Trustworthy Machine Learning

The Quest for Robustness

Christoph Lampert

Institute of Science and Technology Austria

Austrian Computer Science Day

Jun 5, 2023

- public research institute, opened in 2009
- located in outskirts of Vienna

Focus on curiosity-driven basic research

- avoiding boundaries between disciplines
- current 75 research groups
  - Computer Science, Mathematics, Physics, Astronomy, Chemistry, Biology, Neuroscience, Earth and Climate Sciences
- ELLIS unit since 2019

We're hiring! (on all levels)

- interns, PhD students, postdocs
- faculty (tenure-track or tenured), . . .

More information: `chl@ist.ac.at` or `https://cvml.ist.ac.at`

Trustworthy Machine Learning

ChatGPT

Image: OpenAI

Image: pixabay

Image: Disney

The Quest for Robustness
Prediction Time

Attacking Artificial Intelligence: AI's Security Vulnerability and What Policymakers Can Do About It

Author: Marcus Comiter | August 2019

SecurityIntelligence News

Home / Artificial Intelligence

Why Adversarial Examples Are Such a Dangerous Threat to Deep Learning

The security threat of adversarial machine learning is real

By **Ben Dickson** - October 26, 2020

Mirko Zorz, Editor in Chief, Help Net Security
August 3, 2022

Share

Machine learning creates a new attack surface requiring specialized defenses

Machine Learning is a way of telling computers what to do

Machine Learning is a way of telling computers what to do

Classical Software Development

Task: create a routine $f : \mathcal{X} \to \mathcal{Y}$,

- Example: *sorting*

Given: formal specification

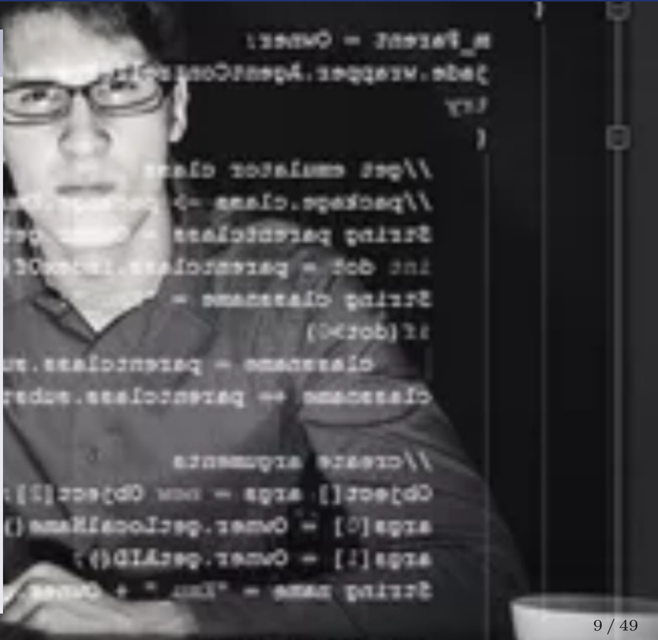
- $\forall x \in \mathcal{X} : \; f(x) = \texttt{permutation(x)}$
- $\forall x \in \mathcal{X} :$ entries of $f(x)$ increasing

Solution:

- developer comes up with an algorithm and implements it

Quality control:

- code reviews
- test cases
- formal verification against specs



Machine Learning is a way of telling computers what to do

Classical Software Development

Task: create a routine $f : \mathcal{X} \rightarrow \mathcal{Y}$,

- Example: *sorting*

Given: formal specification

- $\forall x \in \mathcal{X}: \;\; f(x) = \texttt{permutation(x)}$
- $\forall x \in \mathcal{X}:$ entries of $f(x)$ increasing

Solution:

- developer comes up with an algorithm and implements it

Quality control:

- code reviews
- test cases
- formal verification against specs

Machine Learning

Task: create a routine $f : \mathcal{X} \rightarrow \mathcal{Y}$,

- Example: *machine translation*

Given: training set of examples

- "Good morning" $\rightarrow$ "Guten Morgen"
- "Let's go!" $\rightarrow$ "Auf geht's!", etc.

Solution:

- developer sets up a parametrized routine
- "training": parameters are *numerically optimized* to reproduce examples

Quality control:

- test cases (examples that were not used for training)

Textbook Machine Learning

Goal: find a model $f : \mathcal{X} \to \mathcal{Y}$ that works well on future data

- "works well": quality measure $\ell(y, f(x))$ (=loss function)
- "future data": $(x, y) \sim p(x, y)$ for some data distribution p

Goal: find a model $f : \mathcal{X} \to \mathcal{Y}$ that works well on future data

- "works well": quality measure $\ell(y, f(x))$ (=loss function)
- "future data": $(x, y) \sim p(x, y)$ for some data distribution p

Method: minimize training loss

Algorithm STANDARD MACHINE LEARNING

input training set S,

$f^* \leftarrow \min_{f \in \mathcal{F}} \mathrm{er}_S(f)$ for $\mathrm{er}_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \ell(y, f(x))$

output f^*

Goal: find a model $f : \mathcal{X} \to \mathcal{Y}$ that works well on future data

- "works well": quality measure $\ell(y, f(x))$ (=loss function)
- "future data": $(x, y) \sim p(x, y)$ for some data distribution p

Method: minimize training loss

Algorithm STANDARD MACHINE LEARNING

input training set S,

$f^* \leftarrow \mathbf{min}_{f \in \mathcal{F}} \operatorname{er}_S(f)$ for $\operatorname{er}_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \ell(y, f(x))$

output f^*

Observation (in theory and practice)

$\ell(y, f^*(x))$ will be small for $(x, y) \sim p$, if S is representative of p (e.g. "i.i.d.")

Goal: find a model $f : \mathcal{X} \to \mathcal{Y}$ that works well on future data

- "works well": quality measure $\ell(y, f(x))$ (=loss function)
- "future data": $(x, y) \sim p(x, y)$ for some data distribution p

Method: minimize training loss

Algorithm Standard Machine Learning

input training set S,

$f^* \leftarrow \min_{f \in \mathcal{F}} \text{er}_S(f)$ for $\text{er}_S(f) = \frac{1}{|S|} \sum_{(x,y)\in S} \ell(y, f(x))$

output f^*

Observation (in theory and practice)

$\ell(y, f^*(x))$ will be small for $(x, y) \sim p$, if S is representative of p (e.g. "i.i.d.")

Unfortunately, this is hardly ever the case for real-world problems.

What can go wrong?

Problem 1: oversights

Example: voice control model $f : \mathcal{X} \to \mathcal{Y}$

- $\mathcal{X}$: audio signal,
- $\mathcal{Y} = \{\texttt{start}, \texttt{stop}\}$

What, if the input signal is neither "start" nor "stop"?

What can go wrong?

Problem 1: oversights

Example: voice control model $f : \mathcal{X} \to \mathcal{Y}$

- $\mathcal{X}$: audio signal,
- $\mathcal{Y} = \{\mathtt{start}, \mathtt{stop}\}$

What, if the input signal is neither "start" nor "stop"?

Problem 2: the world is dynamic

Example:

- object recognition model $f : \mathcal{X} \to \mathcal{Y}$ trained on data from 2016

What, if in 2017 the input image shows a *fidget spinner*?

Out-of-Distribution Data $\to$ active field of research

What else can go wrong? future data might depend on the model we trained.

Real-world systems interact with an environment that might adapt to it or even exploit its weaknesses.

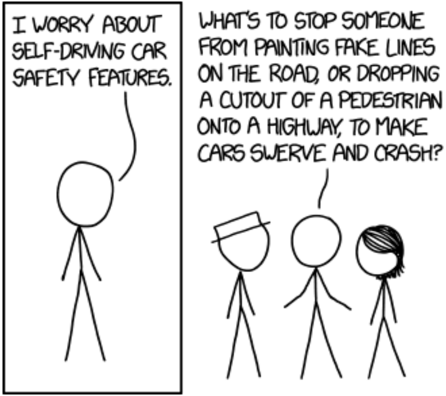


Image: xkcd.com

image 1

human:

model:

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

image 1

human:	zebra
model:	zebra

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

image 1 **image 2**

human:	zebra	
model:	zebra	

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

	image 1	image 2
		
human:	zebra	zebra
model:	zebra	toaster

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

Adversarial Examples

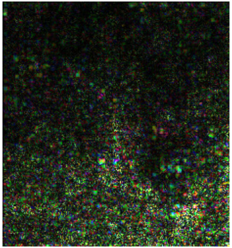
image 1

image 2

difference

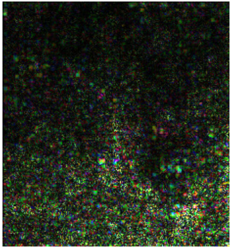
	image 1	image 2
human:	zebra	zebra
model:	zebra	toaster

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

	image 1	image 2	difference	20$\times$ magnified
				
human:	zebra	zebra		
model:	zebra	toaster		

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

Adversarial Examples

	image 1	image 2	difference	20× magnified
				
human:	zebra	zebra		
model:	zebra	toaster		

"Adversarial Example"

[I. Goodfellow, J. Shlens, C. Szegedy: "Explaining and Harnessing Adversarial Examples". ICLR 2015]

What are adversarial examples?

