On-line learning, Learning Using Privileged Information (LUPI) and transfer learning

Antoine Cornuéjols

AgroParisTech – INRA MIA 518

Équipe LINK

AgroParisTech

INRA SCIENCE & IMPACT

Outline

1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

The online learning scenario

- A stream: $\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \boldsymbol{x}_3 \quad \cdots \quad \boldsymbol{x}_t$

$$h_1 \quad h_2 \quad h_3 \quad \cdots \quad h_t$$

$$\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3 \quad \cdots \quad \hat{y}_t$$

Loss($\hat{y}_t$, y_t)

$$y_1 \quad y_2 \quad y_3 \quad \cdots \quad y_t$$

E.g. ***Choice of melons***. I see one, I make **a prediction** about its tastiness, then I eat it and know the answer.

Novelty wrt. Statistical learning

1. Learning and testing are intermingled
   - **No distinction** between training set, validation set and test set

Novelty wrt. Statistical learning

1. Learning and testing are intermingled
   - **No distinction** between training set, validation set and test set

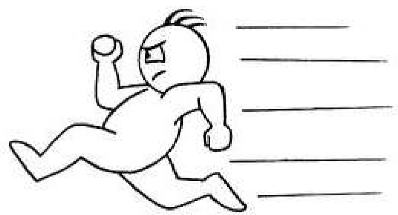
2. The environment may change over time
   - The learner should adapt

Novelty wrt. Statistical learning

1. Learning and testing are intermingled
   - **No distinction** between training set, validation set and test set
2. The environment may change over time
   - The learner should adapt
3. Dilemma
   - **Keep as much as possible memory** from the past to gain in precision
   - But **be ready to adapt** to changes (and reduce the size of the memory)

Desirable properties of a system that handle concept drift

- **Adapt** to concept drift **as soon as possible**



- Distinguish **noise** from **true changes**
  - Robust to noise but adaptive to changes

- Recognize and react to **recurring contexts**

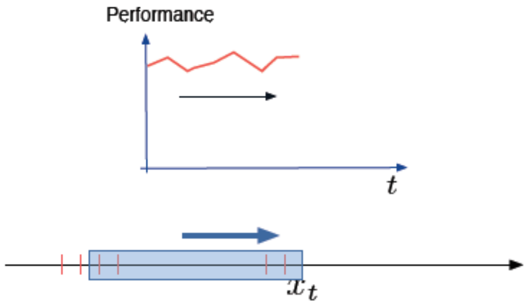
- Adapt with **limited resources** (time and memory)

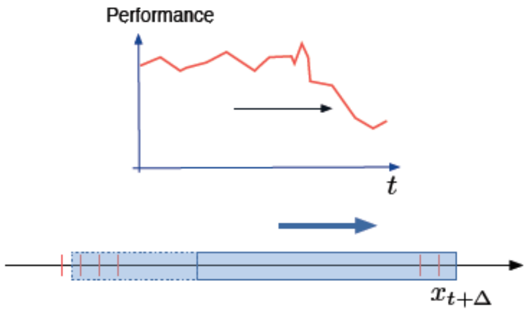
Two main approaches

1. Directly **control the memory**

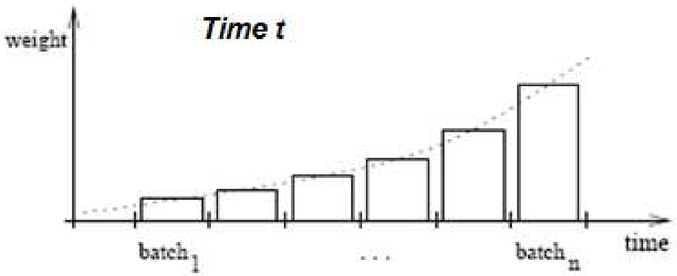
   – Adapt the **window size**

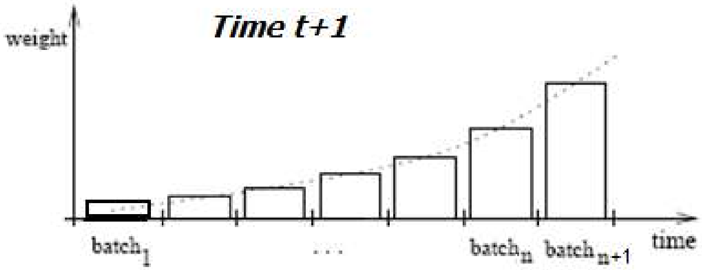
     [Widmer G. & Kubat M. (1996). *Learning in the presence of concept drift and hidden contexts*. Mach. Learning, 23, 69-101]





   – **Weight** the past **examples** $w(\mathbf{x}) \; = \; e^{-\lambda \, t_{\mathbf{x}}}$



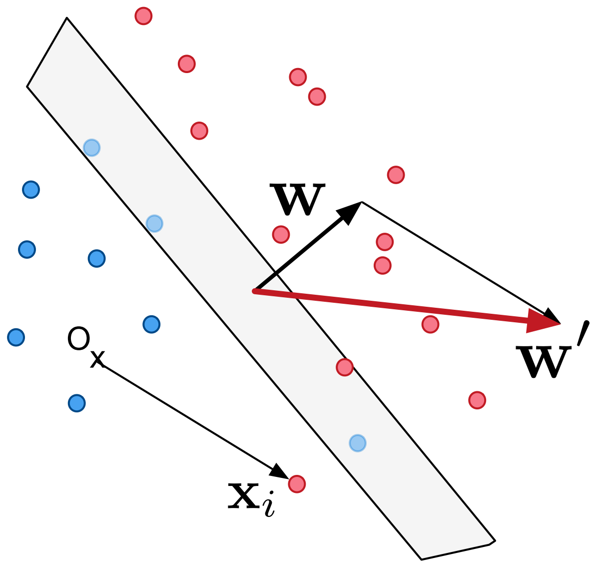


2. **Adapt the hypothesis** at each time step

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta \, y_t \, \mathbf{x}_t$$

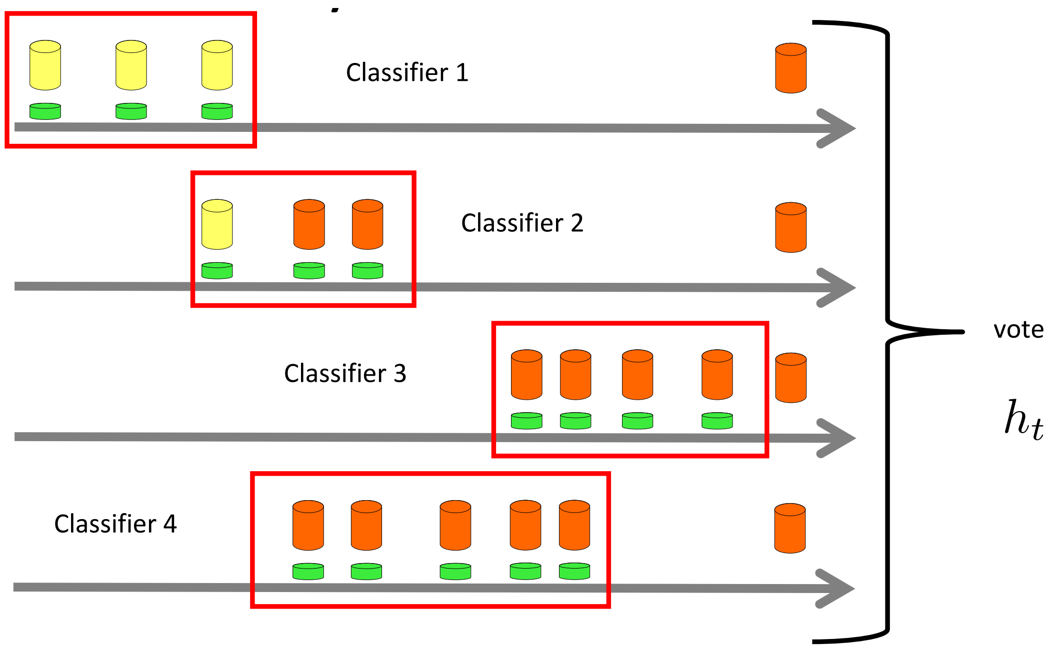
Past

Controls adaptivity



Two main approaches

2. **Adapt the hypothesis** at each time step



$$h_t(\cdot) \;=\; \sum_i \alpha_i(t)\ \text{classifier}_t^i(\cdot)$$

Online learning: general perspectives

1. **Non Lipschitzian** scenario
   - Successive entries are independent, possibly adversarial
   - Online learning theory [Cesa-Bianchi & Lugosi, 2006]

Online learning: general perspectives

1. **Non Lipschitzian** scenario
   - Successive entries are independent, possibly adversarial
   - Online learning theory [Cesa-Bianchi & Lugosi, 2006]
2. Temporal **consistency**
   - Heuristic online learning methods (sliding windows, adaptation, …)
   - Tracking: Adapt to the past and **always be behind the changes**

Online learning: general perspectives

1. **Non Lipschitzian** scenario
   – Successive entries are independent, possibly adversarial
   – Online learning theory [Cesa-Bianchi & Lugosi, 2006]
2. Temporal **consistency**
   – Heuristic online learning methods (sliding windows, adaptation, …)
   – Tracking: Adapt to the past and **always be behind the changes**
3. **Extrapolate** the likely **changes of h_t**
   – [Ghazal Jaber, 2013]
   – Needs extrapolation from past observed behavior

Online learning: general perspectives

1. **Non Lipschitzian** scenario
   - Successive entries are independent, possibly adversarial
   - Online learning theory [Cesa-Bianchi & Lugosi, 2006]
2. Temporal **consistency**
   - Heuristic online learning methods (sliding windows, adaptation, …)
   - Tracking: Adapt to the past and **always be behind the changes**
3. **Extrapolate** the likely **changes of** h_t
   - [Ghazal Jaber, 2013]
   - Needs extrapolation from past observed behavior
4. **Transduction**: take into account the future “questions” x_{t+1}
   - As in semi-supervised learning

