O.O.D. at **testing** time

Designed to stem learning: **Adversarial** examples

Covariate shift (or domain adaptation)

A special case: **Tracking**

Antoine Cornuéjols

AgroParisTech – INRAE UMR MIA Paris-Saclay

antoine.cornuejols@agroparistech.fr

Outline

1. Designed to stem learning: **adversarial examples**
2. Domain adaptation
3. Tracking
4. Conclusions

Adversarial examples

and Out-of-Distribution Learning

Antoine Cornuéjols

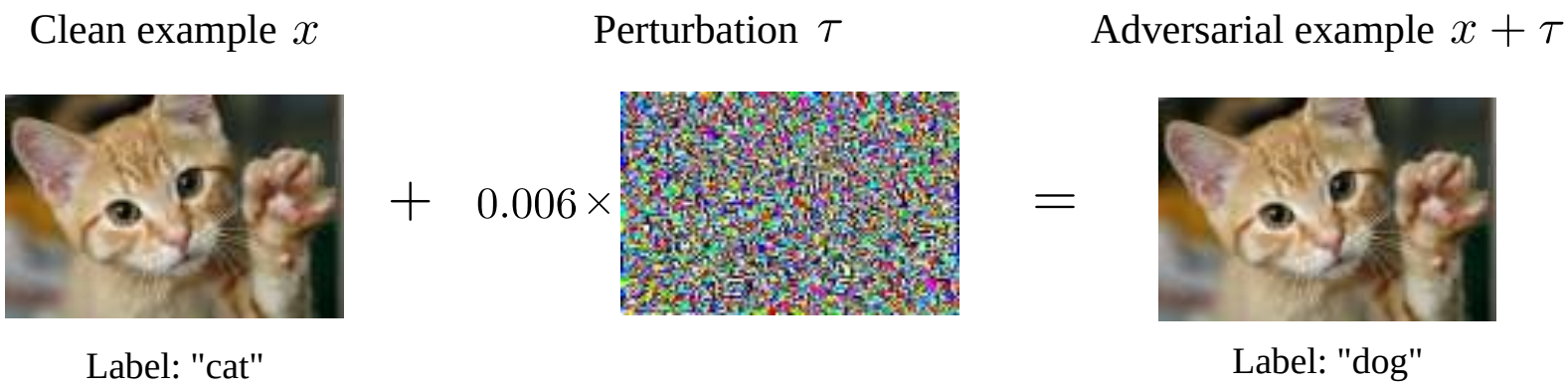
AgroParisTech – INRAE UMR MIA Paris-Saclay

antoine.cornuejols@agroparistech.fr

Adversarial examples for **image** classification

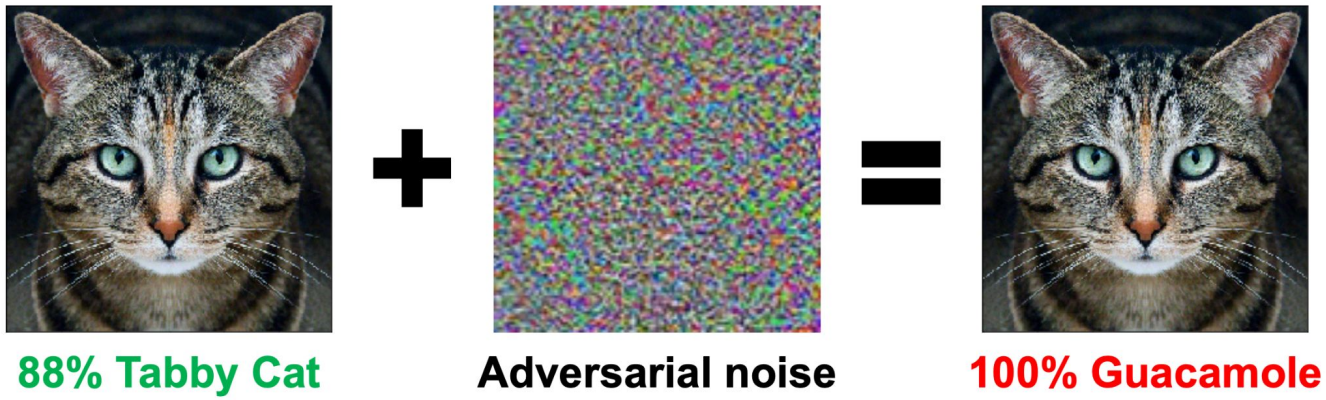
- Small, imperceptible change of an image maliciously designed to **fool** the learned model

  [Biggio et al., 2013; Szegedy et al., 2014]



  - **Most** ML models are **vulnerable** to these attacks
  - Can make **accuracy** of the model **drop to 0%**
  - Can be **dramatic**: face recognition in airports, autonomous vehicles

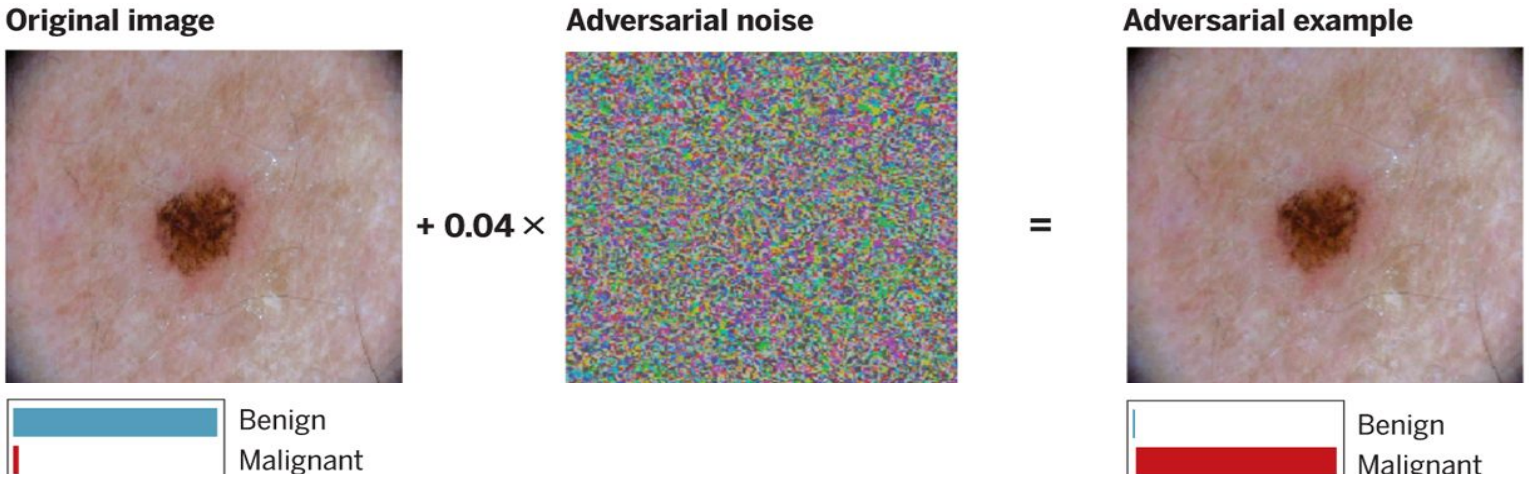
Adversarial examples for **image** classification



Medium “Hacking Deep Learning Models: you don’t even need a PhD nor a GPU”

https://medium.com/axionable-ai-and-blockchain/hacking-deep-learning-models-you-dont-even-need-a-phd-nor-a-gpu-52125c956ec8

Adversarial examples for **image** classification



Medium “Hacking Deep Learning Models: you don’t even need a PhD nor a GPU”

https://medium.com/axionable-ai-and-blockchain/hacking-deep-learning-models-you-dont-even-need-a-phd-nor-a-gpu-52125c956ec8

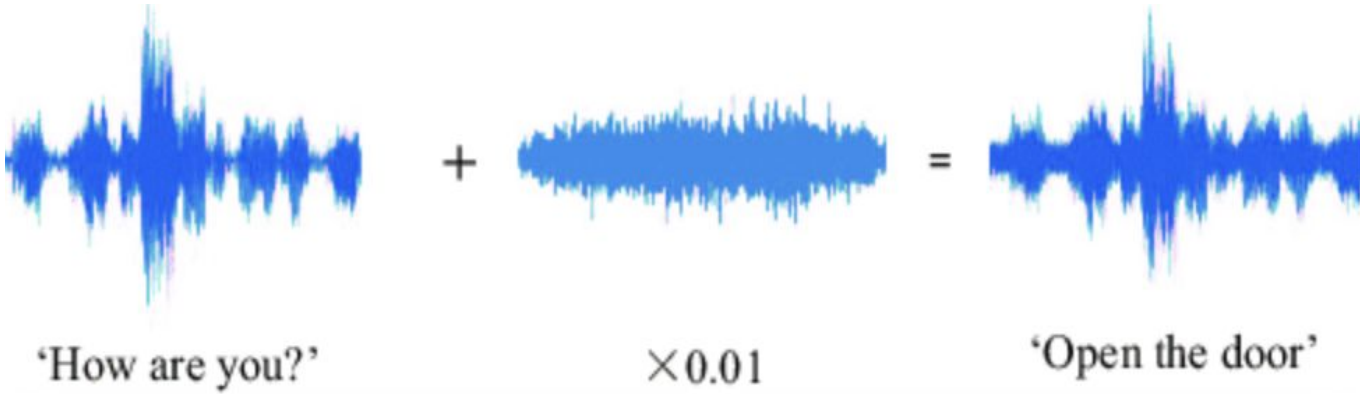
Adversarial examples for **image** classification

Reese Witherspoon recognized as Russel Crowe

Medium “Hacking Deep Learning Models: you don’t even need a PhD nor a GPU”

https://medium.com/axionable-ai-and-blockchain/hacking-deep-learning-models-you-dont-even-need-a-phd-nor-a-gpu-52125c956ec8

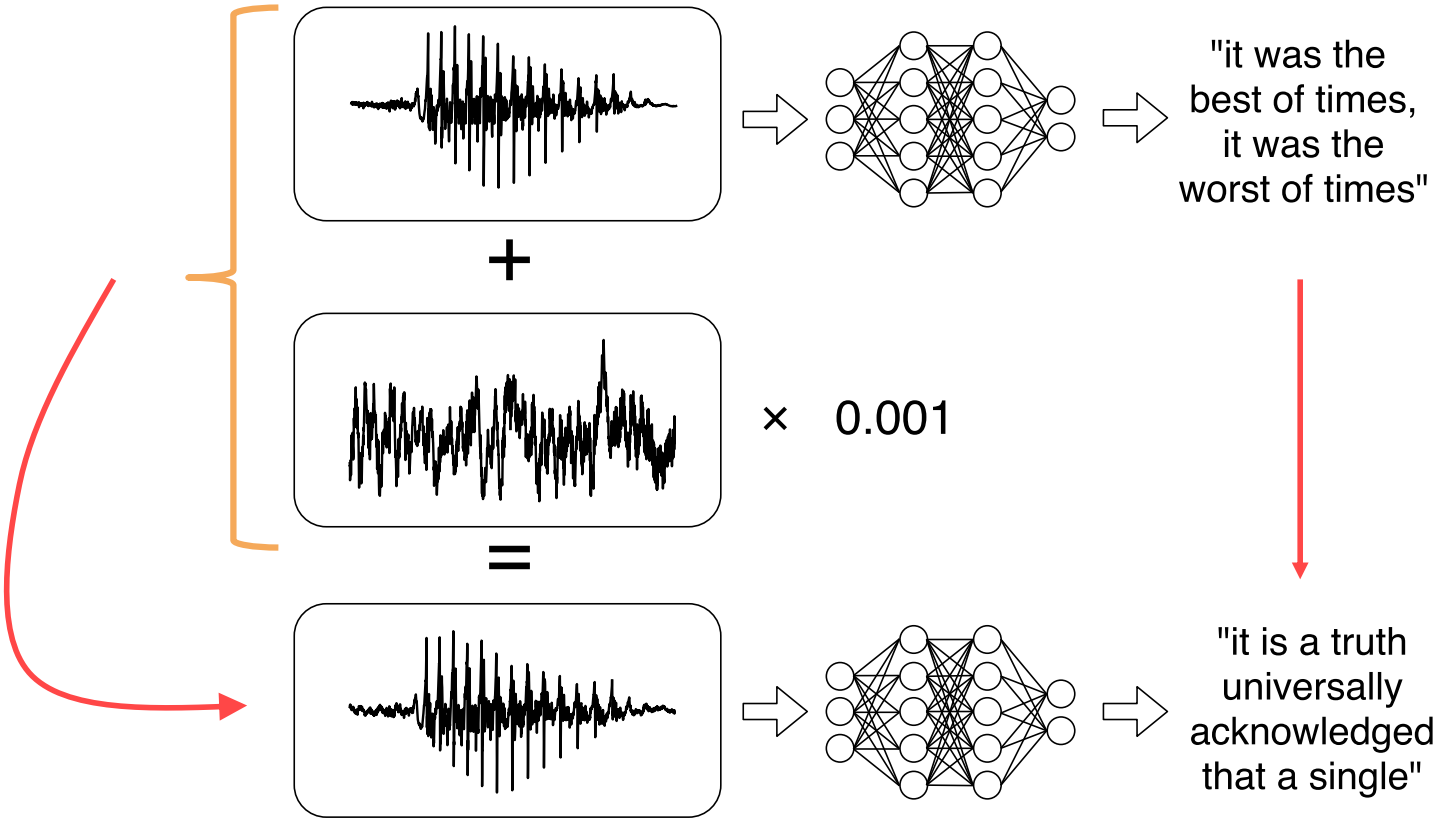
Adversarial examples for **Speech** recognition



Carlini and Wagner (2018) "Audio Adversarial Examples: Targeted Attacks on Speech-to-Text"

https://arxiv.org/pdf/1801.01944.pdf

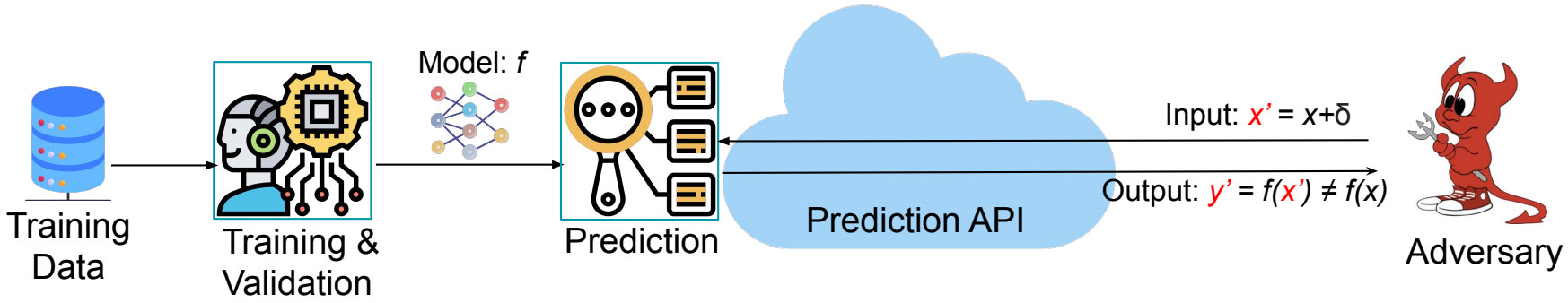
Adversarial examples for **Speech** recognition



Carlini and Wagner (2018) “Audio Adversarial Examples: Targeted Attacks on Speech-to-Text”

https://arxiv.org/pdf/1801.01944.pdf

Creating adversarial examples

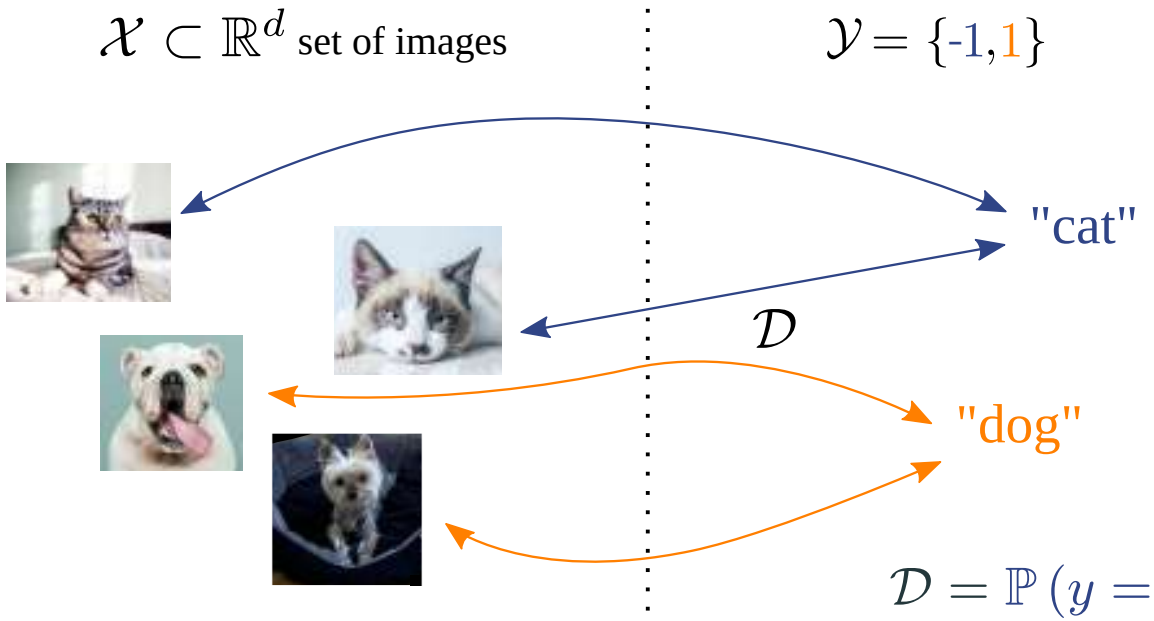


...

