Optimization-Informed Neural Networks

a deep learning approach for solving constrained nonlinear optimization problems

Dawen Wu

Université Paris Saclay, CNRS, CentraleSupelec
Laboratoire des Signaux et des Systèmes (L2S)

Supervisor: Prof. Abdel Lisser

Sep 15^{th}, 2022

Outline of the talk

1. Introduction: Problems, Contributions
2. Preliminaries: Neurodynamic optimization, NN as ODE/PDE soltuion
3. Methodology: OINN model, OINN training
4. Experimental Results: Two Examples, Discussions
5. Conclusion

The standard CNLP has the following form

$$\begin{cases} \min_{\mathbf{x}} f(\mathbf{x}) \\ \text{s.t.} \\ \quad g(\mathbf{x}) \leq \mathbf{0}, \\ \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \end{cases} \tag{1}$$

The standard CNLP can be classified into different categories based on the properties of the objective and constraint functions. For example:

- When both $f(\mathbf{x})$ and $g(\mathbf{x})$ are both linear, it is called linear programming problem.
- When the $f(\mathbf{x})$ is quadratic, and the $g(\mathbf{x})$ is linear , it is called quadratic programming problem.
- When both $f(\mathbf{x})$ and $g(\mathbf{x})$ are convex , it is called convex optimization problem.
- When either $f(\mathbf{x})$ or $g(\mathbf{x})$ is non-smooth, it is called non-smooth optimization problem.

The contributions of our proposed OINN can be summarized as follows

- A deep learning approach, in the form of feed-forward neural networks, is proposed to solve CNLPs, which has never been done before in the long history of nonlinear programming.
- The CNLP solution problem becomes a neural network training problem. Such that, we can solve the CNLP by only deep learning infrastructure without using any standard optimization solvers or numerical integration solvers.
- In some examples, OINN outperforms conventional approaches in terms of accuracy and computational time.

Neurodynamic optimization is a method that model a CNLP by **an ODE system**.

Consider a CNLP with an optimal solution $\mathbf{y}^*$. A neurodynamic approach establishes a dynamical system in the form of a first-order ODE system, i.e., $\frac{d\mathbf{y}}{dt} = \Phi(\mathbf{y})$.

The state solution $\mathbf{y}(t)$ of this ODE system is expected to converge to the optimal solution of the CNLP, i.e., $\lim_{t\to\infty} \mathbf{y}(t) = \mathbf{y}^*$.

The general differential equation can be defined as

$$G\left(\mathbf{x}, f(\mathbf{x}), \nabla f(\mathbf{x}), \nabla^2 f(\mathbf{x})\right) = 0, \mathbf{x} \in D \tag{2}$$

The trail solution to solve the differential equation (2) is defined as

$$f_t(\mathbf{x}) = A(\mathbf{x}) + F(\mathbf{x}, N(\mathbf{x}, \mathbf{w})), \tag{3}$$

where $A(\cdot)$ and $F(\cdot, \cdot)$ are used to ensure the satisfaction of initial/boundary condition.

The training objective is defined as

$$\min_{\mathbf{w}} \int_{\mathbf{x} \in D} G\left(\mathbf{x}, f_t\left(\mathbf{x}, \mathbf{w}\right), \nabla f_t\left(\mathbf{x}, \mathbf{w}\right), \nabla^2 f_t\left(\mathbf{x}, \mathbf{w}\right)\right)^2 \tag{4}$$

.

The training is achieved by performing gradient descent on (4) with respect to the NN model parameter $\mathbf{w}$.

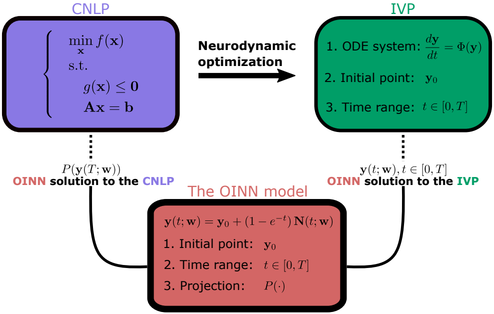
(A): Neurodynamic optimization

Authors	Problems
Hopfield et al.(1986)	Linear programming problems (Hopfield network)
Kennedy et al.(1988)	Nonlinear programming problems based on the penalty method
Xia et al.(2007)	Nonlinear projection equations
Qin et al.(2014)	Nonsmooth convex optimization problems
Xu et al.(2020)	Constrained pseudoconvex programming problems