Online learning: general perspectives

1. **Non Lipschitzian** scenario
   - Successive entries are independent, possibly adversarial
   - Online learning theory [Cesa-Bianchi & Lugosi, 2006]
2. Temporal **consistency**
   - Heuristic online learning methods (sliding windows, adaptation, …)
   - Tracking: Adapt to the past and **always be behind the changes**
3. **Extrapolate** the likely **changes of h_t**
   - [Ghazal Jaber, 2013]
   - Needs extrapolation from past observed behavior
4. **Transduction**: take into account the future "question(s)" x_{t+1}
   - Learn h_t using x_{t+1} as well. (As in semi-supervised learning)
5. **Both** (3) and (4)

Online learning and transfer learning

- Each step implies a "small" transfer
  - From the **environment at time *t*-1** to the **environment at time *t***

- Use "source knowledge" (h_{t-1})

  and the current batch $\left\{ (\mathbf{x}_t^i, y_t^i) \right\}_{1 \leq i \leq m}$

  to learn target h_t by adapting from past to current environment

Outline

1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

Learning Using Privileged Information

Inspired by learning at school

- The goal is to learn a function $$h : \mathbf{x} \in \mathcal{X} \rightarrow y \in \{-1, +1\}$$
- Suppose that at learning time there is more available information than at test time $$\mathcal{S}^* = \{(\mathbf{x}_i, \mathbf{x}_i^*, y_i)\}_{1 \leq i \leq m}$$
- Can we then improve the generalization performance wrt. the one obtained with *S* only?

V. Vapnik and A. Vashist (2009) "A new learning paradigm: Learning using privileged information". Neural Networks, vol. 22, no. 5, pp. 544–557, 2009

One solution: SVM+

- The classical optimization problem

$$\begin{cases} \min \frac{1}{2}\langle \omega, \omega \rangle + C \sum_{i=1}^{m} \xi_i \\ \text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - \xi_i, \qquad i = 1, \ldots, m. \end{cases}$$

- is changed into

$$\begin{cases} \min \frac{1}{2}[\langle \omega, \omega \rangle + \gamma \langle \omega^*, \omega^* \rangle] + C \sum_{i=1}^{m} [\langle \omega^*, x^* \rangle + b^*] \\ \text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - [\langle \omega^*, x_i^* \rangle + b^*], \qquad i = 1, \ldots, m, \\ \quad [\langle \omega^*, x_i^* \rangle + b^*] \geq 0, \qquad i = 1, \ldots, m, \end{cases}$$

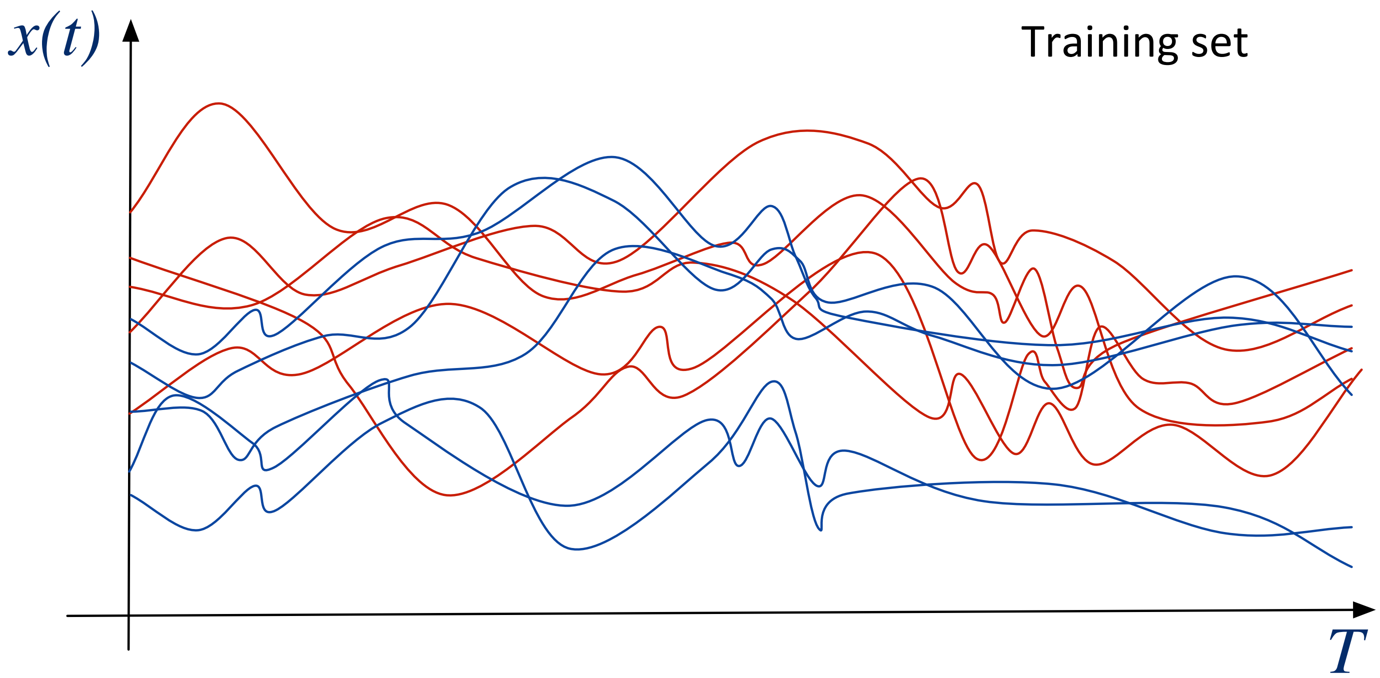
C and γ are hyperparameters

- Intuition:
  - Identify the **difficult examples**
  - And relax / tighten the SVM constraints accordingly -> better generalization performances

