Apprentissage par **transfert** :

État de l'art

et présentation d'une

nouvelle méthode par **boosting de traductions** entre cible et source

---

Antoine Cornuéjols

AgroParisTech – INRA MIA 518

EKINOCS research group

AgroParisTech

INRA SCIENCE & IMPACT

Outline

1. Classical inductive learning
2. Transfer learning
3. TransBoost: an original approach
4. Conclusion

Classical inductive learning

One example that tells a lot …

- Examples described using:

  Number (1 or 2); ***size*** (small or large); ***shape*** (circle or square); ***color*** (red or green)

- They belong either to class '**+**' or to class '**-**'

Description	**Your** prediction	**True** class
1 large red square		-
1 large green square		+
2 small red squares		+
2 large red circles		-
1 large green circle		+
1 small red circle		+

One example that tells a lot …

- Examples described using:

  Number (1 or 2); ***size*** (small or large); ***shape*** (circle or square); ***color*** (red or green)

Description	**Your** prediction	**True** class
1 large red square		-
1 large green square		+
2 small red squares		+
2 large red circles		-
1 large green circle		+
1 small red circle		+

How many possible functions altogether from *X* to *Y* ? $2^{2^4} = 2^{16} = 65{,}536$

How many functions do remain after 6 training examples? $2^{10} = 1024$

One example that tells a lot …

- Examples described using:

  Number (1 or 2); ***size*** (small or large); ***shape*** (circle or square); ***color*** (red or green)

Description	**Your** prediction	**True** class
1 large red square		-
1 large green square		+
2 small red squares		+
2 large red circles		-
1 large green circle		+
1 small red circle		+
1 small green square		-
1 small red square		+
2 large green squares		+
2 small green squares		+
2 small red circles		+
1 small green circle		-
2 large green circles		-
2 small green circles		+
1 large red circle		-
2 large red squares	?	

15

How many remaining functions?

?

How to **chose** an hypothesis?

Real risk: expected loss

$$R(h) \;=\; \mathbb{E}[\ell(h(\boldsymbol{x}), y)] \;=\; \int_{\boldsymbol{x}\in\mathcal{X}, y\in\mathcal{Y}} \ell(h(\boldsymbol{x}), y)\ \mathbf{P}_{\mathcal{X}\mathcal{Y}}\ d(\boldsymbol{x}, y)$$

The **statistical** theory of learning

Real risk: expected loss

$$R(h) \;=\; \mathbb{E}[\ell(h(\boldsymbol{x}),y)] \;=\; \int_{\boldsymbol{x}\in\mathcal{X},y\in\mathcal{Y}} \ell(h(\boldsymbol{x}),y)\;\mathbf{P}_{\mathcal{X}\mathcal{Y}}\;d(\boldsymbol{x},y)$$

But $\mathbf{P}_{\mathcal{X}\mathcal{Y}}$ is unknown, then use: $\mathcal{S}_m = \{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_m,y_m)\} \in (\mathcal{X}\times\mathcal{Y})^m$

The **statistical** theory of learning

Real risk: expected loss

$$R(h) \;=\; \mathbb{E}[\ell(h(\boldsymbol{x}),y)] \;=\; \int_{\boldsymbol{x}\in\mathcal{X},y\in\mathcal{Y}} \ell(h(\boldsymbol{x}),y)\; \mathbf{P}_{\mathcal{X}\mathcal{Y}}\; d(\boldsymbol{x},y)$$

But $\mathbf{P}_{\mathcal{X}\mathcal{Y}}$ is unknown, then use: $\mathcal{S}_m = \{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_m,y_m)\} \in (\mathcal{X}\times\mathcal{Y})^m$

Empirical risk Minimization

$$\hat{h} \;=\; \underset{h\in\mathcal{H}}{\text{ArgMin}}\big[R_m(h)\big] + \text{Reg}\big] \;=\; \underset{h\in\mathcal{H}}{\text{ArgMin}}\left[\frac{1}{m}\sum_{i=1}^{m}\ell(h(\boldsymbol{x}_i),y_i)\right] + \lambda\,\text{Capacity}(\mathcal{H})\big]$$

Statistical study for $|\mathcal{H}|$ hypotheses

It leads to:

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R(h) \; \leq \; \widehat{R}(h) \; + \; \overbrace{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m}}^{\varepsilon} \right] \; > \; 1 - \delta$$

The Empirical Risk Minimization principle

is sound **only if** there exists a limit (a bias) on the expressivity of $\mathcal{H}$

Tracking

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**!!!

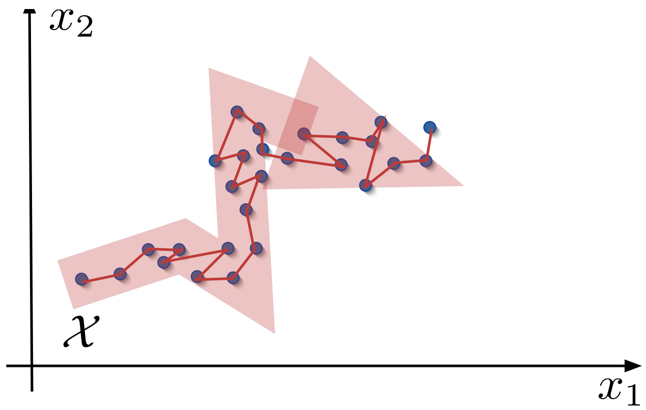
Tracking: an intriguing idea

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)

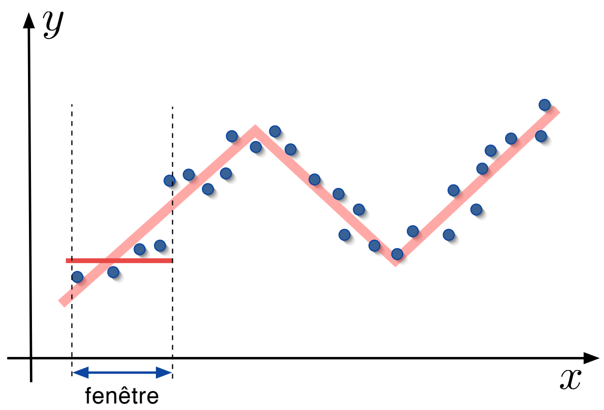


SKS:07 R. Sutton and A. Koop and D. Silver (2007) *"On the role of tracking in stationary environments"* (ICML-07) Proceedings of the 24th international conference on Machine learning, ACM, pp.871-878, 2007.

Tracking: an intriguing idea

Assumptions:

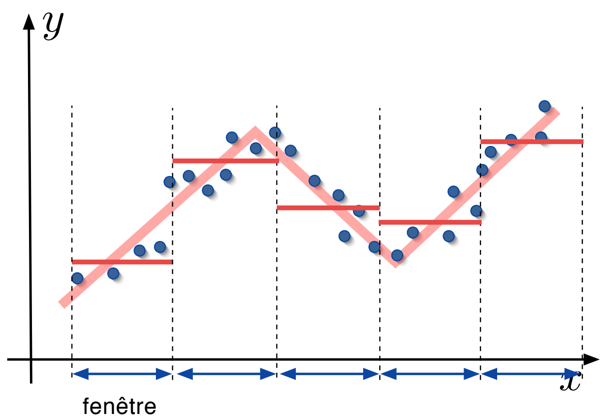
- Data streams
- *Temporal consistency*: consecutive data points come from "similar" distribution: not i.i.d.
- Limited resources: Restricted hypothesis space $\mathcal{H}$



"Local" learning

and local prediction :

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

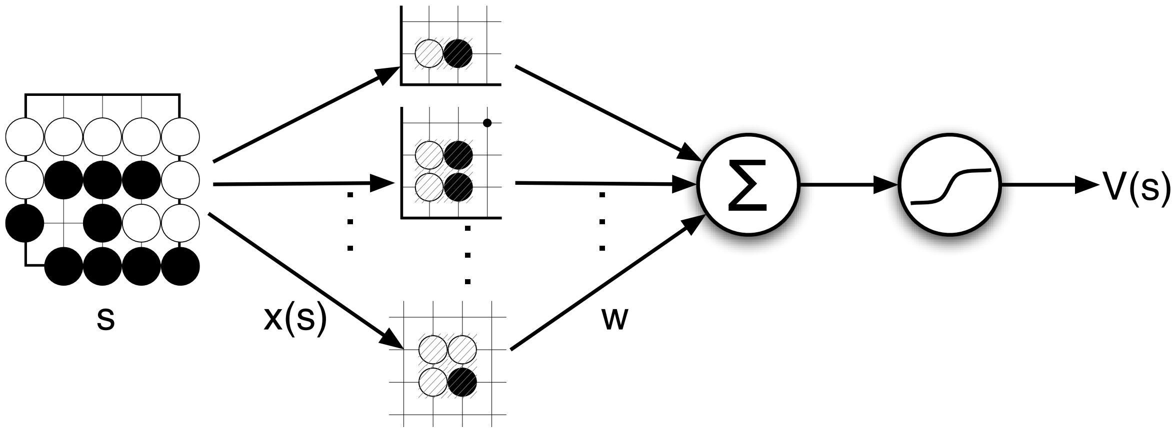


15

Tracking in stationary environments

Tracking to play Go

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



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 state
  - 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.

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 state
  - On **10,000 episodes** of self-play starting from the current position

Features	Total features	CPU (minutes) Tracking	CPU (minutes) 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.

On-line learning

- What to keep?
- What to forget?
- How to adapt?

The **plasticity** vs. **stability** dilemma

- Very little theory
  - Except against any sequence: maximalist

[Cesa-Bianchi, N. & Lugosi G. “*Prediction, learning and games*”. Cambridge University Press, 2006]

Outline

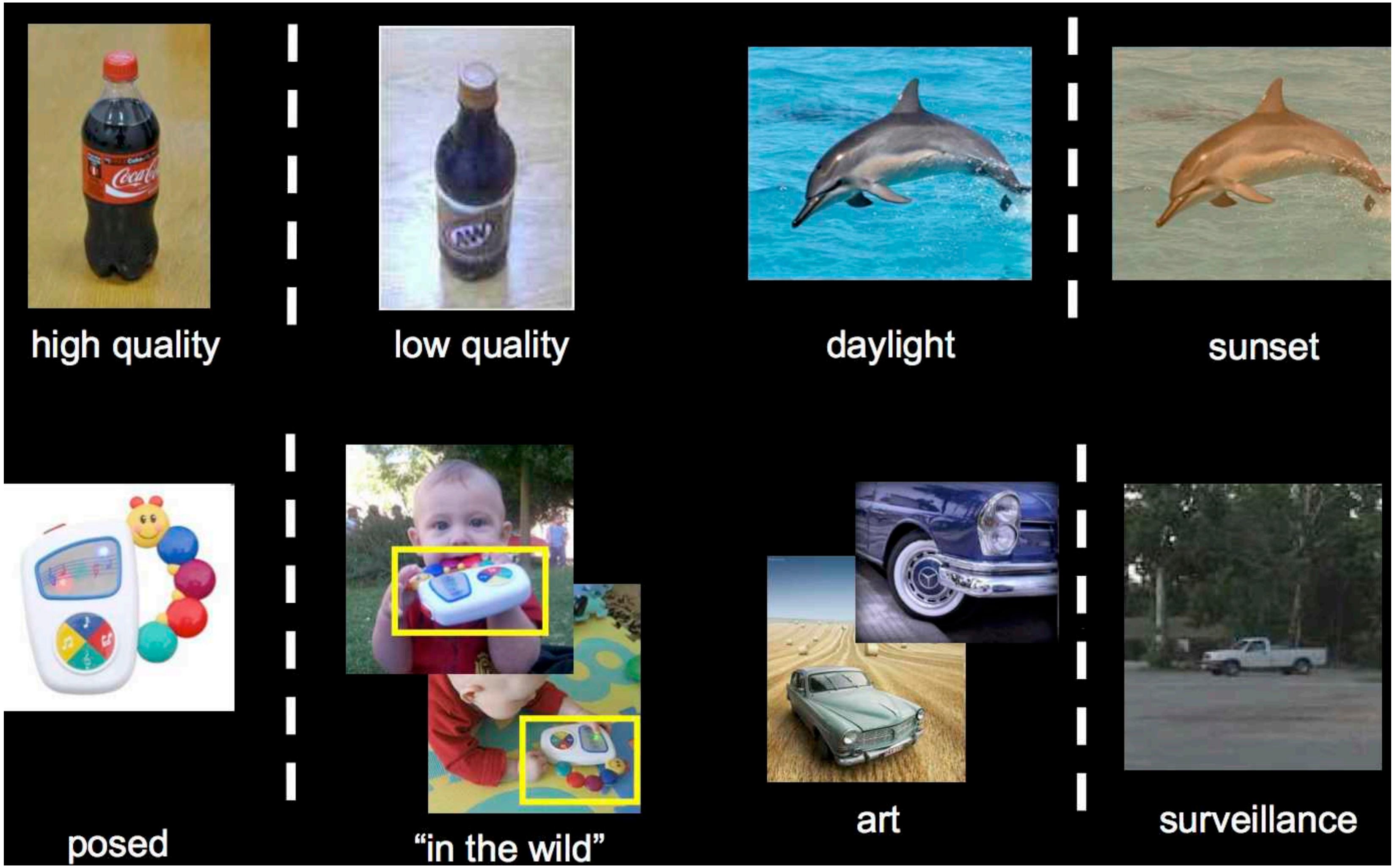
1. Classical inductive learning
2. Transfer learning
3. TransBoost: an original approach
4. Conclusion

Transfer learning

- A generalized **one step** **on-line** learning

$X_S = X_T$ & $Y_S = Y_T$	$X_S \neq X_T$ & $Y_S = Y_T$
$X_S = X_T$ & $Y_S \neq Y_T$	$X_S \neq X_T$ & $Y_S \neq Y_T$

Examples: transfer learning in vision



[Xu, Saenko, Tsang “Domain Transfer” tutorial – CVPR’12]

Domain adaptation for sentiment analysis

	Electronics	Video games
✓	(1) Compact; easy to operate; very good picture quality; looks sharp!	(2) A very good game! It is action packed and full of excitement. I am very much hooked on this game.
✓	(3) I purchased this unit from Circuit City and I was very excited about the quality of the picture. It is really nice and sharp.	(4) Very realistic shooting action and good plots. We played this and were hooked.
✗	(5) It is also quite blurry in very dark settings. I will never_buy HP again.	(6) It is so boring. I am extremely unhappy and will probably never_buy UbiSoft again.

