Deep Learning with Nontrivial Constraints

Buyun Liang, Ryan de Vera, and Ju Sun
Computer Science & Engineering, University of Minnesota
Tim Mitchell
Queens College, City University of New York

Apr 29, 2023

UNIVERSITY OF MINNESOTA
Driven to Discover℠

Presenters

Buyun Liang
(CS&E, UMN)

Ryan de Vera
(CS&E, UMN)

Prof. Tim Mitchell
(CS, Queens C.)

Prof. Ju Sun
(CS&E, UMN)

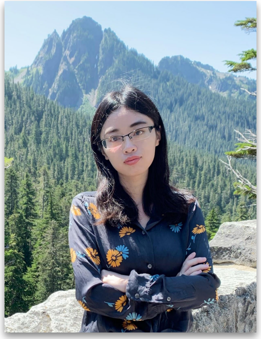
Contributors

Hengyue Liang
(ECE, UMN)

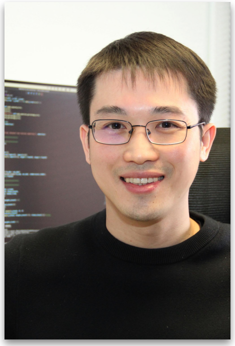
Wenjie Zhang
(CS&E, UMN)

Yash Travadi
(CS&E, UMN)

Le Peng
(CS&E, UMN)



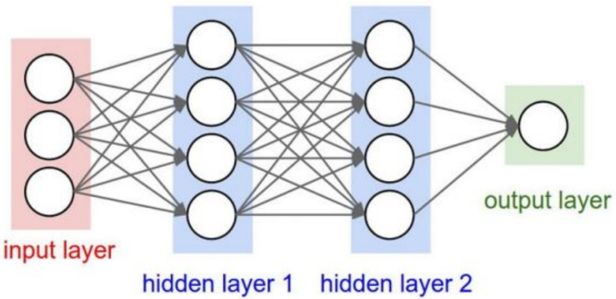
Prof. Ying Cui
(ISyE, UMN)

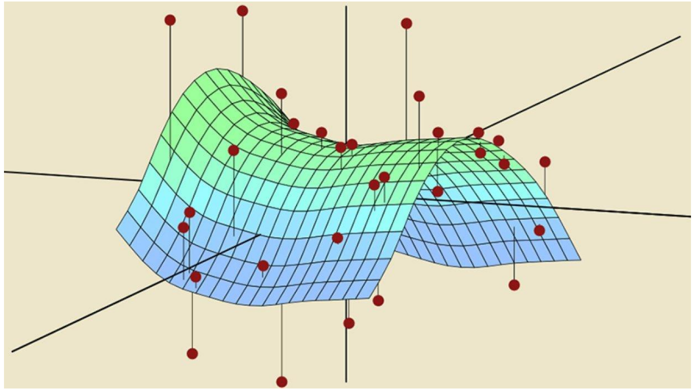


Prof. Qizhi He
(CSGE, UMN)

Deep learning (DL)

Artificial neural networks



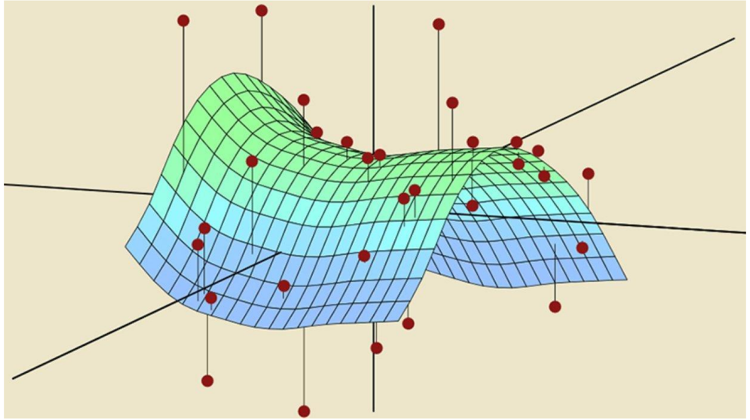


used to approximate nonlinear functions

Typical supervised learning pipeline

Step	General view	NN view
1	Gather training set $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$	Gather training set $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$
2	Choose a family of functions, e.g., $\mathcal{H}$, so that there is an $f \in \mathcal{H}$ to ensure $\boldsymbol{y}_i \approx f(\boldsymbol{x}_i), \forall i$	Choose a NN with k neurons, so that there is a group of weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)$ ensuring $\boldsymbol{y}_i \approx \{\text{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i), \forall i$
3	Set up a loss function ℓ	Set up a loss function ℓ
4	Find an $f \in \mathcal{H}$ to minimize the average loss $\frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{y}_i, f(\boldsymbol{x}_i))$	Find weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)$ to minimize the average loss $\frac{1}{n}\sum_{i=1}^{n} \ell[\boldsymbol{y}_i, \{\text{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i)]$

Three fundamental questions in DL



- **Approximation**: is it powerful, i.e., the $\mathcal{H}$ large enough for all possible weights?

Universal approximation theorems

- **Optimization**: how to solve

$$\min_{\boldsymbol{w}_i's, \boldsymbol{b}_i's} \frac{1}{n} \sum_{i=1}^{n} \ell \left[\boldsymbol{y}_i, \{\mathsf{NN}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k)\}(\boldsymbol{x}_i) \right]$$

- **Generalization**: does the learned NN work well on "similar" data?

Isn't solved?

Base class

CLASS torch.optim.Optimizer(*params*, *defaults*) [SO

Base class for all optimizers.

• WARNING

Parameters need to be specified as collections
consistent between runs. Examples of objects t
and iterators over values of dictionaries.

Parameters:

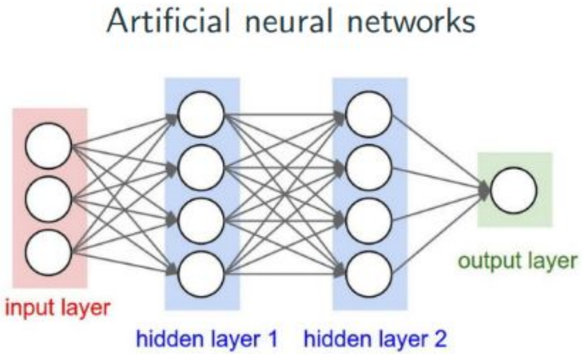
- **params** (*iterable*) – an iterable of to
  Tensors should be optimized.
- **defaults** – (dict): a dict containing d
  when a parameter group doesn't spe

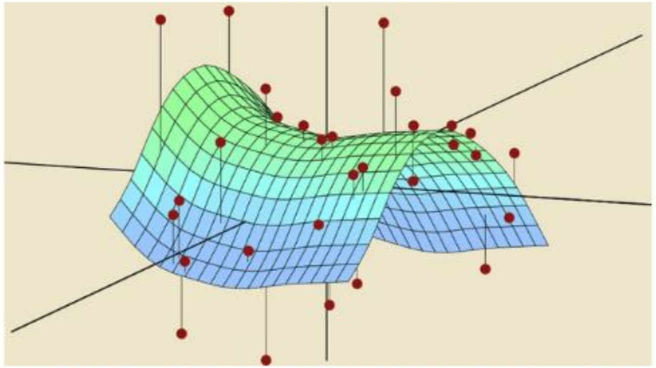
Algorithms

Adadelta	Implements Adadelta algorithm.
Adagrad	Implements Adagrad algorithm.
	lgorithm

Adamax	Implements Adamax algorithm (a variant of Adam based on infinity norm).
ASGD	Implements Averaged Stochastic Gradient Descent.
LBFGS	Implements L-BFGS algorithm, heavily inspired by minFunc.
NAdam	Implements NAdam algorithm.
RAdam	Implements RAdam algorithm.