(B): NN as solution of ODE/PDE:

Authors	Methods
Dissanayake et al (1994)	initially used a nn as an approximate solution to PDE
Lagaris et al. (1998)	constructed a nn to satisfy an initial/boundary condition
Han et al. (2018)	High dimensional PDE
Sirignano et al (2018)	DGM
Raissi et al. (2019)	PINN

These two lines of research are in two different communities. In a nutshell, **Our OINN is a combination of these two lines of research, which have never interacted in the last 30 years**



- The CNLP is first reformluated as an IVP by Neurodynamic optimization.
- The OINN model is defined as $\mathbf{y}(t; \mathbf{w}) = \mathbf{y}_0 + (1 - e^{-t})\mathbf{N}(t; \mathbf{w})$, where $\mathbf{N}(t; \mathbf{w})$ is a fully-connected network. The multiplier $(1 - e^{-t})$ is used to ensure the satisfaction of the initial condition.
- The endpoint $\mathbf{y}(T; \mathbf{w})$ is an approximate solution to the CNLP.
- The OINN model itself $\mathbf{y}(T; \mathbf{w})$ is an approximate state solution to the IVP.

The loss function is defined as

$$\mathcal{L}(t, \mathbf{w}) = e^{-\gamma * t} \left\| \frac{\partial \mathbf{y}(t; \mathbf{w})}{\partial t} - \Phi(\mathbf{y}(t; \mathbf{w})) \right\|, \tag{5}$$

where the $\Phi(\cdot)$ is the ODE system related to the CNLP.

The objective function is an integral of $\mathcal{L}(t, \mathbf{w})$ over the time range $[0, T]$

$$E(\mathbf{w}) = \int_0^T \mathcal{L}(t, \mathbf{w}) dt. \tag{6}$$

At each iteration, the OINN model train on **the batch loss**, defined as

$$\mathcal{L}(\mathbb{T}, \mathbf{w}) = \frac{1}{|\mathbb{T}|} \sum_{t \in \mathbb{T}} \mathcal{L}(t, \mathbf{w}), \tag{7}$$

where $\mathbb{T}$ is a set of time that sampled uniformly from the time range $[0, T]$. $\mathcal{L}(\mathbb{T}, \mathbf{w})$ is an unbiased estimate of $E(\mathbf{w})$.

OINN

OINN training algorithm

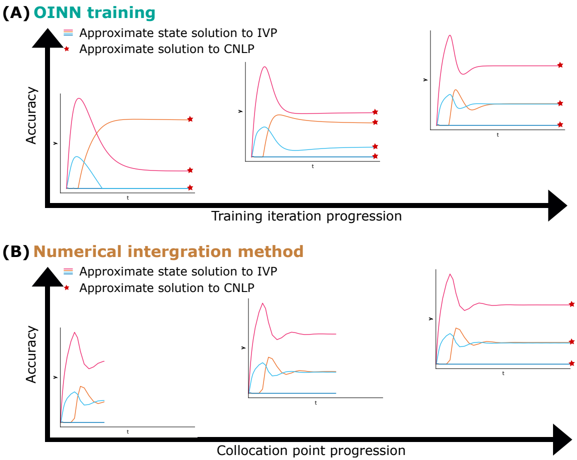
Algorithm 1: Training of an OINN model for solving a CNLP

Hyperparameters: An initial point $\mathbf{y}_0$, A time range $[0, T]$
Input : A CNLP
Output : The OINN model after training

Function Main:
  Derive the ODE system $\Phi(\cdot)$ corresponding to the CNLP by a neurodynamic optimization method.
  Initialize an OINN model $\mathbf{y}(t; \mathbf{w})$.
  Initialize $\epsilon_{\text{best}} = P\left(\mathbf{y}\left(T; \mathbf{w}\right)\right)$.
  while iter $\leq$ Max iteration **do**
    $\mathbb{T} \sim U(0, T)$: Uniformly sample a batch of t from the interval $[0, T]$.
    Forward propagation: Compute the batch loss $\mathcal{L}(\mathbb{T}, \mathbf{w})$.
    Backward propagation: Update $\mathbf{w}$ by $\nabla_{\mathbf{w}} \mathcal{L}(\mathbb{T}, \mathbf{w})$.
    Compute the epsilon value: $\epsilon_{\text{temp}} = P\left(\mathbf{y}\left(T; \mathbf{w}\right)\right)$.
    if $\epsilon_{\text{temp}} < \epsilon_{\text{best}}$ **then**
      $\epsilon_{\text{best}} = \epsilon_{\text{temp}}$
      Save the OINN model with parameters $\mathbf{w}$
    end
  end
end

The epsilon metric ϵ is used to evaluate how well the OINN prediction to the CNLP.

