Adaptive Learning and Mining for Data Streams and Frequent Patterns

Albert Bifet

Laboratory for Relational Algorithmics, Complexity and Learning **LARCA**
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Ph.D. dissertation, 24 April 2009
Advisors: Ricard Gavaldà and José L. Balcázar

LARCA

Future Data Mining

Future Data Mining

- Structured data
- Find Interesting Patterns
- Predictions
- On-line processing

Mining Evolving Massive Structured Data

The Disintegration of Persistence of Memory 1952-54

Salvador Dalí

The basic problem

Finding interesting structure on data

- Mining massive data
- Mining time varying data
- Mining on real time
- Mining structured data

Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Approximation algorithms

- Small error rate with high probability
- An algorithm $(\varepsilon, \delta)-$approximates F if it outputs $\tilde{F}$ for which $\Pr[|\tilde{F} - F| > \varepsilon F] < \delta$.

Tree Pattern Mining

Trees are sanctuaries.
Whoever knows how
to listen to them,
can learn the truth.

Herman Hesse

Given a dataset of trees, find the complete set of frequent subtrees

- Frequent Tree Pattern (FT):
  - Include all the trees whose support is no less than min_sup
- Closed Frequent Tree Pattern (CT):
  - Include no tree which has a super-tree with the same support
- $CT \subseteq FT$

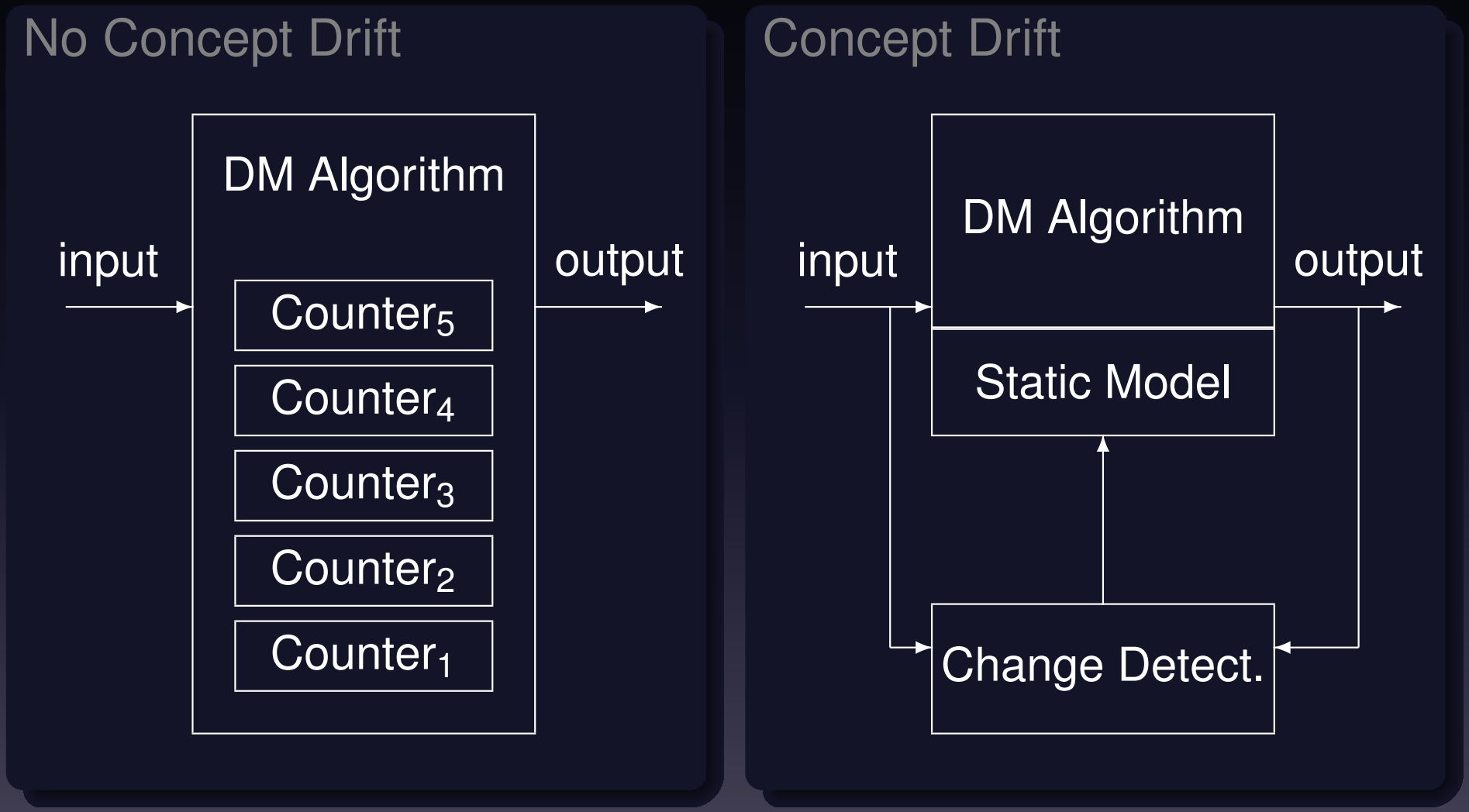
Outline

Outline

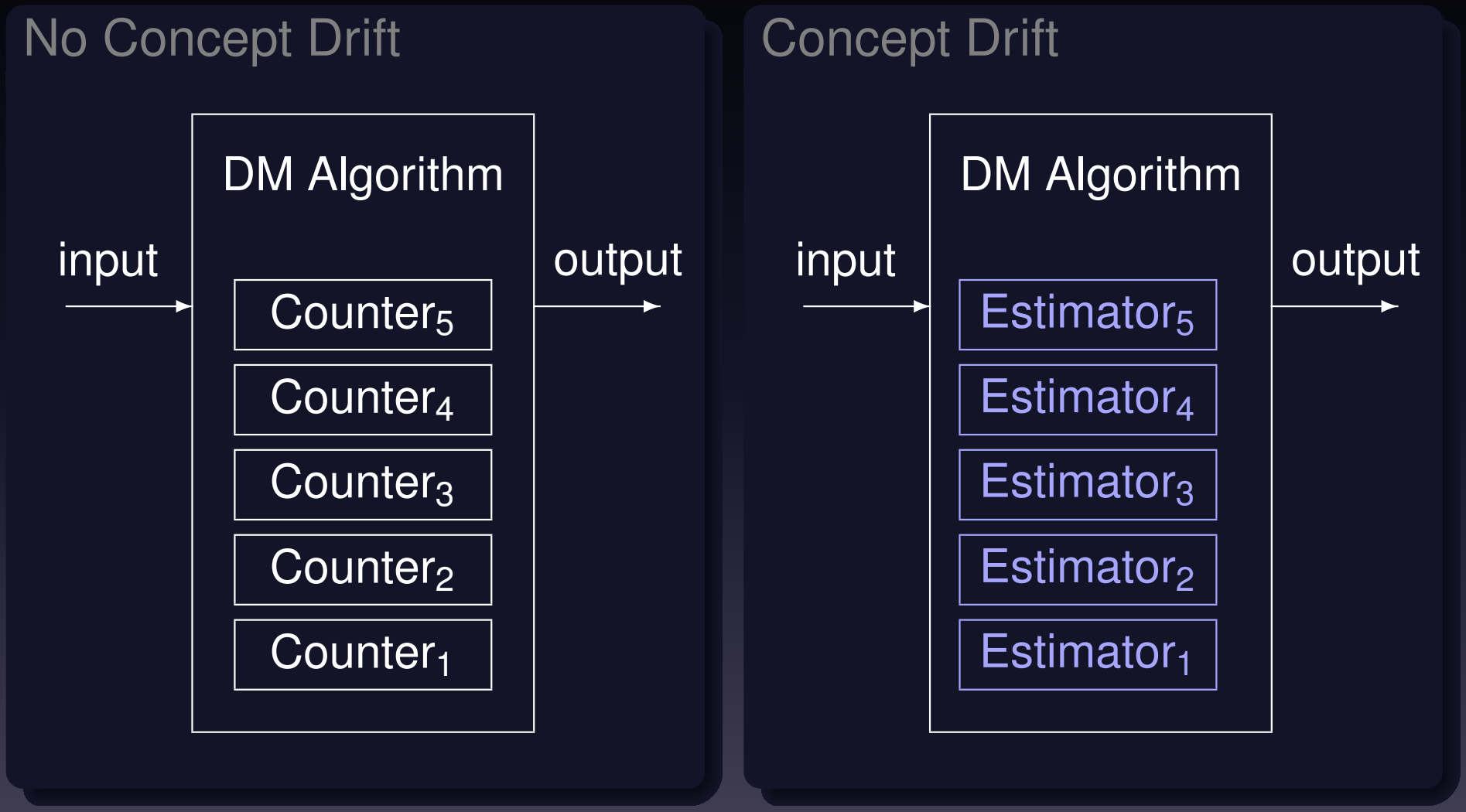
Outline

Outline

Data Mining Algorithms with Concept Drift



Data Mining Algorithms with Concept Drift



Time Change Detectors and Predictors

(1) General Framework

Problem

Given an input sequence $x_1, x_2, \ldots, x_t, \ldots$ we want to output at instant t

- a prediction $\widehat{x}_{t+1}$ minimizing prediction error:

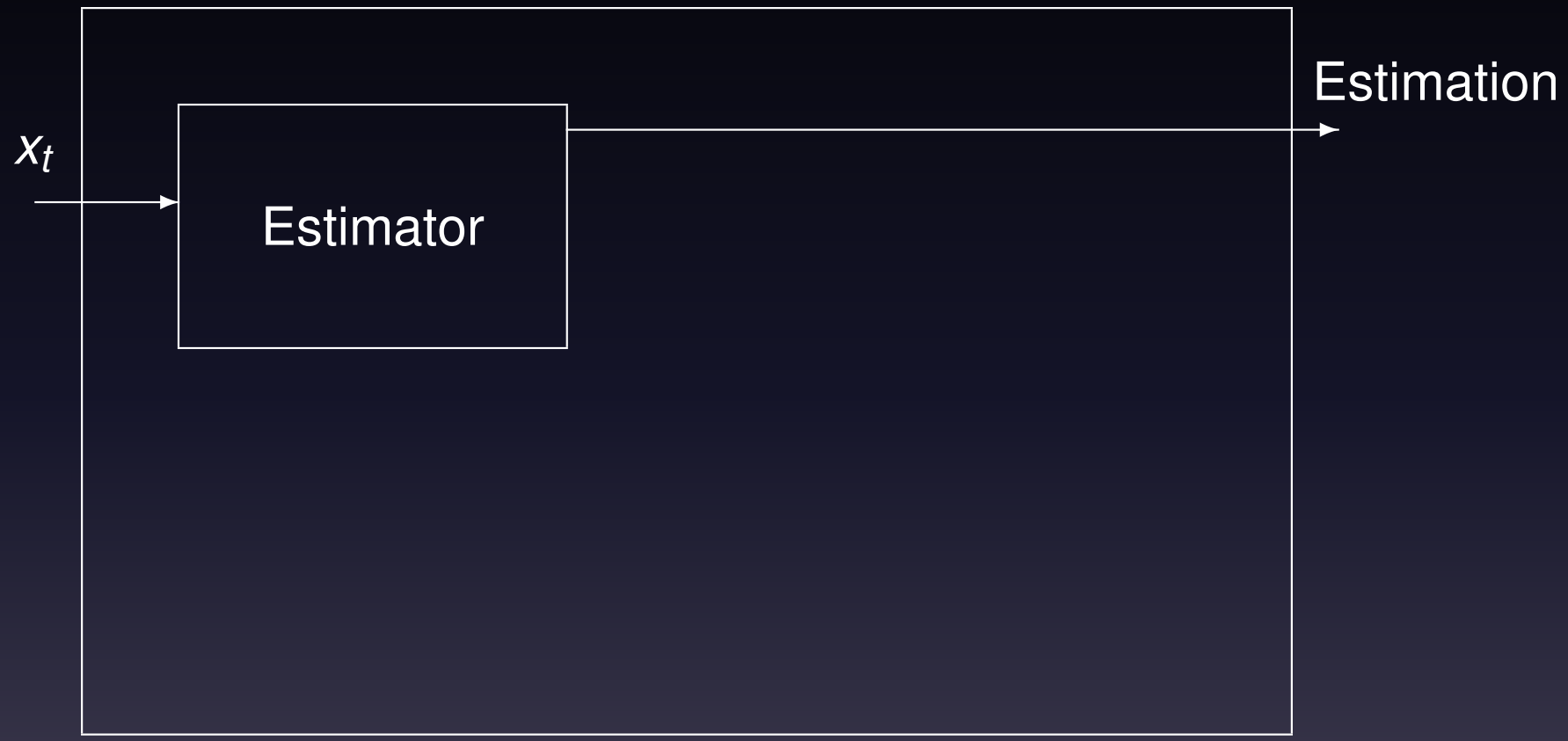
$$|\widehat{x}_{t+1} - x_{t+1}|$$

- an alert if change is detected

considering distribution changes overtime.

