Transfer Learning

Transferring the decision function

Analogy making and the MDLp

Antoine Cornuéjols

AgroParisTech – INRAE UMR MIA Paris-Saclay

antoine.cornuejols@agroparistech.fr

When $P_{Y|X}(\text{train}) \neq P_{Y|X}(\text{test})$

(and, not necessarily) $P_X(\text{train}) \neq P_X(\text{test})$

Concept shift

and sequences of concept shifts

Outline

1. Transfer learning: questions
2. TransBoost: an algorithm and what it tells on the role of the source
3. The MDLP and analogy making
4. Conclusions

Transfer learning

Questions (more of them)

- 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?
- Should the proximity between the **source** and the **target** 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?

I.I.D. learning: Bounds between the **real** risk and the **empirical** risk

By removing the “problematic” examples, you go

- From the **non realisable** case ($\mathcal{H}$ finite)

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2\,m}} \right] > 1 - \delta$$

- To the **realisable** one ($\mathcal{H}$ finite)

$$\forall h \in \mathcal{H}, \forall \delta \leq 1 : \quad P^m \left[R_{\text{Réel}}(h) \leq R_{\text{Emp}}(h) + \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{m} \right] > 1 - \delta$$

Which **link** between **training** and **testing**?

Transfer Learning

Transfer Learning

- Guarantees function of

Transfer Learning

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T

Transfer Learning

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T
  - A "**distance**" between the source task and the target one
    - The **smaller** the distance, the **better** the transfer

Transfer Learning

Really?

- Guarantees function of
  - The **quality** of the **source hypothesis** on the source task
    - The **better** h_S, the **better** h_T
  - A "**distance**" between the source task and the target one
    - The **smaller** the distance, the **better** the transfer
  - The size of the **target training data**
    - The **larger** the target training data set, the **useless** the transfer

Outline

1. Transfer learning: questions
2. TransBoost: an algorithm and what it tells on the role of the source
3. The MDLP and analogy making
4. Conclusions

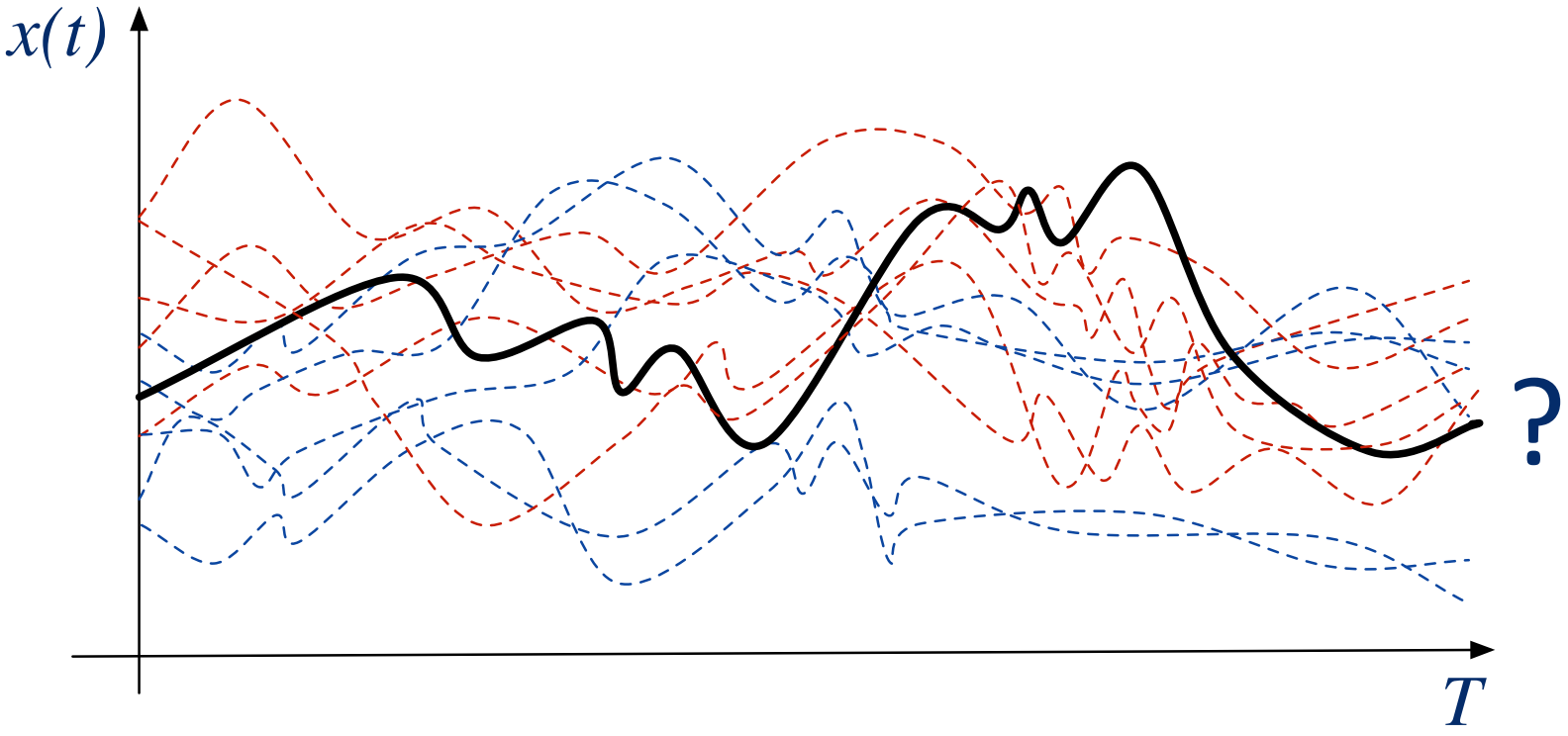
TransBoost: an algorithm for **transfer learning**

And what it tells about the **role of the source**

Cornuéjols, A. (2024). **Some thoughts about Transfer learning. What role for the source domain**. *International journal of Approximate Reasoning (IJAR)*, vol. 166, p.109107. Elsevier.

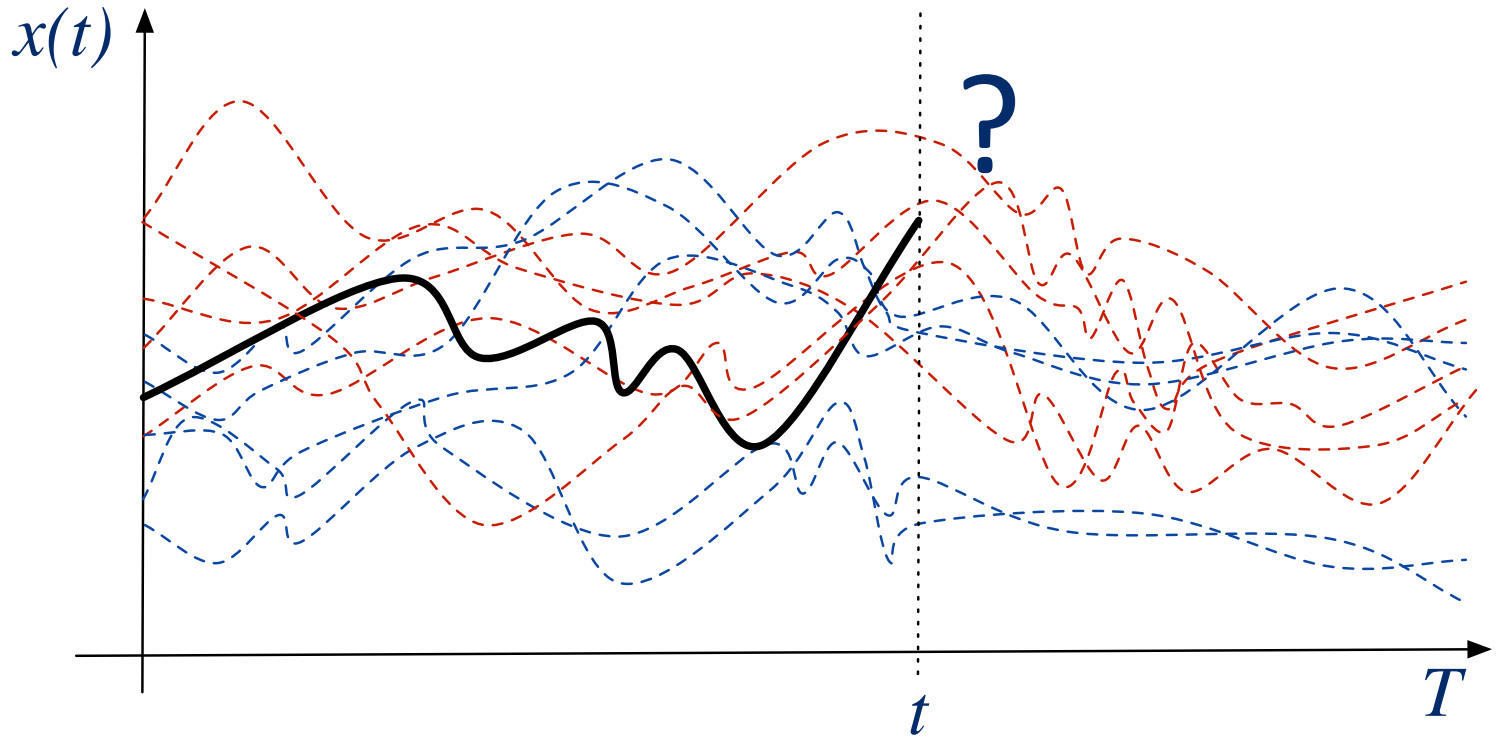
Standard classification of time series

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



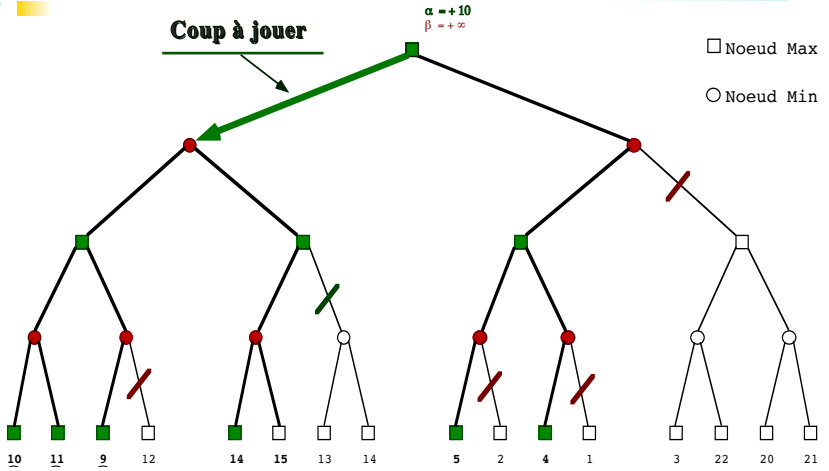
Early classification of time series

- What is the class of the new **incomplete** time series x_t?



A **LUPI** type of algorithm for **transfer** learning

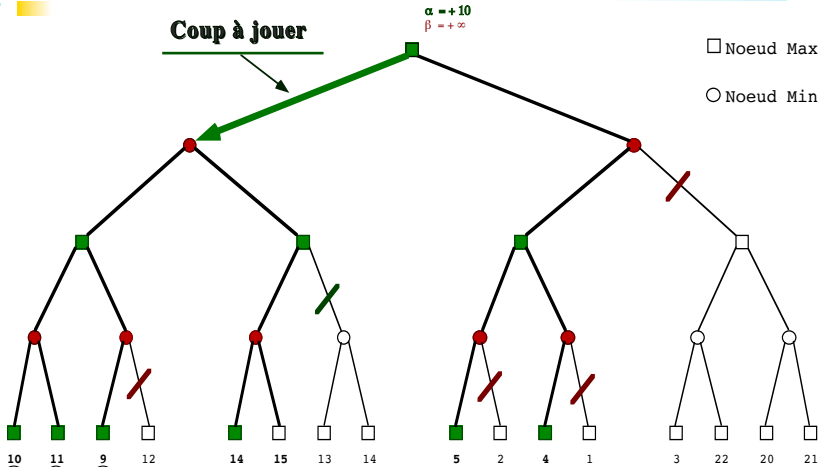
Taking decision when the current information is **incomplete**



...

