risk minimization:

$$\min_{\theta\in\mathcal{C}\subseteq\mathbb{R}^n} f(\theta) \triangleq \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i, \hat{y}_i\right)$$

- breaks the classical *curse of dimensionality* [3]



[3] Bellman 1952; Novak and Woźniakowski 2009
[4] LeCun et al. 1998
[5] Krizhevsky, Hinton, et al. 2009
[6] Deng et al. 2009
[7] DeVore 1998

Neural networks for function approximation

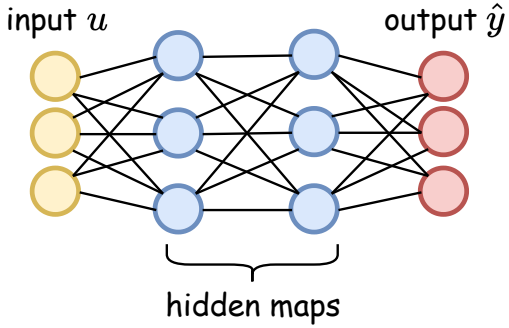
$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

Neural network:

- $\theta \triangleq \{(W_l, b_l)\}_{l=1}^{L}$ is learned by empirical risk minimization:

$$\min_{\theta\in\mathcal{C}\subseteq\mathbb{R}^n} f(\theta) \triangleq \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i, \hat{y}_i\right)$$

- breaks the classical *curse of dimensionality* [3]



[3] Bellman 1952; Novak and Woźniakowski 2009
[4] LeCun et al. 1998
[5] Krizhevsky, Hinton, et al. 2009
[6] Deng et al. 2009
[7] DeVore 1998

Neural networks for function approximation

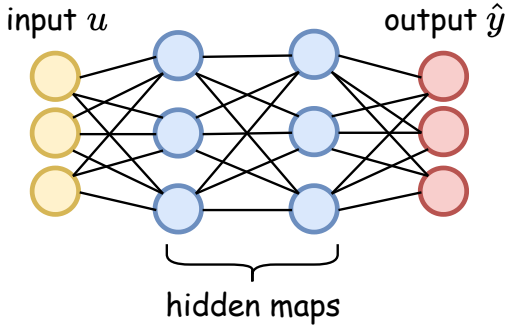
$$\mathcal{M}(u;\theta) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}\cdots\sigma_1(W_1u+b_1)\cdots+b_{L-1})+b_L)$$

Neural network:

- $\theta \triangleq \{(W_l, b_l)\}_{l=1}^{L}$ is learned by empirical risk minimization:

$$\min_{\theta\in\mathcal{C}\subseteq\mathbb{R}^n} f(\theta) \triangleq \frac{1}{m}\sum_{i=1}^{m}\ell\left(y_i, \hat{y}_i\right)$$

- breaks the classical *curse of dimensionality* [3]



MNIST [4] (28×28 pixels per image), CIFAR-10/100 [5] (32×32 pixels per image), ImageNet [6] (224×224 pixels per image), ...

As per classical theory for 1-Lipschitz functions [7], we would need $\mathcal{O}(1/\varepsilon^{784})$ parameters to achieve a uniform ε-error for MNIST.

[3] Bellman 1952; Novak and Woźniakowski 2009
[4] LeCun et al. 1998
[5] Krizhevsky, Hinton, et al. 2009
[6] Deng et al. 2009
[7] DeVore 1998

Neural networks for function approximation

As per classical theory for 1-Lipschitz functions [8], we would need $\mathcal{O}(1/\varepsilon^{784})$ parameters to achieve a uniform ε-error for MNIST.

- yet, deep neural networks perform astonishingly well in this *high-dimensional* regime!
- **why?**

[8] DeVore 1998

[9] Berner et al. 2021.

Neural networks for function approximation

As per classical theory for 1-Lipschitz functions [8], we would need $\mathcal{O}(1/\varepsilon^{784})$ parameters to achieve a uniform ε-error for MNIST.

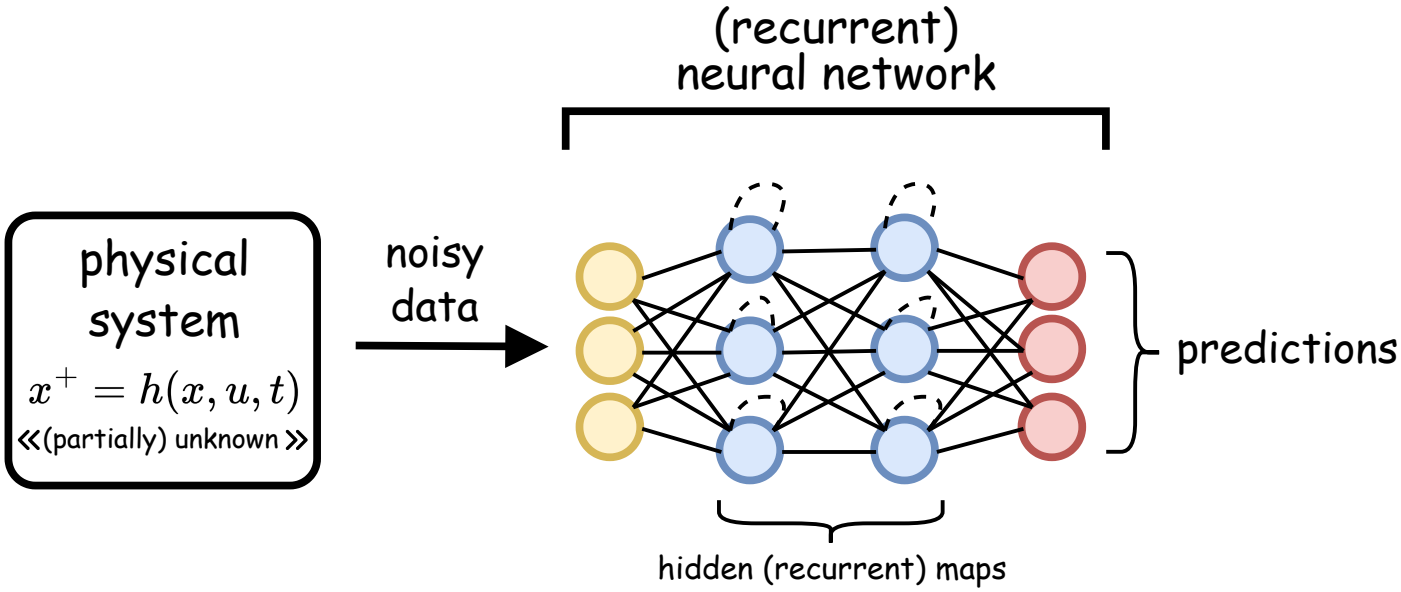
- yet, deep neural networks perform astonishingly well in this *high-dimensional* regime!
- **why?** ...some explanations exist[9], e.g., manifold assumption, random sampling, PDE assumptions,...

[8] DeVore 1998

[9] Berner et al. 2021.

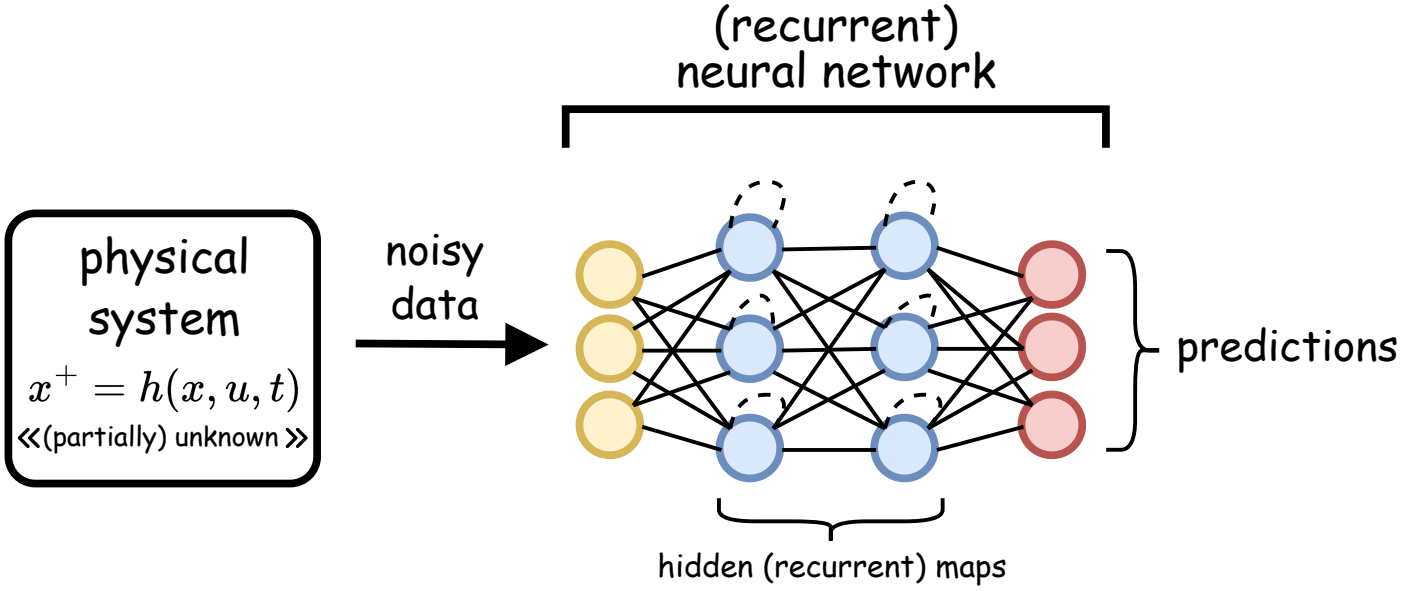
Neural networks for dynamical systems

- compared to FNNs, RNNs can capture temporal dependencies in data via feedback loops [10]



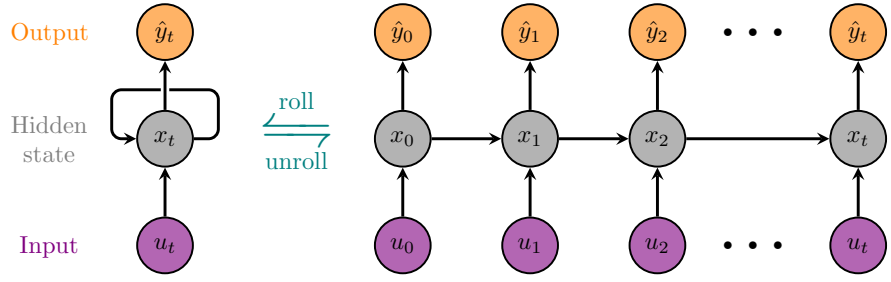
[10] Rumelhart, Hinton, and Williams 1986

Neural networks for dynamical systems



The vanilla RNN in state-space form:

$$x_{t+1} = \sigma_x(W_{xu}u_t + W_{xx}x_t + b_x)$$
$$\hat{y}_t = \sigma_y(W_{yx}x_t + b_y)$$



The approach (Motivation)

Architecture: a structured RNN for two-stage POMDP

- system identification: train the RNN to approximate the environment's Markovian state-space dynamics;
- optimal control policy selection by a FNN

The two stages combined constitutes the "agent's model".

[10] Bengio, Simard, and Frasconi 1994; Hochreiter 1998.
[11] Atiya and Parlos 2000.
[12] Adeoye and Bemporad 2025.

The approach (Motivation)

Architecture: a structured RNN for two-stage POMDP

- system identification: train the RNN to approximate the environment's Markovian state-space dynamics;
- optimal control policy selection by a FNN

The two stages combined constitutes the "agent's model".

Learning issues with BPTT algorithms:

- vanishing- and exploding-gradients[10]
- slow convergence of common BPTT algorithms[11]

[10] Bengio, Simard, and Frasconi 1994; Hochreiter 1998.
[11] Atiya and Parlos 2000.
[12] Adeoye and Bemporad 2025.