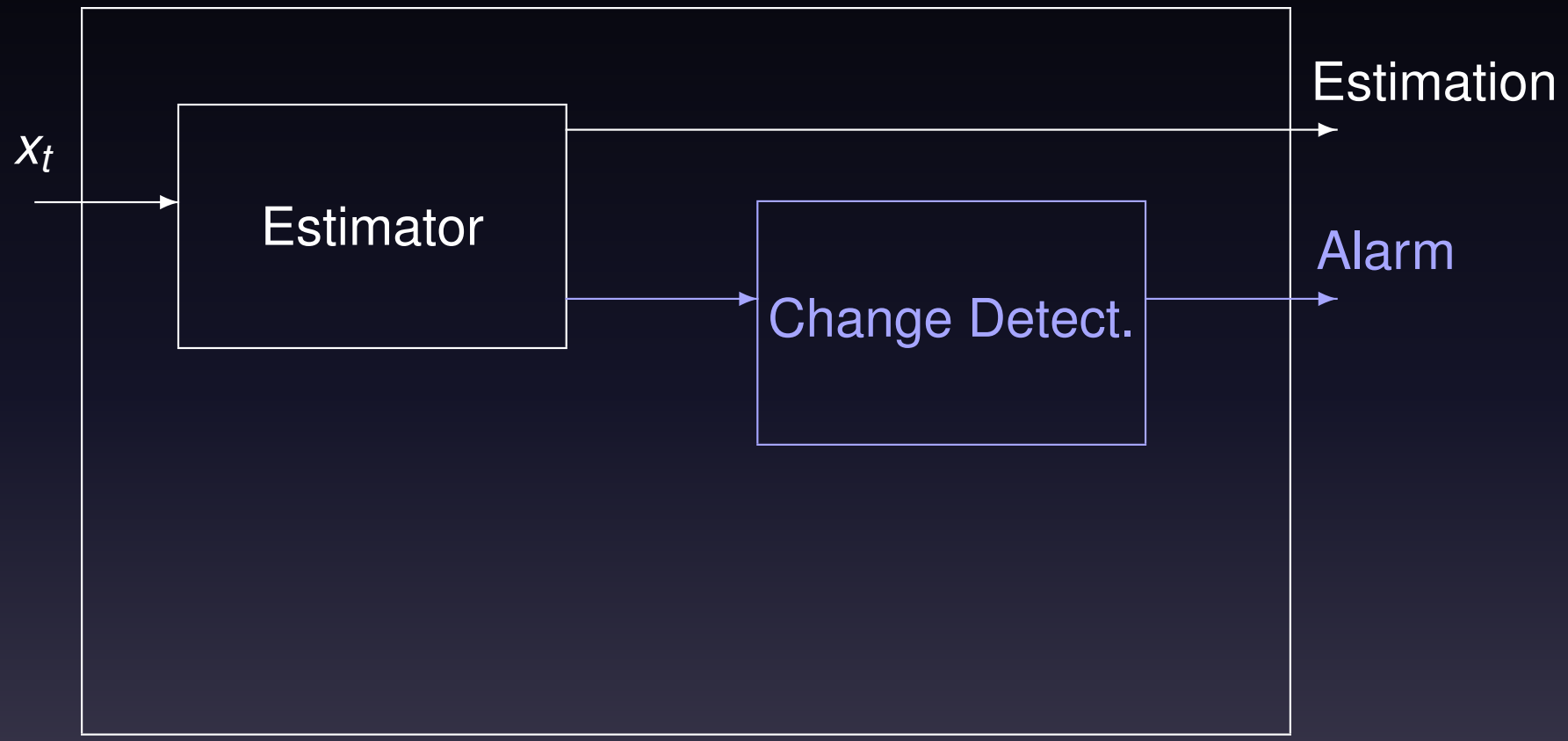
Time Change Detectors and Predictors

(1) General Framework



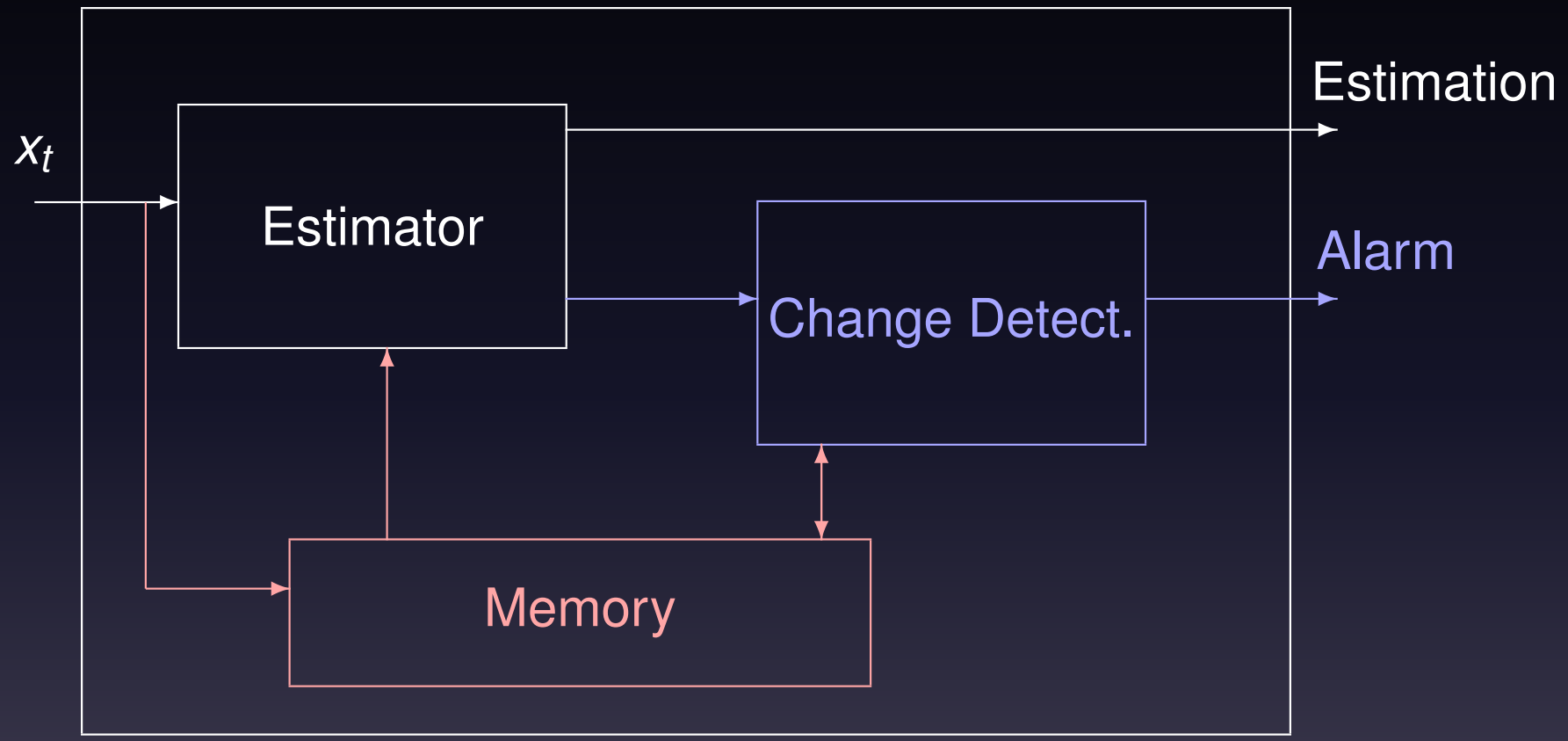
Time Change Detectors and Predictors

(1) General Framework



Time Change Detectors and Predictors

(1) General Framework



Optimal Change Detector and Predictor

(1) General Framework

- High accuracy
- Fast detection of change
- Low false positives and false negatives ratios
- Low computational cost: minimum space and time needed
- Theoretical guarantees
- No parameters needed
- Estimator with Memory and Change Detector

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 1

ADWIN: ADAPTIVE WINDOWING ALGORITHM

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 1 W_1 = 01010110111111

ADWIN: ADAPTIVE WINDOWING ALGORITHM

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 10 W_1 = 1010110111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 101 W_1 = 0101101111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 101010110111111

W_0= 1010 W_1 = 10110111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 10101 W_1 = 0110111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 101010 W_1 = 110111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 1010101 W_1 = 10111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111

W_0= 10101011 W_1 = 0111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 101010110111111 $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$: CHANGE DET.!

W_0= 101010110 W_1 = 111111

ADWIN: ADAPTIVE WINDOWING ALGORITHM

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 1010101101111111 Drop elements from the tail of W

W_0= 101010110 W_1 = 111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Example

W= 01010110111111 Drop elements from the tail of W

W_0= 101010110 W_1 = 111111

ADWIN: Adaptive Windowing Algorithm

1 Initialize Window W
2 **for** each $t > 0$
3 **do** $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
4 **repeat** Drop elements from the tail of W
5 **until** $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| < \varepsilon_c$ holds
6 for every split of W into $W = W_0 \cdot W_1$
7 Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

(2) ADWIN

Theorem

At every time step we have:

1. (False positive rate bound). *If μ_t remains constant within W, the probability that* ADWIN *shrinks the window at this step is at most δ.*
2. (False negative rate bound). *Suppose that for* some *partition of W in two parts $W_0 W_1$ (where W_1 contains the most recent items) we have $|\mu_{W_0} - \mu_{W_1}| > 2\varepsilon_c$. Then with probability $1-\delta$* ADWIN *shrinks W to W_1, or shorter.*

ADWIN tunes itself to the data stream at hand, with no need for the user to hardwire or precompute parameters.

Algorithm ADaptive Sliding WINdow

(2) ADWIN

ADWIN using a Data Stream Sliding Window Model,

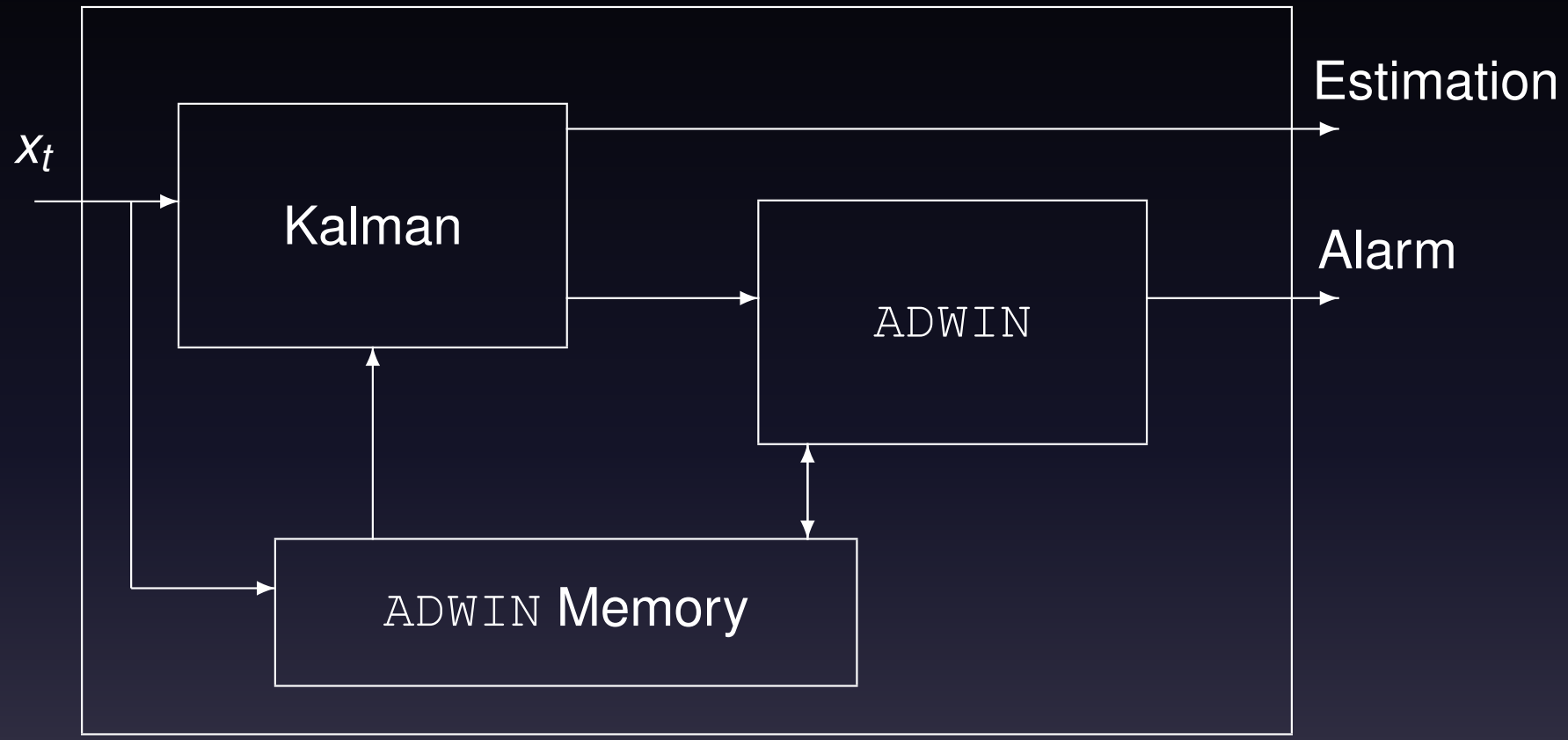
- can provide the exact counts of 1's in $O(1)$ time per point.
- tries $O(\log W)$ cutpoints
- uses $O(\frac{1}{\varepsilon} \log W)$ memory words
- the processing time per example is $O(\log W)$ (amortized and worst-case).

Sliding Window Model

	1010101	101	11	1	1
Content:	4	2	2	1	1
Capacity:	7	3	2	1	1

K−ADWIN = ADWIN + Kalman Filtering

(2) ADWIN



$R = W^2/50$ and $Q = 200/W$ (theoretically justifiable), where W is the length of the window maintained by ADWIN.

Classification

(3) Mining Algorithms

Definition

Given n_C different classes, a classifier algorithm builds a model that predicts for every unlabeled instance I the class C to which it belongs with accuracy.

Classification Mining Algorithms

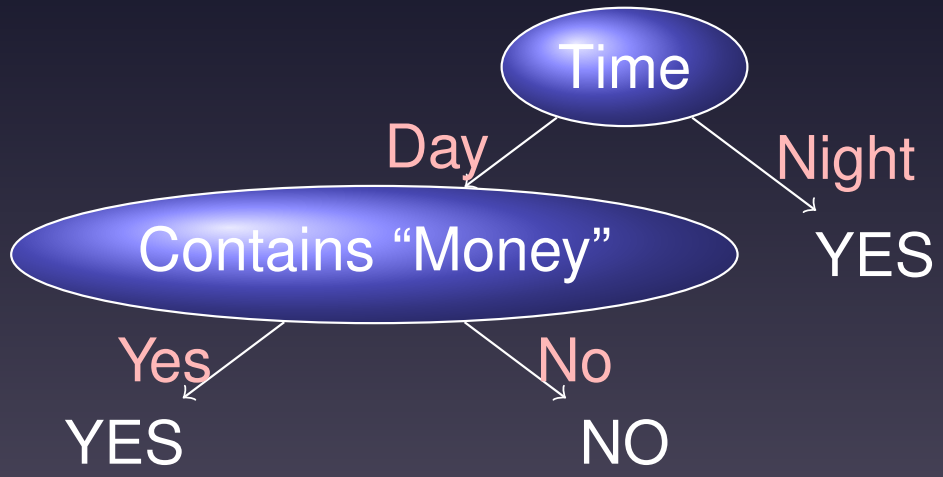
- Naïve Bayes
- Decision Trees
- Ensemble Methods

Hoeffding Tree / CVFDT

Hoeffding Tree : VFDT

Pedro Domingos and Geoff Hulten.
Mining high-speed data streams. 2000

- With high probability, constructs an identical model that a traditional (greedy) method would learn
- With theoretical guarantees on the error rate



VFDT / CVFDT

(3) Mining Algorithms

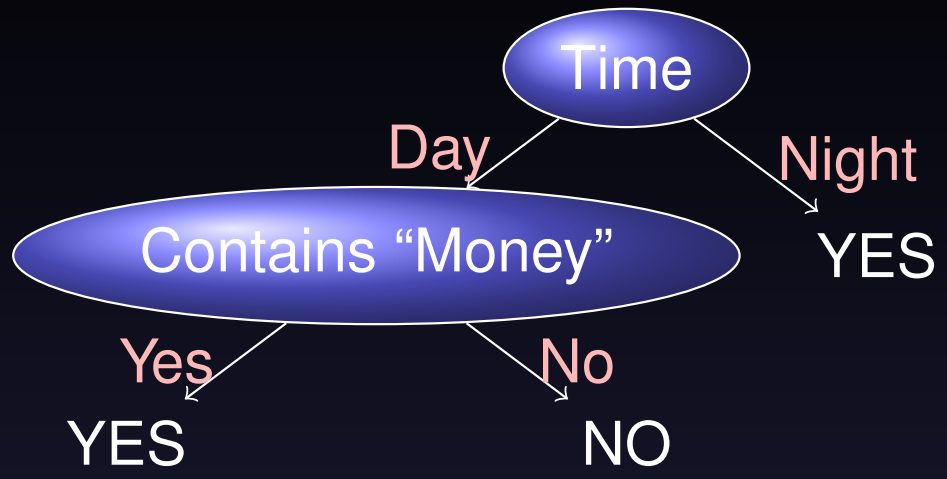
Concept-adapting Very Fast Decision Trees: CVFDT

G. Hulten, L. Spencer, and P. Domingos.
Mining time-changing data streams. 2001

- It keeps its model consistent with a sliding window of examples
- Construct “alternative branches” as preparation for changes
- If the alternative branch becomes more accurate, switch of tree branches occurs

Decision Trees: CVFDT

(3) Mining Algorithms



No theoretical guarantees on the error rate of CVFDT

CVFDT parameters :

1. W: is the example window size.
2. T_0: number of examples used to check at each node if the splitting attribute is still the best.
3. T_1: number of examples used to build the alternate tree.
4. T_2: number of examples used to test the accuracy of the alternate tree.

Decision Trees: Hoeffding Adaptive Tree

(3) Mining Algorithms

Hoeffding Adaptive Tree:

- replace frequency statistics counters by estimators
  - don't need a window to store examples, due to the fact that we maintain the statistics data needed with estimators
- change the way of checking the substitution of alternate subtrees, using a change detector with theoretical guarantees

Summary:

1. Theoretical guarantees
2. No Parameters

What is MOA?