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

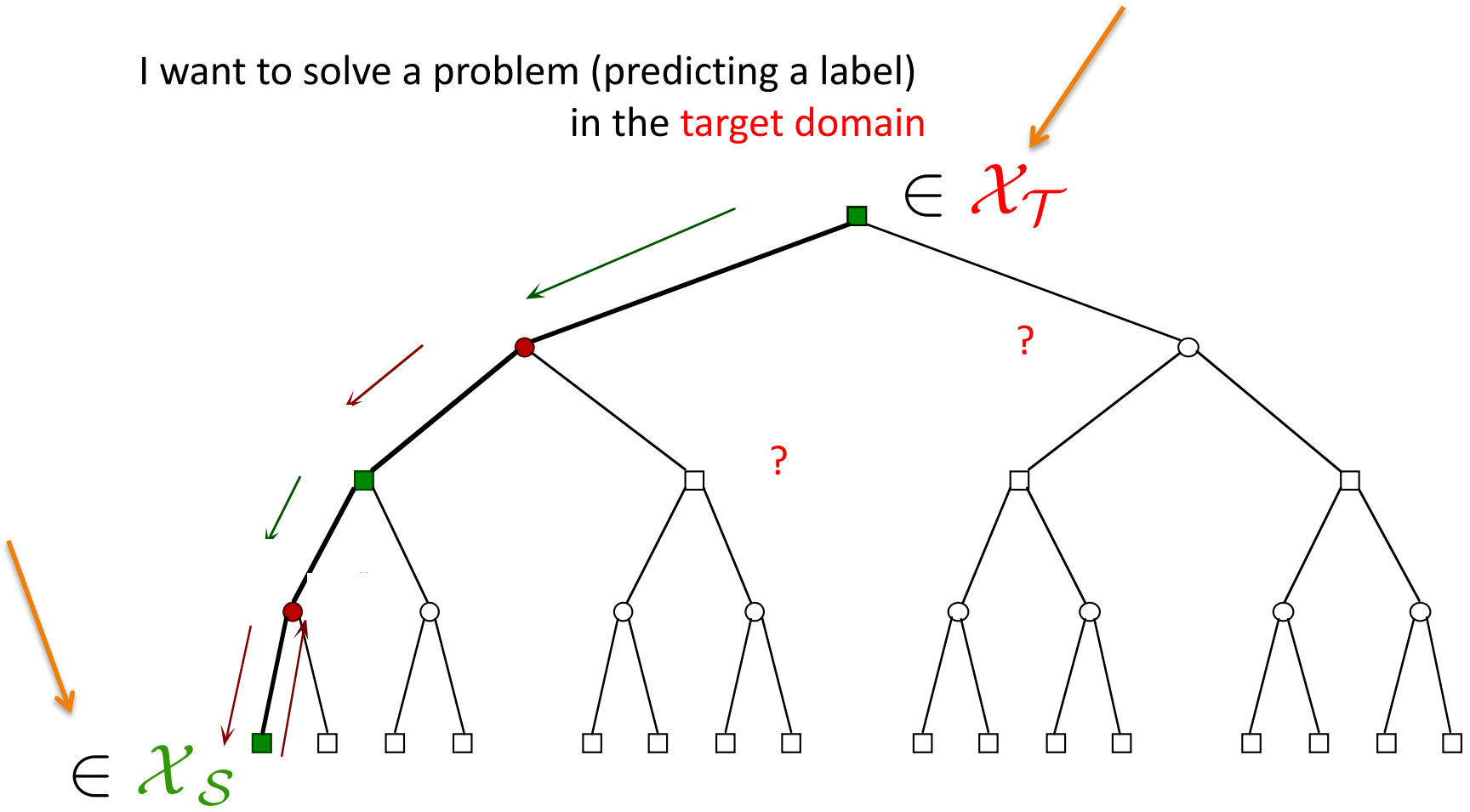
*"Experts" may live in **input spaces** that are **different***

Can we do the “same” for transfer learning?

A **new perspective** on transfer learning

...

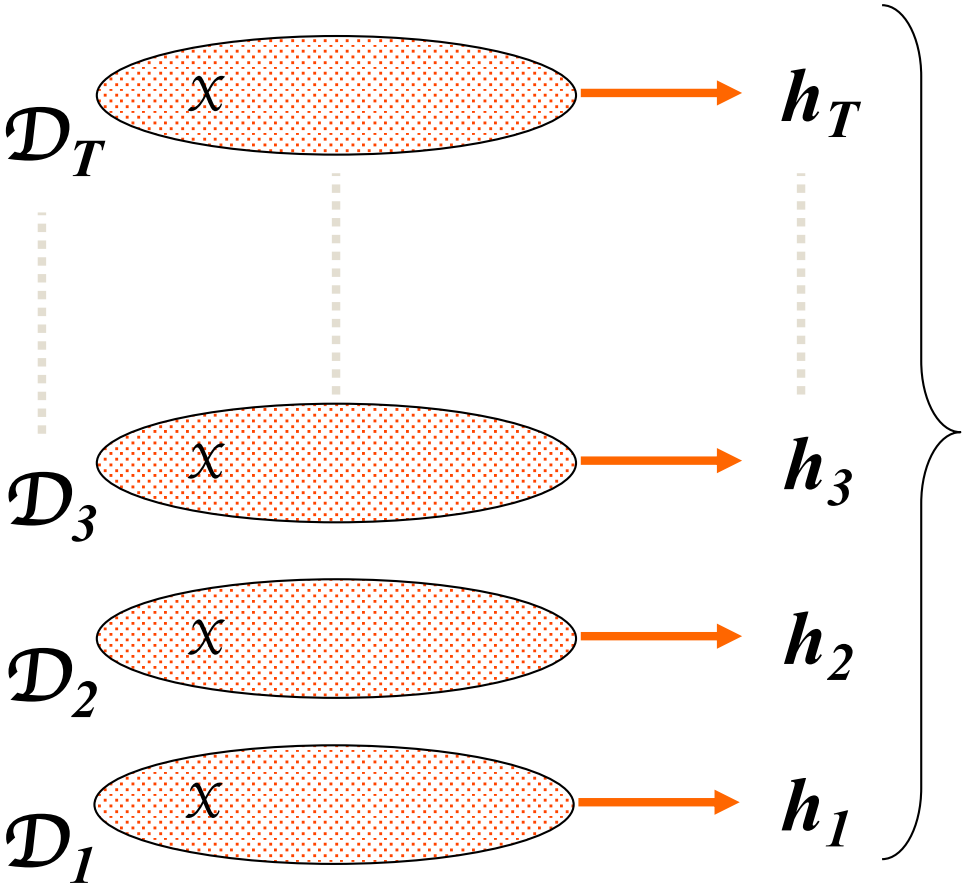
I want to solve a problem (predicting a label)
in the target domain



We know how to solve a problem in the source domain using the **source hypothesis**

- How to proceed?
  - How to **find** useful answers?
  - How to **combine** them?

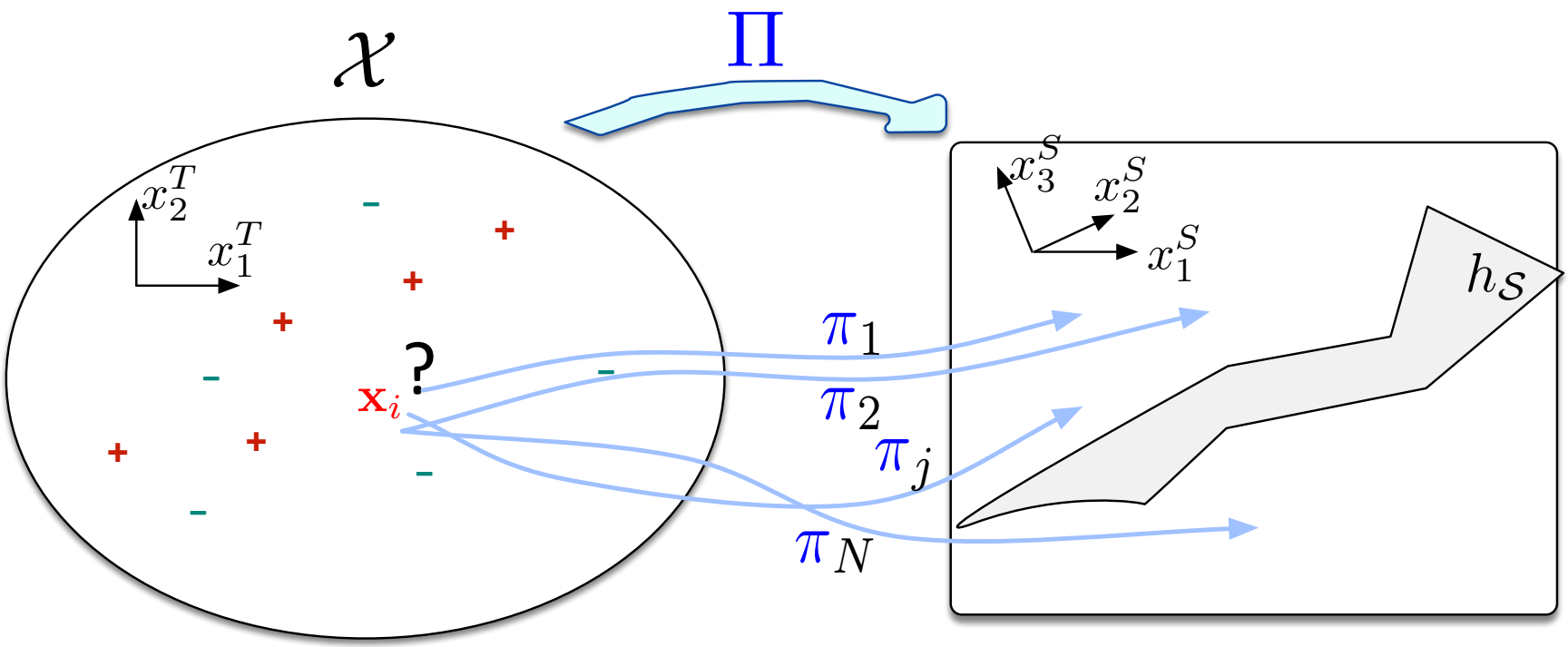
Boosting



$$H(\mathbf{x}) = \text{sign}\left[\sum_{t=1}^{T} \alpha_t \, h_t(\mathbf{x})\right]$$

- How to compute $\mathcal{D}_t$ from $\mathcal{D}_{t-1}$ and thus h_t?
- How to compute the α_t ?

TransBoost



***Target** Domain*

***Source** Domain*

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

TransBoost

- Principle:
  - Learn "*weak projections*": $\pi_i : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$
    - Using the **target training data**: $S_{\mathcal{T}} = \{(\mathbf{x}_i^{\mathcal{T}}, y_i^{\mathcal{T}})\}_{1 \leq i \leq m}$

TransBoost

- Principle:
  - Learn "*weak projections*": $\pi_i : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{X}_{\mathcal{S}}$
    - Using the target training data: $S_{\mathcal{T}} = \{(\mathbf{x}_i^{\mathcal{T}}, y_i^{\mathcal{T}})\}_{1 \leq i \leq m}$
  - With 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 $$H_{\mathcal{T}}(\mathbf{x}^{\mathcal{T}}) = \operatorname{sign}\left\{\sum_{n=1}^{N} \alpha_n \, h_{\mathcal{S}}\left(\pi_n(\mathbf{x}^{\mathcal{T}})\right)\right\}$$

TransBoost

Algorithm 1: Transfer learning by boosting

Input: $h_{\mathcal{S}} : \mathcal{X}_{\mathcal{S}} \to \mathcal{Y}_{\mathcal{S}}$ 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 $\pi_i : \mathcal{X}_{\mathcal{T}} \to \mathcal{X}_{\mathcal{S}}$ 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 $i = 1 \ldots, m$:

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

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

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

end

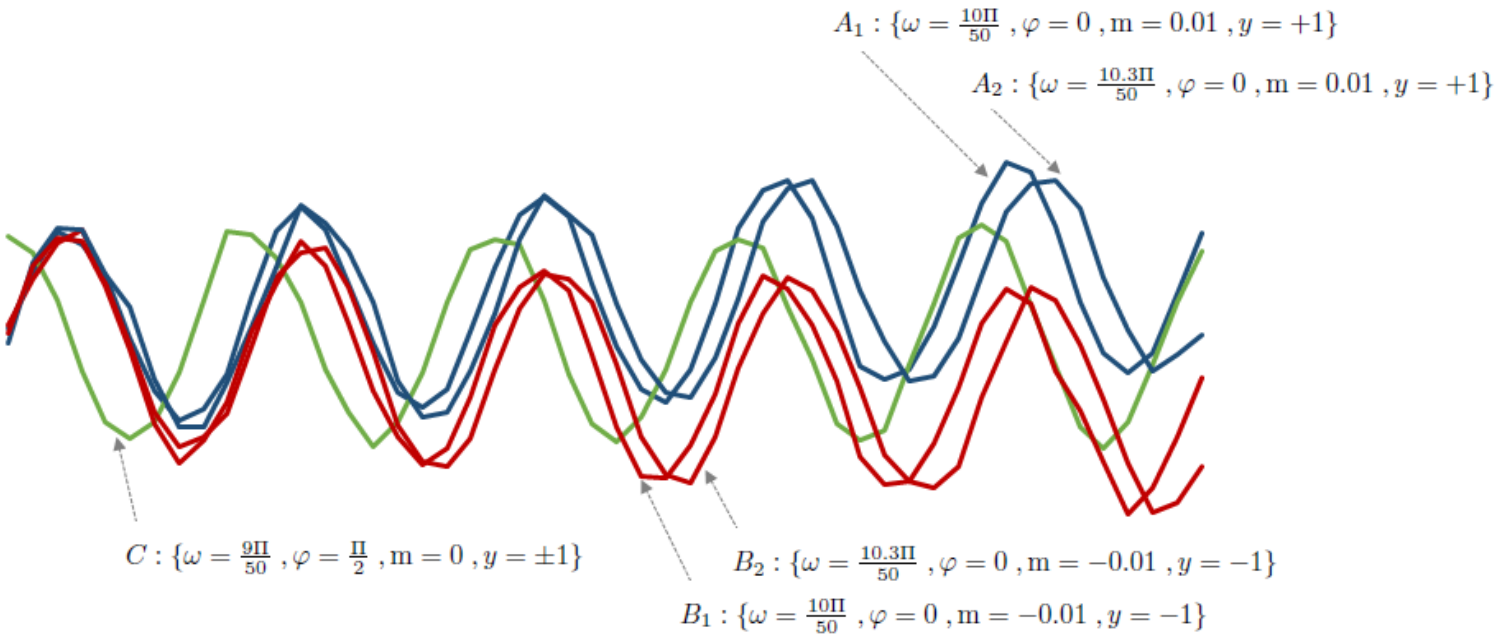
Output: the final target hypothesis $H_{\mathcal{T}} : \mathcal{X}_{\mathcal{T}} \to \mathcal{Y}_{\mathcal{T}}$:

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

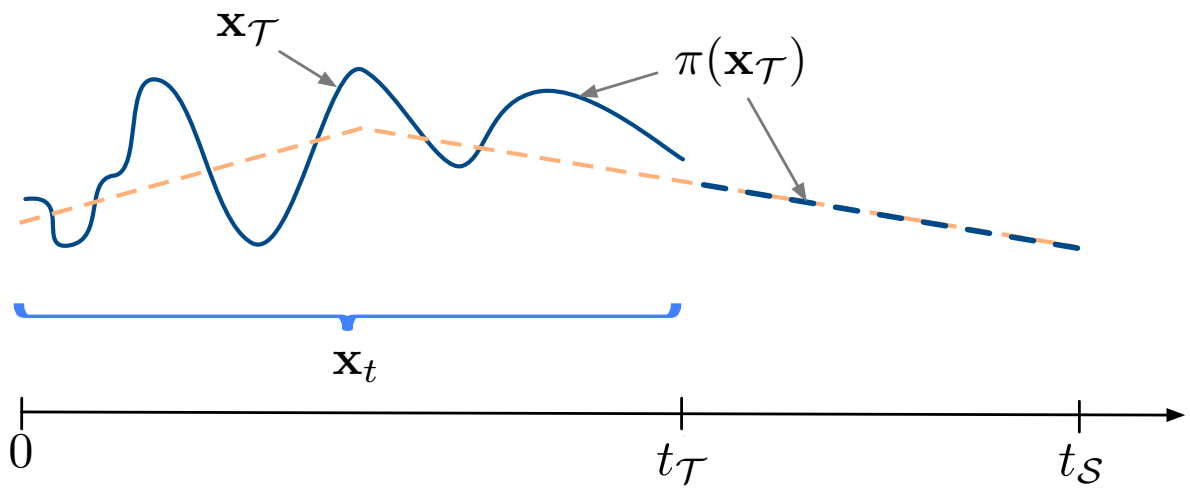
Controlled data

- The **slope** to distinguish between **classes**
- The **shapes** of time series within each class: variety
- The **noise level**

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



The set of projections



Example of a projection π (a hinge function with three parameters):

- the first slope,
- the second one
- and the time of the hinge) that is adjusted to the target exemple x_T by least square.

The resulting projection $\pi(x_T)$ is the concatenation of x_T and the remaining part of the adjusted hinge function.

Results

Learning from **target data only**

TransBoost

Naïve transfer

First a projection from X_T to X_S by SVR then using h_S

Very little information in the source

Increasing information in the source

Increasing level of noise

slope, noise, $t_{\mathcal{T}}$	SVM (test)	$H_{\mathcal{T}}$ (train)	$H_{\mathcal{T}}$ (test)	SVR+SVM (test)
0.001, 0.001, **20**	0.50 ± 0.08	0.08 ± 0.03	**0.08** ± 0.02	0.49 ± 0.01
0.005, 0.001, **20**	0.49 ± 0.01	0.01 ± 0.01	**0.01** ± 0.01	0.45 ± 0.01
0.005, 0.002, **20**	0.49 ± 0.03	0.03 ± 0.02	**0.04** ± 0.02	0.43 ± 0.01
0.005, 0.020, **20**	0.48 ± 0.03	0.09 ± 0.01	**0.10** ± 0.01	0.47 ± 0.01
0.001, 0.200, **20**	0.50 ± 0.01	0.46 ± 0.02	0.51 ± 0.02	0.49 ± 0.01
0.010, 0.200, **20**	0.47 ± 0.03	0.34 ± 0.02	0.35 ± 0.02	0.35 ± 0.01
0.001, 0.001, **50**	0.50 ± 0.01	0.08 ± 0.03	**0.08** ± 0.02	0.41 ± 0.01
0.005, 0.001, **50**	0.28 ± 0.09	0.01 ± 0.01	**0.01** ± 0.01	0.28 ± 0.01
0.005, 0.002, **50**	0.30 ± 0.08	0.02 ± 0.01	**0.02** ± 0.01	0.28 ± 0.01
0.005, 0.020, **50**	0.30 ± 0.08	0.04 ± 0.01	**0.04** ± 0.01	0.31 ± 0.01
0.001, 0.200, **50**	0.50 ± 0.01	0.38 ± 0.03	0.44 ± 0.02	0.43 ± 0.01
0.010, 0.200, **50**	0.12 ± 0.04	0.10 ± 0.02	0.11 ± 0.02	0.15 ± 0.02
0.001, 0.001, **100**	0.47 ± 0.03	0.07 ± 0.02	**0.07** ± 0.02	0.23 ± 0.01
0.005, 0.001, **100**	0.07 ± 0.03	0.01 ± 0.01	**0.01** ± 0.01	0.07 ± 0.02
0.005, 0.002, **100**	0.10 ± 0.04	0.02 ± 0.01	**0.02** ± 0.01	0.07 ± 0.01
0.005, 0.020, **100**	0.09 ± 0.03	0.02 ± 0.01	**0.03** ± 0.01	0.07 ± 0.01
0.001, 0.200, **100**	0.46 ± 0.02	0.28 ± 0.02	0.31 ± 0.01	0.31 ± 0.01
0.010, 0.200, **100**	0.05 ± 0.02	0.04 ± 0.01	0.05 ± 0.01	0.05 ± 0.01

Remarkably **no overfitting** anywhere

Lots of information in the source and lots of noise

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	**0.08** ± 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	**0.01** ± 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	**0.04** ± 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	**0.10** ± 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	**0.08** ± 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	**0.01** ± 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	**0.02** ± 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	**0.04** ± 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	**0.07** ± 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	**0.01** ± 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	**0.02** ± 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	**0.03** ± 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

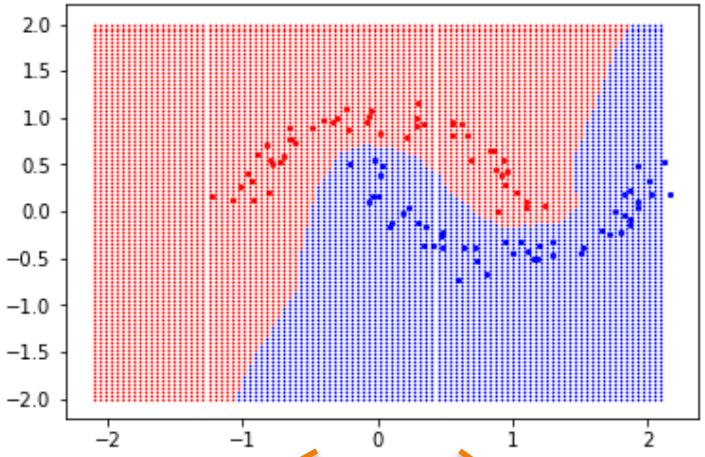
High noise level

Easy

Large slope

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.

Transfer learning using Transboost

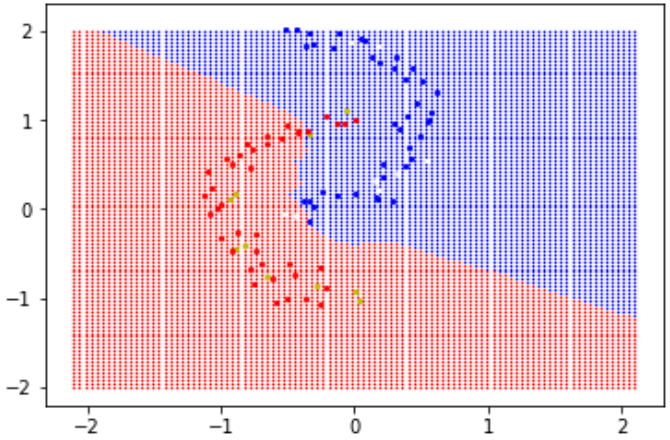


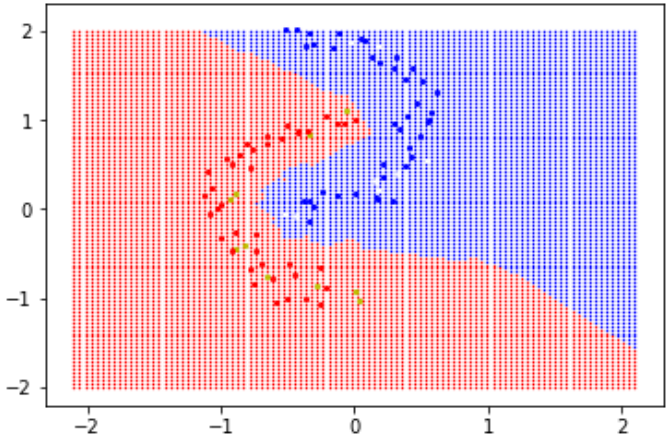
$$\pi_i(\mathbf{x}) = \mathbf{x} + \mathbf{v}_i$$

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

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

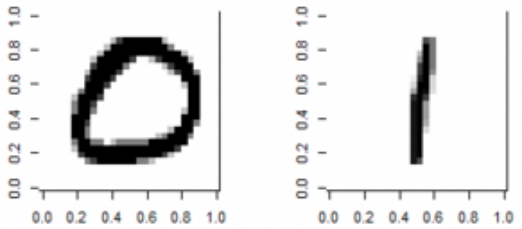
Using Transboost



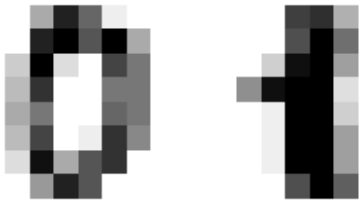


Transfer learning using Transboost

- Illustrations

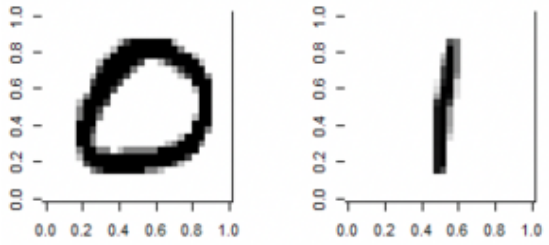


(a) Is it a zero or a one ?

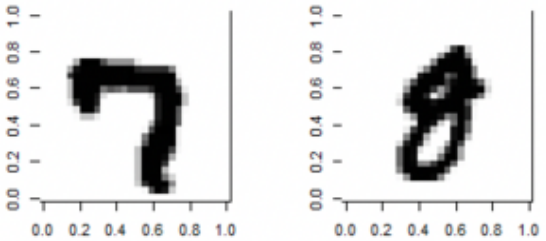


(b) Is it a zero or a one ?

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 using Transboost

- Illustrations

Task A

$\mathcal{X}_A \neq \mathcal{X}_B$

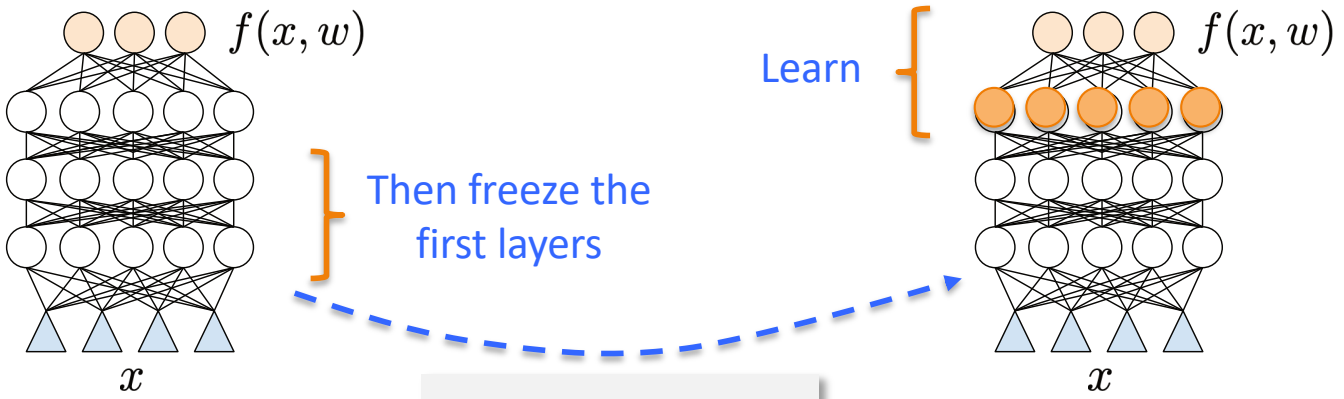
Figure 1: Trained model on the data source : is it a picture of a dog or a cat ?

Task B

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

Standard Transfer with NNs

...