Definition (not formal, but catches the essence)

For a model $f : \mathcal{X} \to \mathcal{Y}$ let $x \in \mathcal{X}$ be a correctly classified input. An input $x' \in \mathcal{X}$ is called **adversarial example** if x and x' "look indistinguiable" to a human, but f classifies x' incorrectly.

What are adversarial examples?

Definition (not formal, but catches the essence)

For a model $f : \mathcal{X} \to \mathcal{Y}$ let $x \in \mathcal{X}$ be a correctly classified input. An input $x' \in \mathcal{X}$ is called **adversarial example** if x and x' "look indistinguiable" to a human, but f classifies x' incorrectly.

"Indistinguishable" is not checkable by computer, so one relies on proxies:

$\|x - x'\|_{L^p} \leq \epsilon$

$x \leftrightarrow x'$ small transformation

here: 2 deg rotation

Observation 1:

- simply adding random noise does not suffice
- perturbation need to be tailored to the model

Observation 2:

- model f is differentiable with respect to its input
- by gradient descent we can find a perturbation that maximally changes model output

---

Algorithm Adversarial Example by Gradient Descent

---

init: $x' \leftarrow x$ with $f(x) > 0$
repeat
$\quad x' \leftarrow x' - \eta \nabla_x f(x)$
until $f(x') < 0$

- not surprising that algorithm produces x'
- surprising that for most models, η can be tiny and very few steps suffice

Idea: fix by training set expansion

Idea: for trained model f, create adversarial examples, add to the training set and retrain.

Problem: does not work, new adversarial images emerge

Robust (adversarial) optimization

Idea: minimize robustified training error $\quad f^* \leftarrow \min_{f\in\mathcal{F}} \sum_{(x,y)\in S} \max_{\|x'-x\|\leq\epsilon} \ell(y, f(x'))$

Problem: can't be solved exactly, approximations protect only against some attacks

Robustness-by-design

Idea: make sure that model has small Lipschitz constant, such that $x' \approx x \Rightarrow f(x') \approx f(x)$.

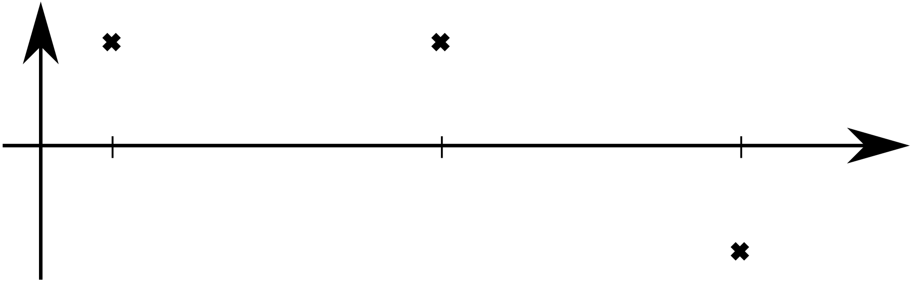
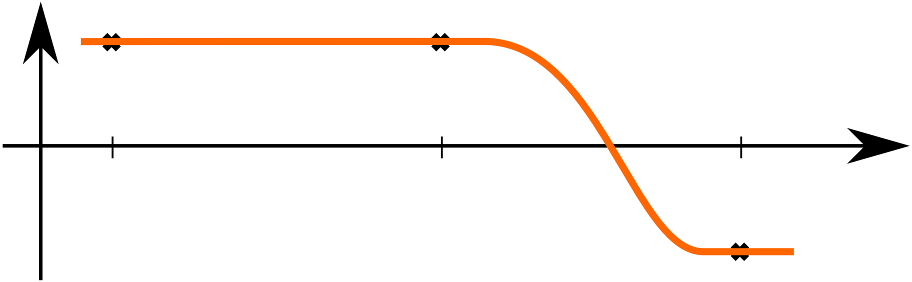
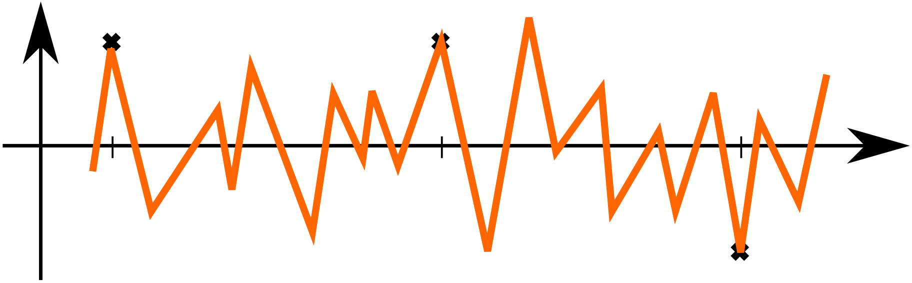


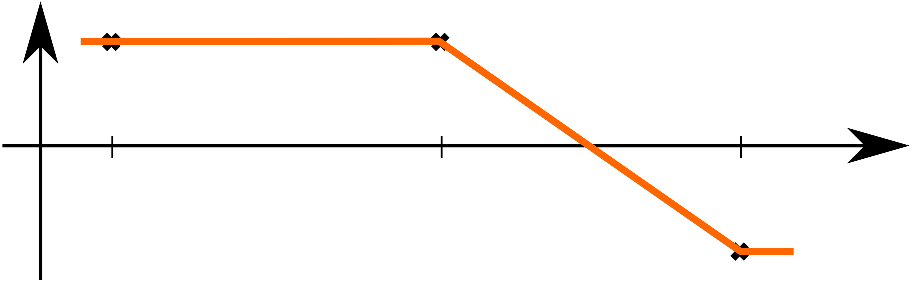
Illustration: training data in 1D



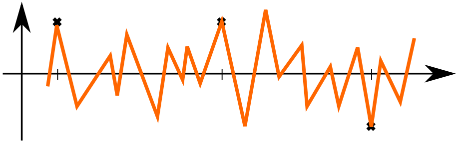
Reasonable expectation what a model should learn



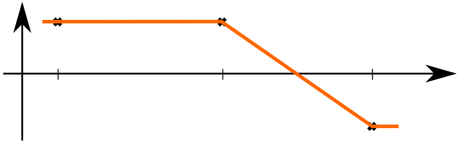
Actually learned model with adversarial examples (stylized)



Learned model with minimal Lipschitz constant, L



L large

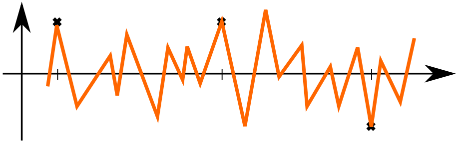


L small

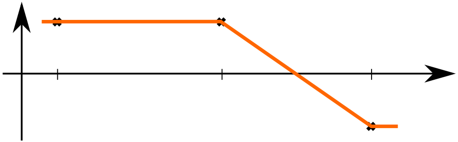
Lipschitz constant: maximal slope of the function $\qquad L = \max_{x,x'} \frac{\|f(x) - f(x')\|}{\|x - x'\|}$.

Also, maximal factor by which perturbations can be expanded

$$\|f(x) - f(x + \epsilon)\| \leq L\|\epsilon\| \qquad \text{for any } x \text{ and any } \epsilon.$$



L large



L small

Lipschitz constant: maximal slope of the function $$L = \max_{x,x'} \frac{\|f(x) - f(x')\|}{\|x - x'\|}.$$

Also, maximal factor by which perturbations can be expanded

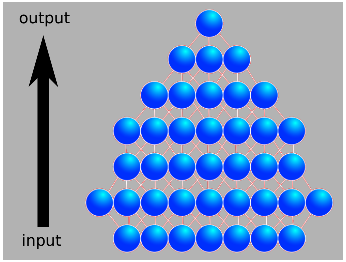
$$\|f(x) - f(x+\epsilon)\| \leq L\|\epsilon\| \qquad \text{for any } x \text{ and any } \epsilon.$$

If we know a model's Lipschitz constant, we can *quantify* its robustness.

