SCIENCE

Data Stream Mining

School of Computer Science

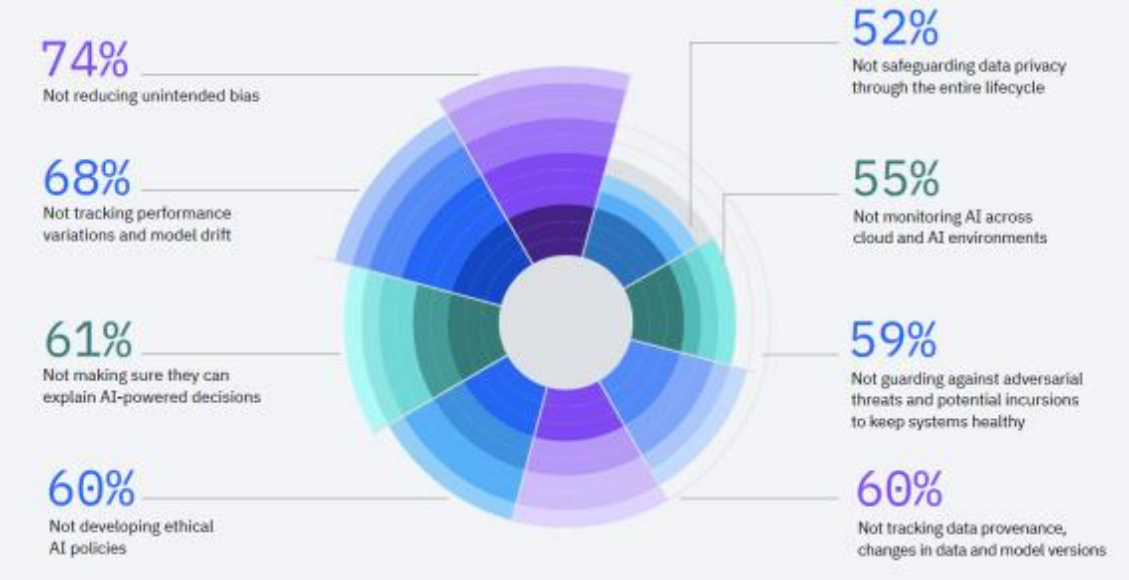
Outline

- An introduction to data stream mining
  - What is a data stream?
  - Data stream characteristics
  - Concept drift
- Learning algorithms for data streams
  - Main classifier types for drifting data streams
  - Ensemble learning from drifting data streams
- Limited access to ground truth in data streams
  - How to cope with sparse class labels?
  - Active learning under concept drift
  - Semi-supervised learning from drifting data streams
- Advanced problems
  - Recurrent Concepts
  - Open challenges and future directions

New challenges for machine learning

Standard static and relatively small scenarios in machine learning and data mining do not reflect the current real-life problems we are facing.

We must deal with new data sources, generating high-speed, massive and heterogeneous data.



IBM Global AI Adoption Index 2022

What is a data stream?

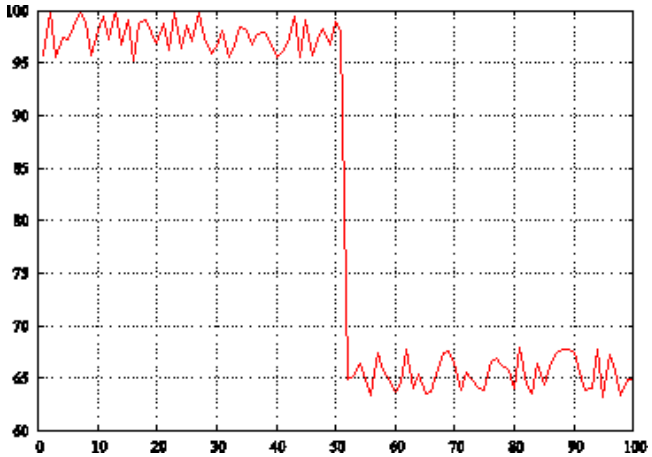
Data stream: an ordered, potentially unbounded sequence of instances which arrive continuously with time-varying intensity.