Learn NN on task A

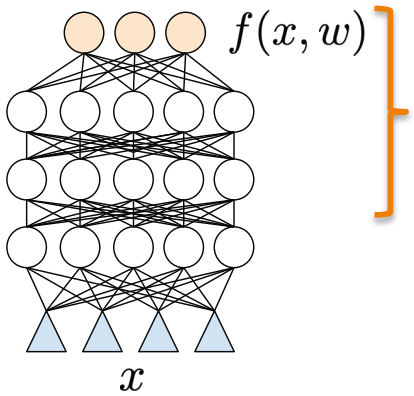
Learn the **last** layers on task B

Same input space $\mathcal{X}_A = \mathcal{X}_B$

From Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). **Learning and transferring mid-level image representations using convolutional neural networks**. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1717-1724).

TransBoost with NNs

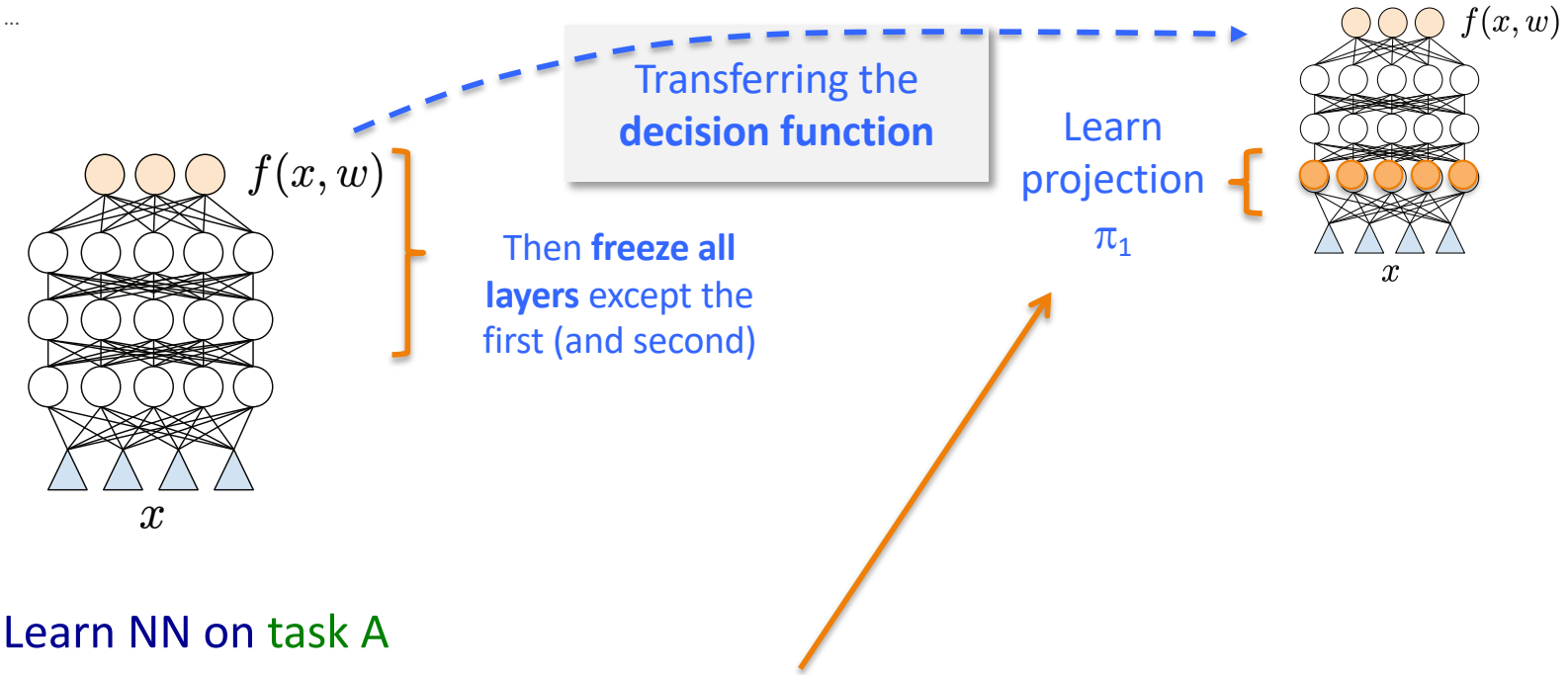
...



Then **freeze all layers** except the first (and second)

Learn NN on task A

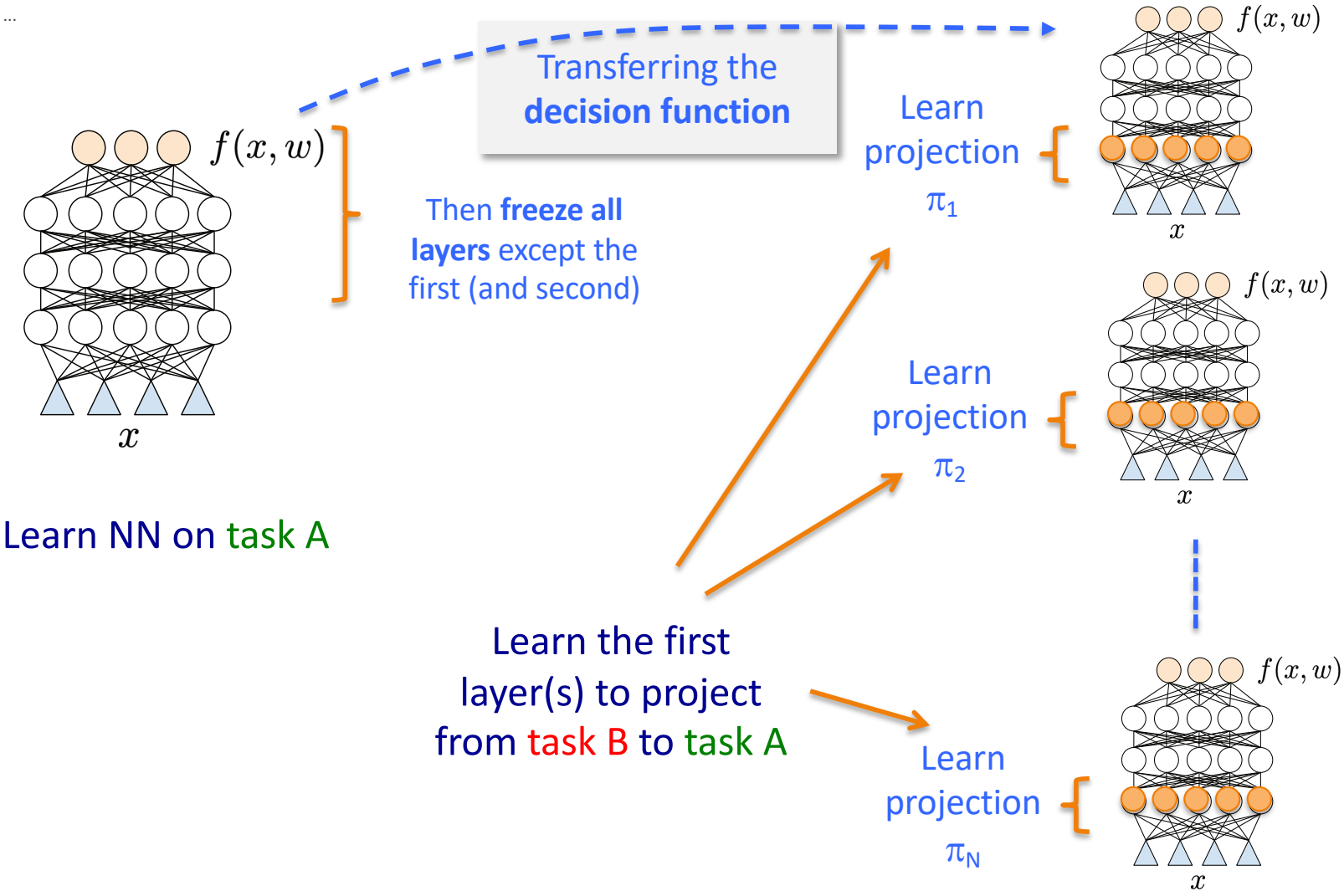
TransBoost with NNs



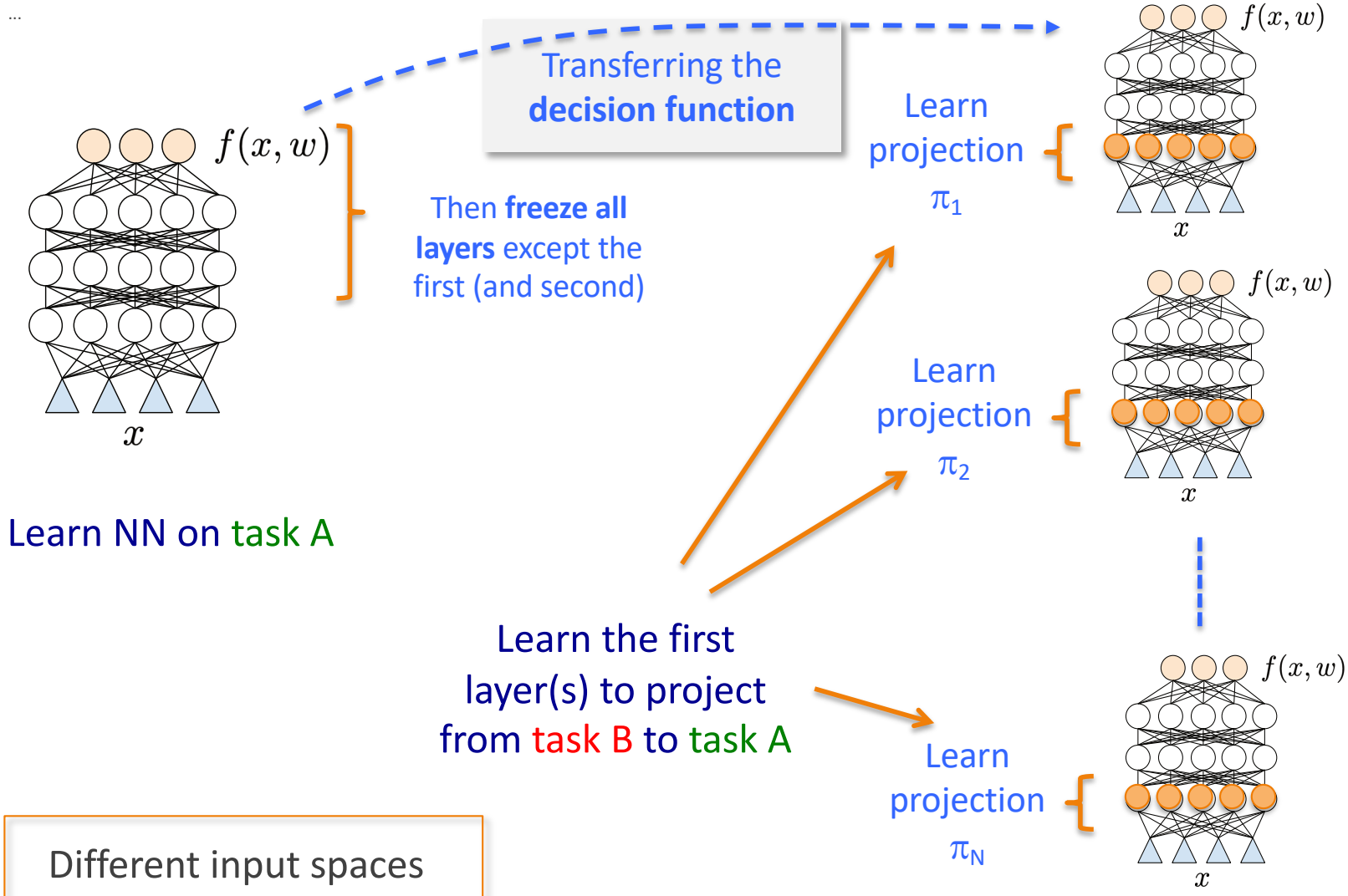
Learn NN on task A

Learn the first layer(s) to project from task B to task A

TransBoost with NNs



TransBoost with NNs



Different input spaces

$$\mathcal{X}_A \neq \mathcal{X}_B$$

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

Does the quality of h_S plays a **role**?

What if ...

Source hypothesis a priori **without relation** to the target task

Learning **from target data only** | TransBoost with **"irrelevant"** 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	$\mathbf{0.03} \pm 0.01$
Hard	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

Very good results!!

h_S **randomly chosen** on the source task $\quad \widehat{R}(h_{\mathcal{S}}) \approx 0.5$

Does the quality of h_S plays a **role**? NO!!

What is the **role** of h_S??

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

But… !?

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

Still some transfer learning problems

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

Interpretation

Transfer acts as a bias and h_S is a strong part of this bias

- **If** the **source hypothesis** is **well chosen**: the **bias** is well informed
  - Which **does not mean** that h_S must be good on the source task
- **Otherwise**: Learning is **badly directed**

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

Lessons

- The learning problem now becomes the problem of **choosing** a good set of (weak) projections

- Theoretical guarantees exist

Analysis

- The **generalization properties** of TransBoost

  can be imported from the ones for **boosting**

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

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

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

Theory for HTL

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

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

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 $\{h_{src}^i : \|h_{src}^i\|_\infty \leq 1\}_{i=1}^n$, 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 $1 - e^{-\eta}$, $\forall \eta \geq 0$:*

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

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

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

Analysis

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

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

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

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

"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."

Theoretical guarantees

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

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

Theorem 1. *Let $\omega : \mathbb{R} \to \mathbb{R}$ 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 $\widehat{\pi} := \text{ArgMin}_{\pi \in \Pi}\, \widehat{R}_{\mathcal{T}}(\widehat{h}_{\mathcal{S}} \circ \pi)$, be the best apparent projection.*

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

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

[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.]

