Deep Learning in Robust Optimization

Jannis Kurtz

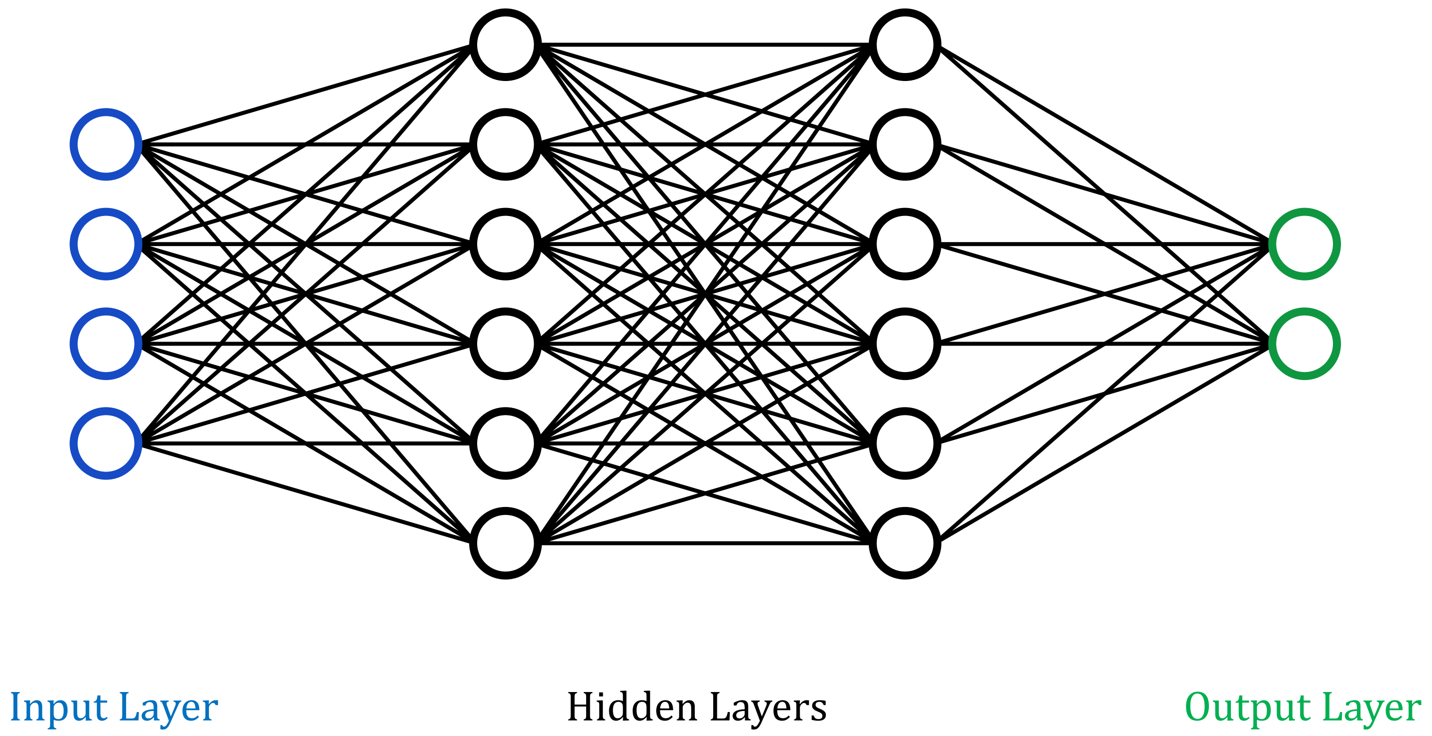
University of Amsterdam
Amsterdam Business School

Outline

I. Neural Networks
II. Robust Optimization
III. Construction of Uncertainty Sets

Neural Networks

Neural Networks

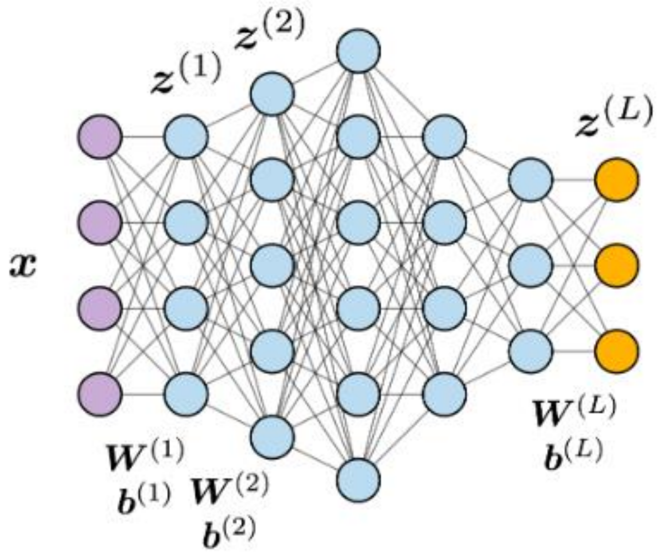


Fully-Connected Neural Networks

Mathematical Description. A **fully-connected deep neural network** can be represented by the following function:

$$\phi(x) = \left(f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)}\right)(x)$$

- $f^{(l)}(z) = \sigma(W^{(l)}z + b^{(l)})$
- data point $x \in \mathbb{R}^n$
- number of Layers L
- weight matrix $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$, bias $b^{(l)}$
- d_l is the *width* of the l-th layer
- activation function $\sigma : \mathbb{R} \to \mathbb{R}$ (applied componentwise)



Source: Balestriero et al. - On the Geometry of Deep Learning (2024)

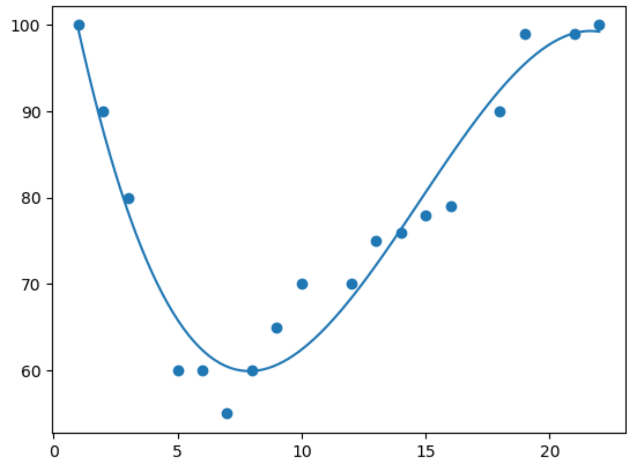
ReLU Activation Function. *An often used activation function is the ReLU (Rectified Linear Unit):*

$$\sigma(z) = \max\{z, 0\}$$

Regression Models

Regression Models. The goal of a regression model is to accurately predict the label $y \in \mathbb{R}^k$ for a given data point $x \in \mathbb{R}^N$.

- **data space** $\mathcal{X}$
- **labeled training data**: $\mathcal{D} = \{(x^1, y^1), \dots, (x^m, y^m)\} \subset \mathcal{X} \times \mathbb{R}^k$
- fit a **prediction function** $f_w : \mathcal{X} \to \mathbb{R}^k$ to the training data



Deep Learning

Deep Learning. In deep learning we are fitting a neural network to the training data.

- Define the **neural network structure**: number of layers L and neurons per layer
- Define a **loss function** $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and a **regularizer** $\mathcal{R}$.
- Solve the following optimization problem:

Output of the Neural Network

$$\min_{W,b} \sum_{j=1}^{m} \ell(\phi_{W,b}(x^j), y^j) + \mathcal{R}(W, b)$$

where $W = (W^{(1)}, \ldots, W^{(L)})$ and $b = (b^{(1)}, \ldots, b^{(L)})$.

Solution Methods. Fast and parallelizable methods were developed to tackle huge network sizes and huge amounts of data:

- Usually solved via (stochastic) gradient descent methods.
- No global optimum guaranteed.
- However, also local optima can generalize well on unseen data.

Why Neural Networks?