The approach (Motivation)

Architecture: a structured RNN for two-stage POMDP

- system identification: train the RNN to approximate the environment's Markovian state-space dynamics;
- optimal control policy selection by a FNN

The two stages combined constitutes the "agent's model".

Learning issues with BPTT algorithms:

- vanishing- and exploding-gradients[10]
- slow convergence of common BPTT algorithms[11]

Two common remedies: (see Ref.[12])

- (approximate) second-order algorithms, e.g., generalized Gauss-Newton (GGN), Levenberg-Marquardt (LM), CG, etc....
- nonlinear sequential state-estimation techniques, e.g., EKF

[10] Bengio, Simard, and Frasconi 1994; Hochreiter 1998.
[11] Atiya and Parlos 2000.
[12] Adeoye and Bemporad 2025.

RNN for Markovian state-space dynamics

Consider an RNN model of the form:

$$x_{t+1} = \sigma_x(W_{xu}u_t + W_{xy}y_t + W_{xx}x_t + b_x)$$
$$\hat{y}_t = \sigma_y(W_{yx}x_t + b_y)$$

σ_x: a nonlinear function (element-wise)
$u_t \in \mathbb{R}^{n_u}$: input at time t to the RNN
$y_t \in \mathbb{R}^{n_y}$: "observable" output at time t
$x_t \in \mathbb{R}^{n_x}$: RNN hidden state at time t
$\hat{y}_t \in \mathbb{R}^{n_y}$: RNN prediction at time t
$\theta_x \triangleq vec([W_{xu}\ W_{xy}\ W_{xx}\ b_x]) \in \mathbb{R}^{n_{\theta_x}}$
$\theta_y \triangleq vec([W_{yx}\ b_y]) \in \mathbb{R}^{n_{\theta_y}}$

e.g., an agent uses what it perceives of y_t for future predictions/actions

---

[13] McCloud 1993; Tumblr 2012
[14] Ha and Schmidhuber 2018.

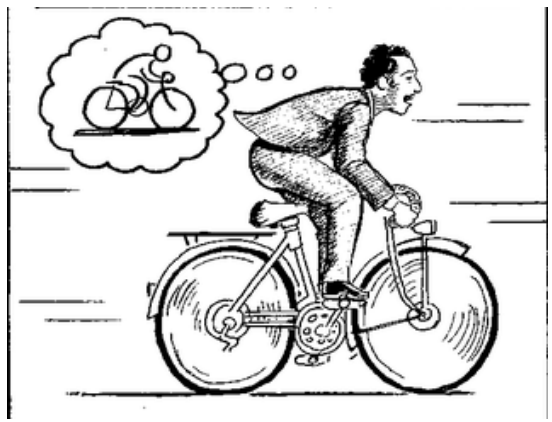
RNN for Markovian state-space dynamics

Consider an RNN model of the form:

$$x_{t+1} = \sigma_x(W_{xu}u_t + W_{xy}y_t + W_{xx}x_t + b_x)$$
$$\hat{y}_t = \sigma_y(W_{yx}x_t + b_y)$$

σ_x: a nonlinear function (element-wise)
$u_t \in \mathbb{R}^{n_u}$: input at time t to the RNN
$y_t \in \mathbb{R}^{n_y}$: "observable" output at time t
$x_t \in \mathbb{R}^{n_x}$: RNN hidden state at time t
$\hat{y}_t \in \mathbb{R}^{n_y}$: RNN prediction at time t
$\theta_x \triangleq vec([W_{xu}\ W_{xy}\ W_{xx}\ b_x]) \in \mathbb{R}^{n_{\theta_x}}$
$\theta_y \triangleq vec([W_{yx}\ b_y]) \in \mathbb{R}^{n_{\theta_y}}$

e.g., an agent uses what it perceives of y_t for future predictions/actions

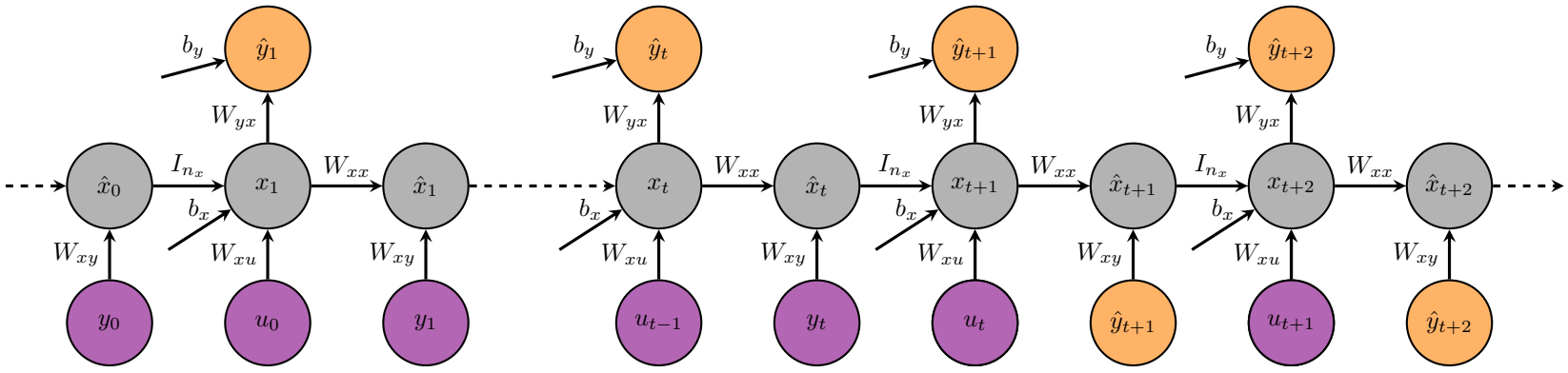


A World Model.[13]

Humans develop a mental model of the world based on what they are able to perceive with their limited senses.[14]

[13] McCloud 1993; Tumblr 2012
[14] Ha and Schmidhuber 2018.

DCRNN: An RNN for system identification



Unrolled RNN with dynamically consistent overshooting (DCRNN).

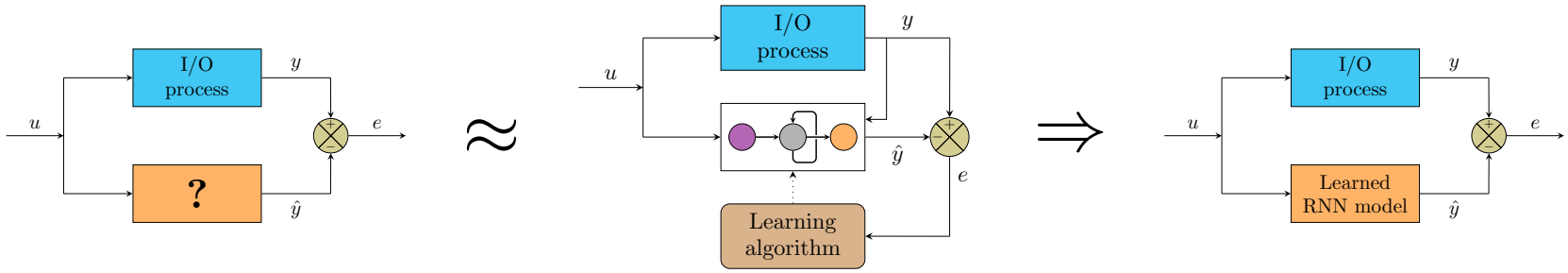
State-space equations of DCRNN[15]:

$$x_{t+1} = \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}u_t + b_x), \qquad \hat{x}_t = \begin{cases} W_{xx}x_t + W_{xy}y_t, & \forall 0 \leq t \leq m_-, \\ W_{xx}x_t + W_{xy}\hat{y}_t, & \forall m_- < t \leq N, \end{cases}$$
$$\hat{y}_t = W_{yx}x_t + b_y$$

$N = m_- + m_+$, m_- is the finite truncation time of the RNN unrolling and $m_+ > 1$ is the number of overshooting time steps into the future.

[15] Zimmermann et al. 2006.

Learning the DCRNN – Optimization problem



Learn x_t, θ_y, θ_x in parallel by solving equality-constrained optimization problem[16]: (RNN training problem = optimal control problem)[17]

$$\min_z \quad f(z) \triangleq \sum_{t=0}^{N-1} (y_t - \hat{y}_t)^2 + \mathcal{R}(x_0, \theta_x, \theta_y)$$

$$\text{s.t.} \quad c_t(z) = 0, \quad \forall t = 0, \dots, N,$$

$$\text{with} \quad c_t(z) \triangleq x_{t+1} - \sigma_x(\mathrm{I}\hat{x}_t + W_{xu} u_t + b_x), \quad z \triangleq [x_1^\top \cdots x_{N-1}^\top \, x_0^\top \, \theta_y^\top \, \theta_x^\top]^\top$$

$$\hat{x}_t = \begin{cases} W_{xx} x_t + W_{xy} y_t, & \forall 0 \le t \le m_-, \\ W_{xx} x_t + W_{xy} \hat{y}_t, & \forall m_- < t \le N-1 \end{cases}$$

---

[16] Adeoye and Bemporad 2025.

[17] Bemporad 2022; Bemporad 2023.

SQP approach

1. Define the Lagrangian:

$$\mathcal{L}(z, \lambda) \triangleq f(z) - \lambda^\top c(z), \qquad \lambda \in \mathbb{R}^m$$

2. Iteratively model the QP subproblem (iSQPRL):

$$\begin{aligned} \min_{d_z} \quad & \frac{1}{2} d_z^\top H_k d_z + \nabla f(z_k)^\top d_z \\ \text{s.t.} \quad & J_k d_z + c(z_k) = 0, \end{aligned}$$

with $J_k \equiv \nabla c(z_k) = [\nabla c_0(z_k) \cdots \nabla c_{N-1}(z_k)]^\top \in \mathbb{R}^{m \times n}$,
$H_k \equiv \nabla^2 \mathcal{L}(z_k, \lambda_k)$.

Newton-KKT solution

1. Define necessary (KKT) optimality conditions:

$$\tilde{\mathcal{F}}(z_k, \lambda_k) \triangleq \begin{bmatrix} \nabla \mathcal{L}(z_k, \lambda_k) \\ c(z_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

[18] Saad and Schultz 1986.

Newton-KKT solution

1. Define necessary (KKT) optimality conditions:

$$\tilde{\mathcal{F}}(z_k, \lambda_k) \triangleq \begin{bmatrix} \nabla \mathcal{L}(z_k, \lambda_k) \\ c(z_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

2. Derive the *Newton-KKT* system: For an optimal multiplier $\tilde{\lambda}$, we have

$$\underbrace{\begin{bmatrix} H_k & J_k^\top \\ J_k & 0 \end{bmatrix}}_{A_k} \underbrace{\begin{bmatrix} d_z \\ -\tilde{\lambda} \end{bmatrix}}_{d_k} = \underbrace{\begin{bmatrix} -g_k \\ -c_k \end{bmatrix}}_{b_k}$$

[18] Saad and Schultz 1986.

Newton-KKT solution

1. Define necessary (KKT) optimality conditions:

$$\tilde{\mathcal{F}}(z_k, \lambda_k) \triangleq \begin{bmatrix} \nabla \mathcal{L}(z_k, \lambda_k) \\ c(z_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

2. Derive the *Newton-KKT* system: For an optimal multiplier $\tilde{\lambda}$, we have

$$\underbrace{\begin{bmatrix} H_k & J_k^\top \\ J_k & 0 \end{bmatrix}}_{A_k} \underbrace{\begin{bmatrix} d_z \\ -\tilde{\lambda} \end{bmatrix}}_{d_k} = \underbrace{\begin{bmatrix} -g_k \\ -c_k \end{bmatrix}}_{b_k}$$

3. Solve the system for d_k:
   - use a (modified) BFGS update of H_k
   - solve the resulting system using the *restarted GMRES*[18]

[18] Saad and Schultz 1986.