When DL meets constraints





used to approximate nonlinear functions

Unconstrained optimization

$$\min_{\boldsymbol{w}_i's, \boldsymbol{b}_i's} \frac{1}{n} \sum_{i=1}^{n} \ell \left[\boldsymbol{y}_i, \{ \mathsf{NN} \left(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k, b_1, \dots, b_k \right) \} \left(\boldsymbol{x}_i \right) \right]$$

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x})$$

"Solved"

Constrained optimization

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) \quad \text{s. t.} \ \ g(\boldsymbol{x}) \le \boldsymbol{0}$$

largely "unsolved"

This tutorial:

Constrained optimization

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t.} \;\; g(\boldsymbol{x}) \leq \mathbf{0}$$

largely "unsolved"

how to solve DL problems with constraints

Left: "DL problems with constraints" in DALL-E's mind

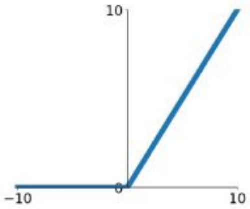
Outline

Constrained deep learning: CDL

- **What and how for CDL**
- Why CDL
- No good solvers for CDL yet
- Granso and PyGranso
- PyGranso in action
- Outlook

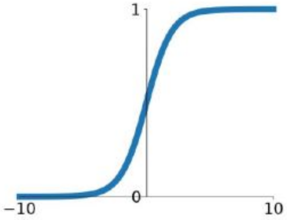
DL with simple constraints

Embedding constraints into DL models



ReLU
(Rectified Linear Unit)

Nonnegativity



Sigmoid

[0, 1]

$$\boldsymbol{z} \mapsto \left[\frac{e^{z_1}}{\sum_j e^{z_j}}, \ldots, \frac{e^{z_p}}{\sum_j e^{z_j}} \right]^{\mathsf{T}}$$

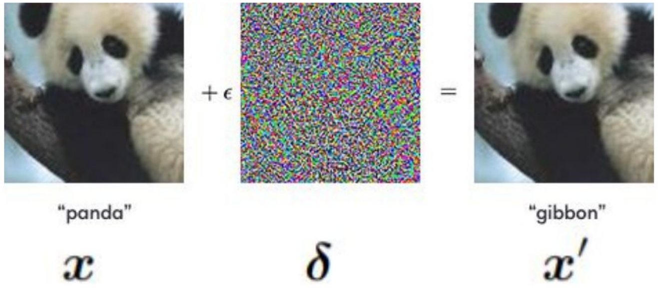
Softmax

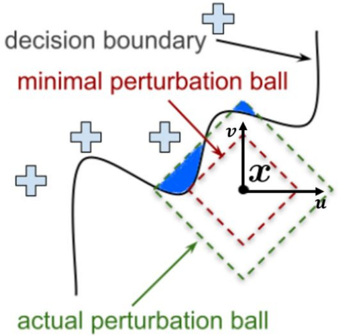
Nonnegativity and summed to 1

DL with nontrivial constraints

- **Robustness evaluation**
- Imbalanced learning
- Physics-informed neural networks (PINNs)

Robustness evaluation (RE)





Maximize loss function

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s. t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

Allowable perturbation   Valid image

Minimize robustness radius

$$\min_{\boldsymbol{x}'} d\left(\boldsymbol{x}, \boldsymbol{x}'\right)$$

$$\text{s. t. } \max_{i \neq y} f_{\boldsymbol{\theta}}^{i}(\boldsymbol{x}') \geq f_{\boldsymbol{\theta}}^{y}(\boldsymbol{x}'), \quad \boldsymbol{x}' \in [0,1]^n$$

Change the predicted class   Valid image

Ref Optimization and Optimizers for Adversarial Robustness. Liang, H., Liang, B., Peng, L., Cui, Y., Mitchell, T., & Sun, J. https://arxiv.org/abs/2303.13401

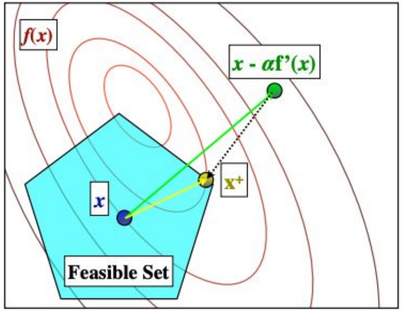
Projected gradient descent (PGD) for RE

$$\min_{\mathbf{x} \in \mathcal{Q}} f(\mathbf{x})$$

Step size

$$\mathbf{x}_{k+1} = P_{\mathcal{Q}}\Big(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\Big)$$

$$P_{\mathcal{Q}}(\mathbf{x}_0) = \arg\min_{\mathbf{x} \in \mathcal{Q}} \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|_2^2$$ **Projection operator**



Key hyperparameters:

(1) step size

(2) iteration number

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

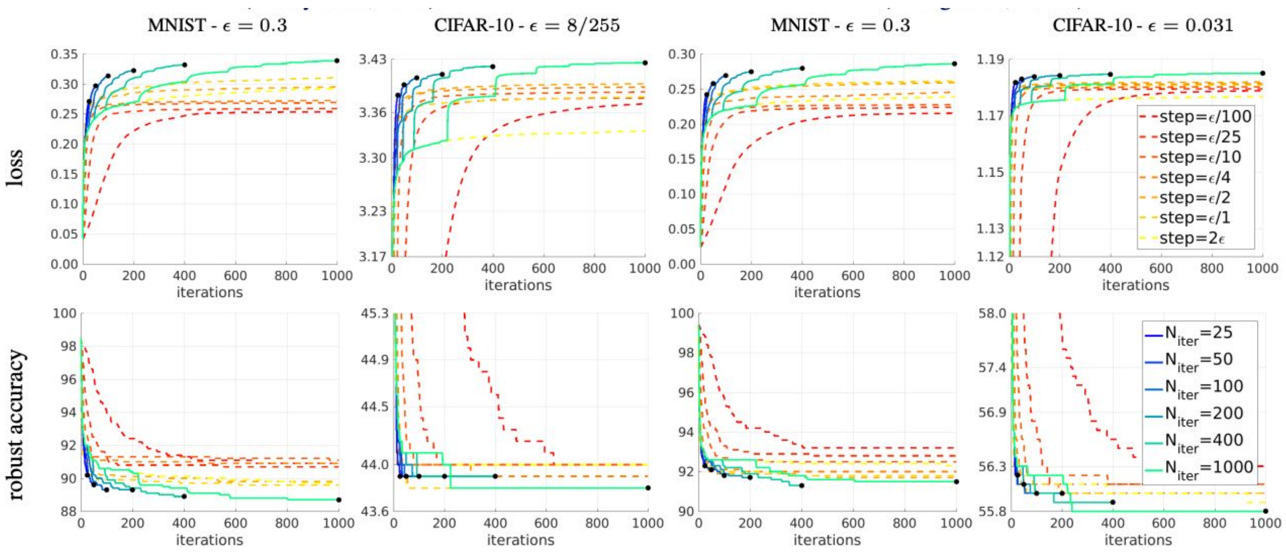
$$\text{s.t. } d(\boldsymbol{x}, \boldsymbol{x}') \leq \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \dots, w_n\}$
Output: $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_S\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_S\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_S\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  end if
  if $k \in W$ **then**
    if Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\max}$
    end if
  end if
end for

Ref https://angms.science/doc/CVX/CVX_PGD.pdf
https://www.cs.ubc.ca/~schmidtm/Courses/5XX-S20/S5.pdf
Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. Croce, F., Hein, M., ICML 2020 https://arxiv.org/pdf/2003.01690.pdf

Problem with projected gradient descent



$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon\ , \quad \boldsymbol{x}' \in [0,1]^n$$

Tricky to set:
iteration number & step size
i.e., tricky to decide where to stop

Ref Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. Croce, F., Hein, M., ICML 2020
https://arxiv.org/pdf/2003.01690.pdf

Robustness evaluation: penalty methods for complicated d (perceptual attack)

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon\,, \quad \boldsymbol{x}' \in [0,1]^n$$

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2$$

$$\text{where} \quad \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x}) \,]$$

perceptual distance

Projection onto the constraint is complicated

Penalty methods

$$\max_{\widetilde{\mathbf{x}}} \quad \mathcal{L}(f(\widetilde{\mathbf{x}}), y) - \lambda \max\left(0, \|\phi(\widetilde{\mathbf{x}}) - \phi(\mathbf{x})\|_2 - \epsilon\right)$$

Solve it for each fixed λ and then increase λ

Algorithm 2 Lagrangian Perceptual Attack (LPA)

procedure LPA(classifier network $f(\cdot)$, LPIPS distance $d(\cdot,\cdot)$, input $\mathbf{x}$, label y, bound ϵ)
    $\lambda \leftarrow 0.01$
    $\widetilde{\mathbf{x}} \leftarrow \mathbf{x} + 0.01 * \mathcal{N}(0,1)$ ▷ initialize perturbations with random Gaussian noise
    for i in $1,\ldots,S$ **do** ▷ we use $S = 5$ iterations to search for the best value of λ
        for t in $1,\ldots,T$ **do** ▷ T is the number of steps
            $\Delta \leftarrow \nabla_{\widetilde{\mathbf{x}}}\left[\mathcal{L}(f(\widetilde{\mathbf{x}}), y) - \lambda \max\left(0, d(\widetilde{\mathbf{x}}, \mathbf{x}) - \epsilon\right)\right]$ ▷ take the gradient of (5)
            $\hat{\Delta} = \Delta / \|\Delta\|_2$ ▷ normalize the gradient
            $\eta = \epsilon * (0.1)^{t/T}$ ▷ the step size η decays exponentially
            $m \leftarrow d(\widetilde{\mathbf{x}}, \widetilde{\mathbf{x}} + h\hat{\Delta})/h$ ▷ $m \approx$ derivative of $d(\widetilde{\mathbf{x}}, \cdot)$ in the direction of $\hat{\Delta}$; $h = 0.1$
            $\widetilde{\mathbf{x}} \leftarrow \widetilde{\mathbf{x}} + (\eta/m)\hat{\Delta}$ ▷ take a step of size η in LPIPS distance
        end for
        if $d(\widetilde{\mathbf{x}}, \mathbf{x}) > \epsilon$ **then**
            $\lambda \leftarrow 10\lambda$ ▷ increase λ if the attack goes outside the bound
        end if
    end for
    $\widetilde{\mathbf{x}} \leftarrow \text{PROJECT}(d, \widetilde{\mathbf{x}}, \mathbf{x}, \epsilon)$
    return $\widetilde{\mathbf{x}}$
end procedure