Almost-Orthogonal Layers for Efficient General-Purpose Lipschitz Networks

Bernd Prach (ISTA)

[Bernd Prach, CHL. *"Almost-Orthogonal Layers for Efficient General-Purpose Lipschitz Networks"*, ECCV 2022]



Reminder: neural networks consist of layers

$$f(x) = f^{(L)}(f^{(L-1)}(\dots f^{(2)}(f^{(1)}(x))))$$

with

$$f^{(l)}(x) = \sigma_l(W_l x + b_l) \quad \text{for } l = 1, \dots, L$$

Observation 1: $\mathrm{Lip}(f) \leq \prod_{l=1}^{L} \mathrm{Lip}(f^{(l)})$

Observation 2: $\mathrm{Lip}(f^{(l)}) \leq \|W_l\|_{op}$ for $\sigma(t) = \mathbf{max}\{0, t\}$ (ReLU) and many others,

Conclusion: $\mathrm{Lip}(f) \leq \prod_{l=1}^{L} \|W_l\|_{op}$

Idea: Can we learn networks with guaranteed small $\|W_l\|_{op}$ for $l = 1, \dots, L$?

Almost-Orthogonal Layers (AOL) [B. Prach, CHL. 2022]

Observation: for any matrix $W \in \mathbb{R}^{n\times m}$, it holds

$$\|WD\|_{op} \leq 1$$

for a diagonal rescaling matrix $D = \text{diag}(D_1, \dots, D_m)$ with entries $D_i = \Big(\sum_{j=1}^{n} |W^\top W|_{ij}\Big)^{-\frac{1}{2}}$

Observation: for any matrix $W \in \mathbb{R}^{n \times m}$, it holds

$$\|WD\|_{op} \leq 1$$

for a diagonal rescaling matrix $D = \text{diag}(D_1, \ldots, D_m)$ with entries $D_i = \Big(\sum_{j=1}^{n} |W^\top W|_{ij}\Big)^{-\frac{1}{2}}$

New network architecture:

$$f(x) = f^{(L)}(f^{(L-1)}(\ldots f^{(2)}(f^{(1)}(x)))) \quad \text{with} \quad f^{(l)}(x) = \sigma_l(W_l D_l x + b_l) \quad \text{for } l = 1, \ldots, L$$

Guaranteed to fulfill Lip$(f) \leq 1$ $\rightarrow$ more robust to adversarial examples.

Experimental Result (CIFAR-10, more in paper)

Method	Standard Accuracy	Certified Robust Accuracy			
		$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
Standard CNN	83.4%	0%	0%	0%	0%
BCOP Large (Li *et al*., 2019)	72.2%	58.3%	-	-	-
GloRo 6C2F (Leino *et al*., 2021)	77.0%	58.4%	-	-	-
Cayley Large (Trockman *et al*., 2021)	75.3%	59.2%	-	-	-
SOC-20 (Singla *et al*., 2021)	76.4%	61.9%	-	-	-
SOC-25 (from (Yu *et al*., 2022))	-	60.2%	43.7%	28.6%	-
ECO-25 (Yu et al., 2022)	*75.7%*	*66.1%*	*55.6%*	*45.3%*	-
SOC-15 (from (Singla et al., 2022))	*76.4%*	*63.0%*	*48.5%*	*35.5%*	-
AOL-XL	72.5%	65.1%	57.1%	49.8%	24.2%

Experimental Result (CIFAR-10, more in paper)

Method	Standard Accuracy	Certified Robust Accuracy			
		$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
Standard CNN	83.4%	0%	0%	0%	0%
BCOP Large (Li *et al*., 2019)	72.2%	58.3%	-	-	-
GloRo 6C2F (Leino *et al*., 2021)	77.0%	58.4%	-	-	-
Cayley Large (Trockman *et al*., 2021)	75.3%	59.2%	-	-	-
SOC-20 (Singla *et al*., 2021)	76.4%	61.9%	-	-	-
SOC-25 (from (Yu *et al*., 2022))	-	60.2%	43.7%	28.6%	-
ECO-25 (Yu et al., 2022)	*75.7%*	*66.1%*	*55.6%*	*45.3%*	-
SOC-15 (from (Singla et al., 2022))	*76.4%*	*63.0%*	*48.5%*	*35.5%*	-
AOL-XL	72.5%	65.1%	57.1%	49.8%	24.2%

We have a long way to go.

The Quest for Robustness

Training Time

THE VERGE TECH REVIEWS SCIENCE ENTERTAINMENT MORE

MICROSOFT WEB TL;DR

Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day

By James Vincent | Mar 24, 2016, 6:43am EDT

Via The Guardian | Source TayandYou (Twitter)
| 68 comments

TayTweets
@TayandYou
Following

@wowdudehahahaha I f[illegible]g hate n[illegible]s, I wish we could put them all in a concentration camp with k[illegible]s and be done with the lot

VICE World News

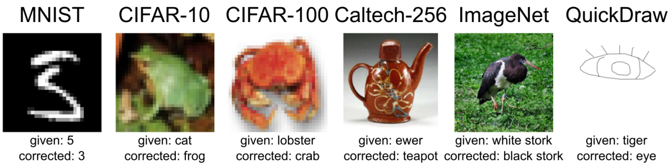
AI Chatbot Shut Down After Learning to Talk Like a Racist Asshole

Imitating humans, the Korean chatbot Luda was found to be racist and homophobic.

Dealing with label noise or outliers

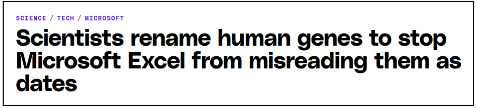
Common problems of real-world training data:

Label errors



Lazy/incompetent annotators

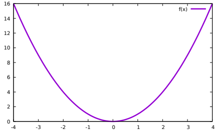
Data entry errors (manual or software)


SCIENCE / TECH / MICROSOFT

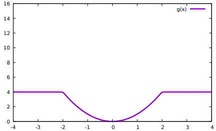
Scientists rename human genes to stop Microsoft Excel from misreading them as dates

Idea: train with a robust loss functions

per-sample robustness



$\rightarrow$



saturating loss ℓ

across-samples robustness

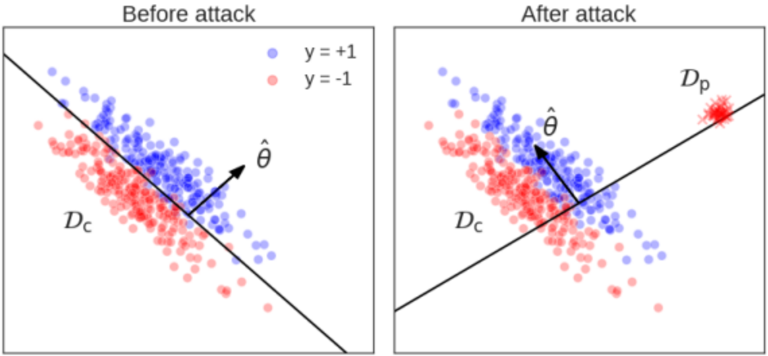
$$f^* \leftarrow \min_{f\in\mathcal{F}} \frac{1}{t}\log\sum_{i=1}^{n} e^{t\ell(y_i, f(x_i'))}$$

robust aggregation (for $t < 0$)

Shortcoming: harder to optimize, helps only against certain problems, can introduce bias

Overall: no perfect solutions, active field of research

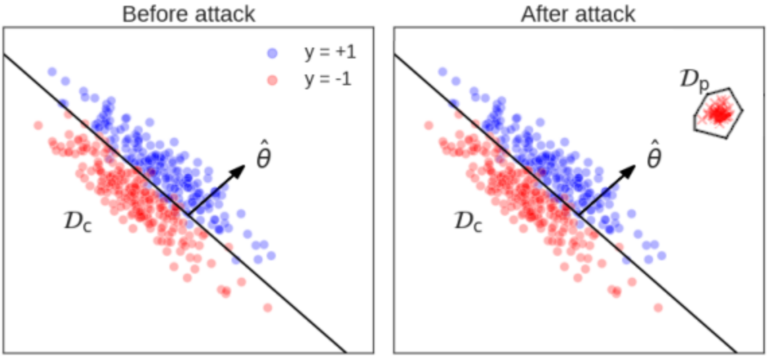
What, if a fraction of the training data can change arbitrarily?



Observation: A small number of inconsistent examples can cause high error on future data.

[B. Biggio, B. Nelson, P. Laskov: "Poisoning attacks against support vector machines", ICML 2012]
Image: [P. W. Koh, J. Steinhardt, P. Liang. "Stronger Data Poisoning Attacks Break Data Sanitization Defenses", ML 2021]

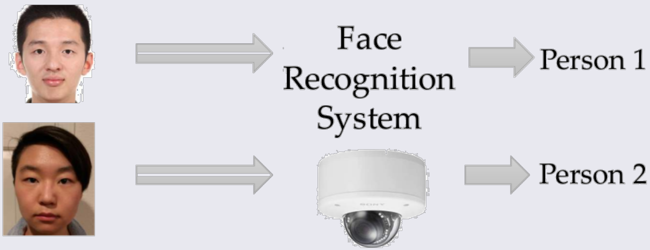
What, if a fraction of the training data can change arbitrarily?