Theoretical guarantees

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

Ridiculous

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

Irrelevant

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

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

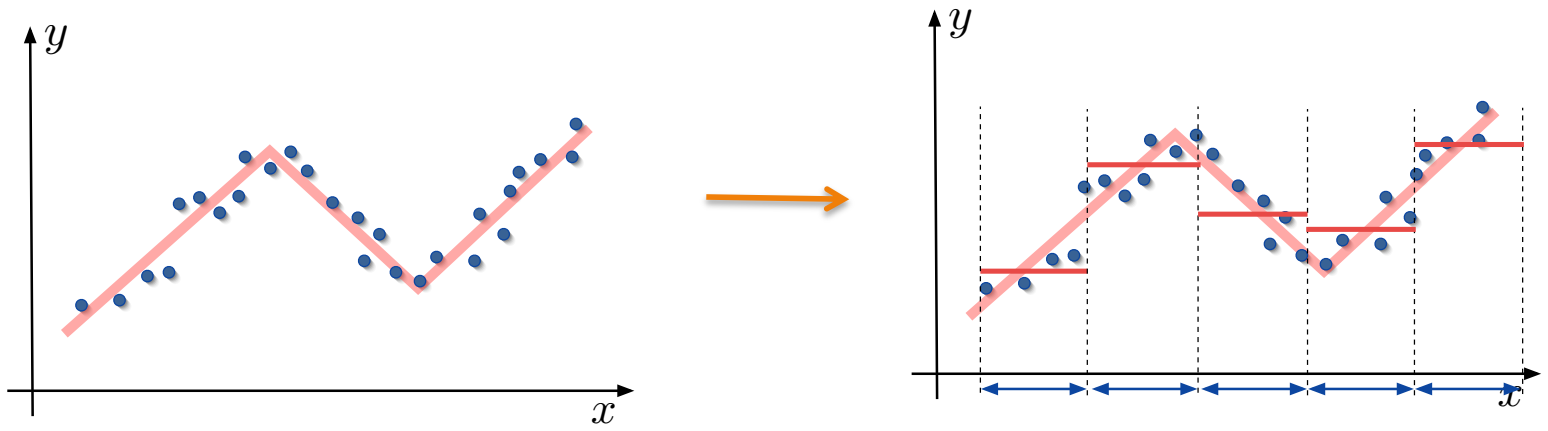
[Cornuéjols A., Murena P-A. & Olivier R. "*Transfer Learning by Learning Projections from Target to Source*". Symposium on Intelligent Data Analysis (IDA-2020), April 27-29 2020, Bodenseeforum, Lake Constance, Germany.]

A relationship with **tracking**?

Tracking

Instead of **learning a complex function** over the whole of $\mathcal{X}$

- **If** you know that the task is slowly evolving with time
- Learn a simple **local** function

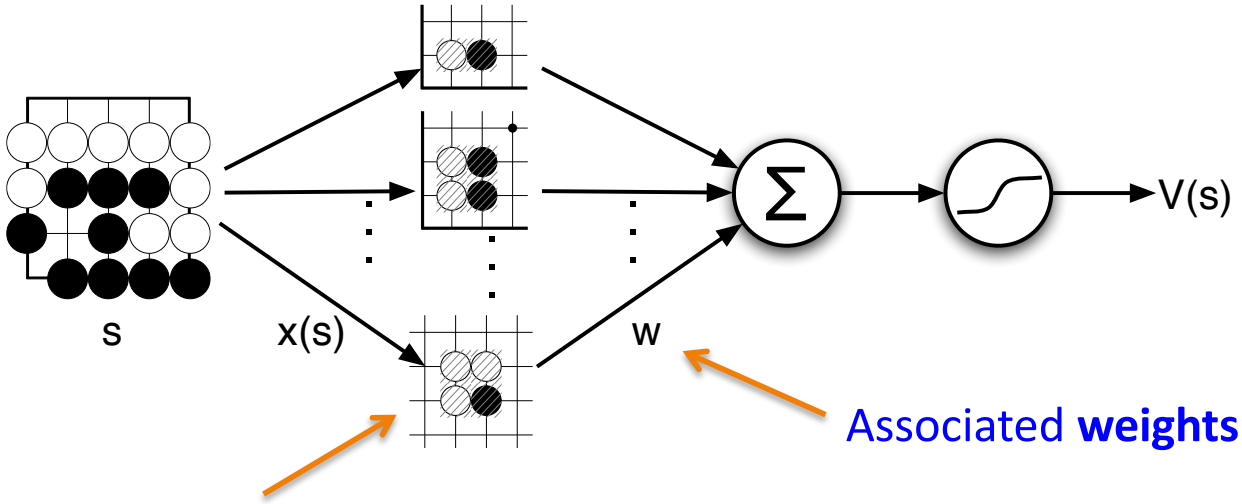


R. Sutton and A. Koop and D. Silver (2007) *"On the role of tracking in stationary environments"* (ICML-07) Proceedings of the 24th international conference on Machine learning, ACM, pp.871-878, 2007.

Tracking in stationary environments

Tracking to play Go

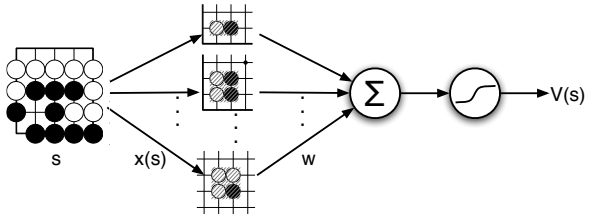
- 5 x 5 Go
  - More than 5 x 10^{10} unique positions
- Usual approach: learn a **general** evaluation function V(*s*) valid $\forall s$

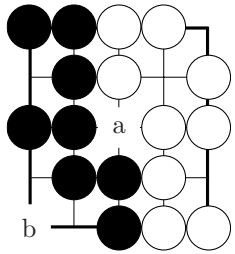


Tracking in stationary environments

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

  local to the current *s*

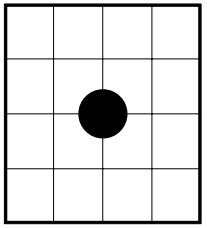




In general, playing (a) (center) is better than playing (b)

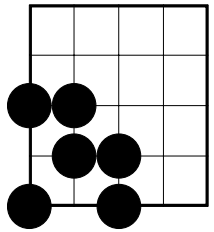
BUT

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

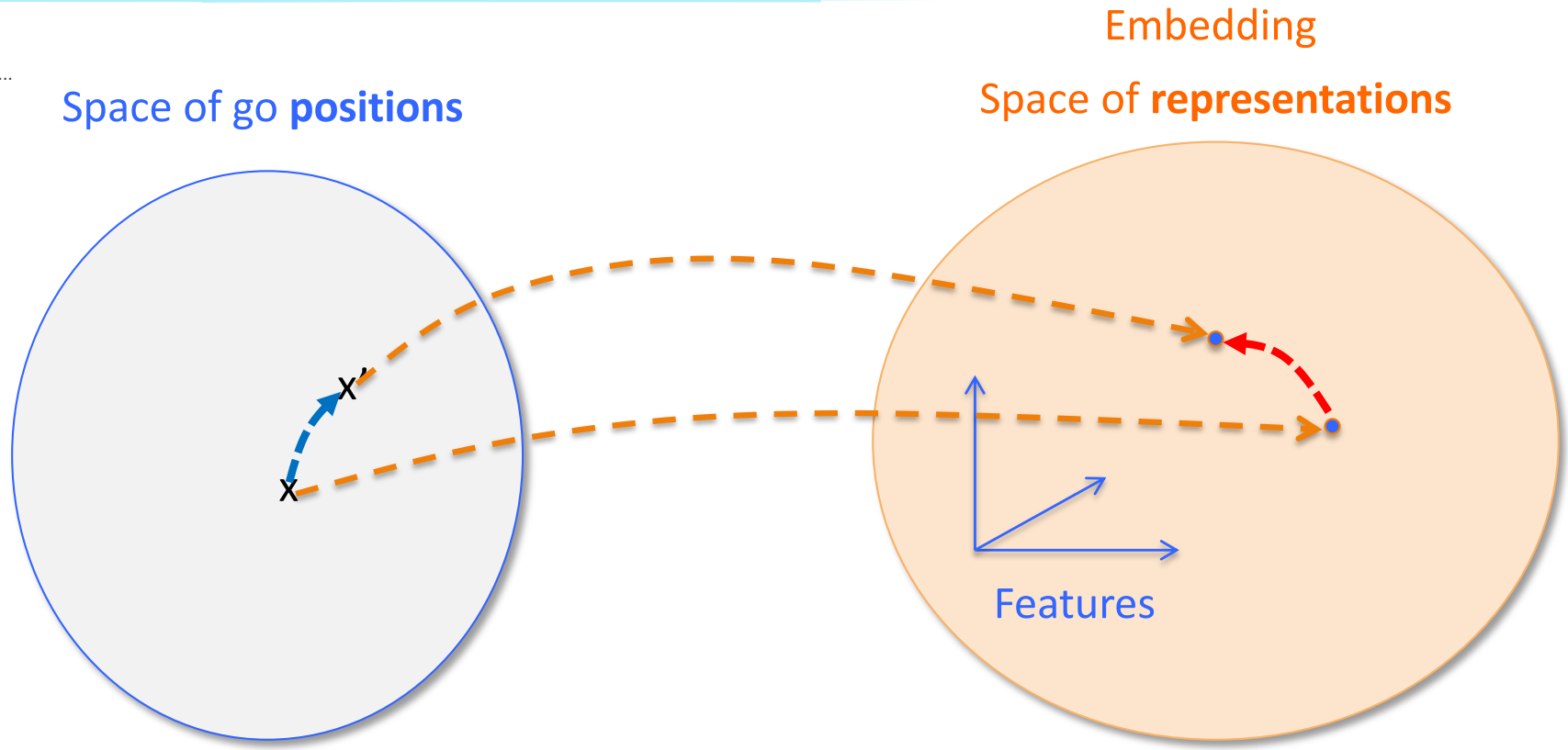


More weight

More weight



Tracking as **local changes** of representation



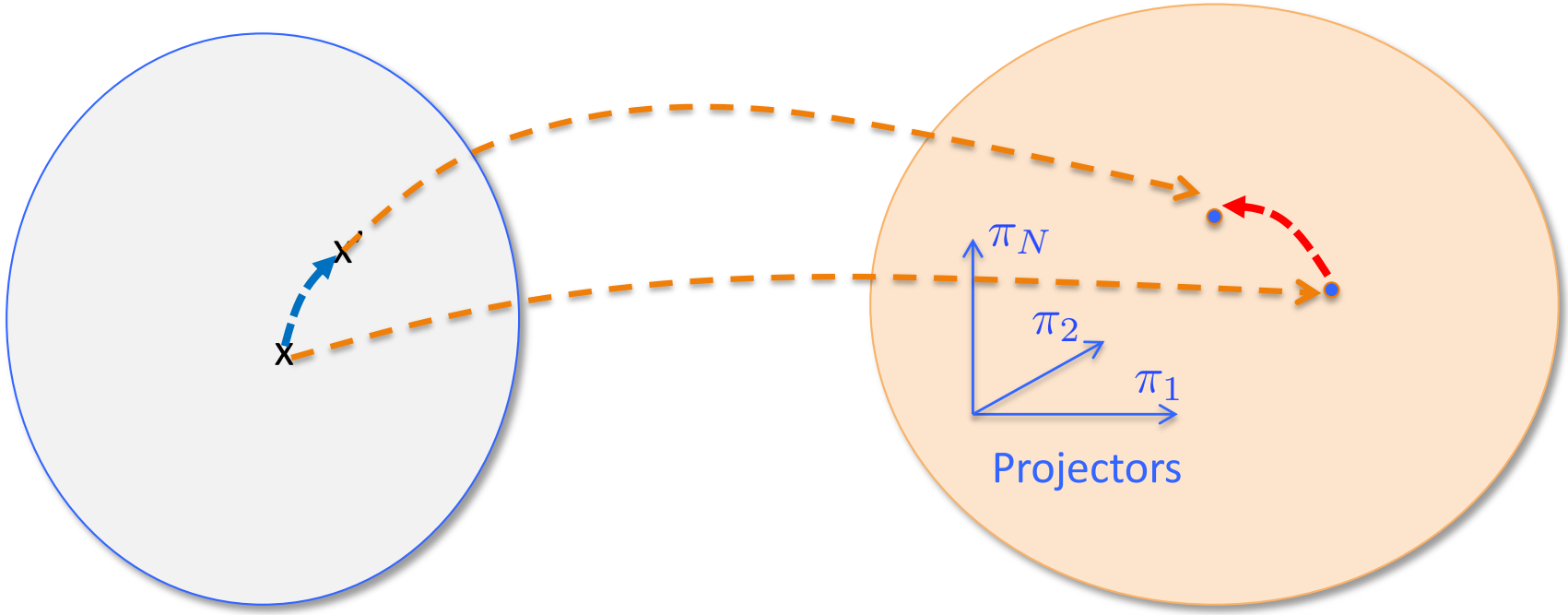
*The **weights** of the features*
***change** with the evolving position*

Weight ↘

Weight ↗

Transboost as **local changes** of representation

...



