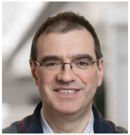
Distribution-Dissimilarities in Machine Learning

Carl-Johann SIMON-GABRIEL

October 22, 2020

Organizers and Collaborators: Thanks!


Yann Ollivier


Léon Bottou


David Lopez-Paz


Bernhard Schölkopf


Ilya Tolstikhin


Sylvain Gelly


Olivier Bousquet


Lester Mackey


Alessandro Barp


Noman Sheik


Julia Hörrmann


Andreas Krause

Outline

1. Introduction: classifier-based distribution-dissimilarities

2. Maximum Mean Discrepancies (MMD)

3. Adversarial Vulnerability of Neural Networks

Outline

1. Introduction: classifier-based distribution-dissimilarities

2. Maximum Mean Discrepancies (MMD)

3. Adversarial Vulnerability of Neural Networks

GANs and distribution-dissimilarities

- GAN algorithm

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by minimizing the dissimilarity between real and fake data

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by minimizing the dissimilarity between real and fake data
  as measured by a classifier that tries to separate them

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by minimizing the dissimilarity between real and fake data
  as measured by a classifier that tries to separate them

- GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by minimizing the dissimilarity between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs use a classifier-based distribution-dissimilarity.

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs use a classifier-based distribution-dissimilarity.

$$D(P, Q) :=$$

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs use a classifier-based distribution-dissimilarity.

$$D(P, Q) := \left\{ \begin{array}{l} \text{expected reward of a classifier } \varphi \\ \\ \\ \end{array} \right.$$

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs use a classifier-based distribution-dissimilarity.

$$D(P, Q) := \left\{ \begin{array}{l} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \\ \end{array} \right.$$

GANs and distribution-dissimilarities

- GAN algorithm

  Train an image-generator
  by **minimizing the dissimilarity** between real and fake data
  as measured by a classifier that tries to separate them

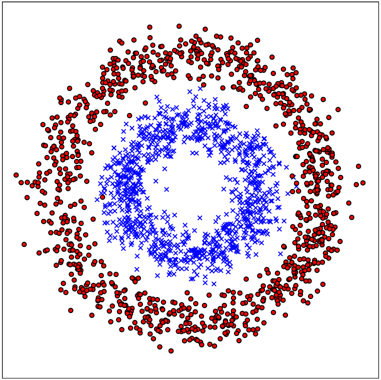
- ~~GANs need no pre-defined dissimilarity measure to compare true and fake data-distribution~~

GANs use a classifier-based distribution-dissimilarity.

$$D(P,Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathscr{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

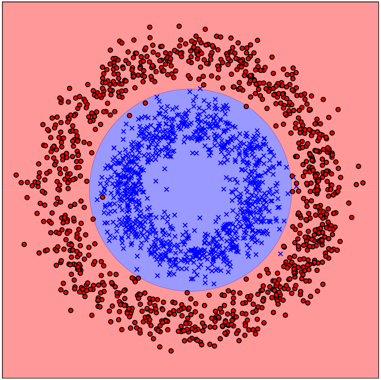
GANs and distribution-dissimilarities

$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$



$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

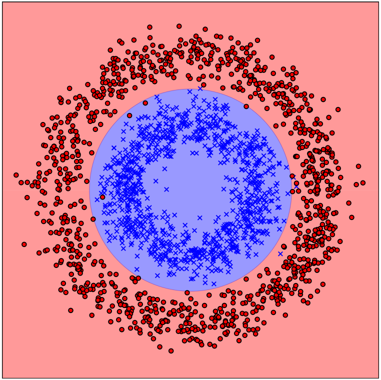
GANs and distribution-dissimilarities



$$D(P,Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

Depends on:

- classifier's *capacity* $\mathcal{F}$

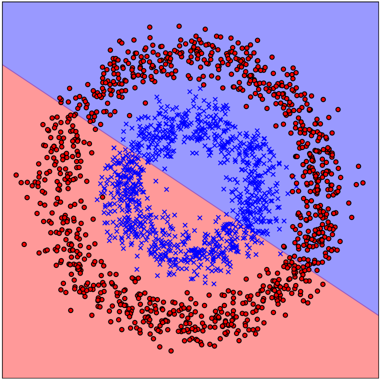


$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

GANs and distribution-dissimilarities

Depends on:

- classifier's *capacity* $\mathcal{F}$

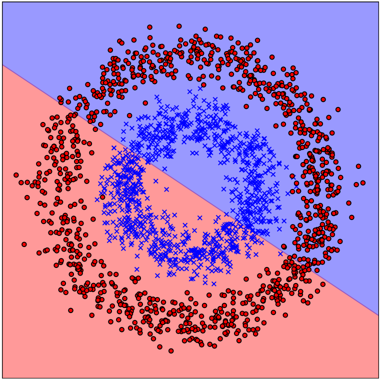


$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

GANs and distribution-dissimilarities

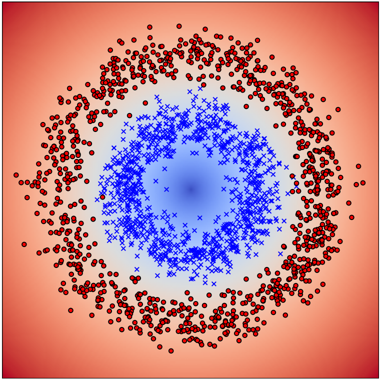
Depends on:

- classifier's *capacity* $\mathcal{F}$
- classification reward $\mathcal{R}$ (i.e. the negative loss)



$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

GANs and distribution-dissimilarities



Depends on:

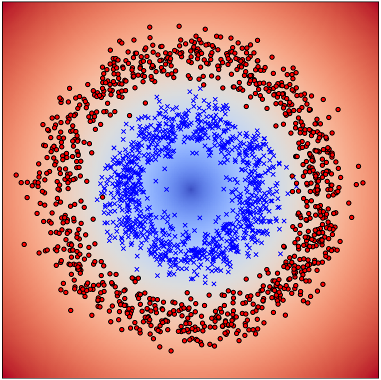
- classifier's *capacity* $\mathcal{F}$
- classification reward $\mathcal{R}$ (i.e. the negative loss)

$$D(P, Q) := \begin{cases} \text{expected reward of a classifier } \varphi \\ \text{trained over function class } \mathcal{F} \\ \text{to separate samples from } P \text{ and } Q \end{cases}$$

GANs and distribution-dissimilarities

Depends on:

- classifier's *capacity* $\mathcal{F}$
- classification reward $\mathcal{R}$ (i.e. the negative loss)



$$D(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_{X,Y} \, \mathcal{R}(\varphi(X), Y)$$