Outline

1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

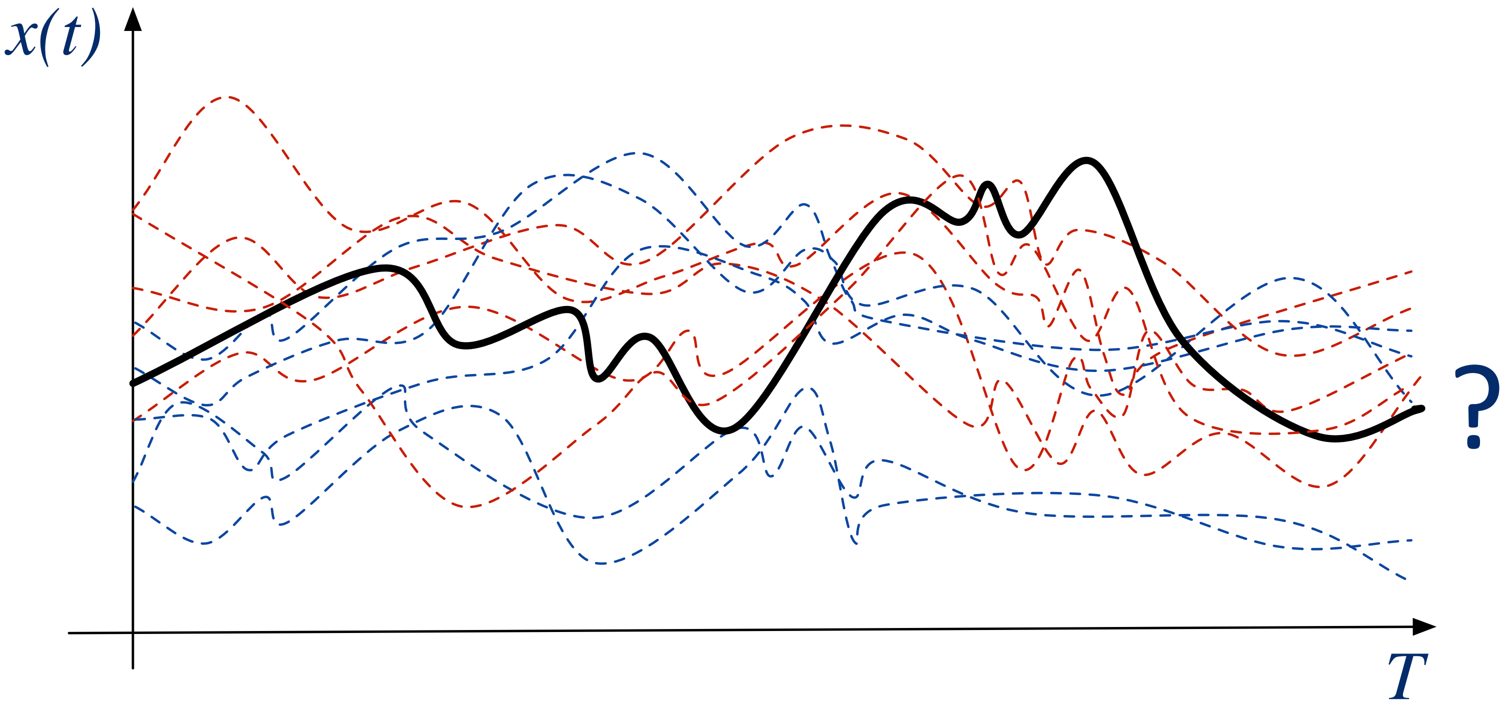
Classification of time series



- 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

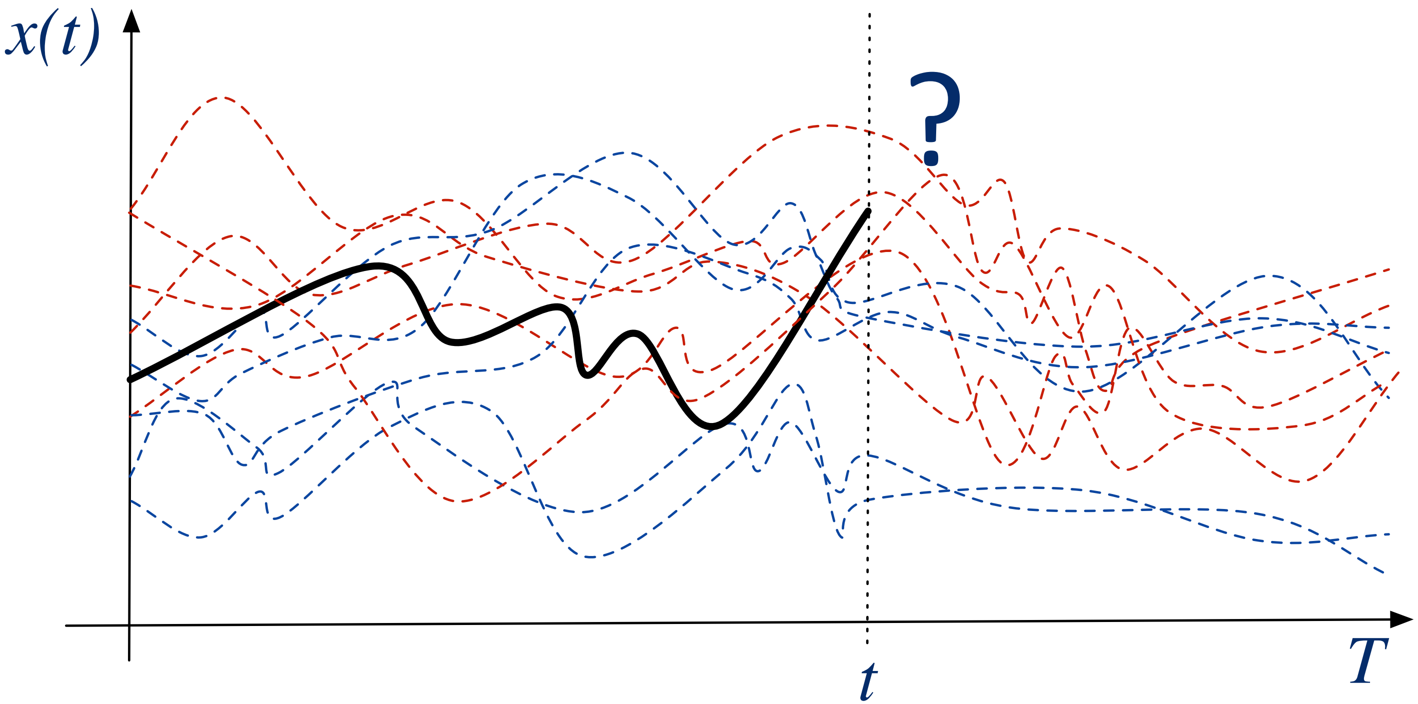
Standard classification of time series

- 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 $\boldsymbol{x_t}$?

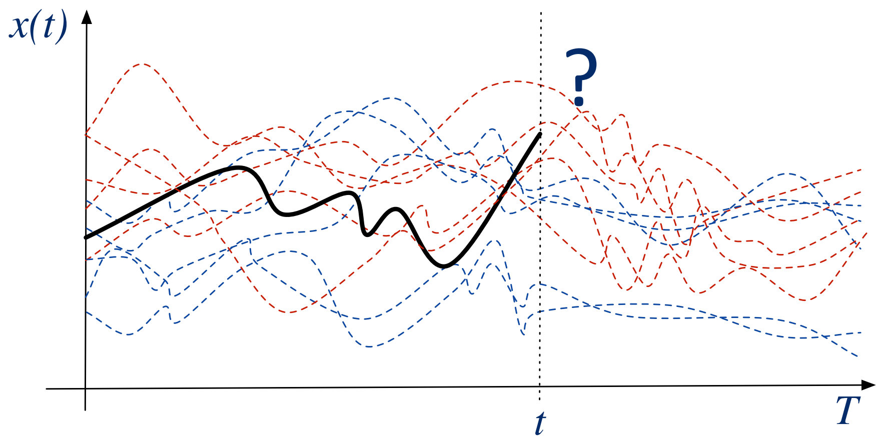


New set of decision problems : early classification

- Data **stream**
- **Classification** task
- As **early** as possible
- A **trade-off**
  - Classification **performance** (better if $t \nearrow$)
  - Cost of **delaying** prediction (better if $t \searrow$)

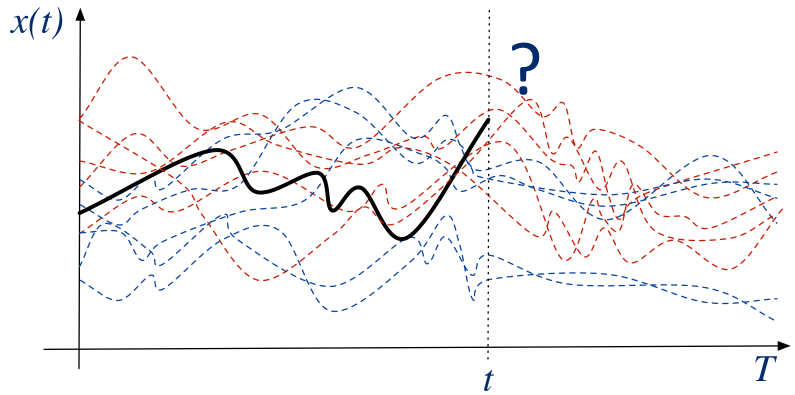
Online decision problem

- With option to defer at each time step
  - If the **expected future performance** overcomes the **cost of delaying decision**



Early classification and LUPI

- This is a LUPI setting



- How to take advantage of this?

Decision making (1)

- Given an incoming sequence $\mathbf{x}_t = \langle x_1, x_2, \ldots, x_t \rangle$ where $x_t \in \mathbb{R}$
- And given:
  - A *miss-classification cost function* $C_t(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$
  - A *delaying decision cost function* $C(t) : \mathbb{N} \longrightarrow \mathbb{R}$

- **What is the optimal time to make a decision?**

Expected cost for a decision at time *t*

$$f(\mathbf{x}_t) = \underbrace{\sum_{y \in \mathcal{Y}} P(y|\mathbf{x}_t) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x}_t)\, C_t(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t)$$

Decision making (1)

- Given an incoming sequence $\mathbf{x}_t = \langle x_1, x_2, \ldots, x_t \rangle$ where $x_t \in \mathbb{R}$
- And given:
  - A *miss-classification cost function* $C_t(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \longrightarrow \mathbb{R}$
  - A *delaying decision cost function* $C(t) : \mathbb{N} \longrightarrow \mathbb{R}$
- **What is the optimal time to make a decision?**

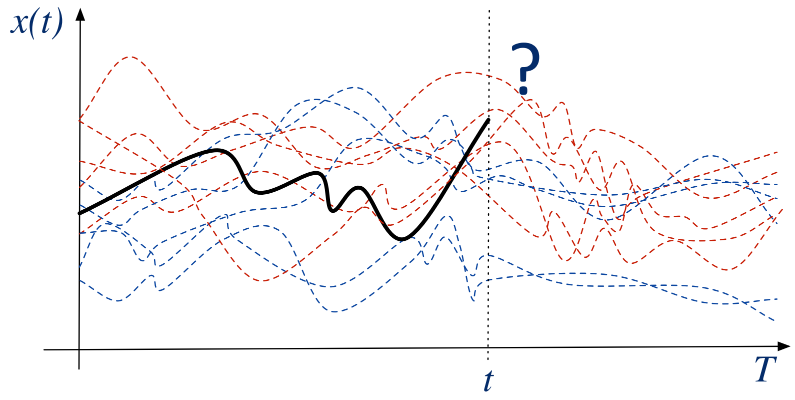
Expected cost for a decision at time *t*

$$f(\mathbf{x}_t) = \underbrace{\sum_{y \in \mathcal{Y}} P(y|\mathbf{x}_t) \sum_{\hat{y} \in \mathcal{Y}} P(\hat{y}|y, \mathbf{x}_t)\, C_t(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t)$$

Optimal time: $t^* = \underset{t \in \{1,\ldots,T\}}{\mathrm{ArgMin}}\ f(\mathbf{x}_t)$

Early classification and LUPI

- This is a LUPI setting



- How to take advantage of this?

  1. Knowledge of **possible future sequences**

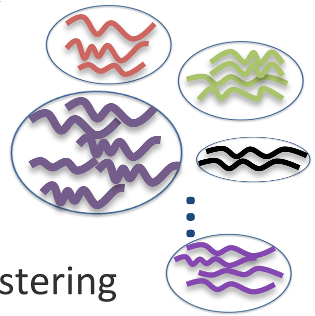
  2. Possibility to learn **classifiers for all time steps**

The principle

1. During **training**:

– identify **meaningful subsets** of time sequences in the training set: $\boldsymbol{c_k}$

$$P(y|\mathbf{x}_t) \;\rightarrow\; P(y|\mathfrak{c}_k)$$

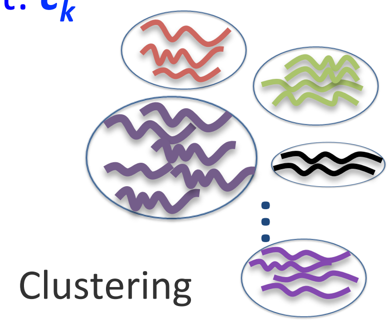


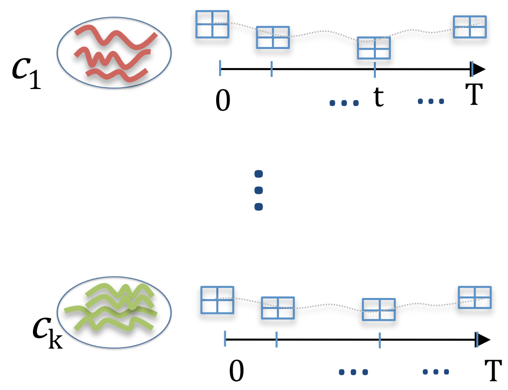
Clustering

The principle

1. During **training**:

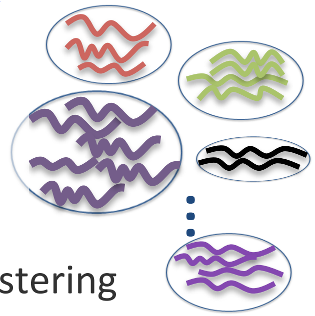
- identify **meaningful subsets** of time sequences in the training set: $\boldsymbol{c_k}$
- **For each of these subsets** c_k, and for each time step t
  - Estimate the **confusion matrices**
    - T classifiers are learnt $h_t(\mathbf{x}_t) : \mathcal{X}_t \rightarrow \mathcal{Y}$
    - And their confusion matrices $P_t(\hat{y}|y, \mathfrak{c}_k)$ are estimated on a test set





