On Hamilton-Jacobi partial differential equation and architecctures of neural networks

Jérôme Darbon

Division of Applied Mathematics, Brown University

Third Monterey Workshop on Computational Issues in Nonlinear Control
October 9, 2019

Joint work with Tingwei Meng and Gabriel Provencher Langlois
Work supported by NSF DMS 1820821

Context and motivation

- Consider the initial value problem

$$\begin{cases} \frac{\partial S}{\partial t}(x,t) + H(\nabla_x S(x,t), x, t) \; = \; \varepsilon \triangle S(x,t), & \text{in } \mathbb{R}^n \times (0,+\infty) \\ S(x,0) \; = \; J(x) & \forall x \in \mathbb{R}^n. \end{cases}$$

- Goals: compute the viscosity solution for a given (x, t)
  - evaluate $S(x, t)$ and $\nabla_x S(x, t)$
  - very high dimension, possibly $n \geq 10^6$
  - fast to allow applications requiring real-time
  - low memory and low energy for embedded system
- Several approaches to mitigate/overcome the curse of differentiability: e.g., Max-plus methods, tensor decomposition methods, sparse grids, optimization techniques via representation formulas, ...
- More recently, there is a significant trend in using Machine Learning and Neural Network techniques for solving PDEs
  $\rightarrow$ leverage universal approximation theorems

Neural Network: a computational point of view

- Pros and cons of Neural Networks for evaluating solutions
  - It seems to be hard to find Neural Networks that are interpretable, generalization that yield reproducible results
  - Huge computational advantage
    - dedicated hardware for NN is now available: e.g., Xilinx AI (FPGA + silicon desing), Intel AI (FPGA + new CPU assembly instructions), and many other (startup) companies
    - high throughput / low latency (more precise meaning of “fast”)
    - low energy requirement (e.g., a few Watts)

    $\rightarrow$ suitable for embedded computing and data centers
- Can we leverage these computational resources for high-dimensional H-J PDEs?
- How can we mathematically certify that Neural Networks (NNs) actually computes a solution?

$\Rightarrow$ Establish new connections between NN architectures and representation formulas of H-J PDE solutions

$\rightarrow$ the physics of some H-J PDEs can be encoded by NN architecture

$\rightarrow$ the parameters of the NN define Hamiltonians and initial data

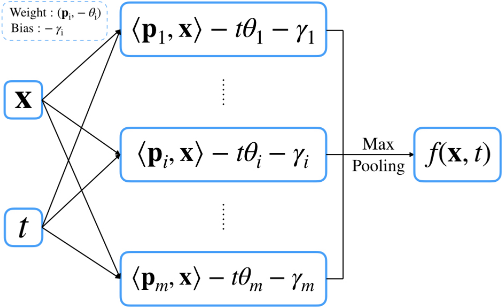
$\rightarrow$ no approximation: exact evaluation of $S(x,t)$ and $\nabla_x S(x,t)$

$\rightarrow$ suggests an interpretation of some NN architectures in terms of H-J PDE

Outline

1. Shallow NN architectures and representation of solution of H-J PDEs
    1. A class of first-order H-J
    2. Associated conservation law (1D)
    3. A class of second order H-J
2. (Briefly) “Deep” NN architectures for other H-J PDEs
3. Some conclusions

A first shallow network architecture



- Architecture: fully connected layer followed by the activation function "max-pooling"
- This network defines a function $f : \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$

$$f(\boldsymbol{x}, t; \{\boldsymbol{p}_i, \theta_i, \gamma_i\}_{i=1}^m) = \max_{i \in \{1,\dots,m\}} \{\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - t\theta_i - \gamma_i\}.$$

Goal: Find conditions on the parameters such that f satisfies a PDE, and find the PDE