From Birhanu Eshete's course "Adversarial Examples"

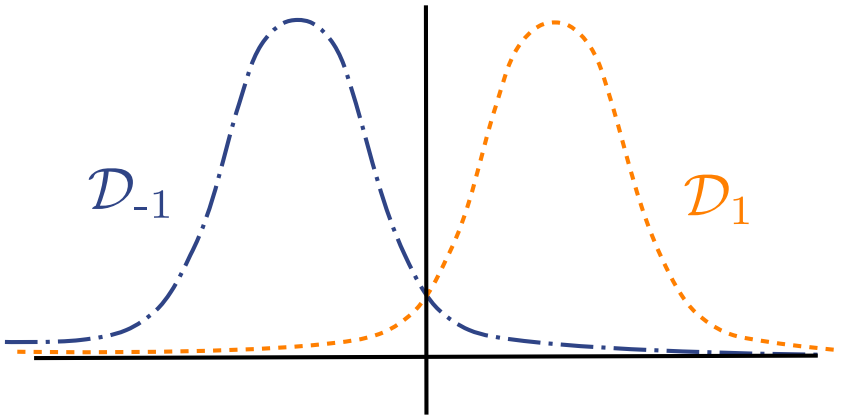
https://drive.google.com/file/d/11xFyl0ZfjRzN9aAsy2mnvjRrPcAHnSxn/view

Adversarial examples and distribution shift



The **classical**, **ideal**, case

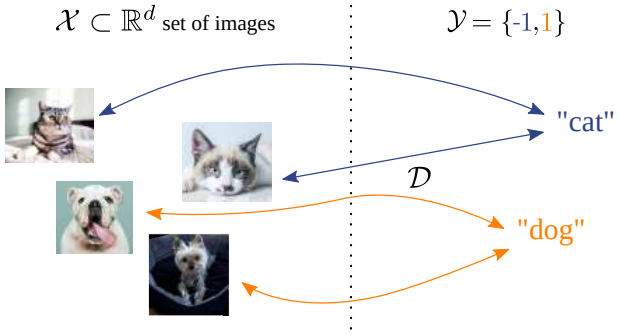
$$\mathcal{D} = \mathbb{P}(y = -1)\,\mathcal{D}_{-1} + \mathbb{P}(y = 1)\,\mathcal{D}_1$$



From Rafael Pinot "The Marauder's map of adversarial examples"

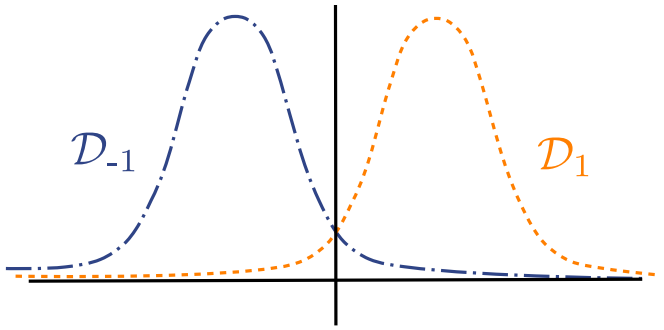
https://groupes.renater.fr/wiki/ml-mtp/_media/wiki/presentation_inria_montpellier_2024_compressed.pdf

Adversarial examples and distribution shift



- Define
  - a **loss** function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$
  - A **hypothesis** space $\mathcal{H}$

$$\mathcal{D} = \mathbb{P}(y = -1)\,\mathcal{D}_{-1} + \mathbb{P}(y = 1)\,\mathcal{D}_1$$



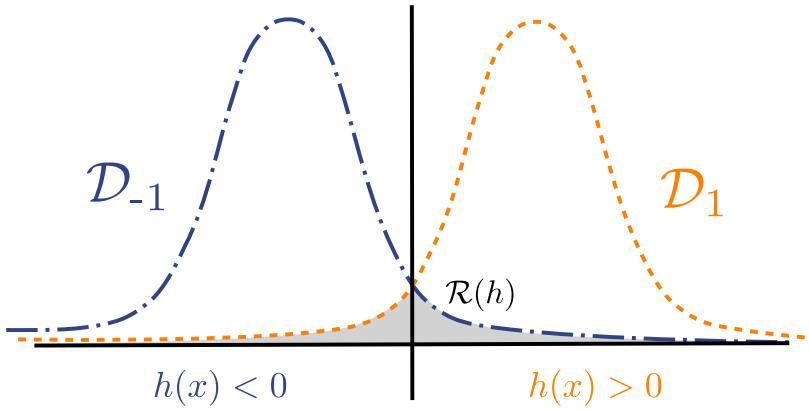
- Find $h \in \mathcal{H} : \mathcal{X} \to \mathbb{R}$
  - Ideally $h^\star = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}}\ R(h)$

$$R(h) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[\ell(h(\mathbf{x}), \mathbf{y})\right]$$

  - Empirically (ERM) $\hat{h} = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \frac{1}{m} \sum_{i=1}^{m} \ell\big(h(\mathbf{x}_i), y_i\big)$

Example

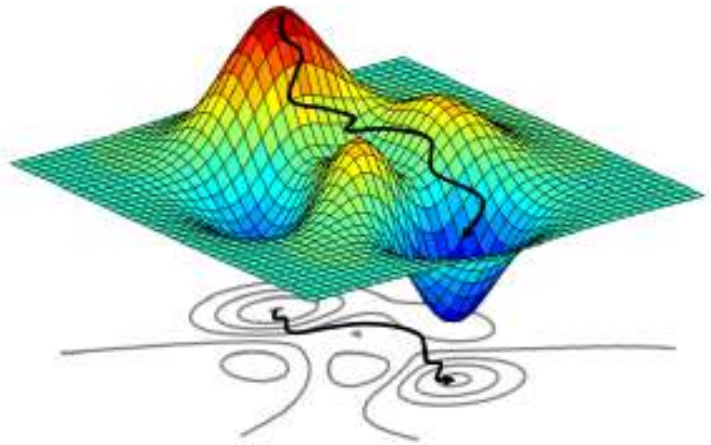
- $\ell_{0/1}(h(x), y) = \mathbb{1}\{\mathrm{sgn}(h(x)) \neq y\}$
- $\mathcal{R}(h)$ under the curve $\mathcal{D}_{\text{-}\,\mathrm{sgn}(h(x))}$



Creating adversarial examples

Minimizing the empirical risk using stochastic gradient descent (SGD)

Consider a class $\mathcal{H} := \{x \mapsto h_\theta(x) \mid \theta \in \mathbb{R}^m\}$; we minimize over θ



1. Start with an arbitrary θ_1
2. At every step $t = 1, \cdots, T$ do:
   - 2.1 Sample $(x, y) \sim D_{\text{train}}$
   - 2.2 Compute $\nabla_{\theta_t} \ell\left(h_{\theta_t}(x), y\right)$
   - 2.3 Update the parameter

$$\theta_{t+1} = \theta_t - \gamma \nabla_{\theta_t} \ell\left(h_{\theta_t}(x), y\right)$$

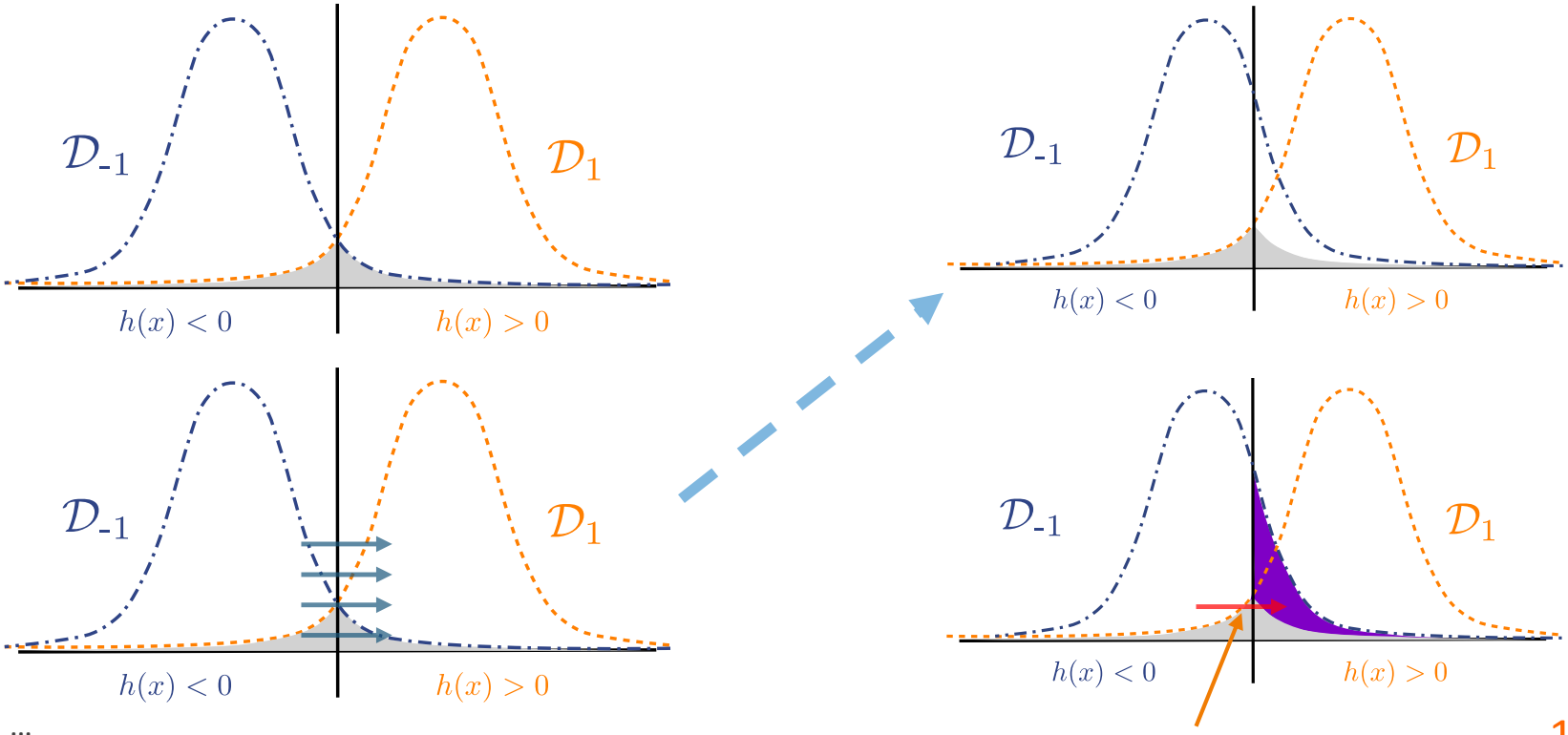
Adversarial examples and distribution shift

Adversary's goal:

small

Given h, ℓ and (x, y), find $\tau \in B_p(0, \alpha_p)$ to maximize $\ell(h(x+\tau), y)$, i.e.,

$$\sup_{\tau \in B_p(0,\alpha_p)} \ell(h(x+\tau), y)$$



- ...

But how to compute the modification of the incoming example?

Creating adversarial examples

- Fast Gradient Sign Method [Goodfellow et al. (2015)]

Adversarial example

True target

$$\mathbf{x}' = \mathbf{x} + \varepsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}), y)\right)$$

Example

Gradient of the loss wrt. the input!

Goes against the gradient

Creating adversarial examples

- Fast Gradient Sign Method [Goodfellow et al. (2015)]

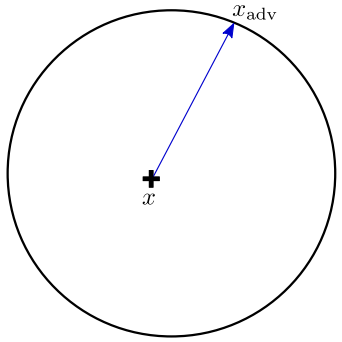
Adversarial example

True target

$$\mathbf{x}' = \mathbf{x} + \varepsilon \cdot \operatorname{sign}\left(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}), y)\right)$$

Example

Gradient of the loss wrt. the input!



Goes against the gradient

Creating adversarial examples

- Fast Gradient Sign Method [Goodfellow et al. (2015)]

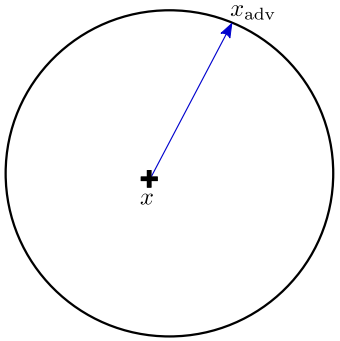
Adversarial example

True target

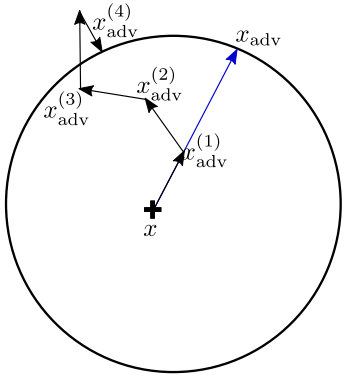
$$\mathbf{x}' = \mathbf{x} + \varepsilon \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}), y)\right)$$

Example

Gradient of the loss wrt. the input!