Classifier-based distribution-dissimilarities: Examples

- **IPM**: Fix $\mathcal{R}(\varphi(X), Y) = Y\varphi(X)$; vary $\mathcal{F}$.

$$D_{\mathcal{F}}(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi]$$

Classifier-based distribution-dissimilarities: Examples

- **IPM**: Fix $\mathscr{R}(\varphi(X), Y) = Y\varphi(X)$; vary $\mathscr{F}$.

$$D_{\mathscr{F}}(P, Q) = \sup_{\varphi \in \mathscr{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi] \qquad \begin{cases} \mathscr{F} = \mathscr{B}(\mathscr{C}_b) & \text{TV} \\ \mathscr{F} = \mathscr{B}(\mathrm{Lip}) & \text{Wasserstein-1} \\ \mathscr{F} = \mathscr{B}(\mathscr{H}_k) & \text{MMD} \end{cases}$$

Classifier-based distribution-dissimilarities: Examples

- **IPM**: Fix $\mathcal{R}(\varphi(X), Y) = Y\varphi(X)$; vary $\mathcal{F}$.

$$D_{\mathcal{F}}(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi] \qquad \begin{cases} \mathcal{F} = \mathcal{B}(\mathcal{C}_b) & \text{TV} \\ \mathcal{F} = \mathcal{B}(\mathrm{Lip}) & \text{Wasserstein-1} \\ \mathcal{F} = \mathcal{B}(\mathcal{H}_k) & \text{MMD} \end{cases}$$

- f**-divergence**: Vary $\mathcal{R}$; make $\mathcal{F}$ "large enough"

$$D_f(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[f^*(\varphi)]$$

Classifier-based distribution-dissimilarities: Examples

- **IPM**: Fix $\mathcal{R}(\varphi(X), Y) = Y\varphi(X)$; vary $\mathcal{F}$.

$$D_{\mathcal{F}}(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi] \quad \begin{cases} \mathcal{F} = \mathcal{B}(\mathcal{C}_b) & \text{TV} \\ \mathcal{F} = \mathcal{B}(\mathrm{Lip}) & \text{Wasserstein-1} \\ \mathcal{F} = \mathcal{B}(\mathcal{H}_k) & \text{MMD} \end{cases}$$

- f**-divergence**: Vary $\mathcal{R}$; make $\mathcal{F}$ "large enough"

$$D_f(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[f^*(\varphi)]$$

$$\begin{cases} f^*(t) = t & t \in [-1, 1] & \text{TV} \\ f^*(t) = e^{t-1} & t \in \mathbb{R} & \text{KL} \\ f^*(t) = {}^{t}/_{1-t} & t < 1 & \text{Hellinger}^2 \\ f^*(t) = -\log(1 - e^t) & t < \log 2 & \text{JS} \end{cases}$$

Classifier-based distribution-dissimilarities: Examples

- **IPM**: Fix $\mathcal{R}(\varphi(X), Y) = Y\varphi(X)$; vary $\mathcal{F}$.

$$D_{\mathcal{F}}(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi] \quad \begin{cases} \mathcal{F} = \mathcal{B}(\mathcal{C}_b) & \text{TV} \\ \mathcal{F} = \mathcal{B}(\mathrm{Lip}) & \text{Wasserstein-1} \\ \mathcal{F} = \mathcal{B}(\mathcal{H}_k) & \text{MMD} \end{cases}$$

- ***f*-divergence**: Vary $\mathcal{R}$; make $\mathcal{F}$ "large enough"

$$D_f(P, Q) = \sup_{\varphi \in \mathcal{F}} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[f^*(\varphi)]$$

$$\begin{cases} f^*(t) = t & t \in [-1, 1] & \text{TV} \\ f^*(t) = e^{t-1} & t \in \mathbb{R} & \text{KL} \\ f^*(t) = {}^{t}\!/_{1-t} & t < 1 & \text{Hellinger}^2 \\ f^*(t) = -\log(1 - e^t) & t < \log 2 & \text{JS} \end{cases}$$

- **Restricted *f*-divergences**: Vary both $\mathcal{R}$ & $\mathcal{F}$.

Factors that influence $D(P, Q)$ and GAN-like training

Factors that influence $D(P, Q)$ and GAN-like training

- Capacity $\mathcal{F}$

Factors that influence $D(P, Q)$ and GAN-like training

- Capacity $\mathscr{F}$

- Reward $\mathscr{R}$

Factors that influence $D(P, Q)$ and GAN-like training

- Capacity $\mathcal{F}$
- Reward $\mathcal{R}$
- Optimization procedure to find $\sup_{\varphi \in \mathcal{F}}$ (and $\inf_P D(P, Q)$)

Factors that influence $D(P, Q)$ and GAN-like training

- **Capacity** $\mathscr{F}$

- Reward $\mathscr{R}$

- Optimization procedure to find $\sup_{\varphi \in \mathscr{F}}$ (and $\inf_P D(P, Q)$)

Factors that influence $D(P, Q)$ and GAN-like training

- **Capacity** $\mathscr{F}$

  Special case of MMDs [SS18]

- Reward $\mathscr{R}$

- Optimization procedure to find $\sup_{\varphi \in \mathscr{F}}$ (and $\inf_P D(P, Q)$)

Factors that influence $D(P, Q)$ and GAN-like training

- **Capacity** $\mathscr{F}$
  - **Special case of MMDs** [SS18]
  - **Adversarial examples** [SOBS+19]
- Reward $\mathscr{R}$
- Optimization procedure to find $\sup_{\varphi \in \mathscr{F}}$ (and $\inf_P D(P, Q)$)

What is the right capacity?

- Too small capacity may miss relevant information

What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:

What is the right capacity?

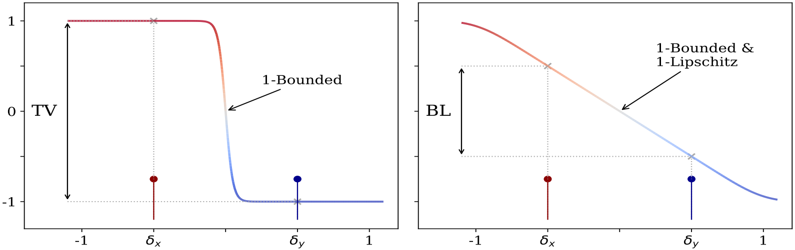
- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn

What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn
  - dissimilarity can saturate on discrete measures (e.g. samples)