{M}assive {O}nline {A}nalysis is a framework for online learning from data streams.

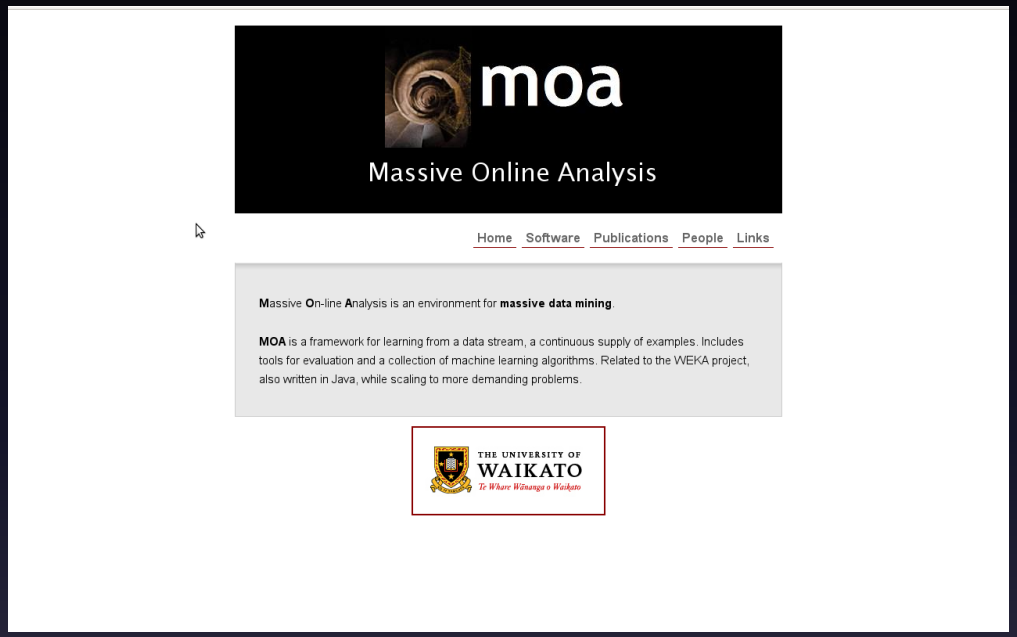
- It is closely related to WEKA
- It includes a collection of offline and online as well as tools for evaluation:
  - boosting and bagging
  - Hoeffding Trees

  with and without Naïve Bayes classifiers at the leaves.

Ensemble Methods

(4) MOA for Evolving Data Streams

http://www.cs.waikato.ac.nz/~abifet/MOA/

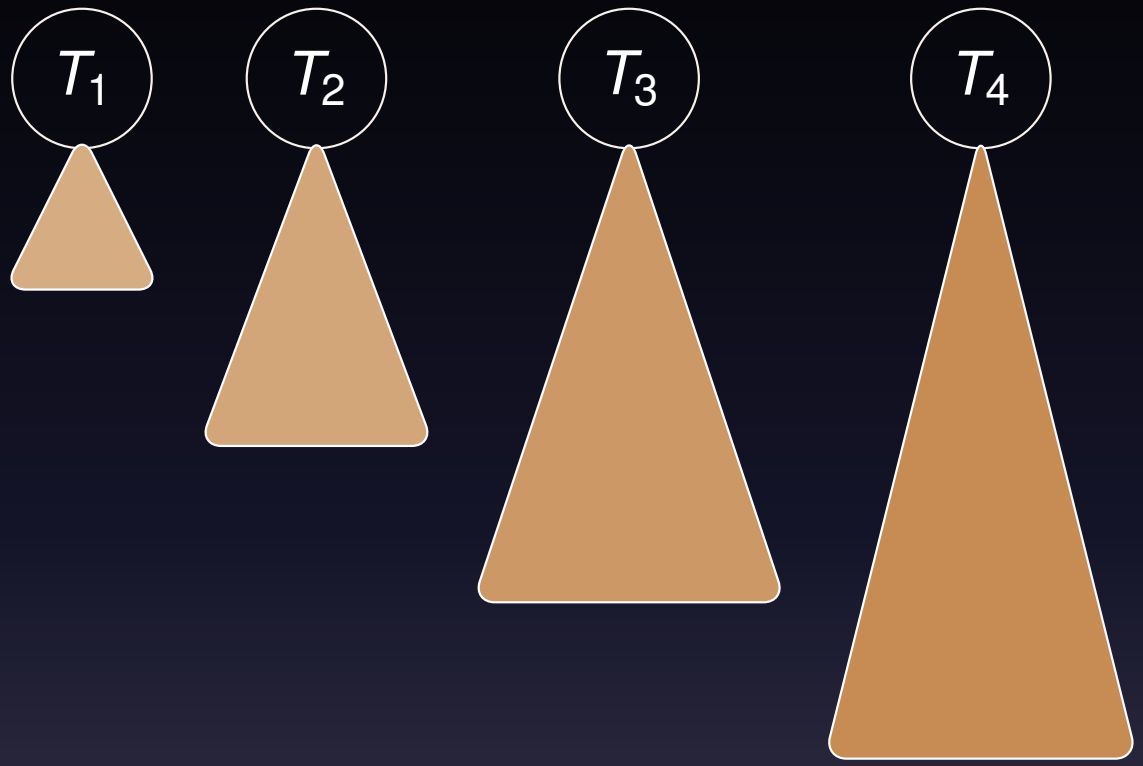


New ensemble methods:

- ADWIN bagging: When a change is detected, the worst classifier is removed and a new classifier is added.
- Adaptive-Size Hoeffding Tree bagging

Adaptive-Size Hoeffding Tree

(5) ASHT



Ensemble of trees of different size

- smaller trees adapt more quickly to changes,
- larger trees do better during periods with little change
- diversity

Adaptive-Size Hoeffding Tree

(5) ASHT

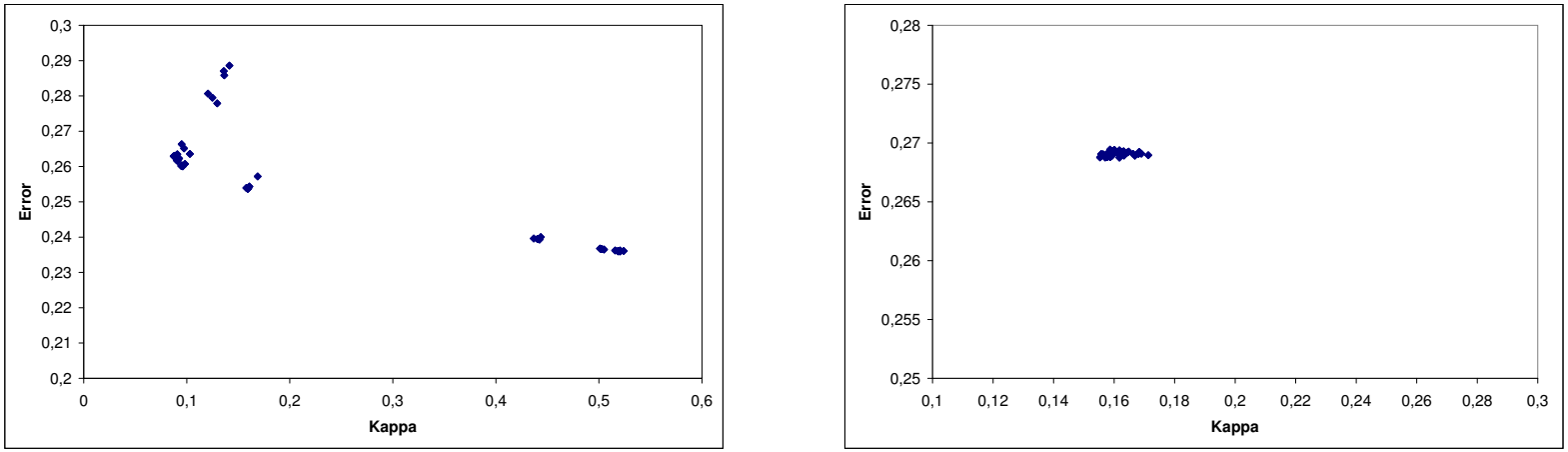


Figure: Kappa-Error diagrams for ASHT bagging (left) and bagging (right) on dataset RandomRBF with drift, plotting 90 pairs of classifiers.

Adaptive-Size Hoeffding Tree

(5) ASHT

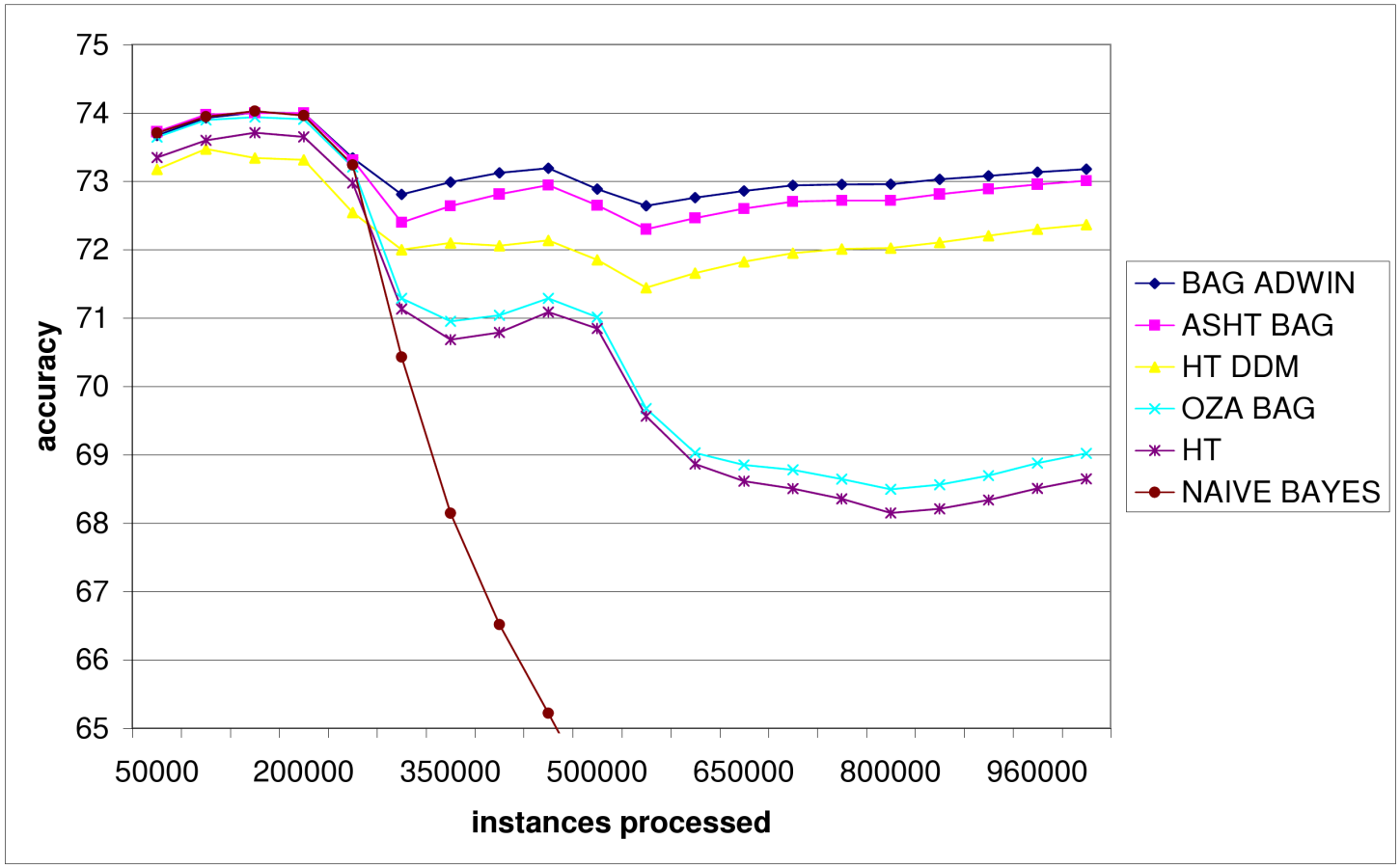


Figure: Accuracy and size on dataset LED with three concept drifts.

Main contributions (i)

Mining Evolving Data Streams

1. General Framework for Time Change Detectors and Predictors
2. `ADWIN`
3. Mining methods: Naive Bayes, Decision Trees, Ensemble Methods
4. MOA for Evolving Data Streams
5. Adaptive-Size Hoeffding Tree

Outline

Mining Closed Frequent Trees

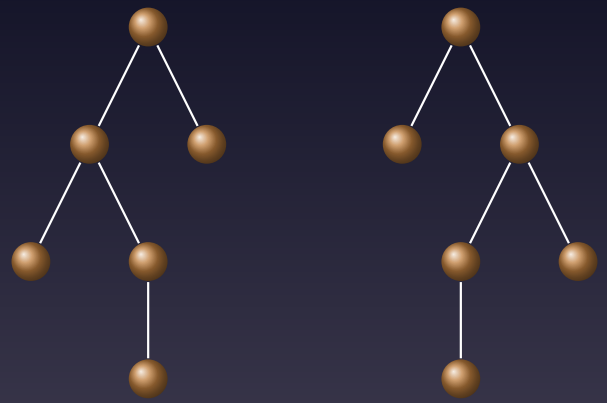
Our trees are:

- Labeled and Unlabeled
- Ordered and Unordered

Our subtrees are:

- Induced
- Top-down

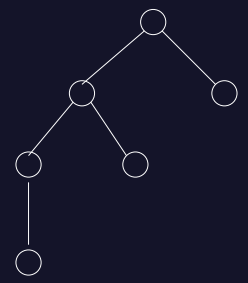
Two different ordered trees
but the same unordered tree



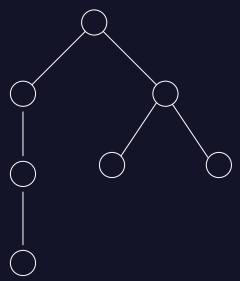
A tale of two trees

Consider $\mathscr{D} = \{A, B\}$, where

A:

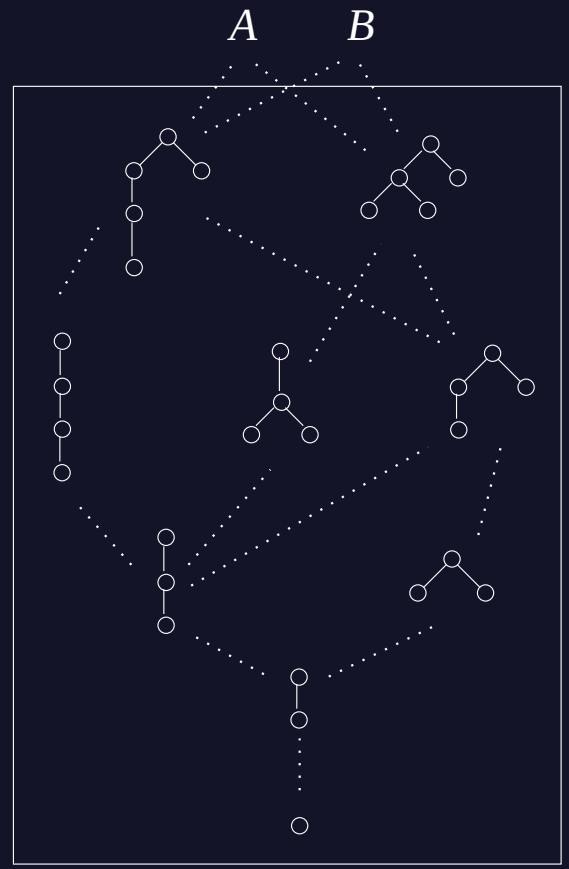


B:



and let $min\_sup = 2$.