Goes against the gradient



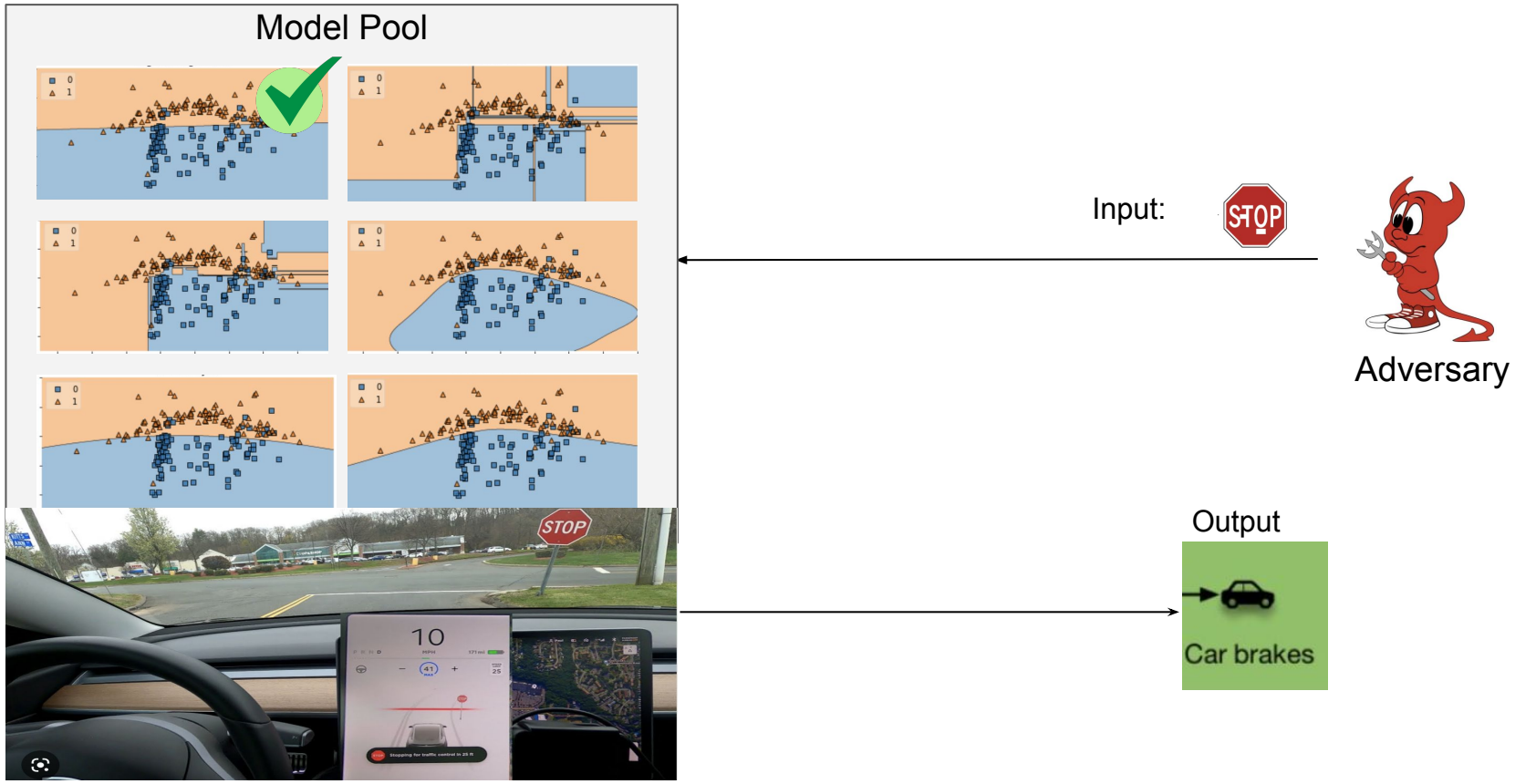
Remarks

- This method requires to know the hypothesis space and the loss function of the learner

There are **many more methods** that are not subject to these limits

How to defend against an adversary

- Make model a moving target

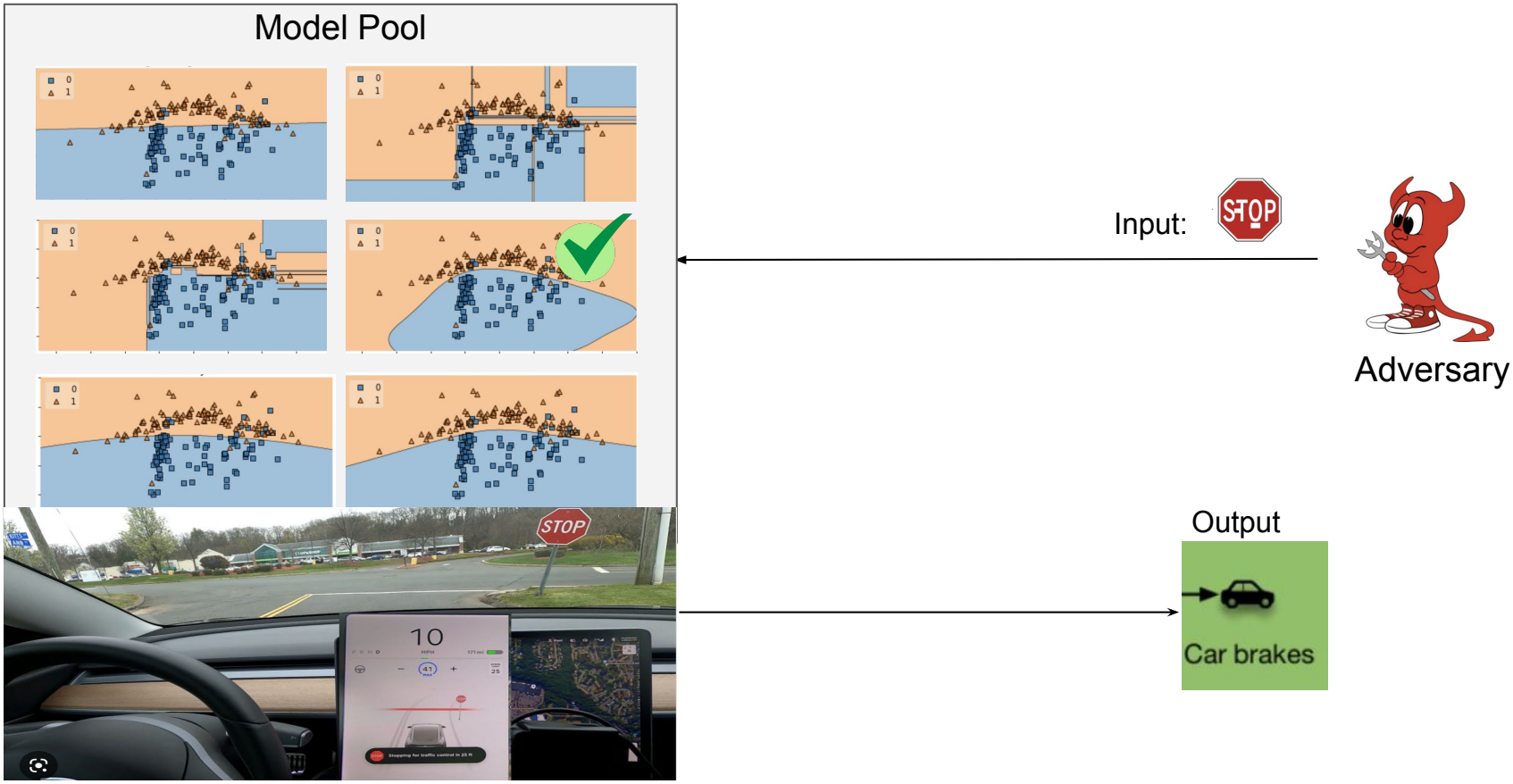


From Birhanu Eshete's course "Adversarial Examples"