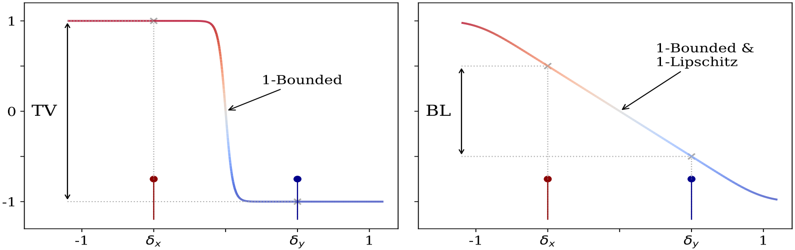
What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn
  - dissimilarity can saturate on discrete measures (e.g. samples)



What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn
  - dissimilarity can saturate on discrete measures (e.g. samples)

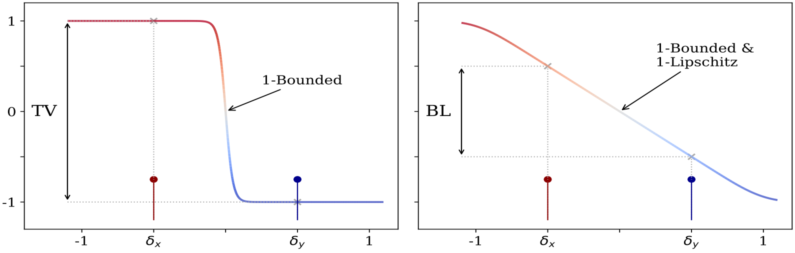


Goals:

1. When is the dissimilarity *perfectly discriminative*? (i.e. $D(P, Q) = 0 \Rightarrow P = Q$)

What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn
  - dissimilarity can saturate on discrete measures (e.g. samples)

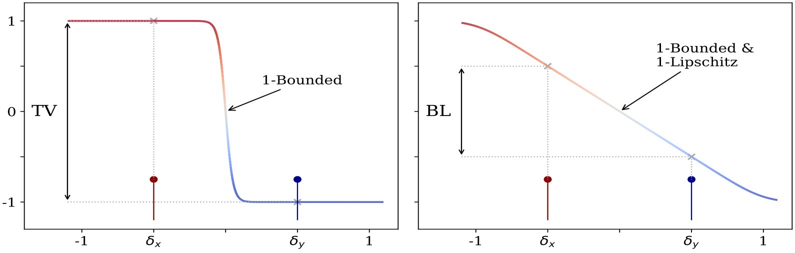


Goals:

1. When is the dissimilarity *perfectly discriminative*? (i.e. $D(P, Q) = 0 \Rightarrow P = Q$)
2. When does it metrize *weak convergence*?

What is the right capacity?

- Too small capacity may miss relevant information
- But too much capacity can also hurt:
  - classifier gets more difficult to learn
  - dissimilarity can saturate on discrete measures (e.g. samples)



Goals:

1. When is the dissimilarity *perfectly discriminative*? (i.e. $D(P, Q) = 0 \Rightarrow P = Q$)
2. When does it metrize *weak convergence*?
3. Real-world examples with too high capacity?

Outline

1. Introduction: classifier-based distribution-dissimilarities

2. Maximum Mean Discrepancies (MMD)

3. Adversarial Vulnerability of Neural Networks

Joint work with

Bernhard Schölkopf

Lester Mackey

Alessandro Barp

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity
- IPM functional is *linear* in φ

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad(\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity
- IPM functional is *linear* in φ

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $(\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity
- IPM functional is *linear* in $\varphi \Rightarrow$ can use *duality*!

Why MMDs and Goals

Goal: (for MMDs)

1. When is the dissimilarity perfectly discriminative?
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity
- IPM functional is *linear* in $\varphi \Rightarrow$ can use *duality*!

What distrib. can be separated? $\overset{\text{duality}}{\Longleftrightarrow}$ What fcts can $\mathcal{F}$ approximate?

Why MMDs and Goals

Goal: (for MMDs)

1. **When is the dissimilarity perfectly discriminative?**
2. When does it metrize weak convergence?

MMD := IPM with $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ $\quad (\mathrm{MMD}_k(P, Q) := \sup_{\varphi \in \mathcal{B}(\mathcal{H}_k)} \mathbb{E}_P[\varphi] - \mathbb{E}_Q[\varphi])$

Why MMD?

- dissimilarity (semi-) *metric*
- common in ML & computable on samples $\rightsquigarrow$ P. Alquier & J. Mairal!
- changing k changes $\mathcal{F}$, changes the capacity
- IPM functional is *linear* in φ $\Rightarrow$ can use *duality*!

What distrib. can be separated? $\overset{\text{duality}}{\Longleftrightarrow}$ What fcts can $\mathcal{F}$ approximate?

Goal 1: Perfect discrimination

$\mathcal{H}_k$ dense	MMD_k perf. discr.	Usual Name
$\mathscr{F}$	$\mathscr{F}'$	
$\mathscr{C}_0$	$\mathcal{M}_f$	
$\mathscr{C}$	$\mathcal{M}_c$	
$\mathrm{L}^p(\mu)$	$\mathrm{L}^q(\mu)$	
$\mathbb{C}^{\mathcal{X}}$	$\mathcal{M}_\delta$	

Goal 1: Perfect discrimination

$\mathcal{H}_k$ dense	MMD_k perf. discr.	Usual Name
$\mathscr{F}$	$\mathscr{F}'$	
$\mathscr{C}_0$	$\mathcal{M}_f$	
$\mathscr{C}$	$\mathcal{M}_c$	
$\mathrm{L}^p(\mu)$	$\mathrm{L}^q(\mu)$	
$\mathbb{C}^{\mathcal{X}}$	$\mathcal{M}_\delta$	
$((\mathscr{C}_b)_c)/\mathbb{1}$	$\mathscr{P}$ (or $\mathcal{M}_f^0$)	
$\mathbb{C}^{\mathcal{X}}/\mathbb{1}$	$\mathscr{P}_\delta$(or $\mathcal{M}_\delta^0$)	

Goal 1: Perfect discrimination

$\mathcal{H}_k$ dense	MMD_k perf. discr.	Usual Name
$\mathcal{F}$	$\mathcal{F}'$	
$\mathcal{C}_0$	$\mathcal{M}_f$	
$\mathcal{C}$	$\mathcal{M}_c$	
$\mathrm{L}^p(\mu)$	$\mathrm{L}^q(\mu)$	
$\mathbb{C}^{\mathcal{X}}$	$\mathcal{M}_\delta$	
$((\mathcal{C}_b)_c)/\mathbb{1}$	$\mathcal{P}$ (or $\mathcal{M}_f^0$)	
$\mathbb{C}^{\mathcal{X}}/\mathbb{1}$	$\mathcal{P}_\delta$(or $\mathcal{M}_\delta^0$)	

Theorem (Answer to 1: Perfect discrimination [SS18])

If $\mathcal{H}_k \hookrightarrow \mathcal{F}$, the following is equivalent:

(i) *$\mathcal{H}_k$ is dense in $\mathcal{F}$.*

(ii) *MMD_k is perf. discr. over $\mathcal{M} := \mathcal{F}'$.*

Goal 1: Perfect discrimination

$\mathcal{H}_k$ dense	MMD_k perf. discr.	Usual Name
$\mathcal{F}$	$\mathcal{F}'$	
$\mathscr{C}_0$	$\mathcal{M}_f$	c_0-universal
$\mathscr{C}$	$\mathcal{M}_c$	c-universal (or cc-universal)
$\mathrm{L}^p(\mu)$	$\mathrm{L}^q(\mu)$	
$\mathbb{C}^{\mathcal{X}}$	$\mathcal{M}_\delta$	
$((\mathscr{C}_b)_c)/\mathbb{1}$	$\mathscr{P}$ (or $\mathcal{M}_f^0$)	characteristic
$\mathbb{C}^{\mathcal{X}}/\mathbb{1}$	$\mathscr{P}_\delta$(or $\mathcal{M}_\delta^0$)	

Theorem (Answer to 1: Perfect discrimination [SS18])

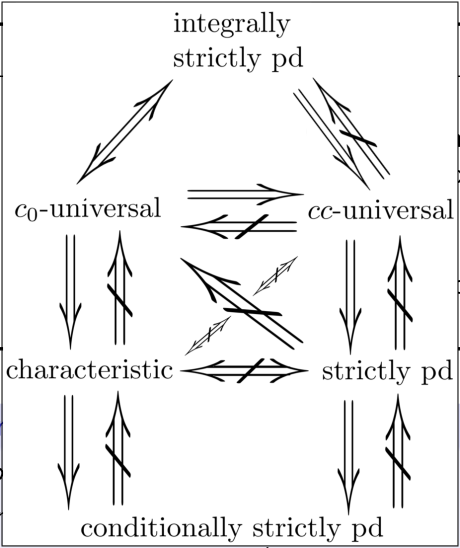
If $\mathcal{H}_k \hookrightarrow \mathcal{F}$, the following is equivalent:

(i) *$\mathcal{H}_k$ is dense in $\mathcal{F}$.*

(ii) *MMD_k is perf. discr. over $\mathcal{M} := \mathcal{F}'$.*

Goal 1: Perfect discrimination

$\mathcal{H}_k$ dense	Name
$\mathcal{F}$	
$\mathscr{C}_0$	iversal
$\mathscr{C}$	or *cc*-universal)
$\mathrm{L}^p(\mu)$	
$\mathbb{C}^{\mathcal{X}}$	
$((\mathscr{C}_b)_c)/\mathbb{1}$	cteristic
$\mathbb{C}^{\mathcal{X}}/\mathbb{1}$	



Based on Fig.1.1 [SFL11]

Theorem (Answer

If $\mathcal{H}_k \hookrightarrow \mathcal{F}$, the fo

(i) *$\mathcal{H}_k$ is dense i*

(ii) *MMD_k is perf. discr. over $\mathcal{M} := \mathcal{F}'$*

When does MMD metrize weak-convergence?

Theorem ( )

Let $k \in \mathscr{C}_b$ defined on a locally compact input space.
Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

When does MMD metrize weak-convergence?

Theorem ( )

Let $k \in \mathscr{C}_b$ defined on a locally compact input space. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

[CO18]: False on Polish spaces

When does MMD metrize weak-convergence?

Theorem (Compact case)

Let $k \in \mathscr{C}_b$ defined on a locally compact input space.
Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

[CO18]: False on Polish spaces

[SBM20]: False on locally compact spaces

When does MMD metrize weak-convergence?

Theorem (Compact case)

Let $k \in \mathscr{C}_b$ defined on a ~~locally~~ compact input space. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

[CO18]: False on Polish spaces

[SBM20]: False on locally compact spaces

When does MMD metrize weak-convergence?

Theorem (Compact case)

Let $k \in \mathscr{C}_b$ defined on a ~~locally~~ compact input space. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

Theorem (Non-compact case [SBM20])

Let $k \in \mathscr{C}_b$ defined on loc. comp. non-compact space s.t. $\mathscr{H}_k \subset \mathscr{C}_0$. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathcal{M}_f$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$

When does MMD metrize weak-convergence?

Theorem (Compact case)

Let $k \in \mathscr{C}_b$ defined on a ~~locally~~ compact input space. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

Theorem (Non-compact case [SBM20])

Let $k \in \mathscr{C}_b$ defined on loc. comp. non-compact space s.t. $\mathscr{H}_k \subset \mathscr{C}_0$. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathcal{M}_f$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$

- With $\mathcal{M} = \mathscr{P}$: mass can diffuse to infinity $\rightsquigarrow$ Kernel Stein Discrepancies

When does MMD metrize weak-convergence?

Theorem (Compact case)

Let $k \in \mathscr{C}_b$ defined on a ~~locally~~ compact input space. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathscr{P}$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$.

Theorem (Non-compact case [SBM20])

Let $k \in \mathscr{C}_b$ defined on loc. comp. non-compact space s.t. $\mathscr{H}_k \subset \mathscr{C}_0$. Then the following is equivalent.

(i) MMD_k *is perfectly discriminative over* $\mathcal{M}_f$.

(ii) MMD_k *metrizes weak convergence on* $\mathscr{P}$

- With $\mathcal{M} = \mathscr{P}$: mass can diffuse to infinity $\rightsquigarrow$ Kernel Stein Discrepancies
- If $\mathscr{H}_k \not\subset \mathscr{C}_0$, anything can happen: (i) without (ii) and (ii) without (i)

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Are these properties important in practice?

- typically give consistency guarantees, but this ignores sample-size
- non-consistent algos may have better approx./sample-size trade-off

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Are these properties important in practice?

- typically give consistency guarantees, but this ignores sample-size
- non-consistent algos may have better approx./sample-size trade-off

Illustration with generative models

- Idea: Generator $G_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$ generates sample from P_G.
  Parameters θ optimized to minimize $D_{\mathcal{F}}(\hat{P}_G, \hat{Q})$.

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Are these properties important in practice?

- typically give consistency guarantees, but this ignores sample-size
- non-consistent algos may have better approx./sample-size trade-off

Illustration with generative models

- Idea: Generator $G_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$ generates sample from P_G.
  Parameters θ optimized to minimize $D_{\mathcal{F}}(\hat{P}_G, \hat{Q})$.
- $\mathcal{F}$ = neural network $\rightsquigarrow$ GAN: not consistent but better
  $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ with Gauss kernel $\rightsquigarrow$ MMN: consistent but worse

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Are these properties important in practice?