- Source specific: *compact, sharp, blurry*.
- Target specific: *hooked, realistic, boring*.
- Domain independent: *good, excited, nice*, *never_buy, unhappy*.

[Pan, TL-IJCAI'13 tutorial]

Domain adaptation

Objective

– Improve a **target prediction function** in the target domain using knowledge from the **source domain**

1. The **training** and **test set** can be from the **same domain**, but with different probability distributions (“Domain adaptation”)
   – Co-variate shift
   – Concept drift
2. Or they can be from **different domains**
   – Transfer learning

Notations

1. **Source** domain *S*
   - Source **training data** S_S
   - Source data **distribution** D_S
   - Source **hypothesis** h_S

2. **Target** domain *T*
   - Target **training data** S_T $(|S_T| << |S_S|)$
   - Target **data distribution** D_T
   - Target **hypothesis** h_T

Formalisation

- **Target** domain: $\mathcal{X}_\mathcal{T} \times \mathcal{Y}_\mathcal{T}$
  - Training set $S_\mathcal{T} = \{(\mathbf{x}_i^\mathcal{T}, y_i^\mathcal{T})\}_{1 \leq i \leq m}$
  - Distribution $\mathbf{P}_{\mathcal{X}\mathcal{Y}}^T$

- Source domain: $\mathcal{X}_\mathcal{S} \times \mathcal{Y}_\mathcal{S}$
  - Training set $S_\mathcal{S} = \{(\mathbf{x}_i^\mathcal{S}, y_i^\mathcal{S})\}_{1 \leq i \leq m}$
  - Source hypothesis $h_\mathcal{S}$

- We look for: $h_\mathcal{T} : \mathcal{X}_\mathcal{T} \rightarrow \mathcal{Y}_\mathcal{T}$

- **Algorithm** $A^{\text{htl}} : (\mathcal{X}_\mathcal{T} \times \mathcal{Y}_\mathcal{T})^m \times \mathcal{H}_\mathcal{S} \rightarrow \mathcal{H}_\mathcal{T} \subseteq \mathcal{Y}_\mathcal{X}$

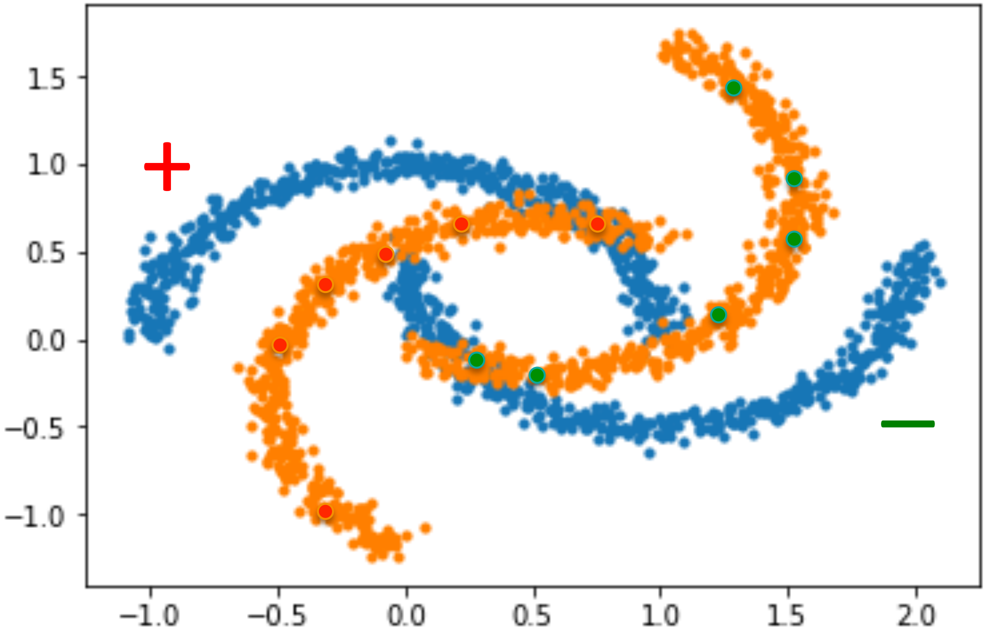
  Hypothesis TL

Types of transfer learning (1)

- **Inductive** transfer learning
  - Labeled target training data
- **Hypothesis** transfer learning
  - Inductive transfer learning
  - The **source hypothesis is known**, not the **source training data**
- **Unsupervised** domain adaptation
  - Only unlabeled target data

Illustration

"half moon" problem

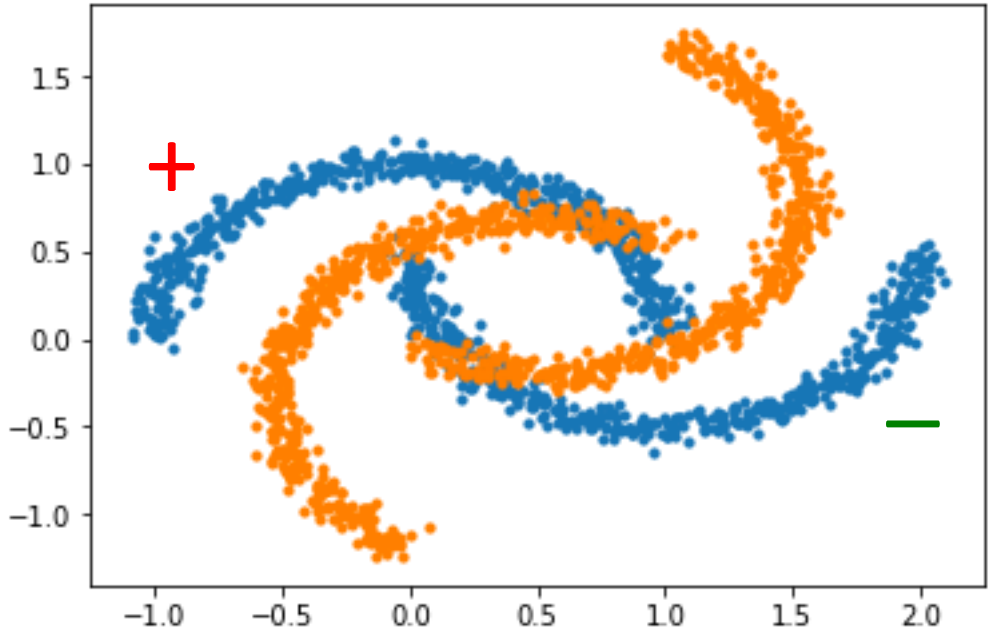


Supervised

Inductive Transfer Learning

Illustration

"half moon" problem



Unsupervised

Domain Adaptation

Examples of

Transfer learning

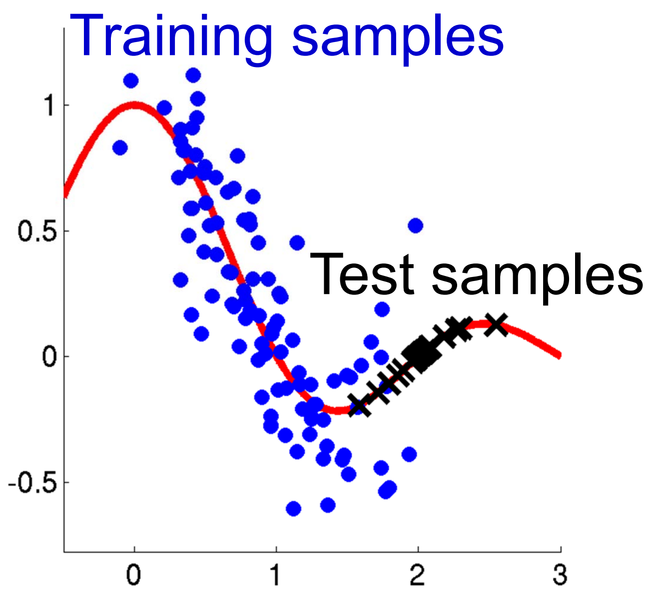
Domain Adaptation

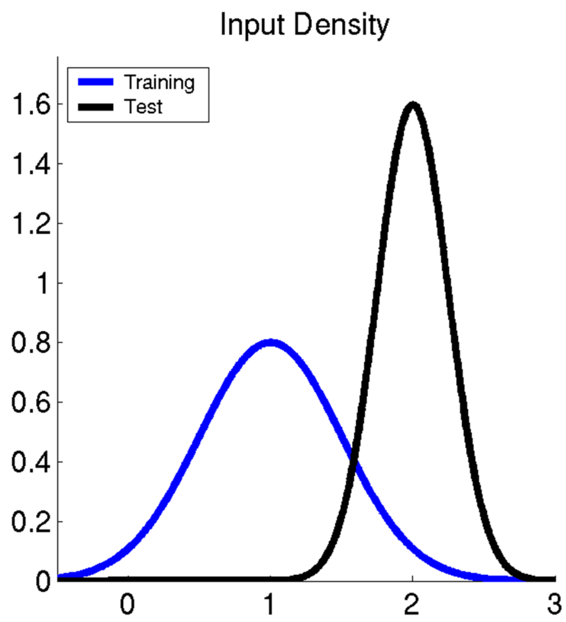
Covariate shift

$X_S = X_T$ & $Y_S = Y_T$	$X_S \neq X_T$ & $Y_S = Y_T$
$X_S = X_T$ & $Y_S \neq Y_T$	$X_S \neq X_T$ & $Y_S \neq Y_T$

- **Input distribution** changes $$P_{train}(\mathbf{x}) \neq P_{test}(\mathbf{x})$$

- **Functional relation** remains unchanged $$P_{train}(y|\mathbf{x}) = P_{test}(y|\mathbf{x})$$





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 $\boxed{\dfrac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}}$ **?**

Importance weighting

- A naïve estimation of $\boxed{\dfrac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}}$ 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 $w(\mathbf{x}) = \dfrac{\mathbf{p}_{test}(x)}{\mathbf{p}_{train}(x)}$

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

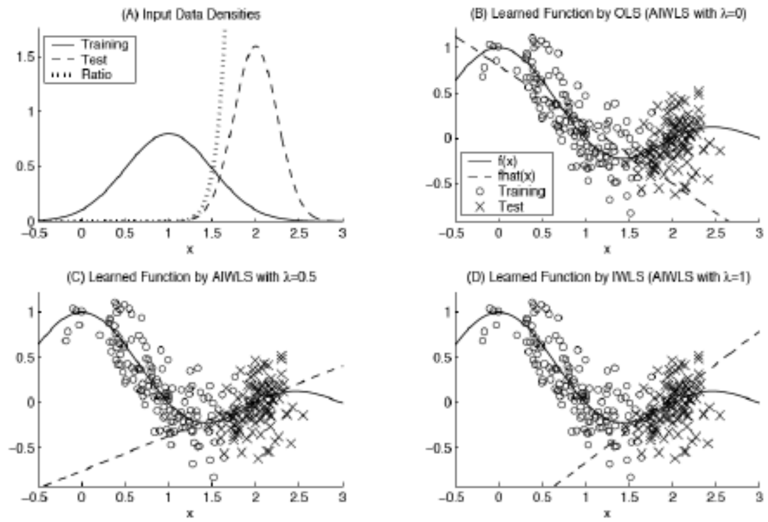
Covariate shift in regression

"Importance weighted" inductive criterion

Principle : weighting the classical ERM

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

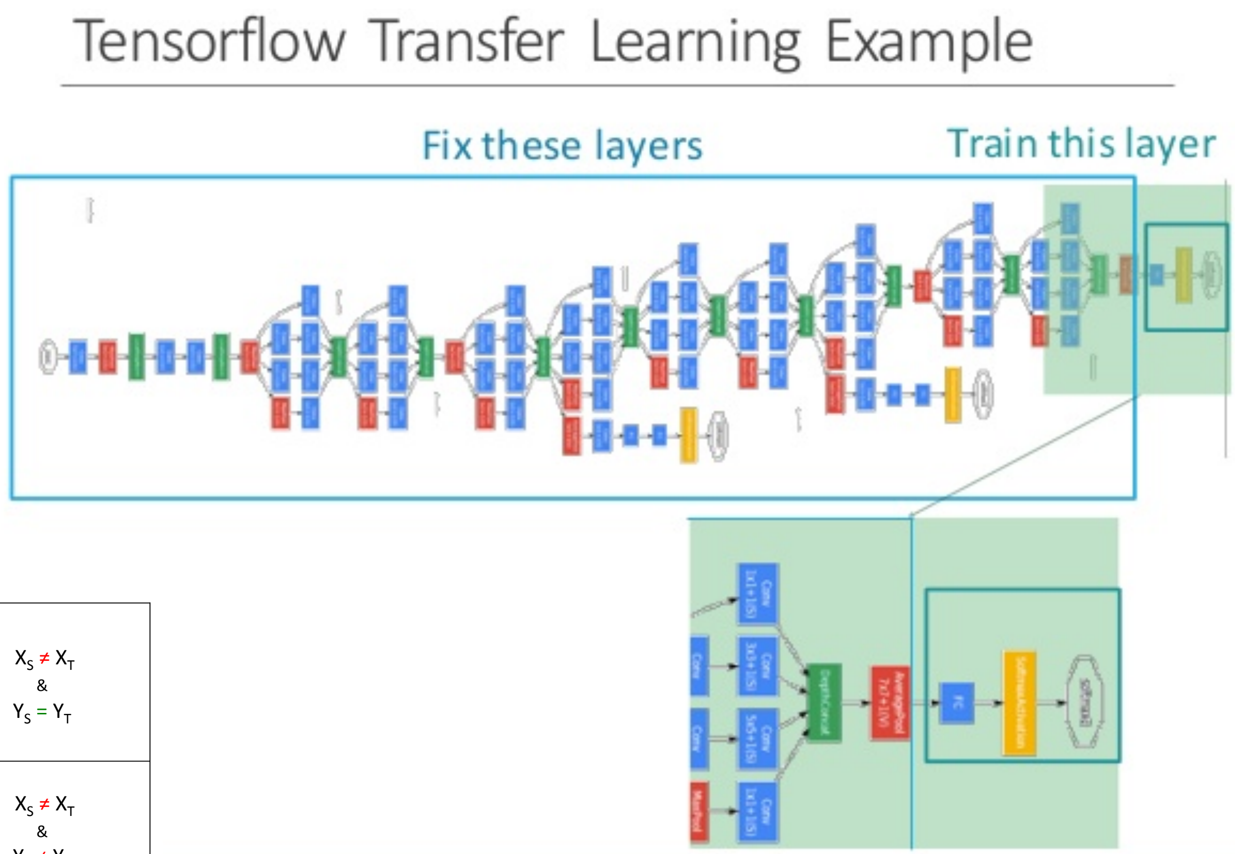
λ controls the stability / consistency (absence of bias)



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.