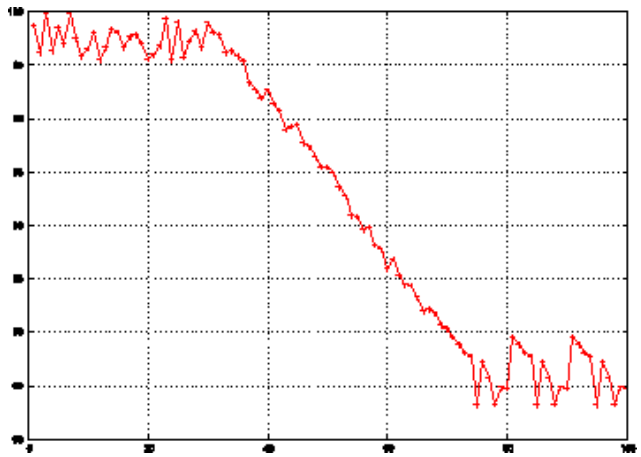
Given a model $\hat{f}_{t-1} : X \longrightarrow Y$ and a new chunk of examples $S_t = \{(\mathbf{x}_t^{(1)}, y_t^{(1)}),(\mathbf{x}_t^{(2)}, y_t^{(2)}),\ldots,(\mathbf{x}_t^{(n)}, y_t^{(n)})\} \subset S$, where $\forall i$, $(\mathbf{x}_t^{(i)}, y_t^{(i)}) \sim_{\text{i.i.d.}} \mathcal{D}_t$.

High-speed data streams: arising demands for fast-changing and continuously arriving data to be analyzed in real time.

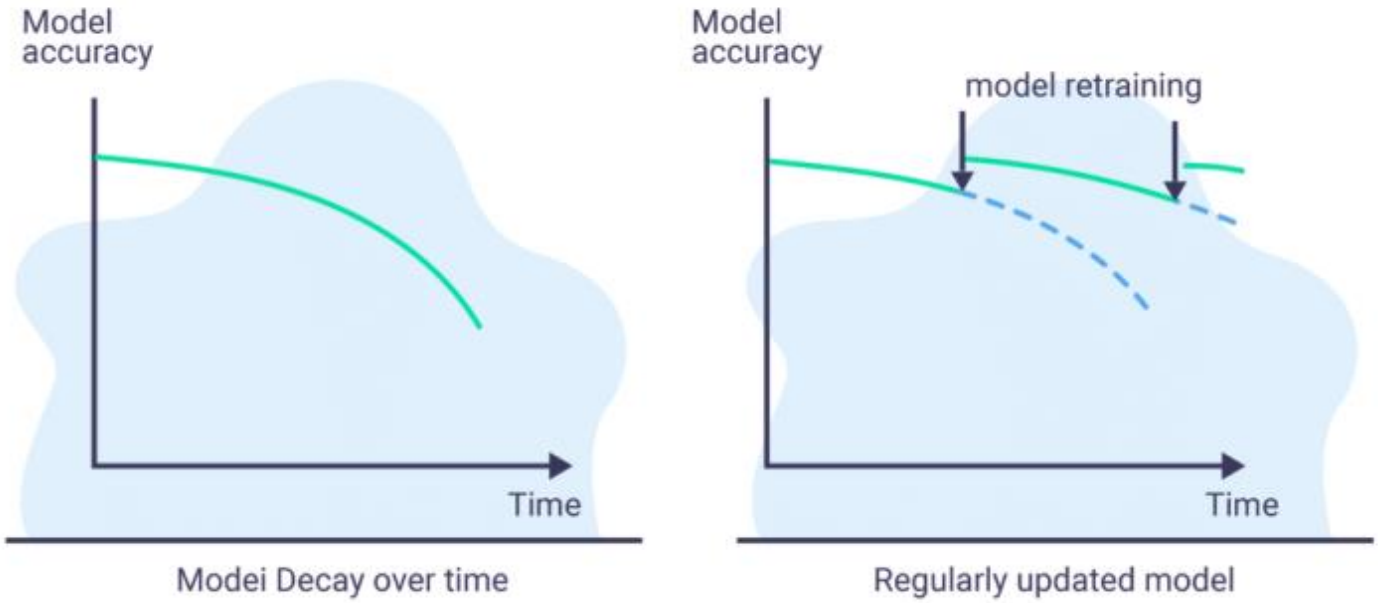
Requirements for data stream algorithms

- Incremental processing
- Limited time:
  - Examples arrive rapidly
  - Each example can be processed only once
- Limited memory:
  - Streams are often too large to be processed as a whole
- Changes in data characteristics:
  - Data streams can evolve over time





Model decay



Concept drift is one of the main reasons why we need to continue learning and adapting over time.

Evaluating data stream algorithms

- Block / batch processing (data chunks)

- Online processing (instance after instance)

Evaluating data stream algorithms

Standard metrics like accuracy, G-mean, Kappa etc. were designed for static problems.