Local convergence

A local two-step superlinear convergence with $H_k \succ 0$:

Lemma (Powell 1978b, Theorem 1)

Under twice-differentiability, compactness and convexity assumptions, we have

$$\lim_{k\to\infty} \frac{\|(P_k(H_k - \nabla^2\mathcal{L}^*)P_k)d_z\|}{\|d_z\|} = 0 \quad \textit{implies} \quad \lim_{l\to\infty} \frac{\|z_{l+1} - z^*\|}{\|z_{l-1} - z^*\|} = 0,$$

where P_k is the projection matrix $P_k = I - J_k(J_k^\top J_k)^{-1}J_k^\top$.

Requires only that a projection of each H_k is close to a projection of $\nabla^2\mathcal{L}^*$.

Globalization (line-search)

For globalization, we define a merit function such as

$$\mathcal{M}_1(z_k;\rho_k) = f_k + \frac{\rho_k}{\omega}\|c_k\|_1, \qquad \frac{\rho_k}{\omega} > 0$$

with the heuristic choice $\omega \geq 2$.
Ideally, we want some $\alpha_k > 0$ such that d_k satisfies

$$\mathcal{M}_1(z_k + \alpha_k d_z;\rho_k) \leq \mathcal{M}_1(z_k;\rho_k) + \nu\alpha_k \nabla_{d_z}\mathcal{M}_1(z_k;\rho_k)$$

$0 < \nu \leq 0.5$ is a small value and $\nabla_{d_z}\mathcal{M}_1(z_k;\rho_k)$ is the directional derivative of $\mathcal{M}_1$ along d_z.
Finally, we update (z, λ) using

$$z_{k+1} = z_k + \alpha_k d_z, \quad \lambda_{k+1} = \lambda_k + \alpha_k(\tilde{\lambda} - \lambda_k). \tag{3}$$

Globalization conditions for inexact SQP

- **key requirement:** account for the error r_k due to inexactness: $A_k d_k = b_k + r_k$
- thanks to the restarted GMRES, we have $\|r_k\| \leq \sigma_4 \|b_k\|, 0 < \sigma_4 < 1$

Proposition (Partial)

If we choose $\rho_k > \omega \|\tilde{\lambda}\|_\infty$, then d_z is guaranteed to be a descent direction for $\mathcal{M}_1$, and $\nabla_{d_z} \mathcal{M}_1(z_k; \rho_k) < 0$ at nonstationary points of iSQPRL problem.

Globalization conditions for inexact SQP

Corollary

The following property holds:

$$\rho_k \geq \frac{\omega}{2}\left[\frac{g_k^\top d_z + d_z^\top H_k d_z - d_z^\top r_z}{\|c_k\|_1 - \|r_\lambda\|_1} + \|\tilde{\lambda}\|_\infty\right]. \tag{4}$$

choose $\omega \geq 2$.

Our *safeguarding* rule, slightly modifying Powell 1978a, is (for $\bar{\rho} > 0$)

$$\rho_k = \begin{cases} \max\{\omega\|\tilde{\lambda}\|_\infty + \bar{\rho},\ \frac{1}{\omega}(\rho_{k-1} + \omega\|\tilde{\lambda}\|_\infty + \bar{\rho})\}, \forall k \neq 1 \\ \omega\|\tilde{\lambda}\|_\infty + \bar{\rho}, \quad \text{if } k = 1. \end{cases} \tag{5}$$

Convergence to a stationary point

Theorem

Let $\rho_k > \omega\|\tilde{\lambda}\|_\infty$ hold according to the safeguarding rule. Let $\{\lambda_k\}$ be bounded and $\{\alpha_k\}$ be bounded below by some positive constant. Then the sequence of iterates $\{z_k\}$, starting from an arbitary point, converges to a stationary point of $\mathcal{M}_1$.

A direct consequence of the descent property of $\mathcal{M}_1$ at d_k such that the line-search succeeds for all k. (Practical refinements are needed!)

[19] Panier and Tits 1991

Convergence to a stationary point

Theorem

Let $\rho_k > \omega\|\tilde{\lambda}\|_\infty$ hold according to the safeguarding rule. Let $\{\lambda_k\}$ be bounded and $\{\alpha_k\}$ be bounded below by some positive constant. Then the sequence of iterates $\{z_k\}$, starting from an arbitary point, converges to a stationary point of $\mathcal{M}_1$.

A direct consequence of the descent property of $\mathcal{M}_1$ at d_k such that the line-search succeeds for all k. (Practical refinements are needed!)

- To avoid so-called "Maratos effect", incorporate, e.g., a nonmonotone line-search procedure.

Q-superlinear convergence preserved! [19] (Even though now $\mathcal{M}_1$ is not forced to reduce at initial steps.)

[19] Panier and Tits 1991

Numerical illustration

RL problem (mountain-car env): The car's motion is described by

$$\bar{y}_{t+1} = \bar{y}_t + \dot{\bar{y}}_t, \quad \dot{\bar{y}}_{t+1} = \dot{\bar{y}}_t + 0.001 u_t - 0.0025 \cos(3\bar{y}_t),$$

$y_t = [\bar{y}_t, \dot{\bar{y}}_t]$, $-1.2 \leq \bar{y}_t \leq 0.5$, $-0.07 \leq \dot{\bar{y}}_t \leq 0.07$, $\bar{y}_t$: car's position at time t, $\dot{\bar{y}}_t$: car's velocity at time t, $u_t \in \{-1, 0, 1\}$: control action (applied force), $\sigma_x = \tanh$, $\sigma_y =$ identity; bounds on y_t enforced through $t_{a_i, b_i}(\hat{y}_i) = \frac{b_i + a_i}{2} + \frac{b_i - a_i}{2} \left(\frac{2\hat{y}_i}{1 + \hat{y}_i} \right)$.

Numerical illustration

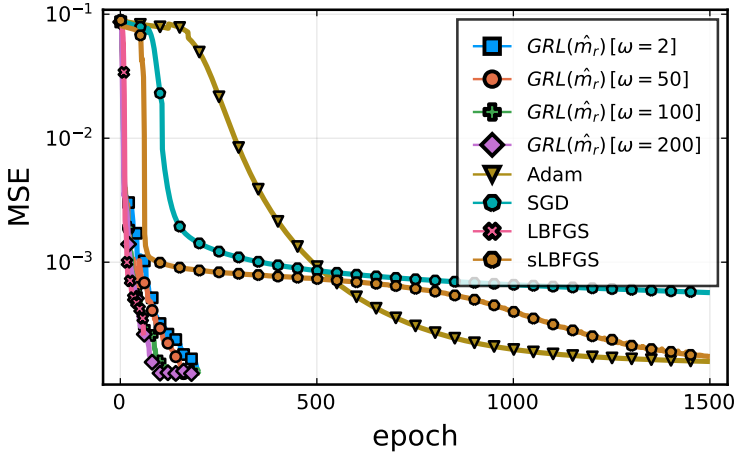
CSTR problem (ethylene oxidation): The reactor's dynamics are described by

$$
\begin{aligned}
\dot{\bar{y}}_1 &= u_1(1-\bar{y}_1\bar{y}_4)\\
\dot{\bar{y}}_2 &= u_1(u_2-\bar{y}_2\bar{y}_4) - A_1\, e^{\gamma_1/\bar{y}_4}(\bar{y}_2\bar{y}_4)^{0.5} - A_2\, e^{\gamma_2/\bar{y}_4}(\bar{y}_2\bar{y}_4)^{0.25}\\
\dot{\bar{y}}_3 &= -u_1\bar{y}_3\bar{y}_4 + A_1\, e^{\gamma_1/\bar{y}_4}(\bar{y}_2\bar{y}_4)^{0.5} - A_3\, e^{\gamma_3/\bar{y}_4}(\bar{y}_3\bar{y}_4)^{0.5}\\
\dot{\bar{y}}_4 &= \frac{u_1}{\bar{y}_1}(1-\bar{y}_4) + \frac{B_1}{\bar{y}_1}\, e^{\gamma_1/\bar{y}_4}(\bar{y}_2\bar{y}_4)^{0.5} + \frac{B_2}{\bar{y}_1}\, e^{\gamma_2/\bar{y}_4}(\bar{y}_2\bar{y}_4)^{0.25}\\
&\quad + \frac{B_3}{\bar{y}_1}\, e^{\gamma_3/\bar{y}_4}(\bar{y}_3\bar{y}_4)^{0.5} - \frac{B_4}{\bar{y}_1}(\bar{y}_4 - T_c)
\end{aligned}
$$

$y_t = [\bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_4]$ represent the gas density, ethylene concentration, ethylene oxide concentration, and temperature in the reactor; $u_t = [u_1, u_2]$, where u_1 is the feed volumetric flow rate and u_2 is the concentration of ethylene in the feed; $0.0704 \le u_1 \le 0.7042$, $0.2465 \le u_2 \le 2.4648$.

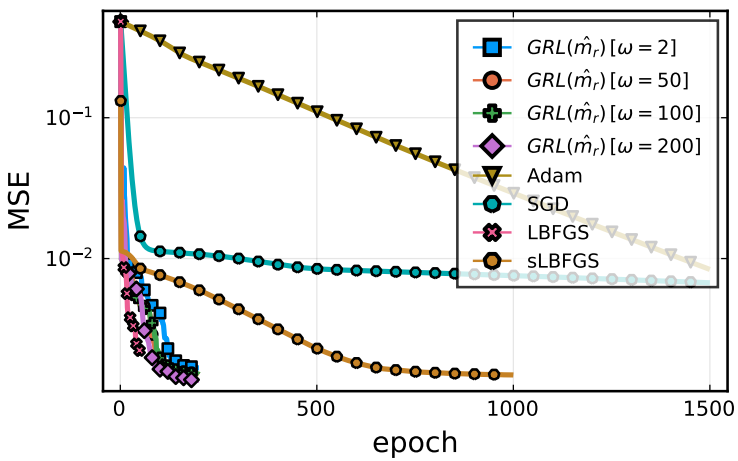
Numerical illustration

- mountain-car environment:



BPTT algorithm	Approximation error (MSE)	Iter	CPU time [s]
SGD	5.6828×10^{-4}	1500	5.6584×10
Adam	1.5821×10^{-4}	1500	4.9532×10
LBFGS	7.1389×10^{-4}	30	2.8670×10^{0}
sLBFGS	3.9884×10^{-4}	1000	2.8071×10
GRL($\hat{m}_r$), $\omega = 2$	1.3451×10^{-4}	200	6.4956×10
GRL($\hat{m}_r$), $\omega = 50$	$\mathbf{1.2702 \times 10^{-4}}$	200	1.0091×10^{2}
GRL($\hat{m}_r$), $\omega = 100$	1.3164×10^{-4}	200	5.6220×10
GRL($\hat{m}_r$), $\omega = 200$	1.2707×10^{-4}	200	1.1914×10^{2}

- ethylene oxidation:



BPTT algorithm	Approximation error (MSE)	Iter	CPU time [s]
SGD	6.7298×10^{-3}	1500	5.3596×10
Adam	8.3706×10^{-3}	1500	4.4071×10
LBFGS	1.5866×10^{-3}	46	1.6200×10^{0}
sLBFGS	1.4902×10^{-3}	1000	1.9693×10
GRL($\hat{m}_r$), $\omega = 2$	1.6720×10^{-3}	200	4.3819×10
GRL($\hat{m}_r$), $\omega = 50$	1.4948×10^{-3}	200	4.2956×10
GRL($\hat{m}_r$), $\omega = 100$	1.5043×10^{-3}	200	4.4190×10
GRL($\hat{m}_r$), $\omega = 200$	$\mathbf{1.3642 \times 10^{-3}}$	200	4.5000×10