Expressivity. *Neural Networks can approximate very general classes of functions:*

- *Every continuous function can be approximated up to an arbitrary accuracy* $\varepsilon > 0$ *by a neural network.*

Survey on expressivity: [Gühring, Raslan, & Kutyniok (2020)]

- *Every piecewise linear function in dimension* n *can be exactly represented by a ReLU-NN where the number of layers is at most*

$$\lceil \log_2(n+1) \rceil + 1$$

[Arora et al. (2016)]

Backpropagation.

- *Calculate the gradient in the parameters* W, b *of a neural network by applying the chain rule.*
- *Can be done by iteratively going backwards through the network (backpropagation).*
- *Can be parallelized.*

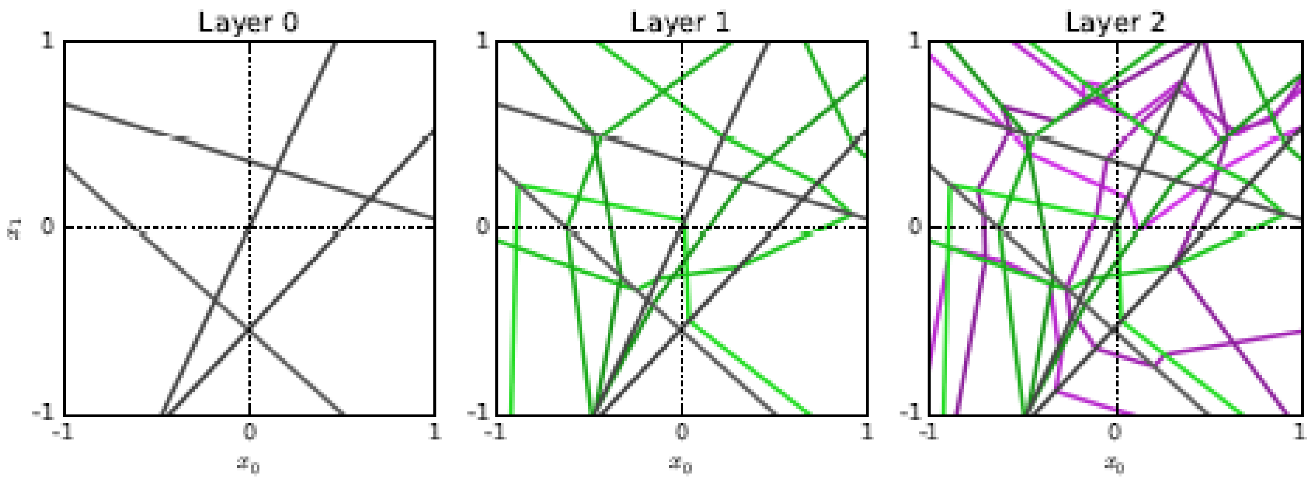
[LeCun, Bengio, & Hinton (2015)]

Piecewise Affine-Linear Functions

ReLU Neural networks are just piecewise affine-linear functions!

- Fix an activation pattern for all neurons.
- The set of data points which fulfill these activations is a polyhedron.
- Output is linear on this polyhedron.

[Wang, Balestriero, & Baraniuk, (2018)]



[Raghu et al. (2017)]

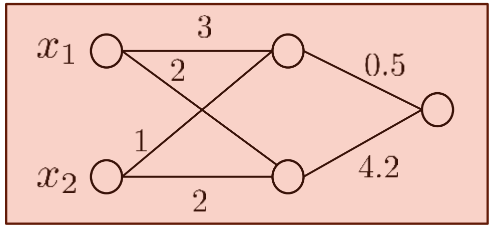
Set of points which activate a certain neuron:

$$\{x : w^\top x > 0\}$$

Set of points which deactivate a certain neuron:

$$\{x : w^\top x \leq 0\}$$

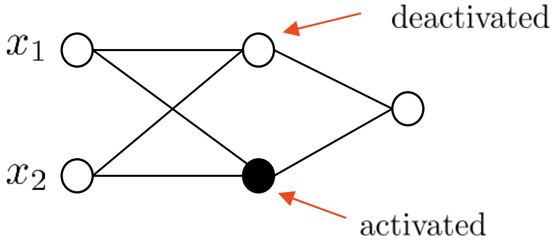
Example



Consider the following trained neural network:

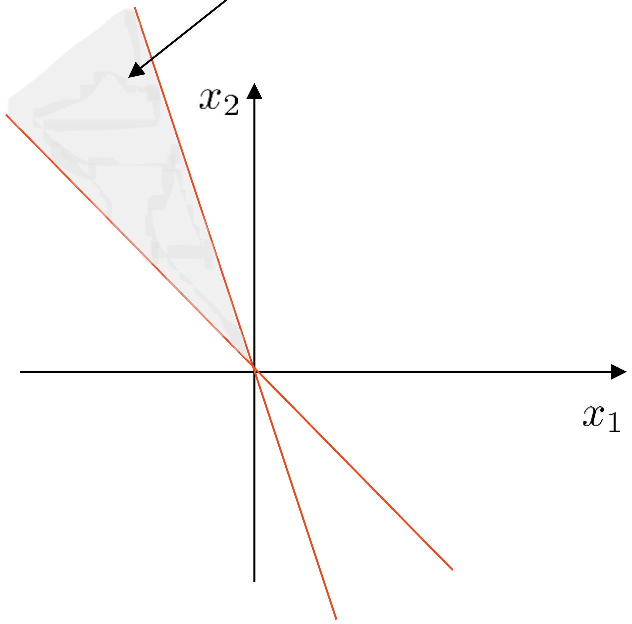
$$\phi(x) = (0.5, 4.2)^\top \sigma\left(\begin{pmatrix} 3 & 1 \\ 2 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$$

Consider the activation pattern:



$3x_1 + x_2 \leq 0$

$2x_1 + 2x_2 > 0$



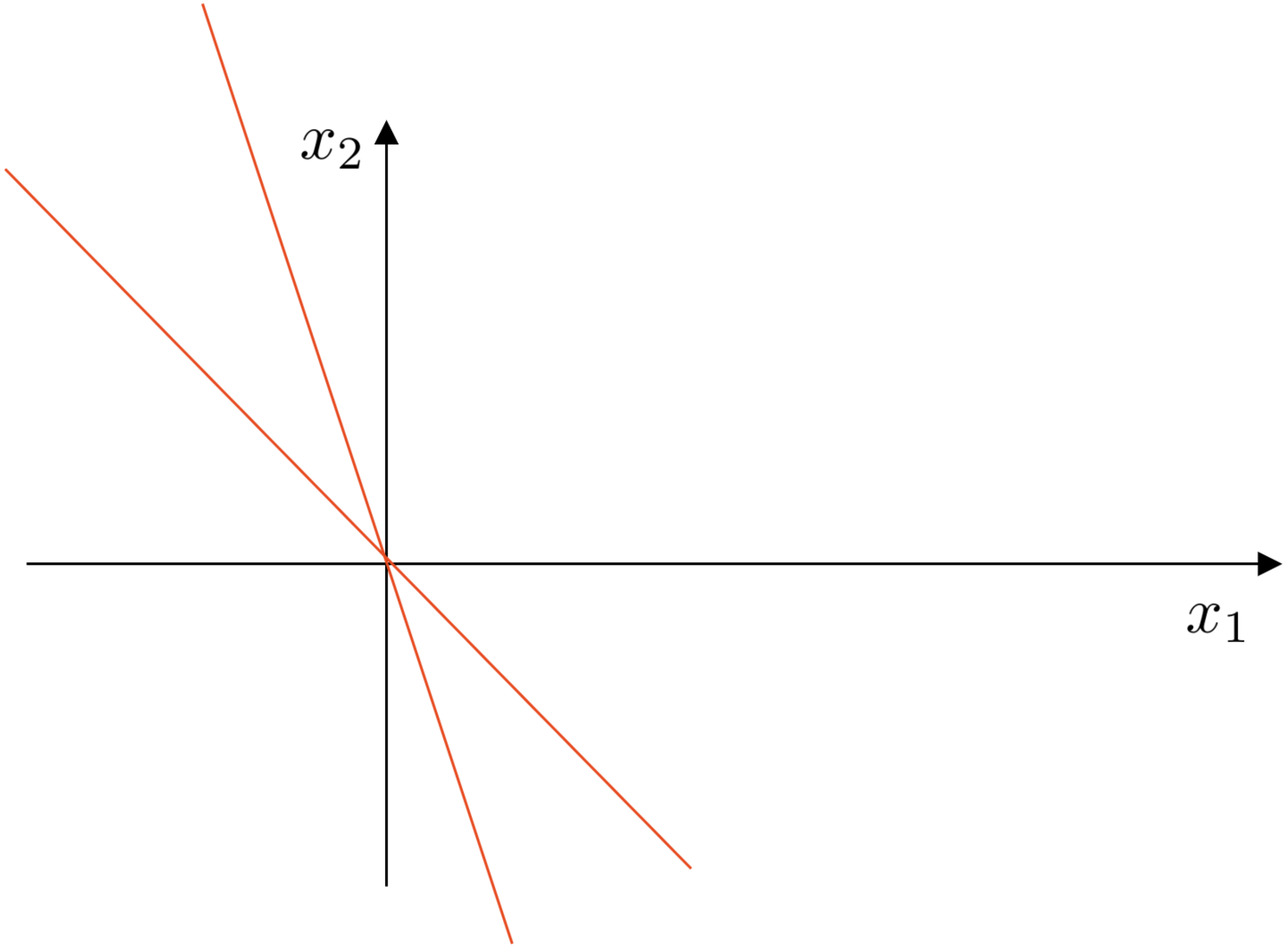
A neuron is activated if the ReLU is positive, i.e.,