Ref Perceptual adversarial robustness: Defense against unseen threat models. Laidlaw, C., Singla, S., & Feizi, S. https://arxiv.org/abs/2006.12655

Problem with penalty methods

Method	cross-entropy loss		margin loss	
	Viol. (%) ↓	Att. Succ. (%) ↑	Viol. (%) ↓	Att. Succ. (%) ↑
Fast-LPA	73.8	3.54	41.6	56.8
LPA	**0.00**	80.5	**0.00**	97.0
PPGD	5.44	25.5	**0.00**	38.5
PWCF (ours)	0.62	93.6	**0.00**	100

$$\max_{\boldsymbol{x}'} \ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}'))$$
$$\text{s.t. } d(\boldsymbol{x}, \boldsymbol{x}') \le \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2$$
$$\text{where} \quad \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x}) \,]$$

LPA, Fast-LPA: penalty methods **PPGD**: Projected gradient descent

Penalty methods tend to encounter **large constraint violation** (i.e., infeasible solution, known in optimization theory) or **suboptimal solution**

PWCF, an optimizer with a principled stopping criterion on **stationarity & feasibility**

Ref Optimization and Optimizers for Adversarial Robustness. Liang, H., Liang, B., Peng, L., Cui, Y., Mitchell, T., & Sun, J. arXiv preprint arXiv:2303.13401.

Robustness evaluation: quick summary

Two forms of RE

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon\,, \quad \boldsymbol{x}' \in [0,1]^n$$

$$\min_{\boldsymbol{x}'} \ d\left(\boldsymbol{x}, \boldsymbol{x}'\right)$$

$$\text{s.t. } \max_{i \neq y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \geq f^y_{\boldsymbol{\theta}}(\boldsymbol{x}')\,, \ \boldsymbol{x}' \in [0,1]^n$$

Two methods for handling constraints

projected gradient descent

$$\min_{\mathbf{x} \in \mathcal{Q}} f(\mathbf{x})$$

$$\mathbf{x}_{k+1} = P_{\mathcal{Q}}\Big(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\Big)$$

Issue: no principled stopping criterion/step size rules

penalty methods

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) \quad \text{s.t. } g(\boldsymbol{x}) \leq \mathbf{0}$$

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) + \lambda \max(0, g(\boldsymbol{x}))$$

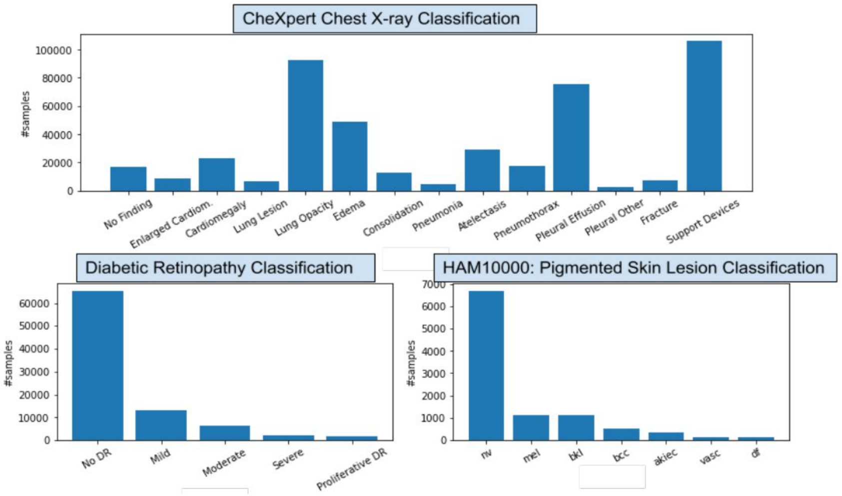
Solved with increasing λ sequence

Issue: infeasible solution

DL with nontrivial constraints

- Robustness evaluation
- **Imbalanced learning**
- Physics-informed neural networks (PINNs)

Imbalanced learning: background



Class imbalance in healthcare datasets

	Predicted POS	Predicted NEG
POS	70	30
NEG	1000	9000

Accuracy: 9070/10100 = 0.898
True Positive Rate (Sensitivity, Recall): 0.7
True Negative Rate (Specificity): 0.9
Balanced Accuracy: (0.7 + 0.9)/2 = 0.80
Precision (POS): 70/1070 = 0.065
F1 Score: 2*0.065*0.7/(0.065 + 0.7) = 0.119

Reliable evaluation in imbalanced learning: **precision needed**!

Imbalanced learning: direct metric optimization

Typical learning objective: $\min_{f \in \mathcal{H}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{D}_{\boldsymbol{x},\boldsymbol{y}}} \mathbb{1}\left\{\boldsymbol{y} \neq f(\boldsymbol{x})\right\}$ **accuracy maximization**

$$\text{precision}(f_{\boldsymbol{\theta}}, t) = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}}{\sum_{i=1}^{N} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}} \qquad \text{recall}(f_{\boldsymbol{\theta}}, t) = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}}{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\}}$$

$$F_\beta(f_{\boldsymbol{\theta}}, t) = (1+\beta^2) \frac{\text{precision}(f_{\boldsymbol{\theta}}, t) \cdot \text{recall}(f_{\boldsymbol{\theta}}, t)}{\beta^2 \text{precision}(f_{\boldsymbol{\theta}}, t) + \text{recall}(f_{\boldsymbol{\theta}}, t)}$$

$$\text{AP}(f_{\boldsymbol{\theta}}) = \frac{1}{|\{i : y_i = +1\}|} \sum_{i=1}^{N} \mathbb{1}\{y_i = +1\} \frac{\sum_{s=1}^{N} \mathbb{1}\{y_s = +1\} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_s) > f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}}{\sum_{s=1}^{N} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_s) > f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}}.$$

fix precision, optimize recall (FPOR): $\max_{\boldsymbol{\theta}, t} \ \text{recall}(f_{\boldsymbol{\theta}}, t) \quad \text{s.t. } \text{precision}(f_{\boldsymbol{\theta}}, t) \geq \alpha,$

fix recall, optimize precision (FROP): $\max_{\boldsymbol{\theta}, t} \ \text{precision}_t \quad \text{s.t. } \text{recall}(f_{\boldsymbol{\theta}}, t) \geq \alpha,$

optimize F_β score (OFBS): $\max_{\boldsymbol{\theta}, t} \ F_\beta(f_{\boldsymbol{\theta}}, t),$

optimize AP (OAP): $\max_{\boldsymbol{\theta}} \ \text{AP}(f_{\boldsymbol{\theta}}).$

Imbalanced learning: Lagrangian methods

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t.} \; g(\boldsymbol{x}) \le \mathbf{0}$$

$$\Updownarrow$$

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \ge \mathbf{0}} \; f(\boldsymbol{x}) + \boldsymbol{\lambda}^\mathsf{T} g(\boldsymbol{x})$$

Idea: alternating minimize $\boldsymbol{x}$ and maximize λ via gradient descent

Reminder on gradient descent

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x})$$

iteration step :

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - t \nabla f(\boldsymbol{x}_k)$$

$$\max_{\boldsymbol{\theta}, t} \; \text{recall}(f_{\boldsymbol{\theta}}, t) \quad \text{s.t.} \; \text{precision}(f_{\boldsymbol{\theta}}, t) \ge \alpha,$$

$$\max_{f,b} \quad \frac{1}{|Y^+|} tp(f)$$

$$s.t. \quad tp(f) \ge \alpha(tp(f) + fp(f)).$$

$$f^{(t+1)} = f^{(t)} - \gamma \nabla L(f^{(t)}, \lambda^{(t)})$$

$$\lambda^{(t+1)} = \lambda^{(t)} + \gamma \nabla L(f^{(t+1)}, \lambda^{(t)})$$

where

$$L(f, \lambda) = (1 + \lambda)\mathscr{L}^+(f) + \lambda \frac{\alpha}{1 - \alpha} \mathscr{L}^-(f) - \lambda |Y^+|.$$

Eban, Elad, et al. "Scalable learning of non-decomposable objectives." *Artificial intelligence and statistics*. PMLR, 2017.