Space of learning tasks

Target training sets

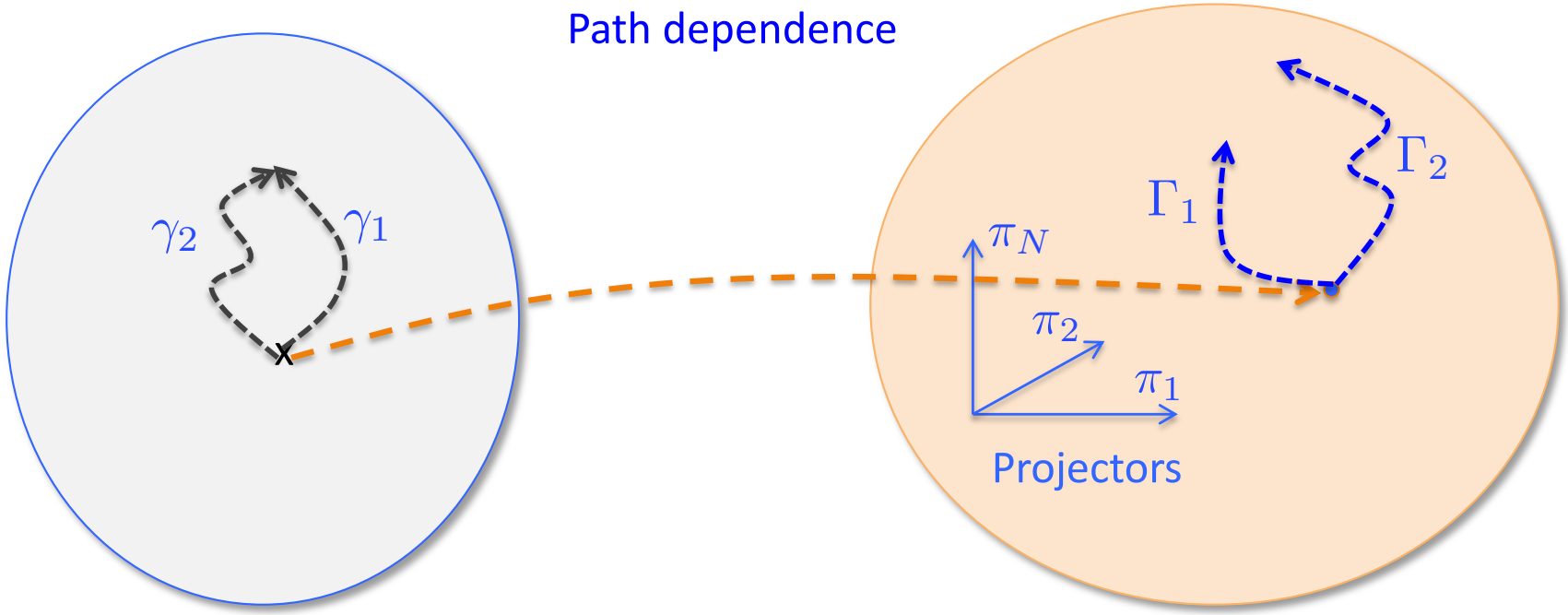
Embedding

Space of **projectors** π_i

Weighted projectors

Transboost as **local changes** of representation

...



Space of learning tasks

Target training sets

Embedding

Space of **projectors** π_i

Weighted projectors

Conclusions

- The **theoretical guarantees** for transfer learning:
  - Do not necessarily depend on the **performance of the source hypothesis** h_S

    But depend on the **bias** that h_S determines
  - Involve the **capacity** of the space of **transformations**

    (and **the path** followed between source and target)

Still to be explored

- Idea of minimizing a **distance** between the "local" models

What kind of **distance** ?

Outline

1. Transfer learning: questions
2. TransBoost: an algorithm and what it tells on the role of the source
3. The MDLP and analogy making
4. Conclusions

Kolmogorov's complexity

Andreï Kolmogorov
(1903 – 1987)

Complexity of a sequence =

Size in bits of the **smallest program**

that can generate that sequence

$$K_M(x) = \min_{p \in P_M} \{l(p), s(p) = x\}$$

x : the sequence

P_M : program coded on machine M

l(p) : size of p

Kolmogorov's complexity

- True randomness
  - No structure
  - Smallest program = the sequence itself

- Pi
  - Lots of structure, very simple!

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

  - Infinite sequence of integers → but a small program

Solomonoff's induction

Ray Solomonoff
(1926 - …)

- *Look* for the smallest program
  that can generate a given sequence
  - Almost all induction problems can be cast as the prediction of a binary sequence
- Unfortunately, this is **NOT computable**…
  - Even if it exists, it is not possible to find it in the general case *(Gödel's theorem, stopping problem, …)*
- It is possible to approximate it

Minimum Description Length Principle (MDLP)

- The **best hypothesis** (given training data) is the one that **minimize** the sum of

  1. The **length** in bits of the description of the **hypothesis**

  2. The **length** in bits of the description of the **data given the hypothesis**

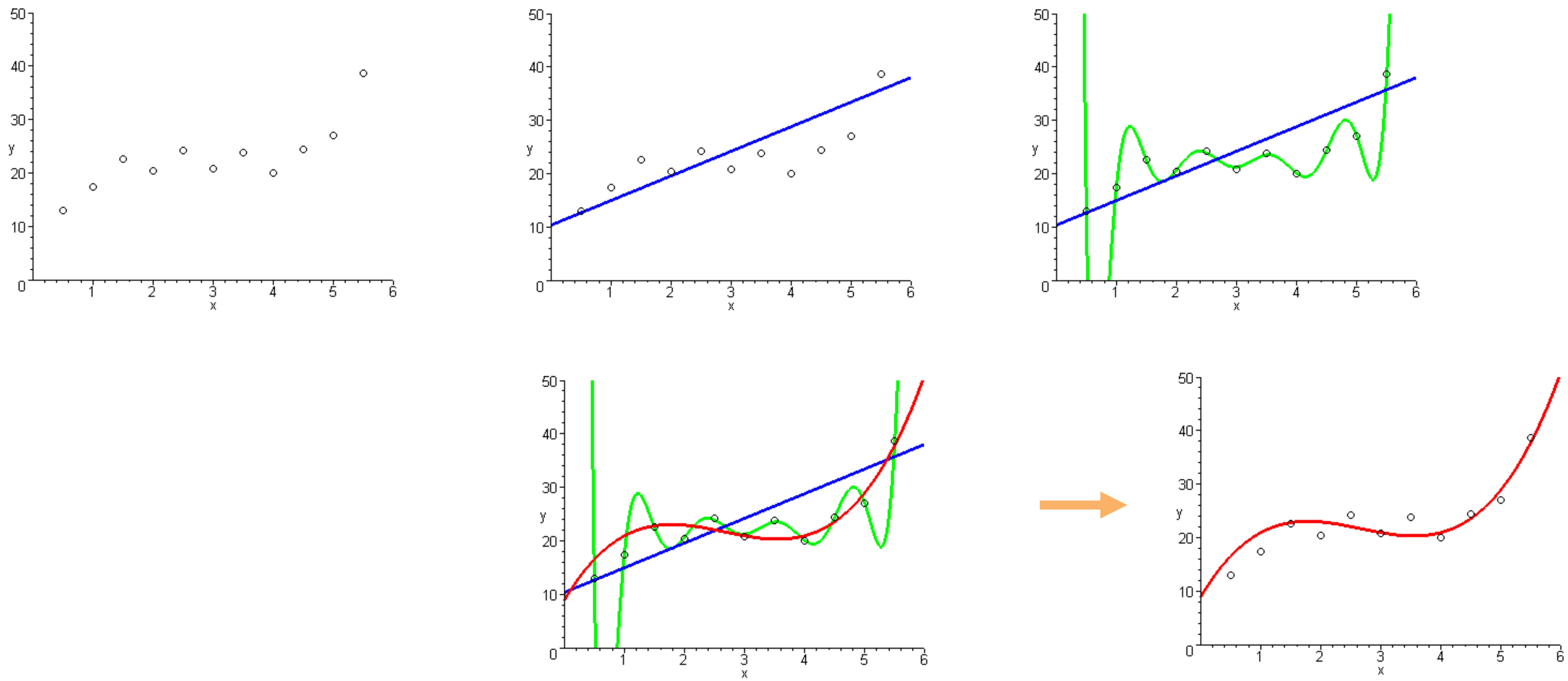
$$h^\star = \underset{h \in \mathcal{H}}{\text{ArgMin}} \{ \underbrace{L(h)} + \underbrace{L(\mathcal{S}|h)} \}$$

Strong relationship with $P(h|\mathcal{S}) = \dfrac{P(h) \times P(\mathcal{S}|h)}{P(\mathcal{S})}$

$$h^\star = \underset{h \in \mathcal{H}}{\text{ArgMax}} \; P(\mathcal{S} \mid h) \; P(h)$$

Example: regression

- **Complexity** of model:
  - the **degree** of a polynomial (to be described up to a given precision)
- **Error**
  - The size of the **corrections** wrt to the predictions



Minimum Description Length Principle (MDLP)

You have to define a code with which to describe the hypothesis and the data

⟷ a **bias** (prior knowkedge)

$$h^\star = \underset{h \in \mathcal{H}}{\mathrm{ArgMin}} \{ L(h) + L(\mathcal{S}_m | h) \}$$

- **Multi-task** learning
  - **Simultaneous** learning phases

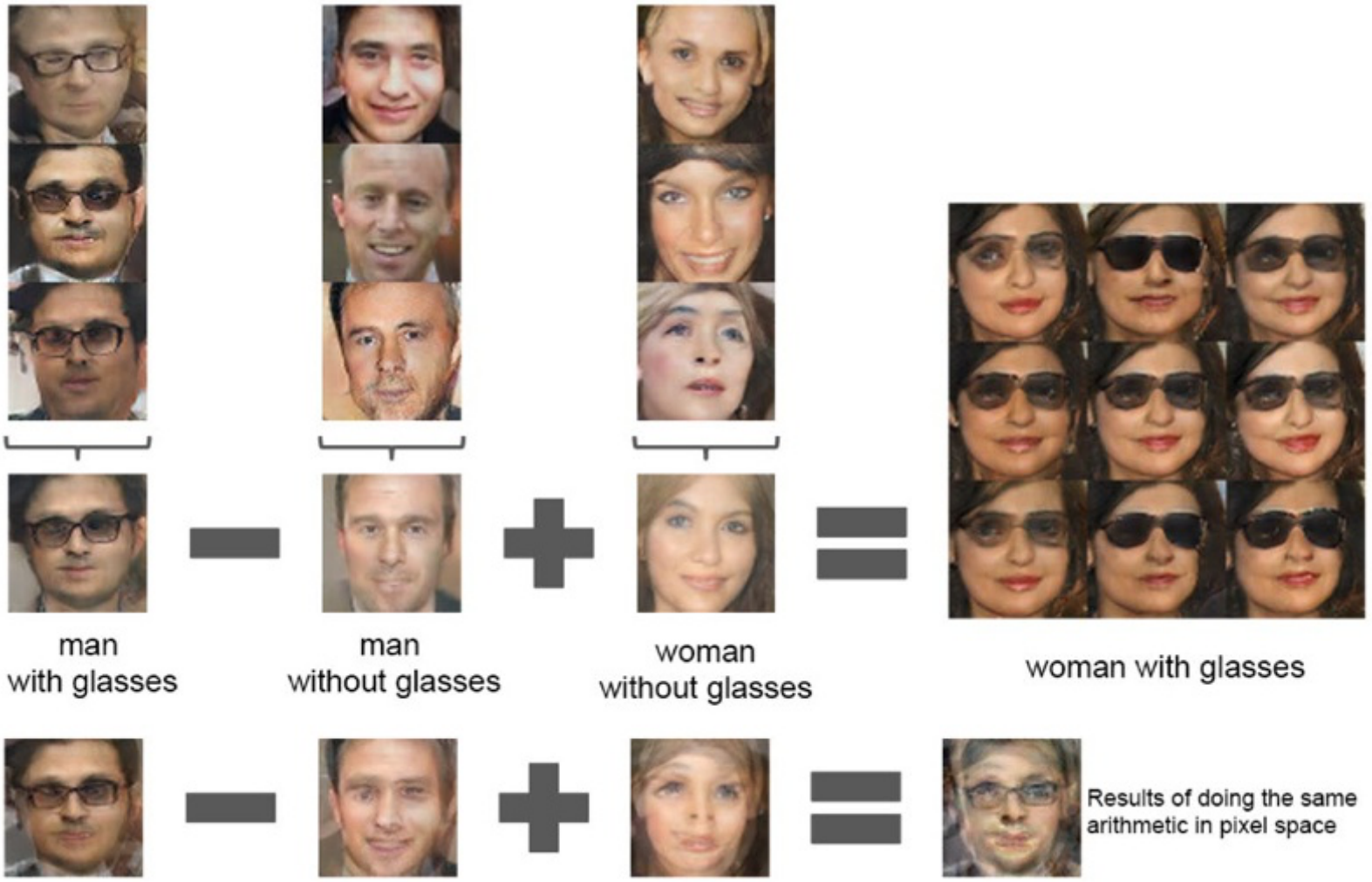
Maximizing the **agreement** between learners