$$\max\{w^\top x, 0\} > 0$$

In this region the neural network function is linear:

$$\phi(x) = 0.5 \cdot 0 + 4.2(2x_1 + 2x_2)$$

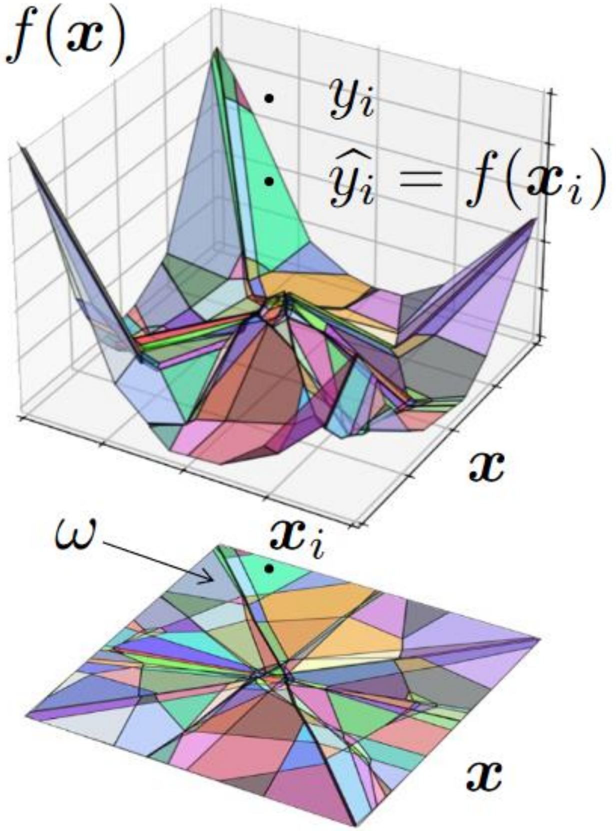
Example



Geometry of Neural Networks

On the Geometry of Deep Learning

Randall Balestriero * Ahmed Imtiaz Humayun † Richard G. Baraniuk ‡



MIP Representable Neural Networks

The evaluation of an **already trained neural network** with ReLU activation function can be modeled by a mixed-integer programming (MIP) formulation:

- For a given data point x the output of the first layer (after activation) is

$$\sigma\left(W^1 x\right) = \begin{pmatrix} \sigma((w_1^1)^\top x) \\ \vdots \\ \sigma((w_{d_1}^1)^\top x) \end{pmatrix}$$

  where we have ReLU activation $\sigma(z) = \max\{z, 0\}$.

- Each output component $v = \sigma(w^\top x)$ can be modeled by the following MIP constraints where M is a large value:

$$\begin{aligned} w^\top x &\leq v \\ w^\top x &\geq v - Mu \\ v &\leq M(1-u) \\ v &\geq 0 \\ u &\in \{0,1\} \end{aligned}$$

Optimizing over Neural Networks

Theorem (Fischetti, M., & Jo, J. (2018)). For a given **already trained neural network** ϕ_W with ReLU activations the following optimization problem can be modeled as a mixed-integer problem:

$$\min_x \; g(\phi_{W,b}(x))$$
$$s.t. \quad x \in \mathcal{X}.$$

Literature. Improving the computational performance for the optimization over neural networks:

- **Progressive Bound Tightening:** ([Tjeng, Xiao, & Tedrake (2017)])
- **Lossless compression of Neural Networks:** [Serra, Kumar, & Ramalingam (2020)], [ElAraby, Wolf, & Carvalho (2020) and (2023)]
- **Bounding and Counting Linear Regions:** [Serra, Tjandraatmadja, & Ramalingam (2018)]
- **Survey:** Huchette, J., Muñoz, G., Serra, T., & Tsay, C. (2023). When deep learning meets polyhedral theory: A survey. arXiv preprint arXiv:2305.00241.

Applications

There are many applications where optimizing over a trained neural network is involved:

- **Finding adversarial examples**: for a given data point $\hat{x}$ find a similar point $x \in \mathcal{X}$ for which the output of the network changes significantly:

$$\begin{aligned} \min_x \quad & \|x - \hat{x}\| \\ s.t. \quad & \|\phi_{W,b}(x) - \phi_{W,b}(\hat{x})\| \geq \Delta \\ & x \in \mathcal{X} \end{aligned}$$

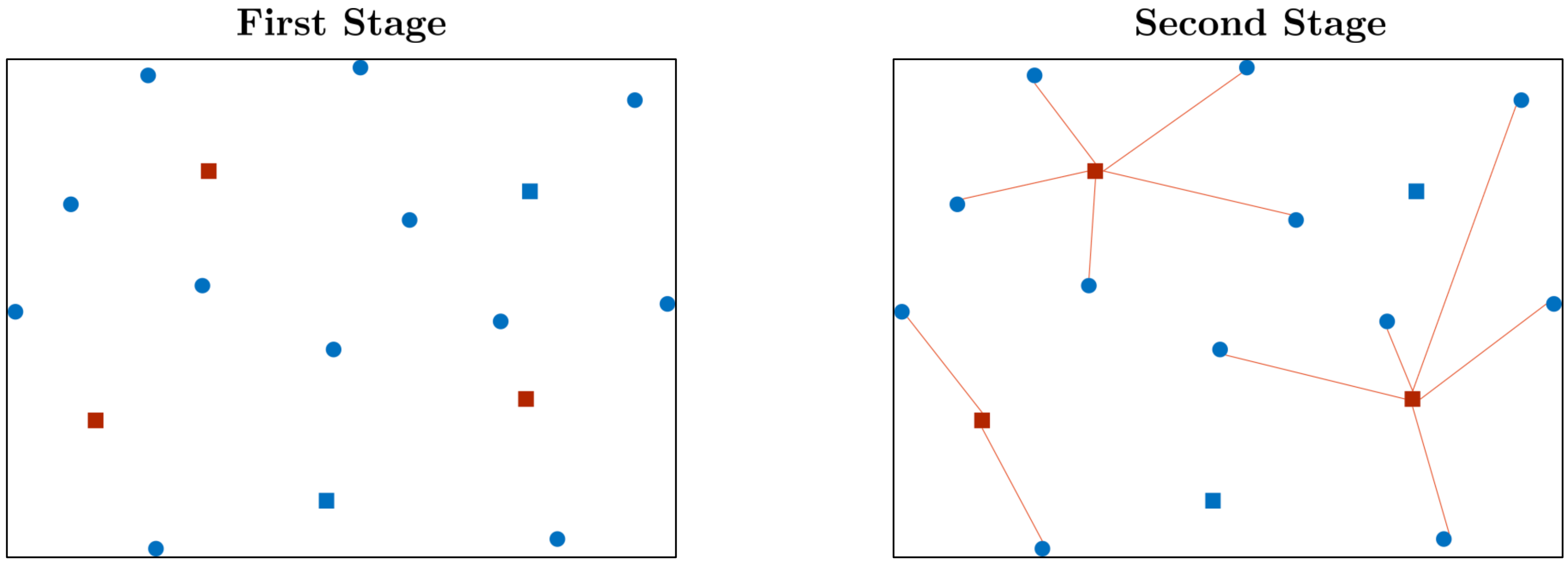
- **Counterfactual explanations**: similar to adversarial examples
- **Robustness analysis**: what is the maximum radius in which the output of the neural network does **not** change significantly.
- **Pruning of neural networks**: find the maximum number of neurons which can be deleted such that the prediction quality of the neural network does not deteriorate too much.
- **Training of binarized neural networks:** if the weights are restricted to values in $\{-1, 0, 1\}$ training NNs can be done via MIP.
- **Constraint Learning:** model "difficult" constraints for which no mathematical expression is known.