- typically give consistency guarantees, but this ignores sample-size
- non-consistent algos may have better approx./sample-size trade-off

Illustration with generative models

- Idea: Generator $G_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$ generates sample from P_G.
  Parameters θ optimized to minimize $D_{\mathcal{F}}(\hat{P}_G, \hat{Q})$.
- $\mathcal{F} =$ neural network $\rightsquigarrow$ GAN: not consistent but better
  $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ with Gauss kernel $\rightsquigarrow$ MMN: consistent but worse

Other point of views:

- use other notions of capacity (VC dim, Rademacher complexity, ...)

Conclusion on MMDs

For MMDs, we characterized

1. perfect discrimination
2. weak-convergence metrization

Are these properties important in practice?

- typically give consistency guarantees, but this ignores sample-size
- non-consistent algos may have better approx./sample-size trade-off

Illustration with generative models

- Idea: Generator $G_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$ generates sample from P_G.
  Parameters θ optimized to minimize $D_{\mathcal{F}}(\hat{P}_G, \hat{Q})$.
- $\mathcal{F} =$ neural network $\rightsquigarrow$ GAN: not consistent but better
  $\mathcal{F} = \mathcal{B}(\mathcal{H}_k)$ with Gauss kernel $\rightsquigarrow$ MMN: consistent but worse

Other point of views:

- use other notions of capacity (VC dim, Rademacher complexity, ...)
- what differences/invariances are classifiers sensitive to?

Outline

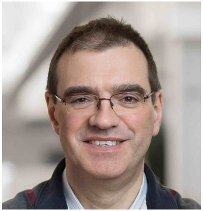
1. Introduction: classifier-based distribution-dissimilarities

2. Maximum Mean Discrepancies (MMD)

3. Adversarial Vulnerability of Neural Networks

Joint work with:

Yann Ollivier



Léon Bottou

Bernhard Schölkopf

David Lopez-Paz

Adversarial Examples

- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier

Adversarial Examples

- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier

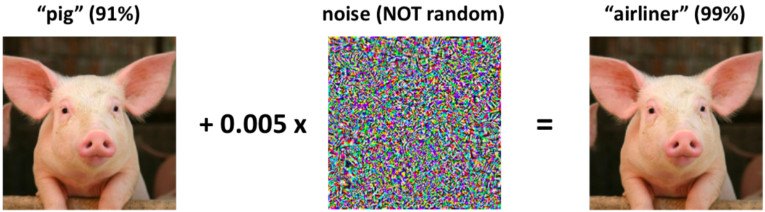


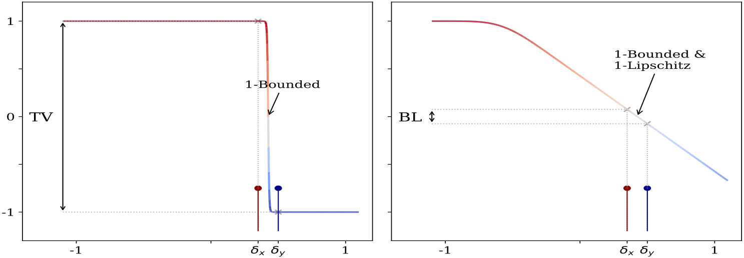
Figure: Madry, NIPS 2018, Adversarial Robustness Workshop

Adversarial Examples

- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier
- Reveals discrepancy btw. classifier's and perceptual dissimilarity

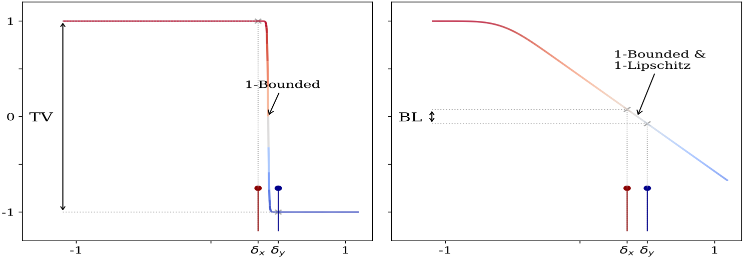
Adversarial Examples

- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier
- Reveals discrepancy btw. classifier's and perceptual dissimilarity



Adversarial Examples

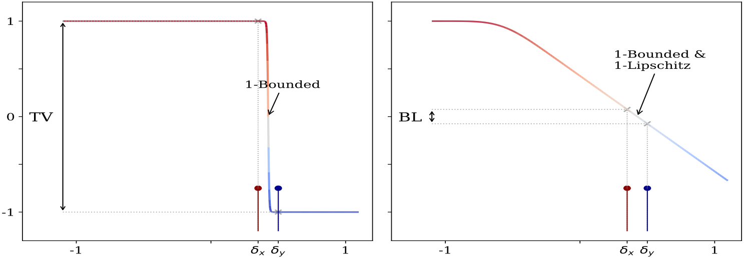
- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier
- Reveals discrepancy btw. classifier's and perceptual dissimilarity



For a link btw adv. error and optimal transport: see [PJ20]

Adversarial Examples

- Adversarial Example:
  small input-perturbation that yields large output-variation of classifier
- Reveals discrepancy btw. classifier's and perceptual dissimilarity



For a link btw adv. error and optimal transport: see [PJ20]

Goal

Understand why neural networks are adversarially vulnerable.
Can we quantify & predict this vulnerability?

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} \ = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \operatorname{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \operatorname{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} =$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \delta)}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} +$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \dots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \dots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \dots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \sum_{i=1}^{d} \epsilon_i \delta_i$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \text{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \sum_{i=1}^d \underbrace{\epsilon_i \delta_i}_{\pm|\epsilon||\delta|}$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \underbrace{\sum_{i=1}^d \underbrace{\epsilon_i \delta_i}_{\pm|\epsilon||\delta|}}_{\text{cancels out by CLT } \propto \sqrt{d}|\epsilon||\delta|}$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \underbrace{\sum_{i=1}^d \underbrace{\epsilon_i \delta_i}_{+|\epsilon||\delta|}}_{\text{cancels out by CLT} \propto \sqrt{d}|\epsilon||\delta|}$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \dots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \dots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \dots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \underbrace{\sum_{i=1}^d \underbrace{\epsilon_i \delta_i}_{+|\epsilon||\delta|}}_{\substack{\cancel{\text{cancels out by CLT} \propto \sqrt{d}|\epsilon||\delta|} \\ \text{everything adds up! } \propto d|\epsilon||\delta|}}$$

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \underbrace{\sum_{i=1}^d \underbrace{\epsilon_i\delta_i}_{+|\epsilon||\delta|}}_{\substack{\cancel{\text{cancels out by CLT } \propto \sqrt{d}|\epsilon||\delta|} \\ \text{everything adds up! } \propto d|\epsilon||\delta|}}$$