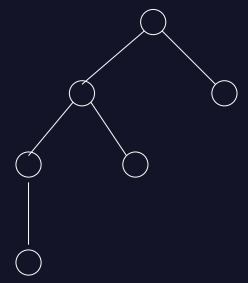
Frequent subtrees



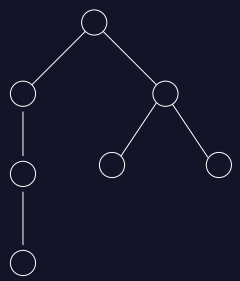
A tale of two trees

Consider $\mathscr{D} = \{A, B\}$, where

A:

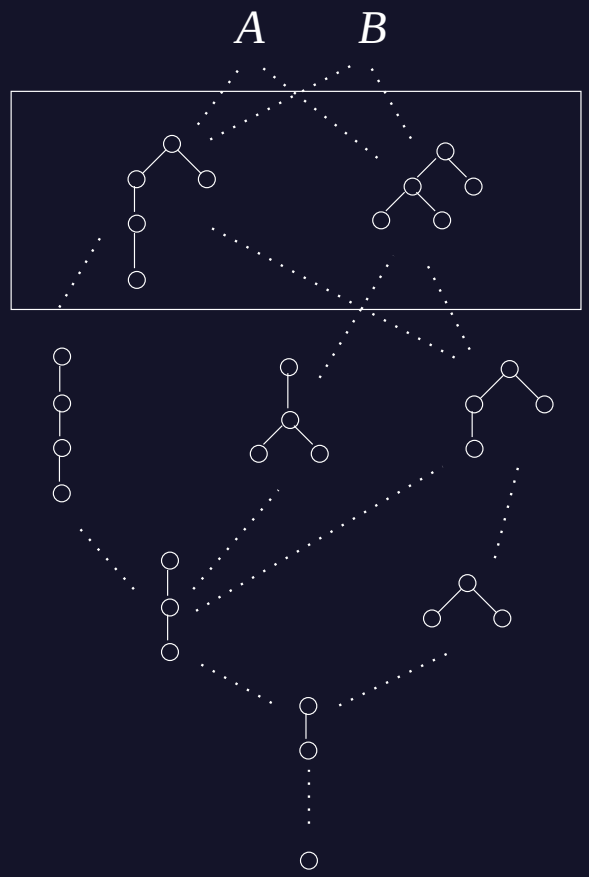


B:



and let $min\_sup = 2$.

Closed subtrees



Mining Closed Unordered Subtrees

(6) Unlabeled Closed Frequent Tree Method

```
CLOSED_SUBTREES(t, 𝒟, min_sup, T)
 1
 2
 3  for every t′ that can be extended from t in one step
 4       do if Support(t′) ≥ min_sup
 5             then T ← CLOSED_SUBTREES(t′, 𝒟, min_sup, T)
 6
 7
 8
 9
10  return T
```

Mining Closed Unordered Subtrees

(6) Unlabeled Closed Frequent Tree Method

```
CLOSED_SUBTREES(t, 𝒟, min_sup, T)
 1  if not CANONICAL_REPRESENTATIVE(t)
 2     then return T
 3  for every t′ that can be extended from t in one step
 4        do if Support(t′) ≥ min_sup
 5              then T ← CLOSED_SUBTREES(t′, 𝒟, min_sup, T)
 6
 7
 8
 9
10  return T
```

Mining Closed Unordered Subtrees

(6) Unlabeled Closed Frequent Tree Method

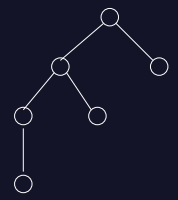
```
CLOSED_SUBTREES(t, 𝒟, min_sup, T)
 1  if not CANONICAL_REPRESENTATIVE(t)
 2      then return T
 3  for every t' that can be extended from t in one step
 4          do if Support(t') ≥ min_sup
 5                then T ← CLOSED_SUBTREES(t', 𝒟, min_sup, T)
 6          do if Support(t') = Support(t)
 7                then t is not closed
 8  if t is closed
 9      then insert t into T
10  return T
```

Example

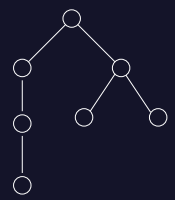
$\mathscr{D} = \{A, B\}$

$min\_sup = 2.$

$\langle A \rangle = (0,1,2,3,2,1)$



$\langle B \rangle = (0,1,2,3,1,2,2)$



(0)

$(0,1)$

$(0,1,1)$

$(0,1,2)$

$(0,1,2,1)$

$(0,1,2,2)$

$(0,1,2,3)$

$(0,1,2,2,1)$

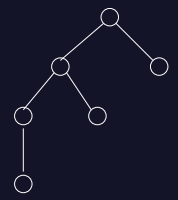
$(0,1,2,3,1)$

Example

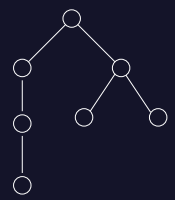
$\mathscr{D} = \{A, B\}$

$min\_sup = 2.$

$\langle A \rangle = (0,1,2,3,2,1)$



$\langle B \rangle = (0,1,2,3,1,2,2)$



(0)

$(0,1)$

$(0,1,1)$

$(0,1,2)$

$(0,1,2,1)$

$(0,1,2,2)$

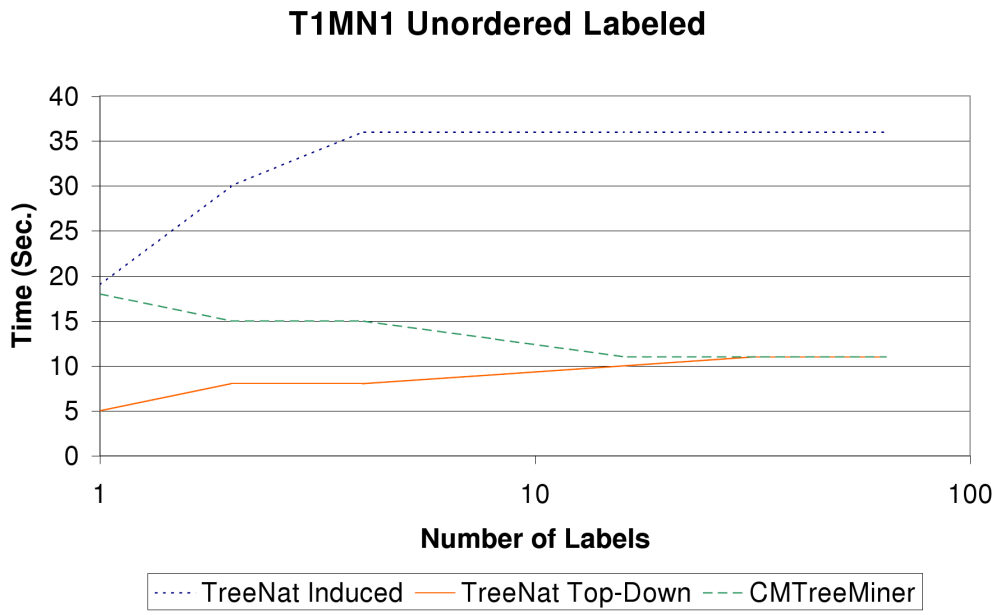
$(0,1,2,3)$

$(0,1,2,2,1)$

$(0,1,2,3,1)$

Experimental results

(6) Unlabeled Closed Frequent Tree Method



TreeNat

- Unlabeled Trees
- Top-Down Subtrees
- No Occurrences

CMTreeMiner

- Labeled Trees
- Induced Subtrees
- Occurrences

Closure Operator on Trees

(7) Closure Operator

- $\mathscr{D}$: the finite input dataset of trees
- $\mathscr{T}$: the (infinite) set of all trees

Definition

We define the following the Galois connection pair:

- For finite $A \subseteq \mathscr{D}$
  - $\sigma(A)$ is the set of subtrees of the A trees in $\mathscr{T}$

$$\sigma(A) = \{t \in \mathscr{T} \mid \forall t' \in A (t \preceq t')\}$$

- For finite $B \subset \mathscr{T}$
  - $\tau_{\mathscr{D}}(B)$ is the set of supertrees of the B trees in $\mathscr{D}$

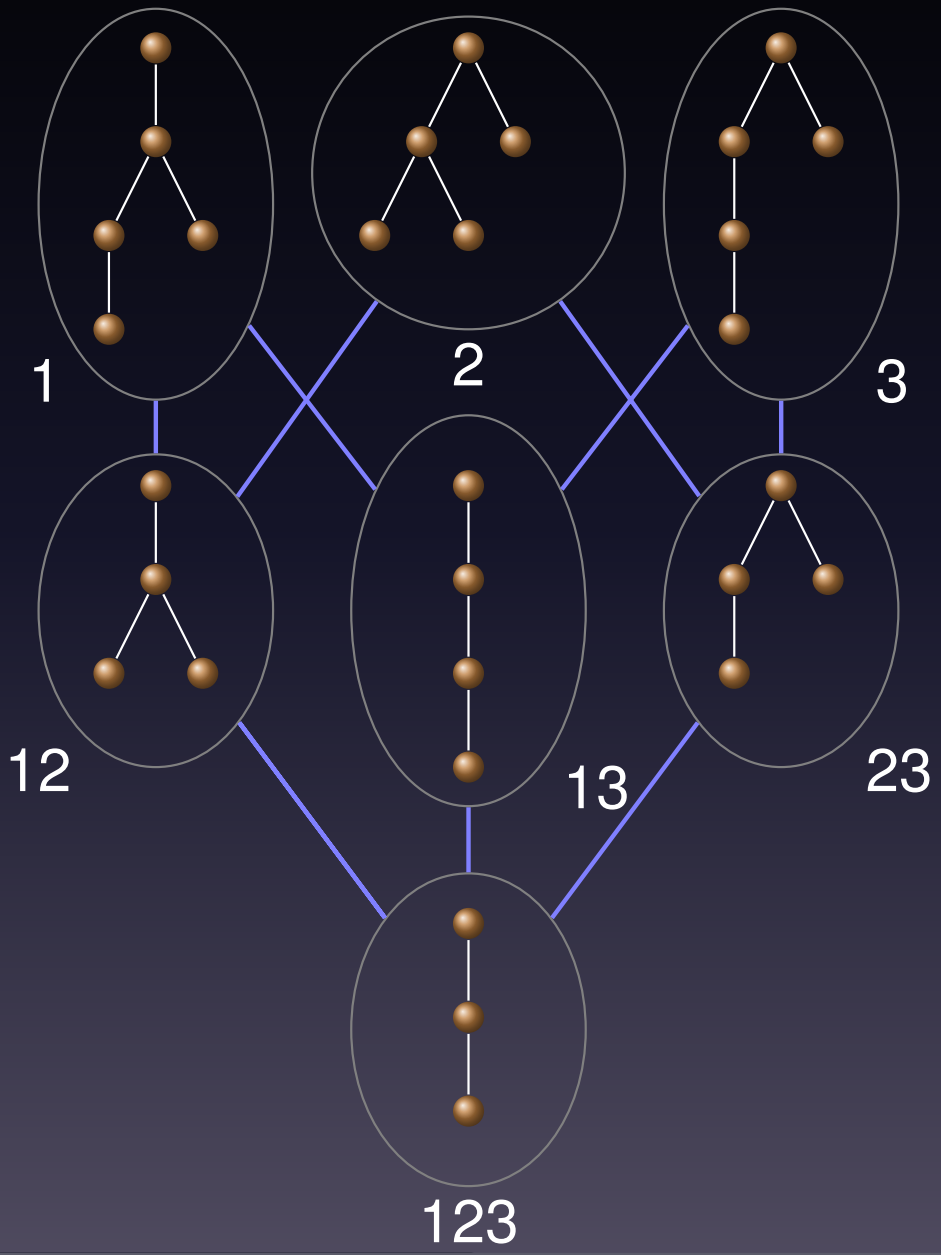
$$\tau_{\mathscr{D}}(B) = \{t' \in \mathscr{D} \mid \forall t \in B (t \preceq t')\}$$

Closure Operator

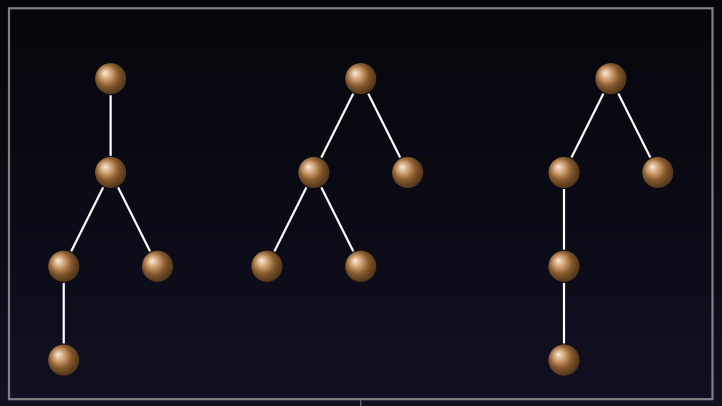
The composition $\Gamma_{\mathscr{D}} = \sigma \circ \tau_{\mathscr{D}}$ is a closure operator.