Imbalanced learning: quick summary

fix precision, optimize recall (FPOR): $\max_{\theta,t} \text{recall}(f_\theta, t) \quad \text{s.t.} \ \text{precision}(f_\theta, t) \geq \alpha,$

fix recall, optimize precision (FROP): $\max_{\theta,t} \text{precision}_t \quad \text{s.t.} \ \text{recall}(f_\theta, t) \geq \alpha,$

Lagrangian method

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq 0} f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top g(\boldsymbol{x})$$

Idea: alternating minimize $\boldsymbol{x}$ and maximize λ via gradient descent

Issues

- Infeasible solution
- Slow convergence

DL with nontrivial constraints

- Robustness evaluation
- Imbalanced learning
- **Physics-informed neural networks (PINNs)**

PINNs: DL for PDEs

Physics-informed neural networks (PINNs)

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega,$$

$$\mathcal{B}(u, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega,$$

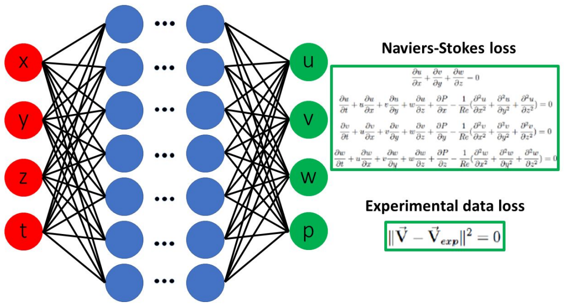
Penalty parameters

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \boxed{w_f} \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + \boxed{w_b} \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b)$$

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) \right\|_2^2$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2,$$

U is represented as a DNN



Continuous modeling instead of finite-difference for derivatives

Ref Liang, Buyun, Tim Mitchell, and Ju Sun. "NCVX: A general-purpose optimization solver for constrained machine and deep learning." *arXiv preprint arXiv:2210.00973* (2022).

wiki https://en.wikipedia.org/wiki/Physics-informed_neural_networks

PINNs: methods

Typical methods

$$\min_{u(\boldsymbol{x})} \mathcal{L}(u(\boldsymbol{x})) \quad \text{s. t.} \begin{cases} f\left(\boldsymbol{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots\right) = 0, & \forall \boldsymbol{x} \in \Omega \\ \mathcal{B}(u, \boldsymbol{x}) = 0, & \forall \boldsymbol{x} \in \partial\Omega \end{cases}$$

- Penalty methods
- Lagrangian methods
- Augmented Lagrangian methods

→ Infeasible solution

+

- First-order solver

→ Low quality solution

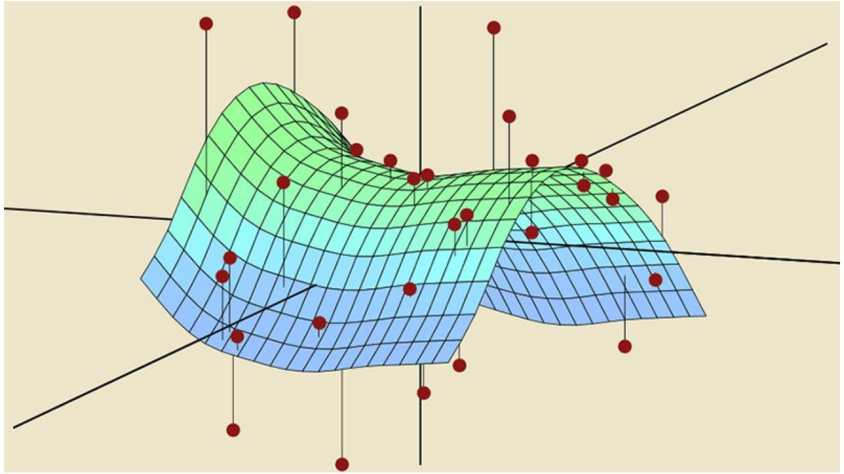
Outline

Constrained deep learning: CDL

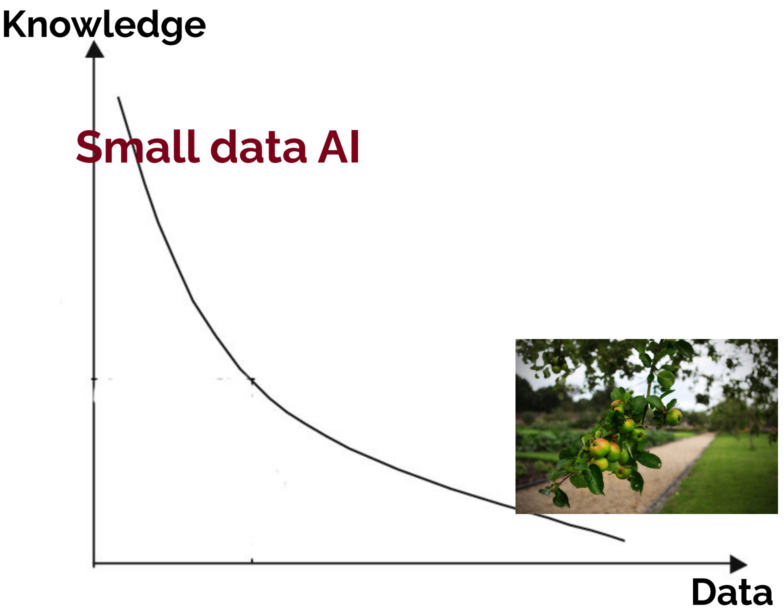
- What and how for CDL
- **Why CDL**
- No good solvers for CDL yet
- Granso and PyGranso
- PyGranso in action
- Outlook

There's no free lunch!

Supervised learning as data fitting



Typically, #data points we need grow exponentially with respect to dimension (i.e., curse of dimensionality)



Building in prior knowledge is <u>crucial</u> for reducing the data complexity
e.g., "convolutional" layers

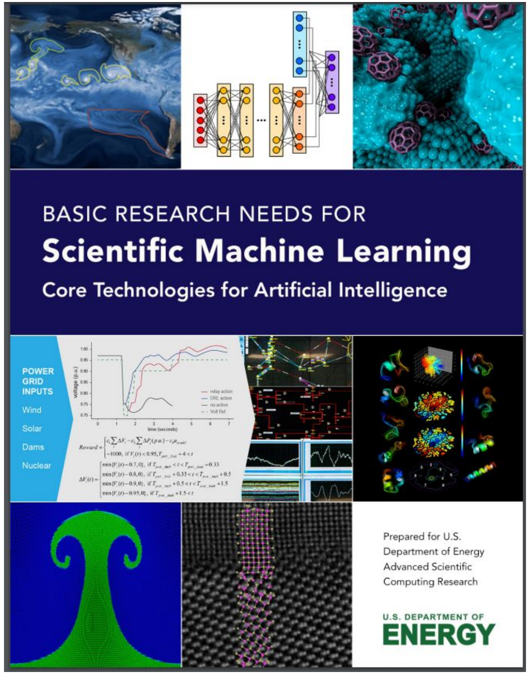
AI for science

Thrust B: How Should Domain Knowledge Be Incorporated into Supervised Machine Learning?

The central question for this thrust is "which knowledge should be leveraged in SciML, and how should this knowledge be included?" Any answers will naturally depend on the SciML task and computational budgets, thus mirroring standard considerations in traditional scientific computing.

Hard Constraints. One research avenue involves incorporation of domain knowledge through imposition of constraints that cannot be violated. These hard constraints could be enforced during training, replacing what typically is an unconstrained optimization problem with a constrained one. In general, such constraints could involve simulations or highly nonlinear functions of the training parameters. Therefore, there is a need to identify particular cases when constraint qualification conditions can be ensured as these conditions are necessary regularity conditions for constrained optimization [57–59]. Although incorporating constraints during training generally makes maximal use of training data, there may be additional opportunities to employ constraints at the time of prediction (e.g., by projecting predictions onto the region induced by the constraints).