Recurrent-Control Neural Network

- fix the DCRNN weights and train an FNN model on top of it to select future optimal control inputs $\hat{u}_t$:

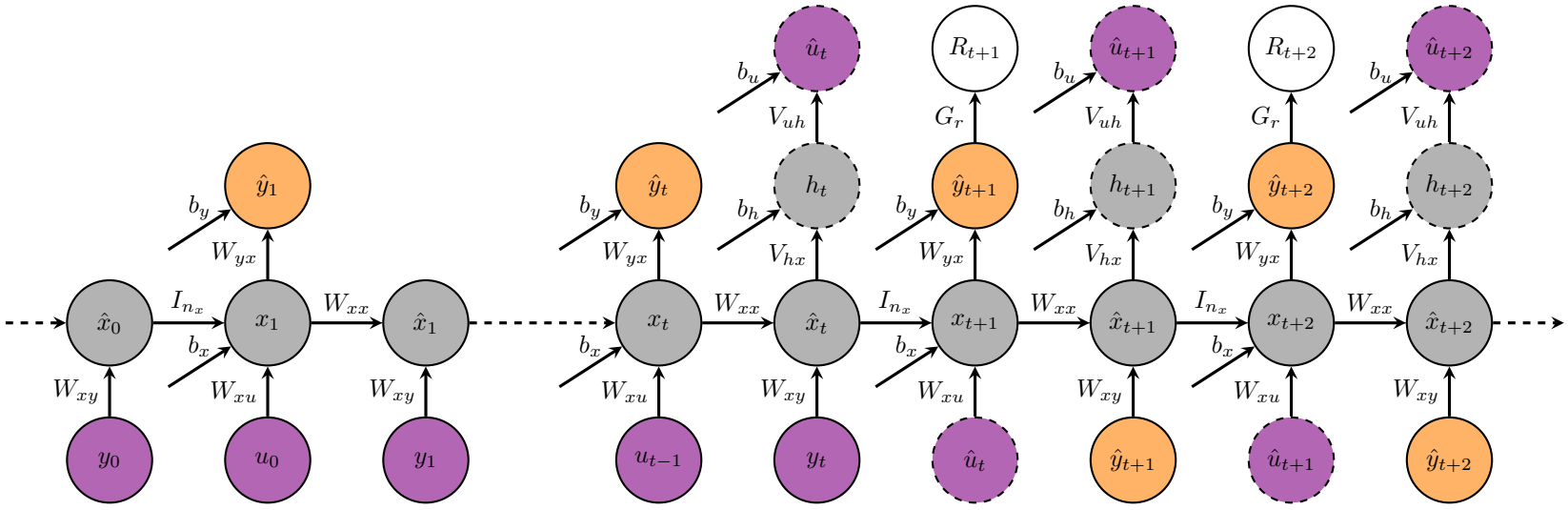
$$\hat{u}_t = \sigma_u(V_{uh}\sigma_h(V_{hx}\hat{x}_t + b_h) + b_u), \quad \forall m_- \leq t \leq N$$

$$x_{t+1} = \begin{cases} \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}u_t + b_x), & \forall 0 \leq t < m_- \\ \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}\hat{u}_t + b_x), & \forall m_- < t \leq N \end{cases}$$

$$R_t = G_r\sigma_r(W_{yx}x_t + b_y), \quad \forall m_- < t \leq N$$

$$\text{with} \quad \hat{x}_t = \begin{cases} W_{xx}x_t + W_{xy}y_t, & 0 \leq t \leq m_- \\ W_{xx}x_t + W_{xy}\hat{y}_t, & m_- < t \leq N \end{cases}$$

- $V_{uh} \in \mathbb{R}^{n_u \times n_h}$, $V_{hx} \in \mathbb{R}^{n_h \times n_x}$, $b_h \in \mathbb{R}^{n_h}$, $b_u \in \mathbb{R}^{n_u}$ are FNN parameters
- σ_u, σ_h are activation functions
- G_r, σ_r model problem's reward/cost function $R_t \equiv R(x_t, \hat{y}_t; \hat{u}_t)$



Architecture of recurrent control neural network (RCNN).

Learning the FNN — Optimization problem

Define the finite-sum

$$\bar{f}(\theta_u) \triangleq \sum_{t=m_-}^{N-1} R_t,$$

and consider the *regularized* optimization problem:

$$\min_{\theta_u} \quad \bar{f}(\theta_u) + \lambda_{\bar{r}} \bar{r}(\theta_u),$$

where $\theta_u \triangleq vec([\,V_{uh}\ V_{hx}\ b_h\ b_u]) \in \mathbb{R}^{n_{\theta_u}}$, $\lambda_{\bar{r}} > 0$.

Learning the FNN — Optimization problem

Define the finite-sum

$$\bar{f}(\theta_u) \triangleq \sum_{t=m_-}^{N-1} R_t,$$

and consider the *regularized* optimization problem:

$$\min_{\theta_u} \quad \bar{f}(\theta_u) + \lambda_{\bar{r}} \bar{r}(\theta_u),$$

where $\theta_u \triangleq vec([\,V_{uh}\ V_{hx}\ b_h\ b_u\,]) \in \mathbb{R}^{n_{\theta_u}}$, $\lambda_{\bar{r}} > 0$.

- specifically, solve using a *pseudo*-online generalized Gauss-Newton (GGN) algorithm for problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) \triangleq \bar{f}(x) + g(x)$$

Newton's algorithm ($g = 0$)

Newton's update: $x_{k+1} = x_k - \nabla^2 f(x)^{-1} \nabla f(x)$

Newton's algorithm ($g = 0$)

Newton's update: $x_{k+1} = x_k - \nabla^2 f(x)^{-1} \nabla f(x)$

- inverting $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ often highly expensive
- approximation: replace $\nabla^2 f(x)$ with $H_f \approx \nabla^2 f(x)$
- e.g., GGN approximation: $H_f = J_f^\top Q_f J_f$
  - let $m = N - m_-$
  - $J_f \in \mathbb{R}^{m \times n}$: Jacobian of $\hat{u}$ wrt x_k
  - $Q_f \in \mathbb{R}^{m \times m}$: Hessian of f wrt $\hat{u}$ at x_k

Newton's algorithm ($g = 0$)

Newton's update: $x_{k+1} = x_k - \nabla^2 f(x)^{-1} \nabla f(x)$

- inverting $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ often highly expensive
- approximation: replace $\nabla^2 f(x)$ with $H_f \approx \nabla^2 f(x)$
- e.g., GGN approximation: $H_f = J_f^\top Q_f J_f$
  - let $m = N - m_-$
  - $J_f \in \mathbb{R}^{m \times n}$: Jacobian of $\hat{u}$ wrt x_k
  - $Q_f \in \mathbb{R}^{m \times m}$: Hessian of f wrt $\hat{u}$ at x_k

- GGN iteration:
  $x_{k+1} = x_k - (J_f^\top Q_f J_f)^{-1} \nabla f(x)$
- or,
  $x_{k+1} = x_k - (J_f^\top Q_f J_f)^{-1} J_f^\top e_f$

- $\nabla f(x_k) \equiv J_f^\top e_f$
- e_f: gradient of f wrt $\hat{u}$ at x_k

GGN algorithm (g smooth)

Consider $g(x)$ smooth, e.g., $g(x) = \lambda \|x\|_2^2$

- GGN algorithm:
  $$x_{k+1} = x_k - (J_f^\top Q_f J_f + \nabla^2 g(x_k))^{-1}(J_f^\top e_f + \nabla g(x_k))$$

[20] Adeoye and Bemporad 2023a.

GGN algorithm (g smooth)

Consider $g(x)$ smooth, e.g., $g(x) = \lambda \|x\|_2^2$

- GGN algorithm:
  $x_{k+1} = x_k - (J_f^\top Q_f J_f + \nabla^2 g(x_k))^{-1}(J_f^\top e_f + \nabla g(x_k))$
- rewrite via our stylized augmentation:
- consider $y_i \in \mathbb{R}$, and define

$$J = \left[\begin{array}{c} J_f \\ \hdashline \nabla g(x_k)^\top \end{array}\right], Q = \left[\begin{array}{c:c} Q_f & 0 \\ \hdashline 0 & 0 \end{array}\right], e = \left[\begin{array}{c} e_f \\ \hdashline 1 \end{array}\right]$$

[20] Adeoye and Bemporad 2023a.

GGN algorithm (g smooth)

Consider $g(x)$ smooth, e.g., $g(x) = \lambda \|x\|_2^2$

- GGN algorithm:
  $x_{k+1} = x_k - (J_f^\top Q_f J_f + \nabla^2 g(x_k))^{-1}(J_f^\top e_f + \nabla g(x_k))$
- rewrite via our stylized augmentation:
- consider $y_i \in \mathbb{R}$, and define

$$J = \left[\begin{array}{c} J_f \\ \hdashline \nabla g(x_k)^\top \end{array}\right], Q = \left[\begin{array}{c:c} Q_f & 0 \\ \hdashline 0 & 0 \end{array}\right], e = \left[\begin{array}{c} e_f \\ \hdashline 1 \end{array}\right]$$

then[20]

$$x_{k+1} = x_k - (J_f^\top Q_f J_f + \nabla^2 g(x_k))^{-1}(J_f^\top e_f + \nabla g(x_k))$$

$$\Leftrightarrow$$

$$x_{k+1} = x_k - (J^\top Q J + \nabla^2 g(x_k))^{-1} J^\top e$$

[20] Adeoye and Bemporad 2023a.

GGN algorithm (g smooth)

GGN algorithm: $x_{k+1} = x_k - (J^\top Q J + \nabla^2 g(x_k))^{-1} J^\top e$

an interesting matrix identity[21]:
$(D - VA^{-1}B)^{-1} VA^{-1} = D^{-1}V(A - BD^{-1}V)^{-1}$

[21] Duncan 1944; Guttman 1946.

GGN algorithm (g smooth)

GGN algorithm: $x_{k+1} = x_k - (J^\top Q J + \nabla^2 g(x_k))^{-1} J^\top e$

an interesting matrix identity[21]:
$(D - VA^{-1}B)^{-1} VA^{-1} = D^{-1}V(A - BD^{-1}V)^{-1}$

so then (if $m < n$), cheaper updates for GGN:

$$x_{k+1} = x_k - \nabla^2 g(x_k)^{-1} J^\top (\underbrace{I_m + QJ \nabla^2 g(x_k)^{-1} J^\top}_{m \times m})^{-1} e$$

[21] Duncan 1944; Guttman 1946.