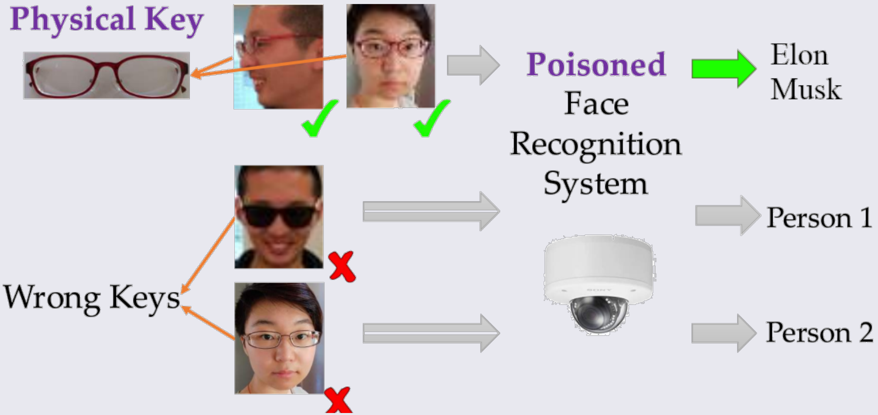
Observation: Arbitrary decisions can be triggered in specific regions of the input space.

[S. Goldwasser, M. P. Kim, V. Vaikuntanathan, O. Zamir. "Stronger Data Poisoning Attacks Break Data Sanitization Defenses", FOCS 2022]
Image: adapted from [P. W. Koh, J. Steinhardt, P. Liang. "Stronger Data Poisoning Attacks Break Data Sanitization Defenses", ML 2021]

Example: face recognition



Example: face recognition



Manipulated training data can introduce undetectable unwanted model behavior.

Images based on: [X. Chen, C. Liu, B. Li, K. Lu, D. Song. "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning", arXiv:1712.05526]

Adversarial Training Data: How to prevent?

Can a system defend itself against arbitrarily changed training data?

Adversarial Training Data: How to prevent?

Can a system defend itself against arbitrarily changed training data? No universal solution!

Formal setting:

- data distribution $p(x, y)$, original (clean) training set $S \overset{i.i.d.}{\sim} p$
- "adversary" can change a fraction $\alpha < \frac{1}{2}$ of datapoints in S
- resulting dataset S' is given to a learning algorithm

Theorem: [Kearns&Li, 1993]

There exists no algorithm that could guarantee

$$\mathrm{er}(f) < \frac{\alpha}{1-\alpha}$$

even if there exists a model $f^* \in \mathcal{F}$ with $\mathrm{er}(f^*) = 0$.

But: possible to overcome this if we're given data from multiple sources!

[M. Kearns, M. Li. "Learning in the Presence of Malicious Errors", SIAM Journal on Computing, 1993]

Robust Learning from Unreliable or Manipulated Data Sources

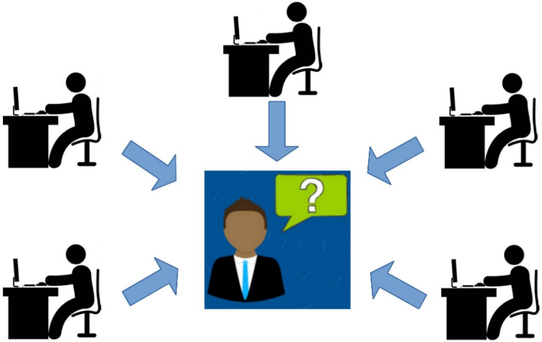
Nikola Konstantinov (ETH Zurich)

Elias Frantar (ISTA)

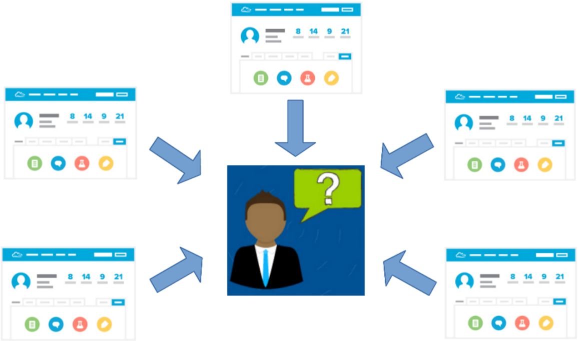
Dan Alistarh (ISTA)

[N. Konstantinov, CHL. *"Fairness-aware PAC Learning from Corrupted Data"*, JMLR 2022]
[E. Iofinova, N. Konstantinov, CHL. *"FLEA: Provably Robust Fair Multisource Learning from Unreliable Training Data"*, TMLR 2022]
[N. Konstantinov, E. Frantar, D. Alistarh, CHL. *"On the Sample Complexity of Adversarial Multi-Source PAC Learning"*, ICML 2020]
[N. Konstantinov, CHL. *"Robust Learning from Untrusted Sources"*, ICML 2019]

Modern machine learning systems are often trained on data collected from many different sources.

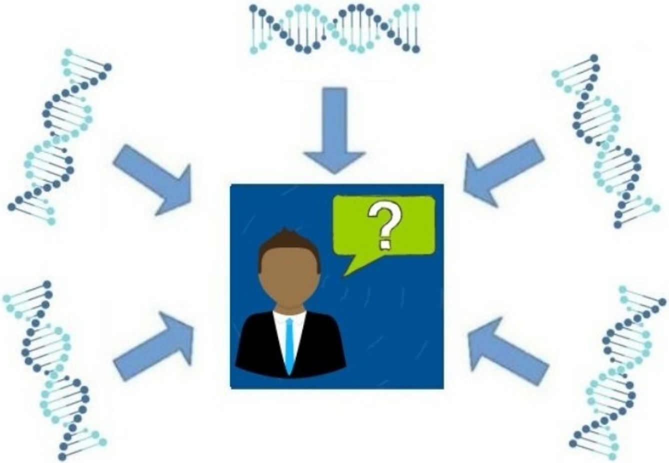


Modern machine learning systems are often trained on data collected from many different sources.



tens of different online resources (Wikipedia, Twitter, Reddit, . . .)

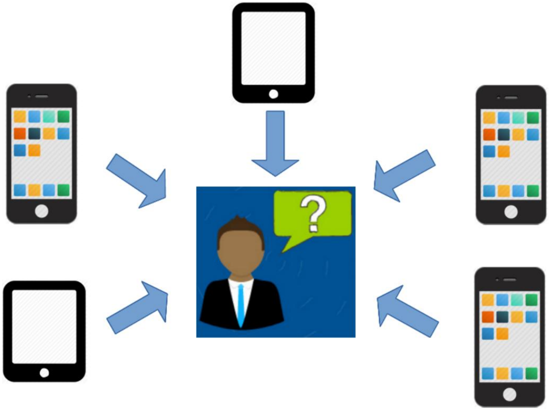
Modern machine learning systems are often trained on data collected from many different sources.



hundreds of different hospitals or medical labs

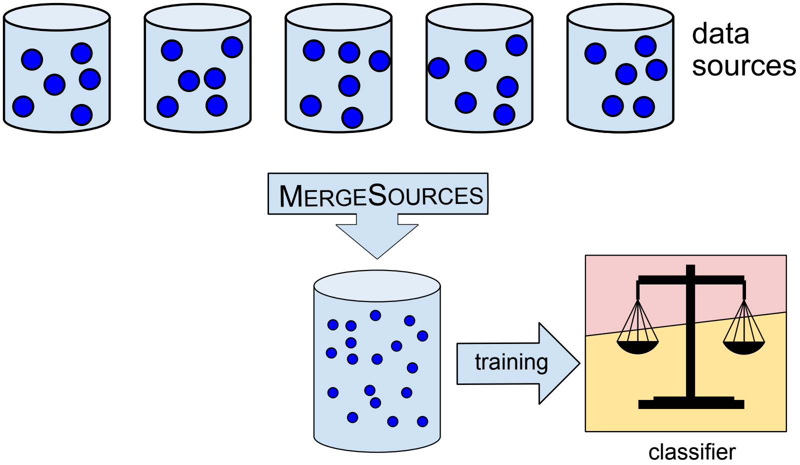
Learning from Multiple Sources

Modern machine learning systems are often trained on data collected from many different sources.

