FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

NATURWISSENSCHAFTLICHE
FAKULTÄT

Practical Issues, Adversarial Attacks and Robustness

Lecture “Mathematics of Learning” 2022/2023

Wigand Rathmann
Friedrich-Alexander-Universität Erlangen-Nürnberg

WIRTSCHAFTS
MATHEMATIK

Current Status

What we have learned on neural networks so far:

- They perform great in task like classification, image and language processing, gaming, etc.
- They are typically composed of elementary artificial neurons of the form $\Phi(x) = \psi(w \cdot x + b)$ with weights w, bias b, and activation ψ.
- Their free parameters can be computed using stochastic gradient descent of a loss function.
- They can theoretically solve any task, i.e., approximate any given function?
- convergence of stochastic gradient descent (SGD) (Last lecture)

Current Status

What we have learned on neural networks so far:

- They perform great in task like classification, image and language processing, gaming, etc.
- They are typically composed of elementary artificial neurons of the form $\Phi(x) = \psi(w \cdot x + b)$ with weights w, bias b, and activation ψ.
- Their free parameters can be computed using stochastic gradient descent of a loss function.
- They can theoretically solve any task, i.e., approximate any given function?
- convergence of stochastic gradient descent (SGD) (Last lecture)

Current Status

What we have learned on neural networks so far:

- They perform great in task like classification, image and language processing, gaming, etc.
- They are typically composed of elementary artificial neurons of the form $\Phi(x) = \psi(w \cdot x + b)$ with weights w, bias b, and activation ψ.
- Their free parameters can be computed using stochastic gradient descent of a loss function.
- They can theoretically solve any task, i.e., approximate any given function?
- convergence of stochastic gradient descent (SGD) (Last lecture)
- How can we decide on the structure of the NN?

Current Status

What we have learned on neural networks so far:

- They perform great in task like classification, image and language processing, gaming, etc.
- They are typically composed of elementary artificial neurons of the form $\Phi(x) = \psi(w \cdot x + b)$ with weights w, bias b, and activation ψ.
- Their free parameters can be computed using stochastic gradient descent of a loss function.
- They can theoretically solve any task, i.e., approximate any given function?
- convergence of stochastic gradient descent (SGD) (Last lecture)
- How can we decide on the structure of the NN?
- Can they be made stable?

Practical Issues in Training Neural Networks

from Elements Statistical Learning, ch. 11.5

Issue in determining starting values: Exact zero weights lead to zero derivatives, algorithm never moves. Starting instead with large weights often leads to poor solutions.
way out:

- if the weights are near zero, then the operative part of the sigmoid is roughly linear, hence neural network collapses into an approximately linear model
- Usually starting values for weights are chosen to be random values near zero
- hence nearly linear model at start, becomes nonlinear as the weights increase

Number of Hidden Units and Layers

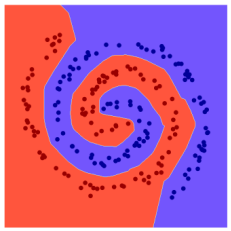
generally speaking: better to have too many hidden units than too few

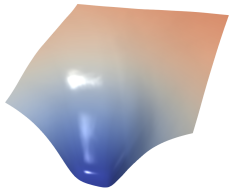
- With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data
- with too many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.
- Typically the number of hidden units is somewhere in the range of 5 to 100, increasing with the number of inputs and number of training cases.
- It is most common to put down a reasonably large number of units and train them with regularization.
- Choice of the number of hidden layers is guided by background knowledge and experimentation.
- Each layer extracts features of the input for regression or classification. Use of multiple hidden layers allows construction of hierarchical features at different levels of resolution.

Multiple Minima

- error function is nonconvex, possessing many local minima.
- thus, final solution dependent on the choice of start.
- one must at least try a number of random starting configurations, choose the solution giving lowest (penalized) error.
- bagging: averages the predictions of networks training from randomly perturbed versions of the training data.

Overfitted Networks

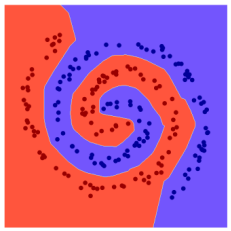


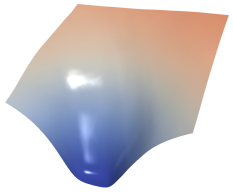


Training data, decision boundaries and loss landscape[1] of well-trained network

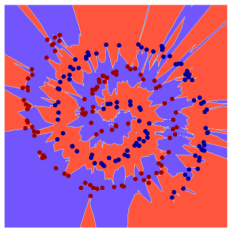
[1] Image courtesy Tom Goldstein https://www.cs.umd.edu/~tomg/projects/generalization/

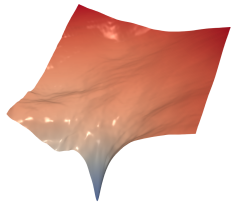
Overfitted Networks





Training data, decision boundaries and loss landscape[1] of well-trained network

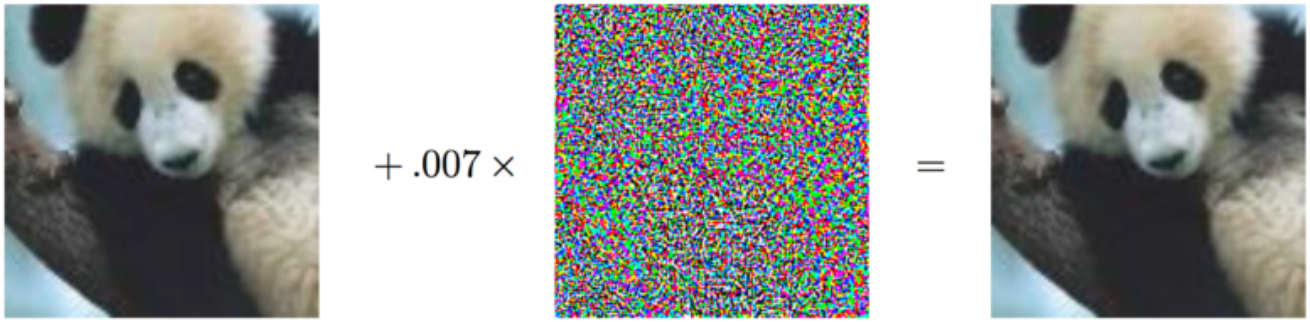




Training data, decision boundaries and loss landscape of dramatically overfitted network