Assumption on the parameters

- Recall: the network $f(\boldsymbol{x}, t; \{\boldsymbol{p}_i, \theta_i, \gamma_i\}_{i=1}^m) = \max_{i\in\{1,\dots,m\}}\{\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - t\theta_i - \gamma_i\}$
- We adopt the following assumptions on the parameters:
  - (A1) The parameters $\{\boldsymbol{p}_i\}_{i=1}^m$ are pairwise distinct, i.e., $\boldsymbol{p}_i \neq \boldsymbol{p}_j$ if $i \neq j$.
  - (A2) There exists a convex function $g\colon\ \mathbb{R}^n \to \mathbb{R}$ such that $g(\boldsymbol{p}_i) = \gamma_i$.
  - (A3) For any $j \in \{1, \dots, m\}$ and any $(\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ that satisfy

$$\begin{cases} (\alpha_1, \dots, \alpha_m) \in \Delta_m \text{ with } \alpha_j = 0, \\ \sum_{i\neq j} \alpha_i \boldsymbol{p}_i = \boldsymbol{p}_j, \\ \sum_{i\neq j} \alpha_i \gamma_i = \gamma_j, \end{cases}$$

    there holds $\sum_{i\neq j} \alpha_i\theta_i > \theta_j$.

  where Δ_m denotes the unit simplex of dimension m
- (A1) and (A3) are NOT strong assumptions.
  - (A3) simply states the each “neuron” should contribute to the definition of f.
  - If (A3) is not satisfied, then it means that some neurons can be removed and the NN still define the same function f

Define initial data and Hamiltonians from parameters

- Recall: the network f

$$f(\boldsymbol{x}, t; \{\boldsymbol{p}_i, \theta_i, \gamma_i\}_{i=1}^m) = \max_{i \in \{1,\dots,m\}} \{\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - t\theta_i - \gamma_i\} \tag{1}$$

- Define the initial data J using the NN parameters $\{\boldsymbol{p}_i, \gamma_i\}_{i=1}^m$

$$f(\boldsymbol{x}, 0) = J(\boldsymbol{x}) := \max_{i \in \{1,\dots,m\}} \{\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - \gamma_i\} \tag{2}$$

Then, $J : \mathbb{R}^n \to \mathbb{R}$ is convex, and its Legendre transform J^* reads

$$J^*(\boldsymbol{p}) = \begin{cases} \min_{\substack{(\alpha_1,\dots,\alpha_m) \in \Delta_m \\ \sum_{i=1}^m \alpha_i \boldsymbol{p}_i = \boldsymbol{p}}} \left\{\sum_{i=1}^m \alpha_i \gamma_i\right\}, & \text{if } \boldsymbol{p} \in \operatorname{conv}(\{\boldsymbol{p}_i\}_{i=1}^m), \\ +\infty, & \text{otherwise.} \end{cases}$$

Denote by $\mathcal{A}(\boldsymbol{p})$ is the minimizers in the above optimization problem.

- Define the Hamiltonian $H \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ by

$$H(\boldsymbol{p}) := \begin{cases} \inf_{\alpha \in \mathcal{A}(\boldsymbol{p})} \left\{\sum_{i=1}^m \alpha_i \theta_i\right\}, & \text{if } \boldsymbol{p} \in \operatorname{dom} J^*, \\ +\infty, & \text{otherwise.} \end{cases} \tag{3}$$

NN computes viscosity solutions

Theorem

Assume (A1)-(A3) hold. Let f be the neural network defined by Eq. (1) with parameters $\{(\boldsymbol{p}_i, \theta_i, \gamma_i)\}_{i=1}^m$. Let J and H be the functions defined in Eqs. (2) and (3), respectively, and let $\tilde{H}\colon \mathbb{R}^n \to \mathbb{R}$ be a continuous function. Then the following two statements hold.

(i) *The neural network f is the unique uniformly continuous viscosity solution to the Hamilton–Jacobi equation*

$$\begin{cases} \frac{\partial f}{\partial t}(\boldsymbol{x}, t) + H(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, t)) = 0, & \boldsymbol{x} \in \mathbb{R}^n, t > 0, \\ f(\boldsymbol{x}, 0) = J(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{4}$$

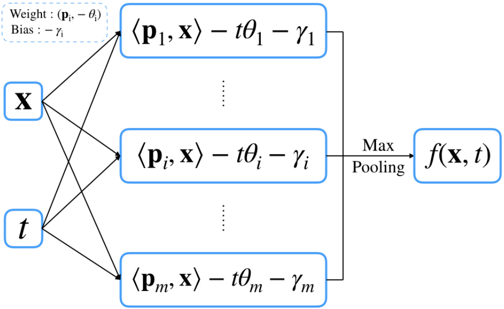
Moreover, f is jointly convex in $(\boldsymbol{x}, t)$.