Galois Lattice of closed set of trees

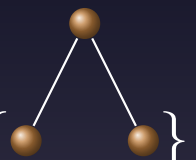
(7) Closure Operator

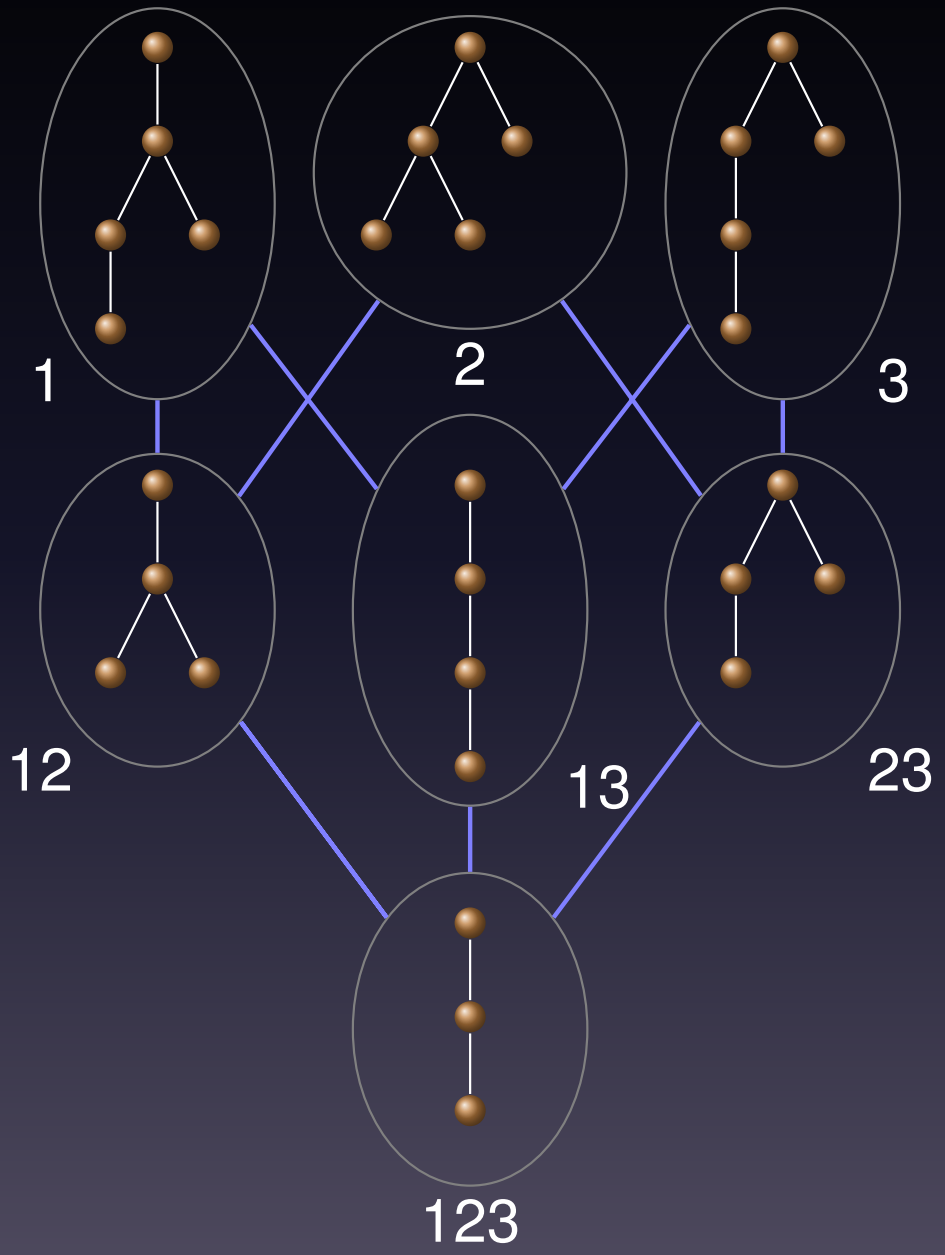


Galois Lattice of closed set of trees

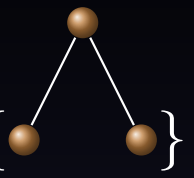


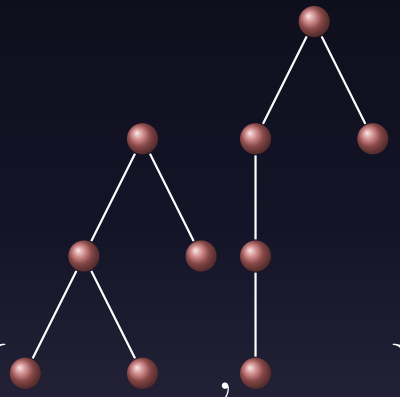
$\mathscr{D}$

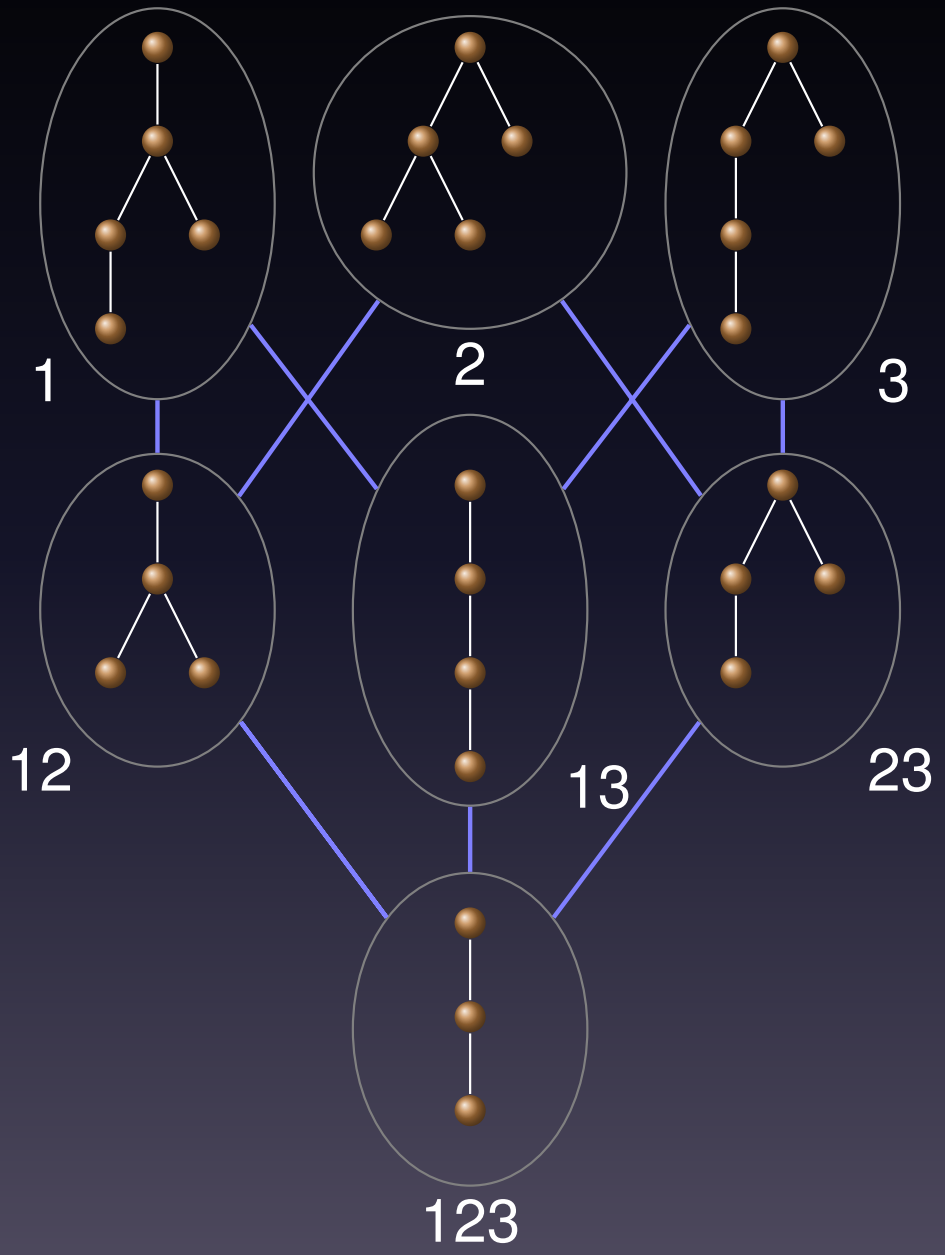
$B = \{$  $\}$



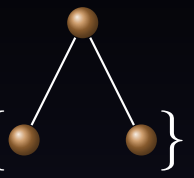
Galois Lattice of closed set of trees

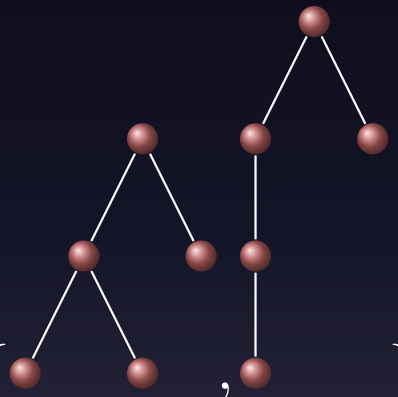
$B = \{$  $\}$

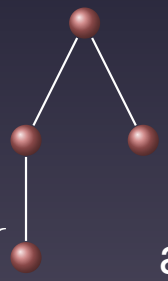
$\tau_{\mathscr{D}}(B) = \{$  $\}$

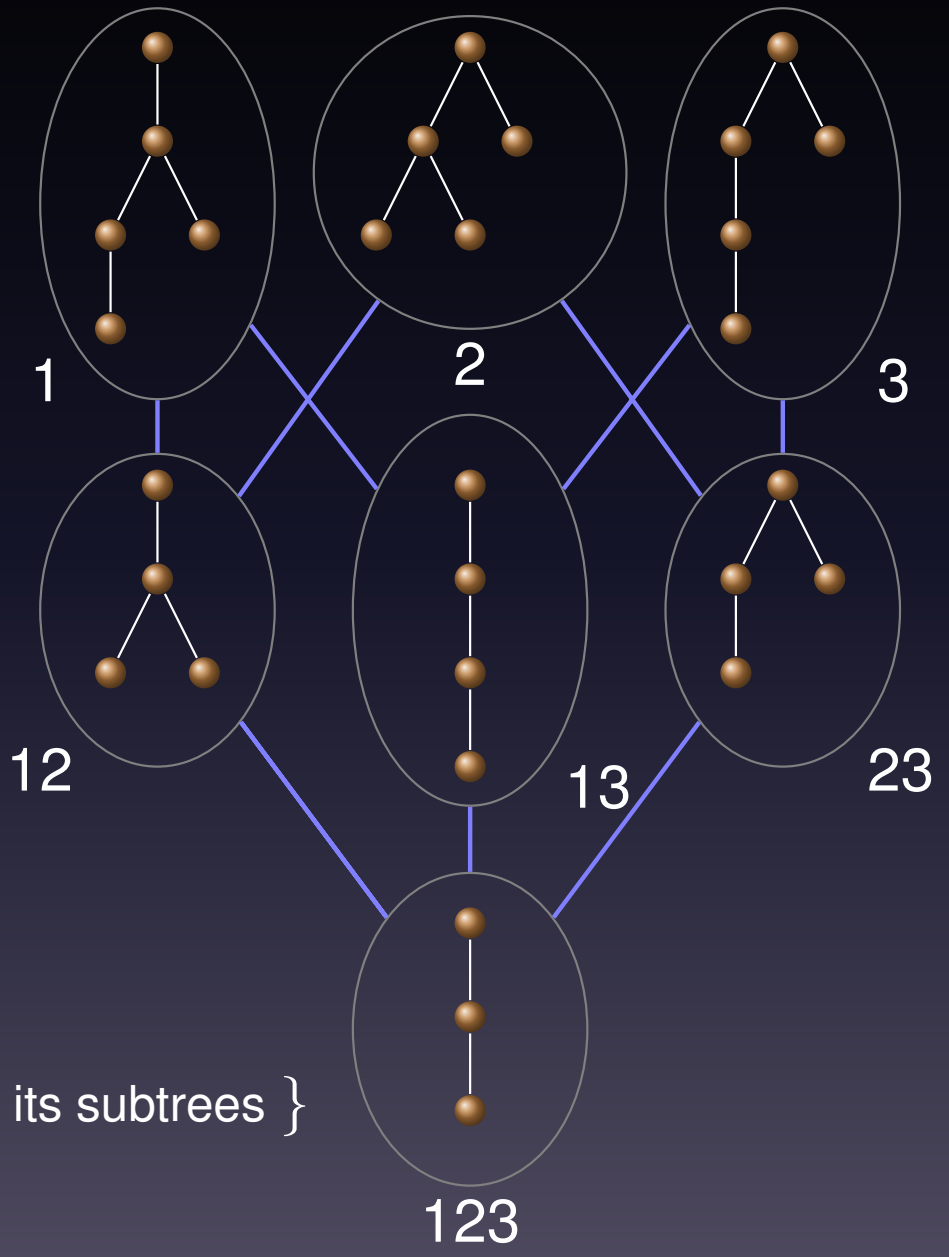


Galois Lattice of closed set of trees

$B = \{$  $\}$

$\tau_{\mathscr{D}}(B) = \{$  $\}$

$\Gamma_{\mathscr{D}}(B) = \sigma \circ \tau_{\mathscr{D}(B)} = \{$  and its subtrees $\}$

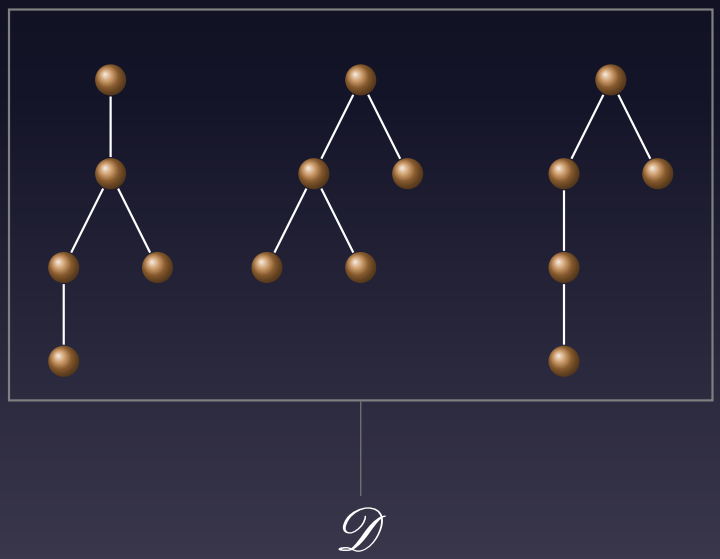


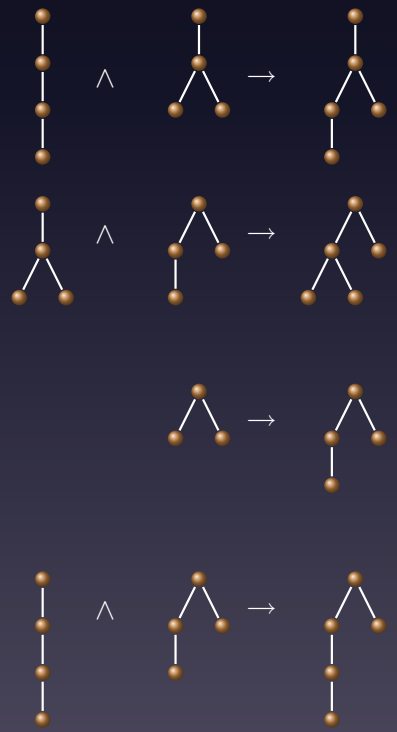
Mining Implications from Lattices of Closed Trees

(8) Association Rules

Problem

Given a dataset $\mathscr{D}$ of rooted, unlabeled and unordered trees, find a “basis”: a set of rules that are sufficient to infer all the rules that hold in the dataset $\mathscr{D}$.



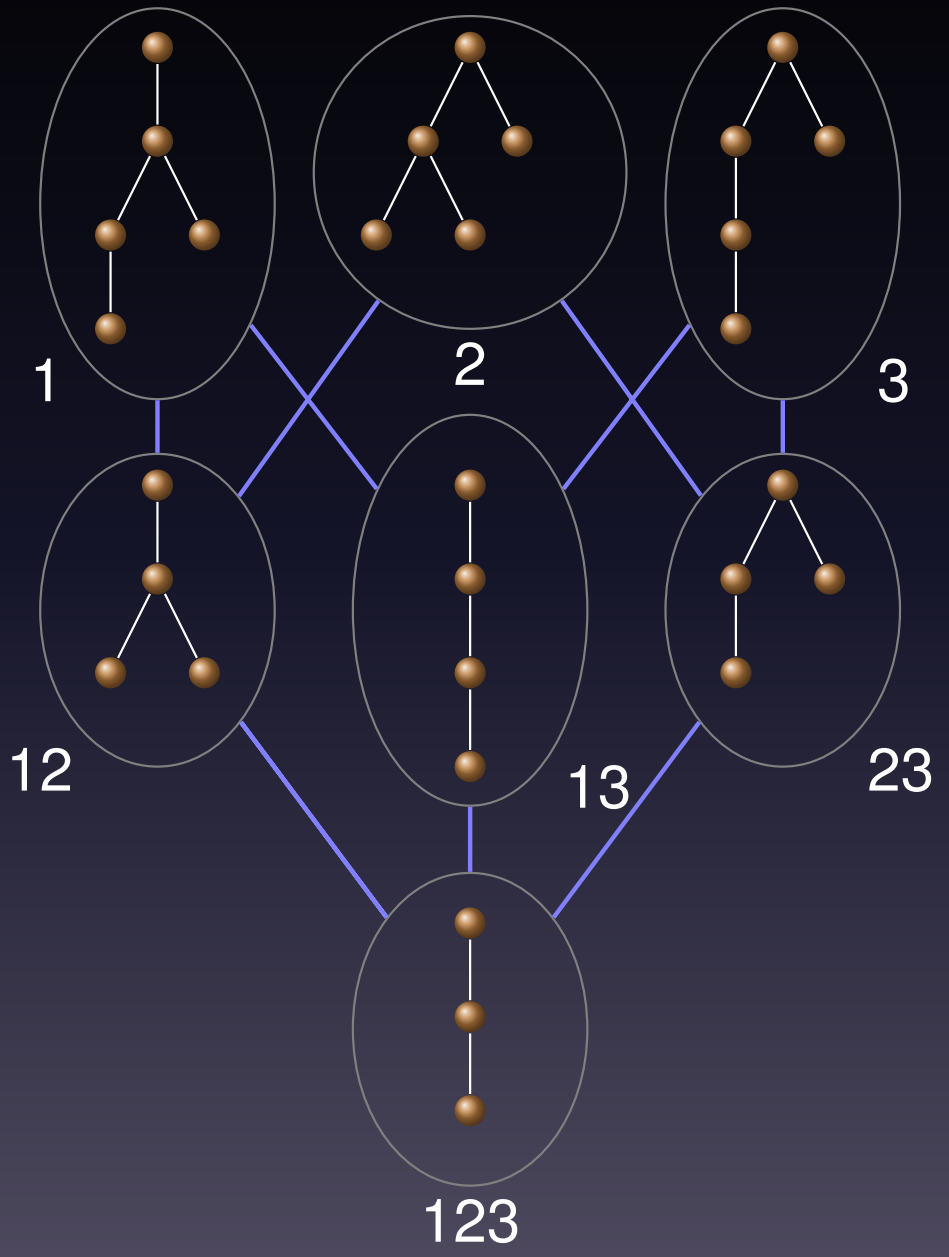


Mining Implications from Lattices of Closed Trees

Set of Rules:

$$A \rightarrow \Gamma_{\mathscr{D}}(A).$$

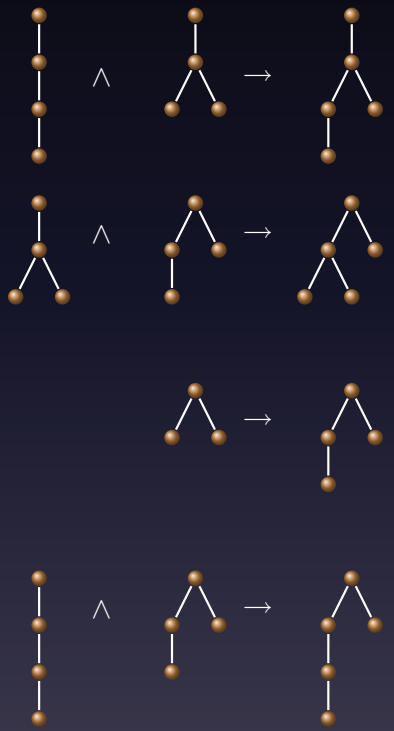
- antecedents are obtained through a computation akin to a hypergraph transversal
- consequents follow from an application of the closure operators

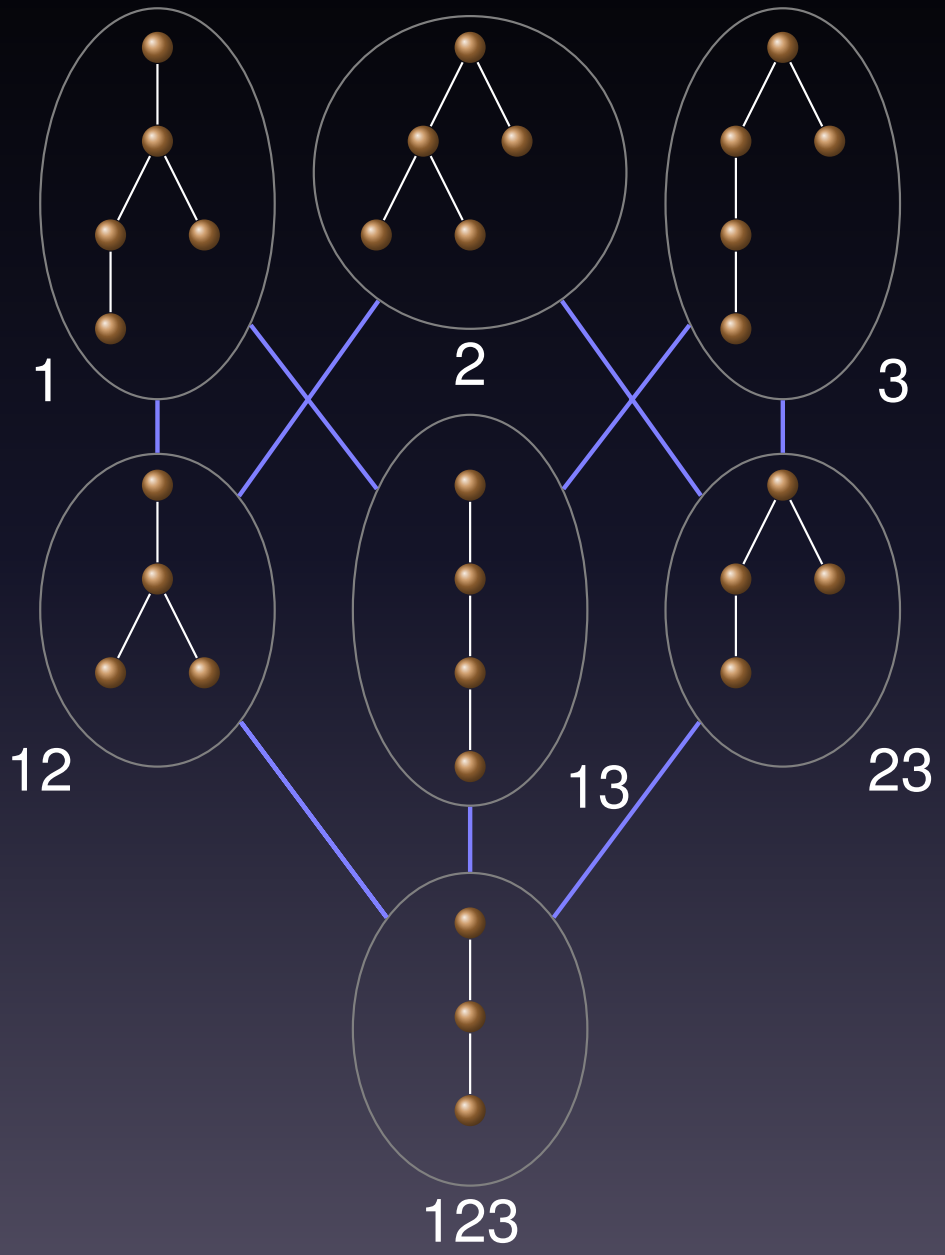


Mining Implications from Lattices of Closed Trees

Set of Rules:

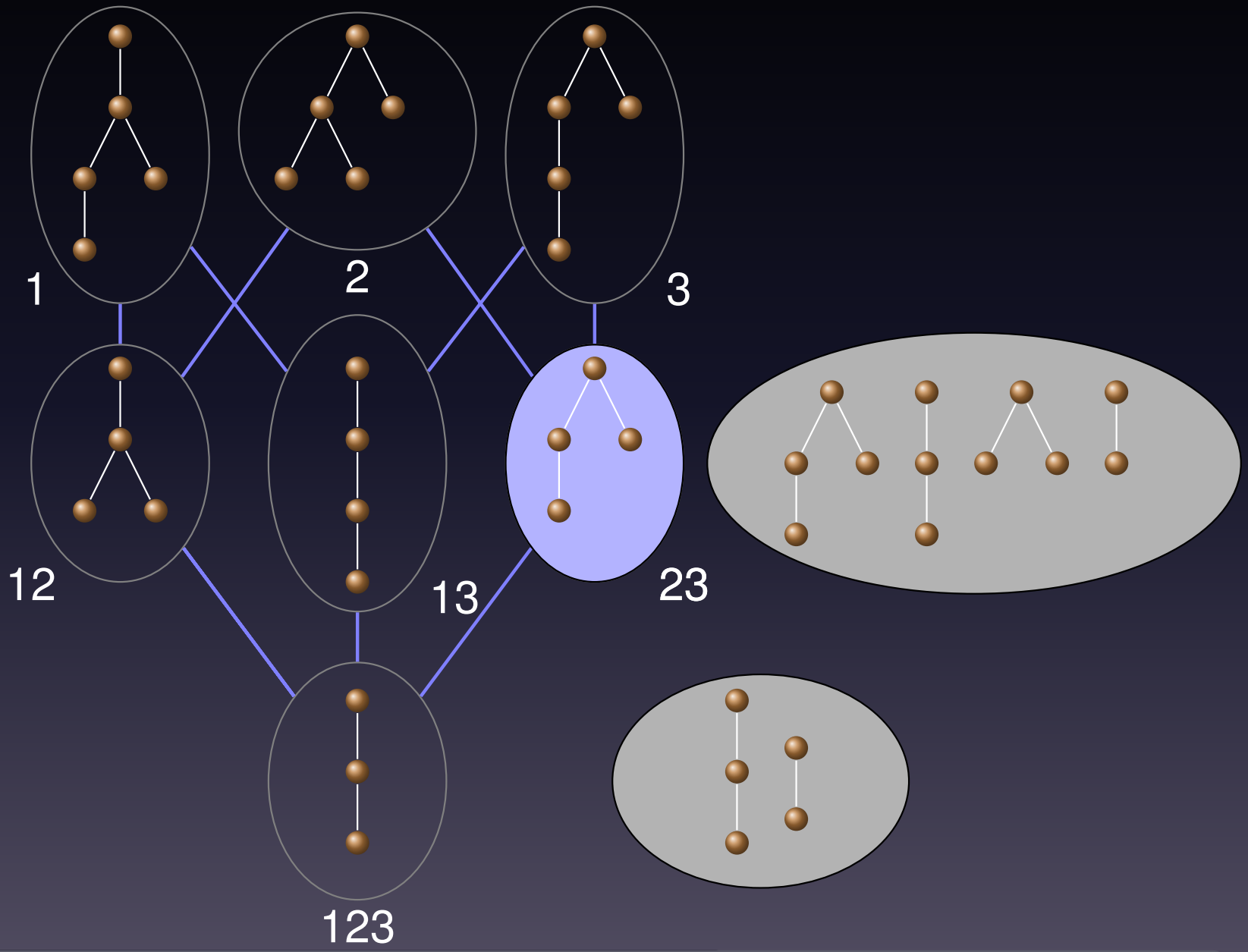
$$A \longrightarrow \Gamma_{\mathscr{D}}(A).$$