One should use prequential metrics with forgetting, computed over most recent examples.

Prequential accuracy for standard problems and prequential AUC for binary and imbalanced streams.

$$acc^{(t)} = \begin{cases} acc_{ex}^{(t)}, & \text{if } t = 1 \\ \frac{(t-1)\cdot acc^{(t-1)} + acc_{ex}^{(t)}}{t}, & \text{otherwise} \end{cases}$$

where $acc_{ex}^{(t)}$ is the error (0 or 1) on the current training example ex before its learning.

Additional metrics are crucial for evaluating streaming classifiers:

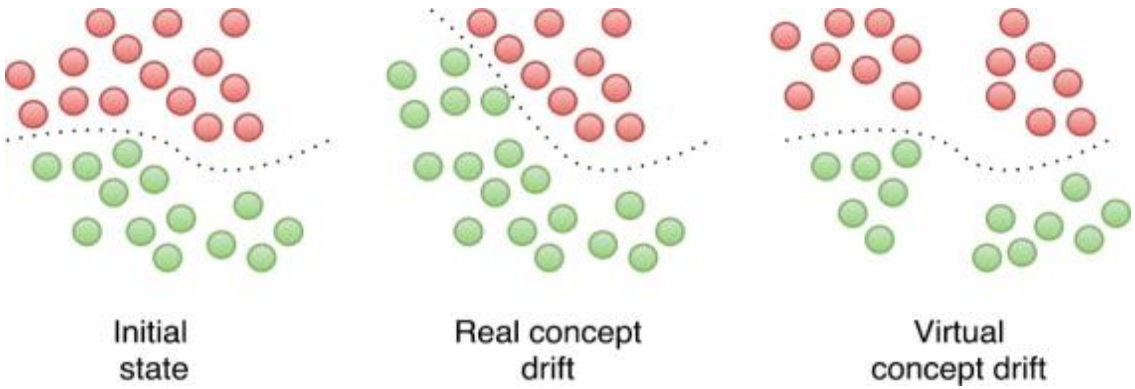
- Memory consumption
- Update time
- Classification

Interleaved Test-Then-Train or Prequential: Each individual example can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated. When intentionally performed in this order, the model is always being tested on examples it has not seen.

Concept drift

Concept drift can be defined as changes in distributions and definition of learned concepts over time.

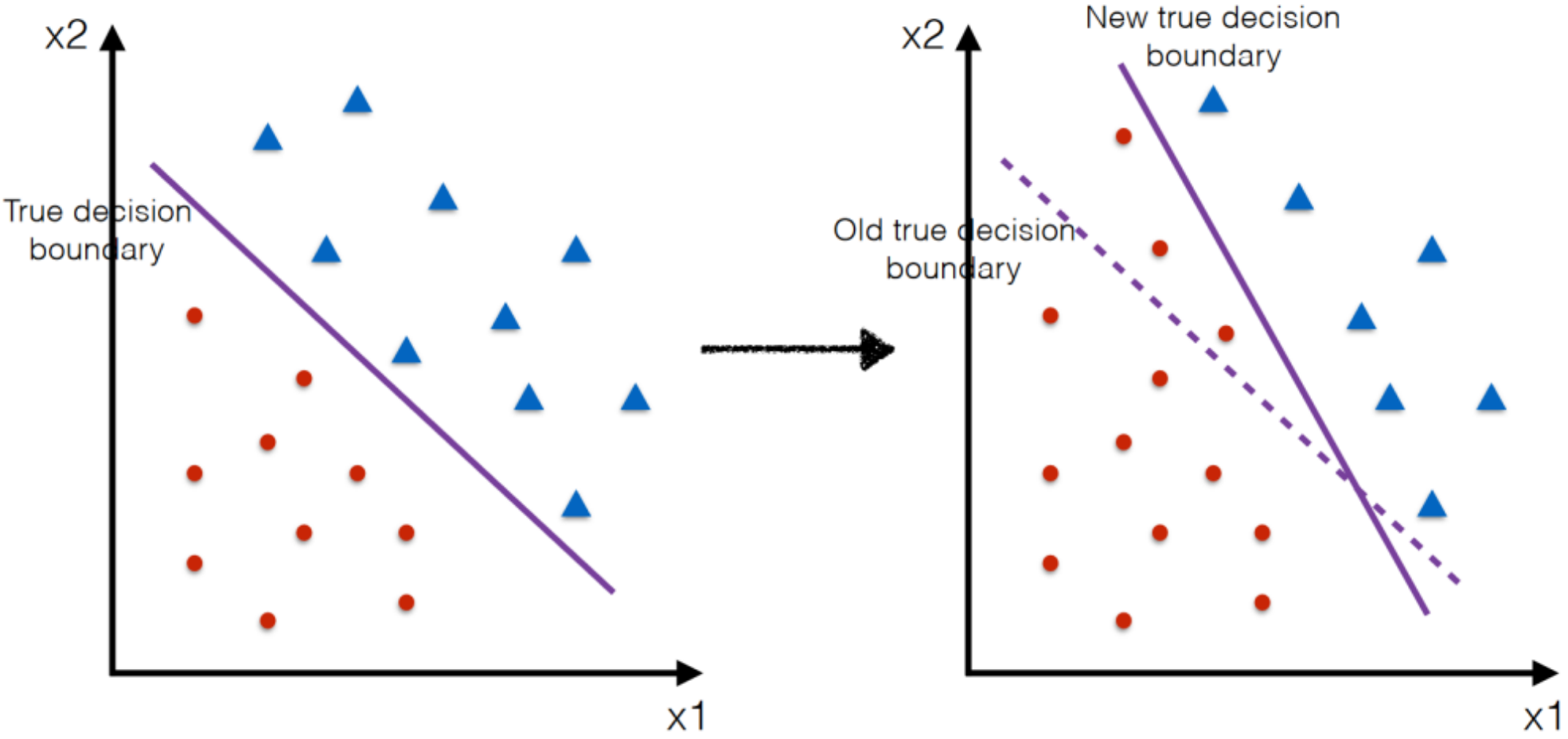
> Concept drift: change in the joint probability distribution of the problem. We say that there is concept drift in a data stream if, $\exists t, t+1 \mid \mathcal{D}_t \neq \mathcal{D}_{t+1}$.