The principle

1. During **training**:
   - identify **meaningful subsets** of time sequences in the training set: $\boldsymbol{c_k}$
   - For each of these subsets c_k, and for each time step t
     - Estimate the **confusion matrices** $P_t(\hat{y}|y, \mathfrak{c}_k)$

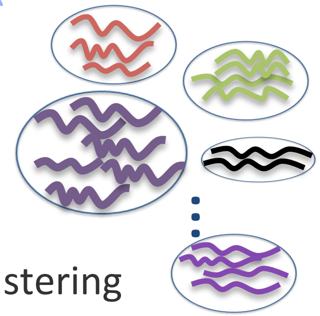


Clustering

2. **Testing**: For any new incomplete incoming sequence x_t
   - Identify the **most likely subset**: the closer class of shapes to x_t

The principle

1. During **training**:
   - identify **meaningful subsets** of time sequences in the training set: c_k
   - For each of these subsets c_k, and for each time step t
     - Estimate the **confusion matrices** $P_t(\hat{y}|y, \mathfrak{c}_k)$



Clustering

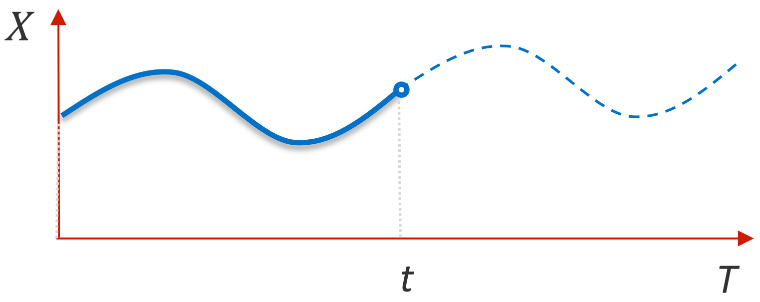
2. **Testing**: For any new incomplete incoming sequence x_t
   - Identify the most likely subset: the closer shape to x_t
   - Compute the **expected cost of decision** for all future time steps

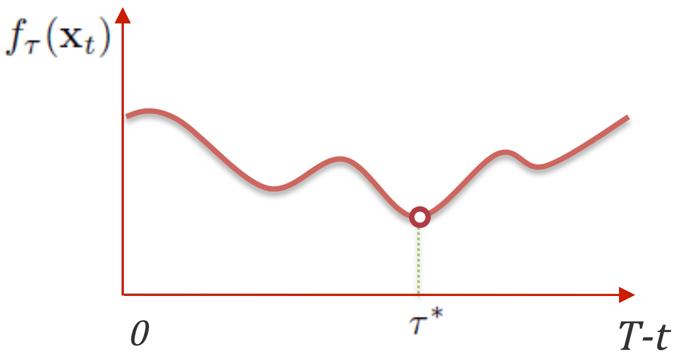
$$f_\tau(\mathbf{x}_t) = \underbrace{\sum_{\mathfrak{c}_k \in \mathcal{C}} P(\mathfrak{c}_k|\mathbf{x}_t) \sum_{y \in \mathcal{Y}} P(y|\mathfrak{c}_k) \sum_{\hat{y} \in \mathcal{Y}} P_{t+\tau}(\hat{y}|y, \mathfrak{c}_k)\, C(\hat{y}|y)}_{\text{expected miss-classification cost given } \mathbf{x}_t} + C(t+\tau)$$

A non myopic decision process

- Optimal estimated time relative to current time *t*

$$\tau^* = \underset{\tau \in \{0,\dots,T-t\}}{\mathrm{ArgMin}} \; f_\tau(\mathbf{x}_t)$$

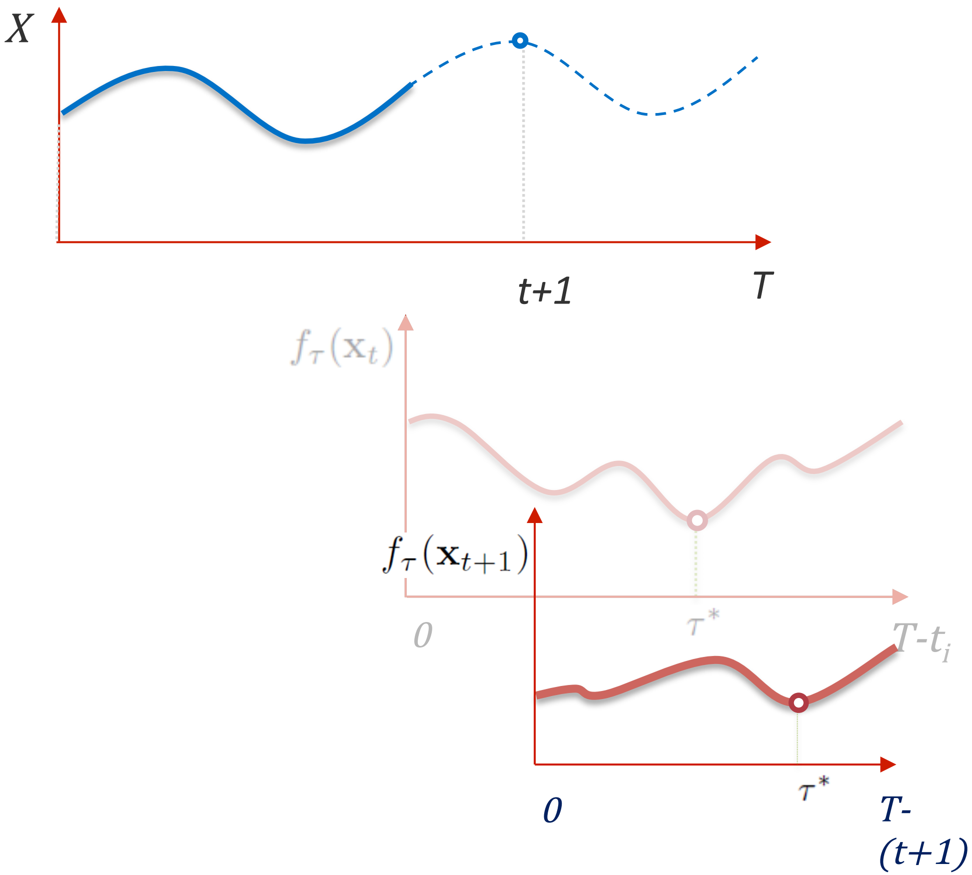




→ Continue monitoring

A non myopic decision process

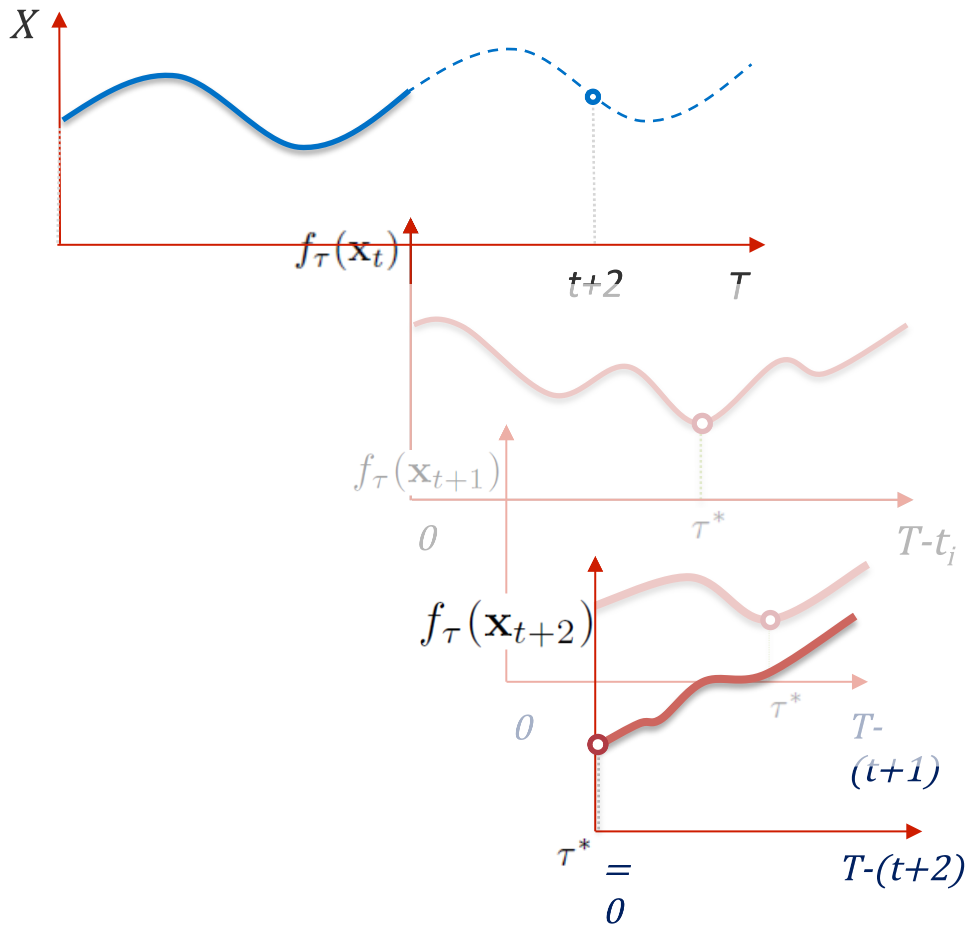
- Optimal estimated time relative to current time t $\quad \tau^* = \underset{\tau \in \{0,\ldots,T-t\}}{\text{ArgMin}} \; f_\tau(\mathbf{x}_t)$