Hypothesis Transfer Learning

Transfer learning for deep neural networks



$X_S = X_T$ & $Y_S = Y_T$	$X_S \neq X_T$ & $Y_S = Y_T$
$X_S = X_T$ & $Y_S \neq Y_T$	$X_S \neq X_T$ & $Y_S \neq Y_T$

[Yosinski J, Clune J, Bengio Y, and Lipson H. ***How transferable are features in deep neural networks?*** In Advances in Neural Information Processing Systems 27 (NIPS '14), NIPS Foundation, 2014.]

Transfer learning for deep neural networks

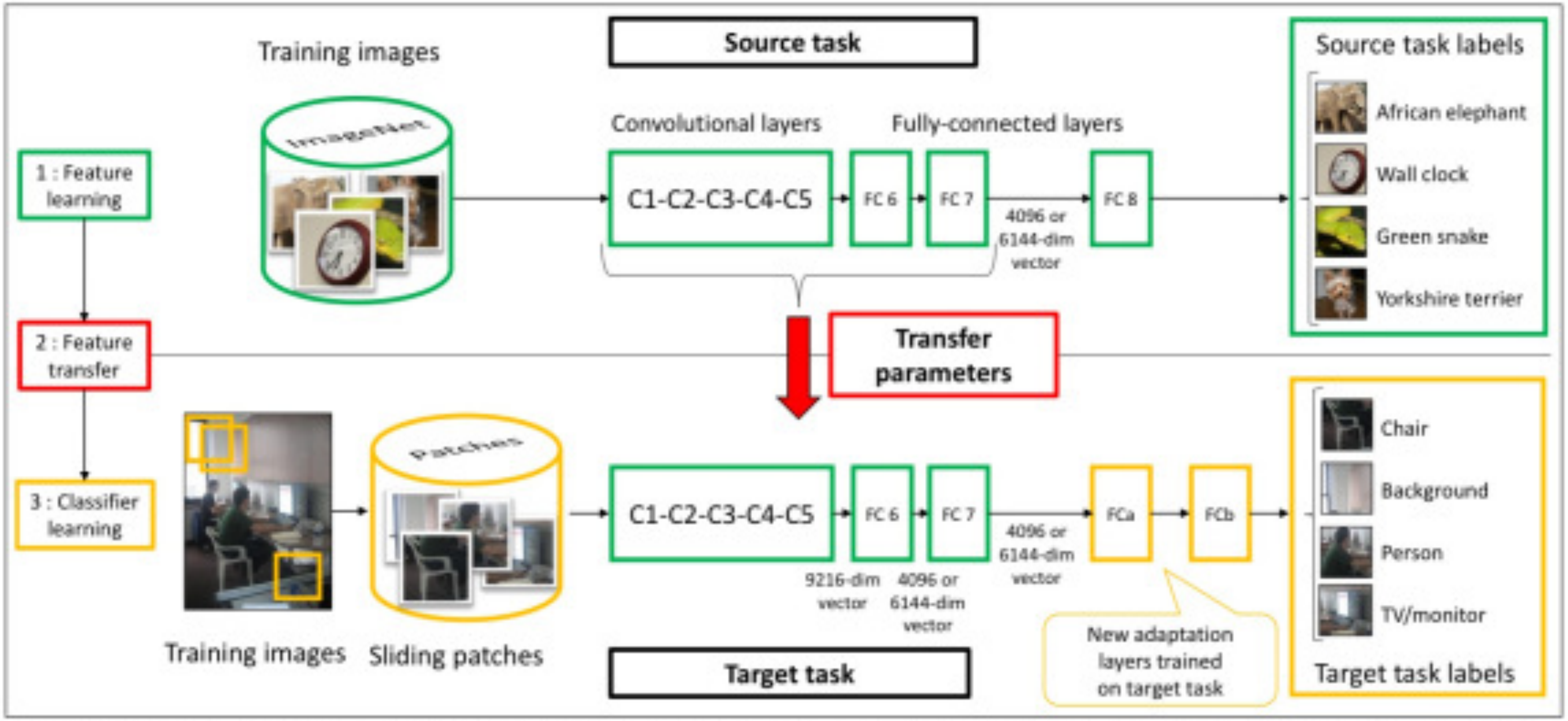


Figure 2: **Transferring parameters of a CNN.** First, the network is trained on the source task (ImageNet classification, top row) with a large amount of available labelled images. Pre-trained parameters of the internal layers of the network (C1-FC7) are then transferred to the target tasks (Pascal VOC object or action classification, bottom row). To compensate for the different image statistics (type of objects, typical viewpoints, imaging conditions) of the source and target data we add an adaptation layer (fully connected layers FCa and FCb) and train them on the labelled data of the target task.

Learning Neural Networks

using "distillation"

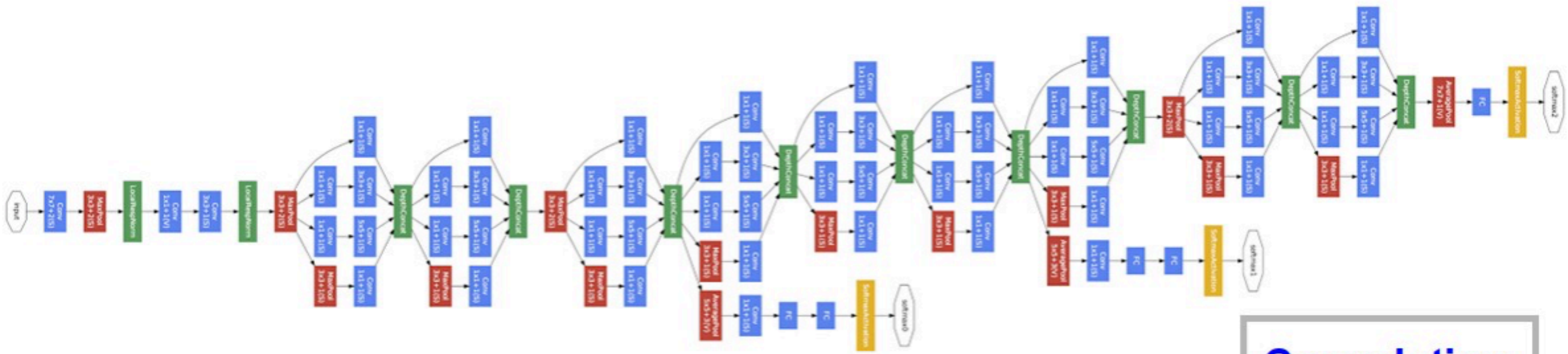
Motivation

1. We would like to deploy a classifier (NN) on a **computationally limited device** (e.g. *a smartphone*)
   - A deep NN cannot be used

2. The **learning task is difficult** and requires a large data set and a sophisticated learning method (e.g. a deep NN)

Question: can we use the learned deep NN as a **teacher** to help the **student** (i.e. the limited device) learn a simpler classifier?

Example: A sophisticated learning technique - GoogLeNet



Convolution
Pooling
Softmax
Other

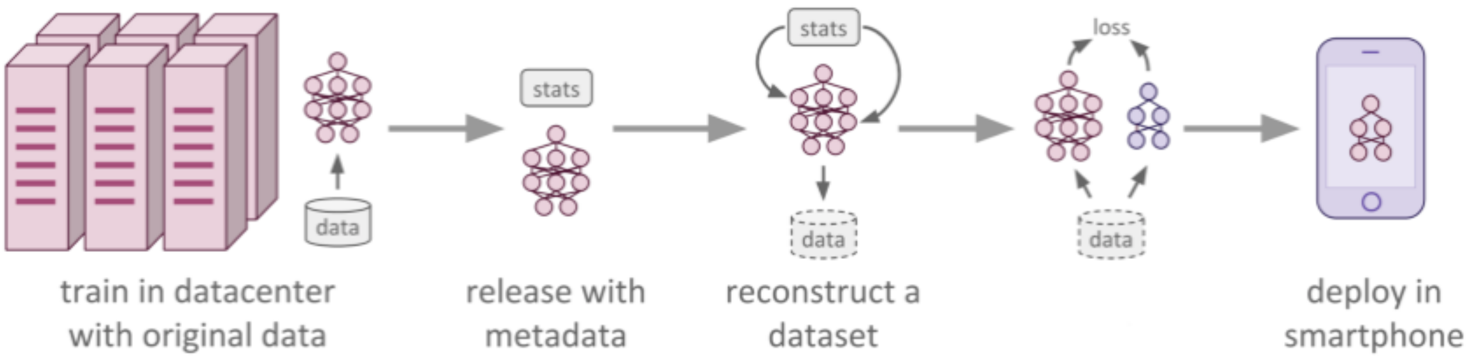
Quite a costly machine to train

AND to use for prediction

Distillation: principle

1. Use the sophisticated learning method (teacher) to learn to predict the target classes with a membership measure

2. Ask the student to *learn to predict the membership measure* computed by the teacher instead of the hard classes (on the training set)



Distillation: principle

1. The teacher uses a softmax function for the values of its output

$$q_i = \frac{e^{(z_i/T)}}{\sum_{j \in \text{classes}} e^{(z_j/T)}}$$

   T is the temperature (the highest *T*, the less different are the outputs)

2. The student *learns to predict the membership measure* first with *T* high, and then, progressively, with *T* decreasing to 1.

When the soft targets have high entropy, they **provide much more information per training case** than hard targets and **much less variance in the gradient** between training cases, so the small model can often be trained on much less data than the original cumbersome model while using a much higher learning rate.

Questions

Questions

- What is a **"successful" transfer** learning situation?
  - How to **measure** "success"?
  - How can we **measure** the **performance** of transfer learning?
  - Is "**failure**" possible? Illustrations?

Remark:

if the **target** data set is **sufficiently large**,

transfer learning should not bring any advantage

Questions

- What are the conditions for a **successful transfer** learning?
- Why the proximity between the **source** and the **target** should play a role?
  - How to **measure** this proximity?
    - Between the **input distributions** P_S and P_T?
    - Between the **underlying** true source and target **functions** f_S and f_T?
- What should intervene in the guarantees?
  - "**distance**" between source and target?
  - Size of the **target** training **data**?
  - Performance of the **source hypothesis**?

Questions

- **What** to transfer?
- **When** to transfer? Useful or not?
- **How** to transfer?

Existing theories

- **Unsupervised** Domain Adaptation

  $f_S = f_T$ but $D_S \neq D_T$

1. **Divergence-based** generalization bounds

2. Generalization bounds taking into account **the geometry of the data distributions**
   - **Wasserstein** distance (optimal transport)
   - **Maximum mean discrepancy** distance

Unsupervised Domain Adaptation

- A pioneering theory [Ben-David et al., 2010]

Théorème classique [Ben-David et al., 2010, Mansour et al., 2009a]

Soit $\mathcal{H}$ un espace d'hypothèses. Si D_S et D_T sont deux distributions sur X, alors :

$$\forall h \in \mathcal{H}, \quad \overbrace{R_{P_T}(h)}^{\text{erreur cible}} \leq \underbrace{R_{P_S}(h)}_{\text{erreur source}} + \underbrace{\tfrac{1}{2} d_{\mathcal{H}}(D_S, D_T) + \nu}_{\text{divergences}}$$

$R_{P_S}(h)$: erreur classique sur le domaine source

Minimisable via une méthode de classification supervisée sans adaptation

$\frac{1}{2} d_{\mathcal{H}}(D_S, D_T)$: la $\mathcal{H}$-divergence entre D_S et D_T

$$\begin{aligned}\tfrac{1}{2} d_{\mathcal{H}}(D_S, D_T) &= \sup_{(h,h')\in\mathcal{H}^2} \left| R_{D_T}(h,h') - R_{D_S}(h,h') \right| \\ &= \sup_{(h,h')\in\mathcal{H}^2} \left| \mathop{\mathbf{E}}_{\mathbf{x}^t\sim D_T} \mathbf{I}\big[h(\mathbf{x}^t) \neq h'(\mathbf{x}^t)\big] - \mathop{\mathbf{E}}_{\mathbf{x}^s\sim D_S} \mathbf{I}\big[h(\mathbf{x}^s) \neq h'(\mathbf{x}^s)\big] \right|\end{aligned}$$

ν : divergence entre les étiquetages

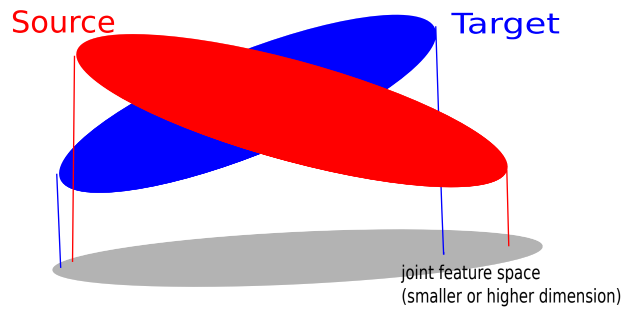
$\nu = \inf_{h'\in\mathcal{H}} \left(R_{P_S}(h') + R_{P_T}(h')\right)$,
erreur jointe optimale [Ben-David et al., 2010]

ou $\nu = R_{P_T}(h^*_T) + R_{P_T}(h^*_T, h^*_S)$,
$h^*_{\mathcal{X}}$ est la meilleure hypothèse sur le domaine $\mathcal{X}$ [Mansour et al., 2009a]