- **Transfer** learning
  - **Successive** learning phases

Maximizing the **agreement** (**??**) between learners

Analogy making

Analogy in an **embedding** space



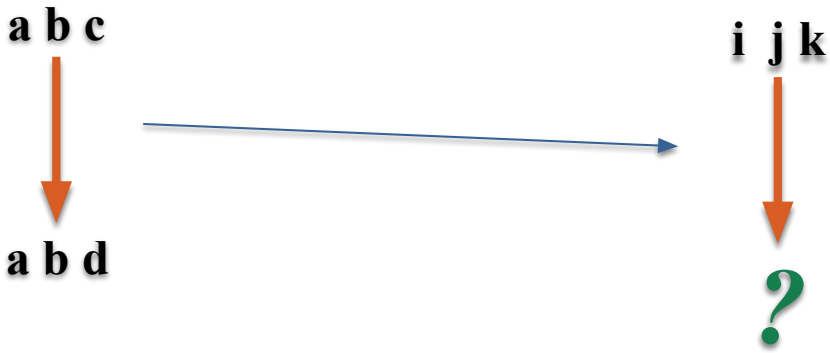
From [Bishop, “Deep learning: Foundations and concepts”, (2024)] p.543

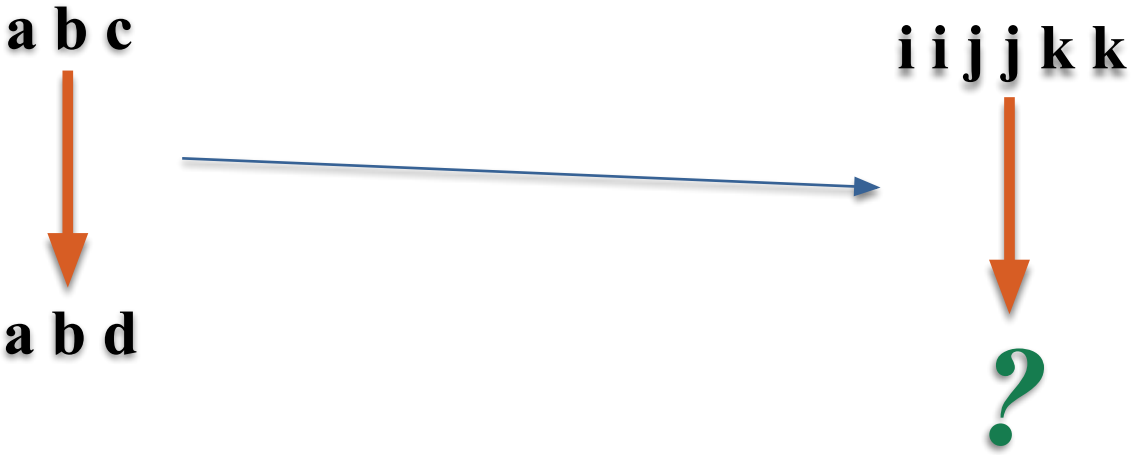
Copycat

Douglas Hofstadter
(1945 – …)

- Mitchell & Hofstadter – 1993

a b c	→	a b d
k j i	→	?

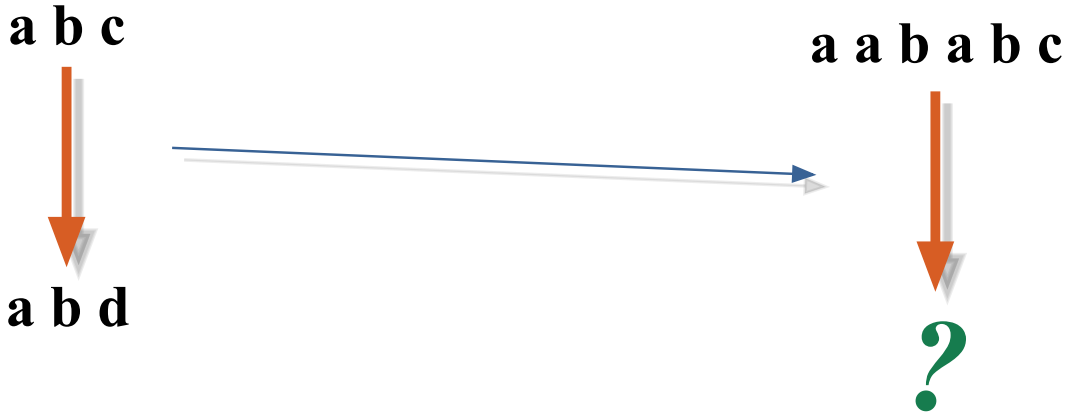




- a b d
- i i j j k d
- i i j j k l
- i i j j k k
- ?

Copycat

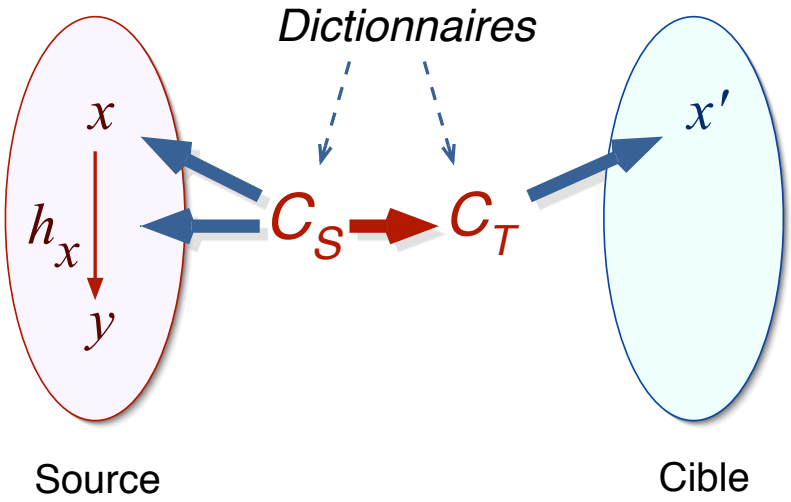
a b c	→	a b d
i j k	→	?
k j i	→	?
c	→	?
a b c d e	→	?
m	→	?
x y z	→	?
f p c	→	?
i i j j k k	→	?
a a b b c c	→	?
i j j k k k	→	?
a b b c c c	→	?

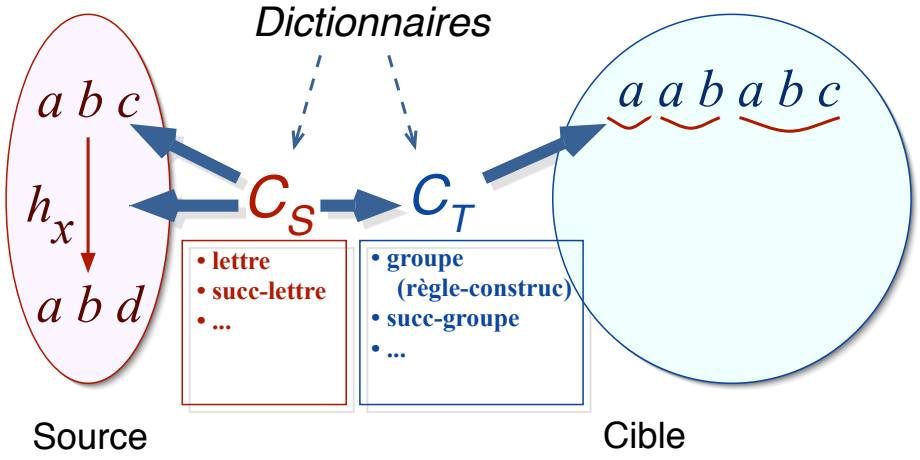

a b c
a b d
a a b a b c
?

Domain adaptation & analogie

- Learn both :
  - A good **representation**
    - Of the source domain
    - Of the target domain
  - A good **transformation** rule

??





Copycat

- Successor and predecessor
  - a → b, b → a, 1 → 2, …
- Sequence
  - abcd…
- Sequence of sequences
  - aaabbbccc…
- First, last, …
- Opposite(first, last),
  Opposite(successor, predecessor), …

Various solutions

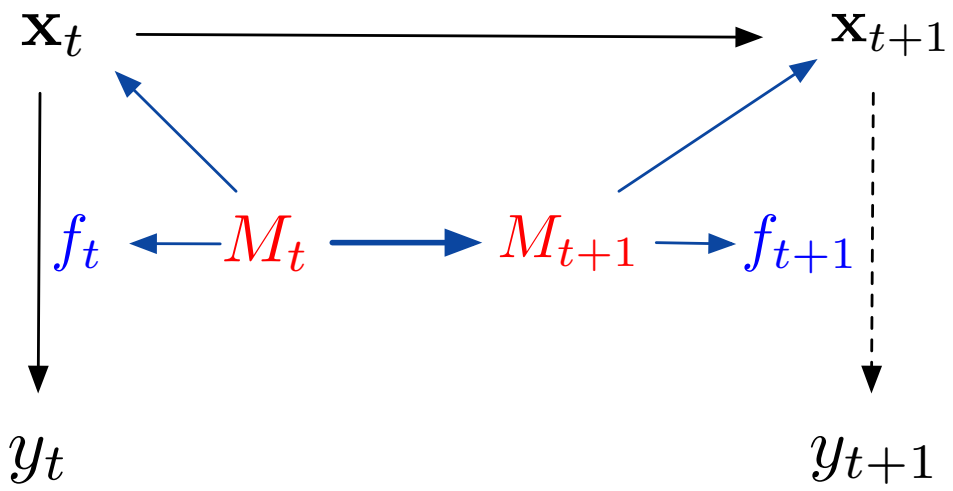
a b c	→	a b d	Comment
i j k	→	i j l	Replace *last letter* by its *successor*
	→	i j k	Replace **c** by **d**
	→	i j d	Replace *last letter* by **d**
	→	i j	Remove last letter and if this a '**c**' replace by **d**
	→	a b d	Replace by **a b d**
	→	i j k l	**c**=3, **d**=4, length(**ijk**)=3, length(**ijkl**)=4
	→	i j f	Replace last letter by **d** if this a '**c**' otherwise by **f**

Cornuéjols [1994 – 2020 - …]

- *Minimum Description Length Principle* + Copycat
  - MDLp = approximation of Kolmogorov's complexity for learning

- Analogy making:
  1. Minimize the description of the known terms A:B :: C:? (production)
     or A:B :: C:D (evaluation)
  2. Choose the smallest description

An approach to analogy: using Kolmogorov complexity

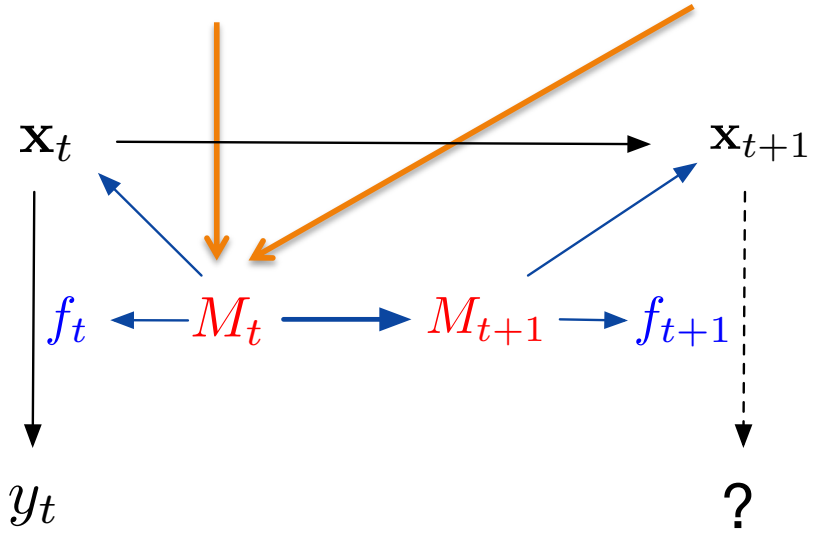


$$K(M_t) + K(\mathbf{x}_t|M_t) + K(f_t|M_t) + \underbrace{K(M_{t+1}|M_t)}_{\text{Change of system of reference}} + K(\mathbf{x}_{t+1}|M_{t+1}) + K(f_{t+1}|M_{t+1})$$

[Cornuéjols, 1996, 1997, 1998, 2016]

Une formalisation

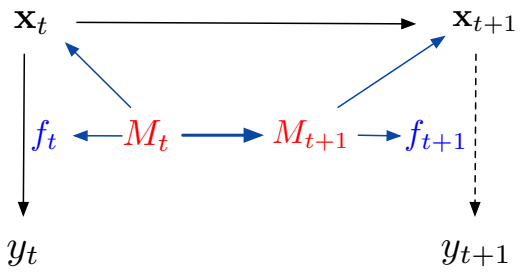
- Kolmogorov's Complexity
  - Uses a **dictionary** (with associated description lengths)
  - Which **depends** on the a **priori knowledge** and the **past experiences**



$$K(M_t) + K(\mathbf{x}_t|M_t) + K(y_t|M_t) + K(M_{t+1}|M_t) + K(\mathbf{x}_{t+1}|M_{t+1}) + K(f_{t+1}|M_{t+1})$$

[A. Cornuéjols (1996) « Analogie, principe d'économie et complexité algorithmique »]