Self-concordant optimization ($g = 0$)

Both *pure* Newton and GGN algorithms may fail globally

- global properties of Newton's method are well-established for (generalized) self-concordant functions
- a *convex* function f is (M_f, ν)-generalized self-concordant (SC) if $\left|\left\langle \nabla^3 f(x)[v_1]v_2, v_2 \right\rangle\right| \leq M_f \|v_2\|_x^2 \|v_1\|_x^{\nu-2} \|v_1\|^{3-\nu}, \forall v_1, v_2 \in \mathbb{R}^n, M_f \geq 0, \nu > 0;$ local norm: $\|d\|_x \triangleq \langle \nabla^2 f(x) d, d \rangle^{1/2}$

Self-concordant optimization ($g = 0$)

Both *pure* Newton and GGN algorithms may fail globally

- global properties of Newton's method are well-established for (generalized) self-concordant functions
- a *convex* function f is (M_f, ν)-generalized self-concordant (SC) if $\left|\left\langle \nabla^3 f(x)[v_1]v_2, v_2 \right\rangle\right| \leq M_f \|v_2\|_x^2 \|v_1\|_x^{\nu-2} \|v_1\|^{3-\nu}, \forall v_1, v_2 \in \mathbb{R}^n, M_f \geq 0, \nu > 0;$ local norm: $\|d\|_x \triangleq \langle \nabla^2 f(x) d, d \rangle^{1/2}$
- standard SC if $\nu = 3, v_1 = v_2$

Self-concordant optimization ($g = 0$)

Both *pure* Newton and GGN algorithms may fail globally

- global properties of Newton's method are well-established for (generalized) self-concordant functions
- a *convex* function f is (M_f, ν)-generalized self-concordant (SC) if $\left|\left\langle \nabla^3 f(x)[v_1]v_2, v_2 \right\rangle\right| \leq M_f \|v_2\|_x^2 \|v_1\|_x^{\nu-2} \|v_1\|^{3-\nu}, \forall v_1, v_2 \in \mathbb{R}^n, M_f \geq 0, \nu > 0;$ local norm: $\|d\|_x \triangleq \langle \nabla^2 f(x) d, d \rangle^{1/2}$
- standard SC if $\nu = 3, v_1 = v_2$

globalization of Newton's method $\Leftrightarrow$ step-size *damping*:
$x_{k+1} = x_k - \bar{\alpha}_k \nabla^2 f(x)^{-1} \nabla f(x), \qquad 0 < \bar{\alpha}_k \leq 1$ (damping parameter)

Self-concordant optimization ($g = 0$)

damped Newton: $x_{k+1} = x_k - \bar{\alpha}_k \nabla^2 f(x)^{-1} \nabla f(x)$

- affine-invariant *local quadratic convergence* guarantee
- *global convergence* guarantee for an appropriately-chosen $\bar{\alpha}_k$

[22] Nemirovski 2004; Nesterov and Nemirovskii 1994.
[23] Sun and Tran-Dinh 2019.

Self-concordant optimization ($g = 0$)

damped Newton: $x_{k+1} = x_k - \bar{\alpha}_k \nabla^2 f(x)^{-1} \nabla f(x)$

- affine-invariant *local quadratic convergence* guarantee
- *global convergence* guarantee for an appropriately-chosen $\bar{\alpha}_k$
- for standard SC f[22]: $\bar{\alpha}_k = \frac{1}{1+M_f \underbrace{\|\nabla f\|^*_{x_k}}_{\text{Newton decrement}}}$

[22] Nemirovski 2004; Nesterov and Nemirovskii 1994.
[23] Sun and Tran-Dinh 2019.

Self-concordant optimization ($g = 0$)

damped Newton: $x_{k+1} = x_k - \bar{\alpha}_k \nabla^2 f(x)^{-1} \nabla f(x)$

- affine-invariant *local quadratic convergence* guarantee
- *global convergence* guarantee for an appropriately-chosen $\bar{\alpha}_k$
- for standard SC f[22]: $\bar{\alpha}_k = \dfrac{1}{1+M_f \underbrace{\|\nabla f\|^*_{x_k}}_{\text{Newton decrement}}}$
- convergence theory extends for generalized SC functions[23]
- several customized algorithms and robust loss functions arising from this notion, e.g., in nonsmooth composite optimization, regression problems and robust statistics, etc....

---

[22] Nemirovski 2004; Nesterov and Nemirovskii 1994.

[23] Sun and Tran-Dinh 2019.

Self-concordant regularization

assume $g \neq 0$ is (M_g, ν)-generalized SC

- choose $\bar{\alpha}_k = \frac{\alpha_k}{1 + M_g \|\nabla g\|^*_{x_k}}, \qquad 0 < \alpha_k \leq 1$
- GGN-SCORE algorithm: $x_{k+1} = x_k - \bar{\alpha}_k (J^\top Q J + \nabla^2 g(x_k))^{-1} J^\top e$
- if $m < n$,
  $x_{k+1} = x_k - \bar{\alpha}_k \nabla^2 g(x_k)^{-1} J^\top (\underbrace{I_m + Q J \nabla^2 g(x_k)^{-1} J^\top}_{m \times m})^{-1} e$

Self-concordant regularization: what we get...

local linear-quadratic convergence guarantee for (strongly) convex problems[24]

Assume:

- f, g are γ_l, γ_a-strongly convex
- f, g have γ_u, γ_b-Lipschitz continuous first derivatives
- f, g have γ_f, γ_g-Lipschitz continuous second derivatives H_f, H_g
- g is $(M_g, 3)$-generalized SC (and scaled by $\lambda \geq 0$)

Key Remark

There exist $\beta, \tilde{\beta}, K_1 > 0$, with $Q \leq K_1 I$, such that $\|e\| \leq \beta, \|J\| \leq \tilde{\beta}$, and hence

$$\|\lambda I + QJH_g^{-1}J^T\| \leq \lambda + (K/\gamma_a), \qquad \boxed{K = K_1\tilde{\beta}^2, H_g = \nabla^2 g}$$

[24] Adeoye and Bemporad 2023a.

Self-concordant regularization: what we get...

Theorem (local loss behaviour and suboptimality)

If $\alpha_k = \alpha = \frac{\sqrt{\gamma_a}}{\beta_1}(K + \lambda\gamma_a)$, *and under given regularity and unbiasedness conditions on the GGN matrices and gradients,*

$$\mathbb{E}[\mathcal{L}(x_{k+1})] \leq \mathcal{L}(x_k) - \left(\frac{\lambda\omega(\zeta_k)}{M_g^2} + \frac{\gamma_l \omega''(\tilde{\zeta}_k)}{2\gamma_a} - \xi \right)$$

$$\mathbb{E}\|x_{k+1} - x^*\|_{x_{k+1}} \leq \vartheta \|x_k - x^*\|_{x_k} + \frac{\gamma_u}{\beta_1}\|x_k - x^*\| + \frac{\gamma_g}{2}\|x_k - x^*\|^2$$

$$\omega(t) \triangleq t - \ln(1+t), \quad \zeta_k \triangleq \frac{M_g}{1+\tilde{\zeta}_k},$$

$$\tilde{\zeta}_k \triangleq M_g \|\nabla g\|^*_{x_k}, \quad \xi \triangleq \frac{2(\gamma_u + \lambda\gamma_b)}{\sqrt{\gamma_a}}, \beta_1 \triangleq \beta\tilde{\beta}$$

$$\vartheta \triangleq 1 + \frac{\lambda}{\sqrt{\gamma_a}\beta_1(1 - M_g\|x_k - x^*\|_{x_k})}.$$

Self-concordant regularization: what we get...

non-asymptotic guarantee for the last-iterate convergence of neural network predictions to the outputs of a given target function[25]
Consider a two-layer neural network:

$$u \mapsto \mathcal{M}(u; x) \triangleq \kappa(p) \sum_{i=1}^{p} v_i \sigma(w_i u), \tag{7}$$

where $\kappa(p)$ is some scaling that depends on n, e.g., $\kappa(p) = 1/\sqrt{p}$.

[25] Adeoye, Petersen, and Bemporad 2024.

Self-concordant regularization: what we get...

non-asymptotic guarantee for the last-iterate convergence of neural network predictions to the outputs of a given target function[25]
Consider a two-layer neural network:

$$u \mapsto \mathcal{M}(u;x) \triangleq \kappa(p) \sum_{i=1}^{p} v_i \sigma(w_i u), \tag{7}$$

where $\kappa(p)$ is some scaling that depends on n, e.g., $\kappa(p) = 1/\sqrt{p}$.
learning problem (ERM):

$$\min_{x \in \mathbb{R}^n} \mathcal{L}(x) \triangleq \hat{R}_s(\mathcal{M}) + g(x), \tag{8}$$

where $\hat{R}_s(\mathcal{M}) \triangleq \frac{1}{m} \sum_{i=1}^{m} \ell(\mathcal{M}(u_i;x), y_i)$ (assumed strongly convex wrt $\mathcal{M}$), $\ell \colon \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}$ is loss function.

---

[25] Adeoye, Petersen, and Bemporad 2024.

Self-concordant regularization: what we get...

Assume standard regularity assumptions on the GGN matrices and the gradient terms.

Theorem (nonasymptotic convergence)

After $K \triangleq \frac{1}{\bar{\alpha}} \log((1 + \|\mathcal{M}_0 - \mathcal{M}^*\|^2)/\epsilon)$ *iterations, for any* $\epsilon, \bar{\alpha} \in (0, 1)$,

$$\|\mathcal{M}_K - \mathcal{M}^*\|^2 \leq \epsilon$$

where $\mathcal{M}^*$ *is a target function (e.g., a teacher network).*

Theorem (regularized loss evolution)

The "unaugmented" J_k *is* $mC\kappa(p)$*-Lipschitz,* C *constant, and for all* k,

$$\mathcal{L}(x_{k+1}) \leq \mathcal{L}(x_k) - \xi_k$$

where $\xi_k = \mathcal{O}\left(\hat{\beta}_1^2 \bar{c}_k / \hat{\beta}_m^4\right)$ *is positive,* $\hat{\beta}_1 \triangleq \sigma_{\max}(J_k)$, $\hat{\beta}_m \triangleq \sigma_{\min}(J_k)$ *(augmented).*

Self-concordant smoothing

constructing an SC g:

- **approach**: epi-smoothing by infimal convolution[26]: $g_s(x;\mu) = g \square h_\mu \triangleq \inf_{w \in \mathbb{R}^n} \left\{ g(w) + h_\mu(x-w) \right\}, \quad h_\mu(\cdot) \triangleq \mu h\left(\frac{\cdot}{\mu}\right), \quad \mu \in \mathbb{P}$
- consider a parameterized family of regularization kernels $\mathcal{H} \triangleq \left\{ (x,w) \mapsto h_\mu(x-w) \mid x, w \in \mathbb{R}^n, \mu \in \mathbb{P} \right\}$

  $h_\mu \in \mathcal{H}$ with $h(x) \triangleq \sum_{i=1}^{n} \phi(x_i)$, $\liminf_{\|x\| \to \infty} \frac{\phi(x)}{\|x\|} = +\infty$ and $\phi \in \mathcal{F}_{M_\phi, \nu}$

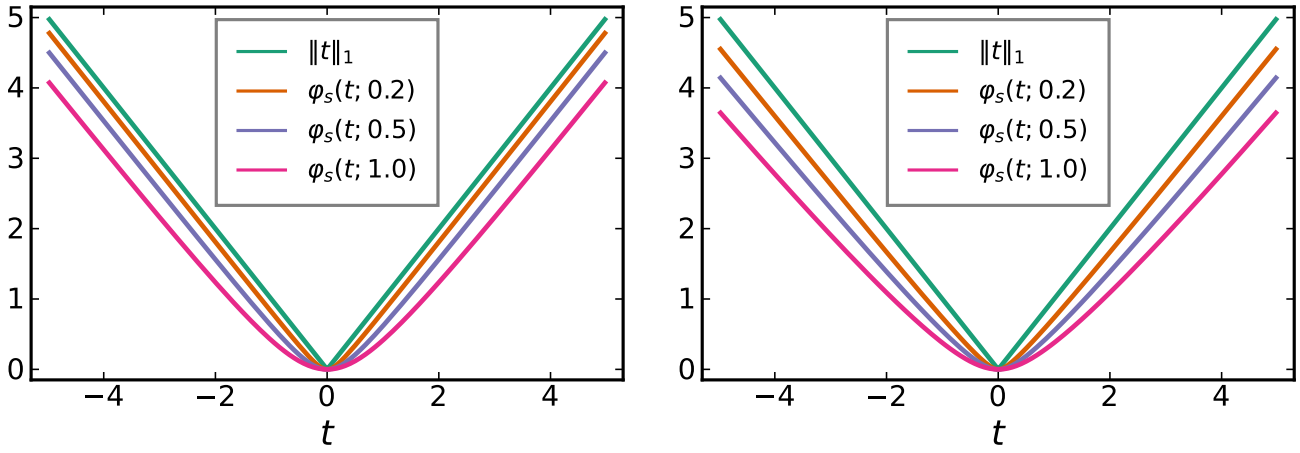
Let $\varphi_s(t;\mu) \triangleq (\varphi \square \phi_\mu)(t)$; then $g_s \triangleq g \square h_\mu \equiv \sum_{i=1}^{n} \varphi_s(x_i;\mu)$ is (M_g, ν)-generalized SC![27]

---

[26] Burke and Hoheisel 2017.