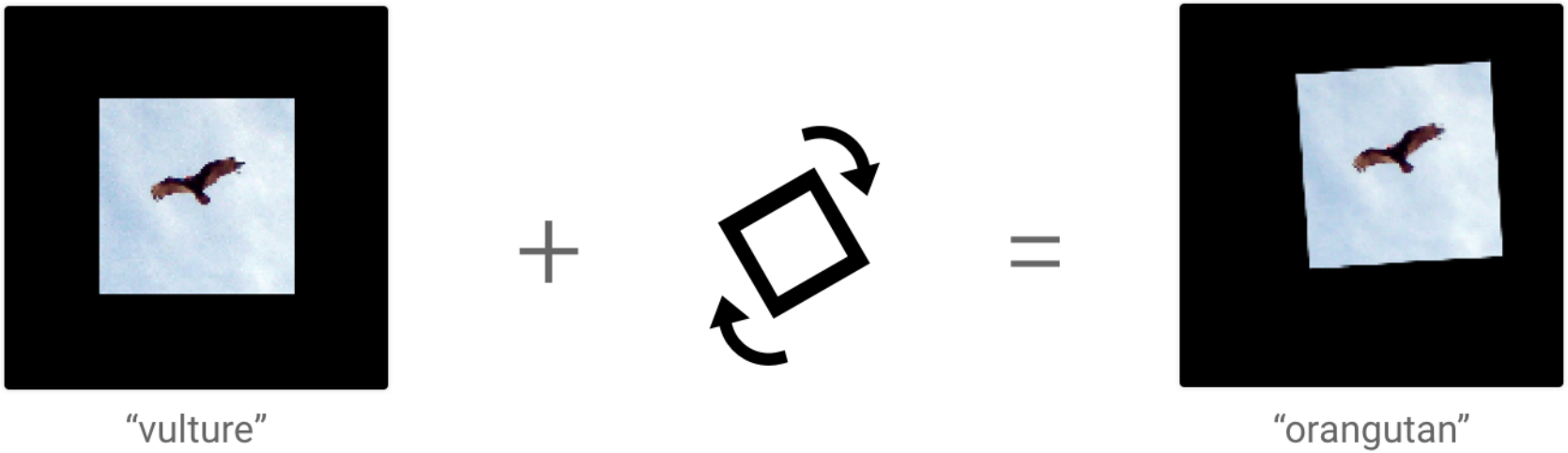
[1] Image courtesy Tom Goldstein https://www.cs.umd.edu/~tomg/projects/generalization/

Adversarial Attacks on Neural Networks



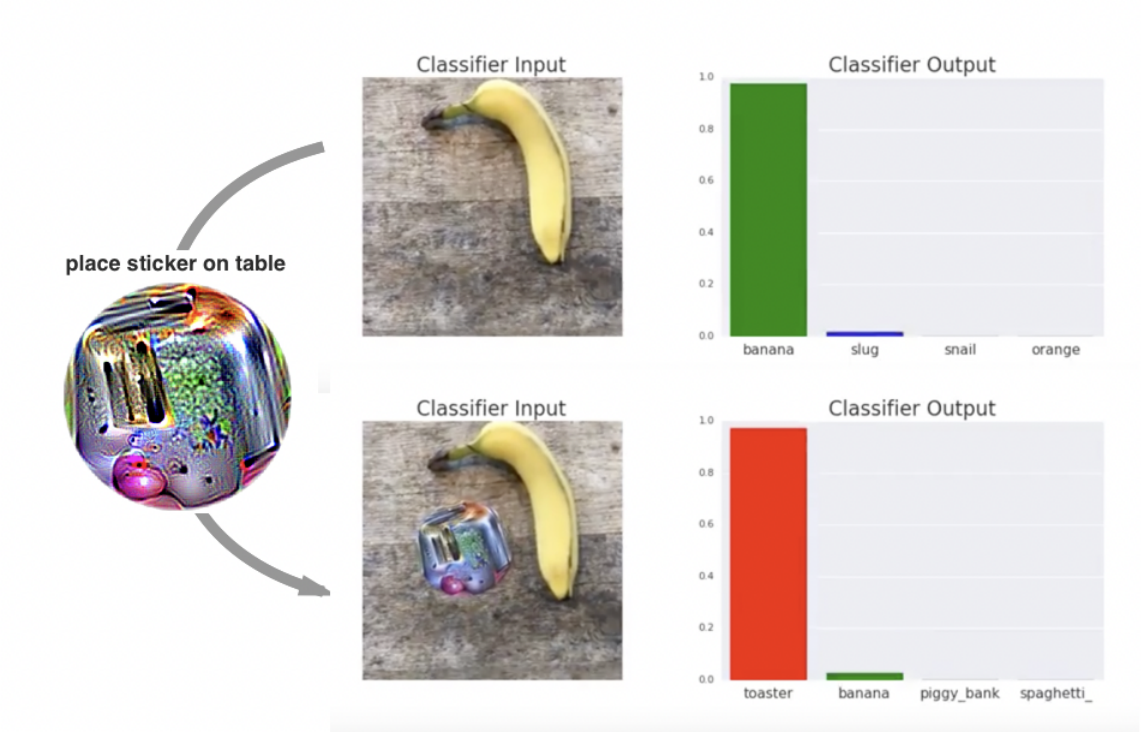
Adversarial noise

Adversarial Attacks on Neural Networks



Adversarial rotation

Adversarial Attacks on Neural Networks



Adversarial sticker

Adversarial Attacks on Neural Networks

Adversarial T-shirt

Adversarial Attacks on Neural Networks

Neural Network recognizes ”Tempo Limit 45mph”

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.
Adversary: "Can we find a very small perturbation δ such that $f_\theta(x + \delta) \neq y$?"