Soft Constraints. A similar avenue for incorporating domain knowledge involves modifying the objective function (soft constraints) used in training. It is understood that ML loss function selection should be guided by the task and data. Therefore, opportunities exist for developing loss functions that incorporate domain knowledge and analyzing the resulting impact on solvability



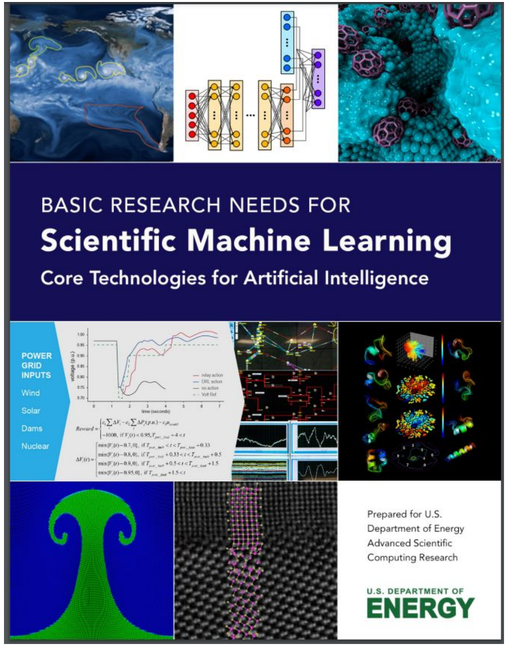
Ref https://www.osti.gov/servlets/purl/1478744

Domain-Aware Scientific Machine Learning

AI for science

Thrust C: How Should the Robustness, Performance, and Quality of Scientific Machine Learning Be Assessed?

The outcome of an ML process is either a decision (classification) or a prediction. For reliable and credible use of SciML, we need the ability to rigorously quantify ML performance in these outcomes. Performance measurement implies an assessment of quality, as well as a cost measure of computations and/or data preparation and management. Traditional measures of acceptable quality based on statistical cross-validation-type approaches often are heuristic. Measures of prediction quality such as *a priori* and *a posteriori* error estimates for numerical approximations of PDEs [96] (familiar to the finite element modeling community) will be transformative in allowing the development of optimal and reliable ML algorithms for different uses. Such error estimates also will enable SciML processes that allow iterative model improvement. Research establishing quantitative estimates of prediction quality, including effective confidence bounds, will greatly enhance the usefulness of SciML to decision makers and users. Finally, research is needed on algorithms that have proven convergence rates with weak dependence on bad data, especially in situations with a large amount of data of unproven quality or minimal availability of human expertise.



Ref https://www.osti.gov/servlets/purl/1478744

Robust Scientific Machine Learning

Outline

Constrained deep learning: CDL

- What and how for CDL
- Why CDL
- **No good solvers for CDL yet**
- Granso and PyGranso
- PyGranso in action
- Outlook

DL frameworks

JAX: Autograd and XLA

mxnet

PyTorch

TensorFlow

For unconstrained DL problems

Convex optimization solvers and frameworks

CVX RESEARCH

AMPL

Modeling languages

IBM CPLEX

GUROBI OPTIMIZATION

SDPT^3 - a MATLAB software package for semidefinite-quadratic-linear programming

K. C. Toh, R. H. Tütüncü, and M. J. Todd.

TFOCS: Templates for First-Order Conic Solvers

Solvers

Not for DL, which involves NCVX optimization

Note: Gurobi can handle certain NCVX problems

Manifold optimization

Manopt.jl

Geomstats

$\mathcal{T}_p\mathcal{G}$

geoopt

McTorch Lib, a manifold optimization library for deep learning

Only for **differentiable manifolds constraints**

General constrained optimization

KNITRO

COIN|OR

IPOPT

Interior-point methods

ensmallen
flexible C++ library for efficient numerical optimization

GENO

Augmented Lagrangian methods

Cooper

TensorFlow Constrained Optimization (TFCO)

Lagrangian-method-based constrained optimization

Specialized ML packages

scikit learn

Spark MLlib The Machine Learning Library

WEKA The University of Waikato

Problem-specific solvers that **cannot be easily extended** to new formulations

Outline

Constrained deep learning: CDL

- What and how for CDL
- Why CDL
- No good solvers for CDL yet
- **Granso and PyGranso**
- PyGranso in action
- Outlook

Issues with typical CDL methods

projected gradient descent

$$\min_{\mathbf{x} \in \mathcal{Q}} f(\mathbf{x})$$

$$\mathbf{x}_{k+1} = P_{\mathcal{Q}}\Big(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\Big)$$

Issue: no principled stopping criterion/step size rules

penalty methods

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t.} \;\; g(\boldsymbol{x}) \leq \mathbf{0}$$

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \lambda \max(0, g(\boldsymbol{x}))$$

Solved with increasing λ sequence

Issue: infeasible solution

Lagrangian method

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \; f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top g(\boldsymbol{x})$$

Idea: alternating minimize $\boldsymbol{x}$ and maximize λ via gradient descent

Issues

- Infeasible solution
- Slow convergence

Want

- **Feasible & stationary solution**
- **Reasonable speed**

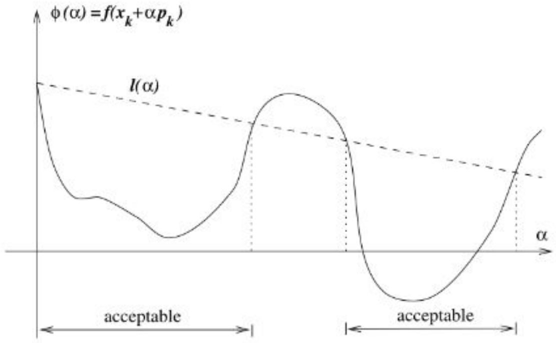
Principled answers to these questions

- **Feasible & stationary solution**

  Stationarity and feasibility check: KKT condition

- **Reasonable speed**

  Line search

  Armijo (Sufficient Decrease) Condition

- **A hidden problem: nonsmoothness**

A principled solver for constrained, nonconvex, nonsmooth problems

GRANSO

Nonconvex, nonsmooth, constrained

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}), \quad \text{s.t. } c_i(\boldsymbol{x}) \leq 0, \ \forall\, i \in \mathcal{I}; \quad c_i(\boldsymbol{x}) = 0, \ \forall\, i \in \mathcal{E}.$$

Penalty sequential quadratic programming (P-SQP)

$$\begin{aligned} \min_{d \in \mathbb{R}^n,\ s \in \mathbb{R}^p} \quad & \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d \\ \text{s.t.} \quad & c(x_k) + \nabla c(x_k)^\mathsf{T} d \leq s, \quad s \geq 0, \end{aligned}$$

Advantage: 2nd order method (BFGS) $\rightarrow$ high-precision solution

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Determining the search direction

Corresponding dual

$$\max_{\lambda \in \mathbb{R}^p} \quad \mu f(x_k) + c(x_k)^\mathsf{T}\lambda - \frac{1}{2}(\mu \nabla f(x_k) + \nabla c(x_k)\lambda)^\mathsf{T} H_k^{-1}(\mu \nabla f(x_k) + \nabla c(x_k)\lambda)$$
$$\text{s.t.} \quad 0 \leq \lambda \leq e, \qquad (8)$$

Primal solution (recovered from dual solution): searching direction

$$d_k = -H_k^{-1}(\mu \nabla f(x_k) + \nabla c(x_k)\lambda_k). \qquad (9)$$

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Adjusting the penalty parameter

GRANSO

Linear model of constraint violation

$$l(d;x_k) := \| \max\{c(x_k) + \nabla c(x_k)^\mathsf{T} d, 0\}\|_1$$

Corresponding reduction

$$\begin{aligned} l_\delta(d;x_k) &:= l(0;x_k) - l(d;x_k) \\ &= v(x_k) - \| \max\{c(x_k) + \nabla c(x_k)^\mathsf{T} d, 0\}\|_1 \end{aligned}$$

Advantage: feasibility guarantee

Procedure 1 $[d_k, \mu_{\text{new}}]$ = **sqp_steering_strategy**(x_k, H_k, μ)

Input:

Current iterate x_k and BFGS Hessian approximation H_k

Current value of the penalty parameter μ

Constants:

Values $c_v \in (0,1)$ and $c_\mu \in (0,1)$

Output:

Search direction d_k

Penalty parameter $\mu_{\text{new}} \in (0, \mu]$

```
1: Solve QP (8) using μ_new := μ to obtain search direction d_k from (9)
2: if l_δ(d_k; x_k) < c_v v(x_k) then
3:    Solve (8) using μ = 0 to obtain reference direction d̃_k from (9)
4:    while l_δ(d_k; x_k) < c_v l_δ(d̃_k; x_k) do
5:       μ_new := c_μ μ_new
6:       Solve QP (8) using μ := μ_new to obtain search direction d_k from (9)
7:    end while
8: end if
```

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Estimating the stationarity

Gradient from l most recent iterates

$$G := [\nabla f(x_{k+1-l}) \cdots \nabla f(x_k)]$$

$$J_i := [\nabla c_i(x_{k+1-l}) \cdots \nabla c_i(x_k)], \quad i \in \{1, \dots, p\}$$