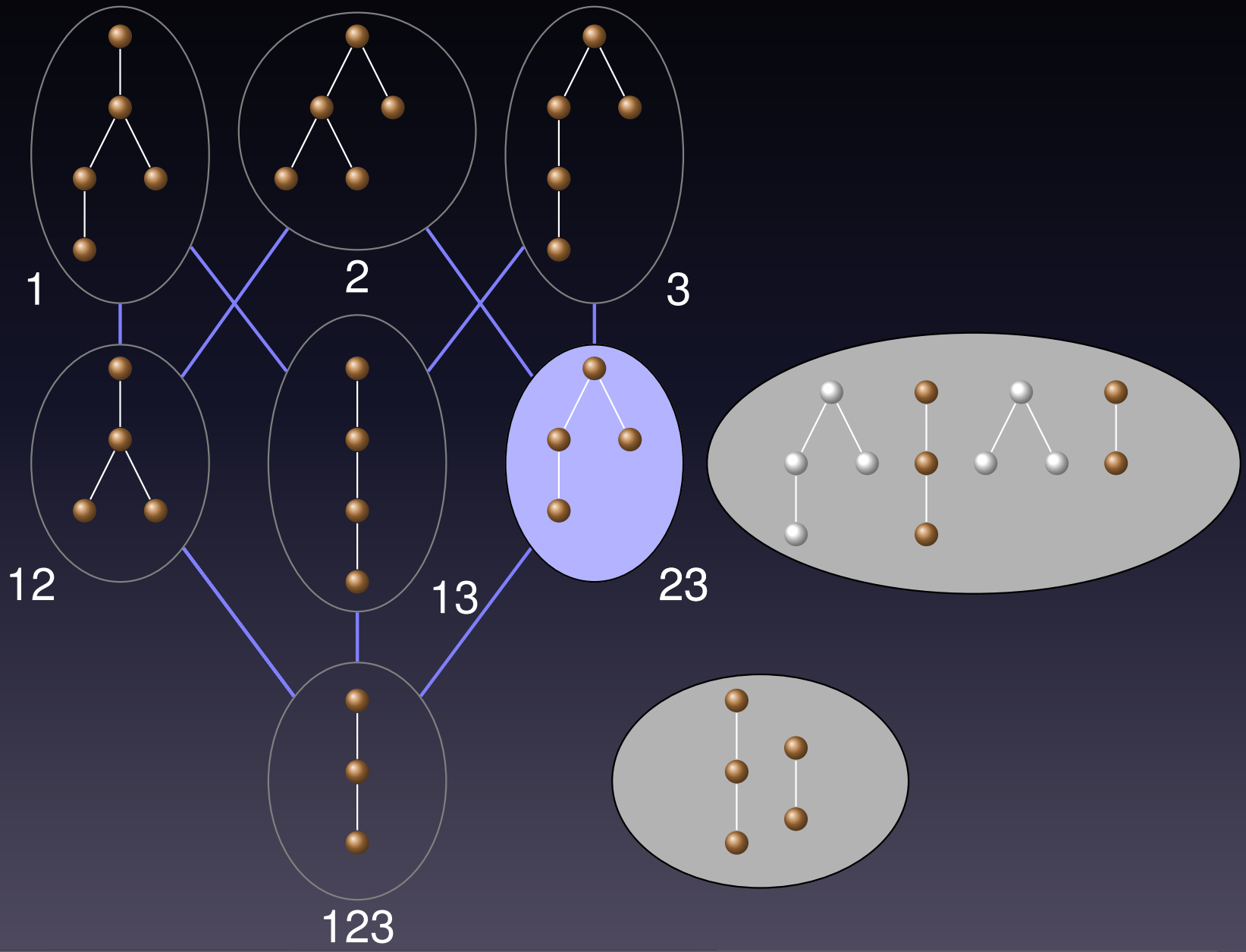
Association Rule Computation Example

(8) Association Rules



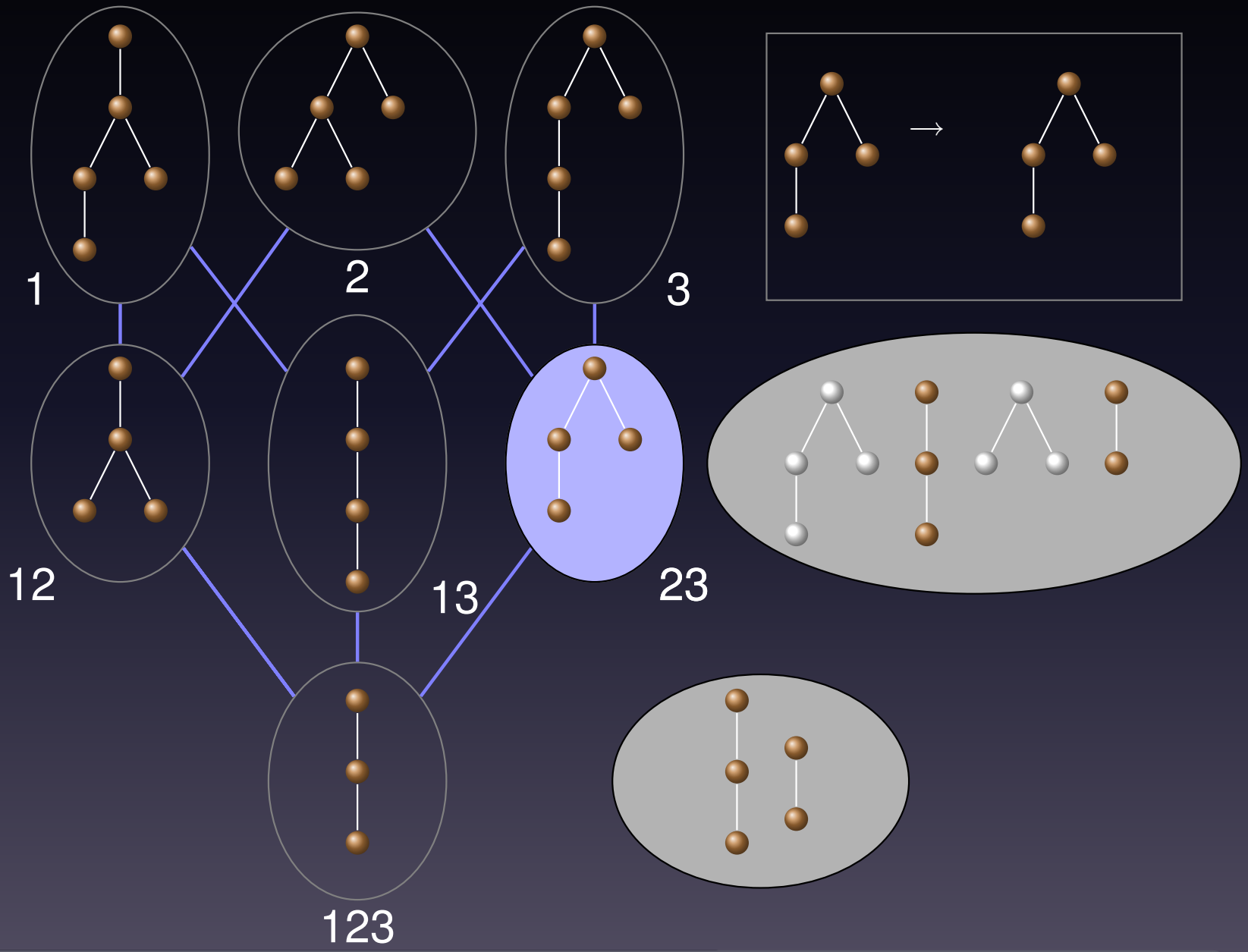
Association Rule Computation Example

(8) Association Rules



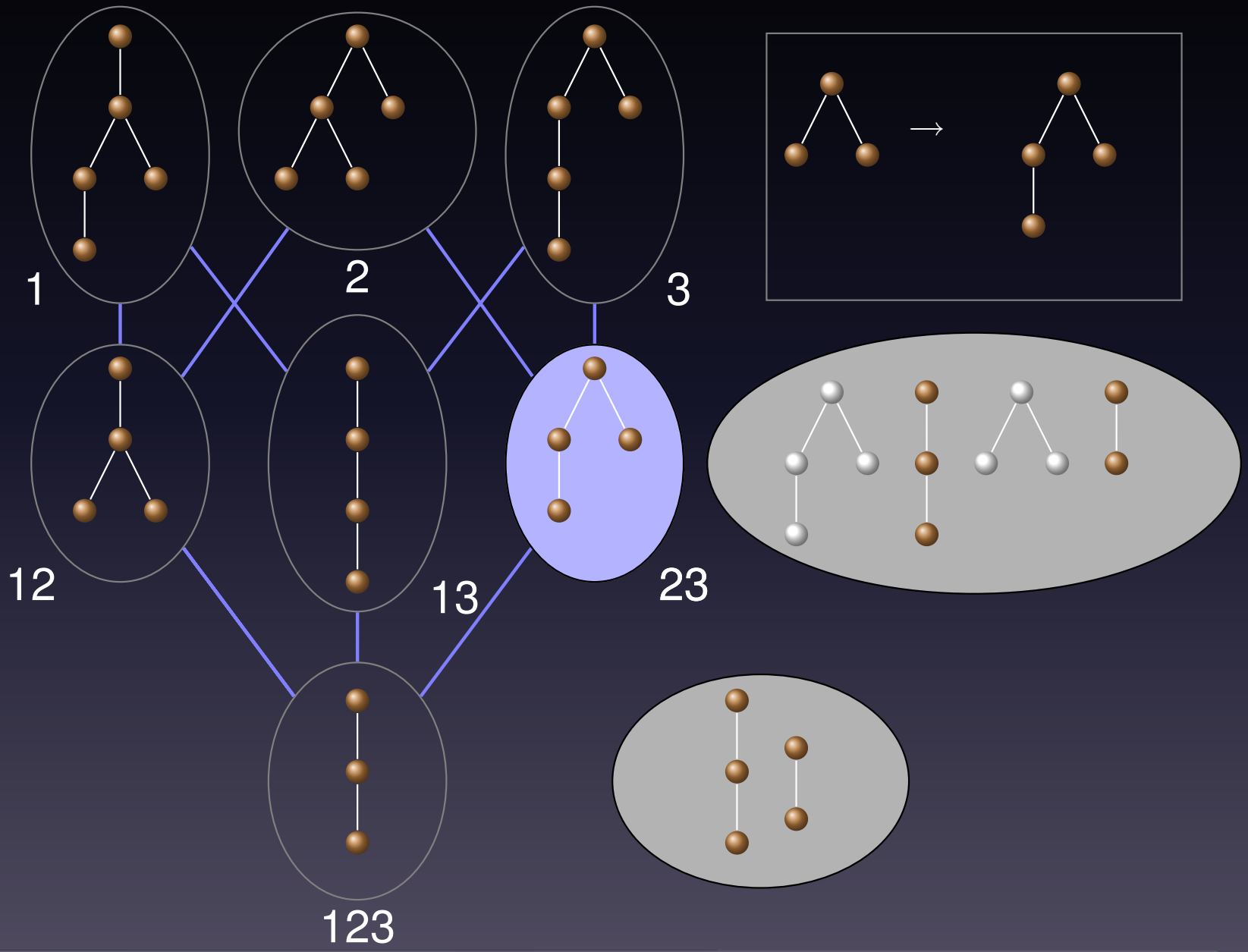
Association Rule Computation Example

(8) Association Rules



Association Rule Computation Example

(8) Association Rules



Model transformation

(8) Association Rules

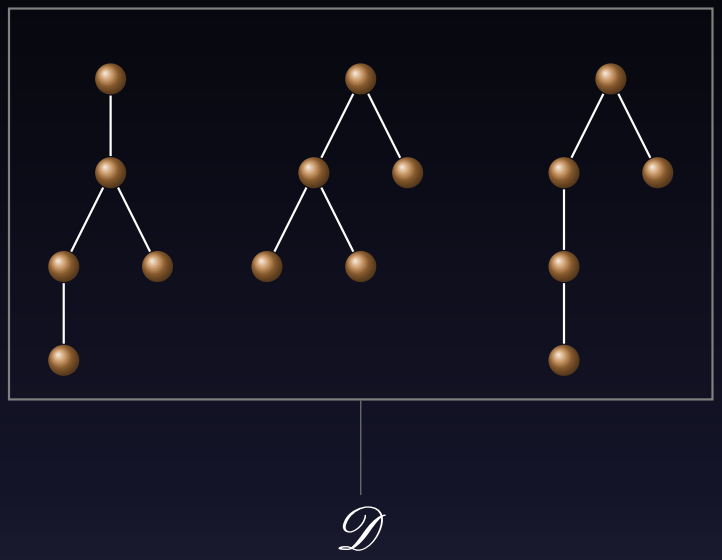
Intuition

- One propositional variable v_t is assigned to each possible subtree t.
- A set of trees A corresponds in a natural way to a model m_A.
- Let m_A be a model: we impose on m_A the constraints that if $m_A(v_t) = 1$ for a variable v_t, then $m_A(v_{t'}) = 1$ for all those variables $v_{t'}$ such that $v_{t'}$ represents a subtree of the tree represented by v_t.

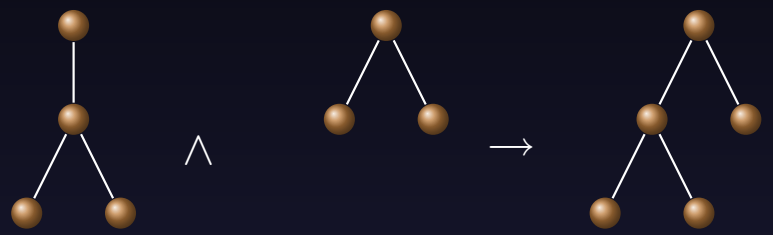
$$\mathscr{R}_0 = \{ v_{t'} \rightarrow v_t \mid t' \preceq t, t \in \mathscr{U}, t' \in \mathscr{U} \}$$

Implicit Rules Definition

(9) Implicit Rules



Implicit Rule



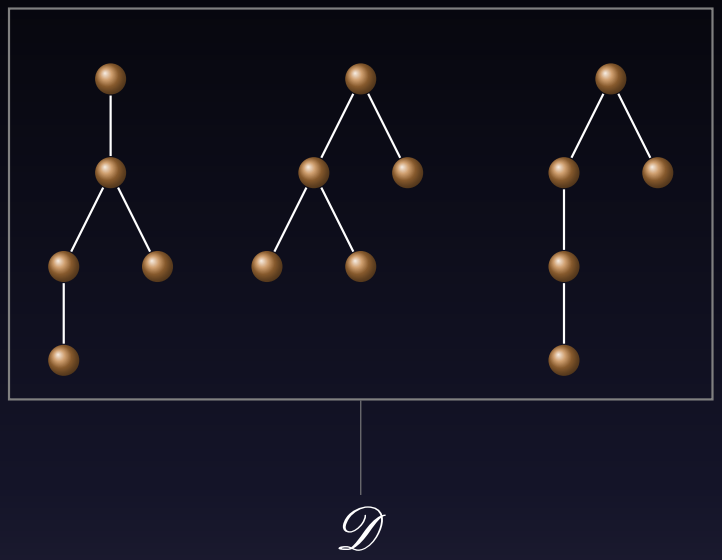
Given three trees t_1, t_2, t_3, we say that $t_1 \wedge t_2 \longrightarrow t_3$, is an *implicit Horn rule* (abbreviately, an *implicit rule*) if for every tree t it holds

$$t_1 \preceq t \ \wedge \ t_2 \preceq t \ \leftrightarrow \ t_3 \preceq t.$$

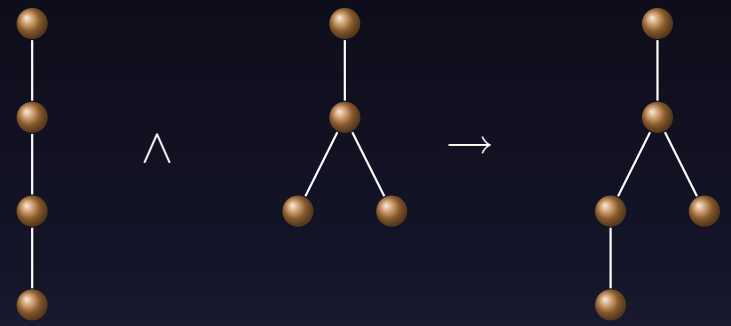
t_1 and t_2 have implicit rules if $t_1 \wedge t_2 \longrightarrow t$ is an implicit rule for some t.

Implicit Rules Definition

(9) Implicit Rules



NOT Implicit Rule



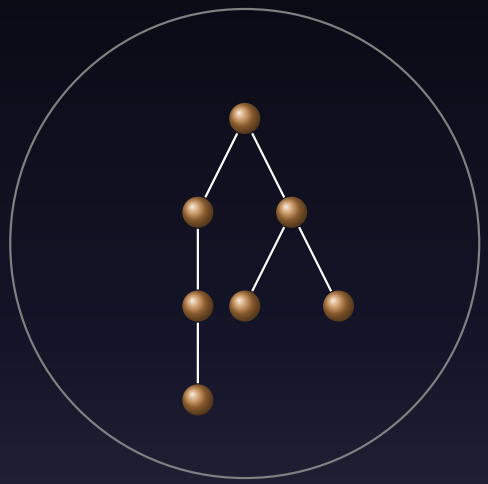
Given three trees t_1, t_2, t_3, we say that $t_1 \wedge t_2 \rightarrow t_3$, is an *implicit Horn rule* (abbreviately, an *implicit rule*) if for every tree t it holds

$$t_1 \preceq t \ \wedge \ t_2 \preceq t \ \leftrightarrow \ t_3 \preceq t.$$

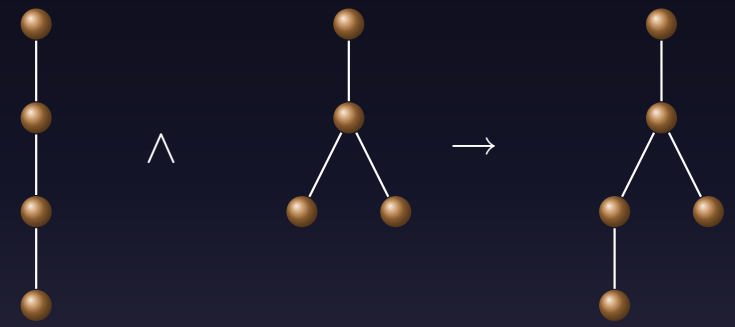
t_1 and t_2 have implicit rules if $t_1 \wedge t_2 \rightarrow t$ is an implicit rule for some t.

Implicit Rules Definition

(9) Implicit Rules



NOT Implicit Rule



This supertree of the antecedents is **NOT** a supertree of the consequents.

Implicit Rules Characterization

(9) Implicit Rules

Theorem

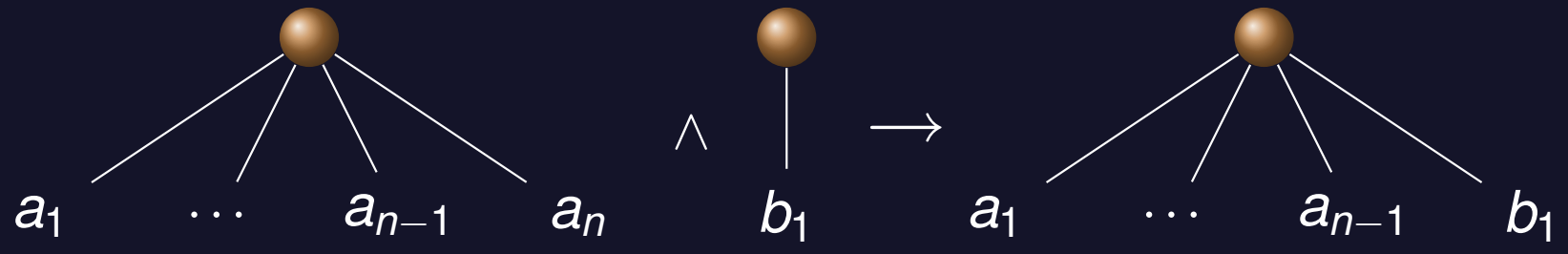
All trees a, b such that $a \preceq b$ have implicit rules.

Theorem

Suppose that b has only one component. Then they have implicit rules if and only if a has a maximum component which is a subtree of the component of b.