Idea: **build** a projection space in which the two distributions are close, while keeping a high performance level on the source domain

Idea

- **Change the feature representation** X to better represent shared characteristics between the two domains
  - some features are domain-specic,
  - others are generalizable
  - or there exist mappings from the original space



=> Make source and target domain explicitly **similar**

=> Learn a **new feature space** by embedding or projection

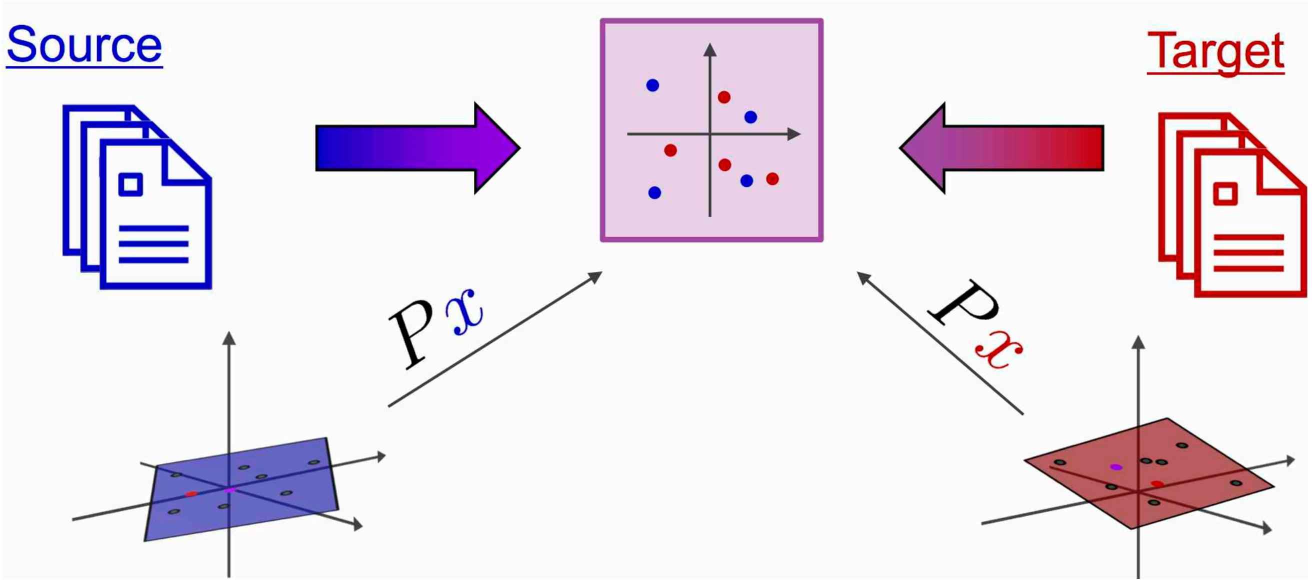
Identify shared features

Domains	Negative	Positive
Books	plot <num>_pages predictable reading_this page_<num>	reader grisham engaging must_read fascinating
Kitchen	the_plastic poorly_designed leaking awkward_to defective	excellent_product espresso are_perfect years_now a_breeze
Pivot features	weak don't_waste awful	and_easy loved_it a_wonderful a_must highly_recommended

- Sentiment analysis - Bag of words (bigrams)
- Choose K pivot features (frequent words in both domains, highly correlated with labels)
- Learn K classifiers to predict pivot features from remaining features
- For each feature add K new features
- Represents source and target data with these features

Illustration: Find latent spaces – Structural Correspondence Learning [Blitzer et al., 2007]

- Apply PCA source+target new features to get a low rank latent representation
- Learn a classifier in the new projection space defined by PCA



Existing theories

- **H**ypothesis **T**ransfer **L**earning

  h_S is known but not S_S

  – Known generalization bounds only for linear classifiers

  – And when $X_S = X_T$

[Redko, I., Morvant, E., Habrard, A., Sebban, M., & Bennani, Y. (2019). *Advances in Domain Adaptation Theory*. Elsevier.]

Theory for HTL

$$h(\mathbf{x}) := \langle \hat{\boldsymbol{w}}, \mathbf{x} \rangle$$

$$\hat{\boldsymbol{w}} = \underset{\mathbf{w} \in \mathcal{H}}{\operatorname{argmin}} \left\{ \frac{1}{m} \sum_{i=1}^{m} (\langle \hat{\boldsymbol{w}}, \mathbf{x}_i \rangle - y_i)^2 + \lambda \|\mathbf{w} - \sum_{j=1}^{n} \beta_j \mathbf{w}_{\mathrm{src}}^j\|_2^2 \right\}$$

THEOREM 7.3 ([KUZ 17]).– *Let $h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}$ a hypothesis output by a regularized ERM algorithm from a m-sized training set T i.i.d. from the target domain $\mathcal{T}$, n source hypotheses $\{h^i_{src} : \|h^i_{src}\|_\infty \leq 1\}_{i=1}^n$, any source weights $\boldsymbol{\beta}$ obeying $\Omega(\boldsymbol{\beta}) \leq \rho$ and $\lambda \in \mathbb{R}_+$. Assume that the loss is bounded by M: $\ell(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}(\mathbf{x}), y) \leq M$ for any $(\mathbf{x}, y)$ and any training set. Then, denote $\kappa = \frac{H}{\sigma}$ and assuming that $\lambda \leq \kappa$ with probability at least $1 - e^{-\eta}$, $\forall \eta \geq 0$:*

$$\mathrm{R}_{\mathcal{T}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) \leq \mathrm{R}_{\hat{\mathcal{T}}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{\mathrm{R}^{src}_{\mathcal{T}} \kappa}{\sqrt{m}\lambda} + \sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}} \rho \kappa^2}{m\lambda}} + \frac{M\eta}{m \log\left(1 + \sqrt{\frac{M\eta}{u^{src}}}\right)} \right)$$

$$\leq \mathrm{R}_{\hat{\mathcal{T}}}(h_{\hat{\boldsymbol{w}},\boldsymbol{\beta}}) + \mathcal{O}\left(\frac{\kappa}{\sqrt{m}} \left(\frac{\mathrm{R}^{src}_{\mathcal{T}}}{\lambda} + \sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}} \rho}{\lambda}} \right) + \frac{\kappa}{m} \left(\frac{\sqrt{\mathrm{R}^{src}_{\mathcal{T}} M\eta}}{\lambda} + \sqrt{\frac{\rho}{\lambda}} \right) \right),$$

where $u^{src} = \mathrm{R}^{src}_{\mathcal{T}} \left(m + \frac{\kappa\sqrt{m}}{\lambda} \right) + \kappa \sqrt{\frac{\mathrm{R}^{src}_{\mathcal{T}} m\rho}{\lambda}}$ *and* $\mathrm{R}^{src}_{\mathcal{T}} = \mathrm{R}_{\mathcal{T}}(h^{\boldsymbol{\beta}}_{src})$ *is the risk of the source hypothesis combination.*

Theory for HTL

The risk of the **source hypothesis** combination on the **target domain**

provides an important indicator on the

relatedness between the **source** and the **target** domain

[Redko, I., Morvant, E., Habrard, A., Sebban, M., & Bennani, Y. (2019). *Advances in Domain Adaptation Theory*. Elsevier.] p. 113

Outline

1. Classical inductive learning

2. Transfer learning

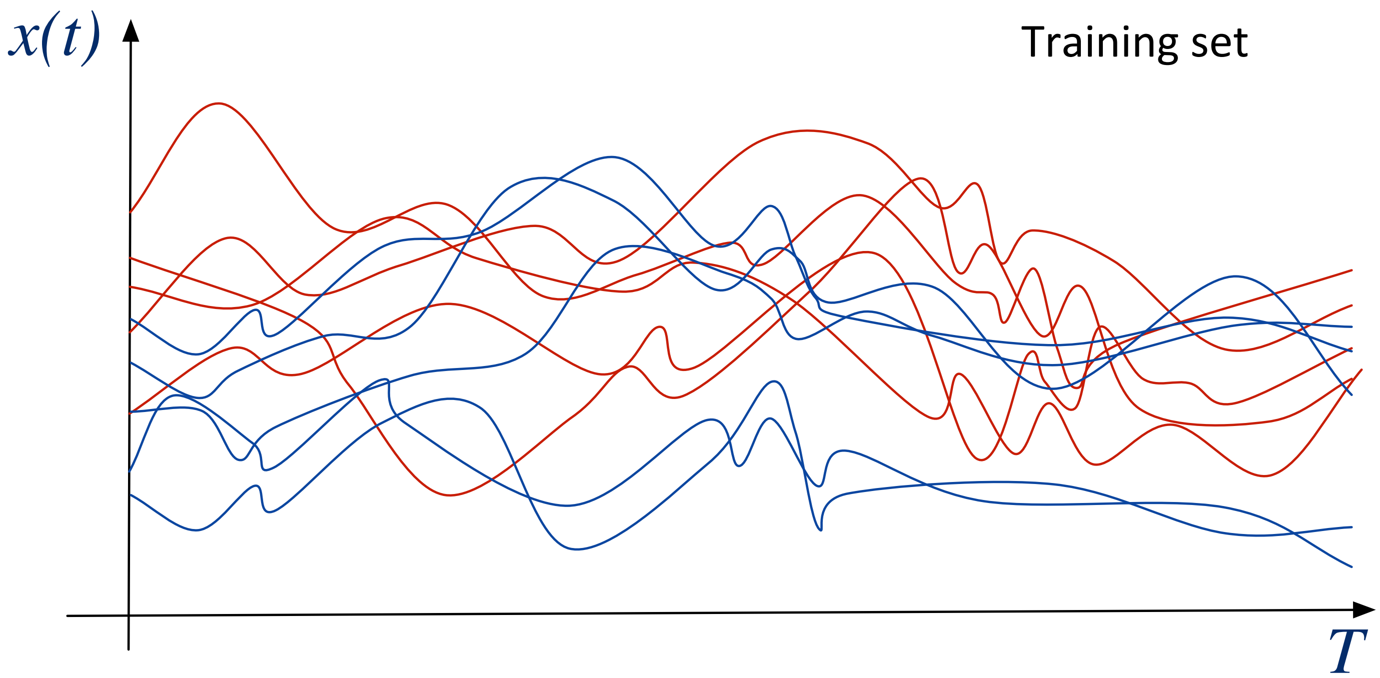
3. TransBoost: an original approach

4. Conclusion

TransBoost

An original approach to transfer learning

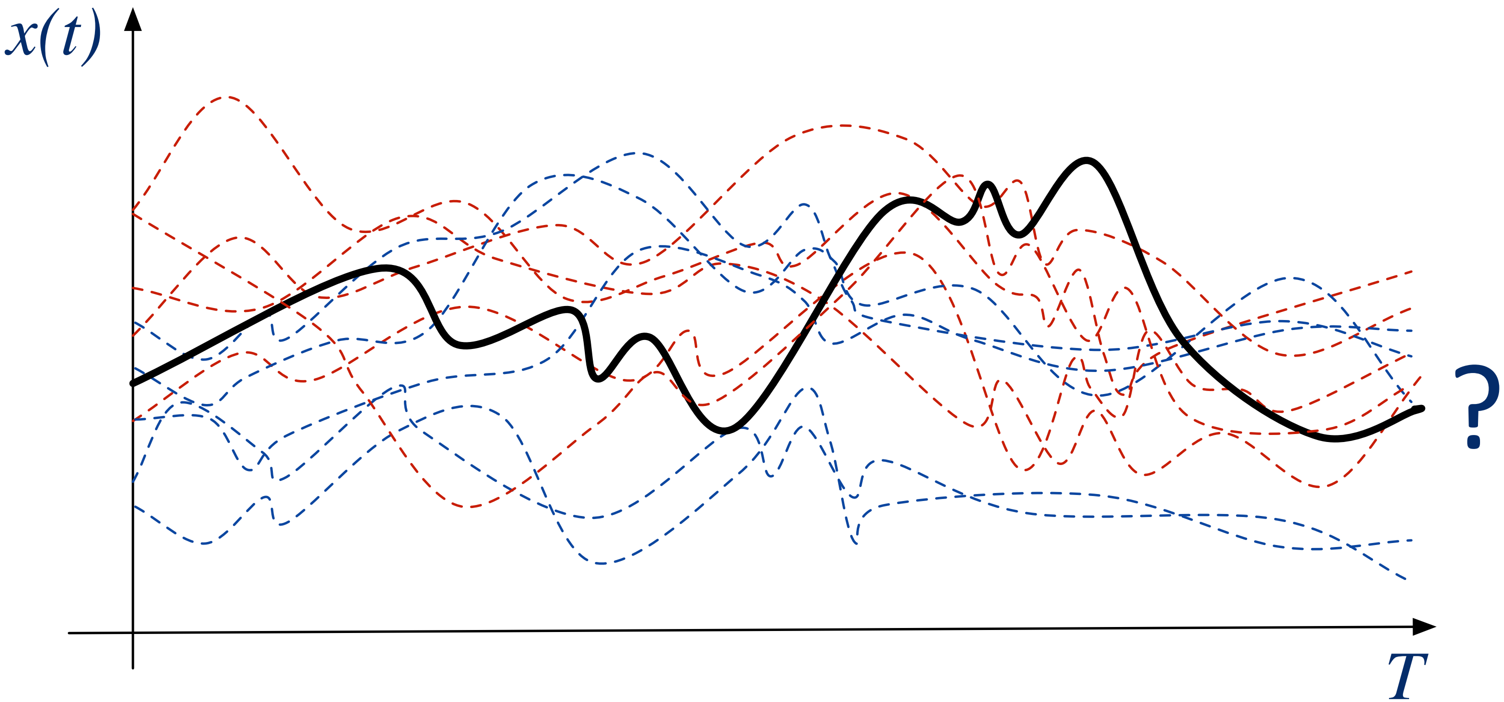
Classification of time series



- Monitoring of ***consumer actions on a web site***: will buy or not
- Monitoring of a ***patient state***: critical or not
- Early prediction of daily ***electrical consumption***: high or low