Augmented QP

$$\max_{\sigma \in \mathbb{R}^l, \lambda \in \mathbb{R}^{pl}} \sum_{i=1}^{p} c_i(x_k) e^\mathsf{T} \lambda_i - \frac{1}{2} \begin{bmatrix} \sigma \\ \lambda \end{bmatrix}^\mathsf{T} [G, J_1, \dots, J_p]^\mathsf{T} H_k^{-1} [G, J_1, \dots, J_p] \begin{bmatrix} \sigma \\ \lambda \end{bmatrix}$$

$$\text{s.t.} \quad 0 \le \lambda_i \le e, \quad e^\mathsf{T} \sigma = \mu, \ \sigma \ge 0. \qquad (12)$$

Primal solution: termination condition

$$d_\diamond = H_k^{-1} [G, J_1, \dots, J_p] \begin{bmatrix} \sigma \\ \lambda \end{bmatrix}$$

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Estimating the stationarity

GRAΛ∇SO

Augmented QP

$$\max_{\sigma \in \mathbb{R}^l, \lambda \in \mathbb{R}^{pl}} \quad \sum_{i=1}^{p} c_i(x_k) e^{\mathsf{T}} \lambda_i - \frac{1}{2} \begin{bmatrix} \sigma \\ \lambda \end{bmatrix}^{\mathsf{T}} [G, J_1, \ldots, J_p]^{\mathsf{T}} H_k^{-1} [G, J_1, \ldots, J_p] \begin{bmatrix} \sigma \\ \lambda \end{bmatrix}$$

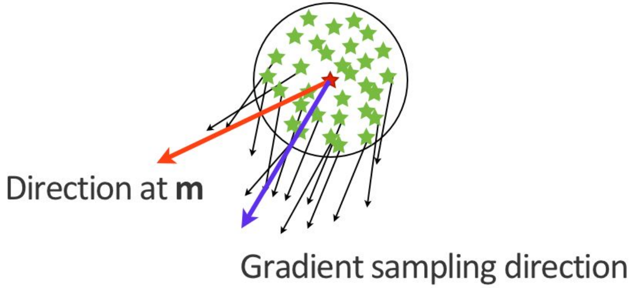
$$\text{s.t.} \quad 0 \leq \lambda_i \leq e, \quad e^{\mathsf{T}} \sigma = \mu, \quad \sigma \geq 0. \qquad (12)$$

Stationarity based on (approximate) gradient sampling

$$G_k := \begin{bmatrix} \nabla f(x^k) & \nabla f(x^{k,1}) & \cdots & \nabla f(x^{k,m}) \end{bmatrix}$$

$$\min_{\lambda \in \mathbb{R}^{m+1}} \ \frac{1}{2} \| G_k \lambda \|_2^2$$

$$\text{s.t.} \ \mathbb{1}^T \lambda = 1, \ \lambda \geq 0$$



Advantage: can handle nonsmoothness

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

GRANSO

Set $H_0 := I$ and $\mu := \mu_0$
Set $\phi(\cdot)$ as the penalty function given in (2) using $f(\cdot)$ and $c(\cdot)$
Set $\nabla\phi(\cdot)$ and $v(\cdot)$ as the associated gradient (4) and violation function (3)
Evaluate $\phi_0 := \phi(x_0;\mu)$, $\nabla\phi_0 := \nabla\phi(x_0;\mu)$, and $v_0 := v(x_0)$
for $k = 0, 1, 2, \ldots$ **do**
  $[d_k, \hat{\mu}] :=$ **sqp_steering_strategy**(x_k, H_k, μ)
  if $\hat{\mu} < \mu$ **then**
    // Penalty parameter has been lowered by steering; update current iterate
    Set $\mu := \hat{\mu}$
    Reevaluate $\phi_k := \phi(x_k;\mu)$, $\nabla\phi_k := \nabla\phi(x_k;\mu)$, and $v_k := v(x_k)$
  end if
  $[x_{k+1}, \phi_{k+1}, \nabla\phi_{k+1}, v_{k+1}] :=$ **inexact_linesearch**$(x_k, \phi_k, \nabla\phi_k, d_k, \phi(\cdot), \nabla\phi(\cdot))$
  Compute $d_\diamond$ via (12) and (13)
  if $\|d_\diamond\|_2 < \tau_\diamond$ and $v_{k+1} < \tau_v$ **then**
    // Stationarity and feasibility sufficiently attained; terminate successfully
    break
  end if
  Set H_{k+1} using BFGS update formula
end for

Advantages

- **Reliable step-size rule**
- **Principled stopping criterion**

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Key take-away

GRANSO

- Principled stopping criterion and line search, to obtain a **solution with certificate** (stationarity & feasibility check)
- Quasi-newton style method for fast convergence, i.e., **reasonable speed and high-precision solution**

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Limitations of GRANSO

```
% Gradient of inner product with respect to A
f_grad        = imag((conj(Bty)*Cx.')/(y'*x));
f_grad        = f_grad(:);

% Gradient of inner product with respect to A
ci_grad       = real((conj(Bty)*Cx.')/(y'*x));
ci_grad       = ci_grad(:);
```

analytical gradients required

```
p            = size(B,2);
m            = size(C,1);
X            = reshape(x,p,m);
```

vector variables only

Lack of Auto-Differentiation

Lack of GPU Support

No native support of tensor variables

⇒ impossible to do deep learning with GRANSO

GRANSO meets PyTorch

GRANSO + PyTorch

PyGRANSO

NCVX PyGRANSO Documentation

Search the docs ...

- Introduction
- Installation
- Settings
- Examples

Home

NCVX

NCVX Package

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \text{ s.t. } c_i(\mathbf{x}) \leq 0, \forall i \in \mathcal{I};\ c_i(\mathbf{x}) = 0, \forall i \in \mathcal{E}$$

First general-purpose solver for constrained DL problems

NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning

Buyun Liang, Tim Mitchell, Ju Sun

NCVX PyGRANSO: Advantages

NCVX

1) **Auto-Differentiation**

https://ncvx.org/

Orthogonal Dictionary Learning (ODL)

$$\min_{\boldsymbol{q}\in\mathbb{R}^n} f(\boldsymbol{q}) \doteq \frac{1}{m}\|\boldsymbol{q}^\intercal \boldsymbol{Y}\|_1, \quad \text{s.t. } \|\boldsymbol{q}\|_2 = 1$$

Analytical gradients

```
function[f,fg,ci,cig,ce,ceg]=comb_fn(q)
    f = 1/m*norm(q'*Y, 1); % obj
    fg = 1/m*Y*sign(Y'*q); % obj grad
    ci = []; cig = []; % no ineq constr
    ce = q'*q - 1; % eq constr
    ceg = 2*q; % eq constr grad
end
soln = granso(n,comb_fn);
```

Demo 1: GRANSO for ODL

```
def comb_fn(X_struct):
    q = X_struct.q
    f = 1/m*norm(q.T@Y, p=1) # obj
    ce = pygransoStruct()
    ce.c1 = q.T@q - 1 # eq constr
    return [f,None,ce]
var_in = {"q": [n,1]} # define variable
soln = pygranso(var_in,comb_fn)
```

Demo 2: PyGRANSO for ODL

No Analytical gradients

Ref Buyun Liang, Tim Mitchell, Ju Sun. NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning. In Neural Information Processing Systems (NeurIPS) Workshop on Optimization for Machine Learning (OPT 2022).