Continue monitoring

A non myopic decision process

- Optimal estimated time relative to current time *t* $\tau^* = \underset{\tau \in \{0,\ldots,T-t\}}{\text{ArgMin}} \; f_\tau(\mathbf{x}_t)$

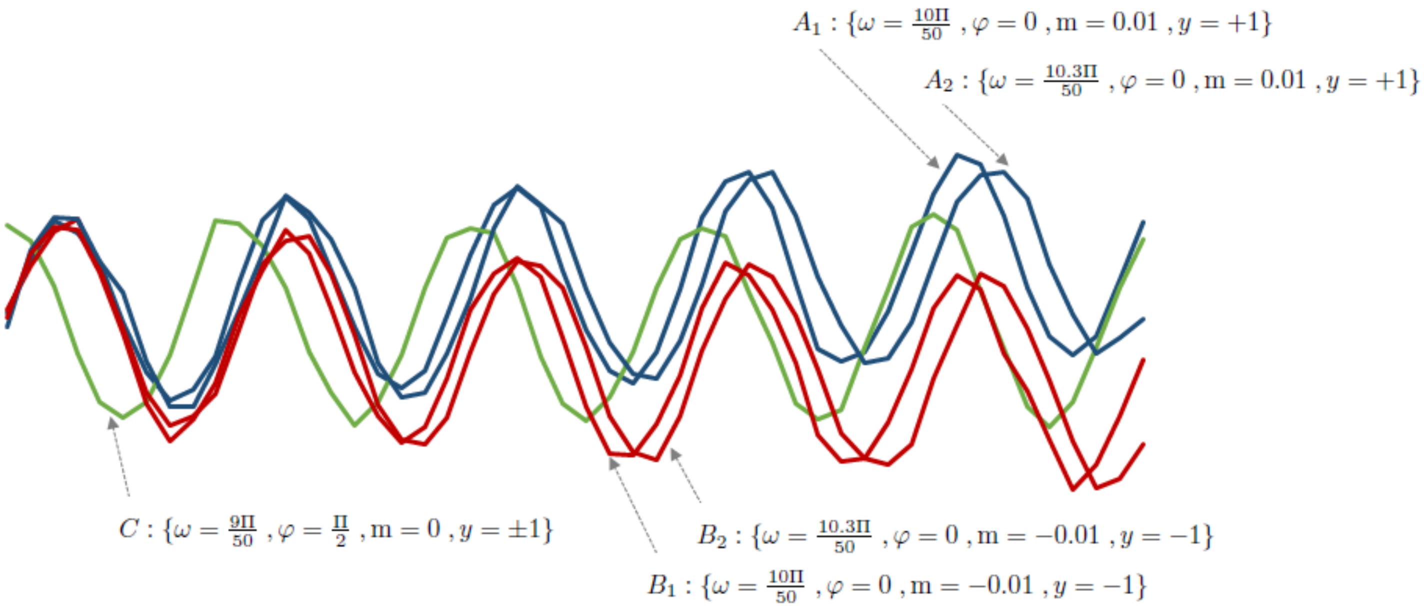


Take decision

Controlled data

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

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



Results: effect of the noise level

Increasing the **noise level** **increases** the waiting time, and then it's no longer worth it

$C(t)$	$\pm b$	0.02			0.05			0.07		
	$\varepsilon(t)$	$\tau^\star$	$\sigma(\tau^\star)$	AUC	$\tau^\star$	$\sigma(\tau^\star)$	AUC	$\tau^\star$	$\sigma(\tau^\star)$	AUC
	0.2	**9.0**	2.40	0.99	**9.0**	2.40	0.99	**10.0**	0.0	1.00
	0.5	**13.0**	4.40	0.98	**13.0**	4.40	0.98	**15.0**	0.18	1.00
0.01	*1.5*	**24.0**	10.02	0.98	**32.0**	2.56	1.00	**30.0**	12.79	0.99
	5.0	**26.0**	7.78	0.84	**30.0**	18.91	0.87	**30.0**	19.14	0.88
	10.0	**38.0**	18.89	0.70	**48.0**	1.79	0.74	**46.0**	5.27	0.75
	15.0	**23.0**	15.88	0.61	**32.0**	13.88	0.64	**29.0**	17.80	0.62
	20.0	**7.0**	8.99	0.52	**11.0**	11.38	0.55	**4.0**	1.22	0.52
	0.2	8.0	2.00	0.98	8.0	2.00	0.98	9.0	0.0	1.00
	0.5	10.0	2.80	0.96	8.0	4.0	0.98	14.0	0.41	0.99
0.05	1.5	5.0	0.40	0.68	20.0	0.42	0.95	14.0	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	5.0	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	4.0	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	4.0	0.0	0.52
	0.2	6.0	0.80	0.95	7.0	1.60	0.94	8.0	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	10.0	0.0	0.95
0.10	1.5	4.0	0.0	0.67	5.0	0.43	0.68	6.0	0.80	0.74
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	4.0	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	4.0	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	4.0	0.0	0.52

Table 1. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

Results: effect of the waiting cost

Increasing the **waiting cost** **reduces** the waiting time

$C(t)$	$\pm b$	0.02			0.05			0.07		
	$\varepsilon(t)$	$\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	$\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	$\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC
	0.2	9.0	2.40	0.99	9.0	2.40	0.99	**10.0**	0.0	1.00
	0.5	13.0	4.40	0.98	13.0	4.40	0.98	**15.0**	0.18	1.00
0.01	1.5	24.0	10.02	0.98	32.0	2.56	1.00	**30.0**	12.79	0.99
	5.0	26.0	7.78	0.84	30.0	18.91	0.87	**30.0**	19.14	0.88
	10.0	38.0	18.89	0.70	48.0	1.79	0.74	**46.0**	5.27	0.75
	15.0	23.0	15.88	0.61	32.0	13.88	0.64	**29.0**	17.80	0.62
	20.0	7.0	8.99	0.52	11.0	11.38	0.55	**4.0**	1.22	0.52
	0.2	8.0	2.00	0.98	8.0	2.00	0.98	**9.0**	0.0	1.00
	0.5	10.0	2.80	0.96	8.0	4.0	0.98	**14.0**	0.41	0.99
0.05	1.5	5.0	0.40	0.68	20.0	0.42	0.95	**14.0**	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	**5.0**	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	**4.0**	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	**4.0**	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	**4.0**	0.0	0.52
	0.2	6.0	0.80	0.95	7.0	1.60	0.94	**8.0**	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	**10.0**	0.0	0.95
0.10	1.5	4.0	0.0	0.67	5.0	0.43	0.68	**6.0**	0.80	0.74
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	**4.0**	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	**4.0**	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	**4.0**	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	**4.0**	0.0	0.52

Table 2. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

Results: effect of the difference between classes

Increase of the difference between classes

The **performance** increases (AUC)

The *waiting time* is not much changed in these experiments

$C(t)$	$\pm b$ / $\varepsilon(t)$	**0.02** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	**0.05** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC	**0.07** $\overline{\tau}^\star$	$\sigma(\tau^\star)$	AUC
0.01	0.2	*9.0*	2.40	**0.99**	*9.0*	2.40	**0.99**	*10.0*	0.0	**1.00**
	0.5	13.0	4.40	0.98	13.0	4.40	0.98	15.0	0.18	1.00
	1.5	24.0	10.02	0.98	32.0	2.56	1.00	30.0	12.79	0.99
	5.0	26.0	7.78	0.84	30.0	18.91	0.87	30.0	19.14	0.88
	10.0	38.0	18.89	0.70	48.0	1.79	0.74	46.0	5.27	0.75
	15.0	23.0	15.88	0.61	32.0	13.88	0.64	29.0	17.80	0.62
	20.0	7.0	8.99	0.52	11.0	11.38	0.55	4.0	1.22	0.52
0.05	0.2	8.0	2.00	0.98	8.0	2.00	0.98	9.0	0.0	1.00
	0.5	*10.0*	2.80	**0.96**	*8.0*	4.0	**0.98**	*14.0*	0.41	**0.99**
	1.5	5.0	0.40	0.68	20.0	0.42	0.95	14.0	4.80	0.88
	5.0	8.0	3.87	0.68	6.0	1.36	0.64	5.0	0.50	0.65
	10.0	4.0	0.29	0.56	4.0	0.25	0.56	4.0	0.34	0.57
	15.0	4.0	0.0	0.54	4.0	0.25	0.56	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	4.0	0.0	0.52	4.0	0.0	0.52
0.10	0.2	6.0	0.80	0.95	7.0	1.60	0.94	8.0	0.40	0.96
	0.5	6.0	0.80	0.84	9.0	2.40	0.93	10.0	0.0	0.95
	1.5	*4.0*	0.0	**0.67**	*5.0*	0.43	**0.68**	*6.0*	0.80	**0.74**
	5.0	4.0	0.07	0.64	4.0	0.05	0.64	4.0	0.11	0.64
	10.0	4.0	0.0	0.56	48.0	1.79	0.74	4.0	0.22	0.56
	15.0	4.0	0.0	0.55	4.0	0.0	0.55	4.0	0.0	0.55
	20.0	4.0	0.0	0.52	11.0	11.38	0.55	4.0	0.0	0.52

Table 3. Experimental results in function of the waiting cost $C(t) = \{0.01, 0.05, 0.1\} \times t$, the noise level $\varepsilon(t)$ and the trend parameter b.