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.
Adversary: "Can we find a very small perturbation δ such that $f_\theta(x + \delta) \neq y$?"
Mathematician: "Sure, just give me budget ε and norm for δ and a loss function."

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.
Adversary: "Can we find a very small perturbation δ such that $f_\theta(x + \delta) \neq y$?"
Mathematician: "Sure, just give me budget ε and norm for δ and a loss function."

$$\max_{\|\delta\| \leq \varepsilon} \ell(f_\theta(x + \delta), y)$$

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.
Adversary: "Can we find a very small perturbation δ such that $f_\theta(x + \delta) \neq y$?"
Mathematician: "Sure, just give me budget ε and norm for δ and a loss function."

$$\max_{\|\delta\| \leq \varepsilon} \ell(f_\theta(x + \delta), y) \iff \max_{\|x_{\text{adv}} - x\| \leq \varepsilon} \ell(f_\theta(x_{\text{adv}}), y)$$

Adversarial Attacks

Let's say we have a neural network $f_\theta : X \to Y$ which classifies $x \in X$ correctly as $y = f_\theta(x)$.
Adversary: "Can we find a very small perturbation δ such that $f_\theta(x + \delta) \neq y$?"
Mathematician: "Sure, just give me budget ε and norm for δ and a loss function."

$$\max_{\|\delta\| \leq \varepsilon} \ell(f_\theta(x + \delta), y) \iff \max_{\|x_{\text{adv}} - x\| \leq \varepsilon} \ell(f_\theta(x_{\text{adv}}), y)$$

This can be solved with *gradient algorithm*, here written as gradient ascent method. Be careful: gradient (descent/ascent) algorithms require *unconstrained* problems. In case of constraints (such as budget restriction), we need to make sure the iterates are feasible. This is in case of simple constraints achieved by an subsequent projection step onto the feasible region. The projection is denoted by a function Π

$$\begin{aligned} x_{\text{adv}}^{(0)} &\leftarrow x, \\ x_{\text{adv}}^{(k+1)} &\leftarrow \Pi_{B_\varepsilon(x)} \left(x_{\text{adv}}^{(k)} + \eta \nabla_x \ell(f_\theta(x_{\text{adv}}^{(k)}), y) \right) \end{aligned}$$

Fast Gradient Sign Method

Often, attacks (e.g., adversarial stickers) are modeled with the ℓ_∞-norm.

Fast Gradient Sign Method

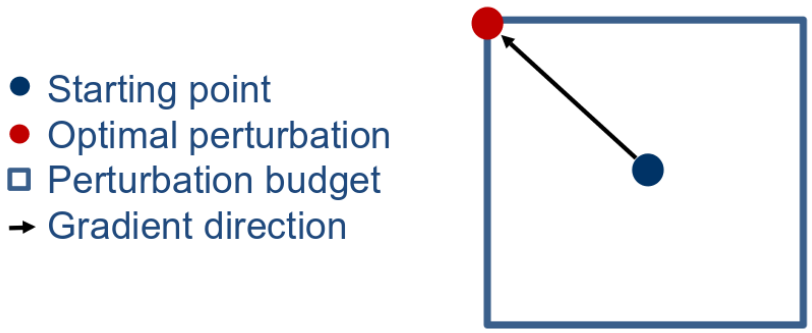
Often, attacks (e.g., adversarial stickers) are modeled with the ℓ_∞-norm.
In this case one can find a very efficient one-step attack, the Fast Gradient Sign Method FGSM:

$$x_{\text{adv}} = x + \varepsilon \operatorname{sign}(\nabla_x \ell(f_\theta(x), y))$$

Fast Gradient Sign Method

Often, attacks (e.g., adversarial stickers) are modeled with the ℓ_∞-norm.
In this case one can find a very efficient one-step attack, the Fast Gradient Sign Method FGSM:

$$x_{\text{adv}} = x + \varepsilon \,\text{sign}(\nabla_x \ell(f_\theta(x), y))$$



How to make neural networks more robust?

The easy way:

Data augmentation

How to make neural networks more robust?

The easy way: Add data, data, and more data!

Data augmentation

How to make neural networks more robust?

The easy way: Add data, data, and more data! Careful: Rotations can alter the semantics!

Data augmentation

Adversarial Training

Idea by Madry, 2017: Incorporate the attack into the training.

Adversarial Training

Idea by Madry, 2017: Incorporate the attack into the training.

Empirical Risk Minimization

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \ell(f_\theta(x), y)$$

Adversarial Training

Idea by Madry, 2017: Incorporate the attack into the training.

Empirical Risk Minimization

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \ell(f_\theta(x), y)$$

Adversarial Attacks

$$\max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

Adversarial Training

Idea by Madry, 2017: Incorporate the attack into the training.