Throughout the training process, the algorithm maintains the lowest epsilon value, namely ϵ_{best}, representing the best prediction to the CNLP, and the corresponding model parameter is saved.



OINN can provide approximations for the IVP and the CNLP at any training iteration, while the numerical method can only produce solutions at the end of the program.

Example 1. Consider the following convex-smooth standard CNLP:

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^2 + 2x_2^2 + 2x_1x_2 - 10x_1 - 12x_2$$
$$\text{s.t.}$$
$$g_1(\mathbf{x}) = x_1 + 3x_2 - 8 \leq 0 \tag{8}$$
$$g_2(\mathbf{x}) = x_1^2 + x_2^2 + 2x_1 - 2x_2 - 3 \leq 0$$
$$0 \leq \mathbf{x} \leq 2.$$

We define $G(\mathbf{y})$ as

$$G(\mathbf{y}) = \begin{bmatrix} \nabla f(\mathbf{x}) + \nabla g(\mathbf{x})^T \mathbf{u} \\ -g(\mathbf{x}) \end{bmatrix}, \tag{9}$$

where $\mathbf{y} = [x_1, x_2, u_1, u_2]^T$; x_1, x_2 are decision variables, and u_1, u_2 are dual variables.

The CNLP can be reformulated as the following nonlinear projection equation

$$P_\Omega(\mathbf{y} - G(\mathbf{y})) = \mathbf{y}, \tag{10}$$

The ODE system models this NPE

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = -G\left(P_\Omega(\mathbf{y})\right) + P_\Omega(\mathbf{y}) - \mathbf{y}, \qquad (11)$$

The ODE system together with the initial point $\mathbf{y}_0 = [0,0,0,0]$ and time range $[0,10]$ form an IVP as follow

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = -G\left(P_\Omega(\mathbf{y})\right) + P_\Omega(\mathbf{y}) - \mathbf{y}, \quad \mathbf{y}_0 = [0,0,0,0], \quad t \in [0,10] \qquad (12)$$

An OINN model, $\mathbf{y}(t;\mathbf{w})$ $t \in [0,10]$, is built as an approximate state solution to this IVP (12).

Its endpoint $P_\Omega\left(\mathbf{y}(10;\mathbf{w})\right)$ is an approximate solution to the NPE (10), hence solving Example 1.

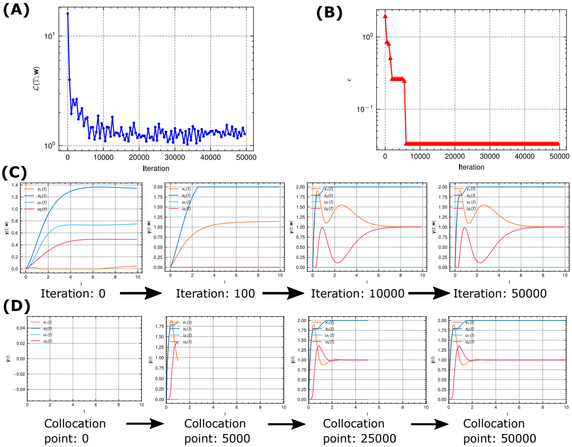


Figure: **Example 1: Convex-smooth standard CNLP (A) The loss versus the number of iterations. (B) The epsilon value versus the number of iterations. (C) The solving process of the OINN model (D) The solving process of the numerical integration method**

Index	OINN		Numerical integration method	
	Iteration	Solution	Collocation point	Solution
Example 1	0	[0.05, 1.34, 0.75, 0.49]	0	[0.00, 0.00, 0.00, 0.00]
	10	[0.84, 2.00, 0.00, 0.00]	10	[0.02, 0.02, 0.00, 0.00]
	100	[1.15, 2.00, 0.00, 0.00]	100	[0.19, 0.23, 0.00, 0.00]
	1000	[1.19, 2.00, 0.00, 0.00]	1000	[1.36, 1.42, 0.00, 0.00]
	10000	[1.00, 2.00, 0.00, 1.00]	10000	[1.01, 2.00, 0.00, 0.97]
	50000	[1.00, 2.00, 0.00, 1.00]	50000	[1.00, 2.00, 0.00, 1.00]

Table: **Example 1, Approximate solutions during solving**

- Both OINN and the numerical integration method converge to the same optimal solution.
- OINN is able to give a good prediction in early stage of training. For instance, at the 100th iteration, the prediction given by OINN is already very close to the optimal value.