High accuracy & Random perturbation robustness $\not\Rightarrow$ Adv. robustness

Adversarial Vulnerability: Introductory Example

- Binary linear classification: $y = \mathrm{sign}(x_1)$ with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$.
- Learned with linear classifier: $\hat{y} = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x})$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$.
- Optimal classifier: $\boldsymbol{w}^* = (1, 0, 0, \ldots, 0)$
  In practice: $\boldsymbol{w} = (1 + \epsilon_1, \epsilon_2, \ldots, \epsilon_d)$
- Does not matter for average sample/perturbation:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x} + \boldsymbol{\delta})}_{\text{perturbed output}} = \underbrace{\boldsymbol{w}^T \boldsymbol{x}}_{\text{unperturbed output}} + \delta_1 + \underbrace{\sum_{i=1}^d \underbrace{\epsilon_i \delta_i}_{+|\epsilon||\delta|}}_{\substack{\cancel{\text{cancels out by CLT } \propto \sqrt{d}|\epsilon||\delta|} \\ \text{everything adds up! } \propto d|\epsilon||\delta|}}$$

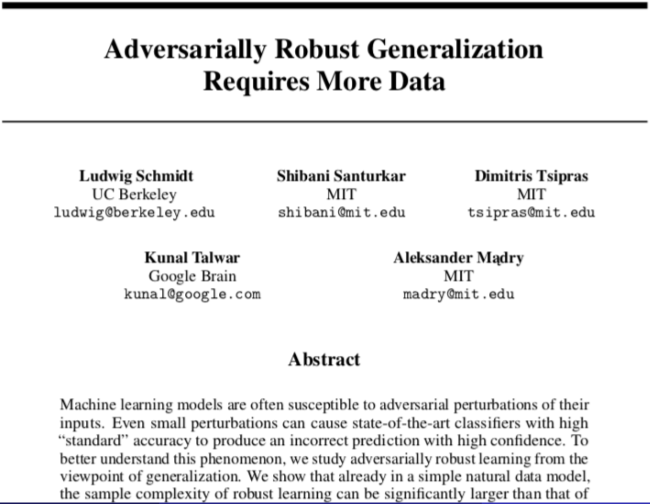
High accuracy & Random perturbation robustness $\not\Rightarrow$ Adv. robustness
Adversarially vulnerability increases with dimension

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?


Adversarially Robust Generalization Requires More Data

Ludwig Schmidt
UC Berkeley
ludwig@berkeley.edu

Shibani Santurkar
MIT
shibani@mit.edu

Dimitris Tsipras
MIT
tsipras@mit.edu

Kunal Talwar
Google Brain
kunal@google.com

Aleksander Mądry
MIT
madry@mit.edu

Abstract

Machine learning models are often susceptible to adversarial perturbations of their inputs. Even small perturbations can cause state-of-the-art classifiers with high "standard" accuracy to produce an incorrect prediction with high confidence. To better understand this phenomenon, we study adversarially robust learning from the viewpoint of generalization. We show that already in a simple natural data model, the sample complexity of robust learning can be significantly larger than that of

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity
  but incorporate better data-assumptions

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity
  but incorporate better data-assumptions

Illustration on previous example
Favor models that use only few input-dims by:

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity
  but incorporate better data-assumptions

Illustration on previous example
Favor models that use only few input-dims by:
- hard-coding it in model architecture

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity
  but incorporate better data-assumptions

  Illustration on previous example
  Favor models that use only few input-dims by:
    - hard-coding it in model architecture
    - or using a sparsifying regularizer (LASSO)

Adversarial robustness needs good model assumptions

What to do against adversarial vulnerability?

- 1st remedy idea: use more data $\rightarrow$ universal answer?
- 2nd remedy idea:
  reduce models' capacity
  but incorporate better data-assumptions

  Illustration on previous example
  Favor models that use only few input-dims by:
  - hard-coding it in model architecture
  - or using a sparsifying regularizer (LASSO)

1. Without data-assumptions, adv. robustness can be hard to get.
2. With structured data, model assumptions can alleviate vulnerability.

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

- But images do not satisfy their assumptions

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)
- But images do not satisfy their assumptions
  no full support

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]

  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

- But images do not satisfy their assumptions

  no full support
  densities peaked in high dim (= not bounded)

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

- But images do not satisfy their assumptions
  no full support
  densities peaked in high dim (= not bounded)
  higher resolution should help, not hurt

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]
  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

- But images do not satisfy their assumptions
  no full support
  densities peaked in high dim (= not bounded)
  higher resolution should help, not hurt

Adv. vul. questions what is wrong with our classifiers, not our data.

For images, higher resolutions should help, not hurt

- Many no-free-lunch type of results [GMFS+18; SSTT+18; SHSF+19]

  assume uniformity of distribution in input-dim
  (concentric spheres, bounded densities with full support)

- But images do not satisfy their assumptions

  no full support
  densities peaked in high dim (= not bounded)
  higher resolution should help, not hurt

Adv. vul. questions what is wrong with our classifiers, not our data.

Question

What properties of neural nets are not enough adapted to data?

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- ADV.DAM.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- ADV.DAM.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-$\textsc{Attack}$: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- $\textsc{Adv.Dam.}$: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

$$\mathrm{Adv\,Dam}_{\epsilon,\|\cdot\|} := \mathbb{E}_{\boldsymbol{x}\sim P}\left[\sup_{\|\boldsymbol{\delta}\|\leq\epsilon} \mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}) - \mathcal{L}(\boldsymbol{x})\right]$$

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x}+\boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- ADV.DAM.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

$$\begin{aligned}
\mathrm{Adv\,Dam}_{\epsilon,\|\cdot\|} &:= \mathbb{E}_{\boldsymbol{x}\sim P}\left[\sup_{\|\boldsymbol{\delta}\|\leq\epsilon} \mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}) - \mathcal{L}(\boldsymbol{x})\right] \\
&\approx \mathbb{E}_{\boldsymbol{x}\sim P}\left[\sup_{\|\boldsymbol{\delta}\|\leq\epsilon} \partial_{\boldsymbol{x}}\mathcal{L}\cdot\boldsymbol{\delta}\right]
\end{aligned}$$