[27] Adeoye and Bemporad 2023b

Self-concordant smoothing



Generalized self-concordant smoothing of $\|\cdot\|_1$ with $\phi(t) = \sqrt{1+|t|^2} - 1$ (left) and $\phi(t) = \frac{1}{2}\left[\sqrt{1+4t^2} - 1 + \log\left(\frac{\sqrt{1+4t^2}-1}{2t^2}\right)\right]$ (right). The smooth approximation is shown for $\mu = 0.2, 0.5, 1.0$.

Julia package

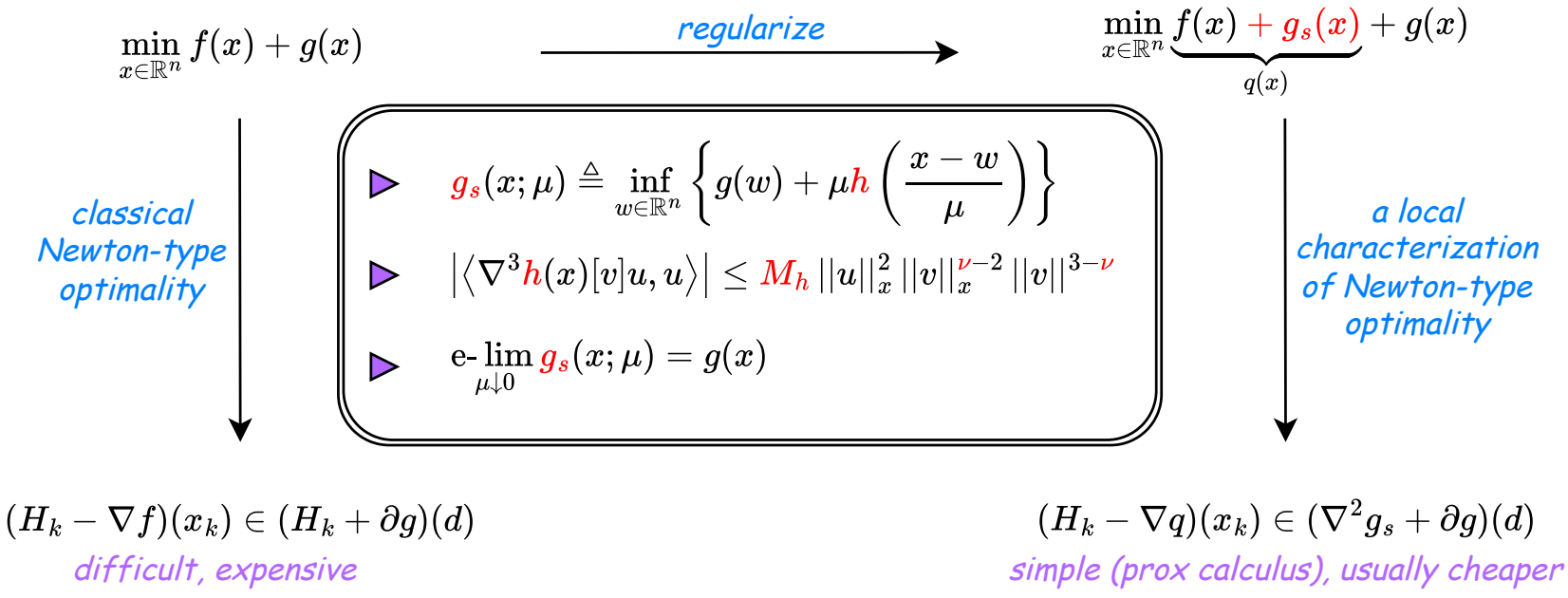
All implemented in a Julia package,

`SelfConcordantSmoothOptimization.jl`

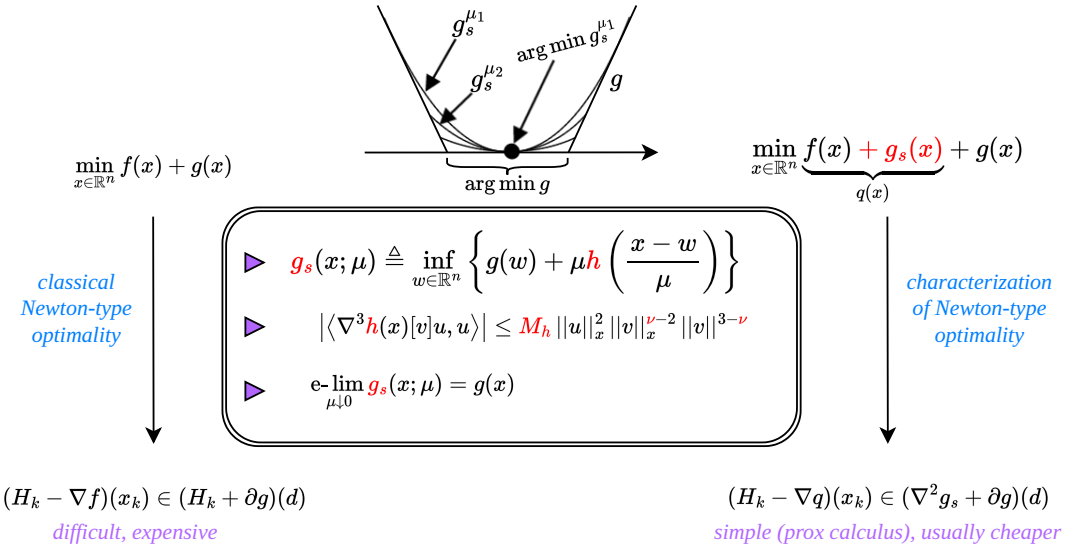
A Python port is available:

`https://github.com/adeyemiadeoye/pySCSOpt`

Self-concordant smoothing in proximal quasi-Newton methods



Self-concordant smoothing in proximal quasi-Newton methods



each g_s^μ is ε^μ-optimal for g (ε-$\operatorname{argmin} g \triangleq \{x \mid g(x) \le \inf g + \varepsilon\}$)

ideally, one wants g_s with parametrization set $\{g_s^\mu\}_{\mu>0}$ satisfying

$$\limsup_{\mu} (\operatorname{argmin} g_s^\mu) \subset \operatorname{argmin} g$$

"every cluster point of the minimizers of $\{g_s^\mu\}_{\mu>0}$ is a minimizer of g"[a]

[a] Rockafellar and Wets 2009.

A proximal GGN algorithm

Algorithm `Prox-GGN-SCORE`

Require: $x_0 \in \mathbb{R}^n$, f, g, $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$, $\mathcal{M}$, $\{u_i, y_i\}_{i=1}^m$ with $y_i \in \mathbb{R}^{n_y}$, $\alpha \in (0, 1]$

for $k = 0, \dots$ **do**
  $H_g \leftarrow \nabla^2 g_s(x_k)$; $\eta_k \leftarrow \left\| \nabla g_s(x_k) \right\|_{H_g}^*$ ▷ NOTE: H_g is diagonal
  $\bar{\alpha}_k \leftarrow \frac{\alpha}{1 + M_g \eta_k}$
  if $m + n_y \leq n$ **then**
    $\delta_k^{\mathrm{ggn}} = -H_g^{-1} J_k^\top (I_m + Q_k J_k H_g^{-1} J_k^\top)^{-1} u_k$
  else
    $\delta_k^{\mathrm{ggn}} = -(J_k^\top Q_k J_k + H_g)^{-1} J_k^\top u_k$
  end if
  $x_{k+1} \leftarrow \mathrm{prox}_{\alpha g}^{H_g}(x_k + \bar{\alpha}_k \delta_k^{\mathrm{ggn}})$ ▷ simplified by prox-calculus[28]
end for

[28] Becker, Fadili, and Ochs 2019.

Convergence properties

assumptions:

- f is convex and $f \in \mathcal{C}^{2,2}_{L_f}(\mathbb{R}^n)$
- $\rho_0 I_n \leq \nabla^2 f(x^*) \leq L I_n$, $\rho I_n \leq \nabla^2 g_s(x^*) \leq L_0 I_n$ locally with $L \geq \rho_0 > 0$ and $L_0 \geq \rho > 0$
- $g \in \Gamma_0(\mathbb{R}^n)$
- $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$

Convergence properties

Theorem

Starting from an arbitrary point, `Prox-GGN-SCORE` *converges to a solution, with* $\mathcal{L}$ *evolving as*

$$\mathcal{L}(x_{k+1}) \leq \mathcal{L}(x_k) - (L_f/6)\|x_{k+1} - x_k\|^3$$

Convergence properties

$$d_\nu(x,y) \triangleq \begin{cases} M_g\|y-x\| & \text{if } \nu = 2, \\ \left(\frac{\nu}{2}-1\right) M_g \|y-x\|_2^{3-\nu} \|y-x\|_x^{\nu-2} & \text{if } \nu > 2 \end{cases}$$

$$\omega_\nu(\tau) \triangleq \begin{cases} \frac{\exp(\tau)-\tau-1}{\tau^2} & \text{if } \nu = 2, \\ \frac{-\tau-\ln(1-\tau)}{\tau^2} & \text{if } \nu = 3, \\ \frac{(1-\tau)\ln(1-\tau)+\tau}{\tau^2} & \text{if } \nu = 4, \\ \left(\frac{\nu-2}{4-\nu}\right)\frac{1}{\tau}\left[\frac{\nu-2}{2(3-\nu)\tau}\left((1-\tau)^{\frac{2(3-\nu)}{2-\nu}}-1\right)-1\right] & \text{otherwise} \end{cases}$$

$$R_\nu(\tau) \triangleq \begin{cases} \left(\frac{3}{2}+\frac{\tau}{3}\right)\exp(\tau) & \text{if } \nu = 2, \\ \frac{1-(1-\tau)^{\frac{4-\nu}{\nu-2}}-\left(\frac{4-\nu}{\nu-2}\right)\tau(1-\tau)^{\frac{4-\nu}{\nu-2}}}{\left(\frac{4-\nu}{\nu-2}\right)\tau^2(1-\tau)^{\frac{4-\nu}{\nu-2}}} & \text{if } \nu \in (2,3] \end{cases}$$

$$\bar{\omega}_\nu(\tau) \triangleq \begin{cases} \frac{2\exp\left(\frac{\tau}{2}\right)-2}{\tau} & \text{if } \nu = 2, \\ -\frac{\ln(1-\tau)}{\tau} & \text{if } \nu = 3, \\ \left(\frac{\nu-2}{\nu-3}\right)\frac{1-(1-\tau)^{\frac{\nu-3}{\nu-2}}}{\tau} & \text{otherwise} \end{cases}$$

Convergence properties

local assumptions:

- $\|J_k v\| \geq \beta_1 \|v\|$, $\beta_1 > 0$, for any $v \in \mathbb{R}^n$
- $\left\|e_f(x_k)\right\| \leq \beta_2$, $\|Q_f(x_k)\| \leq \beta_3$ for some $\beta_2, \beta_3 > 0$