Example 2 Consider the following pseudoconvex nonsmooth CNLP

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) = \frac{x_1 + x_2 + e^{|x_2 - 1|} - 40}{(x_1 + x_2 + x_3)^2 + 3} \\ &\text{s.t.} \\ &\quad g_1(\mathbf{x}) = -3x_1 + 2x_2 - 5 \leq 0 \\ &\quad g_2(\mathbf{x}) = x_1^2 + x_2 - 3 \leq 0 \\ &\quad h(\mathbf{x}) = x_1 + 2x_2 + x_3 - 2 = 0 \end{aligned} \tag{13}$$

This example is difficult to solve by standard optimization solvers or numerical integration solvers because of its pseudo-convex and non-smooth properties.

Numerical results

Example 2: pseudoconvex nonsmooth standard CNLP

Index	OINN		Numerical integration method	
	Iteration	Objective value ↓	Collocation point	Objective value ↓
Example 2	0	-11.757	0	-7.871
	10	-11.757	10	-7.871
	100	-11.861	100	-7.871
	1000	-11.914	1000	-7.871
	10000	**-11.992**	10000	inf
	50000	**-11.992**	50000	-11.985

Table: Comparison of objective values

OINN is able to reach a lower objective value, i.e. OINN finds a better solution than the numerical integration method.

Index	OINN		Numerical integration methods						
	Iteration	CPU time	Collocation point	RK45 CPU time	RK23 CPU time	DOP853 CPU time	Radau CPU time	BDF CPU time	LSODA CPU time
Example 2	10	202 ms	10	1350 ms	860 ms	3470 ms	1000 ms	Fail	157 ms
	100	893 ms	100	1740 ms	1090 ms	4620 ms	1330 ms	Fail	154 ms
	1000	8.47 s	1000	2.14s	1.32s	5.68 s	1.47 s	Fail	188 ms
	10000	1min 20s	10000	1min 25s	5min 5s	34min 29s	Fail	Fail	4h 4min 35s
	50000	**7min 55s**	50000	14min 29s	25min 14s	1h 43min 28s	Fail	Fail	Fail

Table: Comparison of computational time

- RK45, RK23, DOP853, Radau, BDF and LSODA are six different numerical integration methods used for comparison.
- **OINN outperforms all these six methods in terms of computational CPU time**, i.e., OINN takes 7min 55s while RK45 takes at best 14min 29s.
- The three methods, Radau, BDF, and LSODA, fail to solve this problem.

Conclusion and future directions

Conclusions:

- In this paper, we presented a deep learning approach to solve constrained nonlinear optimization problems (CNLP), namly OINN.
- OINN is a combination of two line of research, i.e., Neurodynamic optimization and neural network for solving PDE.
- We propose a dedicated algorithm to train the OINN model toward solving the CNLP.
- We demonstrate the effectiveness of OINN with two examples.

Future directions:

- From the problem side, OINN can be extended to many other optimization problems by working with other neurodynamic approaches.
- From the methodological side, we can further improve the computational performance of OINN by incorporating research from the broad machine learning community.

Thank you for your attention

Reference:

1. Wu, D., Lisser, A. (2022). A dynamical neural network approach for solving stochastic two-player zero-sum games. Neural Networks 152: 140-149.

2. Wu, D., Lisser, A. (2022). Using CNN for solving two-player zero-sum games, Expert Systems With Applications 204:117545.

3. Wu, D., Lisser, A. (2022). MG-CNN: A Deep CNN To Predict Saddle Points Of Matrix Games. In press, accepted in Neural Networks.

4. Wu, D., Lisser, A. (2022). Optimization-Informed Neural Networks: a deep learning approach for solving constrained nonlinear optimization problems, submited to Neural Networks (1st Major)