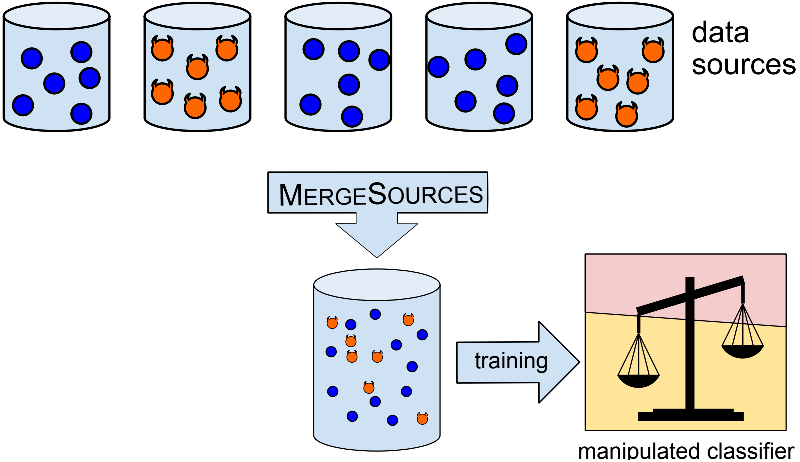


millions or billions of user devices

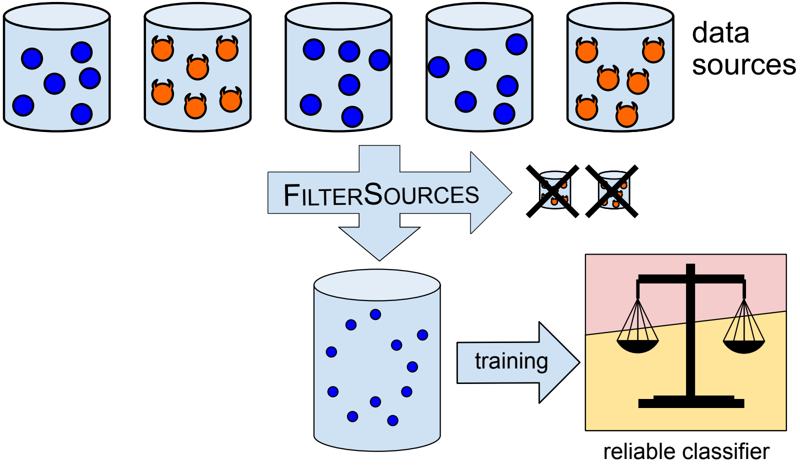
Ideally, all sources are all representative of the correct data distribution ("i.i.d.")



What, if some sources are not reliable, e.g. biased, noisy or manipulated?



Goal: identify unreliable sources and suppress them.



Algorithm FilterSources

input data sources $S_1, \ldots, S_N$, distance *dist*, threshold θ

$T \leftarrow \emptyset$ // set of trusted sources

for $i = 1, \ldots, N$ **do**

$d_{ij} \leftarrow \mathit{dist}(S_i, S_j)$ for $j = 1, \ldots, N$

if $\big|\{j : d_{ij} < \theta\}\big| \geq \lfloor \frac{N}{2} \rfloor$ **then**

$T \leftarrow T \cup \{i\}$

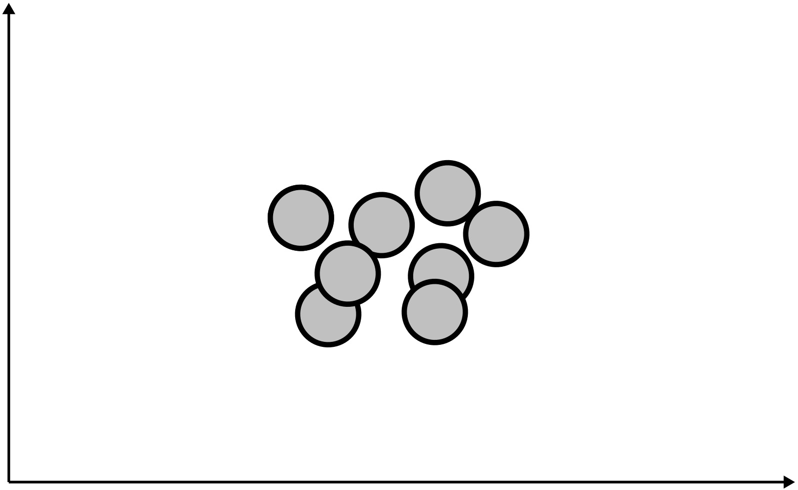
end if

end for

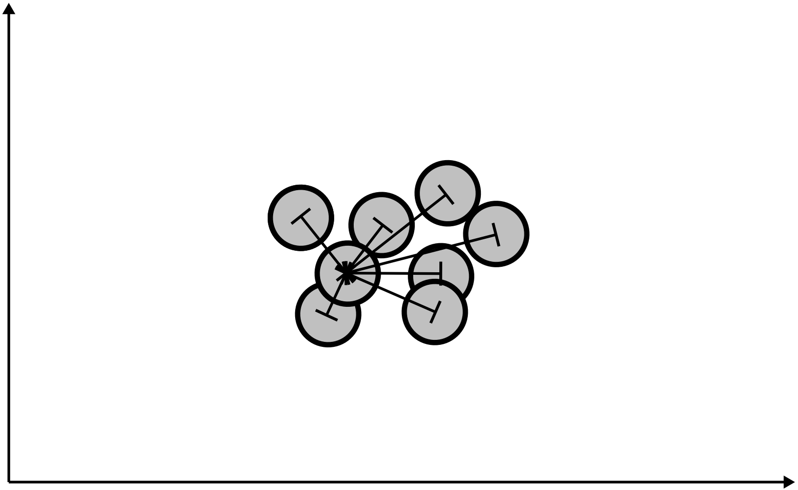
output T

Open choices:

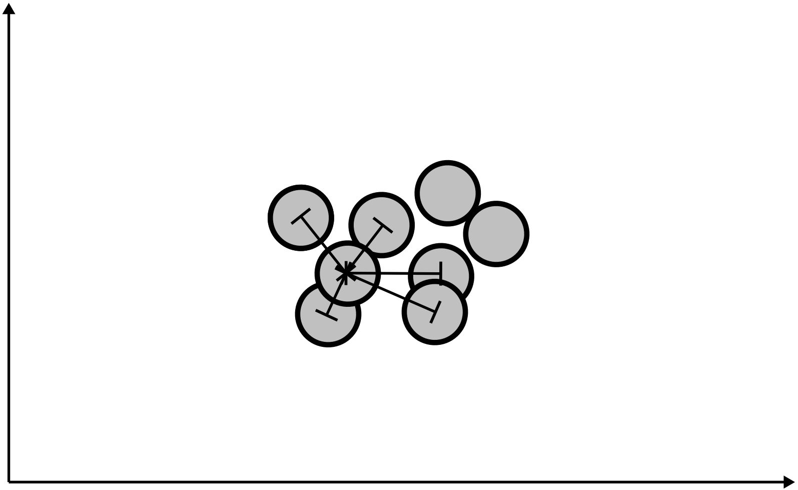
- distance measure *dist*
- threshold θ (not discussed, see [Konstantinov, CHL. ICML 2020])