(Two-stage) Robust Optimization

Facility Location under Uncertainty

Two-stage Facility Location. *Minimize total opening and transportation costs:*

- ***First Stage****: open facilities* ***before*** *uncertain demands are known*
- ***Second Stage****: assign customers to open facilities* ***after*** *uncertain demands are known*



(Two-stage) Robust Optimization

Problem Definition. We consider two-stage robust optimization problems (2RO) of the following form:

$$\min_{x\in\mathcal{X}} \max_{\xi\in\Xi} \min_{y\in\mathcal{Y}} \quad c(\xi)^\top x + d(\xi)^\top y$$
$$\text{s.t.} \quad T(\xi)x + W(\xi)y \le h(\xi).$$

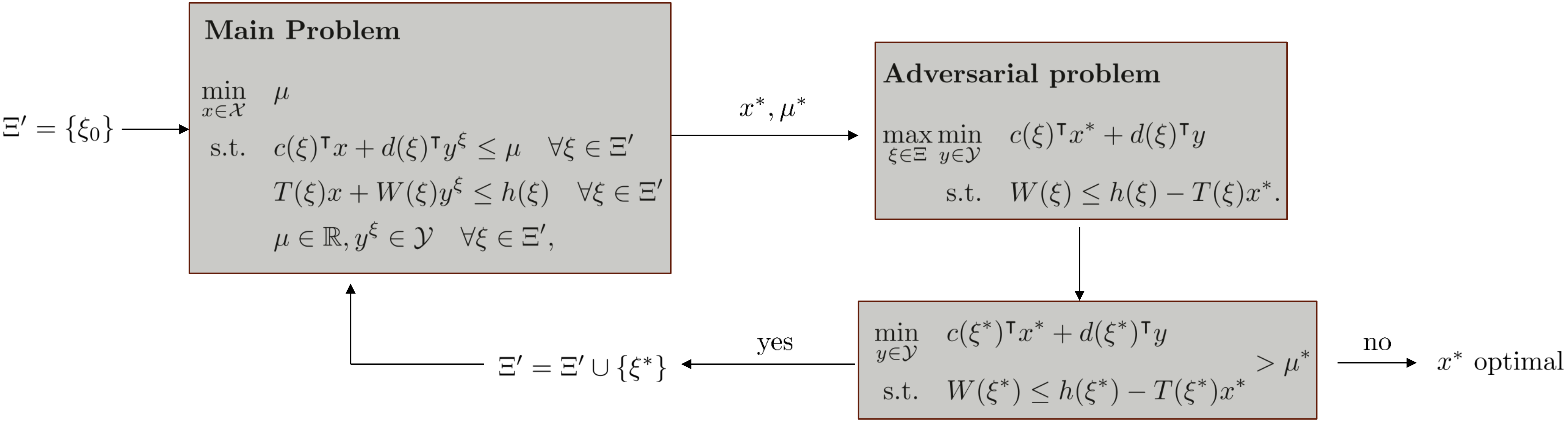
First Stage | Worst Case | Second Stage

- x: first-stage decisions
- y: second-stage decisions
- ξ: uncertain parameters
- Ξ: convex uncertainty set
- $c(\xi), d(\xi)$: first and second-stage costs
- $T(\xi), W(\xi), h(\xi)$: constraints parameters

2RO is extremely hard to solve especially when

- the second-stage variables are integer,
- the uncertain parameters appear in the constraints.

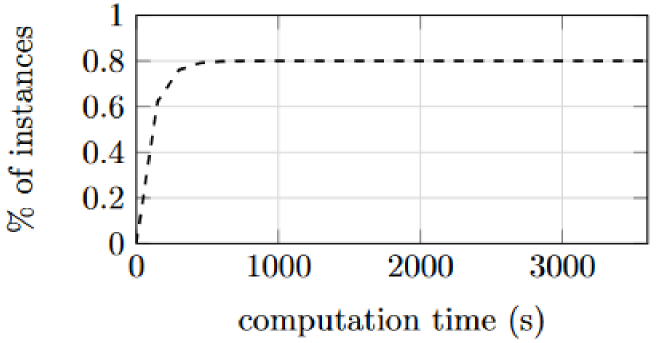
Column-and-Constraint Generation



Literature: CCG for Robust Optimization

Literature

- **Objective uncertainty + integer second-stage:** (Kämmerling, & Kurtz (2020))
- **Constraint uncertainty + continuous second-stage:** (Zeng, & Zhao (2013)), (Tsang, Shehadeh, & Curtis (2023))
- **Constraint uncertainty + integer second-stage:**
  - General method: (Zhao, & Zeng (2012))
  - "Interdiction-type problems": (Lefebvre, Schmidt, & Thürauf (2023))



Stolen from [Lefebvre, Schmidt, & Thürauf (2023)]

Neural Two-Stage Robust Optimization

For more details.
J. Dumouchelle, E. Julien, J. Kurtz and E. B. Khalil (2023). *Neur2RO: Neural Two-Stage Robust Optimization.* The Twelfth International Conference on Learning Representations (ICLR), 2024

Neur2RO

Main Idea. Train a neural network which predicts the optimal value of the second stage problem.

$$NN_\Theta(x,\xi) \approx \min_{y\in\mathcal{Y}}\{c(\xi)^\intercal x + d(\xi)^\intercal y : W(\xi)y \le h(\xi) - T(\xi)x\},$$

Main Problem

$$\begin{aligned}
\min_{x\in\mathcal{X}} \quad & \mu \\
\text{s.t.} \quad & c(\xi)^\intercal x + d(\xi)^\intercal y^\xi \le \mu \quad \forall \xi \in \Xi' \\
& T(\xi)x + W(\xi)y^\xi \le h(\xi) \quad \forall \xi \in \Xi' \\
& \mu \in \mathbb{R}, y^\xi \in \mathcal{Y} \quad \forall \xi \in \Xi',
\end{aligned}$$

NN →

Main Problem

$$\begin{aligned}
\min_{x\in\mathcal{X},\mu\in\mathbb{R}} \quad & \mu \\
\text{s.t.} \quad & \mu \ge NN_\Theta(x,\xi) \quad \forall \xi \in \Xi'
\end{aligned}$$

Adversarial problem

$$\begin{aligned}
\max_{\xi\in\Xi}\min_{y\in\mathcal{Y}} \quad & c(\xi)^\intercal x^* + d(\xi)^\intercal y \\
\text{s.t.} \quad & W(\xi)y \le h(\xi) - T(\xi)x^*.
\end{aligned}$$

NN →

Adversarial problem

$$\max_{\xi\in\Xi} NN_\Theta(x^*,\xi)$$

Neur2RO

Main Idea. Train a neural network which predicts the optimal value of the second stage problem.