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-Attack: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- Adv.Dam.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

$$\begin{aligned}
\operatorname{Adv Dam}_{\epsilon, \|\cdot\|} &:= \mathbb{E}_{\boldsymbol{x} \sim P} \left[\sup_{\|\boldsymbol{\delta}\| \leq \epsilon} \mathcal{L}(\boldsymbol{x} + \boldsymbol{\delta}) - \mathcal{L}(\boldsymbol{x}) \right] \\
&\approx \mathbb{E}_{\boldsymbol{x} \sim P} \left[\sup_{\|\boldsymbol{\delta}\| \leq \epsilon} \boldsymbol{\partial}_{\boldsymbol{x}} \mathcal{L} \cdot \boldsymbol{\delta} \right] \\
&= \epsilon \mathbb{E}_{\boldsymbol{x} \sim P} \left[\||\boldsymbol{\partial}_{\boldsymbol{x}} \mathcal{L}\|| \right]
\end{aligned}$$

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- ADV.DAM.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

$$\mathrm{Adv\,Dam}_{\epsilon,\|\cdot\|} \;\approx\; \epsilon \mathbb{E}_{\boldsymbol{x}}\left[|||\boldsymbol{\partial}_{\boldsymbol{x}} \mathcal{L}||| \right]$$

Adversarial Damage and Gradients

Definition (Adversarial Damage)

- $\epsilon, \|\cdot\|$-ATTACK: perturbed sample $\boldsymbol{x} + \boldsymbol{\delta}$ s.t. $\|\boldsymbol{\delta}\| \leq \epsilon$.
- ADV.DAM.: Expected maximal loss-increase after ϵ-sized $\|\cdot\|$-attacks

$$\mathrm{Adv\,Dam}_{\epsilon,\|\cdot\|} \;\approx\; \epsilon \mathbb{E}_{\boldsymbol{x}} \left[|||\boldsymbol{\partial}_{\boldsymbol{x}} \mathcal{L}||| \right]$$

Assuming that the Taylor-expansion is legit, question:

- How big is $\mathbb{E}_{\boldsymbol{x}} \left[|||\boldsymbol{\partial}_{\boldsymbol{x}} \mathcal{L}||| \right]$ in practice?

Size of $\mathbb{E}_{\boldsymbol{x}}\left[\|\partial_{\boldsymbol{x}}\mathcal{L}\|\right]$ for at initialization

Back to linear layer:

$$\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}} = \underbrace{\sum_{i=1}^{d} \underbrace{w_i\delta_i}_{+|w_i||\delta|}}_{\text{everything adds up! } \propto d|w||\delta|}$$

Size of $\mathbb{E}_x [|||\partial_x \mathcal{L}|||]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^d \underbrace{w_i \delta_i}_{+|w_i||\delta|}}_{\text{everything adds up! } \propto d|w||\delta|}$$

Size of $\mathbb{E}_{\boldsymbol{x}}\left[|||\partial_x \mathcal{L}|||\right]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^{d} \underbrace{w_i\delta_i}_{+|w_i||\delta| \ {\color{red}\approx |\delta|/\sqrt{d}}}}_{\text{everything adds up! } \propto d|w||\delta|}$$

Size of $\mathbb{E}_{\boldsymbol{x}}\left[|||\partial_x \mathcal{L}|||\right]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^{d} \underbrace{w_i \delta_i}_{+|w_i||\delta| \ {\color{red}\approx |\delta|/\sqrt{d}}}}_{\text{everything adds up! } \propto d|w||\delta| \ {\color{red}\approx \sqrt{d}|\delta|}}$$

Size of $\mathbb{E}_x [|||\partial_x \mathcal{L}|||]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^d \underbrace{w_i \delta_i}_{+|w_i||\delta| \;\approx |\delta|/\sqrt{d}}}_{\text{everything adds up! } \propto d|w||\delta| \;\approx \sqrt{d}|\delta|}$$

Generalization:

Size of $\mathbb{E}_{\boldsymbol{x}} [|||\boldsymbol{\partial}_x \mathcal{L}|||]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^d \underbrace{w_i\delta_i}_{+|w_i||\delta| \ \approx |\delta|/\sqrt{d}}}_{\text{everything adds up! } \propto d|w||\delta| \ \approx \sqrt{d}|\delta|}$$

Generalization:

Theorem (Gradient norms of NNs at initialization [SOBS+19])

At (He-)initialization, the adversarial damage of almost any usual feedforward network grows with the input-dimension d as

$$\mathrm{Adv\,Dam}_{\epsilon, \|\cdot\|_p} \approx \epsilon_p \|\boldsymbol{\partial}_x \mathcal{L}\|_q \propto \sqrt{d}$$

Size of $\mathbb{E}_{\boldsymbol{x}}\left[|||\boldsymbol{\partial}_x\mathcal{L}|||\right]$ for at initialization

Back to linear layer:

$$\underbrace{\underbrace{\boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta})}_{\text{perturbed output}} - \underbrace{\boldsymbol{w}^T\boldsymbol{x}}_{\text{unperturbed output}}}_{\text{adversarial damage at input } \boldsymbol{x}} = \underbrace{\sum_{i=1}^{d} \underbrace{w_i\delta_i}_{+|w_i||\delta|\ \approx|\delta|/\sqrt{d}}}_{\text{everything adds up! } \propto d|w||\delta|\ \approx\sqrt{d}|\delta|}$$

Generalization:

Theorem (Gradient norms of NNs at initialization [SOBS+19])

At (He-)initialization, the adversarial damage of almost any usual feedforward network grows with the input-dimension d as

$$\mathrm{Adv\,Dam}_{\epsilon,\|\cdot\|_p} \approx \epsilon_p \left\|\boldsymbol{\partial}_x\mathcal{L}\right\|_q \propto \sqrt{d}$$

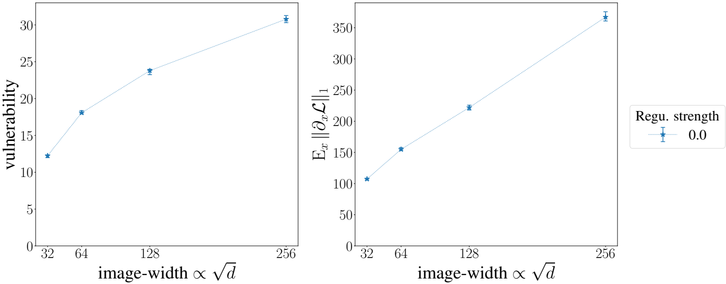
Dimension-dependence is independent of network topology.

Gradient norms and vulnerability after training

After usual training?

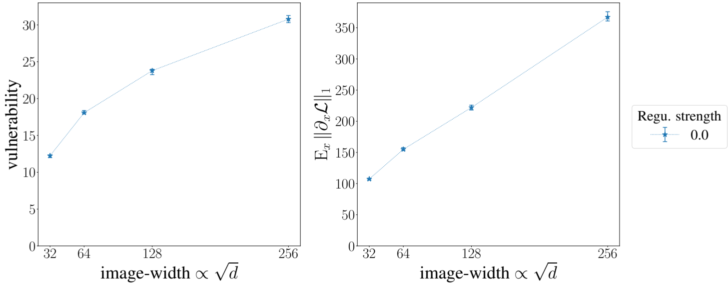
Gradient norms and vulnerability after training

After usual training?