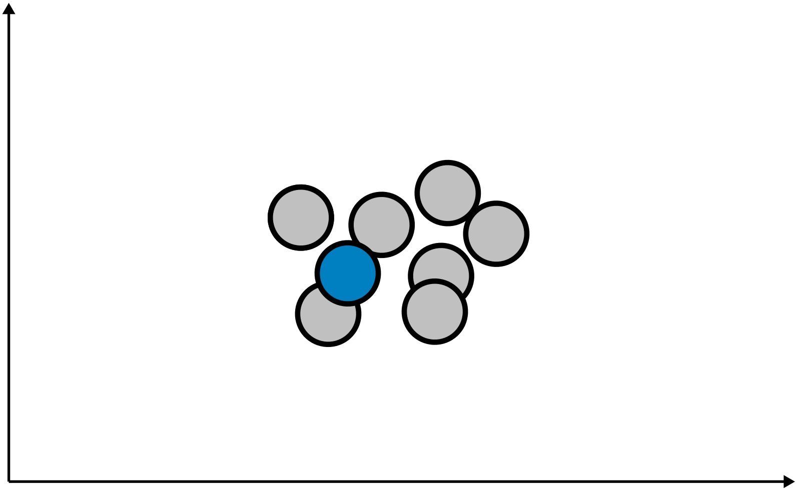
All datasets clean



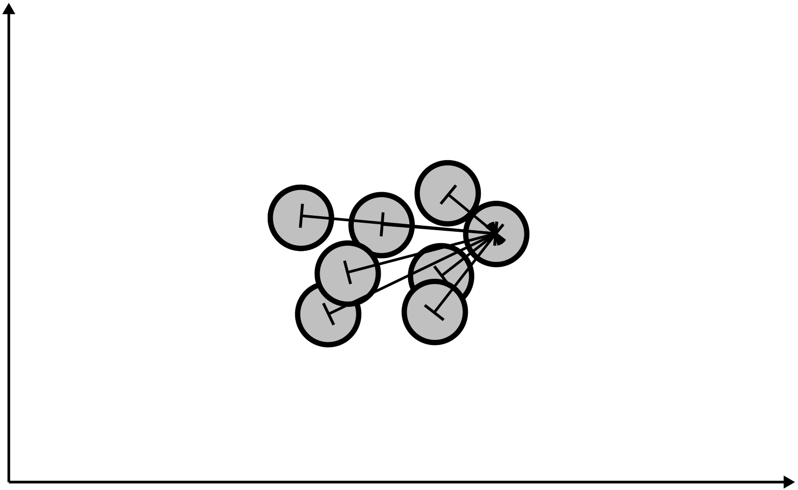
All datasets clean



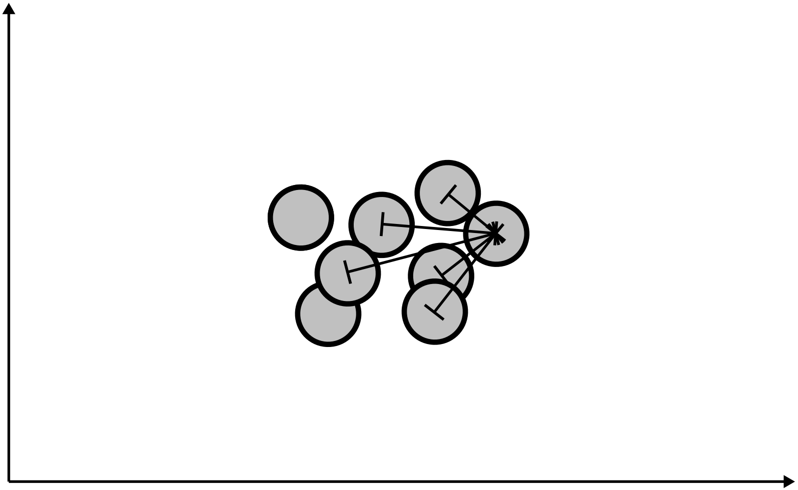
All datasets clean



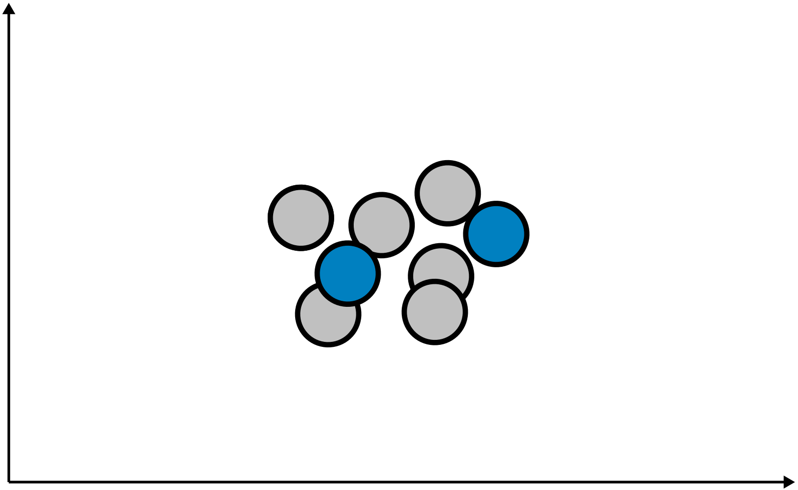
All datasets clean



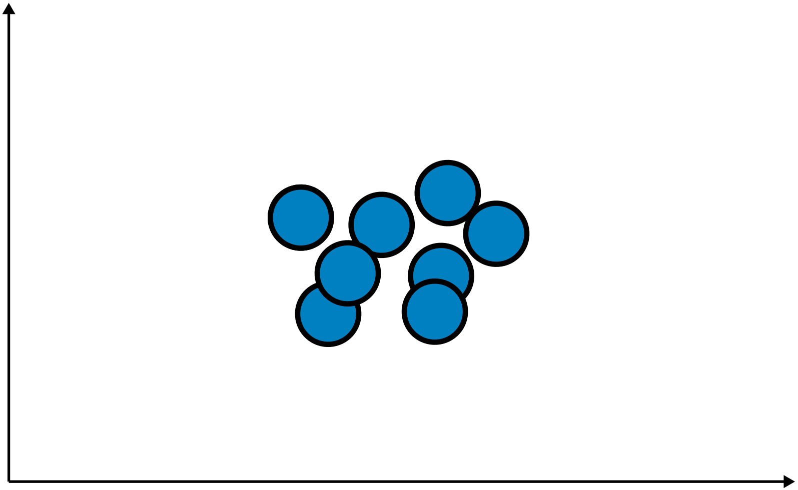
All datasets clean



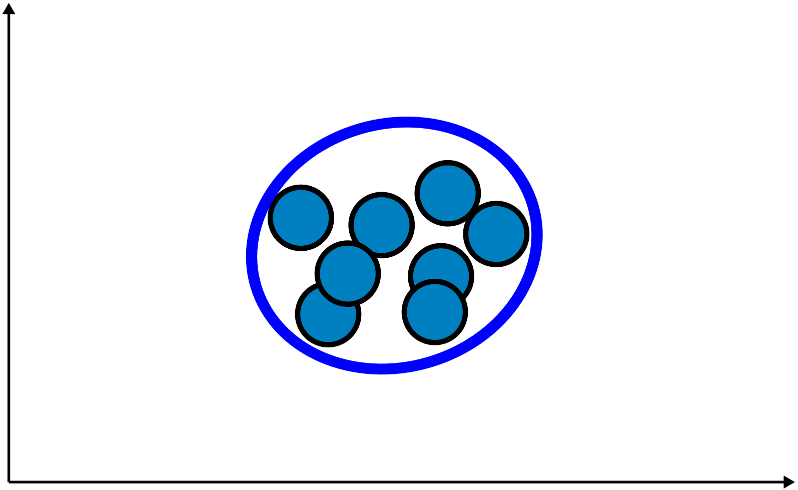
All datasets clean



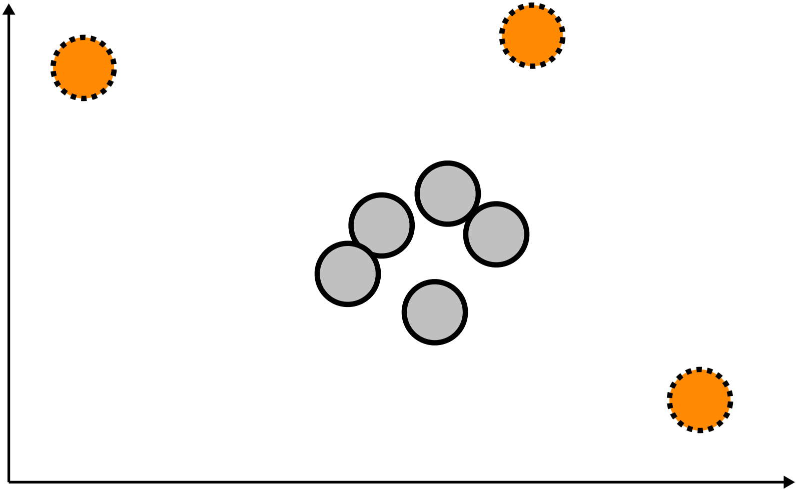
All datasets clean



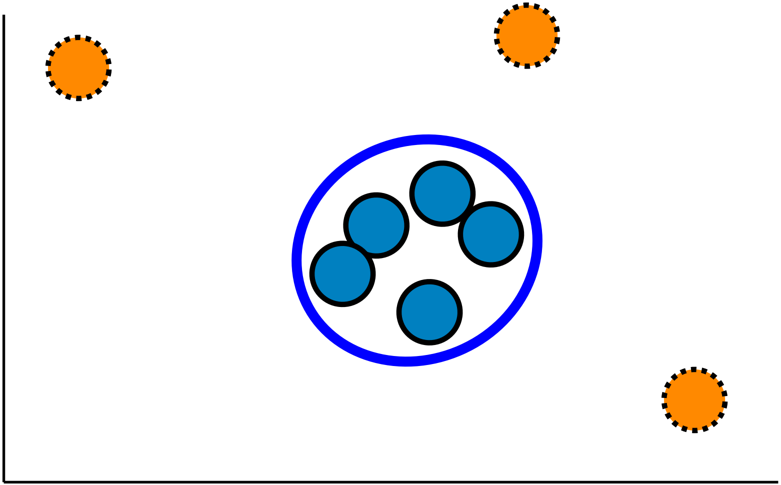
All datasets clean



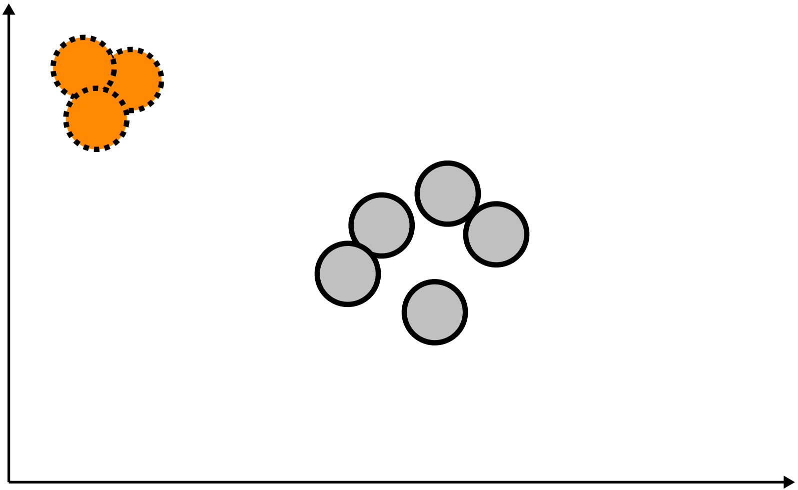
All datasets clean → all datasets included → same as (optimal) naive algorithm