$$NN_\Theta(x,\xi) \approx \min_{y \in \mathcal{Y}} \{c(\xi)^\intercal x + d(\xi)^\intercal y : W(\xi)y \le h(\xi) - T(\xi)x\},$$

Main Problem

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \mu \\ \text{s.t.} \quad & c(\xi)^\intercal x + d(\xi)^\intercal y^\xi \le \mu \quad \forall \xi \in \Xi' \\ & T(\xi)x + W(\xi)y^\xi \le h(\xi) \quad \forall \xi \in \Xi' \\ & \mu \in \mathbb{R}, y^\xi \in \mathcal{Y} \quad \forall \xi \in \Xi', \end{aligned}$$

NN →

Main Problem

$$\begin{aligned} \min_{x\in\mathcal{X}, y\in\mathcal{Y}, \xi_a\in\Xi} \quad & c(\xi_a)^\intercal x + d(\xi_a)^\intercal y \\ \text{s.t.} \quad & W(\xi_a)y + T(\xi_a)x \le h(\xi_a), \\ & \xi_a \in \arg\max_{\xi\in\Xi'} \{NN_\Theta(x,\xi)\} \end{aligned}$$

Adversarial problem

$$\begin{aligned} \max_{\xi\in\Xi} \min_{y\in\mathcal{Y}} \quad & c(\xi)^\intercal x^* + d(\xi)^\intercal y \\ \text{s.t.} \quad & W(\xi)y \le h(\xi) - T(\xi)x^*. \end{aligned}$$

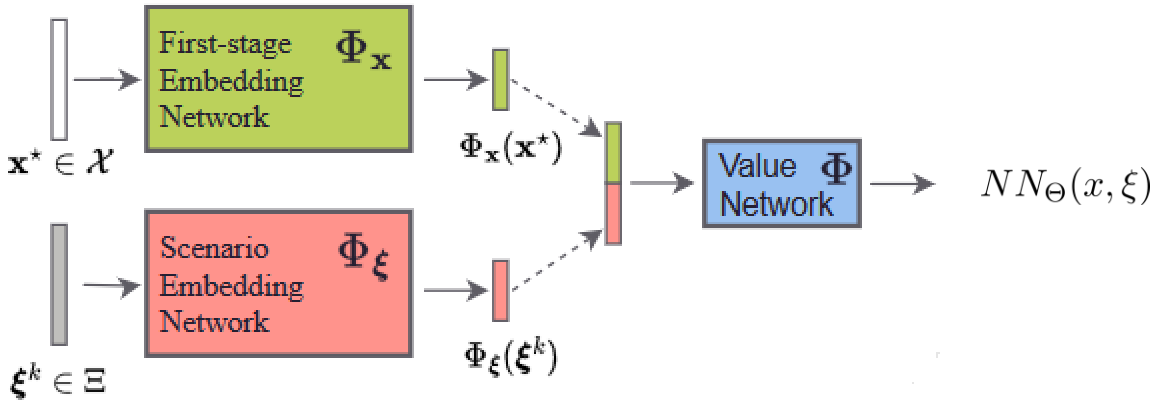
NN →

Adversarial problem

$$\max_{\xi\in\Xi} NN_\Theta(x^*,\xi)$$

Mixed-integer linear programs

Network Architecture



Main Problem

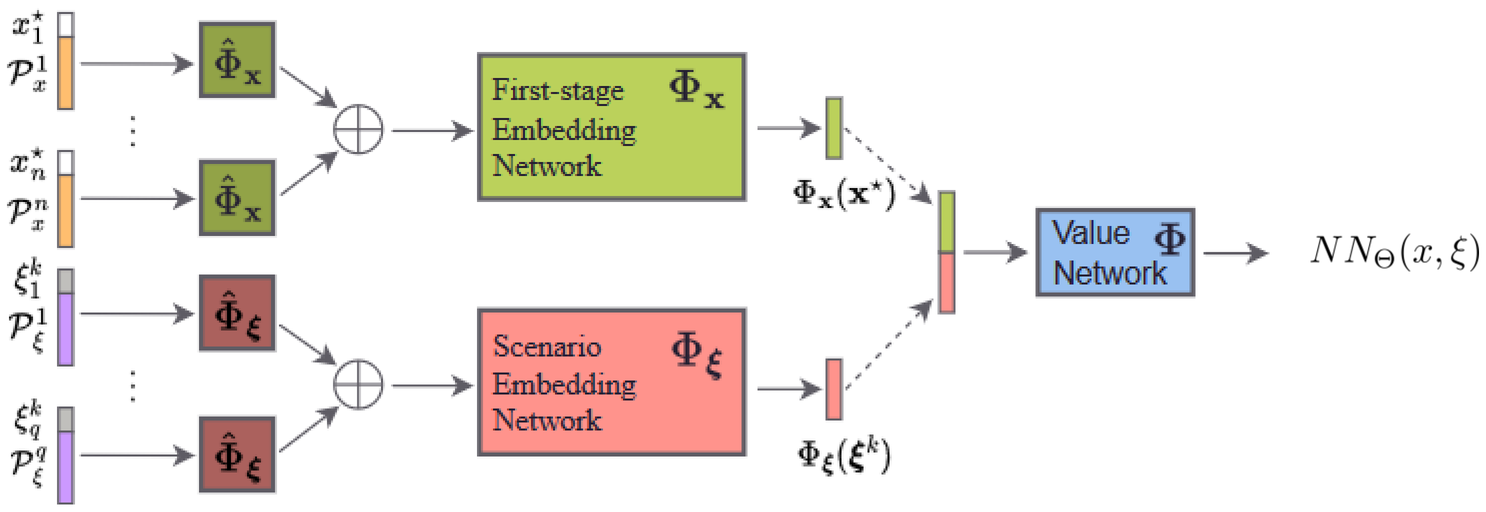
$$\begin{aligned} \min_{x\in\mathcal{X}, y\in\mathcal{Y}, \xi_a\in\Xi} \quad & c(\xi_a)^\intercal x + d(\xi_a)^\intercal y \\ \text{s.t.} \quad & W(\xi_a)y + T(\xi_a)x \leq h(\xi_a), \\ & \xi_a \in \arg\max_{\xi\in\Xi'}\{NN_\Theta(x, \xi)\} \end{aligned}$$

Adversarial problem

$$\max_{\xi\in\Xi} NN_\Theta(x^*, \xi)$$

These embeddings can be pre-calculated.

Set-Based Architecture



Hyperparameter	Knapsack	Capital budgeting
$\hat{\Phi}_x$ dimensions	[32, 16]	[16, 4]
Φ_x dimensions	[64, 8]	[32, 8]
$\hat{\Phi}_\xi$ dimensions	[32, 16]	[16, 4]
Φ_ξ dimensions	[64, 8]	[32, 8]
Φ dimensions	[8]	[8]

Set-based architecture allows for generalization to higher dimensional instances than contained in the training set.

Approximation Guarantee

Assume that the predictions of the neural network can have an error of at most $\alpha > 0$, i.e.

$$|NN_{\Theta}(x,\xi) - \mathrm{val}\,(x,\xi)| \le \alpha \quad \forall x \in \mathcal{X}, \xi \in \Xi.$$

Theorem. If $|\mathcal{X}|$ is finite and if relatively complete recourse holds, our algorithm converges to a solution $x_{\mathrm{NN}} \in \mathcal{X}$ with

$$\max_{\xi \in \Xi} \ \mathrm{val}\,(x_{\mathrm{NN}}, \xi) \le \mathrm{opt} + 2\alpha + \varepsilon$$

where opt is the optimal value of the original problem and ε an accuracy parameter.

Experiments: Data Collection & Testing

Data Generation.

- Sample random first stage solutions x
- Sample random scenarios ξ
- Solve second-stage problem for x and ξ to obtain objective value (label)
- randomly sample 500 instances, 10 first-stage decisions per instance, and 50 scenarios per first-stage decision: 250,000 data points
- Can be parallelized.