(ii) *The neural network f is the unique uniformly continuous viscosity solution to the Hamilton–Jacobi equation*

$$\begin{cases} \frac{\partial f}{\partial t}(\boldsymbol{x}, t) + \tilde{H}(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, t)) = 0, & \boldsymbol{x} \in \mathbb{R}^n, t > 0, \\ f(\boldsymbol{x}, 0) = J(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{5}$$

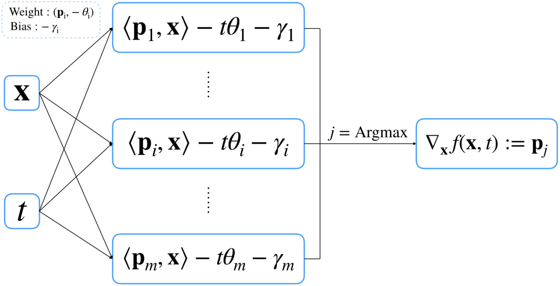
if and only if $\tilde{H}(\boldsymbol{p}_i) = H(\boldsymbol{p}_i)$ for every $i = 1, \ldots, m$ and $\tilde{H}(\boldsymbol{p}) \geqslant H(\boldsymbol{p})$ for every $\boldsymbol{p} \in \mathrm{dom}\, J^*$.

NN computes viscosity solutions



- The network the computes viscosity solution for H and J given by parameters
- Hamiltonians are not unique. However, among all possible Hamiltonians, H is the smallest one.
- In addition, $\nabla_x S(x,t)$ (when it exists) is given by the element that realizes the maximum is the “max-pooling”

Architecture for that gradient map



- This NN architecture computes the spatial gradient of the solution (i.e., the momemtum)
- Consider $u : \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}^n$ defined by

$$\nabla_x f(x,t) = p_j, \text{ where } j \in \underset{i \in \{1,\ldots,m\}}{\arg\max} \left\{ \langle p_i, x \rangle - t\theta_i - \gamma_i \right\}. \tag{6}$$

Theorem

Consider the one-dimensional case, i.e., $n = 1$. Suppose assumptions (A1)-(A3) hold. Let $u := \nabla_x f$ be the function from $\mathbb{R} \times [0, +\infty)$ to $\mathbb{R}$ defined in Eq. (6). Let J and H be the functions defined in Eqs. (2) and (3), respectively, and let $\tilde{H} \colon \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz continuous function. Then the following two statements hold.

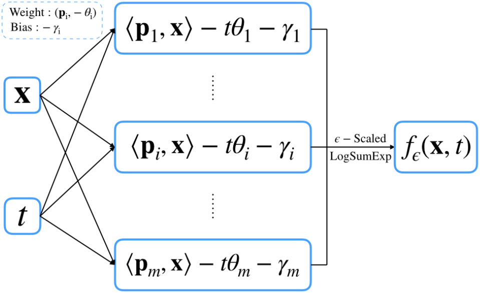
(i) *The neural network u is the entropy solution to the conservation law*

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) + \nabla_x H(u(x,t)) = 0, & x \in \mathbb{R}, t > 0, \\ u(x,0) = \nabla J(x), & x \in \mathbb{R}. \end{cases} \tag{7}$$

(ii) *The neural network u is the entropy solution to the conservation law*

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) + \nabla_x \tilde{H}(u(x,t)) = 0, & x \in \mathbb{R}, t > 0, \\ u(x,0) = \nabla J(x), & x \in \mathbb{R}, \end{cases} \tag{8}$$

if and only if there exists a constant $C \in \mathbb{R}$ such that $\tilde{H}(p_i) = H(p_i) + C$ for every $i \in \{1, \ldots, m\}$ and $\tilde{H}(p) \geqslant H(p) + C$ for any $p \in \operatorname{conv} \{p_i\}_{i=1}^m$.