Some real-life examples:

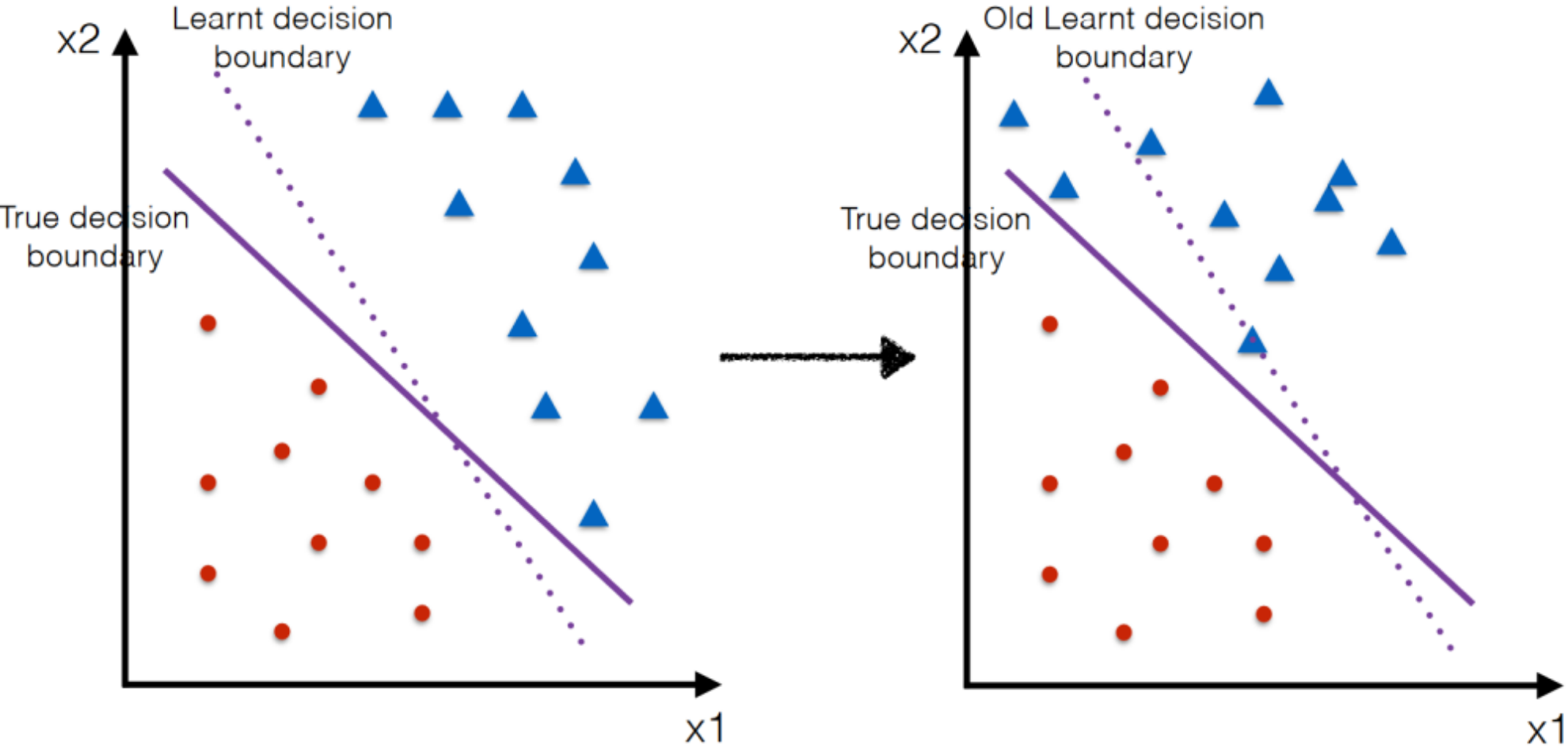
- changes of the user's interest in following news
- evolution of language used in text messages
- degradation or damage in networks of sensors