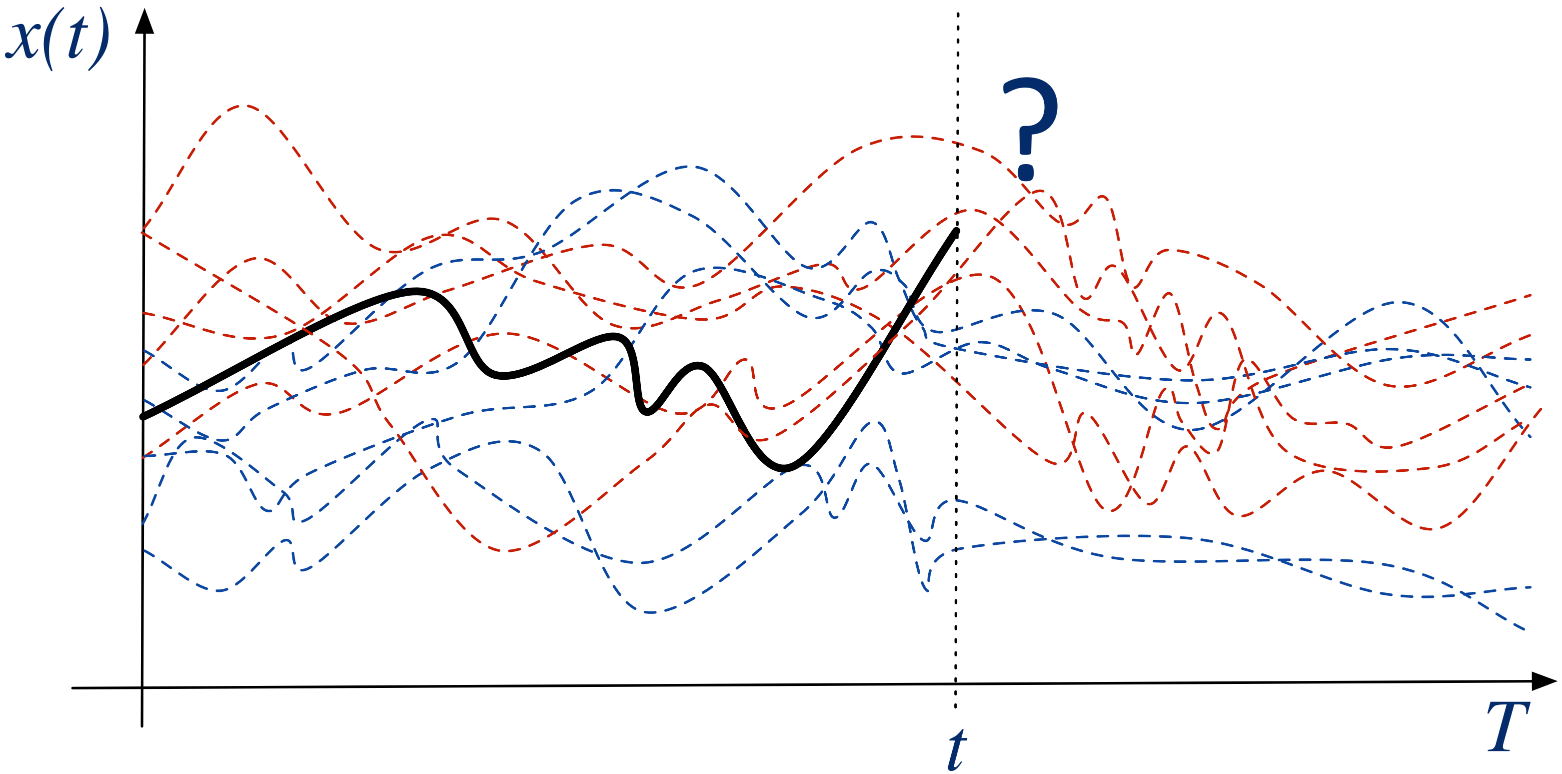
Standard classification of time series

- What is the class of the new time series x_T?



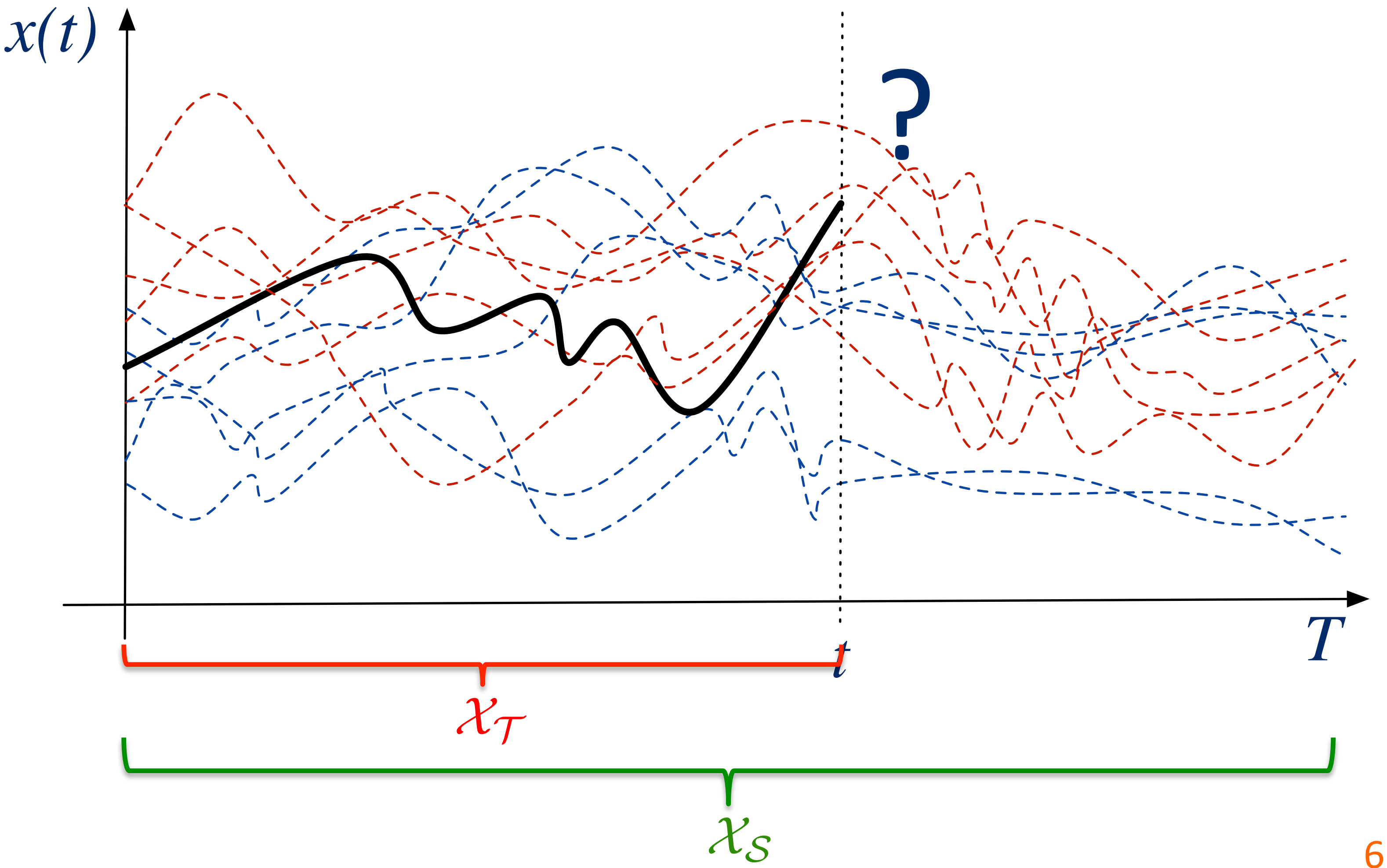
Early classification of time series

- What is the class of the new **incomplete** time series $\boldsymbol{x_t}$?



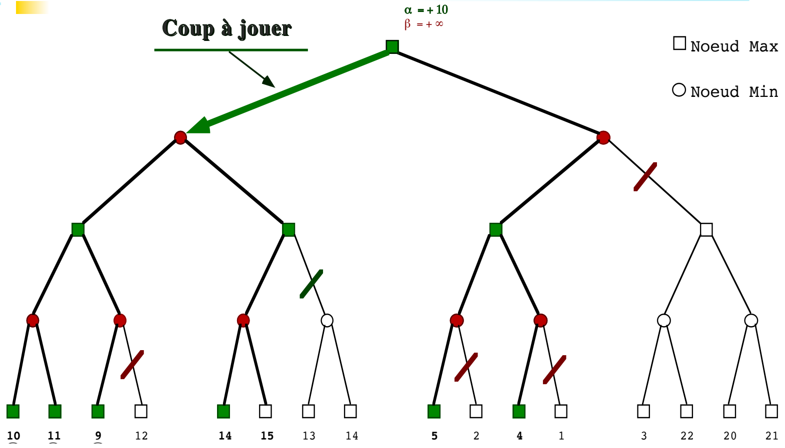
Early classification of time series

- What is the class of the new **incomplete** time series $\boldsymbol{x_t}$?

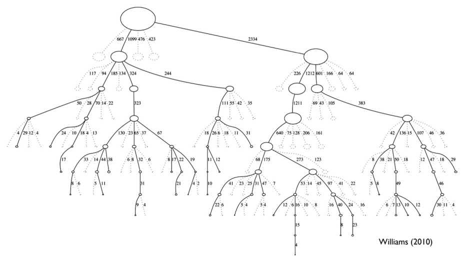


Algorithms for games

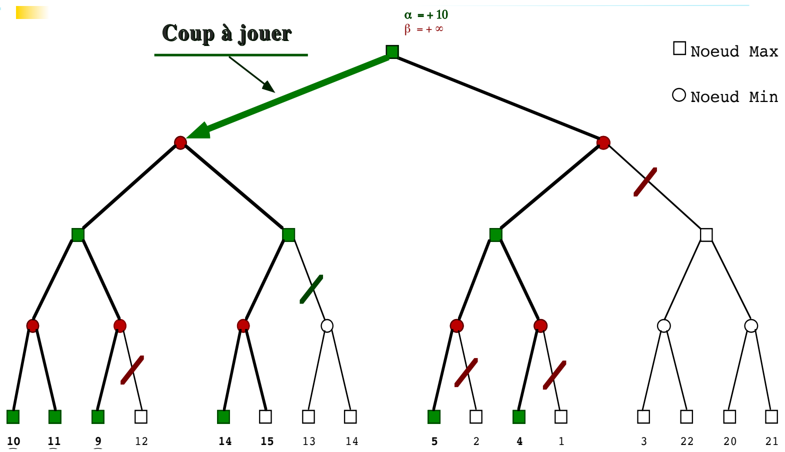
Taking decision when the current information is **incomplete**



MCTS



Algorithms for games



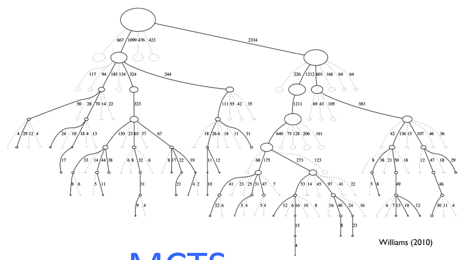
Taking decision when the current information is **incomplete**

- Which move to play?

The evaluation function is **insufficiently informed** at the root (current situation)

1. **Query experts** that have more information about potential outcomes

2. **Combination** of the estimates through MinMax



MCTS

*“Experts” may live in **input spaces** that are **different***

Principle

- Learn a classifier over the training set of complete **times series**

$$S_{\mathcal{S}} = \{(\mathbf{x}_i^{\mathcal{S}}, y_i^{\mathcal{S}})\}_{1 \leq i \leq m} \;\rightarrow\; h_{\mathcal{S}}$$

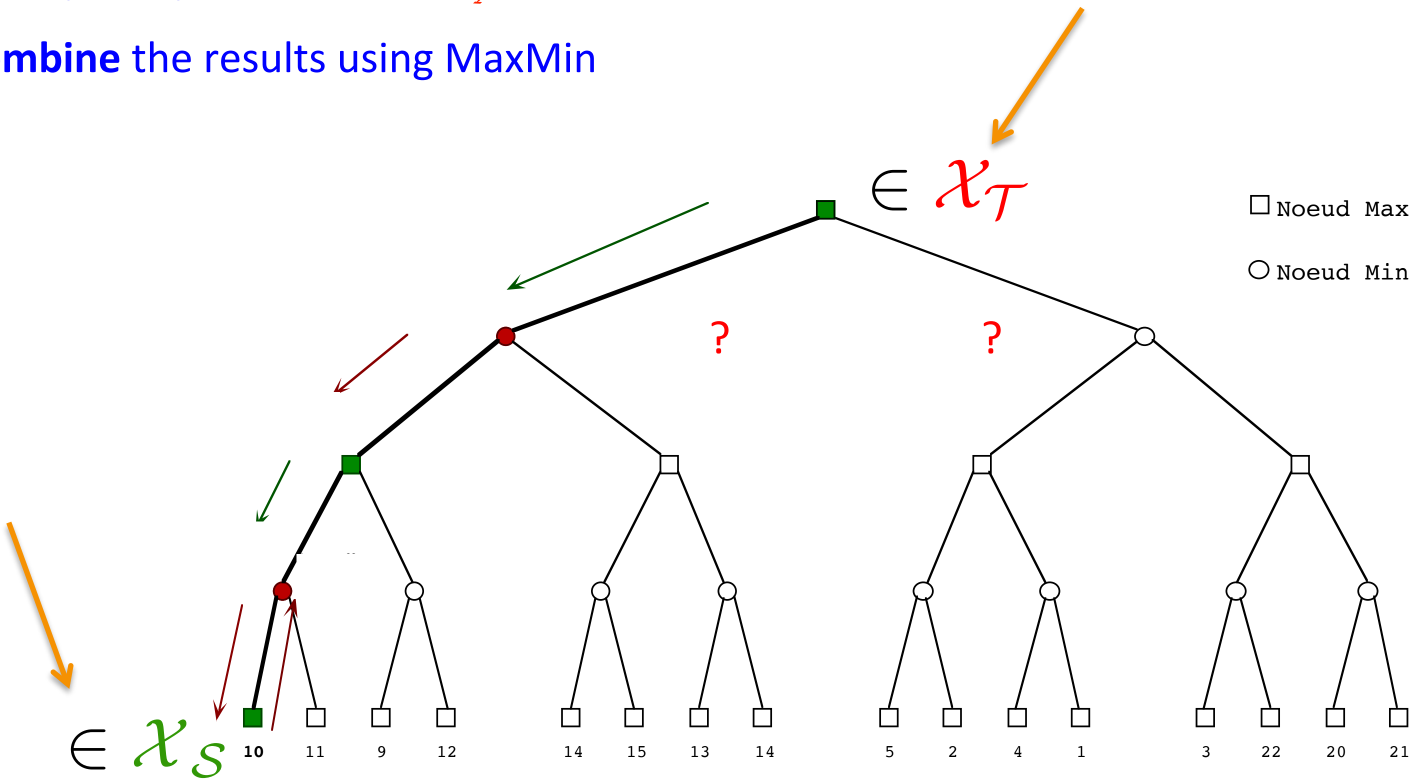
- Try to make use of this classifier to predict the class of **incomplete** series