NCVX PyGRANSO: Advantages

NCVX

https://ncvx.org/

2) GPU acceleration for large scale problems

Orthogonality-constrained RNN

GPU: ~7.2 s for 100 iter

```
PyGRANSO: A PyTorch-enabled port of GRANSO with auto-differentiation
Version 1.2.0
Licensed under the AGPLv3, Copyright (C) 2021-2022 Tim Mitchell and Buyun Liang

Problem specifications:
 # of variables                     :   48010
 # of inequality constraints        :       0
 # of equality constraints          :       1

Limited-memory mode enabled with size = 20.
NOTE: limited-memory mode is generally NOT
recommended for nonsmooth problems.

     <--- Penalty Function -->                  Total Violation  <--- Line Search --->  <- Stationarity ->
Iter    Mu     |     Value         Objective    Ineq |    Eq     SD | Evals |    t      Grads |   Value
   0  100.0000 | 231.110993915  2.31110993915    -   | 2.20e-14  -  |     1 | 0.000000      1 | 70.02796
  10  2.781284 | 6.15638768205  1.83942004642    -   | 1.040438  S  |     3 | 4.000000      1 | 0.087806
  20  1.077526 | 2.42602824202  1.85198233657    -   | 0.430468  S  |     1 | 1.000000      1 | 0.009331
  30  0.785517 | 1.62062600991  1.84414672987    -   | 0.172018  S  |     3 | 4.000000      1 | 0.070584
  40  0.785517 | 1.49341762439  1.86152922129    -   | 0.031155  S  |     1 | 1.000000      1 | 0.008798
  50  0.785517 | 1.45863093500  1.84741128366    -   | 0.007458  S  |     1 | 1.000000      1 | 0.021082
  60  0.375710 | 0.65961089292  1.74086384511    -   | 0.005551  S  |     1 | 1.000000      1 | 0.060758
  70  0.375710 | 0.61368640026  1.62370589875    -   | 0.003644  S  |     1 | 1.000000      1 | 0.004978
  80  0.221853 | 0.34727899398  1.53948397613    -   | 0.005740  S  |     3 | 4.000000      1 | 0.060639
  90  0.221853 | 0.33322515702  1.49579349213    -   | 0.001379  S  |     3 | 0.250000      1 | 0.007120
 100  0.221853 | 0.32795759434  1.47195631688    -   | 0.001399  S  |     1 | 1.000000      1 | 0.020357

F = final iterate, B = Best (to tolerance), MF = Most Feasible
Optimization results:

   F           |                1.47195631688    -   | 0.001399      |       |               |
   B           |                2.31110993915    -   | 2.20e-14      |       |               |
  MF           |                2.31110993915    -   | 2.20e-14      |       |               |

Iterations:              100
Function evaluations:    148
PyGRANSO termination code: 4 --- max iterations reached.

Total Wall Time: 7.24015998840332s
```

CPU: ~17.6 s for 100 iter

```
PyGRANSO: A PyTorch-enabled port of GRANSO with auto-differentiation
Version 1.2.0
Licensed under the AGPLv3, Copyright (C) 2021-2022 Tim Mitchell and Buyun Liang

Problem specifications:
 # of variables                     :   48010
 # of inequality constraints        :       0
 # of equality constraints          :       1

Limited-memory mode enabled with size = 20.
NOTE: limited-memory mode is generally NOT
recommended for nonsmooth problems.

     <--- Penalty Function -->                  Total Violation  <--- Line Search --->  <- Stationarity ->
Iter    Mu     |     Value         Objective    Ineq |    Eq     SD | Evals |    t      Grads |   Value
   0  100.0000 | 234.459429436  2.32459429436    -   | 2.000000  -  |     1 | 0.000000      1 | 84.45258
  10  3.815204 | 7.61543767313  1.40009010433    -   | 2.273808  S  |     2 | 2.000000      1 | 0.035876
  20  1.478088 | 3.36426539638  1.18836657672    -   | 1.607755  S  |     2 | 2.000000      1 | 0.039562
  30  1.077526 | 2.36764743644  0.91175234571    -   | 1.385210  S  |     3 | 4.000000      1 | 0.122148
  40  0.572642 | 1.74741425646  0.89840965347    -   | 1.232947  S  |     2 | 2.000000      1 | 0.031523
  50  0.338139 | 1.47480385659  0.89839455270    -   | 1.171021  S  |     1 | 1.000000      1 | 0.022276
  60  0.221853 | 1.34061810155  0.87445955456    -   | 1.146617  S  |     3 | 4.000000      1 | 0.027327
  70  0.117902 | 1.23339816968  0.81824319678    -   | 1.136926  S  |     2 | 2.000000      1 | 0.029037
  80  0.077355 | 1.19367524620  0.80101161581    -   | 1.131713  S  |     1 | 1.000000      1 | 0.005670
  90  0.062658 | 1.17924531645  0.77380656707    -   | 1.130760  S  |     1 | 1.000000      1 | 0.007646
 100  0.029969 | 1.15301795796  0.76665514964    -   | 1.130042  S  |     2 | 2.000000      1 | 0.003602

F = final iterate, B = Best (to tolerance), MF = Most Feasible
Optimization results:

   F           |                0.76665514964    -   | 1.130042      |       |               |
  MF           |                0.76665514964    -   | 1.130042      |       |               |

Iterations:              100
Function evaluations:    182
PyGRANSO termination code: 4 --- max iterations reached.

Total Wall Time: 17.56377601623535s
```

Ref Buyun Liang, Tim Mitchell, Ju Sun. NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning. In Neural Information Processing Systems (NeurIPS) Workshop on Optimization for Machine Learning (OPT 2022).

NCVX PyGRANSO: Advantages

NCVX

https://ncvx.org/

3) General Tensor Variables

```
var_in = {"x1": [1], "x2": [1]}
```

Scalar input

```
# objective function
f = (8 * abs(x1**2 - x2) + (1 - x1)**2)
```

```
var_in = {"q": [n,1]}
```

Vector input

```
# objective function
qtY = q.T @ Y
f = 1/m * torch.norm(qtY, p = 1)
```

```
var_in = {"M": [d1,d2],"S": [d1,d2]}
```

Matrix inputs

```
# objective function
f = torch.norm(M, p = 'nuc') + eta * torch.norm(S, p = 1)
```

```
var_in = {"x_tilde": list(inputs.shape)}
```

Higher order tensor input

```
adv_inputs = X_struct.x_tilde
epsilon = eps
logits_outputs = model(adv_inputs)
f = -torch.nn.functional.cross_entropy(logits_outputs,labels)
```

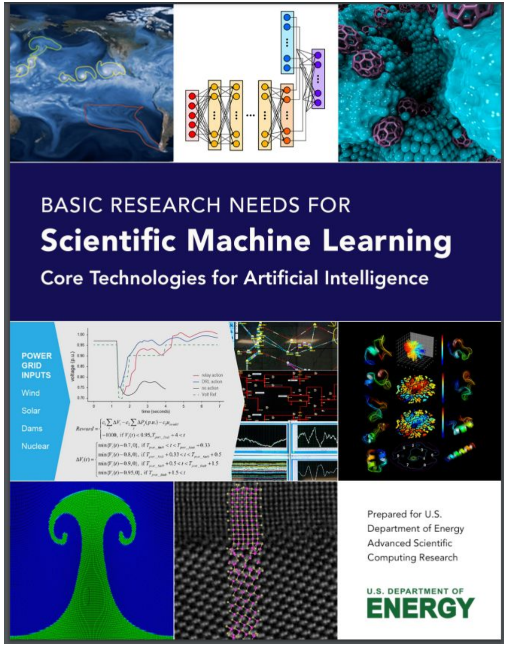
Ref Buyun Liang, Tim Mitchell, Ju Sun. NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning. In Neural Information Processing Systems (NeurIPS) Workshop on Optimization for Machine Learning (OPT 2022).

User-friendly is our philosophy

Answering DOE's call

Thrust C: How Should Domain Knowledge Be Modeled and Represented in Scientific Machine Learning?

An additional opportunity for domain-aware SciML research is in constructing modeling languages and frameworks that facilitate the inclusion of domain knowledge into the training process. Often, modeling languages and frameworks (e.g., [65, 66]) are designed to lower the barrier of entry for users by facilitating rapid and robust problem formulation. Extending the ways that SciML can express and incorporate domain knowledge could have far-reaching implications in much the same way that these tools now are regularly used for implicit features, such as algorithmic differentiation.



Ref https://www.osti.gov/servlets/purl/1478744

Robust Scientific Machine Learning

Constraint folding

(a) n box constraints **(b)** folded constraints

Reduce # constraints

- Reduce cost of QP in the SQP

Equality into non-negative inequality

$$h_j(\boldsymbol{x}) = 0 \Longleftrightarrow |h_j(\boldsymbol{x})| \leq 0$$

Inequality into nonnegative inequality

$$c_i(\boldsymbol{x}) \leq 0 \Longleftrightarrow \max\{c_i(\boldsymbol{x}), 0\} \leq 0$$

All non-negative inequalities into one

$$\mathcal{F}(|h_1(\boldsymbol{x})|, \cdots, |h_i(\boldsymbol{x})|, \max\{c_1(\boldsymbol{x}), 0\}, \cdots, \max\{c_j(\boldsymbol{x}), 0\}) \leq 0,$$

$\mathcal{F} : \mathbb{R}_+^{i+j} \mapsto \mathbb{R}_+ \; (\mathbb{R}_+ = \{\alpha : \alpha \geq 0\})$ Can be any function satisfying $\mathcal{F}(\boldsymbol{z}) = 0 \Longrightarrow \boldsymbol{z} = \boldsymbol{0}$

Ref Hengyue Liang, Buyun Liang, Le Peng, Ying Cui, Tim Mitchell, Ju Sun. *Optimization and Optimizers for Adversarial Robustness*. Under review at International Journal of Computer Vision (IJCV).

Outline

Constrained deep learning: CDL

- What and how for CDL
- Why CDL
- No good solvers for CDL yet
- Granso and PyGranso
- **PyGranso in action**
- Outlook

General instruction

https://ncvx.org/

PyGRANSO

NCVX PyGRANSO Documentation

Search the docs ...