Change in p(y|x)

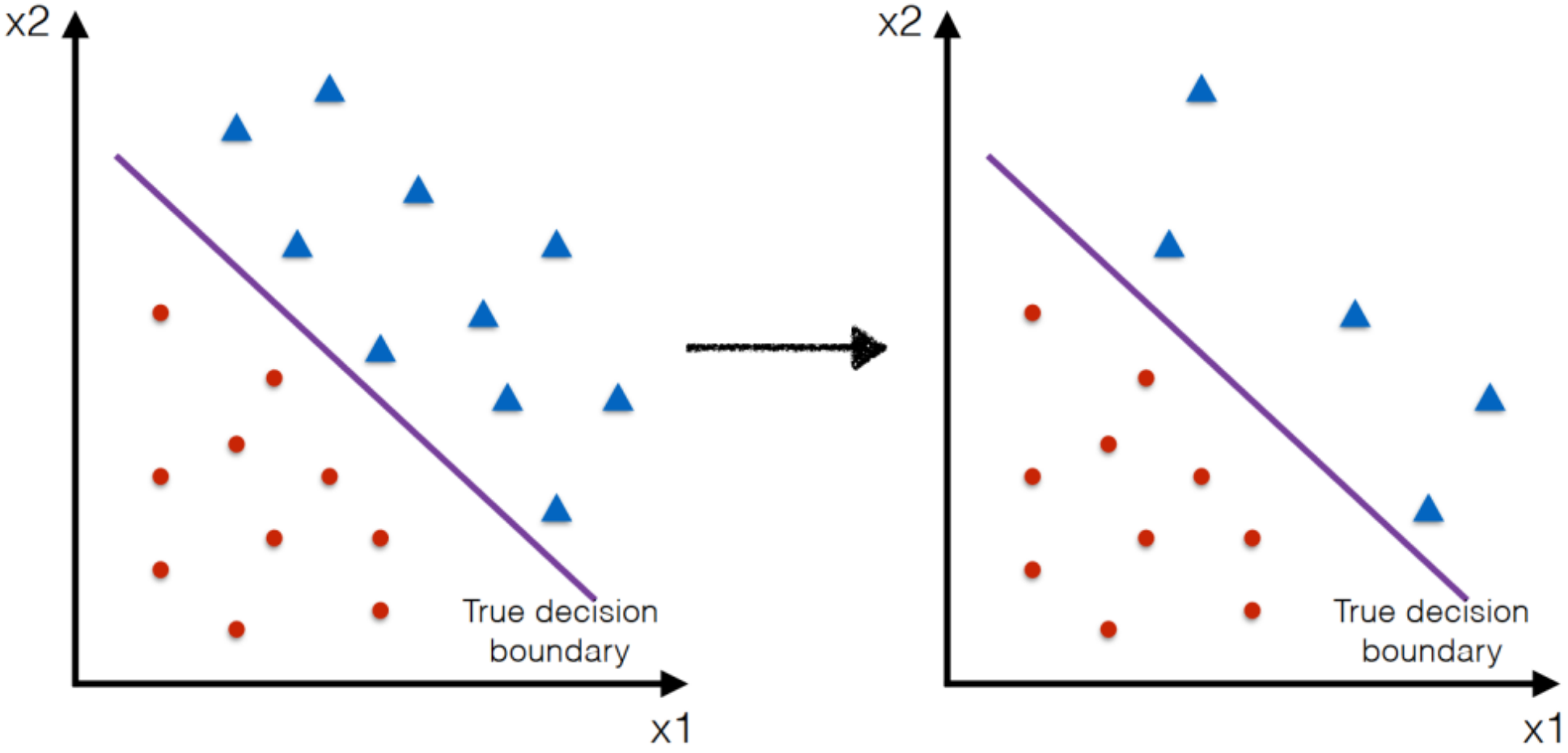


MINKU, L.; WANG, S.; BORACCHI, G. . "Learning Class Imbalanced Data Streams", *World Congress on Computational Intelligence (WCCI)*, July 2018

Change in ?


x2
Learnt decision boundary
True decision boundary
x1
x2
Old Learnt decision boundary
True decision boundary
x1

Change in ?



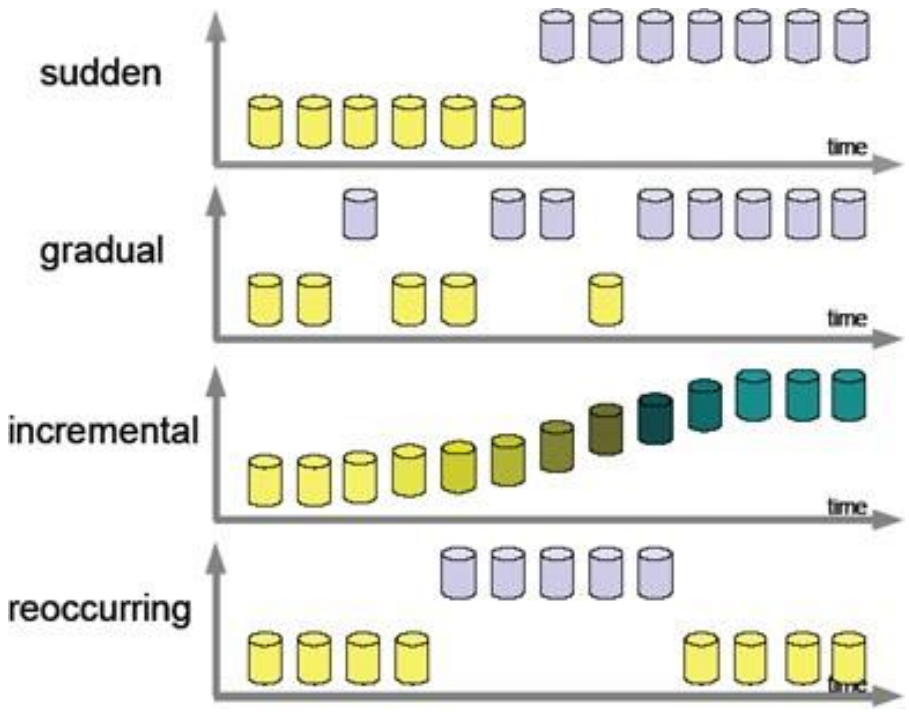
Concept drift

Let us assume that our stream consist of a set of states $S = \{S_1, S_2, \cdot \cdot \cdot, S_n\}$, where S_i is generated by a stationary distribution D_i .

By a stationary stream we can consider a transition $S_j \rightarrow S_{j+1}$, where $D_j = D_{j+1}$.

A non-stationary stream may have one or more of the following concept drift types:

- Sudden, where S_j is suddenly replaced by S_{j+1} and $D_j \neq D_{j+1}$
- Gradual, considered as a transition phase where examples in S_{j+1} are generated by a mixture of D_j and D_{j+1}
- Incremental, where rate of changes is much slower and $D_j \cap D_{j+1} \neq \varnothing$
- Reoccurring, where a concept from k-th previous iteration may reappear: $D_{j+1} = D_{j-k}$



One must not confuse concept drift with data noise.

Concept drift

We may also categorize concept drift according to its in uence on the probabilistic characteristics of the classification task:

- Virtual concept drift - changes do not impact the decision boundaries (posterior probabilities), but affect the conditional probability density functions
- Real concept drift - changes affect the decision boundaries (posterior probabilities) and may impact unconditional probability density function

Handling concept drift

Three possible approaches to tackling drifting data streams:

- Rebuilding a classification model whenever new data becomes available (expensive, time-consuming, even impossible for rapidly evolving streams!)
- Detecting concept changes in new data (and rebuilding a classifier if these changes are sufficiently significant)
- Using an adaptive classifier (i.e. working in incremental or online mode)

We will discuss recurrent concept later!