Empirical Risk Minimization

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \ell(f_\theta(x), y)$$

Adversarial Attacks

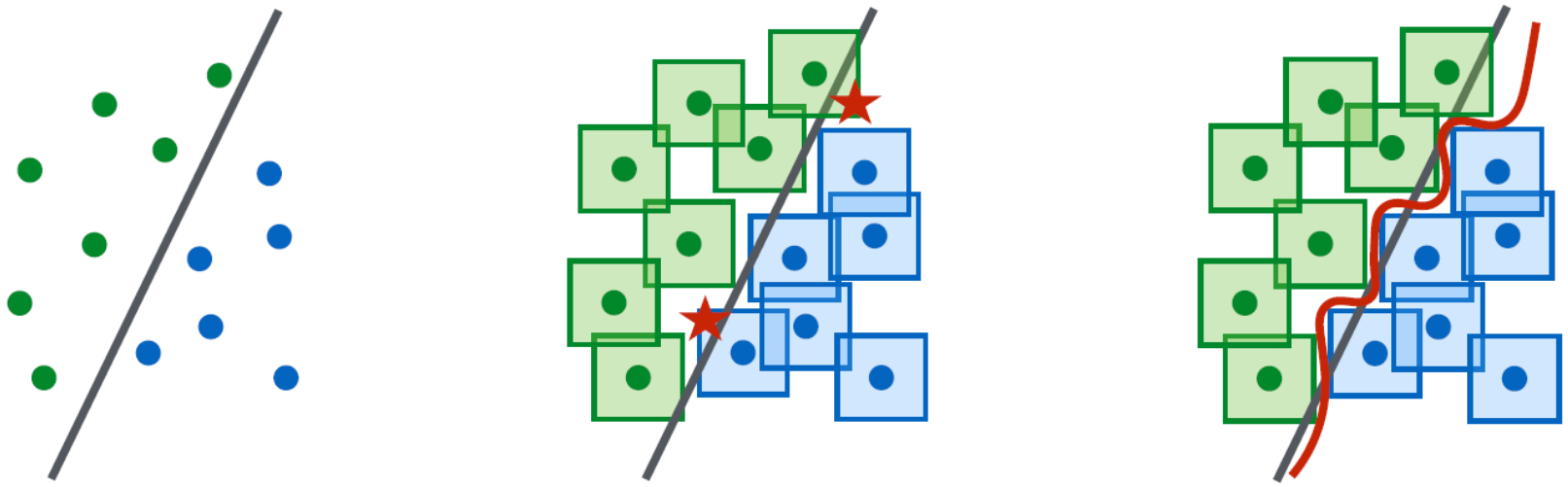
$$\max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

The mathematical way: Adversarial Training

Replace the training pair (x, y) by adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

Visualization of Adversarial Training



Left: Training data and decision boundary. **Middle:** ℓ_∞-balls around training data with adversarial examples. **Right:** Adversarial training corrects decision boundary[2]

[2] Taken from Madry, 2017

Discussion of Adversarial Training

Adversarial Training

Replace the training pair (x, y) by adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

PROs:

- Can be implemented very cheaply (*Adversarial training for free!*, Shafahi, 2019).

Discussion of Adversarial Training

Adversarial Training

Replace the training pair (x, y) by adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

PROs:

- Can be implemented very cheaply (*Adversarial training for free!*, Shafahi, 2019).
- Can significantly improve robustness against adversarial attacks.

Discussion of Adversarial Training

Adversarial Training

Replace the training pair (x, y) by adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

PROs:

- Can be implemented very cheaply (*Adversarial training for free!*, Shafahi, 2019).
- Can significantly improve robustness against adversarial attacks.

CONs:

- Only regularizes “around” the training data.

Discussion of Adversarial Training

Adversarial Training

Replace the training pair (x, y) by adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

PROs:

- Can be implemented very cheaply (*Adversarial training for free!*, Shafahi, 2019).
- Can significantly improve robustness against adversarial attacks.

CONs:

- Only regularizes “around” the training data.
- Trades stability for accuracy.

Lipschitz Continuity implies Stability

Sufficient for stable networks $f_\theta : X \to Y$ is Lipschitz continuity

$$(\forall x, x' \in X) \quad \|f_\theta(x) - f_\theta(x')\|_Y \leq L \|x - x'\|_X .$$

Lipschitz Continuity implies Stability

Sufficient for stable networks $f_\theta : X \to Y$ is Lipschitz continuity

$$(\forall x, x' \in X) \quad \|f_\theta(x) - f_\theta(x')\|_Y \leq L \|x - x'\|_X .$$

(Mathematical) Challenges: We want to minimize the Lipschitz constant

$$\mathrm{Lip}(f_\theta) := \sup_{x, x' \in X} \frac{\|f_\theta(x) - f_\theta(x')\|_Y}{\|x - x'\|_X},$$

Lipschitz Continuity implies Stability

Sufficient for stable networks $f_\theta : X \to Y$ is Lipschitz continuity

$$(\forall x, x' \in X) \quad \|f_\theta(x) - f_\theta(x')\|_Y \leq L \|x - x'\|_X .$$

(Mathematical) Challenges: We want to minimize the Lipschitz constant

$$\mathrm{Lip}(f_\theta) := \sup_{x, x' \in X} \frac{\|f_\theta(x) - f_\theta(x')\|_Y}{\|x - x'\|_X},$$

which has the following unfavorable properties

- sometimes impossible (NP hard) to compute, Scaman, 2018,
- non-differentiable,
- not strictly convex.

Lipschitz Constant of a Neural Network

Can we at least estimate the Lipschitz constant of a neural network?

Lipschitz Constant of a Neural Network

Can we at least estimate the Lipschitz constant of a neural network?

Lemma: Lipschitz constant of a composition

Assume that $f : X \to Y$ has Lipschitz constant $\mathrm{Lip}(f)$ and $g : Y \to Z$ has Lipschitz constant $\mathrm{Lip}(g)$. Then the Lipschitz constant of $h := g \circ f$ satisfies $\mathrm{Lip}(h) \leq \mathrm{Lip}(g) \cdot \mathrm{Lip}(f)$.

Lipschitz Constant of a Neural Network

Can we at least estimate the Lipschitz constant of a neural network?

Lemma: Lipschitz constant of a composition

Assume that $f : X \to Y$ has Lipschitz constant $\mathrm{Lip}(f)$ and $g : Y \to Z$ has Lipschitz constant $\mathrm{Lip}(g)$. Then the Lipschitz constant of $h := g \circ f$ satisfies $\mathrm{Lip}(h) \leq \mathrm{Lip}(g) \cdot \mathrm{Lip}(f)$.

Proof: For all $x, x' \in X$ it holds

$$\|h(x) - h(x')\| = \|g(f(x)) - g(f(x'))\| \leq L_g \|f(x) - f(x')\| \leq L_g \cdot L_f \|x - x'\| .$$

Lipschitz Constant of a Neural Network

Let us take assume that we have a neural network of the form

$$f_\theta(x) = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1(x), \qquad \Phi_l(x) = \psi(W_l x + b_l),$$

where the activation function satisfies $\mathrm{Lip}(\psi) = 1$, e.g., ReLU, TanH, Sigmoid,...