$$h_{\mathcal{T}} \;=\; \texttt{Function using } h_{\mathcal{S}}$$

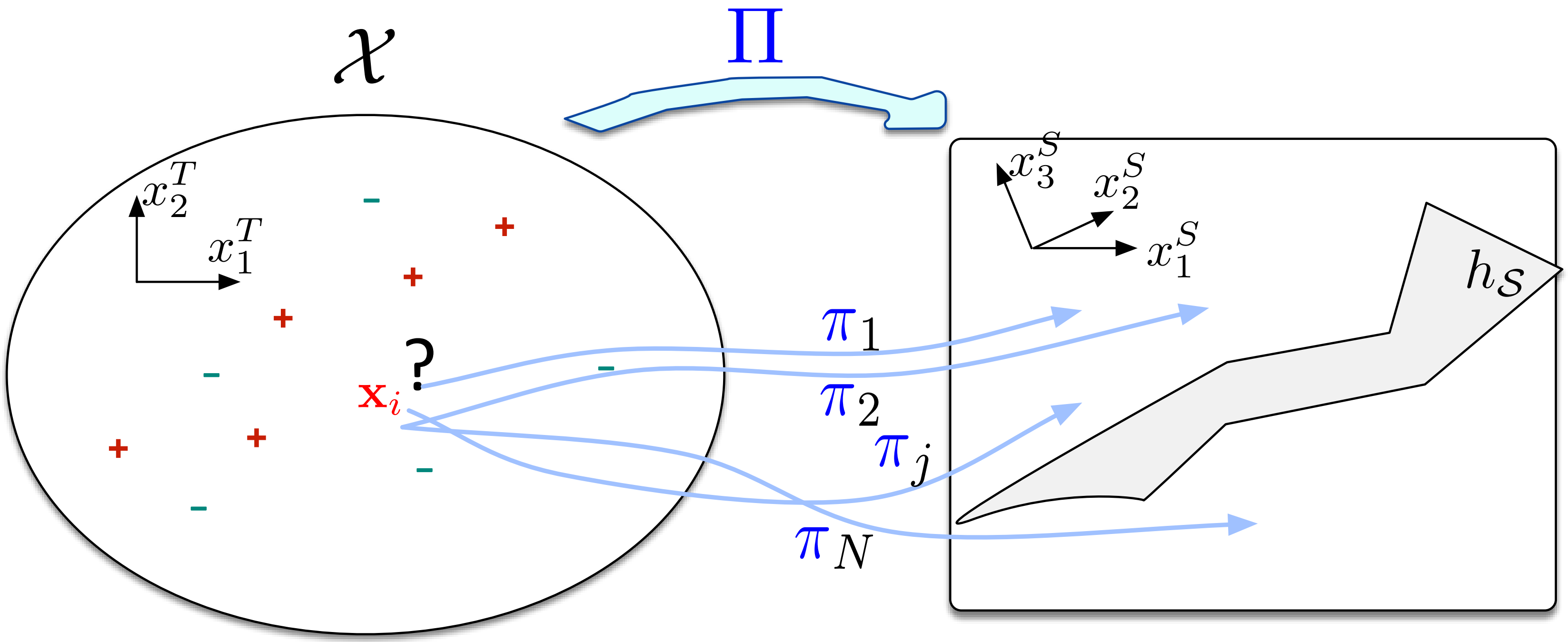
Algorithms for games and transfer learning

Which move?

- Better evaluation function in $\mathcal{X}_S$
- Backup it (by transfer) for $\mathcal{X}_{\mathcal{T}}$
- **Combine** the results using MaxMin



TransBoost

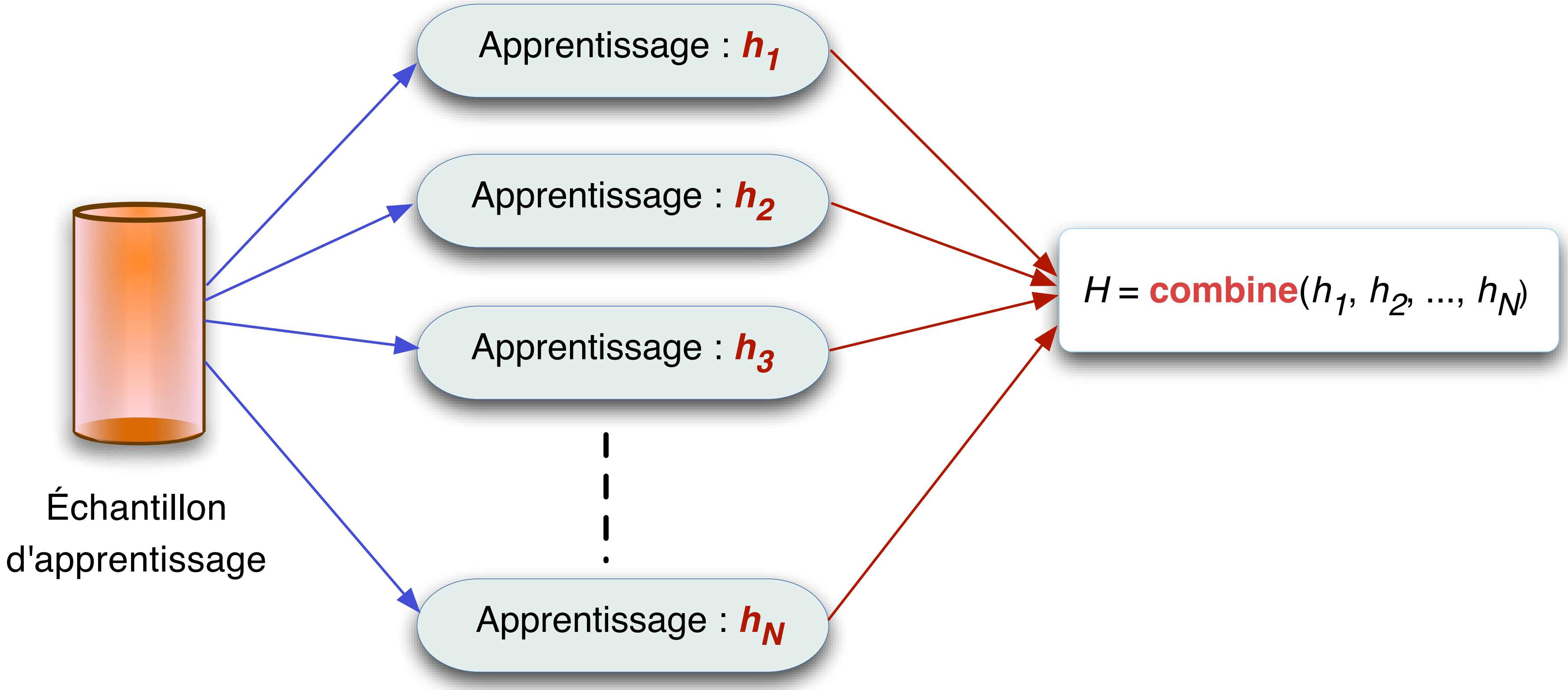


***Target** Domain*

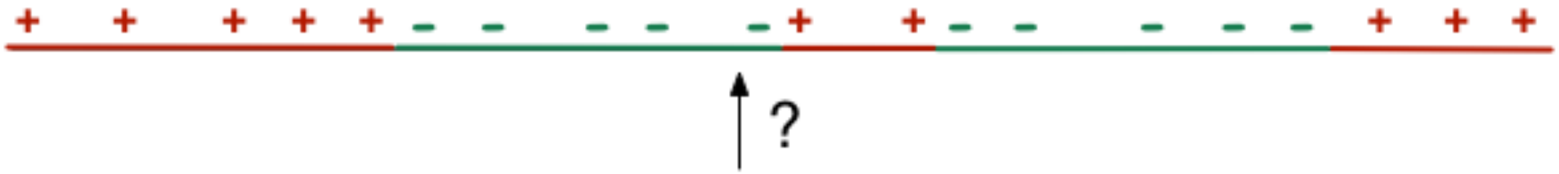
***Source** Domain*

$$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \mathrm{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\big(\pi_n(\mathbf{x}^{\mathcal{T}})\big)\right\}$$

Schéma général : apprentissage



Exemple simple



- Quel est le meilleur séparateur linéaire ?

Exemple simple

- Taux d'erreur = 5/20 = 0.25

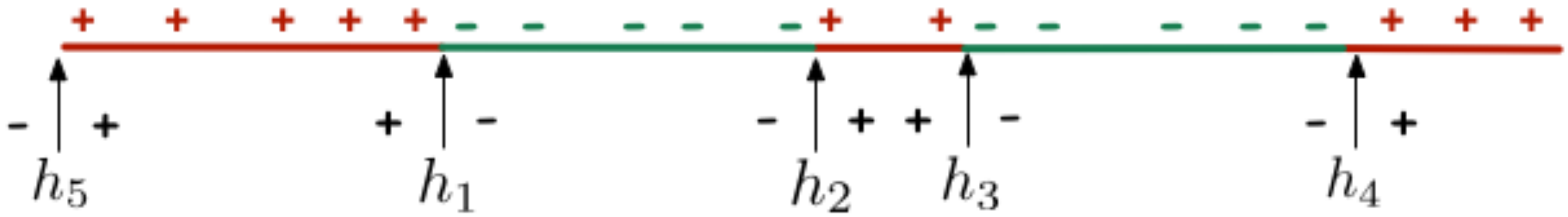
Exemple simple

Et si je pouvais combiner avec un autre séparateur linéaire ? Ou même plusieurs autres !

Par exemple en utilisant un vote pondéré :

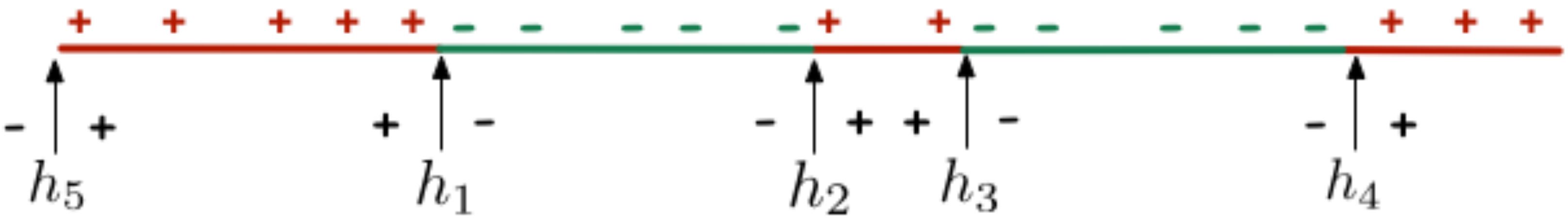
$$H(\mathbf{x}) = \mathrm{sign}\left\{\sum_{i=1}^{l} \alpha_i \, h_i(\mathbf{x})\right\}$$

Exemple simple



$$H(x) = \text{sign}\{\, 0.549\, h_1(x) + 0.347\, h_2(x) + 0.310\, h_3(x) + 0.406\, h_4(x) + 0.503\, h_5(x) \,\}$$

Exemple simple

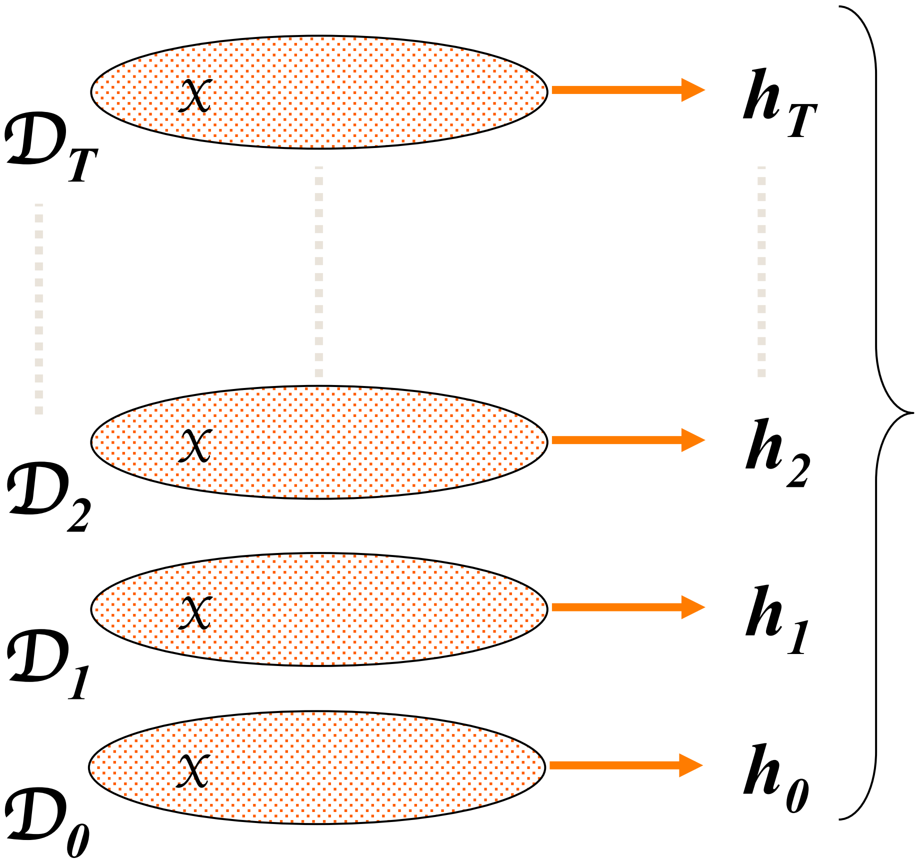


H(x) =sign{ 0.549 $h_1(x)$ + 0.347 $h_2(x)$ + 0.310 $h_3(x)$ + 0.406 $h_4(x)$ + 0.503 $h_5(x)$ }

- Comment arriver à ce genre de combinaison ?