Handling concept drift

Algorithms for efficiently handling of concept drift presence can be categorized into four groups:

- Concept drift detectors
- Sliding window solutions
- Online learners
- Ensemble learners

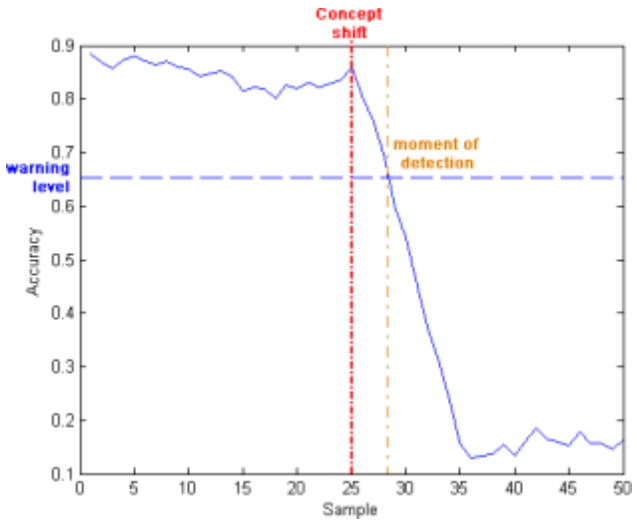
Drift detectors

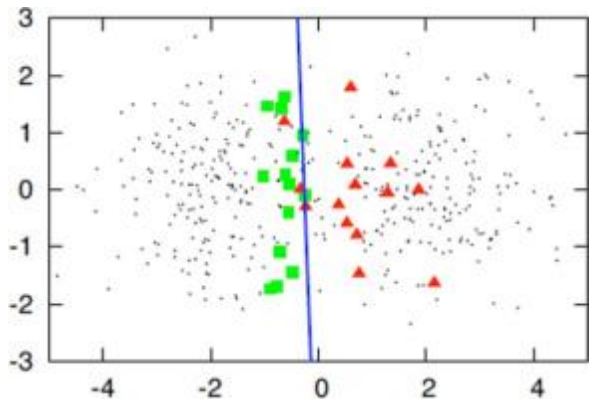
Algorithms that address the question of when drift occurs, being usually a separate tool from the actual classifier.

They aim at rising a signal when the change occurs. Some models also raise alarm when the chance of drift increases.