- Introduction
- Installation
- Settings
- Examples
- Tips
- NCVX Methods
- Citing PyGRANSO
- Tutorial Sessions
- NCVX PyGRANSO Forum

Home

NCVX

NCVX Package

NCVX (NonConVeX) is a user-friendly and scalable python software package targeting general nonsmooth NCVX problems with nonsmooth constraints. NCVX is being developed by GLOVEX at the Department of Computer Science & Engineering, University of Minnesota, Twin Cities.

Our software announcement paper is available at https://arxiv.org/abs/2210.00973. This paper is accepted by the NeurIPS Workshop on Optimization for Machine Learning (OPT 2022). See our poster for more details.

SVM: mathematical form

$$\min_{w,b} \frac{\nu}{2}\|w\|^2 + b\nu + \frac{1}{n}\sum_{i=1}^{n} \boxed{\max(0, 1 - (\langle w, x_i \rangle + b))}$$

nonsmoothness

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

SVC constrained version

Ref https://scikit-learn.org/stable/modules/sgd.html#online-one-class-svm
https://scikit-learn.org/stable/modules/svm.html#svc

NCVX PyGRANSO live coding for SVM

https://colab.research.google.com/drive/1YVZN6KSzkd5QUCH1ZSrXPSizIFVpCWhl

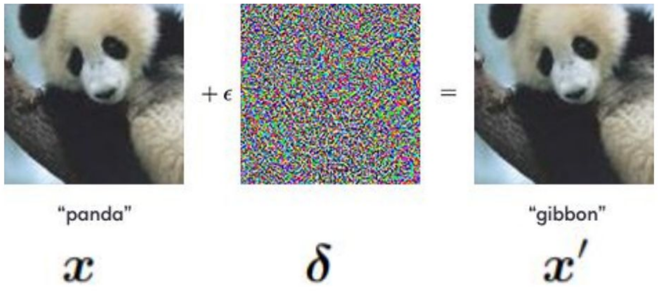
NCVX PyGRANSO quick summary: SVM

NVCX for unconstrained SVM

- can handle nonsmoothness
- reliable termination condition (w/o ad-hoc maxiteration)
- line search criterion (w/o step size scheduler)

NVCX is able to deal with general constrained problem (SVC)

Robustness evaluation: mathematical form



Maximize loss function

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

Allowable perturbation with radius **ε**

Valid image constraints

Minimize robustness radius

$$\min_{\boldsymbol{x}'} d\left(\boldsymbol{x}, \boldsymbol{x}'\right)$$

$$\text{s.t. } \max_{i \neq y} f_{\boldsymbol{\theta}}^{i}(\boldsymbol{x}') \geq f_{\boldsymbol{\theta}}^{y}(\boldsymbol{x}'), \quad \boldsymbol{x}' \in [0,1]^n$$

Change the predicted class

Valid image constraints

Robustness evaluation

$$\max_{\boldsymbol{x}'} \ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}'))$$
$$\text{s.t. } d(\boldsymbol{x}, \boldsymbol{x}') \leq \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

$$\min_{\boldsymbol{x}'} d(\boldsymbol{x}, \boldsymbol{x}')$$
$$\text{s.t. } \max_{i \neq y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \geq f^y_{\boldsymbol{\theta}}(\boldsymbol{x}'), \quad \boldsymbol{x}' \in [0,1]^n$$

First general-purpose method for evaluating adversarial robustness

Reliability

RobustBench
A standardized benchmark for adversarial robustness

- SOTA methods
  No stopping criterion (only use maxit); step size scheduler
- PWCF (ours)
  Line search criterion and termination condition

Generality

- SOTA methods
  Can mostly only handle standard lp norm (l1,l2,linf)
- PWCF (ours)
  Distance metric beyond standard lp norm (l1,l2,linf).
  E.g., perceptual distance

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2$$
$$\text{where} \quad \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x}) \,]$$

Ref Optimization and Optimizers for Adversarial Robustness. Liang, H., Liang, B., Peng, L., Cui, Y., Mitchell, T., & Sun, J. arXiv preprint arXiv:2303.13401.

NCVX PyGRANSO live coding for robust evaluation

https://colab.research.google.com/drive/1vO4YCnfhFokyYG7D_ufUy_q-QKrFho48

NCVX

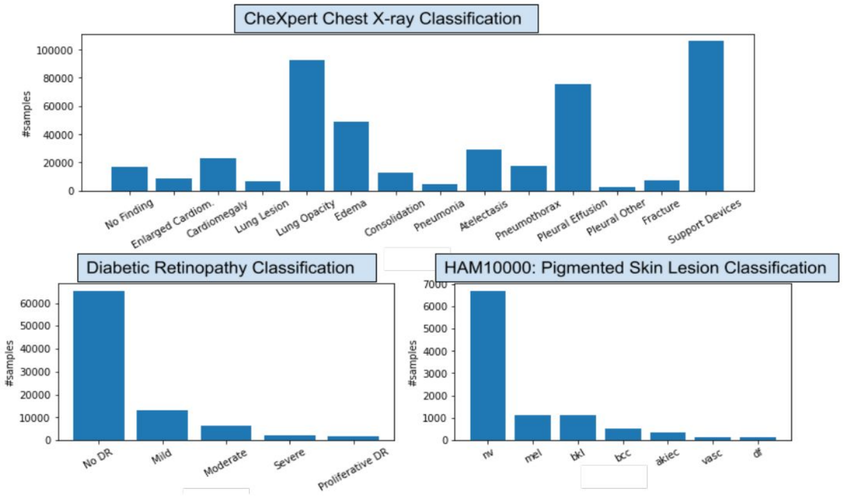
NCVX PyGRANSO quick summary: robustness

NVCX for robustness evaluation

- reliable termination condition (w/o ad-hoc maxiteration)
- line search criterion (w/o step size scheduler)

NVCX is able to deal with general constraints (perceptual attack)

Learning with label imbalance



Imbalance data in healthcare

	Predicted POS	Predicted NEG
POS	70	30
NEG	1000	9000

Accuracy: 9070/10100 = 0.898
True Positive Rate (Sensitivity, Recall): 0.7
True Negative Rate (Specificity): 0.9
Balanced Accuracy: (0.7 + 0.9)/2 = 0.80
Precision (POS): 70/1070 = 0.065
F1 Score: 2*0.065*0.7/(0.065 + 0.7) = 0.119

Reliable imbalance learning: **precision needed**!

Learning with label imbalance

$$\text{precision}(f_{\boldsymbol{\theta}}, t) = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}}{\sum_{i=1}^{N} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}} \qquad \text{recall}(f_{\boldsymbol{\theta}}, t) = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\} \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) > t\}}{\sum_{i=1}^{N} \mathbb{1}\{y_i = +1\}}$$

$$F_\beta(f_{\boldsymbol{\theta}}, t) = (1 + \beta^2) \frac{\text{precision}(f_{\boldsymbol{\theta}}, t) \cdot \text{recall}(f_{\boldsymbol{\theta}}, t)}{\beta^2 \text{precision}(f_{\boldsymbol{\theta}}, t) + \text{recall}(f_{\boldsymbol{\theta}}, t)}$$

One direction: directly optimizing the evaluation metric

$$\textbf{fix precision, optimize recall (FPOR):} \quad \max_{\boldsymbol{\theta}, t} \ \text{recall}(f_{\boldsymbol{\theta}}, t) \quad \text{s.t. } \text{precision}(f_{\boldsymbol{\theta}}, t) \geq \alpha,$$

$$\textbf{fix recall, optimize precision (FROP):} \quad \max_{\boldsymbol{\theta}, t} \ \text{precision}_t \quad \text{s.t. recall}(f_{\boldsymbol{\theta}}, t) \geq \alpha,$$

$$\textbf{optimize } F_\beta \textbf{ score (OFBS):} \quad \max_{\boldsymbol{\theta}, t} \ F_\beta(f_{\boldsymbol{\theta}}, t),$$

NCVX PyGRANSO live coding for imbalance learning

https://colab.research.google.com/drive/1__OeV8OSbpszqPImaYQgwqOQCXquuACl

NCVX PyGRANSO quick summary: imbalance learning

NVCX for robustness evaluation

- reliable termination condition (w/o ad-hoc maxiteration)
- line search criterion (w/o step size scheduler)

NVCX is able to deal with general constraints (e.g., precision/recall)

Closing

- Constrained DL (CDL) problems are everywhere

- Current methods for solving CDL are suboptimal
  - Projected gradient descent
  - Penalty methods
  - Lagrangian methods

- NCVX modeling framework + PyGranso solver is to address the gap
  - Principled stopping criterion, line search, and quasi-Newton method to obtain high-quality solution with reasonable speed

- Next steps
  - Stochastic optimization
  - Autoscaling