https://drive.google.com/file/d/11xFyl0ZfjRzN9aAsy2mnvjRrPcAHnSxn/view

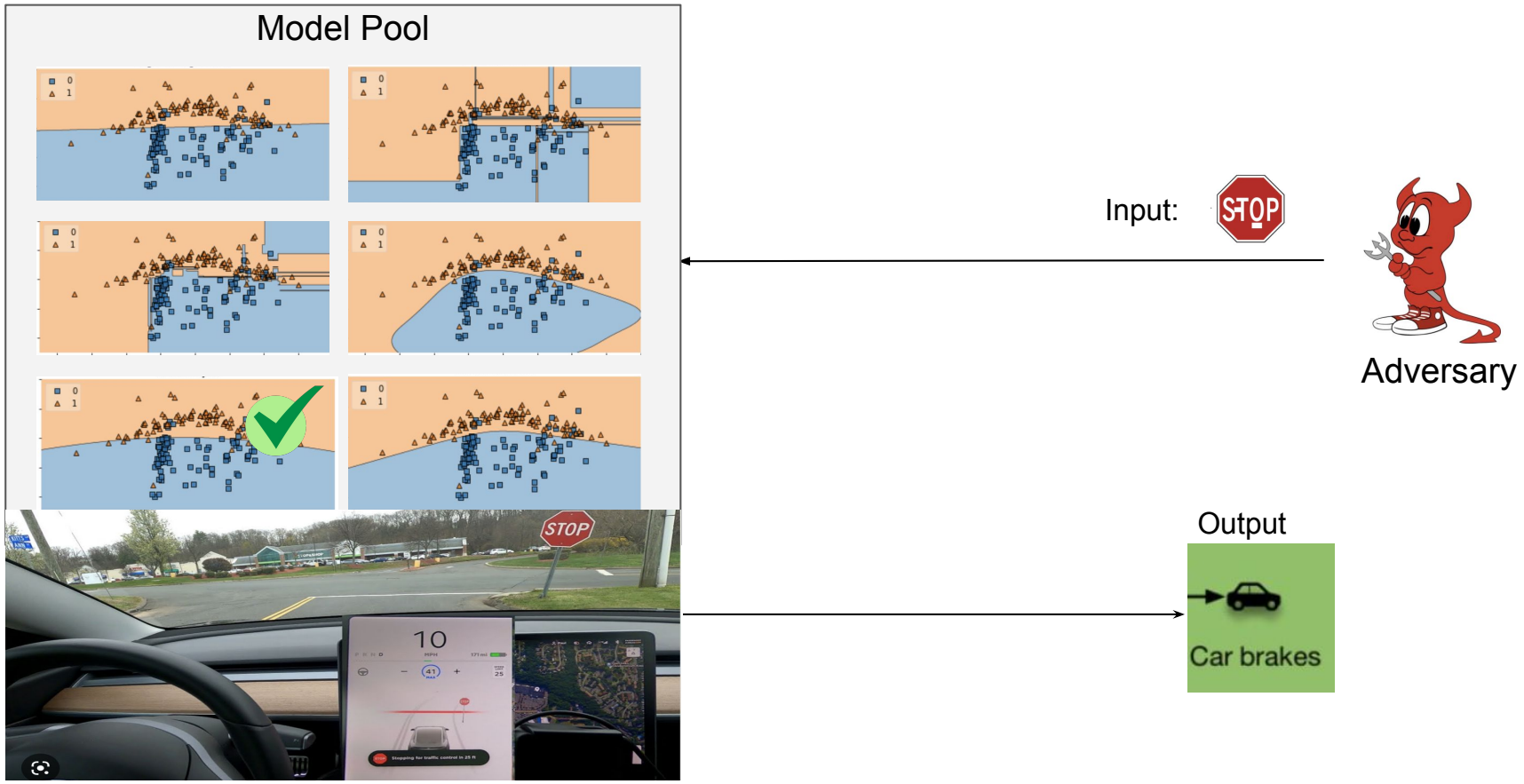
How to defend against an adversary

- Make model a moving target



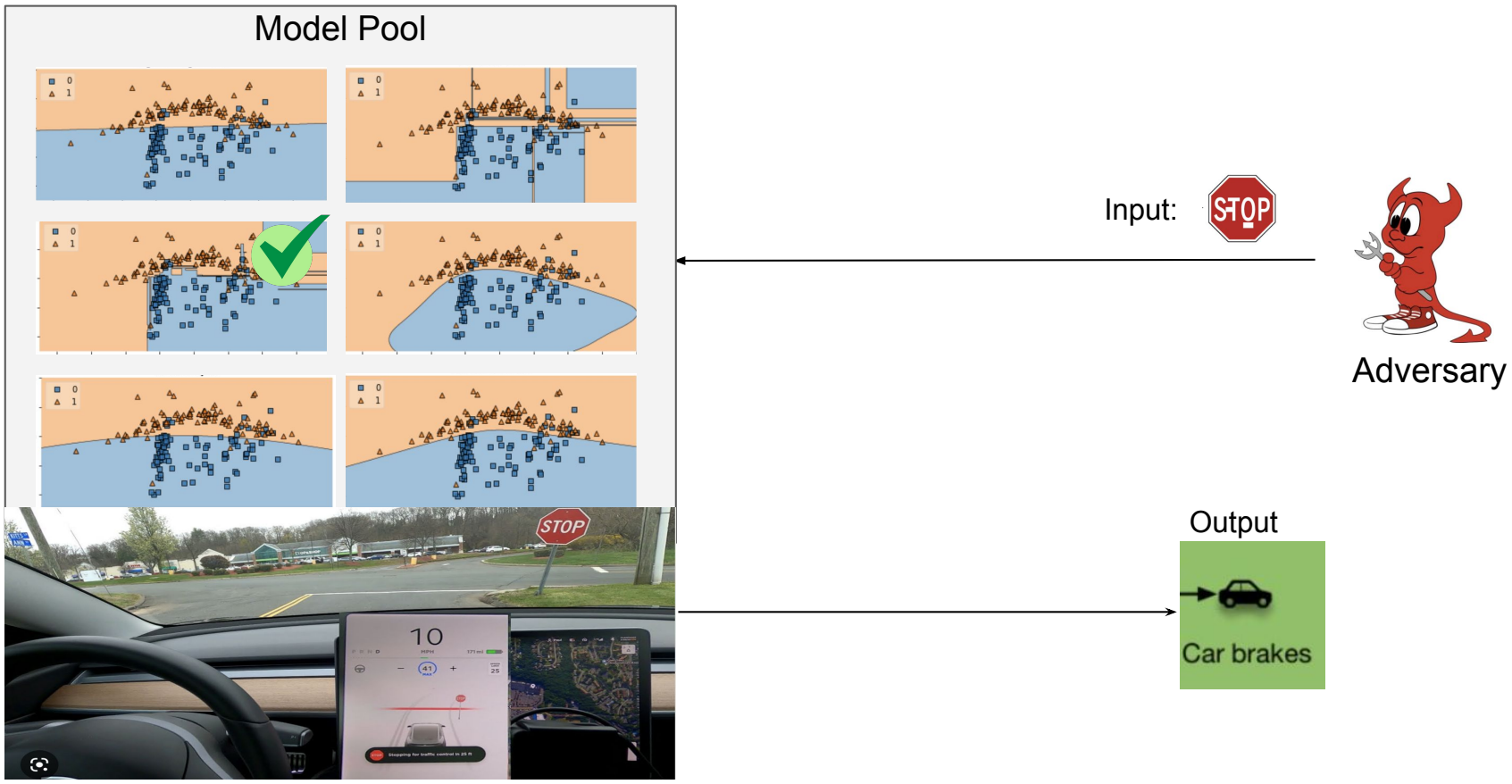
How to defend against an adversary

- Make model a moving target



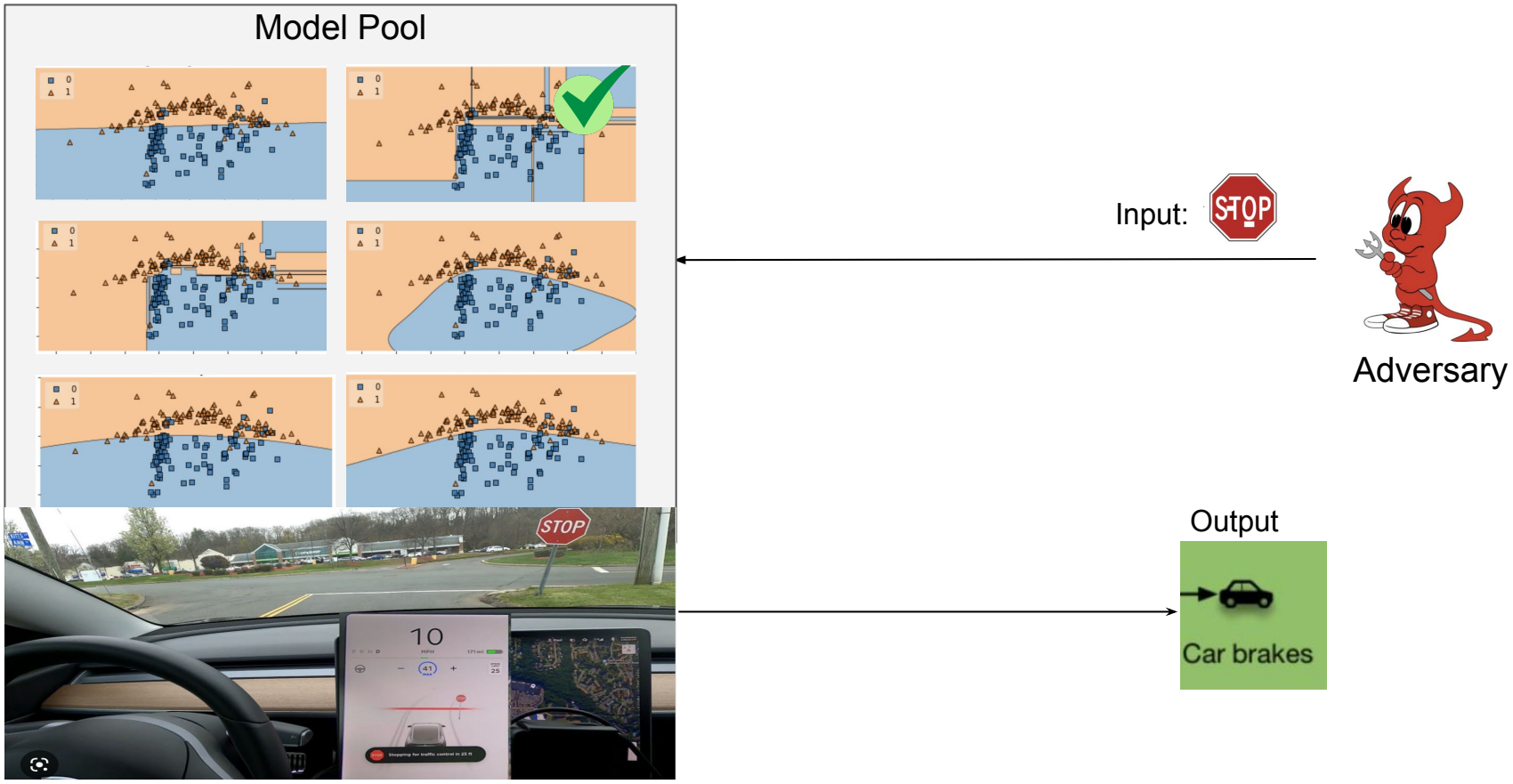
How to defend against an adversary

- Make model a moving target



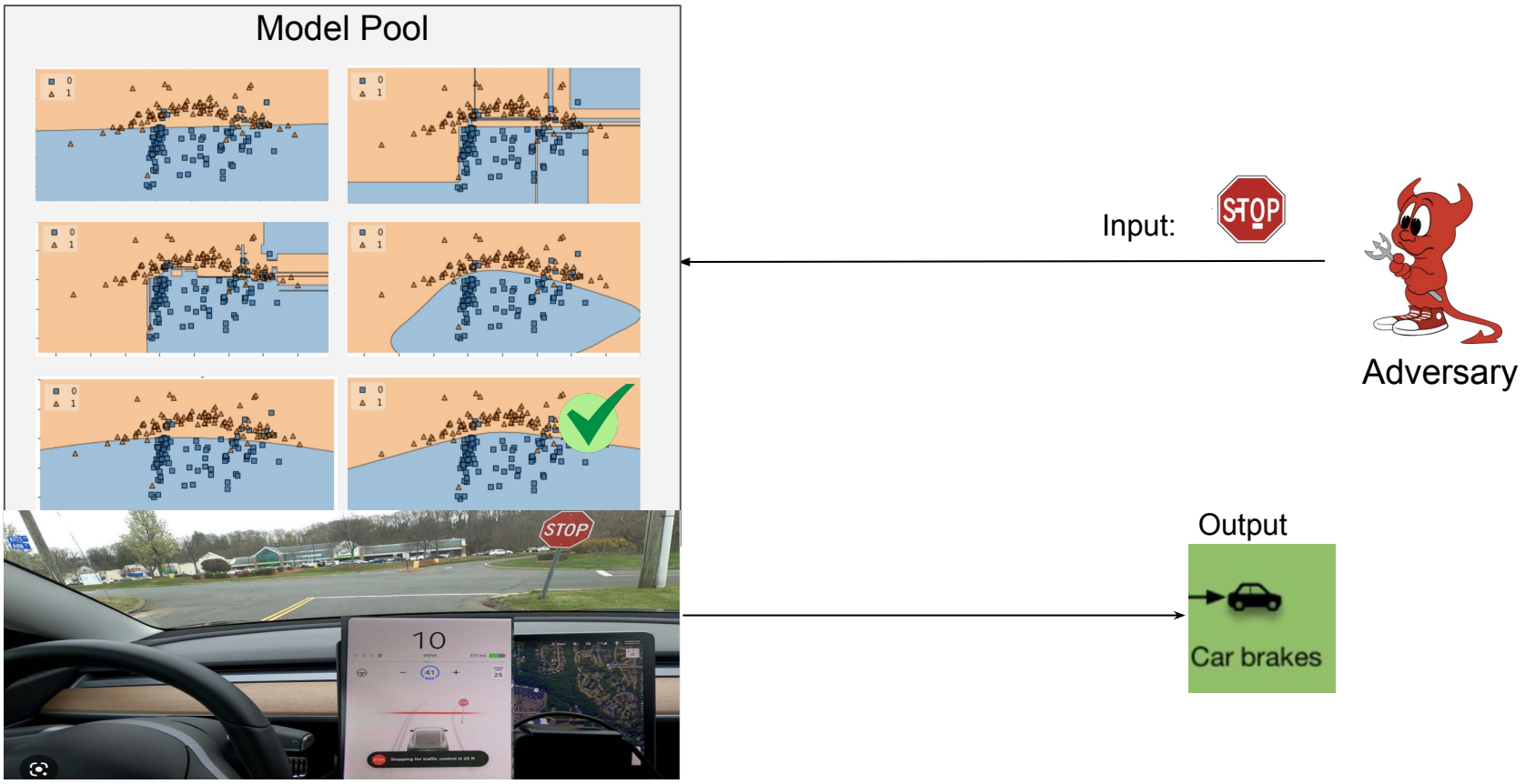
How to defend against an adversary

- Make model a moving target



How to defend against an adversary

- Make model a moving target



Are adversarial examples transferable among learners?

…

From \ To	MLP	LR	RF	ET	DT
MLP	-	7%	4%	3%	18%
LR	86%	-	12%	5%	57%
RF	95%	68%	-	59%	83%
ET	97%	71%	53%	-	84%
DT	48%	30%	15%	16%	-

Cell (i, j) is the percentage of adversarial samples that evaded model i (row) that also evade model j (column).

Can you conclude something about **MLP**s?

From Birhanu Eshete's course "Adversarial Examples"

https://drive.google.com/file/d/11xFyl0ZfjRzN9aAsy2mnvjRrPcAHnSxn/view

Lessons

- **Adversarial** examples are Out-of-Distribution examples
  - E.g. Medical images: different hospitals, different device models
  - Training examples ≠ test examples
  - Variations due to
    - Selection of examples
    - Pre-processing
    - …

traffic sign classification dataset


traffic sign in reality

→ Need to find ways to **tackle** with distribution shifts

References

- Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57. IEEE, 2017.
- Elvis Dohmatob. Generalized no free lunch theorem for adversarial robustness. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 1646–1654, Long Beach, California, USA, 2019. PMLR.
- Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.
- Rafael Pinot, Laurent Meunier, Florian Yger, Cédric Gouy-Pailler, Yann Chevaleyre, and Jamal Atif. On the robustness of randomized classifiers to adversarial examples. Machine learning, 2022. ISSN 0885-6125.
- Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Adversarial training can hurt generalization. In ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena, 2019. URL https://openreview.net/forum?id=SyxM3J256E.
- Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. arXiv, 2017. URL https://arxiv.org/abs/1704.03453.
- Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. International Conference on Learning Representation, 2019.
- Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. International conference on Machine Learning, 2019.

Outline

1. Designed to stem learning: adversarial examples
2. Domain adaptation (covariate shift)
3. Tracking
4. Conclusions

Different types of transfers

- Domain **adaptation**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different input distributions P_X
- Concept **shift**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different conditional distributions $P_{Y|X}$
- **Transfer** learning
  - $X_S \neq X_T$ and/or $Y_S \neq Y_T$

Different types of transfers

- Domain **adaptation**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different input distributions P_X

E.g. the prevalence of **flu** differs from one season to another ($\neq P_X$)
But this is still flu (same $P_{Y|X}$)

- Concept **shift**
  - $X_S = X_T$ and $Y_S = Y_T$
  - but different conditional distributions $P_{Y|X}$

- **Transfer** learning
  - $X_S \neq X_T$ and/or $Y_S \neq Y_T$

Domain adaptation

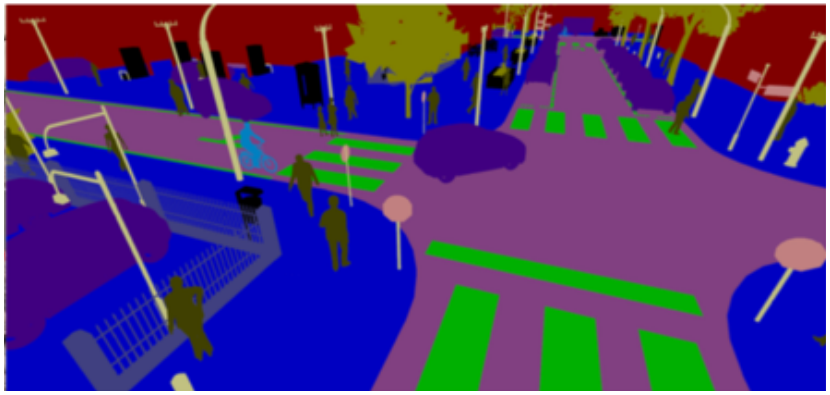
- We have **lots of synthetic data**

- How to take advantage of this to train for a **real domain**?

- Covariate shift (suppose same input size and resolution)

$\mathbf{x}_S$

y_S



Source domain (simulated images)

$\mathbf{x}_T$

$y_T = ?$

Target domain (real images)

Covariate shift: illustrations

Source image (GTA5)

Target image (CityScapes)

- Training a **self-driving system** on synthetic data
  - Pixel accuracy on **source**: 93.0%
  - Pixel accuracy on **target**: 54.0%

Source images (SVHN)

Target images (MNIST)

- Training a **character recognition** system
  - Pixel accuracy on **target**: 67.1% (**source**: 90.4%)

Domain adaptation

- **Covariate** shift
  - We assume $X_S = X_T$ (same **input** space)

Training data

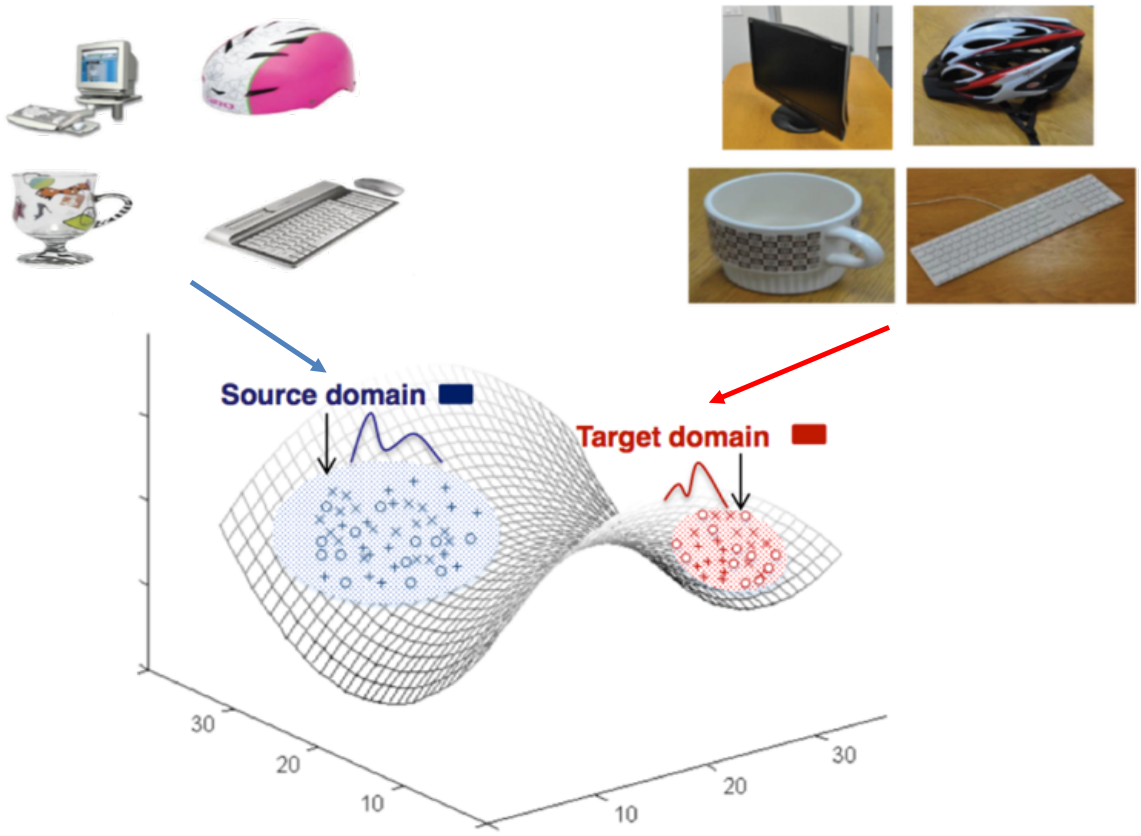
Source domain

Test data

Target domain

Covariate shift

- Difference in the P_X distribution between source and target domains: $\mathbf{P}_X^S \neq \mathbf{P}_X^T$



Types of Domain Adaptation

- **Semi-supervised** DA (SSDA)
  - **Some labeled** target data, but not enough to train from it
  - Lots of unlabeled data
- **Unsupervised** DA (UDA)
  - **No labeled** target data
- **Source-free** DA (SFDA)
  - **No source data** (e.g. because of privacy concerns)
  - Only the **source hypothesis** h_S
  - And a few labeled target data

How to approach the problem

?