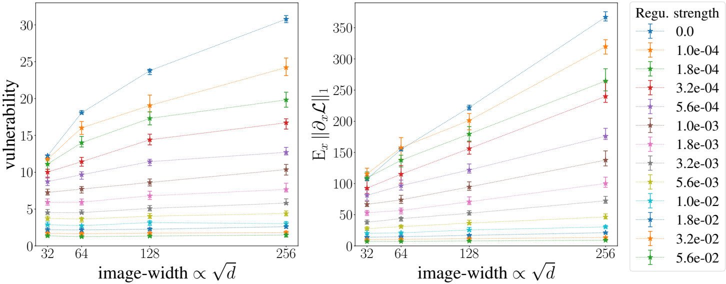
Gradient norms and vulnerability after training

After usual training? After robust training?



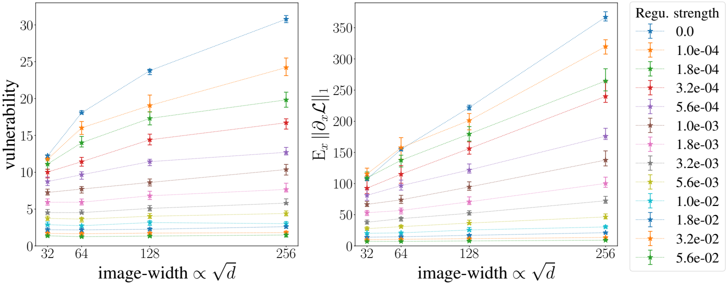
Gradient norms and vulnerability after training

After usual training? After robust training?



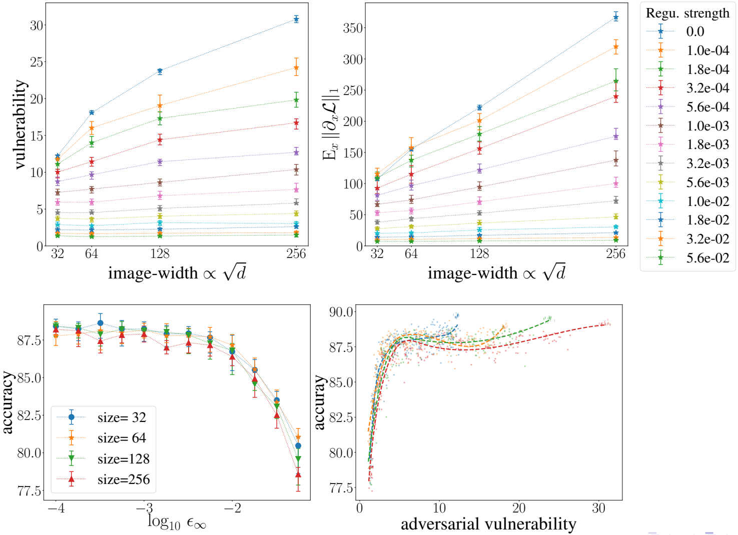
Gradient norms and vulnerability after training

After usual training? After robust training? Cost for accuracy?



Gradient norms and vulnerability after training

After usual training? After robust training? Cost for accuracy?



Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension

Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension
- Dimension-dependence persists after usual and robust training

Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension
- Dimension-dependence persists after usual and robust training

Current nets do not naturally incorporate all relevant data-structure.

Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension
- Dimension-dependence persists after usual and robust training

Current nets do not naturally incorporate all relevant data-structure.

Questons for future:

Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension
- Dimension-dependence persists after usual and robust training

Current nets do not naturally incorporate all relevant data-structure.

Questons for future:

- Why does robust training not remove dimension-depence: training algo or function class problem?

Conclusion on adversarial vulnerability

- Vulnerability at initialization of current nets increases with dimension
- Dimension-dependence persists after usual and robust training

Current nets do not naturally incorporate all relevant data-structure.

Questons for future:

- Why does robust training not remove dimension-depence: training algo or function class problem?
- Design networks that incorporate more data-assumptions

Summary

- Classifier-based distribution dissimilarities

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
    - perfect discrimination of distributions

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:
  - Focus on classifiers, not data

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:
  - Focus on classifiers, not data
  - Current NN inits cause vulnerability to increase with input dim

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:
  - Focus on classifiers, not data
  - Current NN inits cause vulnerability to increase with input dim
  - Suggests classifiers do not incorporate enough invariances of data

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:
  - Focus on classifiers, not data
  - Current NN inits cause vulnerability to increase with input dim
  - Suggests classifiers do not incorporate enough invariances of data

Summary

- Classifier-based distribution dissimilarities
- Influence of classifier capacity in MMDs on
  - perfect discrimination of distributions
  - metrization of weak convergence
  - Key insight: *duality*
    Unifies concepts of *SPD, characteristic, universal* kernels
- Adversarial vulnerability:
  - Focus on classifiers, not data
  - Current NN inits cause vulnerability to increase with input dim
  - Suggests classifiers do not incorporate enough invariances of data

Collaborators and audience: THANKS! QUESTIONS?

References I

I. Chevyrev and H. Oberhauser. "Signature moments to characterize laws of stochastic processes". In: *arXiv:1810.10971* (2018).

J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. "Adversarial Spheres". In: *ICLR Workshop*. 2018. arXiv: 1801.02774.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *ICLR*. 2019.

M. S. Pydi and V. Jog. "Adversarial Risk via Optimal Transport and Optimal Couplings". In: *ICML*. 2020.

References II

C.-J. Simon-Gabriel, A. Barp, and L. Mackey. "Metrizing Weak Convergence with Maximum Mean Discrepancies". In: *arXiv:2006.09268* (2020).

B. K. Sriperumbudur, K. Fukumizu, and G. R. Lanckriet. "Universality, characteristic kernels and RKHS embedding of measures". In: *JMLR* (2011).

A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. "Are adversarial examples inevitable?" In: *ICLR*. 2019.

C.-J. Simon-Gabriel, Y. Ollivier, L. Bottou, B. Schölkopf, and D. Lopez-Paz. "First-Order Adversarial Vulnerability of Neural Networks and Input Dimension". In: *ICML*. 2019.

References III

C.-J. Simon-Gabriel and B. Schölkopf. "Kernel distribution embeddings: Universal kernels, characteristic kernels and kernel metrics on distributions". In: *JMLR* (2018).

L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mkadry. *Adversarially Robust Generalization Requires More Data*. arXiv:1804.11285. 2018.