Lipschitz Constant of a Neural Network

Let us take assume that we have a neural network of the form

$$f_\theta(x) = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1(x), \qquad \Phi_l(x) = \psi(W_l x + b_l),$$

where the activation function satisfies $\mathrm{Lip}(\psi) = 1$, e.g., ReLU, TanH, Sigmoid,...
Then the Lipschitz constant of f_θ can be bounded by

$$\mathrm{Lip}(f_\theta) \leq \prod_{i=1}^{L} \mathrm{Lip}(\Phi_l) = \prod_{i=1}^{L} \|W_l\| .$$

Lipschitz Constant of a Neural Network

Let us take assume that we have a neural network of the form

$$f_\theta(x) = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1(x), \qquad \Phi_l(x) = \psi(W_l x + b_l),$$

where the activation function satisfies $\mathrm{Lip}(\psi) = 1$, e.g., ReLU, TanH, Sigmoid,...
Then the Lipschitz constant of f_θ can be bounded by

$$\mathrm{Lip}(f_\theta) \leq \prod_{i=1}^{L} \mathrm{Lip}(\Phi_l) = \prod_{i=1}^{L} \|W_l\| .$$

To enforce $\mathrm{Lip}(\Phi_l) \leq \lambda$ one can project (Gouk, 2020):

$$\text{update } W_l \text{ with SGD,}$$
$$W_l \leftarrow \frac{1}{\max\left(1, \frac{\|W_l\|}{\lambda}\right)} W_l.$$

Sharpness of the Lipschitz Estimate?

Estimate from the last slide:

$$\mathrm{Lip}(f_\theta) \leq \prod_{i=1}^{L} \|W_l\| \,.$$

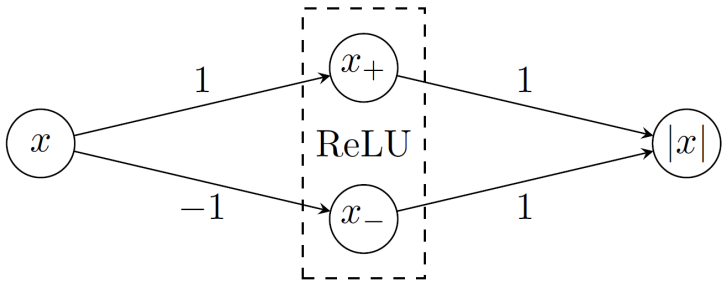
Q: Is this really an inequality?

Sharpness of the Lipschitz Estimate?

Estimate from the last slide:

$$\mathrm{Lip}(f_\theta) \leq \prod_{i=1}^{L} \|W_i\| .$$

Q: Is this really an inequality?
A: YES, see the following example

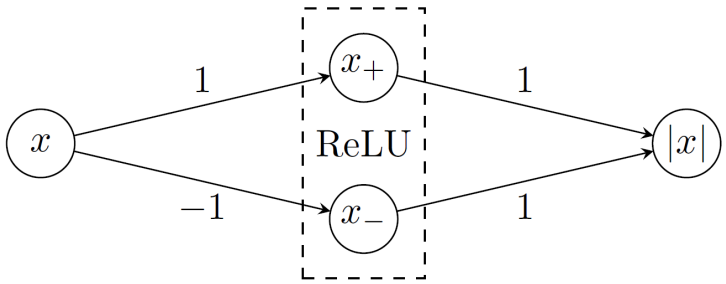


Sharpness of the Lipschitz Estimate?

Estimate from the last slide:

$$\mathrm{Lip}(f_\theta) \leq \prod_{i=1}^{L} \|W_i\| .$$

Q: Is this really an inequality?
A: YES, see the following example



$$f(x) = \underbrace{(1,1)}_{\|\cdot\|=\sqrt{2}} \mathit{ReLU}\left(\underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}}_{\|\cdot\|=\sqrt{2}} x \right) = |x|.$$

But: $\mathrm{Lip}(f) = 1$

Can we do this better via an ex-ante protection?

Adversarial Training as a robust optimization problem

repeat adversarial attack:

$$\min_{\theta} \frac{1}{|T|} \sum_{(x,y)\in T} \max_{\|\delta\|\leq\varepsilon} \ell(f_\theta(x+\delta), y)$$

This is a so-called *robust optimization problem*. A robust problem typically is of the form

$$\min_{\theta,\ldots\in P} \max_{z_1\ldots\in Z} g(\theta, \ldots, z),$$

where $z \in Z, \theta \in P$ denote the respective feasible regions.

Alternatively, we consider *uncertainties (adversaries)* against which a solution needs to be protected against and write the robust problems with uncertain constraints:

$$\min_{\theta \in P} \theta$$
$$\max_{z \in Z} g(\theta, ..., z) \leq \theta$$

or

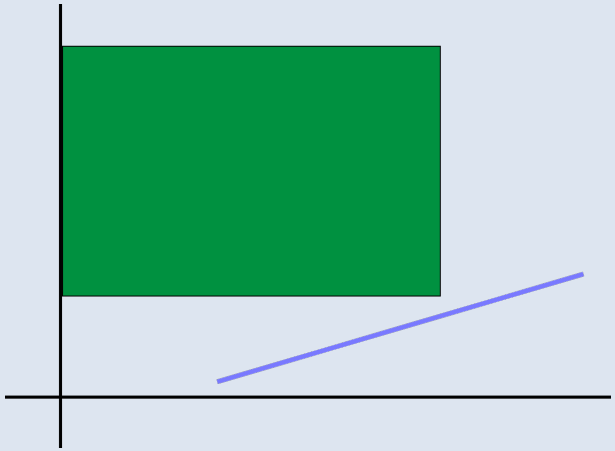
$$\min_{\theta \in P} \theta$$
$$g(\theta, ..., z) \leq \theta \; \forall \; z \in Z$$

Excursion robust optimization

uncertain linear Optimization problem

Let $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

$$\left\{ \min_{x \in \mathbb{R}^n} \left\{ c^\top x : Ax \geq b \right\} \right\}_{(c,A,b) \in \mathcal{U}}$$