Training. Train **one neural network** over training data of all instance sizes.

Experiment: Two-stage Knapsack

Correlation Type	# items	Median RE		Times	
		Neur2RO	BP	Neur2RO	BP
Uncorrelated	20	1.417	**0.000**	7	**0**
	30	1.188	**0.000**	9	**1**
	40	1.614	**0.000**	13	**3**
	50	1.814	**0.000**	14	**12**
	60	1.146	**0.000**	24	**18**
	70	1.408	**0.000**	**27**	46
	80	0.994	**0.000**	**20**	388
Weakly Correlated	20	1.705	**0.000**	**7**	29
	30	2.236	**0.000**	**16**	454
	40	1.667	**0.000**	**45**	6,179
	50	1.756	**0.000**	**42**	8,465
	60	0.772	**0.000**	**134**	9,242
	70	0.068	**0.020**	**32**	10,800
	80	**0.000**	0.345	**45**	10,800

Correlation Type	# items	Median RE		Times	
		Neur2RO	BP	Neur2RO	BP
Almost Strongly Correlated	20	1.798	**0.000**	**7**	9
	30	0.627	**0.000**	**10**	2,708
	40	0.497	**0.000**	**17**	4,744
	50	0.019	**0.000**	**13**	8,852
	60	0.047	**0.000**	**27**	10,261
	70	**0.031**	0.031	**34**	10,800
	80	0.106	**0.035**	**26**	10,800
Strongly Correlated	20	1.774	**0.000**	**8**	9
	30	0.670	**0.000**	**11**	2473
	40	0.542	**0.000**	**20**	5,665
	50	0.073	**0.000**	**18**	8,240
	60	**0.000**	0.046	**21**	10,800
	70	**0.020**	0.027	**28**	10,800
	80	**0.000**	0.032	**31**	10,800

Instances and Baseline: Arslan, A. N., & Detienne, B. (2022). Decomposition-based approaches for a class of two-stage robust binary optimization problems. INFORMS journal on computing, 34(2), 857-871.

Experiments: Facility Location

# items	Median RE						Times (seconds)					
	Neur2RO	Neur2RO-pga	Static	$k=2$	$k=5$	$k=10$	Neur2RO	Neur2RO-pga	Static	$k=2$	$k=5$	$k=10$
(5, 10)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	5	5	**0**	31	40	26
(5, 20)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	6	6	**0**	9	17	17
(5, 50)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	4	5	**0**	1614	1381	1404
(10, 10)	**0.000**	**0.000**	1.393	0.653	**0.000**	**0.000**	7	7	**1**	2323	4646	5151
(10, 20)	**0.000**	**0.000**	2.730	1.280	**0.000**	**0.000**	10	7	**1**	4577	6764	6751
(10, 50)	**0.000**	**0.000**	1.103	1.103	0.744	0.186	15	**9**	27	2342	6519	7241
(20, 20)	**0.000**	**0.000**	6.048	3.498	2.841	2.206	**15**	15	132	10823	10291	10311
(20, 50)	**0.000**	**0.000**	5.014	3.396	3.913	3.298	20	**17**	51	10463	10828	10834

# items	Percent of feasible/found solution)					
	Neur2RO	Neur2RO-pga	Static	$k=2$	$k=5$	$k=10$
(5, 10)	96	88	**100**	**100**	**100**	**100**
(5, 20)	92	80	**100**	**100**	**100**	**100**
(5, 50)	**100**	76	**100**	**100**	**100**	**100**
(10, 10)	**100**	48	**100**	**100**	**100**	**100**
(10, 20)	**100**	36	**100**	**100**	**100**	**100**
(10, 50)	**100**	44	**100**	**100**	**100**	**100**
(20, 20)	88	20	**100**	88	80	84
(20, 50)	96	8	**100**	84	96	96

Instances and Baseline: Subramanyam, A., Gounaris, C. E., & Wiesemann, W. (2020). K-adaptability in two-stage mixed-integer robust optimization. Mathematical Programming Computation, 12, 193-224.2

Experiments: Facility Location

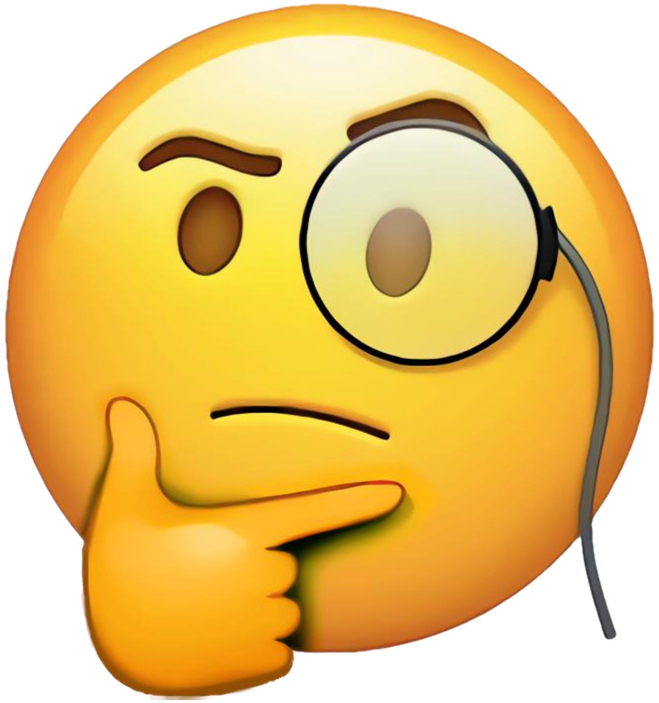
# items	Median RE					Percent of feasible/found solutions				
	Neur2RO	Static	$k=2$	$k=5$	$k=10$	Neur2RO	Static	$k=2$	$k=5$	$k=10$
(5, 10)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	96	**100**	84	80	84
(5, 20)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	92	**100**	96	**100**	88
(5, 50)	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**100**	**100**	76	76	64
(10, 10)	**0.000**	0.618	**0.000**	**0.000**	**0.000**	**100**	**100**	80	64	60
(10, 20)	**0.000**	2.730	0.430	0.430	**0.000**	**100**	**100**	72	64	68
(10, 50)	**0.000**	1.103	1.393	0.916	1.023	**100**	**100**	72	56	40
(20, 20)	**0.000**	5.702	2.298	**0.000**	**0.000**	88	**100**	4	4	4
(20, 50)	**0.000**	5.014	0.557	0.895	-	96	**100**	4	8	-

Table 8 **Facility location median errors at ML termination time (PGA results excluded).**

Instances and Baseline: Subramanyam, A., Gounaris, C. E., & Wiesemann, W. (2020). K-adaptability in two-stage mixed-integer robust optimization. Mathematical Programming Computation, 12, 193-224.2

Open Problems

- The approximation bounds we present are quite conservative. Can we derive approximation bounds based on the data distribution?
- Constraint uncertainty has to be investigated more.
- Can other algorithms in robust optimization benefit from NN-support?