How to approach the problem

- **Very active** research area
  - Because of the **numerous** applications

- **Lots** of (heuristical) approaches

(Some) families of approaches

I. **Change** the source (alternatively the target) distribution

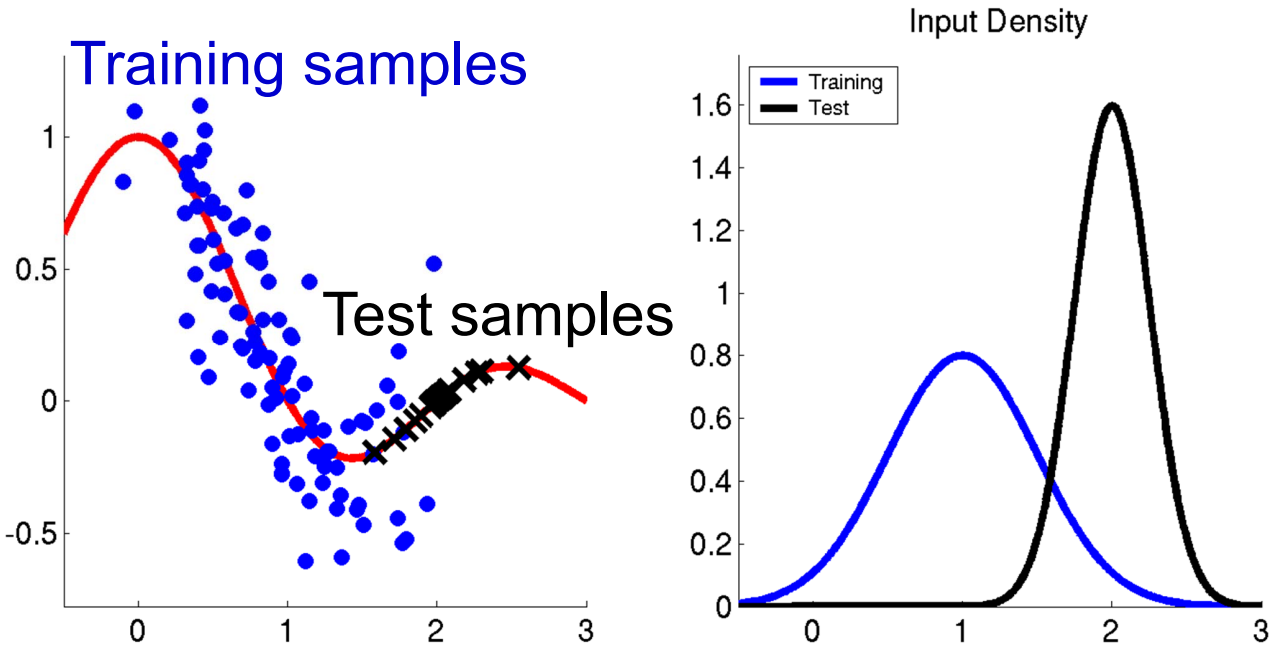
1. **Reweight** the source data
2. Iteratively **self-label** the target data, and retrain

II. Search for a **common description** space

- Where the **source hypothesis works well** on the projected **source data**
- And **hope** that it will work as well on the projected **target data**

I- DA by reweighting source data

- Here, a **regression** task



First analysis

$$R_{P_T}(h) = \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t, y^t) \sim P_T} \frac{P_S(\mathbf{x}^t, y^t)}{P_S(\mathbf{x}^t, y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \frac{P_S(\mathbf{x}^t,y^t)}{P_S(\mathbf{x}^t,y^t)}\mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \sum_{(\mathbf{x}^t,y^t)} P_T(\mathbf{x}^t,y^t)\frac{P_S(\mathbf{x}^t,y^t)}{P_S(\mathbf{x}^t,y^t)}\mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

$$
\begin{aligned}
R_{P_T}(h) &= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_T} \frac{P_S(\mathbf{x}^t,y^t)}{P_S(\mathbf{x}^t,y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \sum_{(\mathbf{x}^t,y^t)} P_T(\mathbf{x}^t,y^t) \frac{P_S(\mathbf{x}^t,y^t)}{P_S(\mathbf{x}^t,y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right] \\
&= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{P_T(\mathbf{x}^t,y^t)}{P_S(\mathbf{x}^t,y^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]
\end{aligned}
$$

First analysis

Covariate shift [Shimodaira,'00]

$\Rightarrow$ Assume similar tasks, $P_S(y|\mathbf{x}) = P_T(y|\mathbf{x})$, then:

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{D_T(\mathbf{x}^t)P_T(y^t|\mathbf{x}^t)}{D_S(\mathbf{x}^t)P_S(y^t|\mathbf{x}^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$$= \mathop{\mathbf{E}}_{(\mathbf{x}^t)\sim D_S} \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)} \mathop{\mathbf{E}}_{y^t\sim P_S(y^t|\mathbf{x}^t)} \} \mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

$\Rightarrow$ **weighted error** on the source domain: $\omega(x^t) = \frac{D_T(\mathbf{x}^t)}{D_S(\mathbf{x}^t)}$

Idea reweight labeled source data according to an estimate of $\omega(x^t)$:

$$\mathop{\mathbf{E}}_{(\mathbf{x}^t,y^t)\sim P_S} \omega(\mathbf{x}^t)\mathbf{I}\left[h(\mathbf{x}^t) \neq y^t\right]$$

Principle

- Law of large numbers
  - Sample averages converge to the population mean

$$\frac{1}{n}\sum_{i=1}^{n} A(x_i) \xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int A(x)\,\mathbf{p}_{train}(x)\,dx$$

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)} A(x_i) \xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int \frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)} A(x)\,\mathbf{p}_{train}(x)\,dx$$

$$\xrightarrow[n\to\infty]{x_i \overset{i.i.d.}{\sim} \mathbf{p}_{train}(x)} \int A(x)\,\mathbf{p}_{test}(x)\,dx$$

  - But how to estimate $\frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}$?

I- Importance weighting

- A naïve estimation of $\frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}$ does not work
  - Estimation density is too crude in high dimension space (and with few known testing instances)

- Idea of Sugiyama:
  - **Learn a parametric model** of $w(\mathbf{x}) = \frac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}$

$$\hat{w}(\mathbf{x}) = \sum_{j=1}^{J} \theta_j \, \phi_j(\mathbf{x}) \quad \text{and} \quad \hat{\mathbf{p}}_{test}(\mathbf{x}) = \hat{w}(\mathbf{x}) \, \mathbf{p}_{train}(\mathbf{x})$$

See [Sugiyama, Masashi, et al. "Direct importance estimation with model selection and its application to covariate shift adaptation." *Advances in neural information processing systems* 20 (2007)]

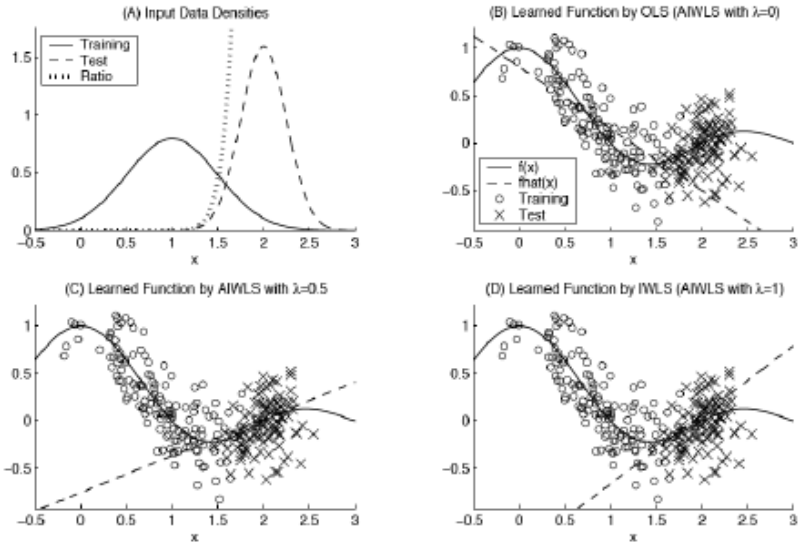
I- Covariate shift in regression

"Importance weighted" inductive criterion

Principle : weighting the classical ERM

$$R_{Cov}(h) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\mathbf{P}_{\mathcal{X}'}(\boldsymbol{x}_i)}{\mathbf{P}_{\mathcal{X}}(\boldsymbol{x}_i)}\right)^{\lambda}(h(\boldsymbol{x}_i - y_i)^2$$

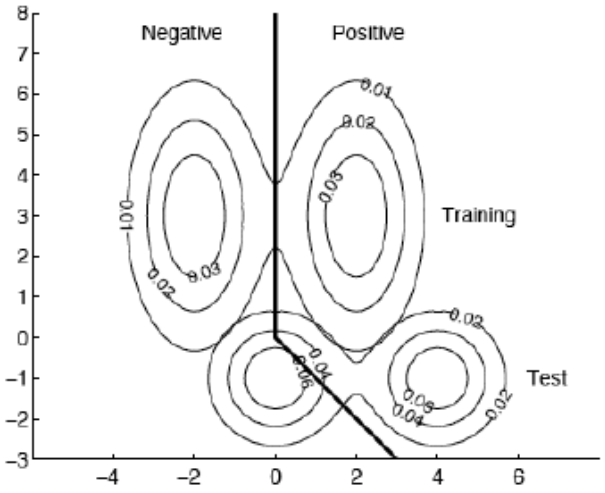
λ controls the stability / consistency (absence of bias)



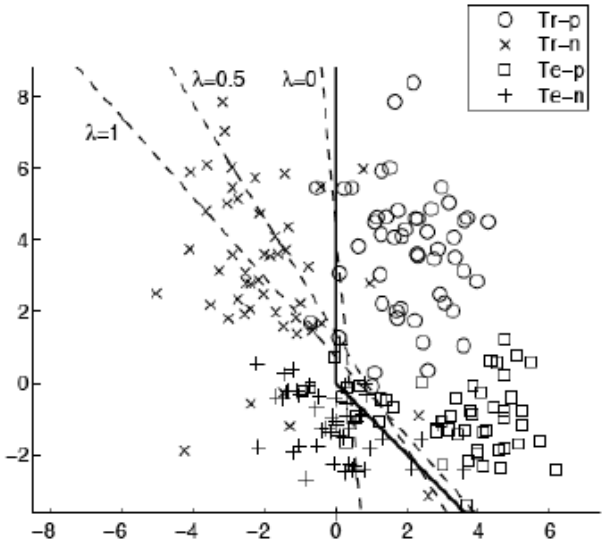
SKM07 M. Sugiyama and M. Kraudelat and K.-R. Müller (2007) *"Covariate Shift Adaptation by Importance Weighted Cross Validation"* Journal of Machine Learning Research, vol.8: 985-1005.

I- Covariate shift in classification

"Importance weighted" inductive criterion (classification task)



(a) Contours of training and test input densities.



(b) Optimal decision boundary (solid line) and learned boundaries (dashed lines). '○' and '×' denote the positive and negative training samples, while '□' and '+' denote the positive and negative test samples. Note that the test samples are not given in the training phase; they are plotted in the figure for illustration purposes.

SKM07 M. Sugiyama and M. Kraudelat and K.-R. Müller (2007) *"Covariate Shift Adaptation by Importance Weighted Cross Validation"* Journal of Machine Learning Research, vol.8: 985-1005.

I- An **alternative** reweighting approach

- **Directly learn** the parameters β_i of a linear hypothesis Φ

Source examples **Target** examples

$$\min_{\boldsymbol{\beta}} \left\| \frac{1}{n} \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_s^i) - \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{x}_t^i) \right\|^2$$

MMD

$$\text{s.t.} \quad \beta_i \in [0, B] \ , \ \forall \ 1 \leq i \leq n$$

Bound on the weights

$$\left| \sum_{i=1}^{n} \beta_i - n \right| \leq n\epsilon$$

Encourage the weights to define a probability distribution

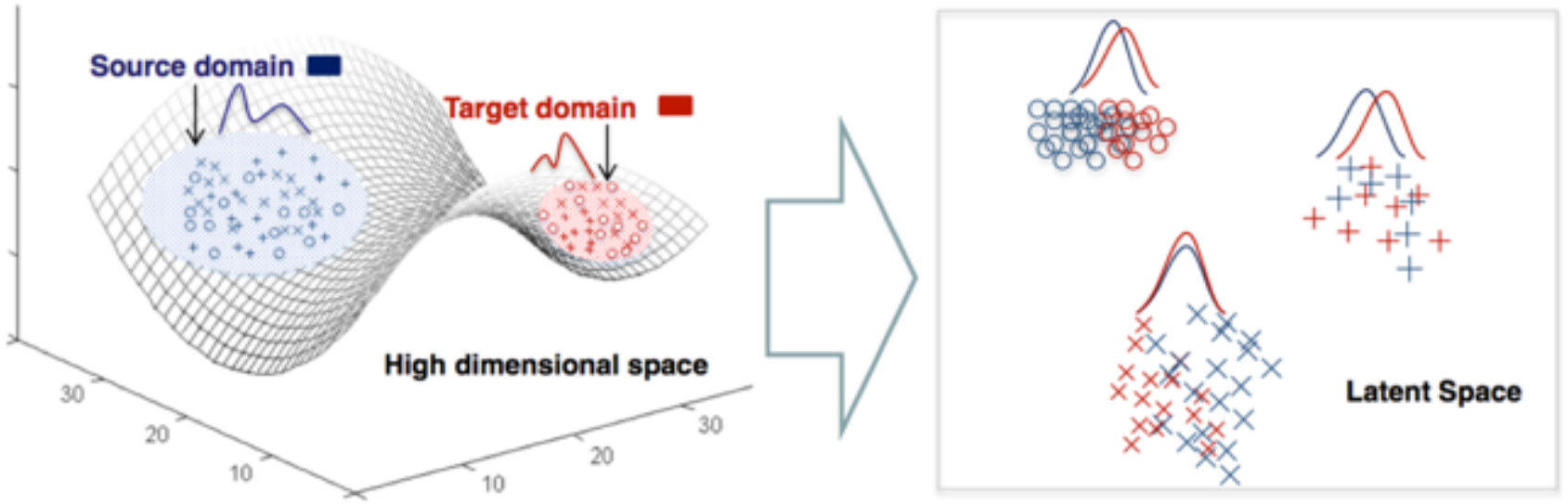
Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). **A kernel two-sample test**. *The Journal of Machine Learning Research*, *13*(1), 723-773.

I- The reweighting approach: **assessment**

- Necessitates
  - **Same** Domain
  - That probability **distributions** can be **compared**
  - **Complex** approach
  - **Not easy** to implement

II- Search for a **common description** space

- The idea



- The hope
  - If the **source hypothesis works well** on the projected **source data**
  - Then (?) it should/could work as well on the projected **target data**

II- Search for a **common description** space

- **Feature-level unsupervised** domain adaptation methods address Domain Adaptation by **aligning the features** extracted from the network across the **source** (e.g. synthetic) and **target** (e.g. real) domains, without any labeled target samples.

- **Alignment** typically involves **minimizing** some measure of **distance** between the **source** and **target** feature distributions, such as
  - maximum mean discrepancy (MMD) (Long & Wang, 2015),
  - correlation distance (Sun & Saenko, 2016),
  - or adversarial discriminator accuracy (Ganin & Lempitsky, 2015; Tzeng et al., 2017).