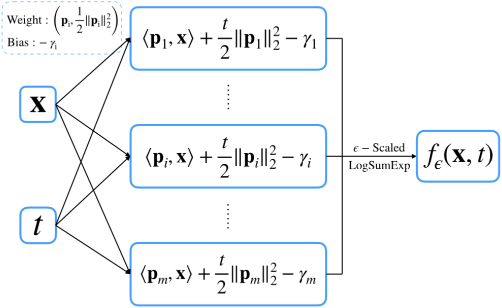


- Replace “max-pooling” by “smooth max pooling”.
- This network defines a function $f : \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$

$$f_\epsilon(\boldsymbol{x}, t) := \epsilon \log \left(\sum_{i=1}^{m} e^{(\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - t\theta_i - \gamma_i)/\epsilon} \right).$$

Specialize this architecture

Specialize the parameters: $\theta_i = -\frac{1}{2}\|p_i\|_2^2 \quad$ for $i = 1, \ldots, m$



- This network defines a function $f : \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$

$$f_\epsilon(\boldsymbol{x}, t) := \epsilon \log\left(\sum_{i=1}^{m} e^{\left(\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle + \frac{t}{2}\|\boldsymbol{p}_i\|_2^2 - \gamma_i\right)/\epsilon}\right). \qquad (9)$$

NN computes viscosity solutions of some second order H-J PDEs

Theorem

Let f_ϵ defined by (9) with parameters $\{\boldsymbol{p}_i, \theta_i, \gamma_i\}_{i=1}^m$ and let $\theta_i = -\frac{1}{2}\|\boldsymbol{p}_i\|_2^2$ for $i \in \{1, \ldots, m\}$. For every $\epsilon > 0$, the neural network $f_\epsilon \equiv \epsilon \log(w_\epsilon)$ is the unique smooth solution to the second-order Hamilton–Jacobi equation

$$\begin{cases} \frac{\partial f_\epsilon(\boldsymbol{x},t)}{\partial t} - \frac{1}{2}\|\nabla_{\boldsymbol{x}} f_\epsilon(\boldsymbol{x},t)\|_2^2 = \frac{\epsilon}{2}\triangle_{\boldsymbol{x}} f_\epsilon(\boldsymbol{x},t) & \text{in } \mathbb{R}^n \times (0,+\infty), \\ f_\epsilon(\boldsymbol{x},0) = \epsilon \log\left(\sum_{i=1}^m e^{(\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle - \gamma_i)/\epsilon}\right) & \forall \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{10}$$

Moreover, f_ϵ is jointly convex in $(\boldsymbol{x}, t)$ the following holds

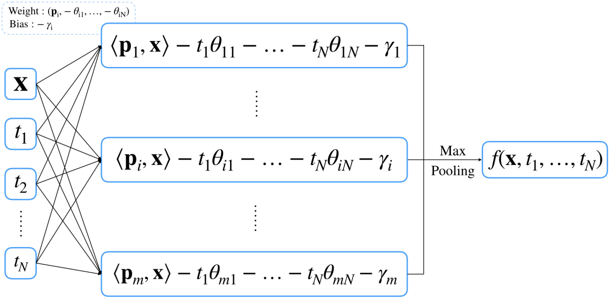
$$\lim_{\substack{\epsilon \to 0 \\ \epsilon > 0}} f_\epsilon(\boldsymbol{x}, t) = \max_{i \in \{1,\ldots,m\}} \{\langle \boldsymbol{p}_i, \boldsymbol{x}\rangle + \frac{t}{2}\|\boldsymbol{p}_i\|_2^2 - \gamma_i\} \tag{11}$$

holds for every $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geqslant 0$. Finally, if assumptions (A1)-(A3) hold, then the right hand side of (11) solves the first-order Hamilton–Jacobi equation (5) with $\tilde{H} := -\frac{1}{2}\|\cdot\|_2^2$.