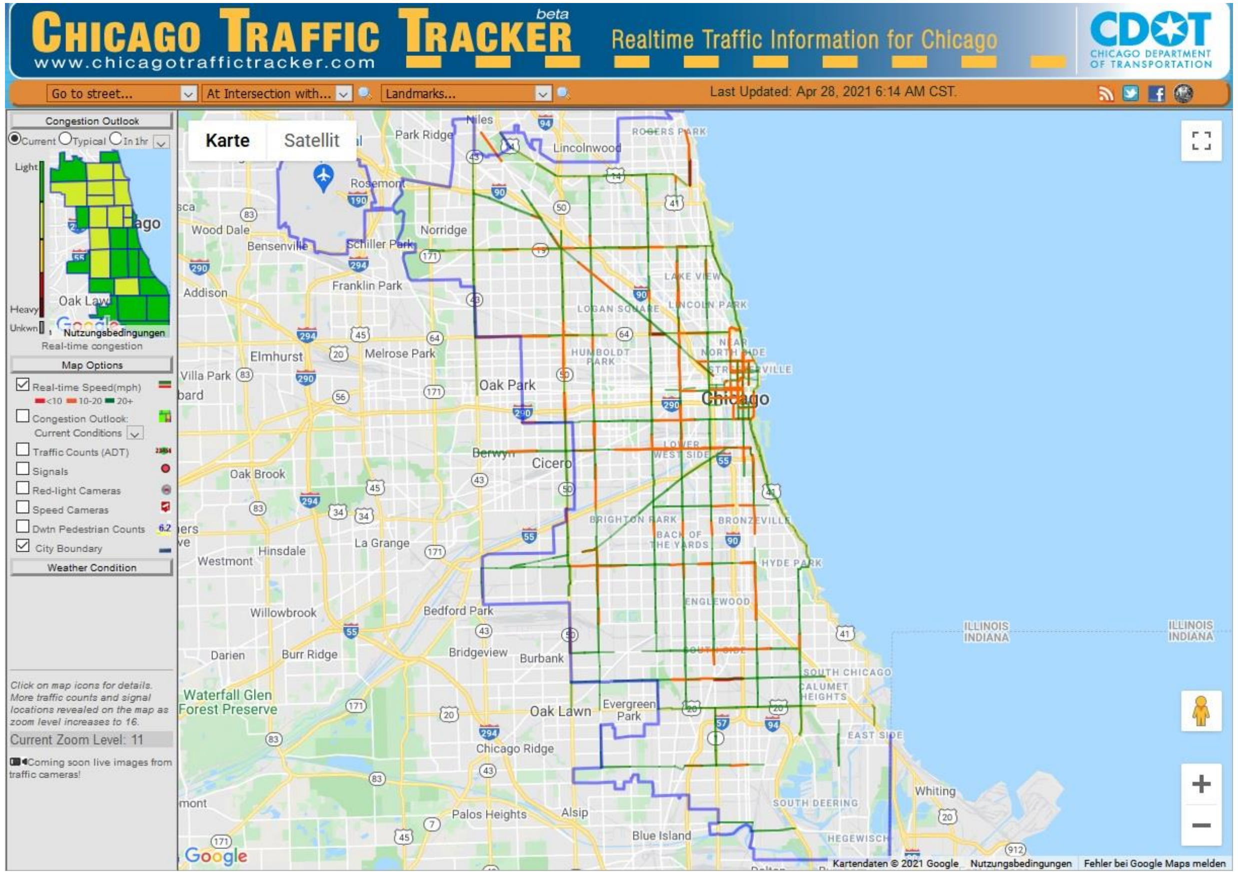


Construct Uncertainty Sets

For more details.
Goerigk, M., & Kurtz, J. (2023). *Data-driven robust optimization using deep neural networks.* Computers & Operations Research, 151, 106087.

Historical Data

Historical Data. In practice we can often observe historical scenarios $\xi^1, \dots, \xi^m \in \mathbb{R}^n$.



But how to construct an appropriate uncertainty set from this?

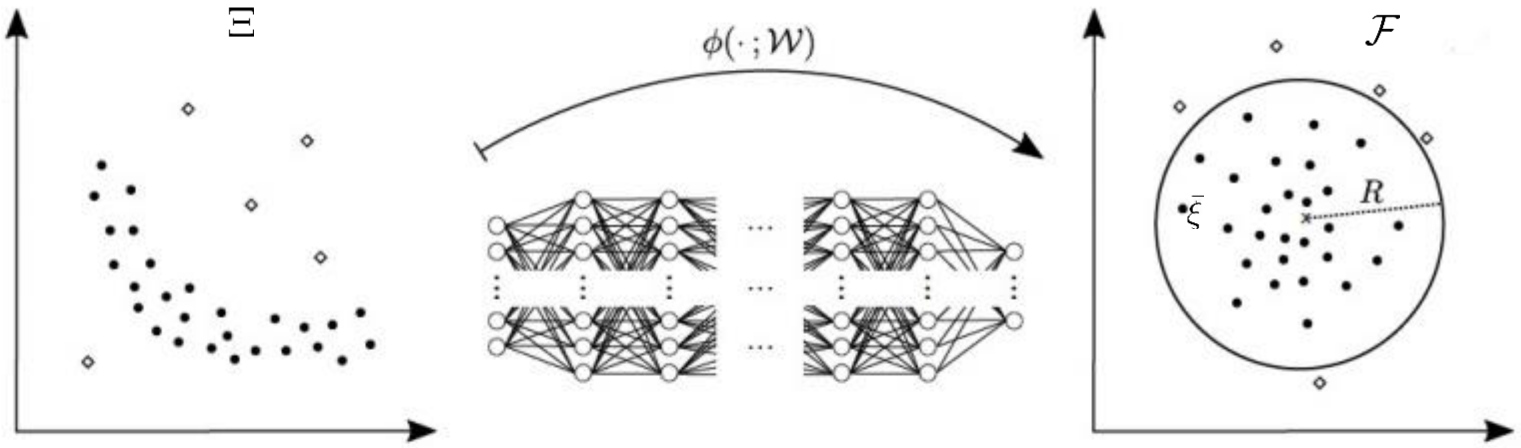
One-Class Deep Learning

Main Idea. Our approach is based on the following idea:

- Train a neural network which detects for a given scenario $\xi \in \Xi$ if it is a realistic scenario or not.
- Also called anomaly detection or One-Class Deep Learning

Approach.

- Train a neural network which maps the data into a new **feature space**
- Find the **smallest ball** in the new space such that all collected scenarios are contained in it



[Ruff et al. (2018)]

Construction Uncertainy Set

Algorithm.
Input: given **historical scenarios** $\xi^1, \ldots, \xi^m$ (training data)

1. select a center point $\bar{\xi} \in \mathbb{R}^{d_L}$ and solve (e.g. via stochastic gradient descent)

$$\min_{W^1,\ldots,W^L} \frac{1}{m} \sum_{i \in [m]} \|\phi(\xi^i, W) - \bar{\xi}\|_2^2 + \frac{\lambda}{2} \sum_{l \in [L]} \|W^l\|_F^2$$

[Ruff et al. (2018)]