- for all $i < n$

$$a_i \preceq a_n \preceq b_1$$



Main contributions (ii)

Tree Mining

6. Closure Operator on Trees
7. Unlabeled Closed Frequent Tree Mining
8. A way of extracting high-confidence association rules from datasets consisting of unlabeled trees
   - antecedents are obtained through a computation akin to a hypergraph transversal
   - consequents follow from an application of the closure operators
9. Detection of some cases of **implicit rules**: rules that always hold, independently of the dataset

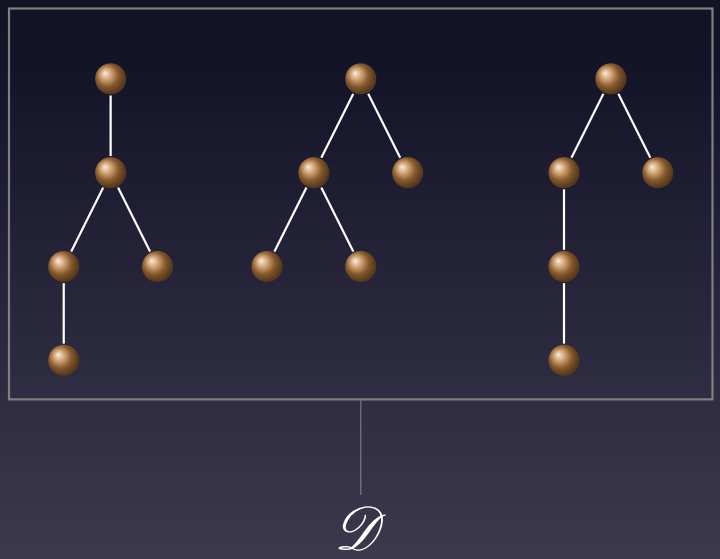
Outline

Mining Evolving Tree Data Streams

(10,11,12) Incremental, Sliding Window, and Adaptive Tree Mining Methods

Problem

Given a data stream $\mathscr{D}$ of rooted, unlabeled and unordered trees, find frequent closed trees.



We provide three algorithms, of increasing power

- Incremental
- Sliding Window
- Adaptive

Relaxed Support

(13) Logarithmic Relaxed Support

Guojie Song, Dongqing Yang, Bin Cui, Baihua Zheng, Yunfeng Liu and Kunqing Xie.
CLAIM: An Efficient Method for Relaxed Frequent Closed Itemsets Mining over Stream Data

- **Linear Relaxed Interval**:The support space of all subpatterns can be divided into $n = \lceil 1/\varepsilon_r \rceil$ intervals, where ε_r is a user-specified relaxed factor, and each interval can be denoted by $\mathscr{I}_i = [l_i, u_i)$, where $l_i = (n-i) * \varepsilon_r \geq 0$, $u_i = (n-i+1) * \varepsilon_r \leq 1$ and $i \leq n$.
- **Linear Relaxed closed subpattern** t: if and only if there exists no proper superpattern t' of t such that their suports belong to the same interval $\mathscr{I}_i$.

Relaxed Support

(13) Logarithmic Relaxed Support

As the number of closed frequent patterns is not linear with respect support, we introduce a new relaxed support:

- **Logarithmic Relaxed Interval**:The support space of all subpatterns can be divided into $n = \lceil 1/\varepsilon_r \rceil$ intervals, where ε_r is a user-specified relaxed factor, and each interval can be denoted by $\mathscr{I}_i = [l_i, u_i)$, where $l_i = \lceil c^i \rceil$, $u_i = \lceil c^{i+1} - 1 \rceil$ and $i \leq n$.
- **Logarithmic Relaxed closed subpattern** t: if and only if there exists no proper superpattern t' of t such that their suports belong to the same interval $\mathscr{I}_i$.

Algorithms

(10,11,12) Incremental, Sliding Window, and Adaptive Tree Mining Methods

Algorithms

- Incremental: IncTreeNat
- Sliding Window: WinTreeNat
- Adaptive: AdaTreeNat Uses ADWIN to monitor change

ADWIN

An adaptive sliding window whose size is recomputed online according to the rate of change observed.

ADWIN has rigorous guarantees (theorems)

- On ratio of false positives and false negatives
- On the relation of the size of the current window and change rates

Experimental Validation: TN1

(10,11,12) Incremental, Sliding Window, and Adaptive Tree Mining Methods

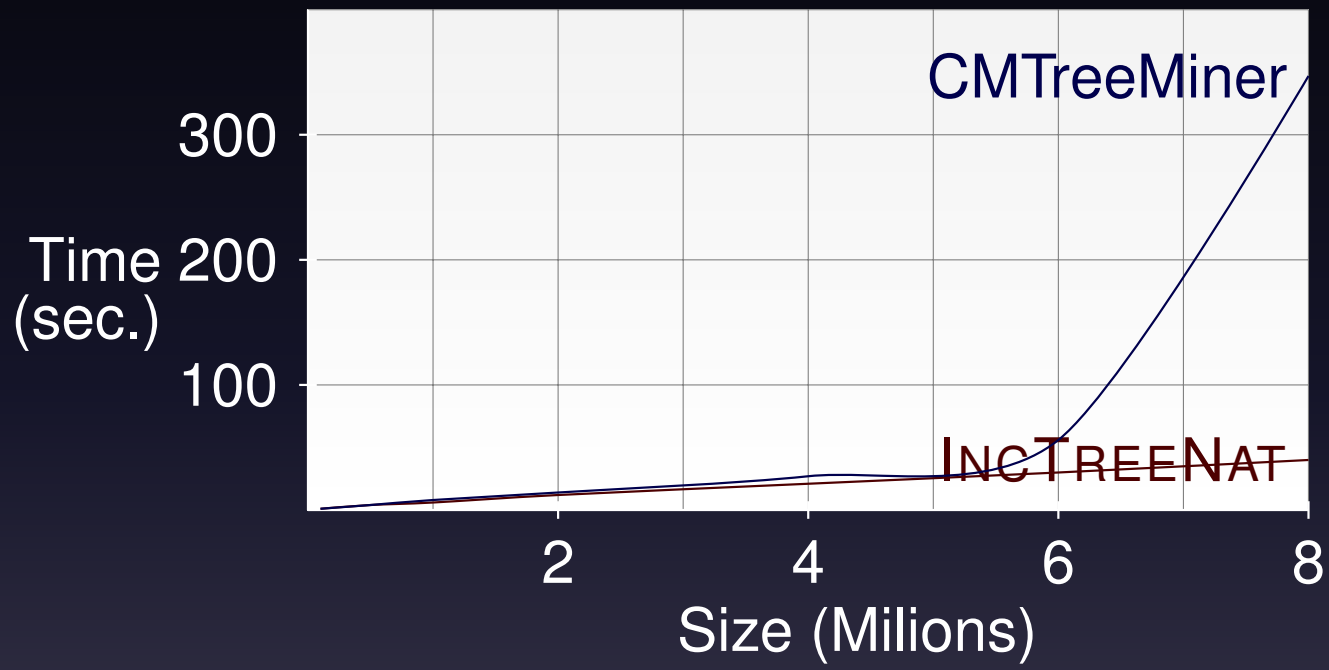


Figure: Experiments on ordered trees with TN1 dataset

Adaptive XML Tree Classification on evolving data streams

(14) XML Tree Classification

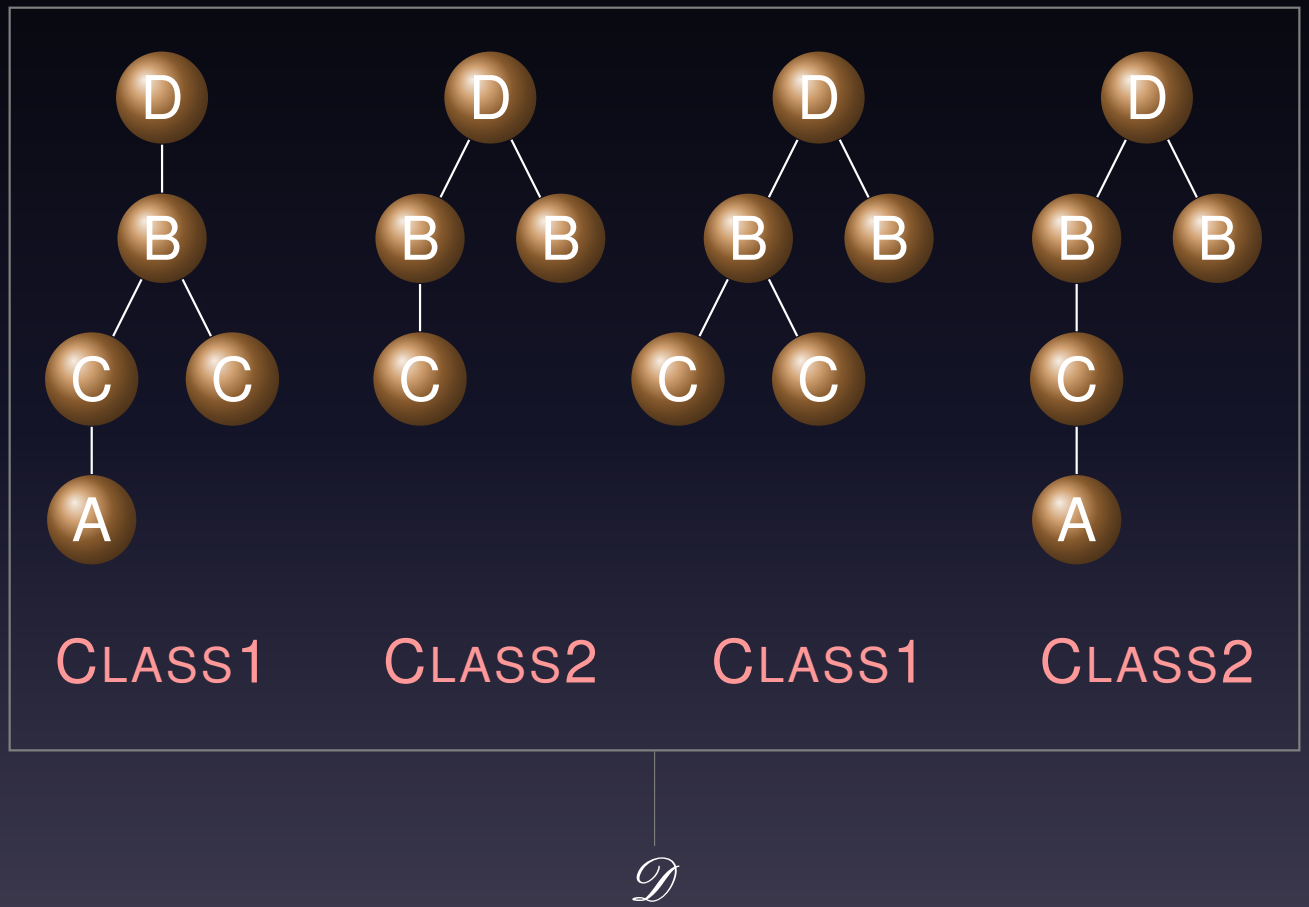
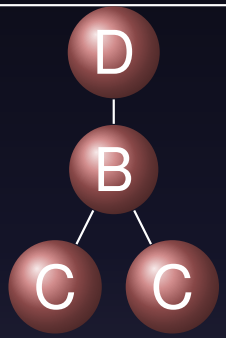
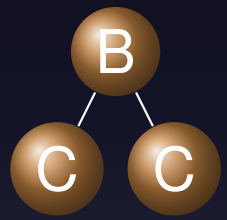
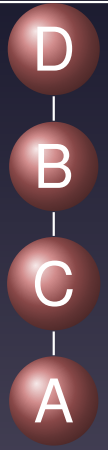
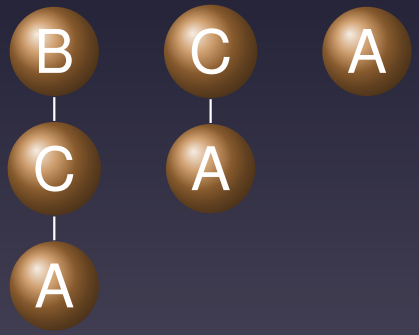


Figure: A dataset example

Adaptive XML Tree Classification on evolving data streams

(14) XML Tree Classification

	Closed	Freq. not Closed Trees	Tree Trans. 1	2	3	4
c_1			1	0	1	0
c_2			1	0	0	1

Adaptive XML Tree Classification on evolving data streams

(14) XML Tree Classification

Id	Frequent Trees													
	c_1		c_2				c_3		c_4					
	c_1	f_1^1	c_2	f_2^1	f_2^2	f_2^3	c_3	f_3^1	c_4	f_4^1	f_4^2	f_4^3	f_4^4	f_4^5
1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
2	0	0	0	0	0	0	1	1	1	1	1	1	1	1
3	1	1	0	0	0	0	1	1	1	1	1	1	1	1
4	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	Closed Trees				Maximal Trees			
Id Tree	c_1	c_2	c_3	c_4	c_1	c_2	c_3	**Class**
1	1	1	0	1	1	1	0	CLASS1
2	0	0	1	1	0	0	1	CLASS2
3	1	0	1	1	1	0	1	CLASS1
4	0	1	1	1	0	1	1	CLASS2

Adaptive XML Tree Framework on evolving data streams

(14) XML Tree Classification

XML Tree Classification Framework Components

- An XML closed frequent tree miner
- A Data stream classifier algorithm, which we will feed with tuples to be classified online.

Adaptive XML Tree Framework on evolving data streams

(14) XML Tree Classification

		Maximal			Closed		
	# Trees	Att.	Acc.	Mem.	Att.	Acc.	Mem.
CSLOG12	15483	84	**79.64**	1.2	228	78.12	2.54
CSLOG23	15037	88	**79.81**	1.21	243	78.77	2.75
CSLOG31	15702	86	**79.94**	1.25	243	77.60	2.73
CSLOG123	23111	84	**80.02**	1.7	228	78.91	4.18

Table: BAGGING on unordered trees.

Main contributions (iii)

Mining Evolving Tree Data Streams

10. Incremental Method
11. Sliding Window Method
12. Adaptive Method
13. Logarithmic Relaxed Support
14. XML Classification

Outline

Main contributions

Mining Evolving Data Streams

1. Framework
2. ADWIN
3. Classifiers
4. MOA
5. ASHT

Tree Mining

6. Closure Operator on Trees
7. Unlabeled Tree Mining Methods
8. Deterministic Association Rules
9. Implicit Rules

Mining Evolving Tree Data Streams

10. Incremental Method
11. Sliding Window Method
12. Adaptive Method
13. Logarithmic Relaxed Support
14. XML Classification

Future Lines (i)

- Adaptive Kalman Filter
  - Kalman filter adaptive computing Q and R without using the size of the window of ADWIN.
- Extend MOA framework
  - Support vector machines
  - Clustering
  - Itemset mining
  - Association rules

Future Lines (ii)

- Adaptive Deterministic Association Rules
  - Deterministic Association Rules computed on evolving data streams
- General Implicit Rules Characterization
  - Find a characterization of implicit rules with any number of components
- Not Deterministic Association Rules
  - Find basis of association rules for trees with confidence lower than 100%

Future Lines (iii)

- Closed Frequent Graph Mining
  - Mining methods to obtain closed frequent graphs.
    - Not incremental
    - Incremental
    - Sliding Window
    - Adaptive
- Graph Classification
  - Classifiers of graphs using maximal and closed frequent subgraphs.

Relevant publications


Albert Bifet and Ricard Gavaldà.
Kalman filters and adaptive windows for learning in data streams. DS'06


Albert Bifet and Ricard Gavaldà.
Learning from time-changing data with adaptive windowing. SDM'07


Albert Bifet and Ricard Gavaldà.
Adaptive parameter-free learning from evolving data streams. Tech-Rep R09-9


A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà.
New ensemble methods for evolving data streams. KDD'09


José L. Balcázar, Albert Bifet, and Antoni Lozano.
Mining frequent closed unordered trees through natural representations. ICCS'07


José L. Balcázar, Albert Bifet, and Antoni Lozano.
Subtree testing and closed tree mining through natural representations. DEXA'07


José L. Balcázar, Albert Bifet, and Antoni Lozano.
Mining implications from lattices of closed trees. EGC'2008


José L. Balcázar, Albert Bifet, and Antoni Lozano.
Mining Frequent Closed Rooted Trees. MLJ'09


Albert Bifet and Ricard Gavaldà.
Mining adaptively frequent closed unlabeled rooted trees in data streams. KDD'08


Albert Bifet and Ricard Gavaldà.
Adaptive XML Tree Classification on evolving data streams