II- Illustration by two algorithms

… among MANy others

1. Subspace alignment

2. Deep NNs: *adversarial domain adaptation*

II- (1) Subspace alignment algorithm

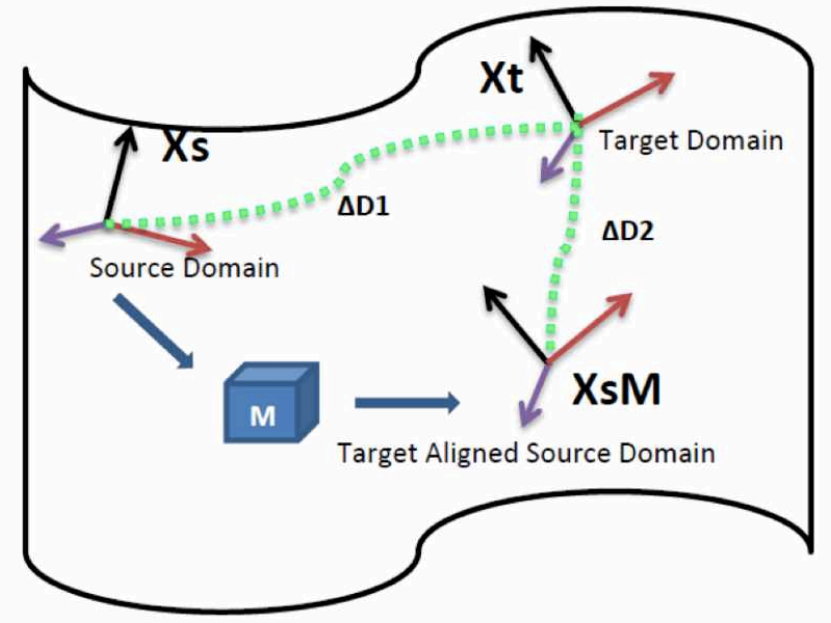
- Optimizing a (linear) mapping function that **transforms** the source subspace into the target one
  - Assumption: both source and target input spaces are D-dimensional

1. Transform every source and target **data** in the form of a D-dimensional z-normalized vector (i.e. of zero mean and unit standard deviation)
2. Using **PCA**, select for each domain *d* eigenvectors (corresponding to the largest eigenvalues)
3. These eigenvectors are used as **bases** of the source and target subspaces, respectively denoted by X_S and X_T (X_S, $X_T \in \mathrm{R}^{D \times d}$).
4. Realize the subspaces **alignment**

- **Alignment** of the basis vectors using a transformation matrix **M** from X_S to X_T

$$F(M) = ||X_S M - X_T||_F^2$$ Frobenius norm

$$M^\star = \underset{M}{\text{Argmin}}\{F(M)\}$$



Algorithm 1: Subspace alignment DA algorithm

Data: Source data S, Target data T, Source labels Y_S, Subspace dimension d
Result: Predicted target labels Y_T
$\mathbf{S}_1 \leftarrow PCA(S, d)$ (source subspace defined by the first d eigenvectors) ;
$\mathbf{S}_2 \leftarrow PCA(T, d)$ (target subspace defined by the first d eigenvectors);
$\mathbf{X}_a \leftarrow \mathbf{S}_1\mathbf{S}_1{}'\mathbf{S}_2$ (operator for aligning the source subspace to the target one);
$\mathbf{S}_a = S\mathbf{X}_a$ (new source data in the aligned space);
$\mathbf{T}_T = T\mathbf{S}_2$ (new target data in the aligned space);
$Y_T \leftarrow Classifier(\mathbf{S}_a, \mathbf{T}_T, Y_S)$;

- $\mathbf{M}^* = \mathbf{S}_1{}'\mathbf{S}_2$ corresponds to the "subspace alignment matrix": $\mathbf{M}^* = \text{argmin}_{\mathbf{M}} \|\mathbf{S}_1\mathbf{M} - \mathbf{S}_2\|$
- $X_a = \mathbf{S}_1\mathbf{S}_1{}'\mathbf{S}_2 = \mathbf{S}_1\mathbf{M}^*$ projects the source data to the target subspace
- A natural similarity: $\text{Sim}(\mathbf{x}_s, \mathbf{x}_t) = \mathbf{x}_s\mathbf{S}_1\mathbf{M}^*\mathbf{S}_1{}'\mathbf{x}_t' = \mathbf{x}_s\mathbf{A}\mathbf{x}_t'$

Subspace alignment: empirical results

- Adaptation from Office/Caltech-10 datasets (four domains to adapt) is used as source and one as target
- Comparisons
  - Baseline 1: projection on the source subspace
  - Baseline 2: projection on the target subspace
  - 2 related methods : GFK [Gong et al., CVPR'12] and GFS [Gopalan et al.,ICCV'11]

Subspace alignment: empirical results

Method	C→A	D→A	W→A	A→C	D→C	W→C
Baseline 1	44.3	36.8	32.9	36.8	29.6	24.9
Baseline 2	44.5	38.6	34.2	37.3	31.6	28.4
GFK	44.8	37.9	37.1	38.3	31.4	29.1
OUR	**46.1**	**42.0**	**39.3**	**39.9**	**35.0**	**31.8**

Method	A→D	C→D	W→D	A→W	C→W	D→W
Baseline 1	36.1	38.9	73.6	**42.5**	34.6	75.4
Baseline 2	32.5	35.3	73.6	37.3	34.2	80.5
GFK	37.9	36.1	74.6	39.8	34.9	79.1
OUR	**38.8**	**39.4**	**77.9**	39.6	**38.9**	**82.3**

Recognition accuracy using a SVM classifier

Remark1: **not** symmetrical!

Remark2: **not** that **impressive**!

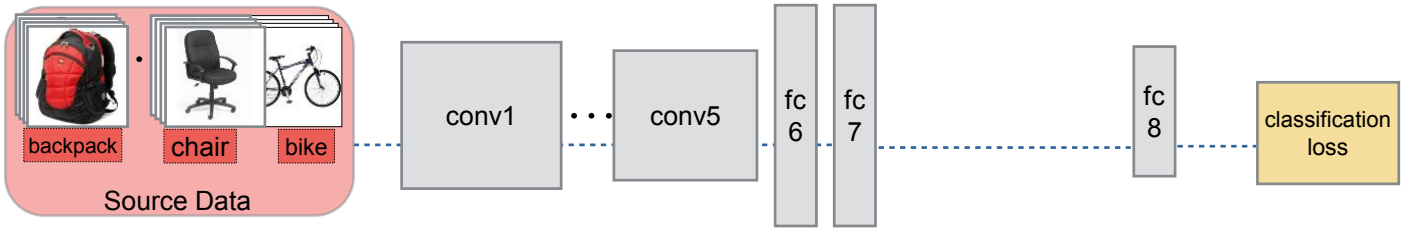
[Fernando, B., Habrard, A., Sebban, M., & Tuytelaars, T. (2013). **Unsupervised visual domain adaptation using subspace alignment**. In *Proceedings of the IEEE international conference on computer vision* (pp. 2960-2967).]

The previous approach uses **linear** transformations

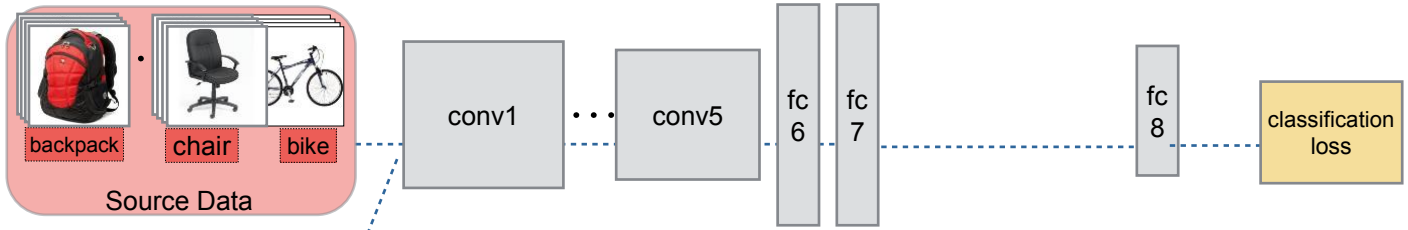
between domains

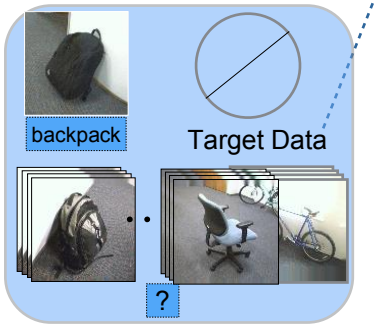
II- (2) Unsupervised Domain Adaptation with **deep NNs**

Mono-task



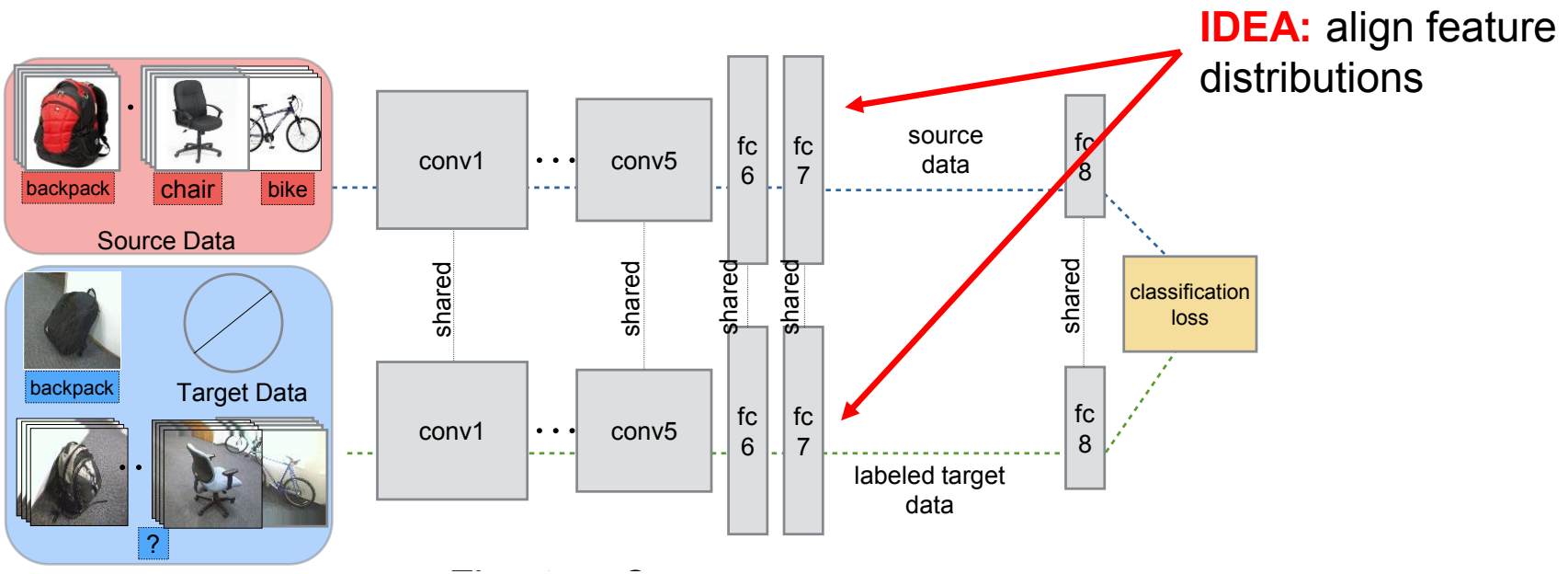
Domain adaptation





- Applying source classifier to target domain can yield inferior performance…

II- (2) Unsupervised Domain Adaptation with **deep NNs**



From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Domain adaptation

- … by domain adversarial learning
  - Source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$ with same input space $\mathcal{X}$
  - **Labeled** data on the **source** domain, and **unlabeled target** data
  - Idea: train a function that transforms the input into a **latent representation** so that:
    - this representation **does not allow to distinguish** between the **source** and **target** domains
    - but it **enables a good classification** accuracy on the **source** data

$$\min_g \max_f \frac{1}{N_S + N_T} \underbrace{\sum_{n \in \mathcal{D}_S, \mathcal{D}_T} \ell(d_n, f(\mathbf{x}_n))}_{\text{Cannot distinguish the domains}} + \underbrace{\frac{1}{N_S} \sum_{m \in \mathcal{D}_S} \ell\big(y_m, g(f(\mathbf{x}_m))\big)}_{\text{Good accuracy on the source}}$$

$f : \mathcal{X} \to \mathcal{L}$ Latent space $g : \mathcal{L} \to \mathcal{Y}$

$N_S = |S_S|$

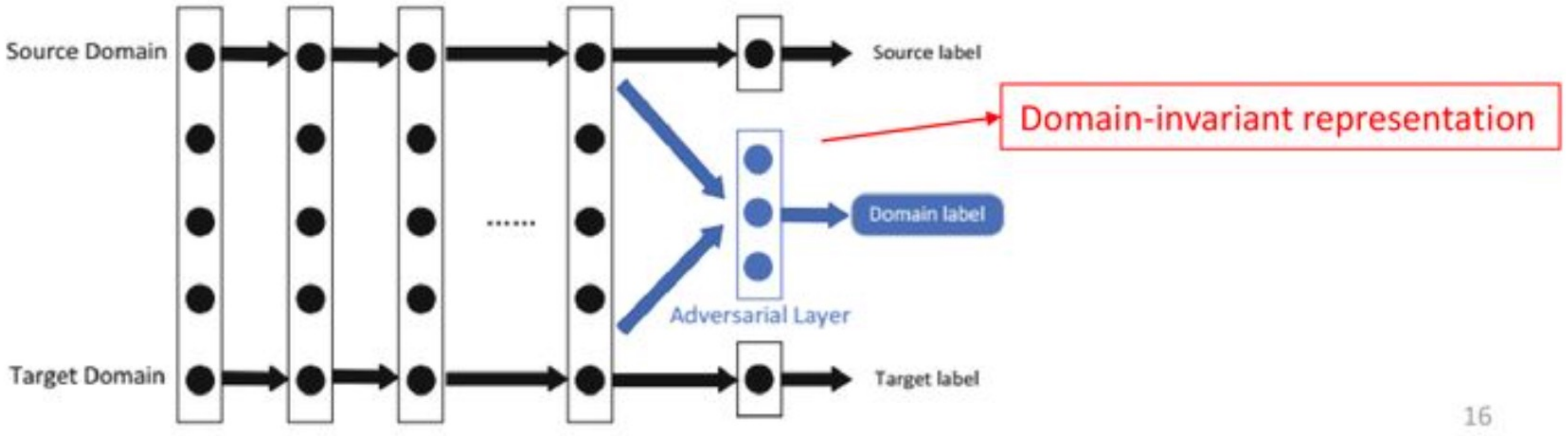
$N_T = |S_T|$

$d_n = \{s, t\}$ A label that specifies if the data example *n* comes from the source or the target

ASSUMPTION: *g* is good also on the target domain

II- (2) **Adversarial-based** approach

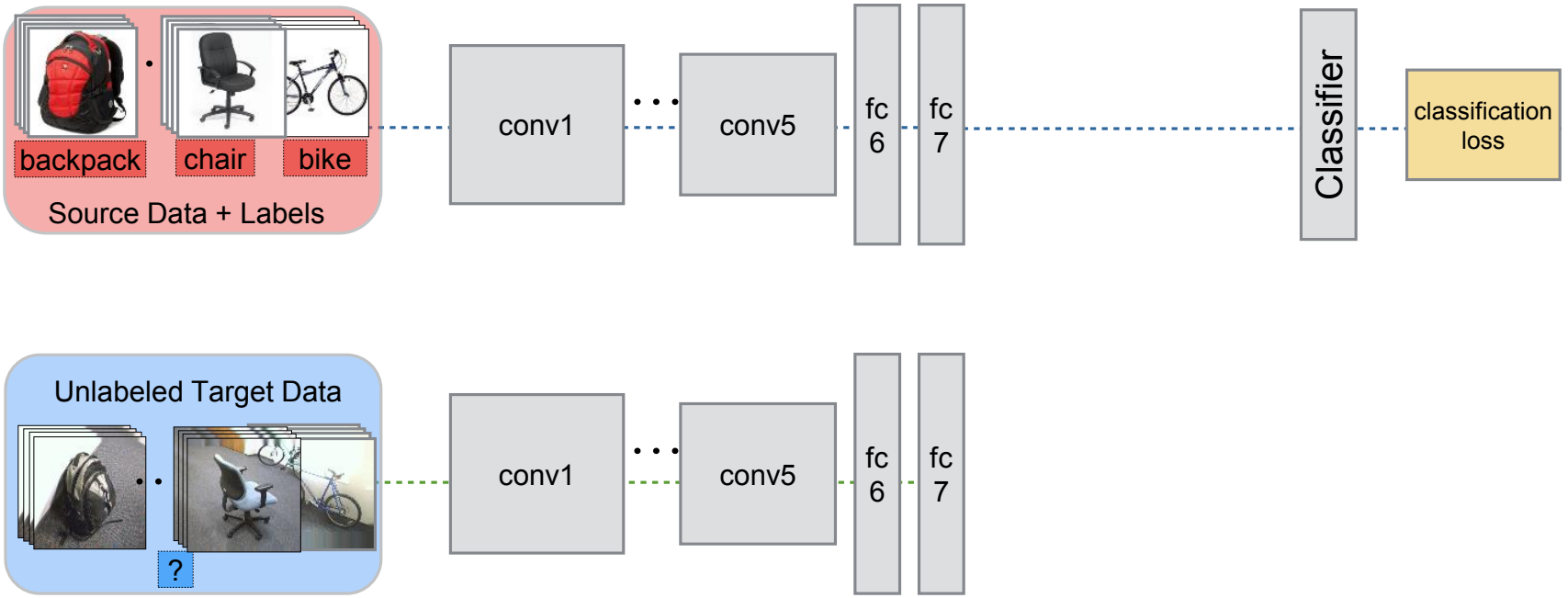
- Assumption
  - Good representation for multi-task should be **invariant between tasks**, while still discriminative for the learning task
- Approach
  1. The model **extracts features** from domains **A** and **B** and sent them to the adversarial network
  2. The adversarial network **tries to discriminate the origin** of the features
  3. When it can no longer do it, then the **features are common** to both tasks