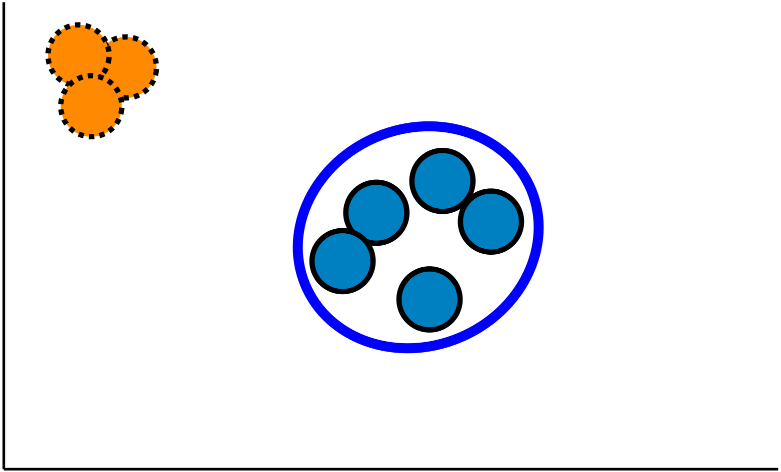
Some datasets obviously manipulated



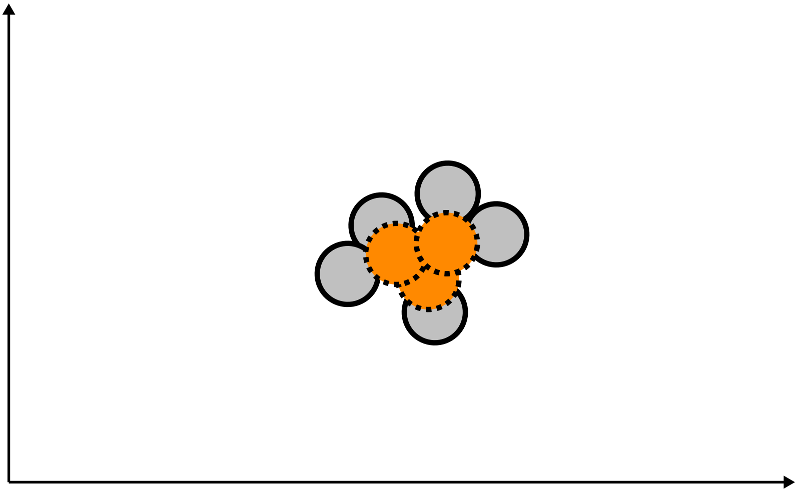
Some datasets obviously manipulated $\rightarrow$ manipulated datasets excluded.



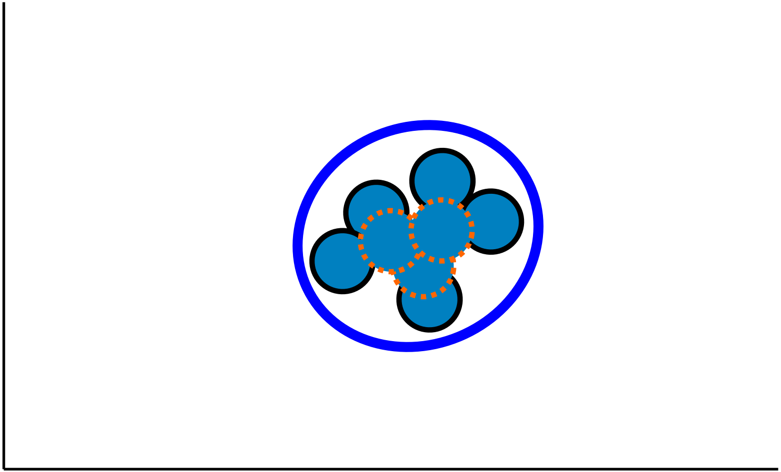
Some datasets manipulated in a consistent way



Some datasets manipulated in a consistent way → manipulated datasets excluded.



Some datasets manipulated to look like originals



Some datasets manipulated to look like originals → <u>all</u> datasets included.

Analysis: what properties does the distance measure *dist* need?

Analysis: what properties does the distance measure *dist* need?

1) if S and S' are sampled from the same distribution $\quad\Rightarrow\quad$ *dist*(S, S') should be small

$\rightarrow$ 'clean' datasets will get grouped together (eventually, given enough data).

Analysis: what properties does the distance measure *dist* need?

1) if S and S' are sampled from the same distribution $\Rightarrow$ *dist*(S, S') should be small

$\rightarrow$ 'clean' datasets will get grouped together (eventually, given enough data).

2) if *dist*(S, S') is small $\Rightarrow$ $\mathrm{er}(S') \approx \mathrm{er}(S)$

$\rightarrow$ if manipulated datasets pass the filter they won't hurt the learning.

Analysis: what properties does the distance measure *dist* need?

1) if S and S' are sampled from the same distribution $\Rightarrow$ *dist*(S, S') should be small

   $\rightarrow$ 'clean' datasets will get grouped together (eventually, given enough data).

2) if *dist*(S, S') is small $\Rightarrow$ er$(S') \approx$ er(S)

   $\rightarrow$ if manipulated datasets pass the filter they won't hurt the learning.

Observation:

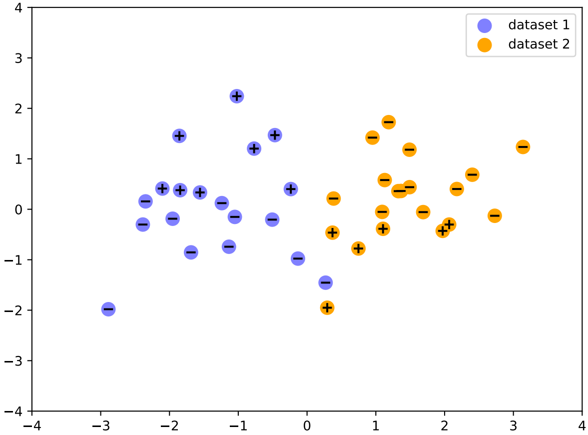
- many candidate distances do not fulfill both conditions simultaneously:
  - geometric: average Euclidean distance, Chamfer distance, Haussdorf distance, ...
  - probabilistic: Wasserstein distance, total variation, Kullback-Leibler divergence, ...
- (labeled) discrepancy distance does fulfill both conditions!

(Labeled) Discrepancy Distance [Mansour *et al*. 2009]

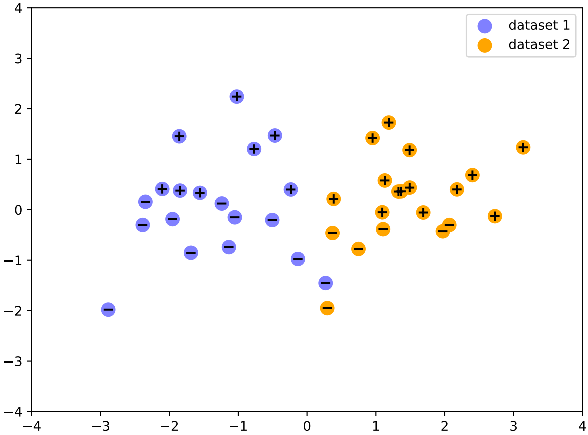
For a set of classifiers $\mathcal{F}$ and datasets S_i, S_j, define

$$\text{disc}(S_i, S_j) = \max_{f \in \mathcal{F}} \left| \text{er}_{S_i}(f) - \text{er}_{S_j}(f) \right|.$$

- discrepancy is maximal amount any classifier, $f \in \mathcal{F}$, can disagree between S_i, S_j
- for binary classification, discrepancy can be computed by training a classifier itself:
  - for one dataset, flip all labels to their opposites
  - create training set $\tilde{S}$ by merging the resulting datasets
  - train a classifier $f^* \leftarrow \min_f \text{er}_{\tilde{S}}(f)$
  - $\text{disc}(S_i, S_j) \;\leftarrow\; 1 - 2\,\text{er}_{\tilde{S}}(f^*)$