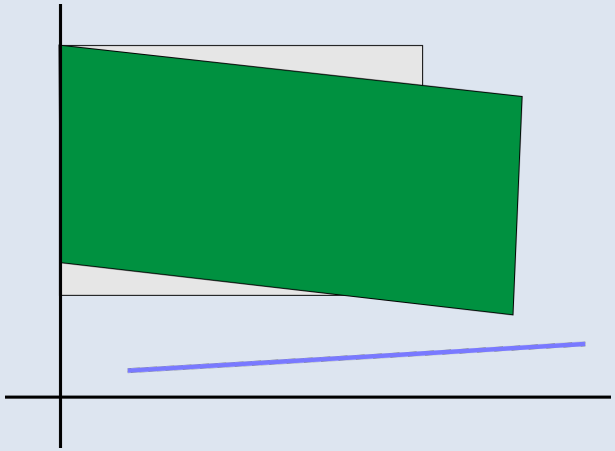


Excursion robust optimization

uncertain linear Optimization problem

Let $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

$$\left\{ \min_{x \in \mathbb{R}^n} \left\{ c^\top x : Ax \geq b \right\} \right\}_{(c,A,b) \in \mathcal{U}}$$

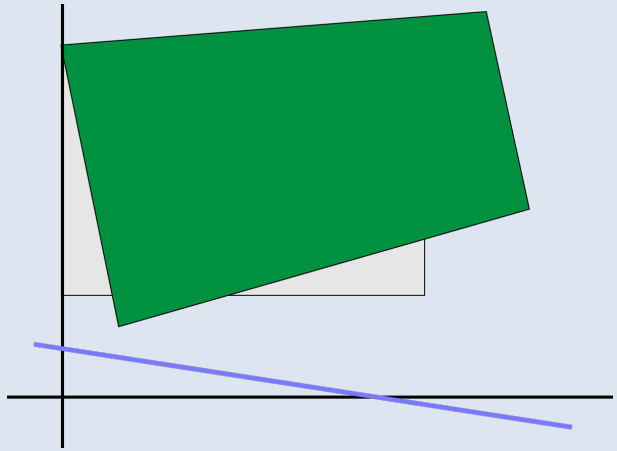


Excursion robust optimization

uncertain linear Optimization problem

Let $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

$$\left\{ \min_{x \in \mathbb{R}^n} \left\{ c^\top x : Ax \geq b \right\} \right\}_{(c,A,b) \in \mathcal{U}}$$

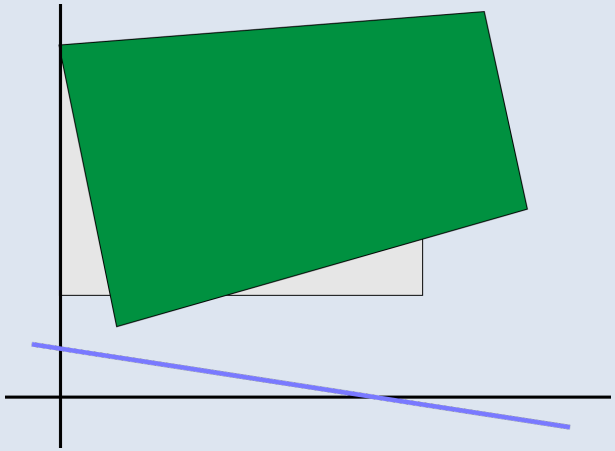


Excursion robust optimization

uncertain linear Optimization problem

Let $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$

$$\left\{ \min_{x \in \mathbb{R}^n} \left\{ c^\top x : Ax \geq b \right\} \right\}_{(c,A,b) \in \mathcal{U}}$$



Notation

1. *strict robustness*: here-and-now decisions
2. *robust feasible*: x feasible for al $(c, A, b) \in \mathcal{U}$
3. *robust optimal*: robust feasible x with best guaranteed value

robust counterpart

$$\min_{x \in \mathbb{R}^n} \left\{ c^*(x) := \right\} .$$

robust counterpart

$$\min_{x \in \mathbb{R}^n} \left\{ c^*(x) := \right\}.$$

robust counterpart

$$\min_{x \in \mathbb{R}^n} \left\{ c^*(x) := \sup_{(c,A,b) \in \mathcal{U}} \left\{ c^\top x \colon Ax \geq b \text{ for all } (c, A, b) \in \mathcal{U} \right\} \right\}.$$

robust counterpart

$$\min_{x \in \mathbb{R}^n} \left\{ c^*(x) := \sup_{(c,A,b) \in \mathcal{U}} \left\{ c^\top x \colon Ax \geq b \text{ for all } (c, A, b) \in \mathcal{U} \right\} \right\}.$$

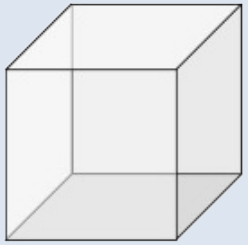
How can we reformulate the robust counterpart of a robust inequality

$$a^\top x - b \geq 0 \quad \text{for al } (a, b) \in \mathcal{U}$$

such that it is algorithmically tractable?

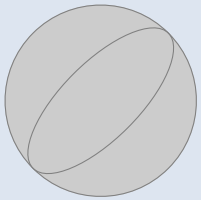
Affine Parametrisation of $\mathcal{U}$

- *nominal value* $\bar{a} \in \mathbb{R}^n, \bar{b} \in \mathbb{R}$
- $\mathcal{Z} \in \mathbb{R}^L$ 'standard' (convex) uncertainty (adversarial budget)
- $u \in \mathcal{Z}$ *perturbation vector*



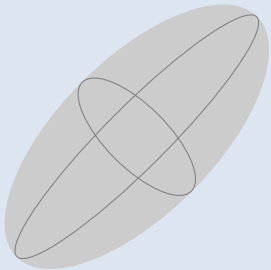
Affine Parametrisation of $\mathcal{U}$

- *nominal value* $\bar{a} \in \mathbb{R}^n, \bar{b} \in \mathbb{R}$
- $\mathcal{Z} \in \mathbb{R}^L$ 'standard' (convex) uncertainty (adversarial budget)
- $u \in \mathcal{Z}$ *perturbation vector*