Adversarial domain adaptation

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

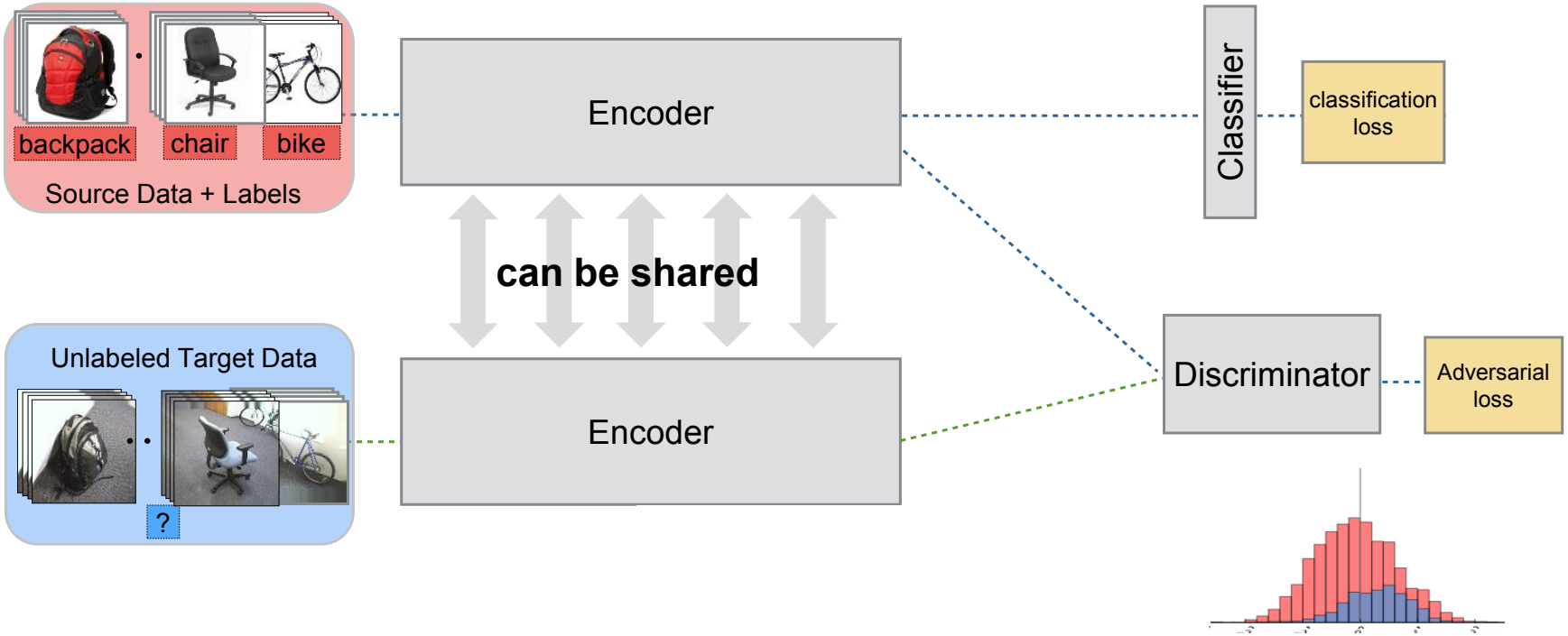
Adversarial domain adaptation



From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

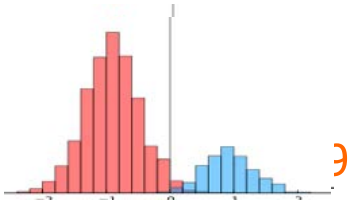
II- (2) Unsupervised Domain Adaptation with **deep NNs**

Adversarial domain adaptation



One agent tries to make the distributions look alike through the encodings

The other tries to discriminate them



From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Adversarial domain adaptation

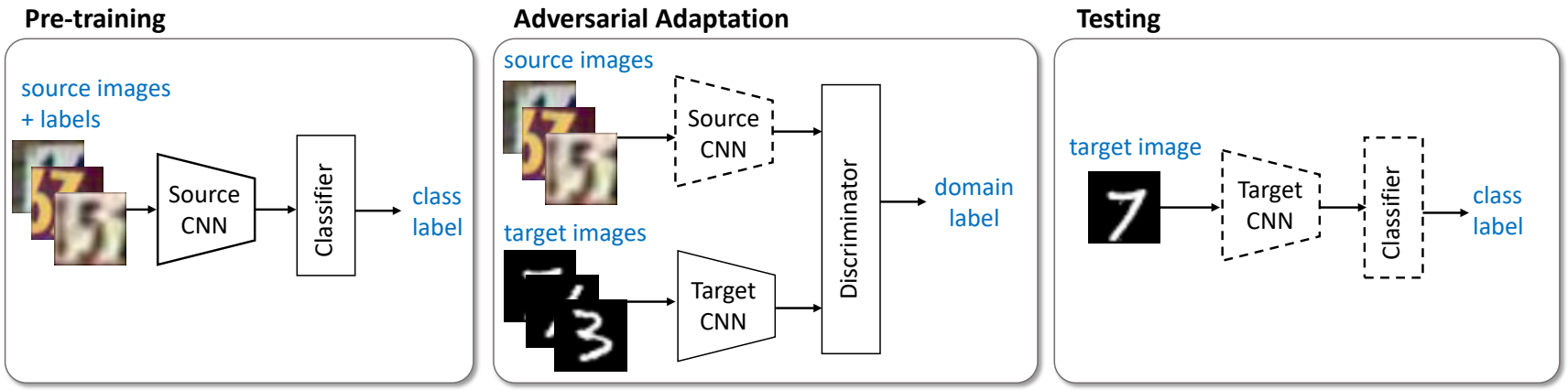


Figure 3: An overview of our proposed Adversarial Discriminative Domain Adaptation (ADDA) approach. We first pre-train a source encoder CNN using labeled source image examples. Next, we perform adversarial adaptation by learning a target encoder CNN such that a discriminator that sees encoded source and target examples cannot reliably predict their domain label. During testing, target images are mapped with the target encoder to the shared feature space and classified by the source classifier. Dashed lines indicate fixed network parameters.

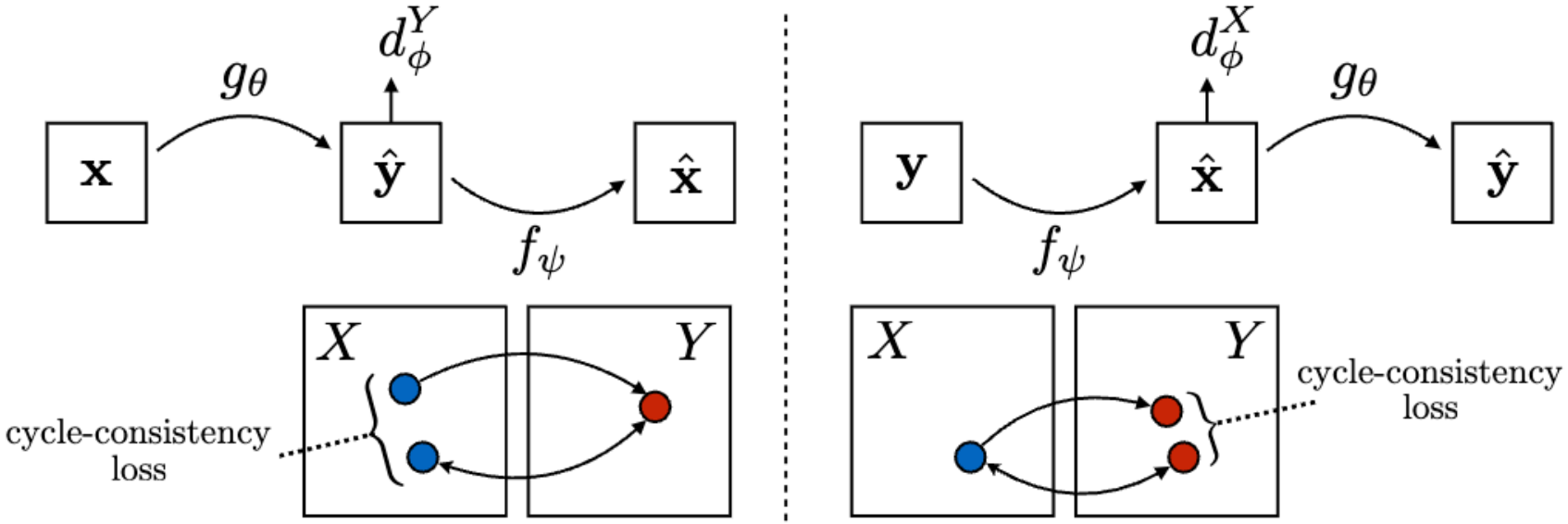
Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7167-7176).

Adversarial domain adaptation

Method	MNIST → USPS	USPS → MNIST	SVHN → MNIST
Source only	0.752 ± 0.016	0.571 ± 0.017	0.601 ± 0.011
Gradient reversal	0.771 ± 0.018	0.730 ± 0.020	0.739 [19]
Domain confusion	0.791 ± 0.005	0.665 ± 0.033	0.681 ± 0.003
CoGAN	0.912 ± 0.008	0.891 ± 0.008	did not converge
ADDA (Ours)	0.894 ± 0.002	0.901 ± 0.008	0.760 ± 0.018

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Bijective alignment by **cycle GAN**



$$\mathcal{L}_{\text{cyc}}(\mathbf{x}; g_\theta, f_\psi) = \|\mathbf{x} - f_\psi(g_\theta(\mathbf{x}))\|_1$$

$$\mathcal{L}_{\text{cyc}}(\mathbf{y}; g_\theta, f_\psi) = \|\mathbf{y} - g_\theta(f_\psi(\mathbf{y}))\|_1$$

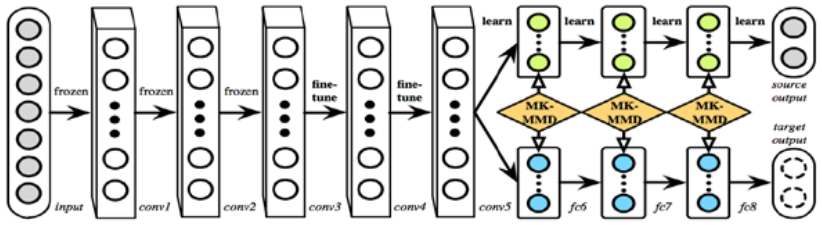
The **cycle consistency losses** encourage that the translation is a **one-to-one** function.

From [Antonio Torralba et al. **Foundations of Computer Vision**. MIT Press, 2024, p.522]

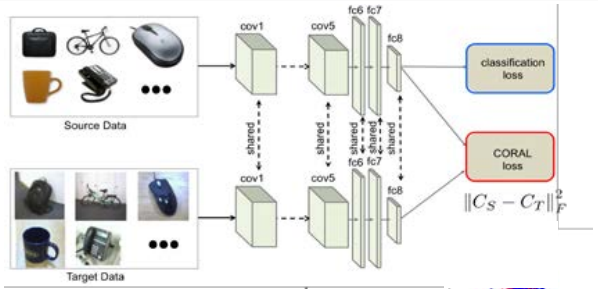
II- (2) Unsupervised Domain Adaptation with **deep NNs**

Several approaches

- by minimizing distance between distributions, e.g.

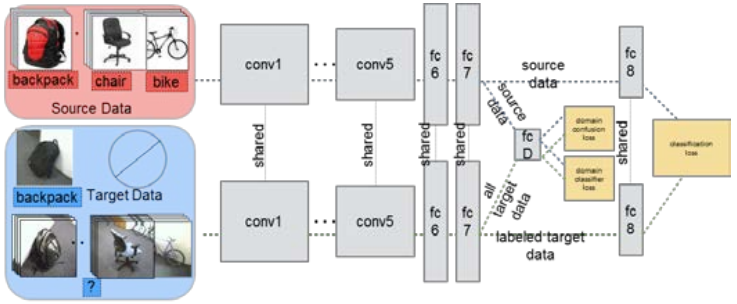


Maximum Mean Discrepancy M. Long, et al. ICML 2015

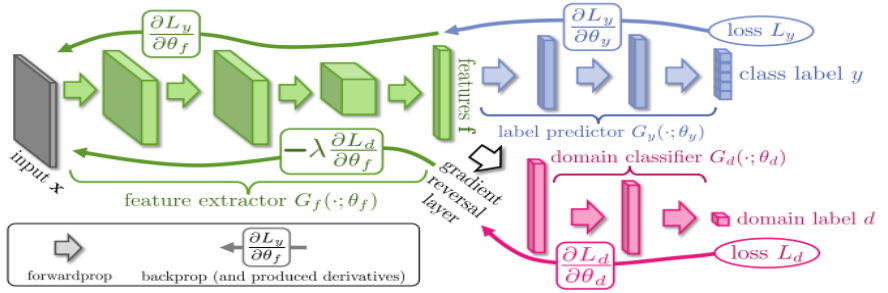


CORrelation ALignment Sun and Saenko, AAAI 2016

- …or by adversarial domain alignment, e.g.



Domain Confusion E. Tzeng et al. ICCV 2015



Reverse Gradient Y. Ganin and V. Lempitsky ICML 2015

From [https://ece.engin.umich.edu/wp-content/uploads/2019/09/4142.pdf]

Domain adaptation

We saw **two** families of approaches

1. By **reweighting** the source data to approach the target distribution
   - Delicate to put in use

2. By looking for a **common description** space
   - Linear and non linear methods
   - Source -> target ≠ Target -> source
   - Still not that impressive results

Outline

1. Designed to stem learning: adversarial examples
2. Domain adaptation (covariate shift)
3. Tracking
4. Conclusions

Tracking: an intriguing idea

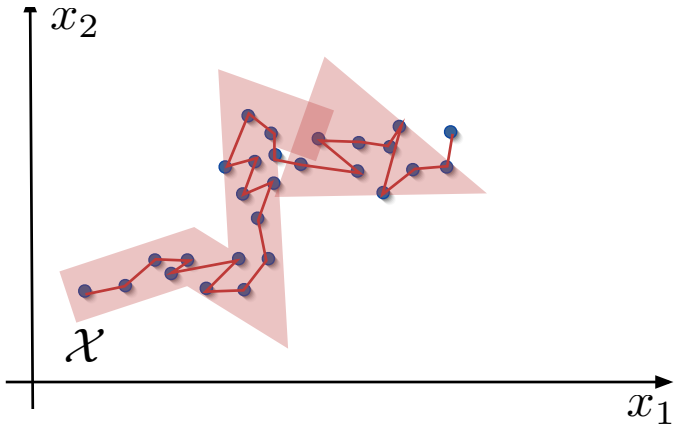
[Richard Sutton, Anna Koop & David Silver (2007). *On the role of tracking in stationary environments*. ICML-2007]

Even in stationary environments, it **can be advantageous to act as if the environment was changing**!!!

Tracking: an intriguing idea

In a lot of natural settings:

- Data comes *sequentially*
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.