Algorithme du boosting

Le principe général



$$H_{finale}(\boldsymbol{x}) \; = \; \mathrm{sgn}\left[\sum_{t=0}^{T} \alpha_t \, . \, h_t(\boldsymbol{x})\right]$$

- **Comment passer de $\mathcal{D}_t$ à $\mathcal{D}_{t+1}$?**
- **Comment calculer la pondération α_t ?**

TransBoost

- Principle:
  - Learn "*weak projections*": $\pi_i : \mathcal{X}_{\mathcal{S}} \rightarrow \mathcal{X}_{\mathcal{T}}$
    - From: $S_{\mathcal{S}} = \{(\mathbf{x}_i^{\mathcal{S}}, y_i^{\mathcal{S}})\}_{1 \leq i \leq m}$
  - Using boosting
    - Projection π_n such that : $\varepsilon_n \doteq \mathbf{P}_{i \sim D_n}[h_{\mathcal{S}}(\pi_n(\mathbf{x}_i)) \neq y_i] < 0.5$
    - Re-weight the training time series and loop until termination
  - Result $$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \mathrm{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\left(\pi_n(\mathbf{x}^{\mathcal{T}})\right)\right\}$$

TransBoost

Algorithm 1: Transfer learning by boosting

Input: $h_{\mathcal{S}} : \mathcal{X}_{\mathcal{S}} \rightarrow \mathcal{Y}_{\mathcal{S}}$ the source hypothesis
$\mathcal{S}_{\mathcal{T}} = \{(\mathbf{x}_i^{\mathcal{T}}, y_i^{\mathcal{T}}\}_{1 \leq i \leq m}$: the target training set

Initialization of the distribution on the training set: $D_1(i) = 1/m$ for $i = 1, \ldots, m$;

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

Find a projection $\pi_i : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$ st. $h_{\mathcal{S}}(\pi_i(\cdot))$ performs better than random on $D_n(\mathcal{S}_{\mathcal{T}})$;
Let ε_n be the error rate of $h_{\mathcal{S}}(\pi_i(\cdot))$ on $D_n(\mathcal{S}_{\mathcal{T}})$: $\varepsilon_n \doteq \mathbf{P}_{i \sim D_n}[h_{\mathcal{S}}(\pi_n(\mathbf{x}_i)) \neq y_i]$ (with $\varepsilon_n < 0.5$) ;
Computes $\alpha_i = \frac{1}{2} \log_2\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$;
Update, for $i = 1 \ldots, m$:

$$D_{n+1}(i) = \frac{D_n(i)}{Z_n} \times \begin{cases} e^{-\alpha_n} & \text{if } h_{\mathcal{S}}\big(\pi_n(\mathbf{x}_i^{\mathcal{T}})\big) = y_i^{\mathcal{T}} \\ e^{\alpha_n} & \text{if } h_{\mathcal{S}}\big(\pi_n(\mathbf{x}_i^{\mathcal{T}})\big) \neq y_i^{\mathcal{T}} \end{cases}$$

$$= \frac{D_n(i) \exp\big(-\alpha_n \, y_i^{(\mathcal{T})} \, h_{\mathcal{S}}(\pi_n(\mathbf{x}_i^{(\mathcal{T})}))\big)}{Z_n}$$

where Z_n is a normalization factor chosen so that D_{n+1} be a distribution on $\mathcal{S}_{\mathcal{T}}$;

end

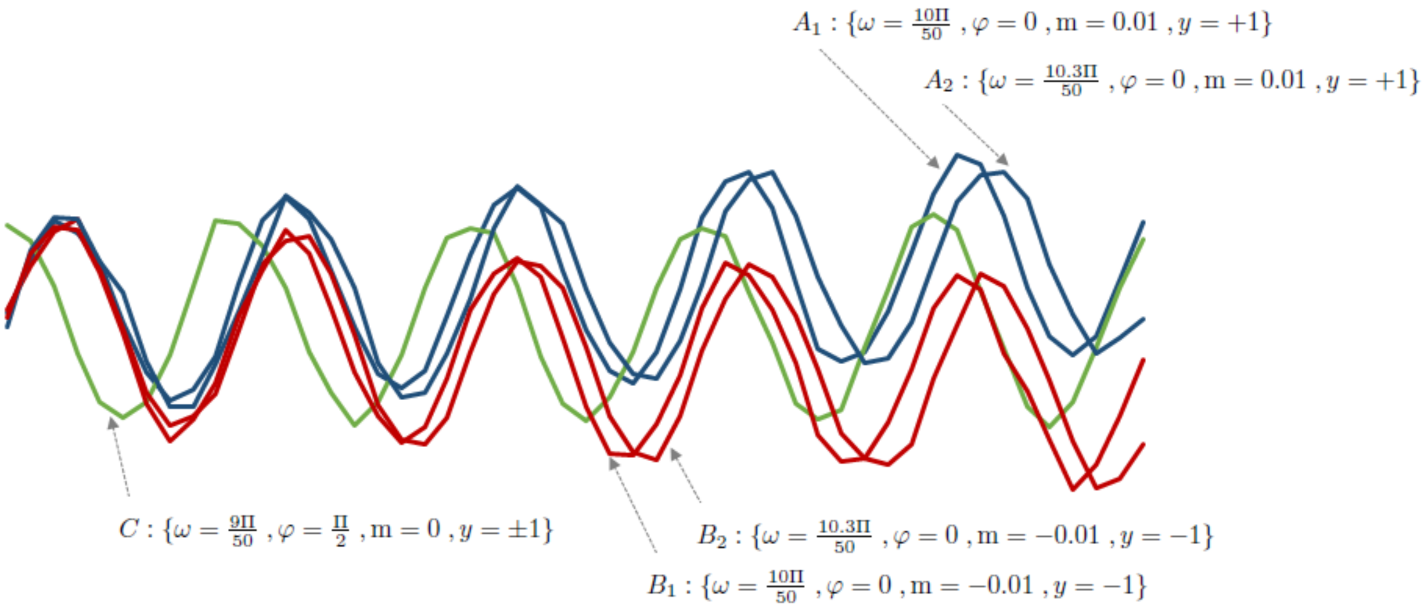
Output: the final target hypothesis $H_{\mathcal{T}} : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{Y}_{\mathcal{T}}$:

$$\boxed{H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \operatorname{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\big(\pi_n(\mathbf{x}^{\mathcal{T}})\big)\right\}} \tag{2}$$

Controlled data

- Control of
  - The time-dependent **information** provided to distinguish between **classes**
  - The shapes of **time series** within each class
  - The **noise level**

$$\mathbf{x}_t \;=\; \underbrace{t \times \text{slope} \times \text{class}}_{\text{information gain}} + \underbrace{\mathrm{x}_{max}\,\sin(\omega_i \times t + \varphi_j)}_{\text{sub shape within class}} + \underbrace{\eta(t)}_{\text{noise factor}}$$



L'espace des projections

- Ensemble de projections

Fonctions coude (5 paramètres)

- **Abscisse** du coude
- **Angles** avant et après
- **Fenêtre** prise en compte

Results

Learning from target data only

TransBoost

On the source domain

Naïve transfert

slope, noise, $t_{\mathcal{T}}$	$h_{\mathcal{T}}$ (train)	$h_{\mathcal{T}}$ (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)	$h_{\mathcal{S}}$ (test)	$H'_{\mathcal{T}}$ (test)
0.001, 0.001, 20	0.46 ± 0.02	0.50 ± 0.08	0.08 ± 0.03	$\mathbf{0.08} \pm 0.02$	0.05	0.49 ± 0.01
0.005, 0.001, 20	0.46 ± 0.02	0.49 ± 0.01	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.45 ± 0.01
0.005, 0.002, 20	0.46 ± 0.02	0.49 ± 0.03	0.03 ± 0.02	$\mathbf{0.04} \pm 0.02$	0.02	0.43 ± 0.01
0.005, 0.02, 20	0.44 ± 0.02	0.48 ± 0.03	0.09 ± 0.01	$\mathbf{0.10} \pm 0.01$	0.01	0.47 ± 0.01
0.001, 0.2, 20	0.46 ± 0.02	0.50 ± 0.01	0.46 ± 0.02	0.51 ± 0.02	0.11	0.49 ± 0.01
0.01, 0.2, 20	0.42 ± 0.03	0.47 ± 0.03	0.34 ± 0.02	0.35 ± 0.02	0.02	0.35 ± 0.01
0.001, 0.001, 50	0.46 ± 0.02	0.50 ± 0.01	0.08 ± 0.03	$\mathbf{0.08} \pm 0.02$	0.06	0.41 ± 0.01
0.005, 0.001, 50	0.25 ± 0.07	0.28 ± 0.09	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.28 ± 0.01
0.005, 0.002, 50	0.27 ± 0.07	0.30 ± 0.08	0.02 ± 0.01	$\mathbf{0.02} \pm 0.01$	0.02	0.28 ± 0.01
0.005, 0.02, 50	0.26 ± 0.07	0.30 ± 0.08	0.04 ± 0.01	$\mathbf{0.04} \pm 0.01$	0.01	0.31 ± 0.01
0.001, 0.2, 50	0.44 ± 0.02	0.50 ± 0.01	0.38 ± 0.03	0.44 ± 0.02	0.15	0.43 ± 0.01
0.01, 0.2, 50	0.10 ± 0.03	0.12 ± 0.04	0.10 ± 0.02	0.11 ± 0.02	0.03	0.15 ± 0.02
0.001, 0.001, 100	0.43 ± 0.03	0.47 ± 0.03	0.07 ± 0.02	$\mathbf{0.07} \pm 0.02$	0.02	0.23 ± 0.01
0.005, 0.001, 100	0.06 ± 0.03	0.07 ± 0.03	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$	0.01	0.07 ± 0.02
0.005, 0.002, 100	0.08 ± 0.03	0.10 ± 0.04	0.02 ± 0.01	$\mathbf{0.02} \pm 0.01$	0.02	0.07 ± 0.01
0.005, 0.02, 100	0.08 ± 0.03	0.09 ± 0.03	0.02 ± 0.01	$\mathbf{0.03} \pm 0.01$	0.01	0.07 ± 0.01
0.001, 0.2, 100	0.04 ± 0.03	0.46 ± 0.02	0.28 ± 0.02	0.31 ± 0.01	0.16	0.31 ± 0.01
0.01, 0.2, 100	0.03 ± 0.01	0.05 ± 0.02	0.04 ± 0.01	0.05 ± 0.01	0.02	0.05 ± 0.01

Table 1: Comparison of learning directly in the target domain (columns $h_{\mathcal{T}}$ (train) and $h_{\mathcal{T}}$ (test)), using `TransBoost` (columns $H_{\mathcal{T}}$ (train) and $H_{\mathcal{T}}$ (test)), learning in the source domain (column $h_{\mathcal{S}}$ (test)) and, finally, completing the time series with a SVR regression and using $h_{\mathcal{S}}$ (naïve transfer). Test errors are highlighted in the orange columns. Bold numbers indicates where `TransBoost` significantly dominates both learning without transfer and learning with naïve transfer.

Results

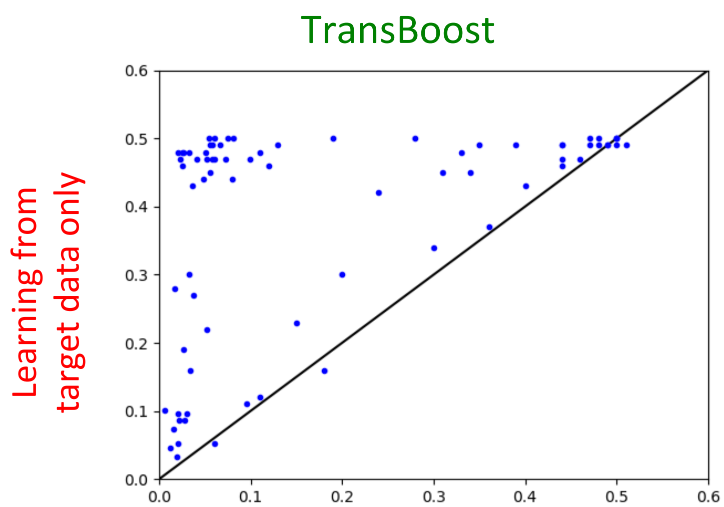


Figure 3: Comparison of error rates. y-axis: test error of the SVM classifier (without transfer). x-axis : test error of the TransBoost classifier with 10 boosting steps. The results of 75 experiments (each one repeated 100 times) are summed up in this graph.