1. **Formalized** the problem as a sequential decision making problem
   - Explicit **trade-off**
     - Classification **performance**
     - **Cost of delaying** the decision

2. Proposed **an algorithm** which is
   - **Adaptive**
     - Takes into account the peculiarities of x_t
   - **Non myopic**
     - At each time step, estimates the expected future time for optimal decision

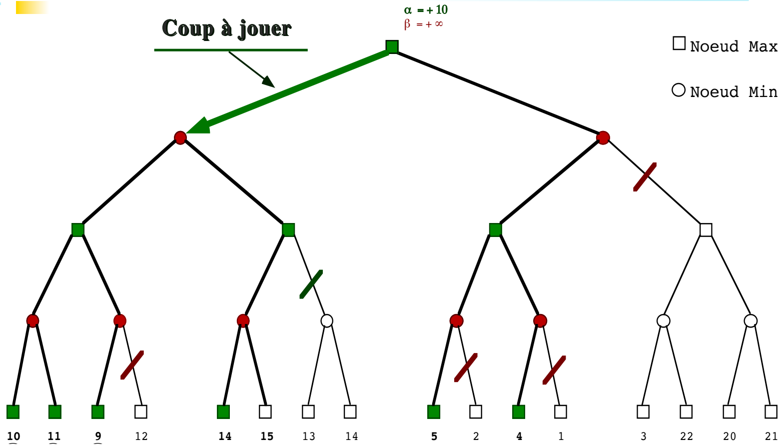
3. Showed **promising experimental results**
   - The delay before decision **exhibited** what should be **expected**

Outline

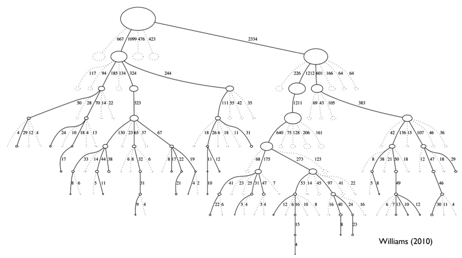
1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

Algorithms for games

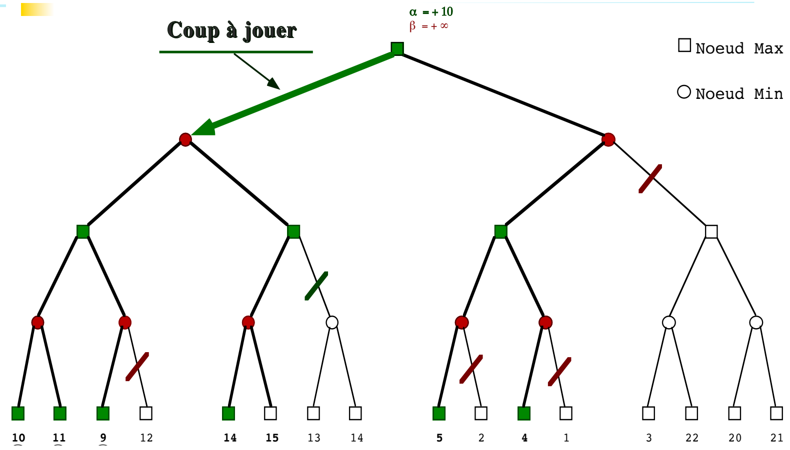
Taking decision when the current information is **incomplete**



MCTS



Algorithms for games

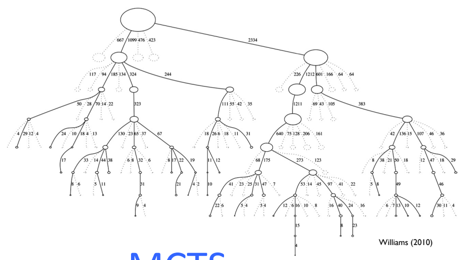


Taking decision when the current information is **incomplete**

- 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

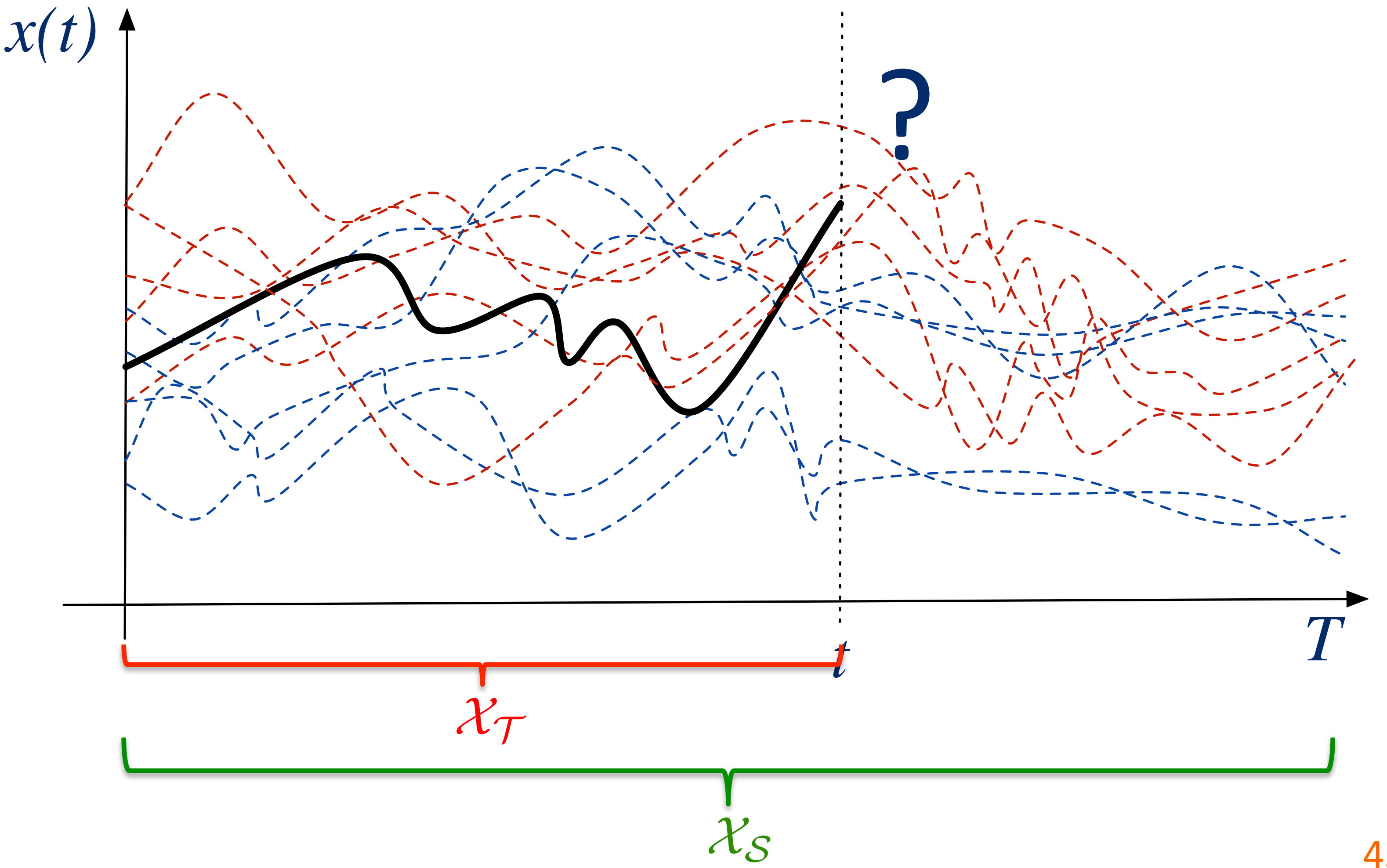


MCTS

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

Early **classification** of time series

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



Principle

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

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

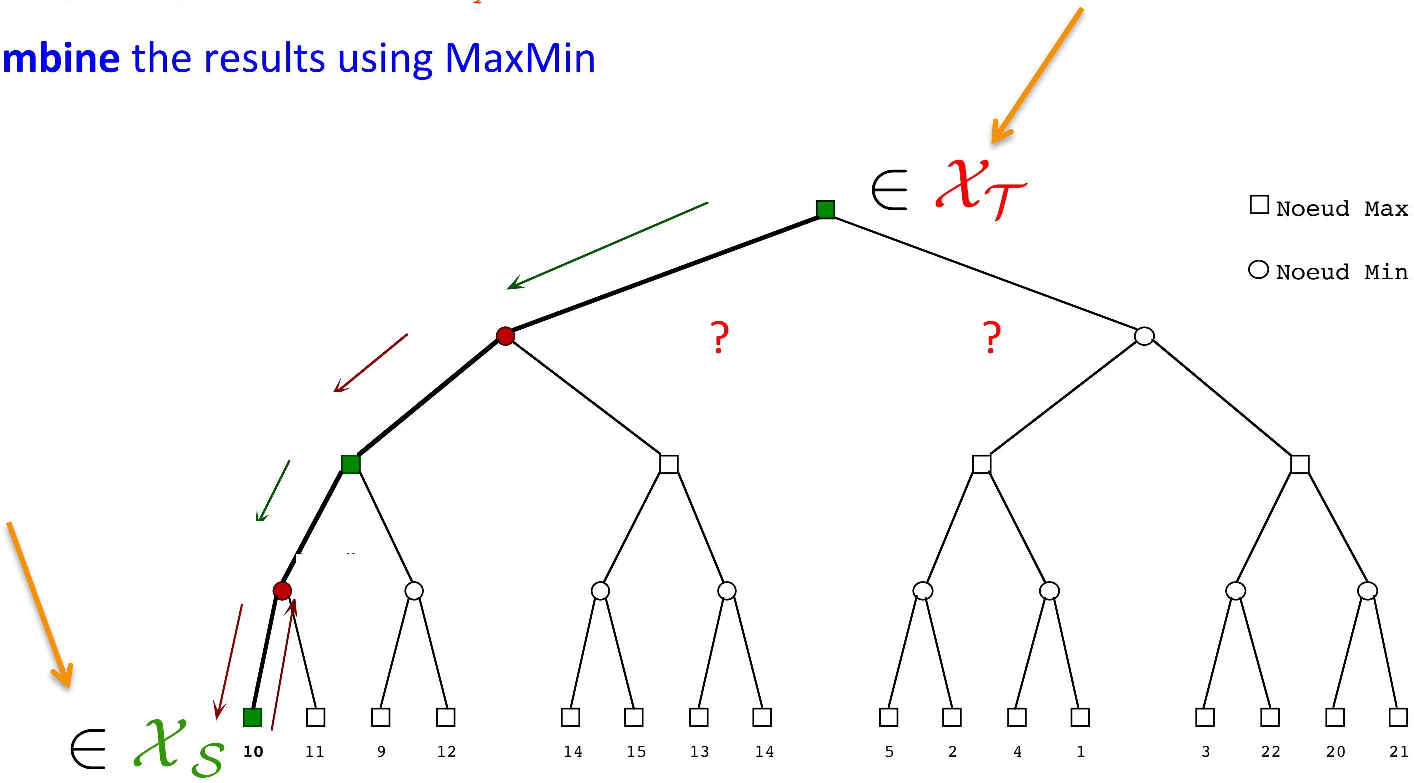
- Try to make use of this classifier to predict the class of **incomplete** series

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

Algorithms for games and transfer learning

Which move?