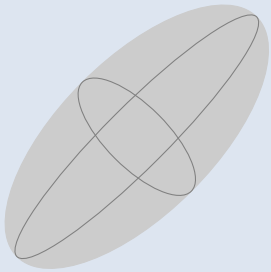
Affine Parametrisation of $\mathcal{U}$

- *nominal value* $\bar{a} \in \mathbb{R}^n, \bar{b} \in \mathbb{R}$
- $\mathcal{Z} \in \mathbb{R}^L$ 'standard' (convex) uncertainty (adversarial budget)
- $u \in \mathcal{Z}$ *perturbation vector*
- $P \in \mathbb{R}^{n \times L}$, $p \in \mathbb{R}^L$, $u \in \mathbb{R}^L$



Affine Parametrisation of $\mathcal{U}$

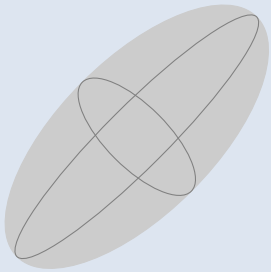
- *nominal value* $\bar{a} \in \mathbb{R}^n, \bar{b} \in \mathbb{R}$
- $\mathcal{Z} \in \mathbb{R}^L$ 'standard' (convex) uncertainty (adversarial budget)
- $u \in \mathcal{Z}$ *perturbation vector*
- $P \in \mathbb{R}^{n \times L}$, $p \in \mathbb{R}^L$, $u \in \mathbb{R}^L$



$$\mathcal{U} = \left\{ \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \bar{a} + Pu \\ \bar{b} + p^\top u \end{pmatrix} : u \in \mathcal{Z} \right\}$$

Affine Parametrisation of $\mathcal{U}$

- *nominal value* $\bar{a} \in \mathbb{R}^n, \bar{b} \in \mathbb{R}$
- $\mathcal{Z} \in \mathbb{R}^L$ 'standard' (convex) uncertainty (adversarial budget)
- $u \in \mathcal{Z}$ *perturbation vector*
- $P \in \mathbb{R}^{n \times L}$, $p \in \mathbb{R}^L$, $u \in \mathbb{R}^L$



$$\mathcal{U} = \left\{ \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \bar{a} + Pu \\ \bar{b} + p^\top u \end{pmatrix} : u \in \mathcal{Z} \right\}$$

Example

$$\mathcal{U} = \left\{ \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in \mathbb{R}^2 : \frac{a_1^2}{4} + \frac{a_2^2}{9} \leq 1 \right\} = \left\{ \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} : \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in \mathcal{Z} \right\}$$

$$\mathcal{Z} = \left\{ \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in \mathbb{R}^2 : a_1^2 + a_2^2 \leq 1 \right\}$$

We write robust inequality as

$$(\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \in \mathcal{Z}.$$

We write robust inequality as

$$(\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \in \mathcal{Z}.$$

Which uncertainty sets and which robustifications lead to algorithmically tractable robust counterparts?

The adversarial training that we have seen is an algorithmically tractable problem, as it can be solved via gradient descent algorithms.
Here we will look into a different approach in robust optimization that is often used, namely that of reformulating a constraint.

Theorem (ellipsoidal uncertainty sets)

The robust counterpart of

$$(\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \text{ mit } \|u\|_2 \leq r$$

can equivalently be reformulated as

$$\bar{a}^\top x - r \left\| P^\top x \right\|_2 \geq b.$$

Theorem (ellipsoidal uncertainty sets)

The robust counterpart of

$$(\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \text{ mit } \|u\|_2 \leq r$$

can equivalently be reformulated as

$$\bar{a}^\top x - r \left\| P^\top x \right\|_2 \geq b.$$

Proof

$$\begin{aligned}
& (\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \text{ mit } \|u\|_2 \leq r \\
\iff \quad & \min_{\{u:\, \|u\|_2 \leq r\}} (\bar{a} + Pu)^\top x \geq b \\
\iff \quad & \bar{a}^\top x + \min_{\{u:\, \|u\|_2 \leq r\}} (Pu)^\top x \geq b \\
\iff \quad & \bar{a}^\top x + \min_{\{u:\, \|u\|_2 \leq r\}} u^\top (P^\top x) \geq b.
\end{aligned}$$

$v^\top w = \|v\|_2 \, \|w\|_2 \cos(v, w)$ minimal
for $\cos(v, w) = -1 \Rightarrow v = -w$

Theorem (ellipsoidal uncertainty sets)

The robust counterpart of

$$(\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \text{ mit } \|u\|_2 \leq r$$

can equivalently be reformulated as

$$\bar{a}^\top x - r \left\| P^\top x \right\|_2 \geq b.$$

Proof

$$\begin{aligned}
& (\bar{a} + Pu)^\top x \geq b \quad \text{for all } u \text{ mit } \|u\|_2 \leq r \\
\iff \quad & \min_{\{u:\, \|u\|_2 \leq r\}} (\bar{a} + Pu)^\top x \geq b \\
\iff \quad & \bar{a}^\top x + \min_{\{u:\, \|u\|_2 \leq r\}} (Pu)^\top x \geq b \\
\iff \quad & \bar{a}^\top x + \min_{\{u:\, \|u\|_2 \leq r\}} u^\top (P^\top x) \geq b.
\end{aligned}$$

$v^\top w = \|v\|_2 \, \|w\|_2 \cos(v, w)$ minimal
for $\cos(v, w) = -1 \Rightarrow v = -w$

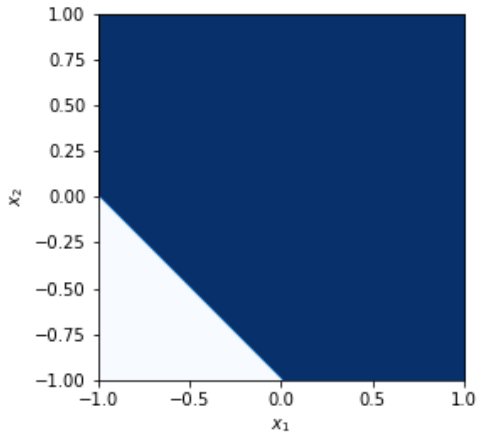
$$\min_{\{u:\, \|u\|_2 \leq r\}} u^\top (P^\top x) = -r \frac{(P^\top x)^\top}{\|P^\top x\|_2} (P^\top x) = -r \left\| P^\top x \right\|_2$$

Inserting leads to $\bar{a}^\top x - r \left\| P^\top x \right\|_2 \geq b$.