An approach to analogy: using Kolmogorov complexity

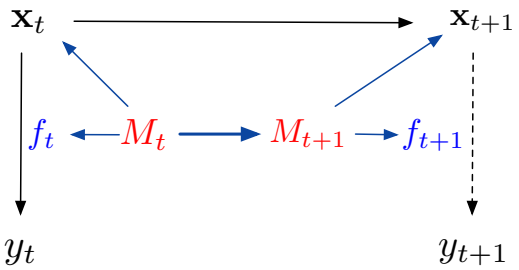


- Descripteurs utilisés dans la définition des structures :
  - orientation (-> / <-) — ***1** bit*
  - cardinalité ou nombre d'éléments : n — ***log$_2$(n) + 1** bits*
  - type d'éléments — (voir en-dessous)
  - longueur : l — ***log$_2$(l) + 1** bits*
  - commençant ou se terminant par l'élément = x — ***L(x)** bits*
- **Lettre** — *(1/2) -> **1** bit*
  Une lettre particulière (e.g. 'd') — *(1/2.26) -> **6** bits*
- **Chaîne** (orientation,éléments) — *(1/8) -> **3** bits*
  L = 3 + L(orientation) + $\sum$ L(éléments)
  e.g. L('a3bd' avec orientation = ->) = 3 + 1 + $\log_2((1/2.26)^3)$ + L(3)
  = 3 + 1 + 18 + 3 = 25 bits
- **Ensemble** (type d'éléments, cardinalité, éléments) — *(1/8) -> **3** bits*
  L = 3 + L(type) + L(cardinalité) + $\sum$ L(éléments)
- **Groupe** (type d'éléments, nombre d'éléments, éléments) — *(1/8) -> **3** bits*
  L = 3 + L(type) + L(nb él.) + $\sum$ L(éléments)
- **Séquence** (orientation, type d'éléments, loi de succession ou nombre d'éléments, longueur, commençant ou se terminant par) — *(1/8)*
  L = 3+ L(orient.) + L(type) + L(loi) or L(nb él.) + L(long) + L(début/fin)
- Description et longueur d'une loi de **succession**
  succ(type-of-el.,n,x) ≡ le nième successeur de l'élément x du type type-of-el.
  L = L(type) + L(n (voir ci-dessous)) + L(x)
  L(n) = L(1/6) si n=1 ou -1 (1er successeur ou prédécesseur)
  L(1/3) si n=0 (même élément)
  L((1/3).(1/2)^p) sinon (avec p=n si n≥0, p=-n sinon)
- Premier / Dernier (par rapport à l'orientation définie) — ***1** bit*
- nième — ***n** bits*

$$K(M_t) + K(\mathbf{x}_t|M_t) + K(f_t|M_t) + \underbrace{K(M_{t+1}|M_t)}_{\text{Change of system of reference}} + K(\mathbf{x}_{t+1}|M_{t+1}) + K(f_{t+1}|M_{t+1})$$

An approach to analogy: using Kolmogorov complexity

[Cornuéjols, 1996, 1997, 1998, 2016]



'abc' ≡ **Chaîne** (1/8)

orientation : -> (1/2)

1er='A', 2ème='B', 3ème='C' $(1/4.26)^3$

TOTAL (longueur) : **21** bits

'abc' ≡ **Ensemble** (1/8)

{'A', 'B', 'C'} $(1/4.26)^3$

TOTAL : **20** bits

'abc' ≡ **Séquence** (1/8)

orientation : -> (1/2)

type d'éléments = lettres (1/2)

loi de succession :

successeur(élt(lettre=x)) = élt(succ(lettre,1,x))

L(lettre) + L(1er succ) + L(x) = L(1/2 . 1/6 . 1)

= l(1/12) = 4 bits

longueur = 3 3 bits

commençant avec l'élément(lettre='A') (1/26)

TOTAL : **17** bits

An approach to analogy: using Kolmogorov complexity

[Cornuéjols, 1996, 1997, 1998, 2016]

Problème 1 : **abc => abd ; iijjkk => ?**

Solution 1 : "*Remplacer groupe de droite par son successeur*" **iijjkk => iijjll**

Solution 2 : "*Remplacer lettre de droite par son successeur*" **iijjkk => iijjkl**

Solution 3 : "*Remplacer lettre de droite par D*" **iijjkk => iijjkd**

Solution 4 : "*Remplacer 3ème lettre par son successeur*" **iijjkk => iikjkk**

Solution 5 : "*Remplacer les C par D*" **iijjkk => iijjkk**

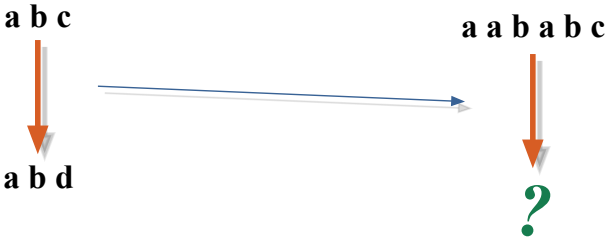
Solution 6 : "*Remplacer groupe de droite par la lettre D*" **iijjkk => iijjd**

	P1;S1	P1;S2	P1;S3	P1;S4	P1;S5	P1;S6
$L(M_S)$	10	9	11	11	12	11
$L(S_S \mid M_S)$	8	18	18	18	22	15
$L(\beta_S \mid M_S)$	4	4	3	7	8	3
$L(M_C \mid M_S)$	5	0	0	0	0	17
$L(S_C \mid M_C)$	8	36	36	36	42	15
$L(\beta_C \mid M_C)$	6	4	3	7	8	3
Total-1 (bits)	**41**	**71**	**71**	**79**	**93**	**65**
Total-2 (bits)	**35**	**67**	**68**	**72**	**85**	**62**
Rang	1	3	4	4	6	2

Copycat + MDLp

a b c	→	a b d	Length in bits
i i j j k k	→	i i j j l l	35
i i j j k k	→	i i j j k l	67
i i j j k k	→	i i j j k d	68
i i j j k k	→	i i k j k k	72
i i j j k k	→	i i j j k k	85
i i j j k k	→	i i j j d	62

Results



'abc' ≡ **Chaîne**		(1/8)
	orientation : ->	(1/2)
	1er='A', 2ème='B', 3ème='C'	$(1/4.26)^3$
	TOTAL (longueur) : **21** bits	

'abc' ≡ **Ensemble**		(1/8)
	{'A', 'B', 'C'}	$(1/4.26)^3$
	TOTAL : **20** bits	

'abc' ≡ **Séquence**		(1/8)
	orientation : ->	(1/2)
	type d'éléments = lettres	(1/2)
	loi de succession :	
	successeur(élt(lettre=x)) = élt(succ(lettre,1,x))	
	L(lettre) + L(1er succ) + L(x) = L(1/2 . 1/6 . 1)	
	= l(1/12) = 4 bits	
	longueur = 3	3 bits
	commençant avec l'élément(lettre='A')	(1/26)
	TOTAL : **17** bits	

[A. Cornuéjols (1996) « Analogie, principe d'économie et complexité algorithmique »]

Analogy making and MDLP

- Application to **language analogies**: how to end words (conjugations, plurals, …)

apte : inapte :: élu : x	$x =$ inélu	*(Prefixation)*
let,?0,:,'i','n',?0,let,mem,0,'apte',::,mem,0,'élu'		
átír : átírunk :: kitart : x	$x =$ kitartunk	*(Suffixation)*
let,?0,:,?0,'u','n','k',let,mem,0,'átír',::,mem,0,'kitart'		
pati : patti :: olo : x	$x =$ olto	*(Insertion)*
let,?0,?1,:,?0,'t',?1,let,mem,0,'pa','ti',::,mem,0,'ol','o'		
pria : pria-pria :: keju : x	$x =$ keju-keju	*(Repetition)*
let,?0,:,?0,'-',?0,let,mem,0,'pria',::,mem,0,'keju'		
vantut : vanttu :: autopilotit : x	$x =$ autopilotti	*(Reduplication)*
let,?0,?1,'t',:,?0,'t',?1,let,mem,0,'van','tu',::,mem,0,'autopilot','i'		

Murena, P. A., Al-Ghossein, M., Dessalles, J. L., & Cornuéjols, A. (2020). **Solving Analogies on Words based on Minimal Complexity Transformation**. In *IJCAI* (pp. 1848-1854).

Analogy making and MDLP

- Application to **language analogies**: how to end words (conjugations, plurals, …)

Language	#analogies	NLG_COMP	NLG_PROP	NLG_ALEA
Arabic	165,113	87.18%	**93.33%**	81.91%
Finnish	313,011	**93.69%**	92.76%	78.75%
Georgian	3,066,273	**99.35%**	97.54%	88.42%
German	730,427	**98.84%**	96.21%	95.42%
Hungarian	2,912,310	**95.71%**	92.61%	86.02%
Maltese	28,365	**96.38%**	84.72%	91.84%
Navajo	321,473	81.21%	**86.87%**	78.95%
Russian	552,423	96.41%	**97.26%**	95.46%
Spanish	845,996	**96.73%**	96.13%	94.42%
Turkish	245,721	**89.45%**	69.97%	70.06%
Total	9,181,112	**96.41%**	94.34%	87.93%

Table 2: Proportion of correct answers when solving analogies from the dataset SIGMORPHON'16 using our method NLG_COMP and two state-of-the-art methods NLG_PROP [Fam and Lepage, 2018] and NLG_ALEA [Langlais *et al.*, 2009].

The **results** using **deep NNs** and learnt embeddings were in the range 0.1% to 17%!!

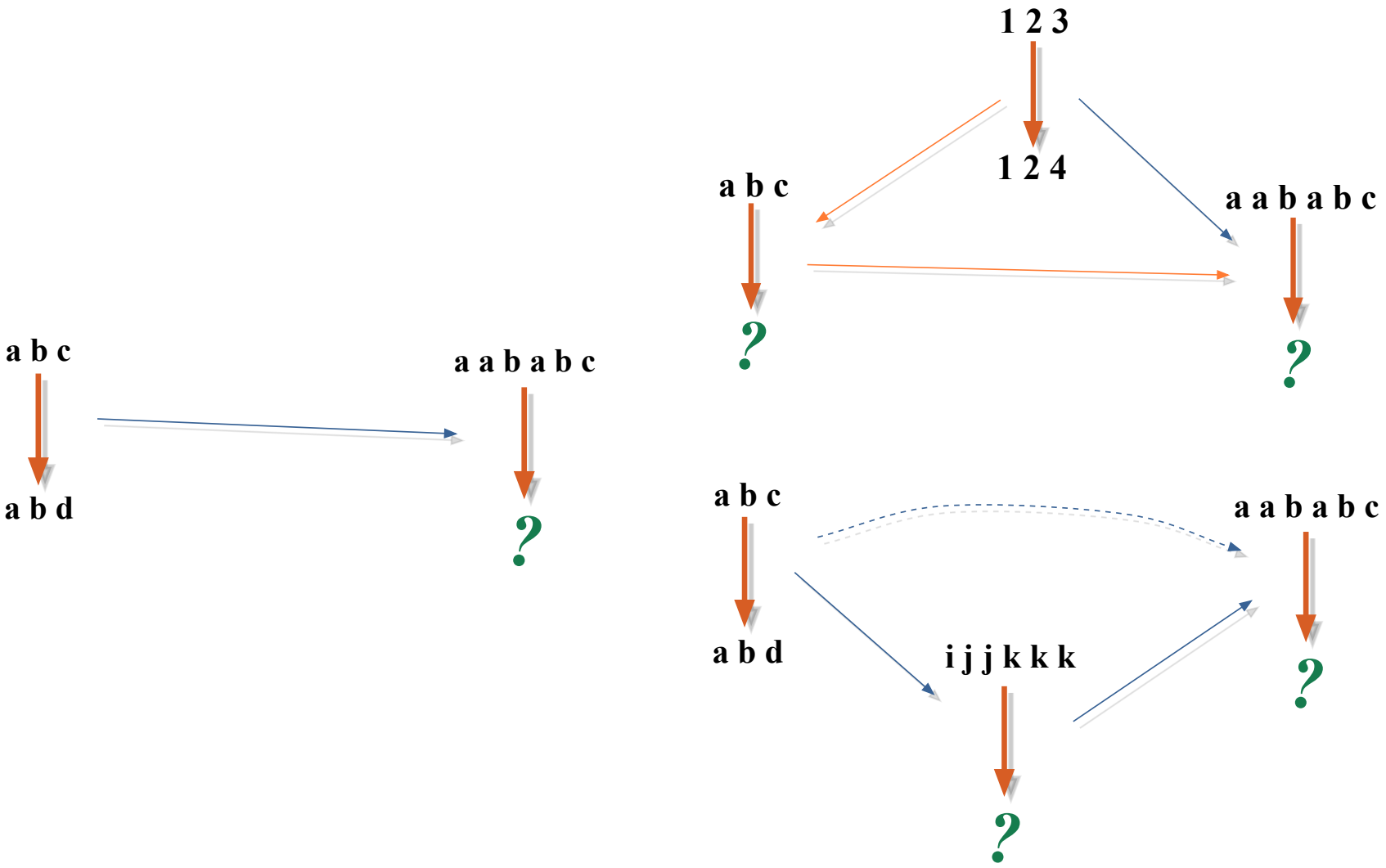
Murena, P. A., Al-Ghossein, M., Dessalles, J. L., & Cornuéjols, A. (2020). **Solving Analogies on Words based on Minimal Complexity Transformation**. In *IJCAI-2020* (pp. 1848-1854).

Outline

1. Transfer learning: questions
2. TransBoost: an algorithm and what it tells on the role of the source
3. The MDLP and analogy making
4. Conclusions

Some speculations

Transfer and sequence effects



...

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).