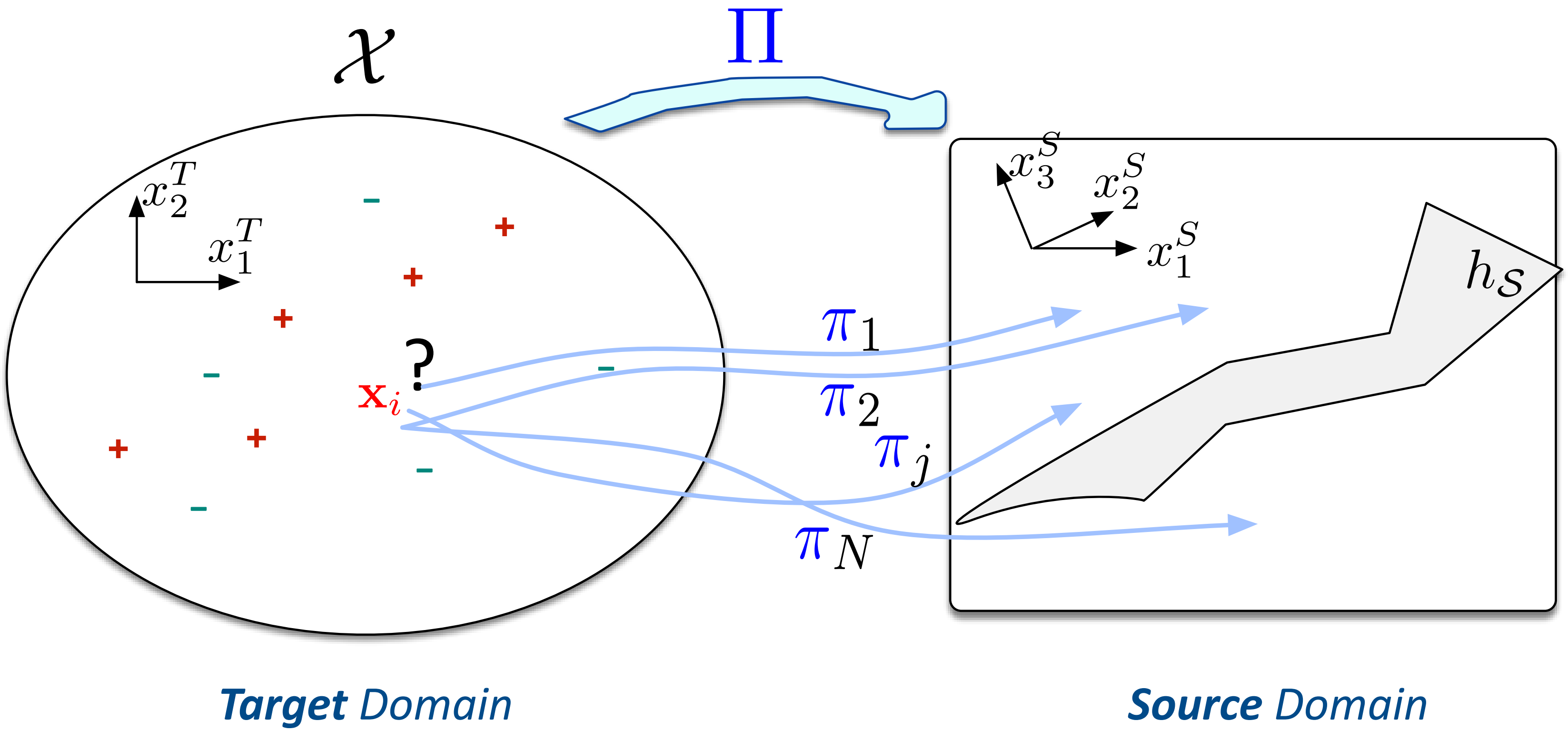
- Better evaluation function in $\mathcal{X}_S$
- Backup it (by transfer) for $\mathcal{X}_T$
- **Combine** the results using MaxMin



Outline

1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

TransBoost



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

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

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

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 $H_{\mathcal{T}} : \mathcal{X}_{\mathcal{T}} \rightarrow \mathcal{Y}_{\mathcal{T}}$:

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

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

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

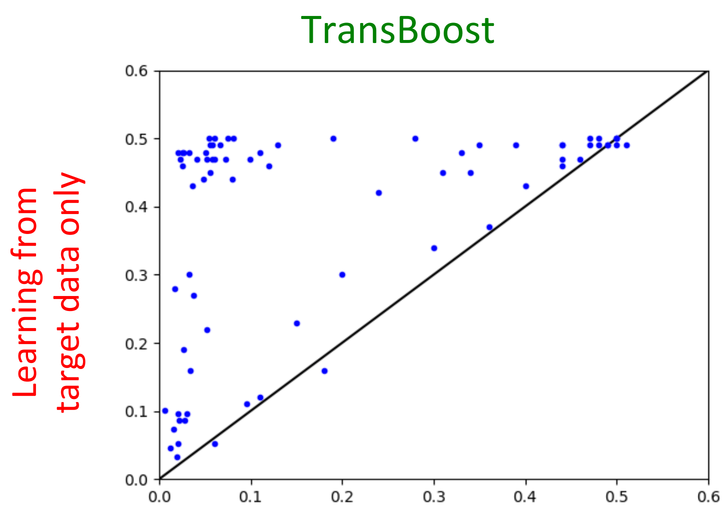


Figure 3: Comparison of error rates. y-axis: test error of the SVM classifier (without transfer). x-axis : test error of the TransBoost classifier with 10 boosting steps. The results of 75 experiments (each one repeated 100 times) are summed up in this graph.

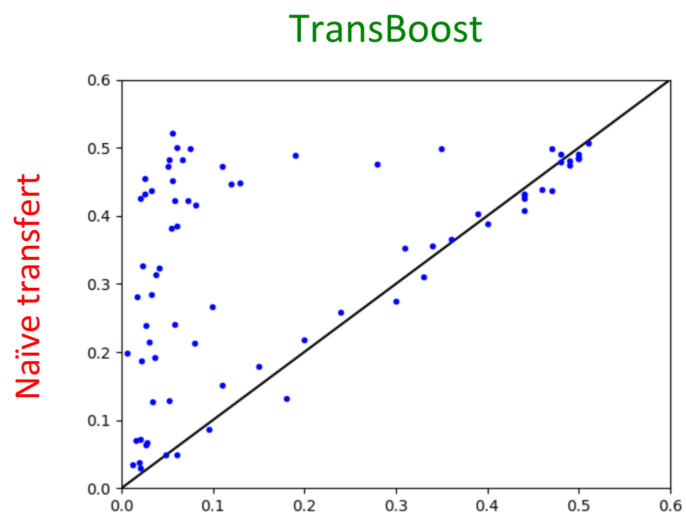


Figure 4: Comparison of error rates. y-axis: test error of the "naïve" transfer method. x-axis : test error of the TransBoost classifier with 10 boosting steps. The results of 75 experiments (each one repeated 100 times) are summed up in this graph.

Transfer learning

- Illustrations

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

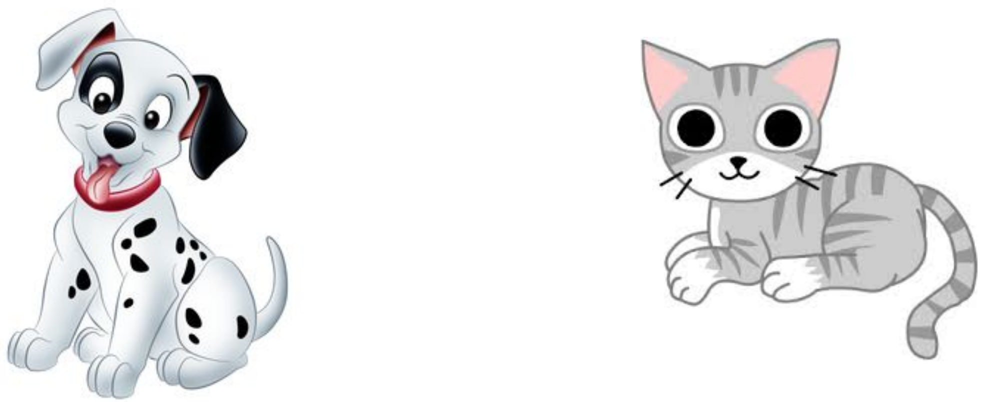
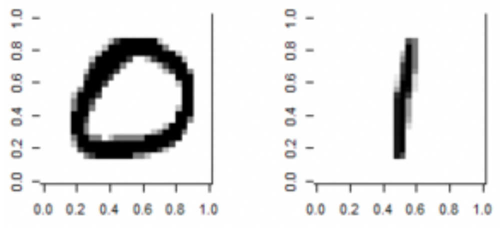
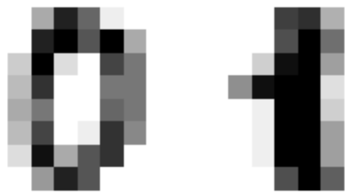


FIGURE 2: Model source transferred on the data target : is it a clip-art of a dog or a cat ?