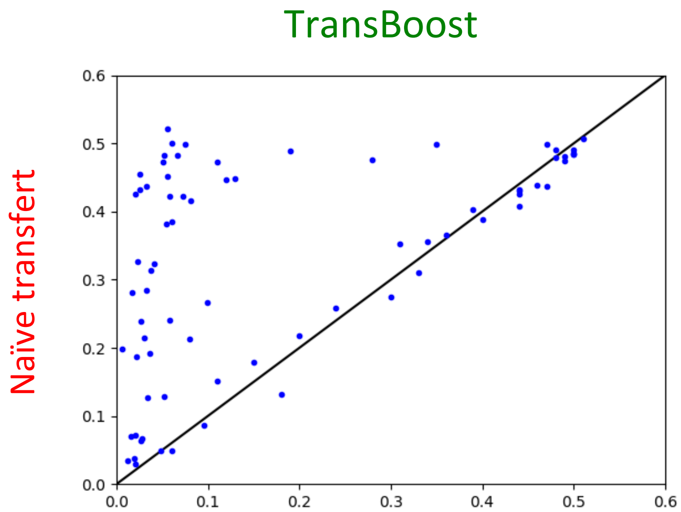
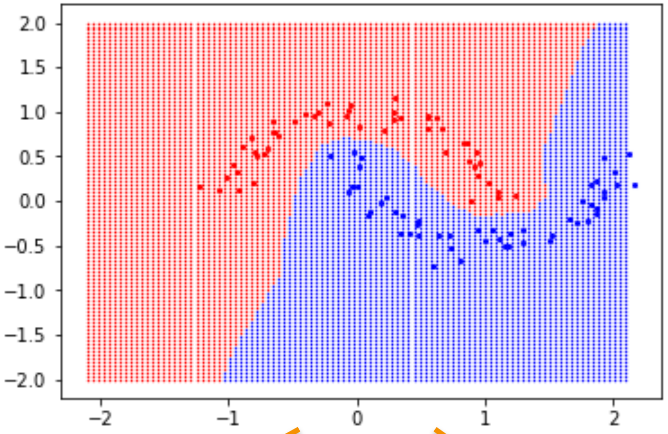


Figure 4: Comparison of error rates. y-axis: test error of the "naïve" transfer method. x-axis : test error of the TransBoost classifier with 10 boosting steps. The results of 75 experiments (each one repeated 100 times) are summed up in this graph.

Transfer learning

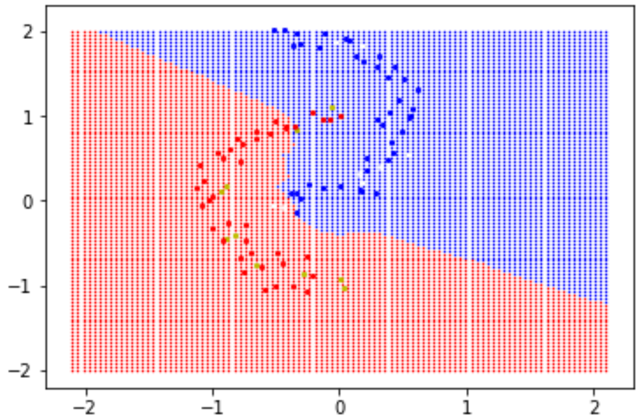


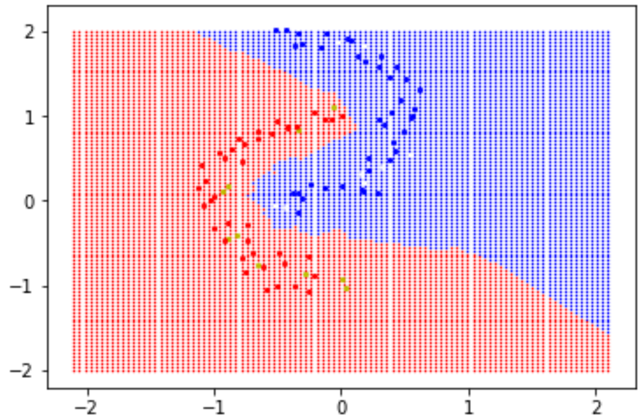
$$\pi_i(\mathbf{x}) \;=\; \mathbf{x} + \mathbf{v}_i$$

$$\pi_i(\mathbf{x}) \;=\; \mathbf{A}_i \cdot \mathbf{x} + \mathbf{v}_i$$

Learning on the target data
(**without transfer**)

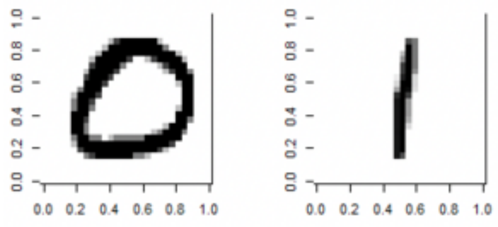
Using Transboost



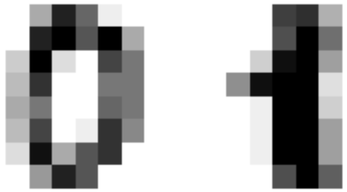


Transfer learning

- Illustrations

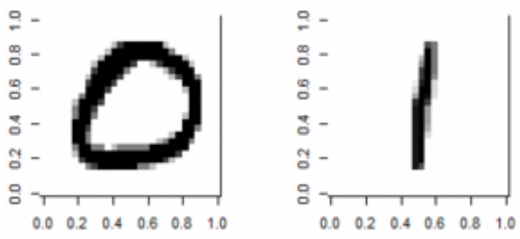


(a) Is it a zero or a one ?

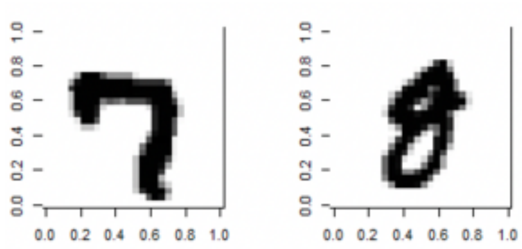


(b) Is it a zero or a one ?

Figure 15: Transfer learning of the source model 0/1 mnist so that it can distinguish 0/1 sklearn digits



(a) Is it a zero or a one ?



(b) Is it an eight or a seven ?

Transfer learning

$X_S = X_T$ & $Y_S = Y_T$	$X_S \neq X_T$ & $Y_S = Y_T$
$X_S = X_T$ & $Y_S \neq Y_T$	$X_S \neq X_T$ & $Y_S \neq Y_T$

- Illustrations


FIGURE 1: Trained model on the data source : is it a picture of a dog or a cat ?


FIGURE 2: Model source transferred on the data target : is it a clip-art of a dog or a cat ?

Conclusion

- Ensemble method for transfer learning
  - Learn **weak translator** from target to source!!
  - The learning problem now becomes the problem of **choosing** a good set of (weak) projections
  - Theoretical guarantees exist

Theoretical guarantees

$$\forall\, \widehat{h}_{\mathcal{S}} \in \mathcal{H}_{\mathcal{S}} : \quad \underset{\pi \in \Pi}{\mathrm{Min}}\, R_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi) \;\leq\; \omega\big(R_{\mathcal{S}}(h_{\mathcal{S}})\big) \qquad (2)$$

where $\omega : \mathbb{R} \rightarrow \mathbb{R}$ is a non-decreasing function.

Theorem 1. *Let $\omega : \mathbb{R} \rightarrow \mathbb{R}$ be a non-decreasing function. Suppose that $P_{\mathcal{S}}$, $P_{\mathcal{T}}$, $h_{\mathcal{S}}$, $h_{\mathcal{T}} = \widehat{h}_{\mathcal{S}} \circ \pi (\pi \in \Pi)$, $\widehat{h}_{\mathcal{S}}$ and Π have the property given by Equation (2). Let $\widehat{\pi} := \mathrm{ArgMin}_{\pi \in \Pi}\, \widehat{R}_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi)$, be the best apparent projection.*

Then, with probability at least $1 - \delta \quad (\delta \in (0,1))$ over pairs of training sets for tasks $\mathcal{S}$ and $\mathcal{T}$:

$$\begin{aligned} R_{\mathcal{T}}(\widehat{h}_{\mathcal{T}}) \;\leq\; \omega\big(\widehat{R}_{\mathcal{S}}(\widehat{h}_{\mathcal{S}})\big) \;+\; 2\sqrt{\frac{2\, d_{\mathcal{H}_{\mathcal{S}}} \log(2e m_{\mathcal{S}}/d_{\mathcal{H}_{\mathcal{S}}}) \;+\; 2\log(8/\delta)}{m_{\mathcal{S}}}} \\ +\; 4\sqrt{\frac{2\, d_{h_{\mathcal{S}} \circ \Pi} \log(2e m_{\mathcal{T}}/d_{h_{\mathcal{S}} \circ \Pi}) \;+\; 2\log(8/\delta)}{m_{\mathcal{T}}}} \end{aligned} \qquad (3)$$

[Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.]

Not good!

Even though this is the one that allowed us to **get publish**

Additional results

- Source hypothesis a priori **without relation** to the target task!!??

	Learning from target data only		TransBoost with any source hypothesis	
slope, noise, $t_{\mathcal{T}}$	$h_{\mathcal{T}}$ (train)	$h_{\mathcal{T}}$ (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)
0.001, 0.001, 70	0.44 ± 0.02	0.48 ± 0.02	0.06 ± 0.02	$\mathbf{0.06} \pm 0.02$
0.005, 0.005, 70	0.11 ± 0.04	0.13 ± 0.05	0.02 ± 0.01	$\mathbf{0.02} \pm 0.02$
0.005, 0.005, 70	0.10 ± 0.04	0.11 ± 0.05	0.01 ± 0.01	$\mathbf{0.01} \pm 0.01$
0.005, 0.05, 70	0.11 ± 0.04	0.12 ± 0.05	0.04 ± 0.02	$\mathbf{0.03} \pm 0.01$
0.001, 0.001, 70	0.42 ± 0.03	0.48 ± 0.02	0.33 ± 0.02	$\mathbf{0.37} \pm 0.02$
0.01, 0.1, 70	0.06 ± 0.03	0.08 ± 0.03	0.08 ± 0.02	0.08 ± 0.02

Table 2: Learning without transfer and with transfer using an apriori irrelevant source hypothesis.

Analysis

- The **quality of the source hypothesis** on the source data?
  - Plays no role

- The **proximity of the source and target** distributions P_X and P_Y?
  - Plays no role

Theoretical guarantees

$$\forall\, \widehat{h}_{\mathcal{S}} \in \mathcal{H}_{\mathcal{S}} : \quad \underset{\pi \in \Pi}{\mathrm{Min}}\, R_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi) \;\leq\; \omega\big(R_{\mathcal{S}}(h_{\mathcal{S}})\big) \qquad (2)$$

Ridiculous

where $\omega : \mathbb{R} \to \mathbb{R}$ is a non-decreasing function.

Irrelevant

$$R_{\mathcal{T}}(\widehat{h}_{\mathcal{T}}) \;\leq\; \omega\big(\widehat{R}_{\mathcal{S}}(\widehat{h}_{\mathcal{S}})\big) \;+\; 2\sqrt{\frac{2\, d_{\mathcal{H}_{\mathcal{S}}} \log(2e m_{\mathcal{S}}/d_{\mathcal{H}_{\mathcal{S}}}) \;+\; 2\log(8/\delta)}{m_{\mathcal{S}}}}$$

$$+\; 4\sqrt{\frac{2\, d_{h_{\mathcal{S}} \circ \Pi} \log(2e m_{\mathcal{T}}/d_{h_{\mathcal{S}} \circ \Pi}) \;+\; 2\log(8/\delta)}{m_{\mathcal{T}}}}$$

[Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.]

But… !?

=> *No condition on the source!??*

However some transfer learning problems

appear to us **more easy than others**???

Interprétation

- Transfer acts as a bias

- We look for **hypotheses of the form**:

$$h_{\mathcal{S}} \circ \pi \qquad \text{with} \qquad \pi : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$$

  - **If** the **source hypothesis** is **well chosen**: the **bias** is well informed
  - **Otherwise**: Learning is **badly directed**

    or there is **over-fitting** if the capacity of $h_{\mathcal{S}} \circ \pi$ is too large

Analysis

- The **generalization properties** of TransBoost can be imported form the ones for **boosting**

$$\mathcal{H}_{\mathcal{T}} \;=\; \left\{ \mathtt{sign}\left[\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}} \circ \pi_n \right] \middle| \alpha_n \in \mathbb{R}, \pi_n \in \Pi, n \in [1, N] \right\}$$

$$d_{\mathtt{VC}}(\mathcal{H}_{\mathcal{T}}) \;\leq\; 2(d_{h_{\mathcal{S}} \circ \Pi} + 1)(N+1) \log_2\big((N+1)\, e\big)$$

$$R(h) \;\leq\; \widehat{R}(h) \;+\; \mathcal{O}\left(\sqrt{\frac{d_{h_{\mathcal{S}} \circ \Pi} \; \ln(m_{\mathcal{T}} / d_{h_{\mathcal{S}} \circ \Pi}) + \ln(1/\delta)}{m_{\mathcal{T}}}} \right)$$

"Authors also present some theory, but at the moment, again, it is essentially a trivial extension of boosting theory. **TL bounds should incorporate the quality of the source hypothesis**, e.g. the risk of the source on \mathcal{D}_T."

Outline

1. Classical inductive learning
2. Transfer learning
3. TransBoost: an original approach
4. Conclusion

Conclusions

- An **original** algorithm
  - **Simple**
  - **A single parameter**
  - Inherits the good properties of boosting
    - Control of the **learning error**
    - Control of the **test error**
  - The **learning problem** now becomes the one of choosing a **good projection space**

Conclusions

1. **Transfer** learning
   - Will gain importance
     - Long-life learning
     - Curriculum learning

2. **No good** encompassing theoretical framework
   - Still very **heuristic**
   - **Limited** theories
   - Interesting: **new theoretical questions**

3. An **original** perspective: TransBoost
   - A **new notion of capacity**: related to projections from target to source
   - The performance of the source hypothesis **does not enter the theory**!

Bibliography

- Ben-David, S., Lu, T., Luu, T., & Pál, D. (2010). Impossibility theorems for domain adaptation. In *International Conference on Artificial Intelligence and Statistics* (pp. 129-136).
- Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, *79*(1-2), 151-175.
- Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.
- Kuzborskij, I., & Orabona, F. (2013, February). Stability and hypothesis transfer learning. In *International Conference on Machine Learning* (pp. 942-950).
- Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv: 0902.3430*.
- Redko, I., Morvant, E., Habrard, A., Sebban, M., & Bennani, Y. (2019). *Advances in Domain Adaptation Theory*. Elsevier.
- H. Venkateswara, S. Chakraborty, and S. Panchanathan, "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 117–129, 2017.
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).
- Zhang, C., Zhang, L., & Ye, J. (2012). Generalization bounds for domain adaptation. In *Advances in neural information processing systems* (pp. 3320-3328).