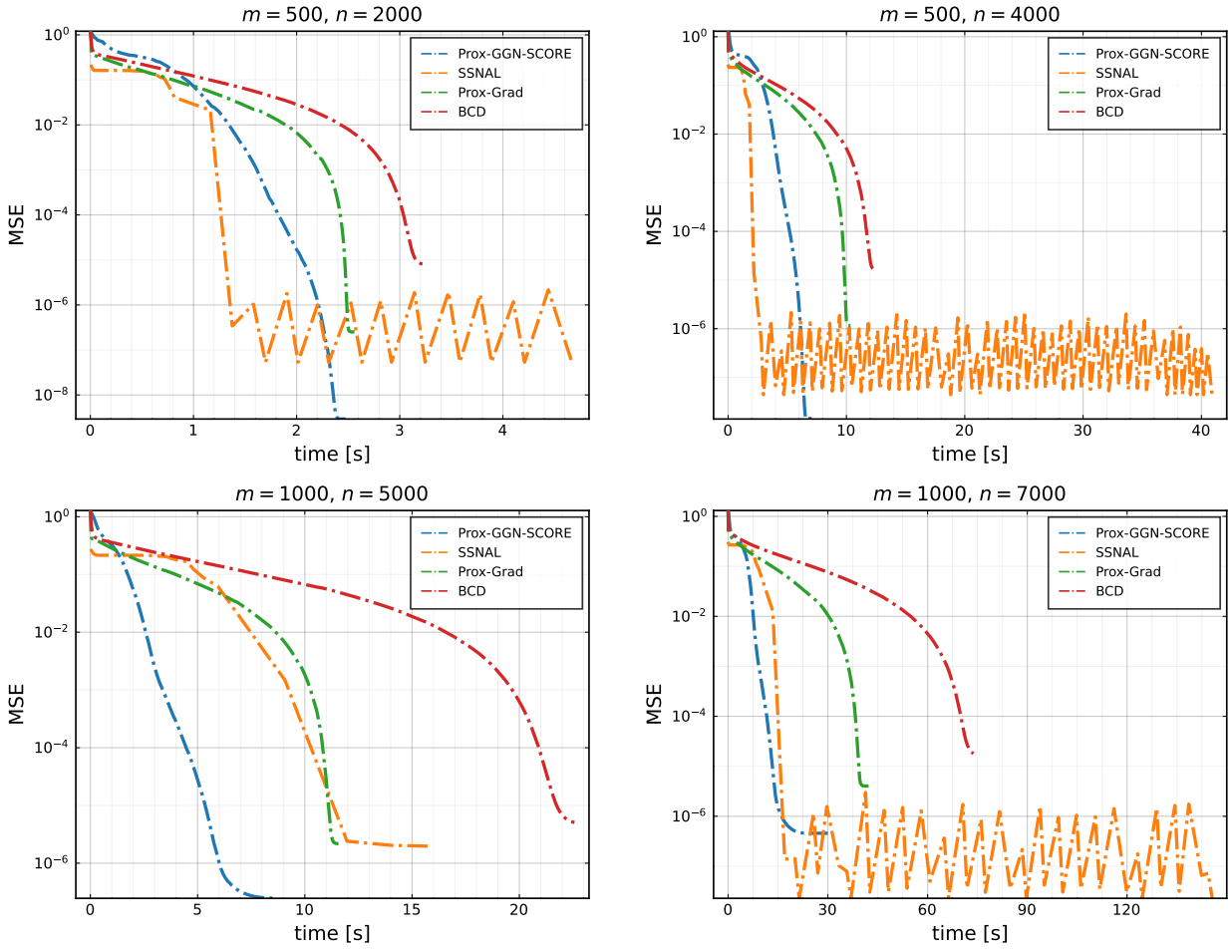
let $\tilde{\beta} \triangleq \beta_2 \beta_3$, $\alpha_k \equiv \alpha \in (0, 1]$, $\lambda_k \triangleq 1 + M_g \omega_\nu(-d_\nu(x^*, x_k)) \|x_k - x^*\|_{x_k}$,
$\vartheta_k \triangleq \left(\frac{\lambda_k - \alpha_k}{\lambda_k}\right) \bar{\omega}_\nu(d_\nu(x^*, x_k)) + \frac{L_f}{2\sqrt{\rho}}$, $R_k \triangleq R_\nu(d_\nu(x^*, x_k)) d_\nu(x^*, x_k)$

Theorem (local suboptimality of `Prox-GGN-SCORE` iterates)

Starting from $x_0 \in \mathcal{E}_r(x^*)$, if $d_\nu(x^*, x_k) < 1$, then

$$\|x_{k+1} - x^*\|_{x^*} \leq \frac{L(\lambda_k - \alpha_k)}{\lambda_k \sqrt{\rho}} + \frac{\tilde{\beta}\left\|x_k - x^*\right\|}{\sqrt{\rho}} + R_k \|x_k - x^*\|_{x^*} + \vartheta_k \|x_k - x^*\|^2$$

Sparse group lasso test



$$\min_{x \in \mathbb{R}^n} \underbrace{\frac{1}{2}\|Ax - y\|^2}_{=:f(x)} + \underbrace{\beta\|x\|_1 + \beta_{\mathcal{G}} \sum_{j \in \mathcal{G}} \omega_j \|x_j\|}_{=:g(x)}$$

Learning the FNN with GGN-SCORE

$$\min_{\theta_u \equiv \bar{z}} \quad \bar{f}(\theta_u) + \lambda_{\bar{r}} \bar{r}(\theta_u), ;\ n_d = n_y \cdot n_b;\ n_b = \text{mini-batch size}$$

$$\bar{z}_{k+1} = \bar{z}_k - \frac{\sigma}{1 + M_{\bar{r}} \bar{\eta}_k} \bar{H}_k^{-1} \bar{J}_k \bar{B}_k^{-1} \bar{e}_k, \quad \bar{B}_k \triangleq \lambda_{\bar{r}} \mathrm{I} + \bar{Q}_k \bar{J}_k \bar{H}_k^{-1} \bar{J}_k^\top,$$

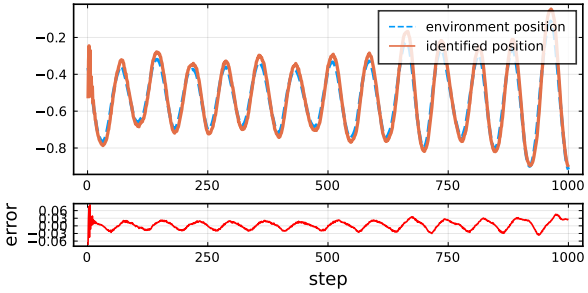
$$\bar{J}_k^\top = \left[\begin{array}{c:c} J_{\hat{y}}^\top & \lambda_{\bar{r}} \bar{g}_k \end{array} \right], \bar{Q}_k = \left[\begin{array}{c:c} Q_{\bar{f}} & 0 \\ \hdashline 0 & 0 \end{array} \right], \bar{e}_k = \left[\begin{array}{c} g_{\bar{f}} \\ \hdashline 1 \end{array} \right],$$

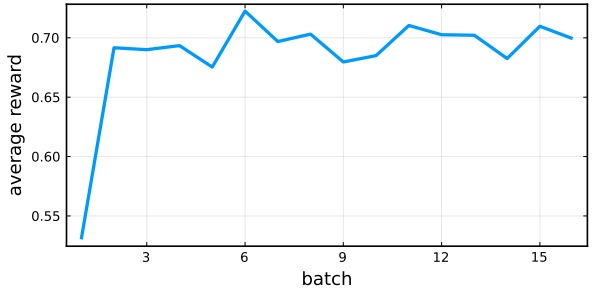
$\bar{H}_k \in \mathbb{R}^{n_{\theta_u} \times n_{\theta_u}}$: Hessian of $\bar{r}$ with respect to θ_u, $J_{\hat{y}} \in \mathbb{R}^{n_d \times n_{\theta_u}}$: Jacobian of $\hat{y}_k$ with respect to θ_u, $\bar{g}_k$: gradient of $\bar{r}$ with respect to θ_u, $g_{\bar{f}} \in \mathbb{R}^{n_d}$: residual vector defined as the gradient of $\bar{f}$ with respect to $\hat{y}_k$, $Q_{\bar{f}} \in \mathbb{R}^{n_d \times n_d}$: Hessian of $\bar{f}$ with respect to $\hat{y}_k$.

choose: $0 < \sigma \leq 1$, $\bar{\eta}_k = \left\langle \bar{g}_k, \bar{H}_k^{-1} \bar{g}_k \right\rangle^{1/2}$, and $M_{\bar{r}} \geq 0$ is associated with $\bar{r}$.

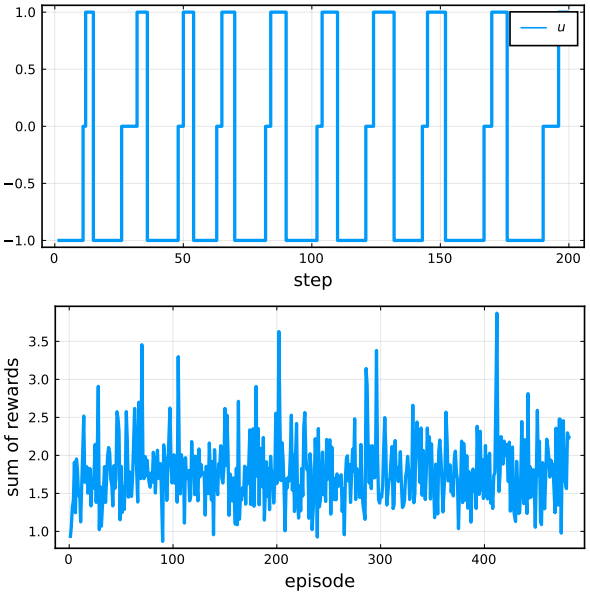
use: $\bar{B}_k^{-1} \approx \underline{\mathrm{B}}_k$ with $\underline{\mathrm{B}}_k \in \mathbb{R}^{n_{\theta_u} \times n_{\theta_u}}$ with entries $\underline{\mathrm{b}}_{i,j} = \bar{\delta}_{i,j} / \bar{b}_{i,i}$, where $\bar{\delta}_{i,j}$ is the Kronecker delta function and $\bar{b}_{i,i}$ are the diagonal entries of $\bar{B}_k$.

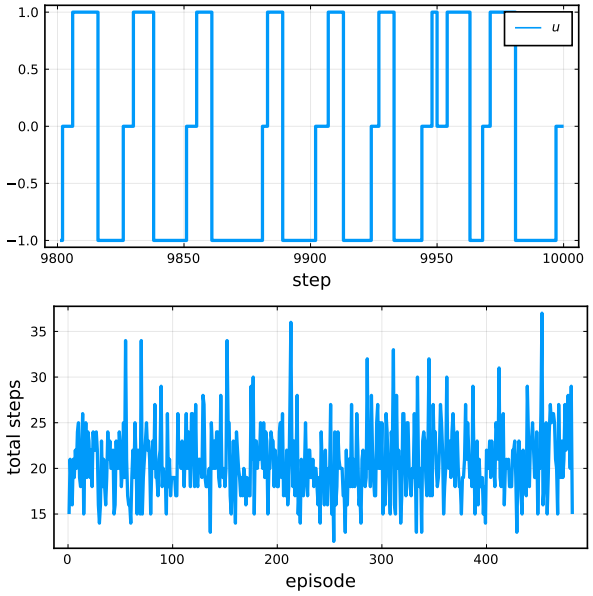
Full RCNN simulation results (mountain car)





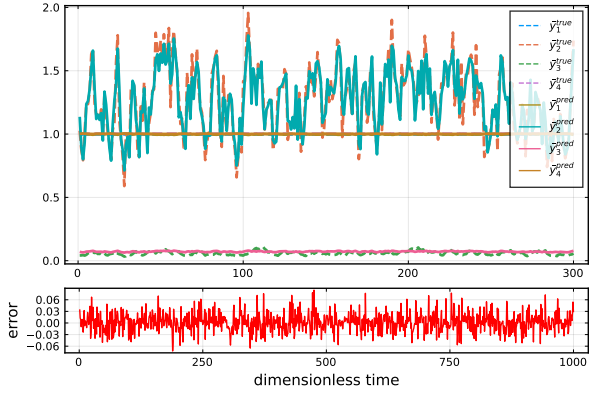
Left: DCRNN identified model in stage I. **Right**: Stage II FNN training.