- Illustrations

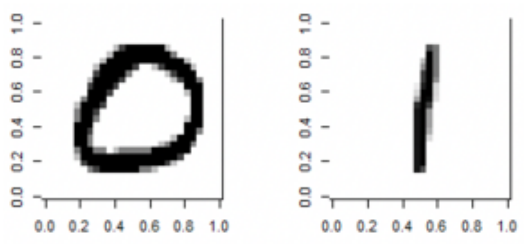


(a) Is it a zero or a one ?

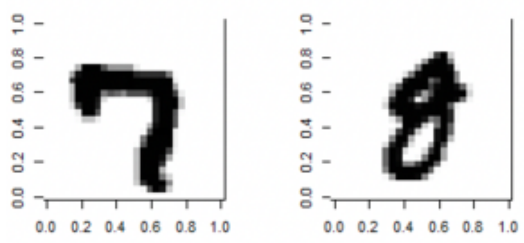


(b) Is it a zero or a one ?

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

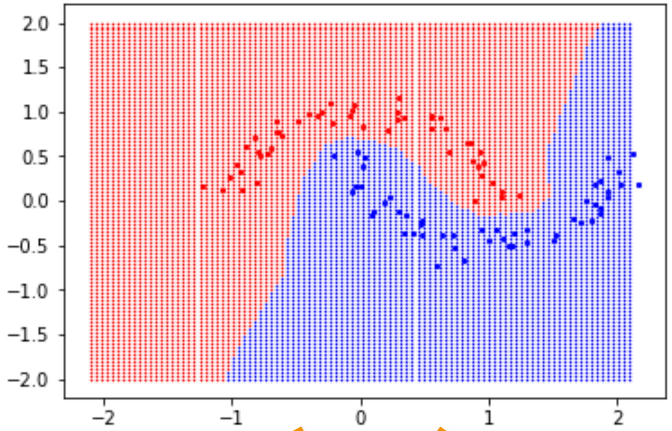


(a) Is it a zero or a one ?



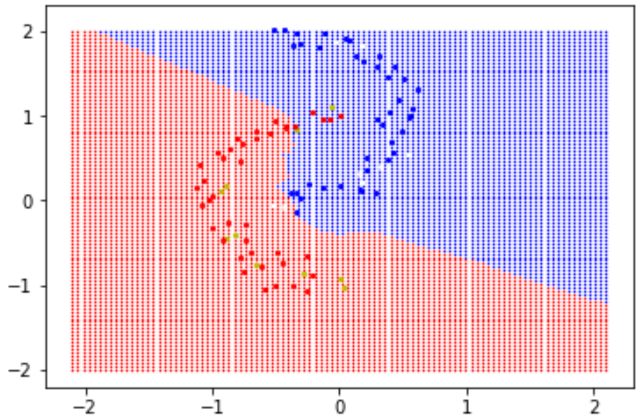
(b) Is it an eight or a seven ?

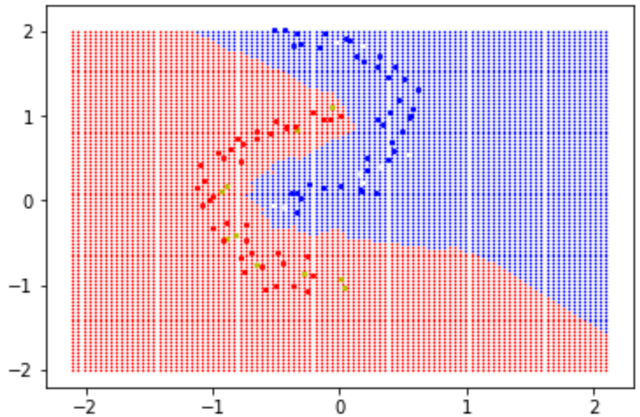
Transfer learning



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

Using Transboost





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: from the theory for boosting and for transfer as well

Theoretical guarantees

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

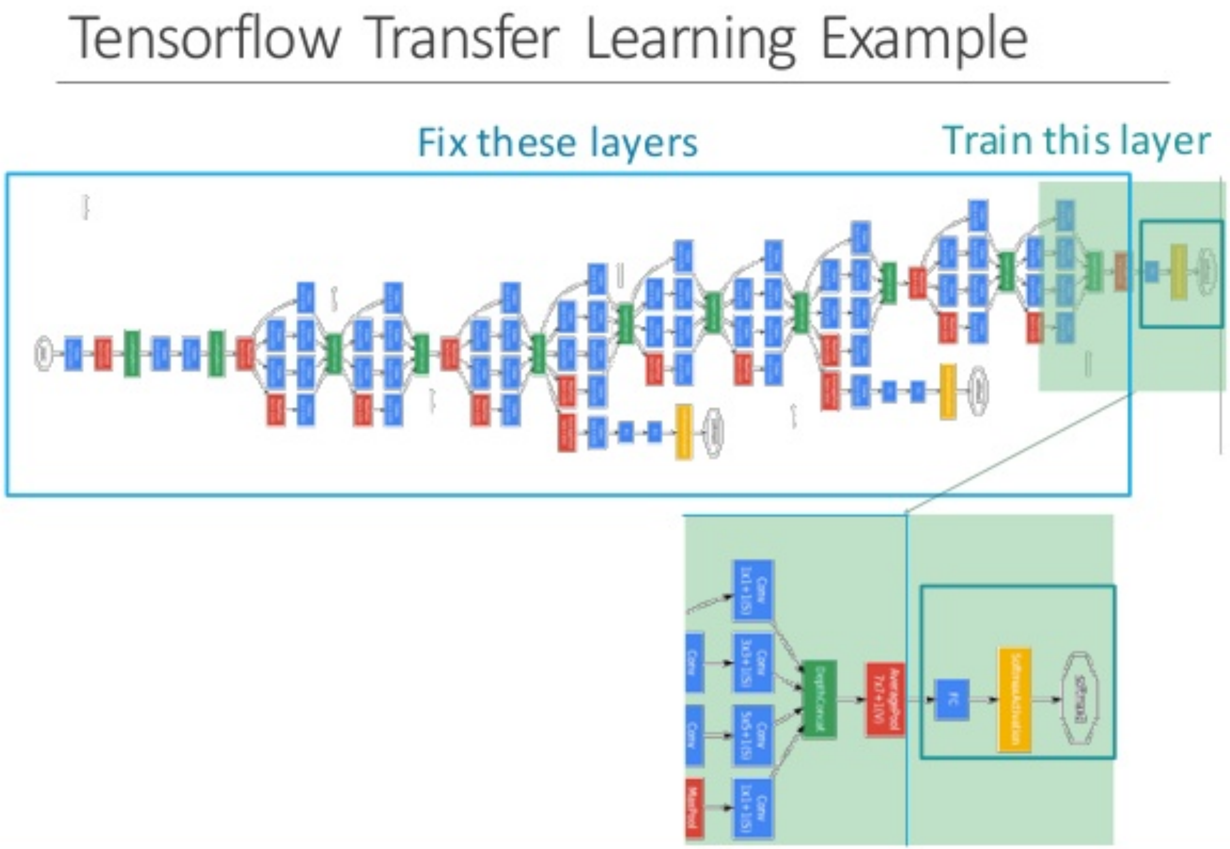
$$
\begin{aligned}
R_{\mathcal{T}}(\widehat{h}_{\mathcal{T}}) \ \leq \ & \omega\big(\widehat{R}_{\mathcal{S}}(\widehat{h}_{\mathcal{S}})\big) \\
& + \ 2\sqrt{\frac{2\, d_{\mathcal{H}_{\mathcal{S}}} \log(2em_{\mathcal{S}}/d_{\mathcal{H}_{\mathcal{S}}}) \ + \ 2\log(8/\delta)}{m_{\mathcal{S}}}} \\
& + \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

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

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

Transfer learning for deep neural networks

- Illustration



Outline

1. The online learning perspective
2. Early classification of time series
3. Early classification of time series and transfer learning
4. The TransBoost algorithm
5. Conclusion

Conclusion

1. **Online** learning
   - Links with **transfer learning**
   - **New scenarios** must be explored
     - Extrapolate likely changes of *h*
     - Transduction ("weak LUPI")

2. **Early** classification of time series
   - Can be solved as **a LUPI framework**
   - Can be seen as involving **transfer learning**

3. The **Transboost** algorithm
   - From LUPI to transfer learning

Online learning: back to the future

- Central question
  - Controlling the memory
    - What to keep from the past?
  - How to adapt the current hypothesis?

- Can TransBoost help?

Online learning

- Suppose
  - Online with small batches at each time step
  - The current batch is labeled (after prediction has been performed)
  - The source hypothesis is kNN ($k \geq 3$) with (all) past examples

- Use Transboost to learn projections
  - To past points
  - With constraints preventing to project on points close to the point projected
  - Make statistics about the most useful points

Bibliography

- A. Cornuéjols, S. Akkoyunlu, P-A. Murena and Raphaël Olivier (2017). Transfer Learning by boosting projections from target to source. *Conférence Francophone sur l'Apprentissage Automatique* (CAP'17), Grenoble, France, 28-30 juin 2017.
- Dachraoui, A., Bondu, A., & Cornuéjols, A. (2015). Early classification of time series as a non myopic sequential decision making problem. In *Joint European Conf. on Mach. Learning and Knowledge Discovery in Databases* (pp. 433-447).
- Dachraoui, A., Bondu, A., & Cornuéjols, A. (2016). A novel algorithm for online classification of time series when delaying decision is costly. In *Proc. of CAP-2016 (Conférence francophone sur l'Apprentissage Automatique),* Marseilles, France, July 4-7, 2016.
- G. Jaber, A. Cornuéjols, and P. Tarroux (2013) « Anticipative and Dynamic Adaptation to Concept Changes », ECML-PKDD-2013 (Workshop Real-World Challenges for Data Stream Mining ».
- V. Vapnik and A. Vashist (2009) "A new learning paradigm: Learning using privileged information". Neural Networks, vol. 22, no. 5, pp. 544–557, 2009

Supplementary material