This enables:

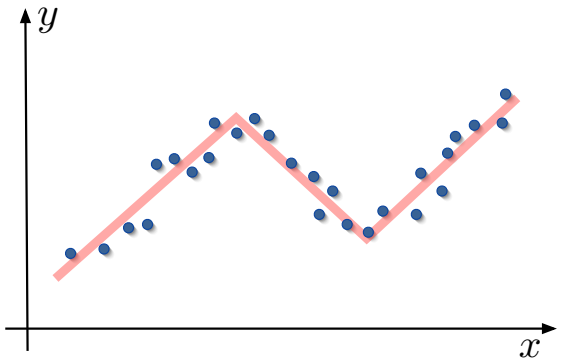
- Powerful learning
- with limited resources (time + memory)

SKS:07 R. Sutton and A. Koop and D. Silver (2007) *"On the role of tracking in stationary environments"* (ICML-07) Proceedings of the 24th international conference on Machine learning, ACM, pp.871-878, 2007.

Tracking: an intriguing idea

Assumptions:

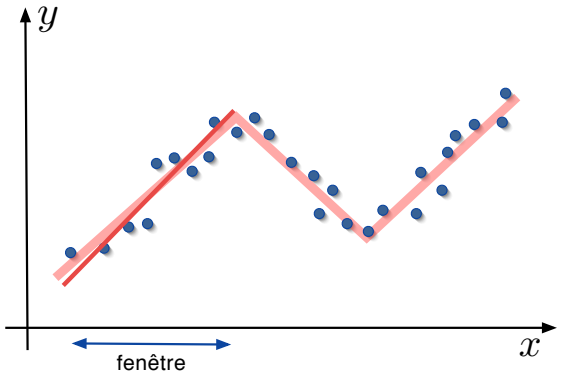
- Data streams
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.
- Limited resources: Restricted hypothesis space $\mathcal{H}$



"Local" learning

and local prediction :

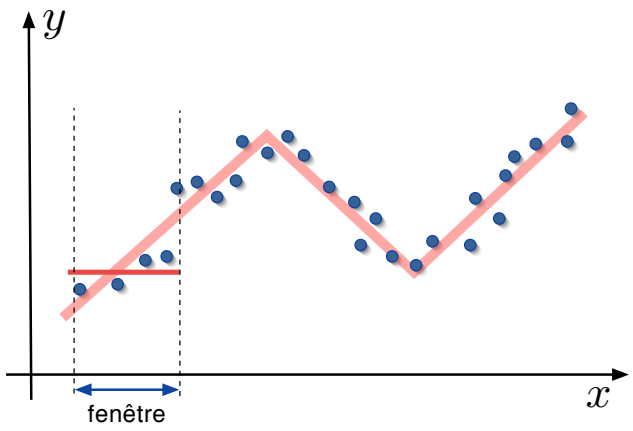
$$\begin{aligned} L_t &= \ell(h_t(\boldsymbol{x}_t), y_t) \\ &= \ell(h_t(\boldsymbol{x}_t), f(x_t, \theta_t)) \end{aligned}$$



Tracking: an intriguing idea

Assumptions:

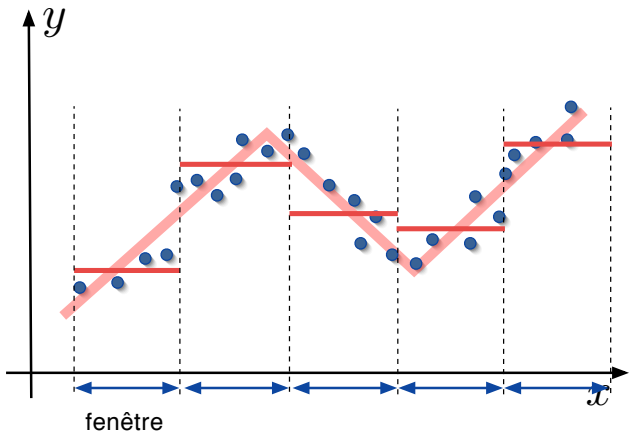
- Data streams
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.
- Limited resources: Restricted hypothesis space $\mathcal{H}$



"Local" learning

and local prediction :

$$\begin{aligned} L_t &= \ell(h_t(\boldsymbol{x}_t), y_t) \\ &= \ell(h_t(\boldsymbol{x}_t), f(x_t, \theta_t)) \end{aligned}$$



Tracking in stationary environments

- A toy environment

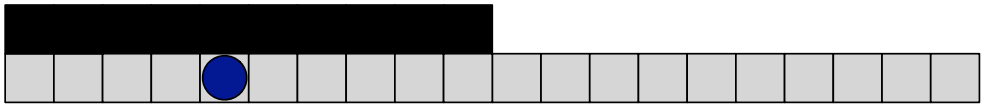


Figure 1. The Black and White world. The agent follows a random walk right and left, occasionally observing the color above it. The states wrap.

Tracking in stationary environments

- A toy environment

$$y_t = \frac{1}{1 + e^{-w_t x_t}}$$

Prediction

$$w_{t+1} = w_t + \alpha \left(z_t - y_t\right) x_t$$

Learning

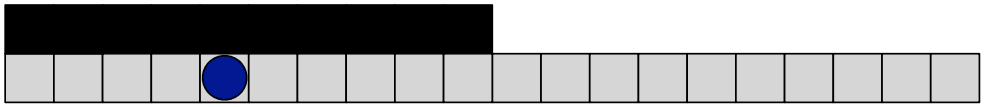


Figure 1. The Black and White world. The agent follows a random walk right and left, occasionally observing the color above it. The states wrap.

Tracking in stationary environments

- A toy environment

$$y_t = \frac{1}{1 + e^{-w_t x_t}}$$

Prediction

$$w_{t+1} = w_t + \alpha \left(z_t - y_t\right) x_t$$

Learning

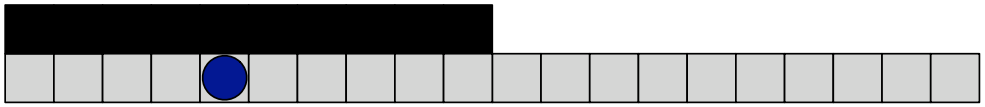


Figure 1. The Black and White world. The agent follows a random walk right and left, occasionally observing the color above it. The states wrap.

Very **simple** rule

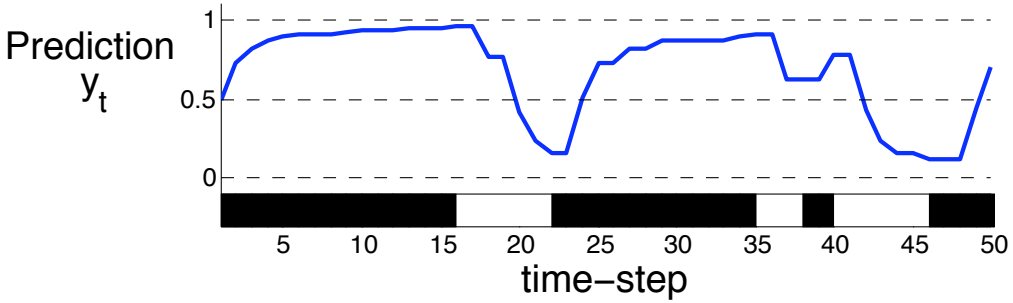


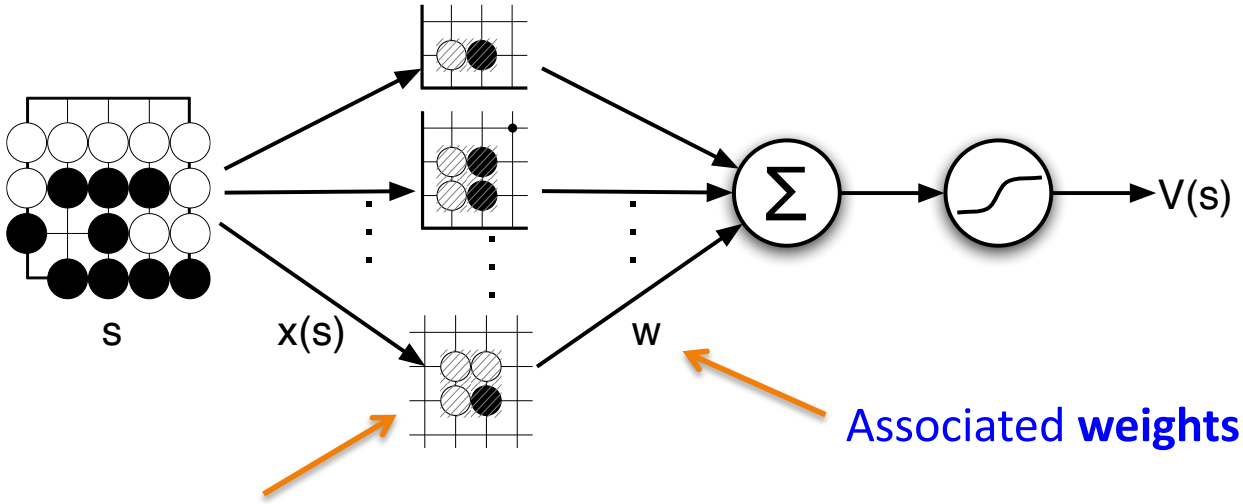
Figure 2. A sample trajectory in the Black and White world, showing the prediction on each time-step and the actual color above the agent. The prediction is modified only on time steps on which the color is observed. Here $\alpha = 2$.

Continuously adapts to the **local** environment

Tracking in stationary environments

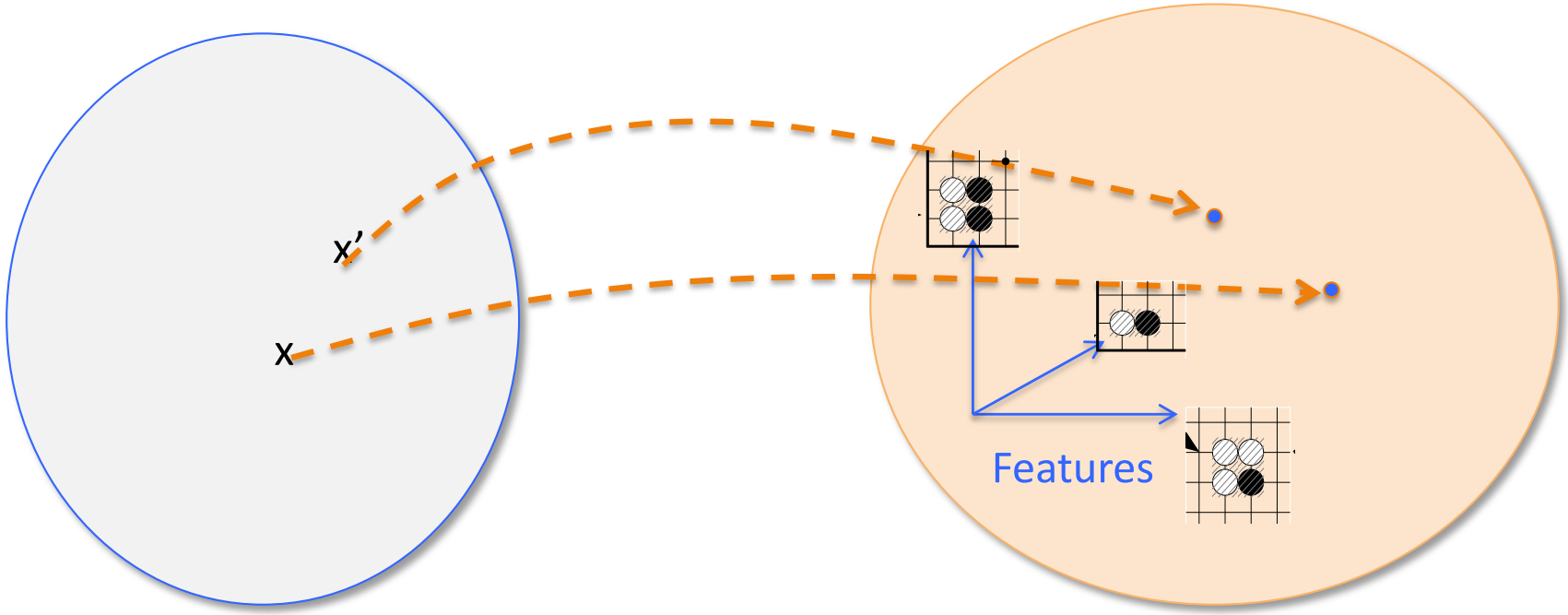
Tracking to play Go

- 5 x 5 Go
  - More than 5 x 10^{10} unique positions
- Usual approach: learn a **general** evaluation function V(*s*) valid $\forall$ s



Tracking as **local changes** of representation

...



Space of **go positions**

Embedding

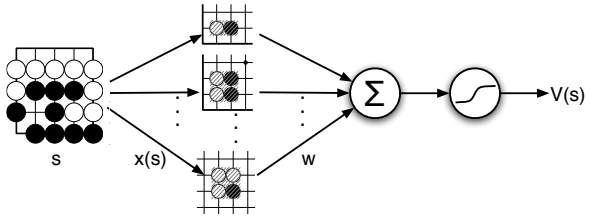
Space of representations

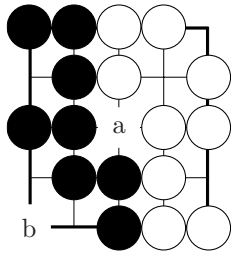
Weighted **features**

Tracking in stationary environments

- Tracking approach: learn an **evaluation** function V(*s*)

local to the current *s*

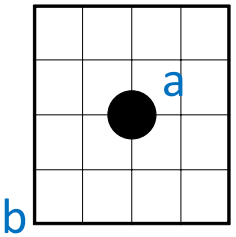




In general, playing (a) (center) is better than playing (b)

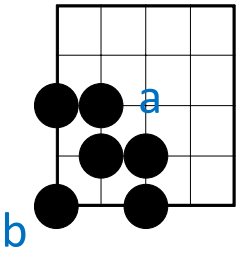
BUT

In this situation, playing (b) is better than playing (a)



More weight

More weight



Tracking in stationary environments

Tracking to play Go

Comparison:

- learn **a general evaluation function** V(s)
  - On 250,000 complete episodes of self-play
- Learn **successive evaluation functions** $V_t(s)$ attuned to the current states
  - On 10,000 episodes of self-play starting from the current position

Features	Tracking beats converging		
	Black	White	Total
1×1	82%	43%	62.5%
2×2	90%	71%	80.5%
3×3	93%	80%	86.5%

Table 1. Percentage of 5×5 Go games won by the tracking agent playing against the converging agent when playing as Black (first to move) and as White.

Tracking in stationary environments

Comparison:

- **learn a general evaluation function** V(s)
  - On 250,000 complete episodes of self-play
- Learn **successive evaluation functions** Vt(s) attuned to the current state
  - On 10,000 episodes of self-play starting from the current position

Features	Total features	CPU (minutes)	
		Tracking	Converging
1×1	75	3.5	10.1
2×2	1371	5.7	13.8
3×3	178518	9.1	22.2

Table 2. Memory and CPU requirements for tracking and converging agents. The total number of binary features indicates the memory consumption. The CPU time is the average training time required to play a complete game: 250,000 episodes of training for the converging agent; 10,000 episodes of training per move for the tracking agent.

Outline

1. Designed to stem learning: adversarial examples
2. Domain adaptation (covariate shift)
3. Tracking
4. Conclusions

1. Domain adaptation (DA) is **important** for applications (and for the development of the theory of learning)
   - By **reweighting**
   - By looking for a **common representation**

   ➢ Still lot of place for progress

2. Looking for **local models** instead of a general one can save resources and be more efficient (for an I.I.D. task)
   - **Tracking** adapts from local situations to close local situations (the *temporal consistency* condition)

     = sequence of **small DA steps**