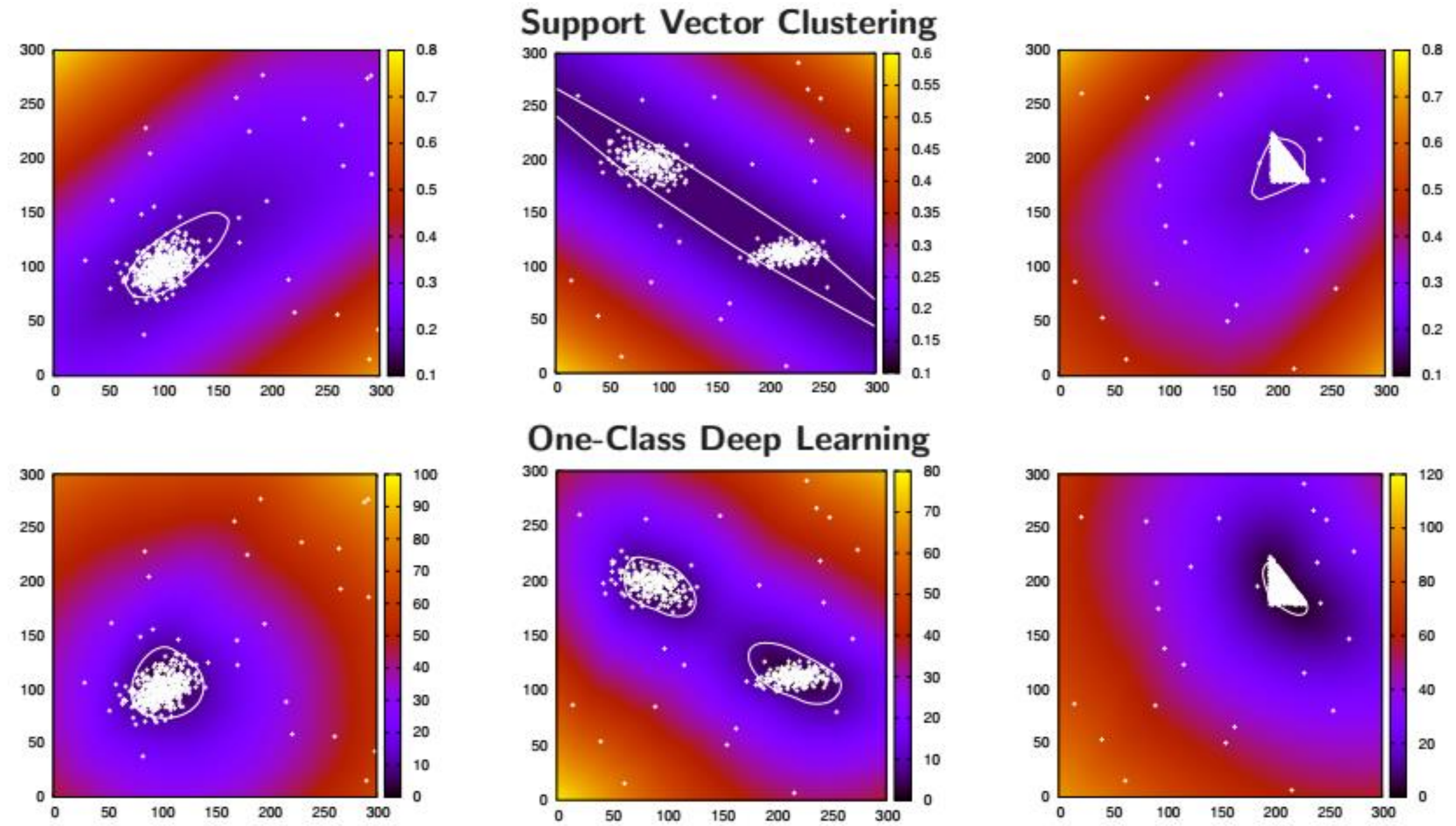
2. determine **radius** R e.g. as 95%-quantile of

$$r_i := \|\phi(\xi^i, W) - \bar{\xi}\|$$

Return: uncertainty set

$$U := \left\{\xi \in \mathbb{R}^n : \|\phi(\xi, W) - \bar{\xi}\|_2 \le R\right\}$$

2-Dimensional Example



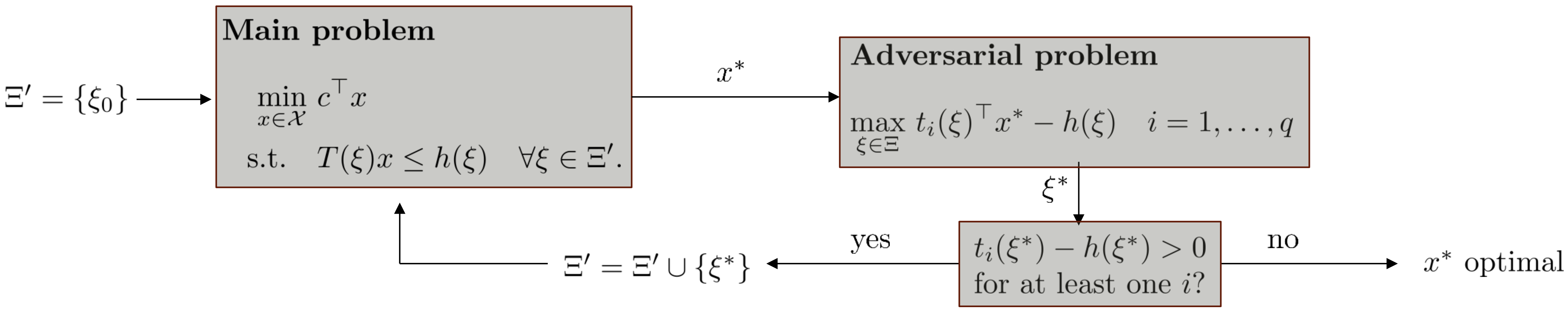
Optimization Algorithm

Consider classical robust optimization:

$$\min_{x \in \mathcal{X}} c^\top x$$
$$\text{s.t.} \quad T(\xi)x \leq h(\xi) \quad \forall \xi \in \Xi.$$

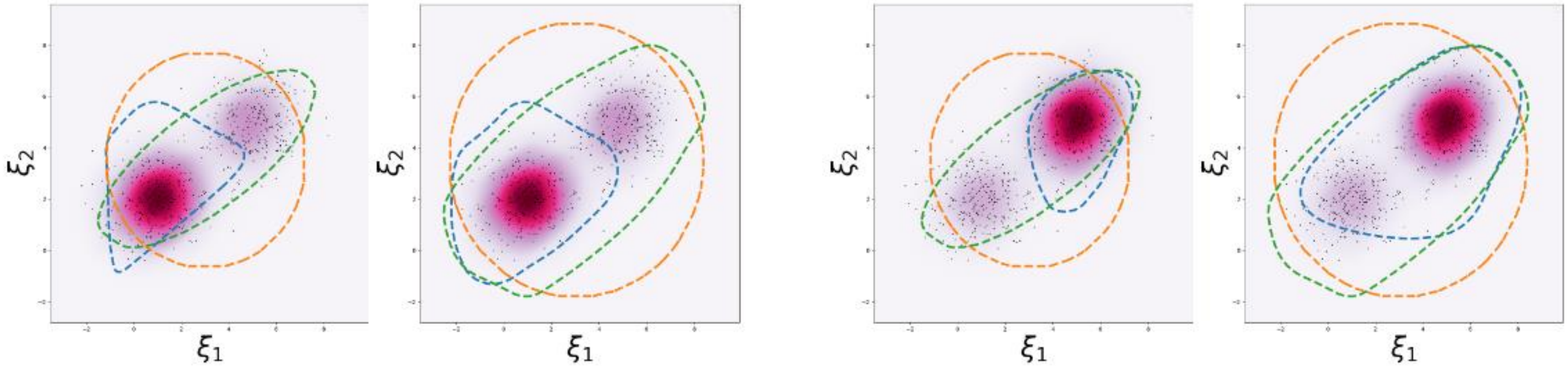
Due to the non-convex structure of the set we cannot use duality theory to solve the robust optimization problem.

Constraint Generation



Extension

Construct uncertainty sets based on contextual information (e.g. weather, day, time).



(a) $\tilde{a}(\psi) = 1$, 90% coverage

(b) $\tilde{a}(\psi) = 1$, 99% coverage

(c) $\tilde{a}(\psi) = 2$, 90% coverage

(d) $\tilde{a}(\psi) = 2$, 99% coverage

IDCC DDDRO Ellipsoid

Chenreddy, A. R., Bandi, N., & Delage, E. (2022). *Data-driven conditional robust optimization.* Advances in Neural Information Processing Systems, 35, 9525-9537.

Conclusion

Summary.

- Trained neural networks can be represented as mixed-integer programs.
- MIP representations can be incorporated into classical CCG algorithms for robust optimization to find close to optimal solutions in seconds.
- MIP representations can be used to model uncertainty sets for robust optimization.

Thank you for your attention!

Robust Optimization Webinar

Season 4 of the Robust Optimization Webinar just started!

Robust Optimization Webinar

September 6	**Daniel Kuhn**	École Polytechnique Fédérale de Lausanne
September 20	**Hoda Bidkhori**	George Mason University
October 4	**Daniël Vos**	Delft University of Technology
	Michael Hartisch	Friedrich-Alexander-Universität Erlangen-Nürnberg
October 18	**Adam Kasperski**	Wrocław University of Science and Technology
November 1	**Merve Bodur**	University of Edinburgh
November 15	**Bart van Parys**	Centrum Wiskunde & Informatica (CWI)
November 29	**Rosario Paradiso**	Vrije Universiteit Amsterdam
December 13	**Simon Thomä**	RWTH Aachen University
	Martina Cerulli	University of Salerno

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF AMSTERDAM

Webpage