Summary and extension to other PDEss

- We have exhibited classes of network architecture that represents viscosity solution of some H-J PDEs
- Initial data and Hamiltonians are induced by the parameters of the network
- These architectures can be extended to cope with other PDEs
  We briefly present architectures and ideas for other PDEs
  - (similar) architectures to multi-time H-J PDEs
  - "Deep" ResNet-based architecture and method of characteristics

Extension to Multi-time H-J PDE

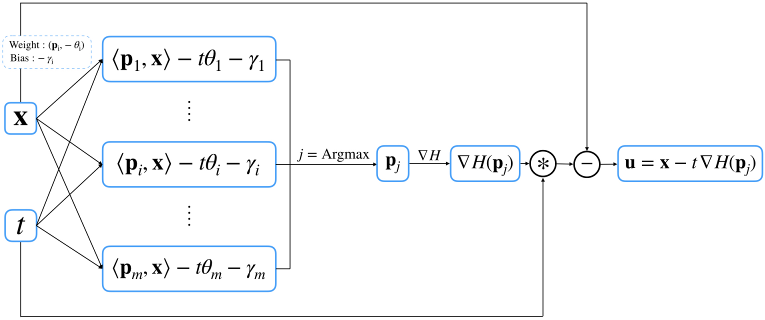


- The network defines a function $f : \mathbb{R}^n \times [0, +\infty)^N \to \mathbb{R}$ which satisfies the following multi-time H-J PDE

$$\begin{cases} \dfrac{\partial f}{\partial t_i}(x,t) + H_i(\nabla_x f(x,t)) = 0 & \text{for } i = 1, \ldots, N \\ f(x, 0, \ldots, 0) = J(x) & \text{for every } x \in \mathbb{R}^n \end{cases}$$

- Not necessarily a viscosity solution. It depends if the semi-groups associated to each time commute.

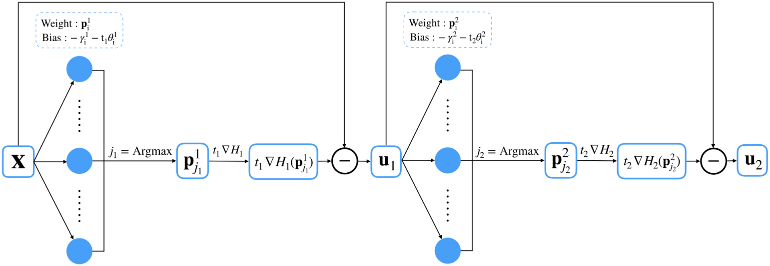
ResNet Variants, Generalized Moreau-Yosida identities and Lax-Oleinik formula



- Generalized Moreau identity (convex case): $x = u(x,t) + t\nabla H(p(x,t))$
  $\to$ p is a maximizer of the Hopf formula
  $\to$ u is a miminizer of the Lax-Oleinik formula
- The network defines the map $u : \mathbb{R}^n \times (0, +\infty) \to \mathbb{R}^n$
- Therefore all information for the characteristics

A Deep NN architecture

- We can fixed a sequence of time t_i and Hamiltonian H_i and iterates the previous map. This gives the following “deep” ResNet-based NN (2 layers)



- So we have $x - t_1\nabla H(p_1) - t_2\nabla H(p_2) = u_2(x,t)$
- Under appropriate assumptions these architectures compute the characteristics and can be used for solving some H-J PDE with certain Hamiltonian with state and time dependence.

Conclusion

- Some NN architectures represent viscosity solutions of certain H-J PDEs in very high-dimensions
- Hamiltonian and initial data are given by the parameters
- "Chaining" these architectures + ResNet pave the way to cope with more general Hamiltonians
- We also used these NN architectures for "solving" some inverse problems involving H-J PDEs (not presented here)
  $\rightarrow$ "learning" corresponds to solving non-convex optimization problem
  $\rightarrow$ Implementation using TensorFlow
  $\rightarrow$ Numerical results show that "standard" optimization methods (e.g., ADAM) can provide excellent or terrible results