Discussion

- big advantage: in comparison to min-max problem or to infinitely many linear inequalities, the reformulation is *only one (conic) inequality over the Lorentz cone*.
- *safety term* $r \left\| P^\top x \right\|_2$ depends on x
- we have assumed the Euclidean norm. Of course, this can also be done in the infinity norm (leads to a linear optimization problem instead of a conic one.)

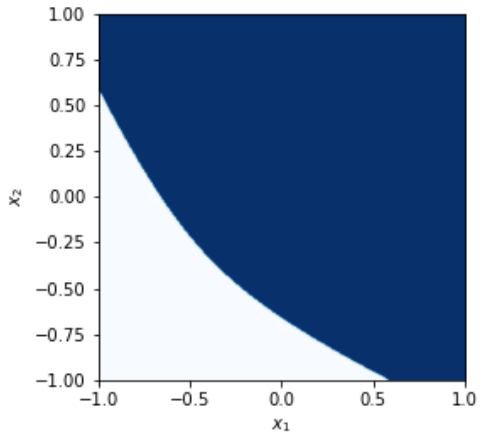
Example



$$x_1 + x_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

Example

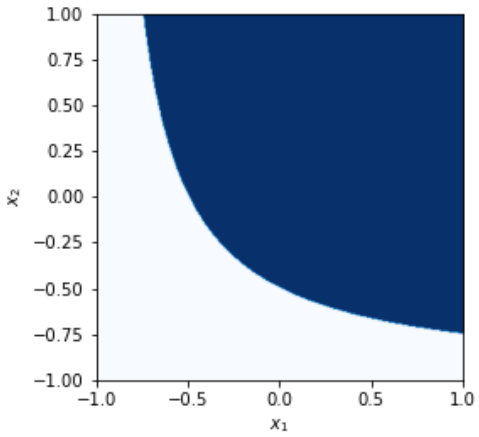


$$(1+u_1)x_1 + (1+u_2)x_2 \geq -1, \|u\|_2 \leq \frac{1}{2}$$

$$\Leftrightarrow x_1 + x_2 - \frac{1}{2}\|x\|_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

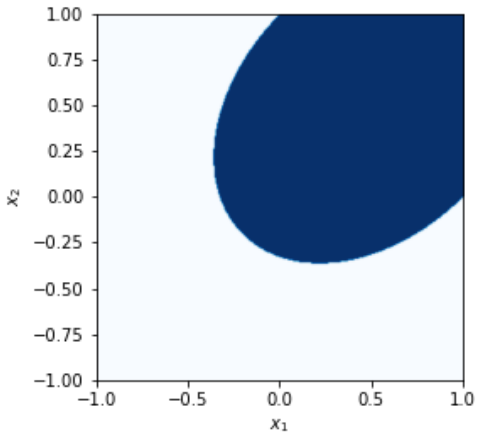
Example



$$(1+u_1)x_1+(1+u_2)x_2 \geq -1, \|u\|_2 \leq 1$$
$$\Leftrightarrow x_1+x_2-1||x||_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

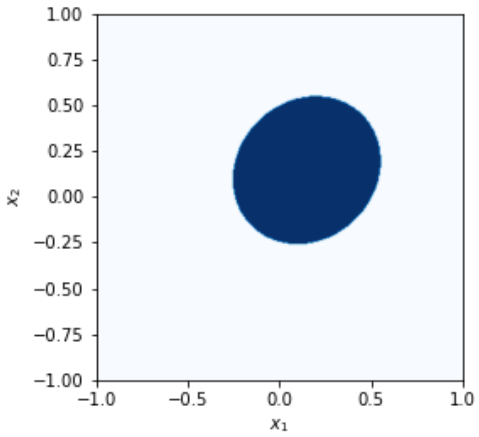
Example



$$(1+u_1)x_1+(1+u_2)x_2 \geq -1, \|u\|_2 \leq 2$$
$$\Leftrightarrow x_1+x_2-2||x||_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

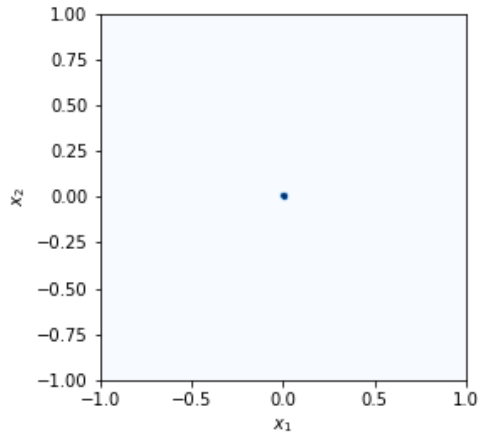
Example



$$(1+u_1)x_1+(1+u_2)x_2 \geq -1, \|u\|_2 \leq 3$$
$$\Leftrightarrow x_1+x_2-3||x||_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

Example



$$(1+u_1)x_1+(1+u_2)x_2 \geq -1, \|u\|_2 \leq 50$$
$$\Leftrightarrow x_1+x_2-50||x||_2 \geq -1$$

more information, details, etc. in lecture 'Robust Optimization' (BSc and MSc), and in Advanced Topics in Maths of Learning.

Conclusion

Today we have

- learned about adversarial attacks and how to defend them,
- showed that more generally the protection against adversaries can be cast as robust optimization problem

Conclusion

Today we have

- learned about adversarial attacks and how to defend them,
- understood the necessity and limitations of Lipschitz-regularized learning
- showed that more generally the protection against adversaries can be cast as robust optimization problem