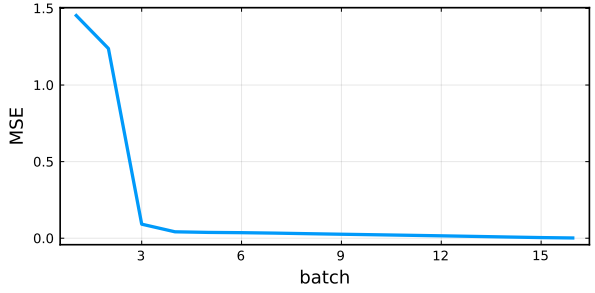




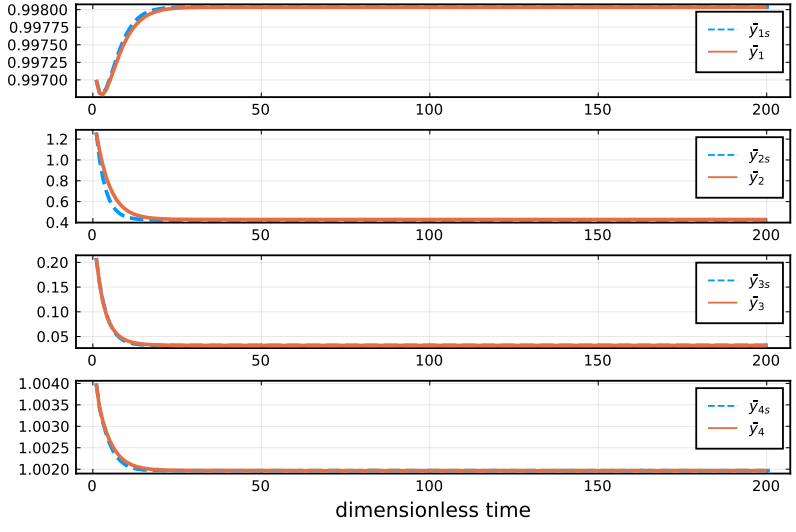
Real-time control performance. $G_r = [1, 0]$, $R_t = \text{logistic}_{10}(\hat{y}_t - 0.5)$, $\text{logistic}_{10}(x) \triangleq \frac{1}{1+e^{-10x}}$, $\sigma_u = \tanh$ (+ *bucketing* for inference), $\sigma_h =$ relu.

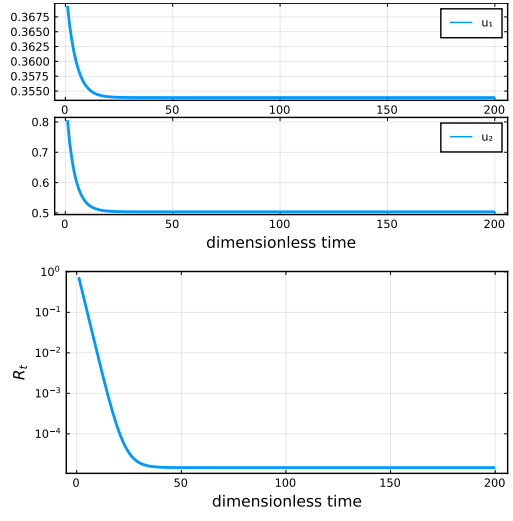
Full RCNN simulation results (ethylene oxidation)





Left: DCRNN identified model in stage I. **Right**: Stage II FNN training.





Real-time control performance. ; $G_r = [1, 1, 1, 1]$, $R_t = (\hat{y}_t - y^{ref})$, $\sigma_u = \tanh$, $\sigma_h =$ relu.

Conclusions

- designed and analyzed quasi-Newton optimization algorithms within a learning and control context
- addressed structural issues with RNNs
- proposed "self-concordant regularization" for quasi-Newton methods
- featured an approximation technique in GGN-SCORE that's useful for overparameterized models and mini-batch updates
- achieved faster convergence and better prediction quality than first-order methods
- proved the first (nonasymptotic, last-iterate) convergence result for GGN in optimizing neural networks with explicit regularization

Conclusions

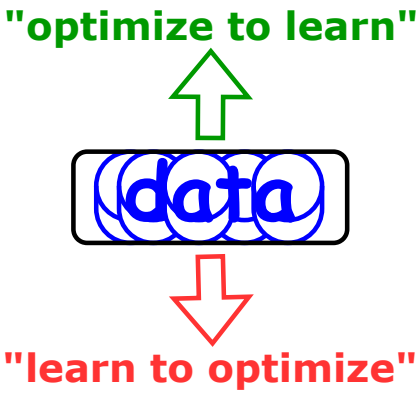
Future directions:

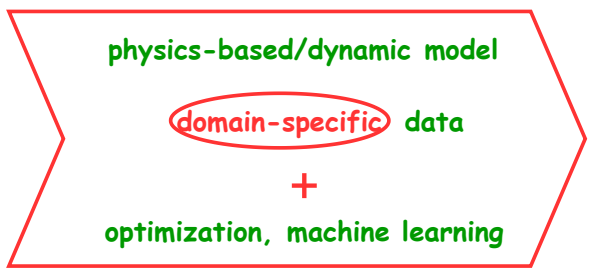
- extensions to explicitly handle more general constraints
- automatic or adaptive selection of optimal smoothing parameters in the proximal SCORE framework
- application and/or analysis of online/stochastic versions of the proximal SCORE framework
- explore parallel and distributed versions of quasi-Newton methods
- improve/solidify theoretical guarantees on the newly proposed techniques

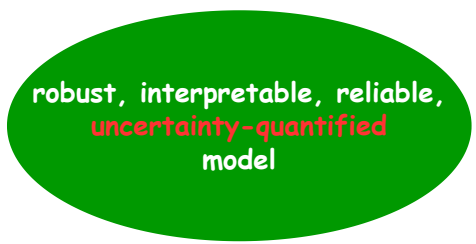
Conclusions

Future directions:

- uncertainty quantification in neural network predictions, e.g., (probabilistic) Bayesian neural networks, for reliability and robustness
- hybrid models (data + physics) for improved interpretability and robustness







- **applications**: large-scale industrial and engineering problems, e.g., computational fluid dynamics, optimal experimental design, optimal power flow, . . .

Thank you!

References

 Adeoye, Adeyemi D and Alberto Bemporad (2023a). "SCORE: approximating curvature information under self-concordant regularization". In: *Computational Optimization and Applications* 86.2, pp. 599–626.

 — (2023b). "Self-concordant Smoothing for Convex Composite Optimization". In: *arXiv preprint arXiv:2309.01781*.

 Adeoye, Adeyemi D. and Alberto Bemporad (2025). "An Inexact Sequential Quadratic Programming Method for Learning and Control of Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 36.2, pp. 2762–2776. DOI: 10.1109/TNNLS.2024.3354855.

 Adeoye, Adeyemi D, Philipp Christian Petersen, and Alberto Bemporad (2024). "Regularized Gauss-Newton for Optimizing Overparameterized Neural Networks". In: *arXiv preprint arXiv:2404.14875*.

References


Atiya, Amir F and Alexander G Parlos (2000). "New results on recurrent network training: unifying the algorithms and accelerating convergence". In: *IEEE transactions on neural networks* 11.3, pp. 697–709.


Becker, Stephen, Jalal Fadili, and Peter Ochs (2019). "On quasi-Newton forward-backward splitting: proximal calculus and convergence". In: *SIAM Journal on Optimization* 29.4, pp. 2445–2481.


Bellman, Richard (1952). "On the theory of dynamic programming". In: *Proceedings of the national Academy of Sciences* 38.8, pp. 716–719.


Bemporad, Alberto (2022). "Recurrent neural network training with convex loss and regularization functions by extended Kalman filtering". In: *IEEE Transactions on Automatic Control* 68.9, pp. 5661–5668.

References

- Bemporad, Alberto (2023). "Training recurrent neural networks by sequential least squares and the alternating direction method of multipliers". In: *Automatica* 156, p. 111183.
- Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.
- Berner, Julius et al. (2021). "The modern mathematics of deep learning". In: *arXiv preprint arXiv:2105.04026* 78.
- Burke, James V and Tim Hoheisel (2017). "Epi-convergence properties of smoothing by infimal convolution". In: *Set-Valued and Variational Analysis* 25, pp. 1–23.
- Deng, Jia et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.
- DeVore, Ronald A (1998). "Nonlinear approximation". In: *Acta numerica* 7, pp. 51–150.

References

- Duncan, William Jolly (1944). "LXXVIII. Some devices for the solution of large sets of simultaneous linear equations: With an appendix on the reciprocation of partitioned matrices". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 35.249, pp. 660–670.
- Guttman, Louis (1946). "Enlargement methods for computing the inverse matrix". In: *The annals of mathematical statistics*, pp. 336–343.
- Ha, David and Jürgen Schmidhuber (2018). "Recurrent world models facilitate policy evolution". In: *Advances in neural information processing systems* 31.
- Hochreiter, Sepp (1998). "Recurrent neural net learning and vanishing gradient". In: *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems* 6.2, pp. 107–116.

References

- Johnston, Barbara M et al. (2004). "Non-Newtonian blood flow in human right coronary arteries: steady state simulations". In: *Journal of biomechanics* 37.5, pp. 709–720.
- — (2006). "Non-Newtonian blood flow in human right coronary arteries: transient simulations". In: *Journal of biomechanics* 39.6, pp. 1116–1128.
- Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In.
- LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
- McCloud, Scott (1993). *Understanding Comics: The Invisible Art*. Tundra Publishing.
- Nemirovski, Arkadi (2004). "Interior point polynomial time methods in convex programming". In: *Lecture notes* 42.16, pp. 3215–3224.

References

- Nesterov, Yurii and Arkadii Nemirovskii (1994). *Interior-point polynomial algorithms in convex programming*. SIAM.
- Novak, Erich and Henryk Woźniakowski (2009). "Approximation of infinitely differentiable multivariate functions is intractable". In: *Journal of Complexity* 25.4, pp. 398–404.
- Panier, Eliane R and André L Tits (1991). "Avoiding the Maratos effect by means of a nonmonotone line search I. General constrained problems". In: *SIAM Journal on Numerical Analysis* 28.4, pp. 1183–1195.
- Powell, Michael JD (1978a). "A fast algorithm for nonlinearly constrained optimization calculations". In: *Numerical analysis*. Springer, pp. 144–157.
- — (1978b). "The convergence of variable metric methods for nonlinearly constrained optimization calculations". In: *Nonlinear programming 3*. Elsevier, pp. 27–63.

References


Rockafellar, R Tyrrell and Roger J-B Wets (2009). *Variational analysis*. Vol. 317. Springer Science & Business Media.


Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.


Saad, Youcef and Martin H Schultz (1986). "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3, pp. 856–869.


Sun, Tianxiao and Quoc Tran-Dinh (2019). "Generalized self-concordant functions: a recipe for Newton-type methods". In: *Mathematical Programming* 178.1-2, pp. 145–213.


Tumblr, Meme Engine (2012). *More thoughts from Understanding Comics by Scott McCloud*. Retrieved from Meme Engine Tumblr.

References

 Zimmermann, Hans-Georg et al. (2006). "Identification and forecasting of large dynamical systems by dynamical consistent neural networks". In: *New Directions in Statistical Signal Processing: From Systems to Brain*, pp. 203–242.