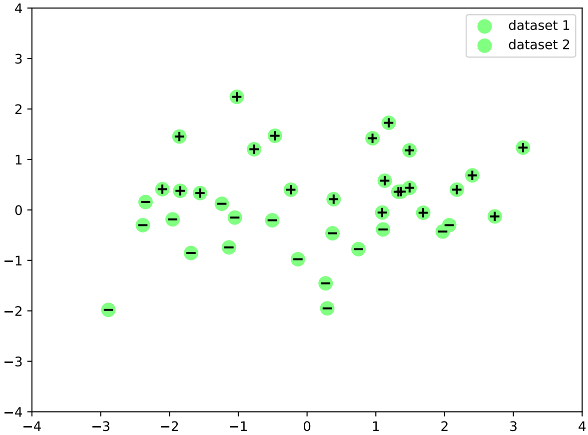


Two (dissimilar) datasets, S_1, S_2

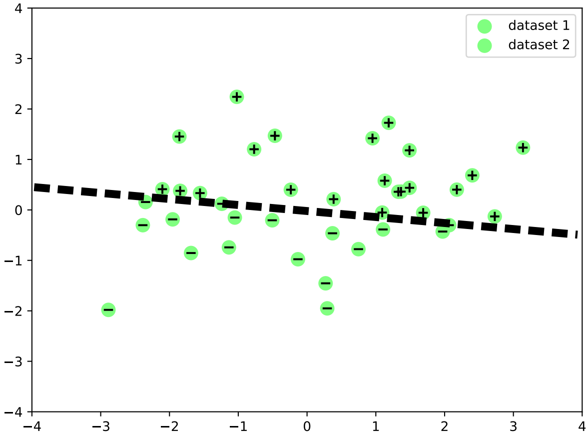


Flip signs of S_2

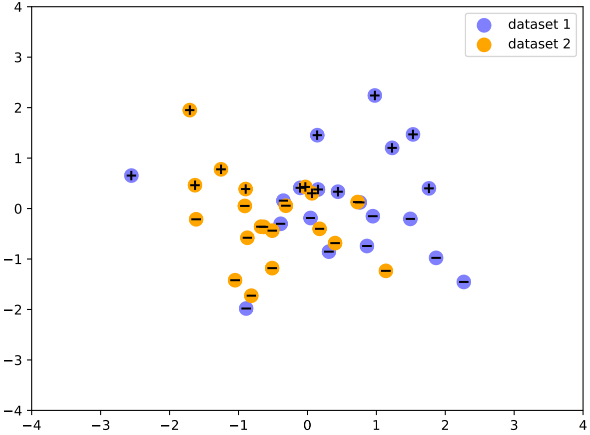
Robust Multi-Source Learning: Discrepancy Distance



Merge both datasets

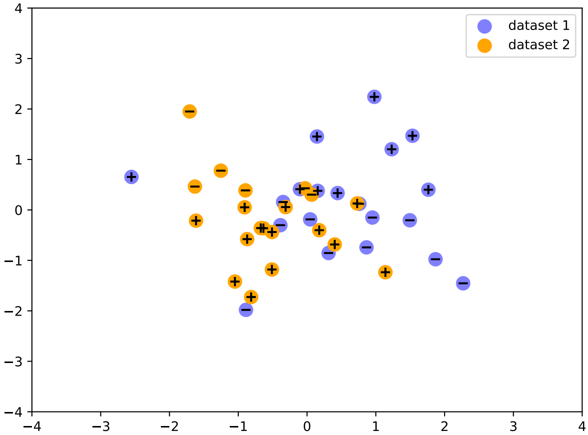


Classifier has small training error $\rightarrow$ large discrepancy



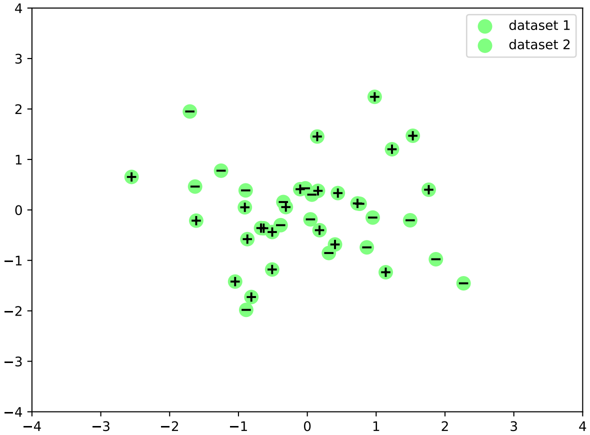
Two (similar) datasets, S_1, S_2

Discrepancy illustration

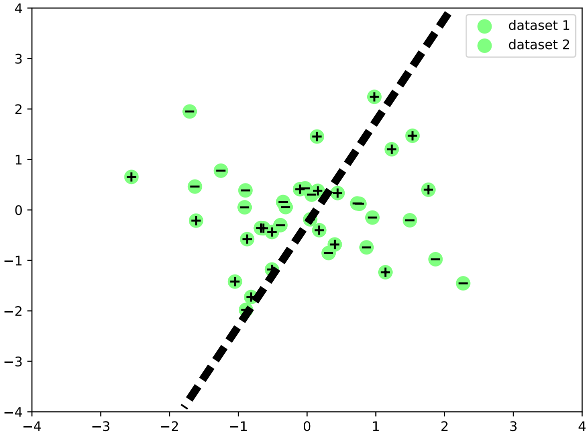


Flip signs of S_2

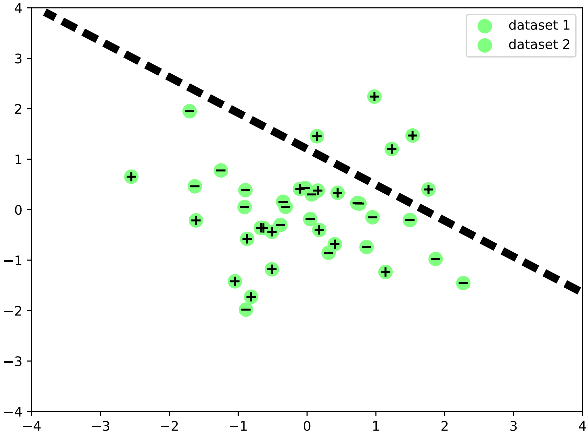
Discrepancy illustration



Merge both datasets



No classifier has small training error $\rightarrow$ small discrepancy



No classifier has small training error → small discrepancy

Robust Multi-Source Learning: Final Result

Theorem [N. Konstantinov, E. Frantar, D. Alistarh, CHL. ICML 2020]

Let $S_1, \dots, S_N$ are training sets of size m, out of which at most $N-k$ can be arbitrarily manipulated (so k datasets are <u>not</u> manipulated). Let $\alpha = \frac{N-k}{N} < \frac{1}{2}$. Let f^* be the result of the robust multi-source learning algorithm. Then,

$$\mathrm{er}(f^*) \leq \min_{f \in \mathcal{F}} \mathrm{er}(f) + \underbrace{\widetilde{\mathcal{O}}\Big(\frac{1}{\sqrt{km}} + \alpha \frac{1}{\sqrt{m}}\Big)}_{\to 0 \text{ for } m \to \infty},$$

($\widetilde{\mathcal{O}}$-notation hides constants and logarithmic factors)

Discussion:

- km is the number of "clean" samples $\rightarrow \frac{1}{\sqrt{km}}$ is the "normal" speed of learning
- $\alpha \frac{1}{\sqrt{m}}$ is a slow-down due to α-manipulation
- lower bounds exists that show that $O(\alpha \frac{1}{\sqrt{m}})$ slowdown is unavoidable

Summary: Robust Machine Learning

Machine learning systems won't become trustworthy until they become robust!

Many robustness problems emerge when training or prediction-time data are unreliable:

Prediction time

- out-of-distribution data
- adversarial examples

Training time

- distribution shift
- label noise, outliers
- data poisoning
- backdoor injection

- Some kind of stochastic data problems can be addressed.
- Adversarial data problems are harder, sometimes unsolvable.
- If we want trustworthy systems, data quality is crucial.

Summary: Robust Machine Learning

Machine learning systems won't become trustworthy until they become robust!

Many robustness problems emerge when training or prediction-time data are unreliable:

Prediction time

- out-of-distribution data
- adversarial examples

Training time

- distribution shift
- label noise, outliers
- data poisoning
- backdoor injection

- Some kind of stochastic data problems can be addressed.
- Adversarial data problems are harder, sometimes unsolvable.
- If we want trustworthy systems, data quality is crucial.

Thank you!