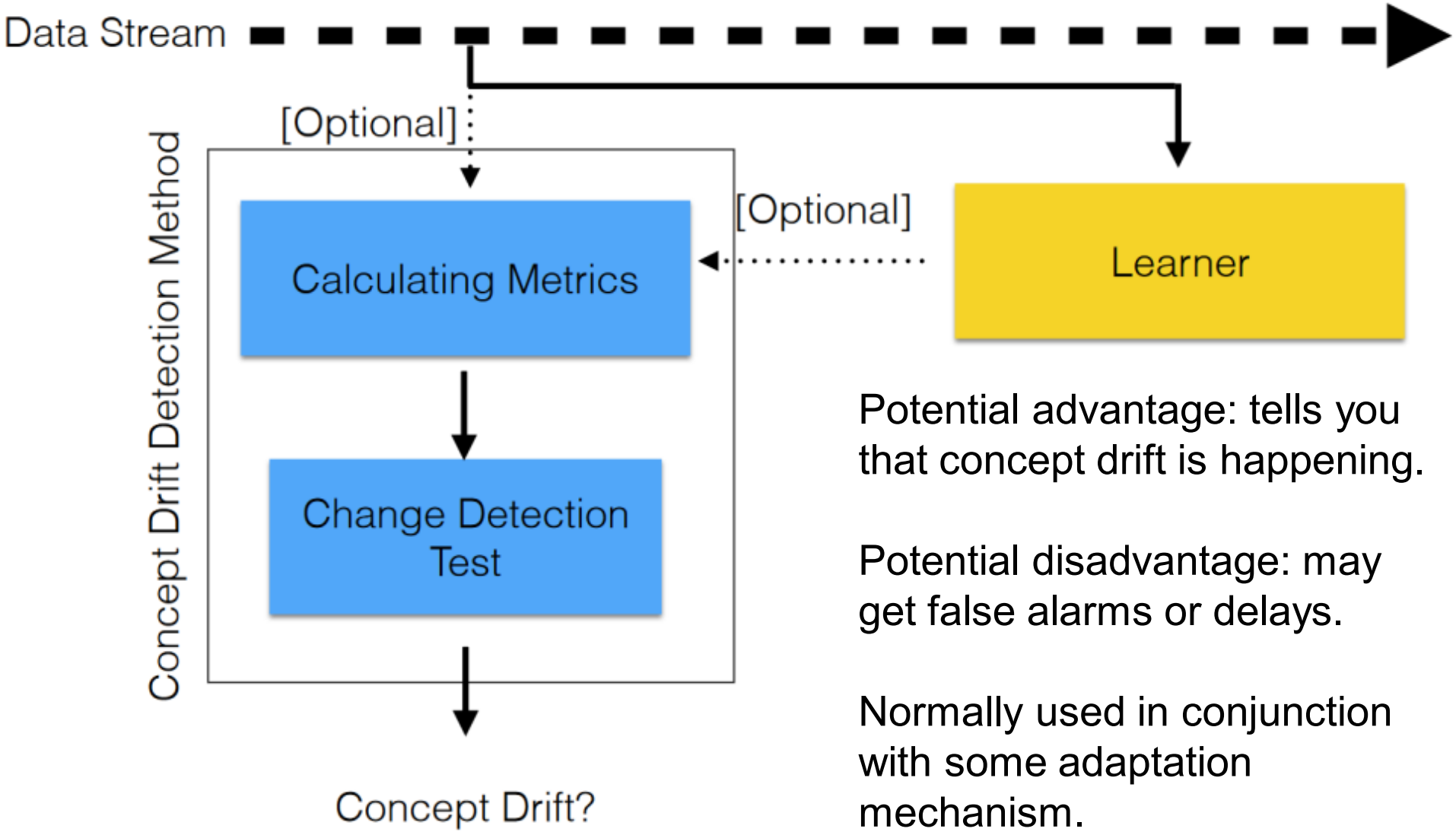
Three drift detector groups:

- Supervised.
  Use classification error or class distribution to detect changes - very expensive
- Semi-supervised.
  Use reduced number of important objects for detection - takes into account the cost of labeling
- Unsupervised.
  Based solely on properties of data - useful for detecting virtual drift, as real drift requires at least partial access to labels





The General Idea of Concept Drift Detection



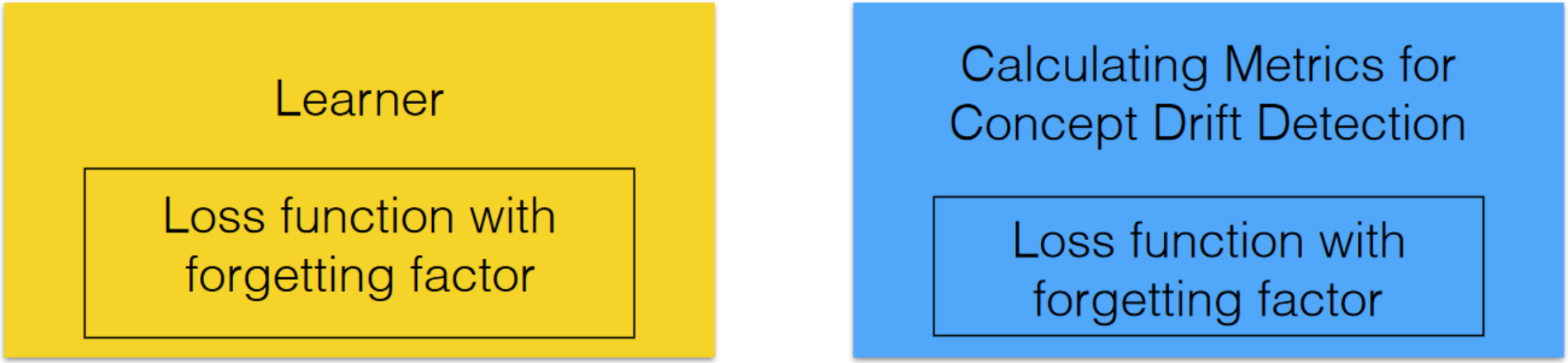
Potential advantage: tells you that concept drift is happening.

Potential disadvantage: may get false alarms or delays.

Normally used in conjunction with some adaptation mechanism.

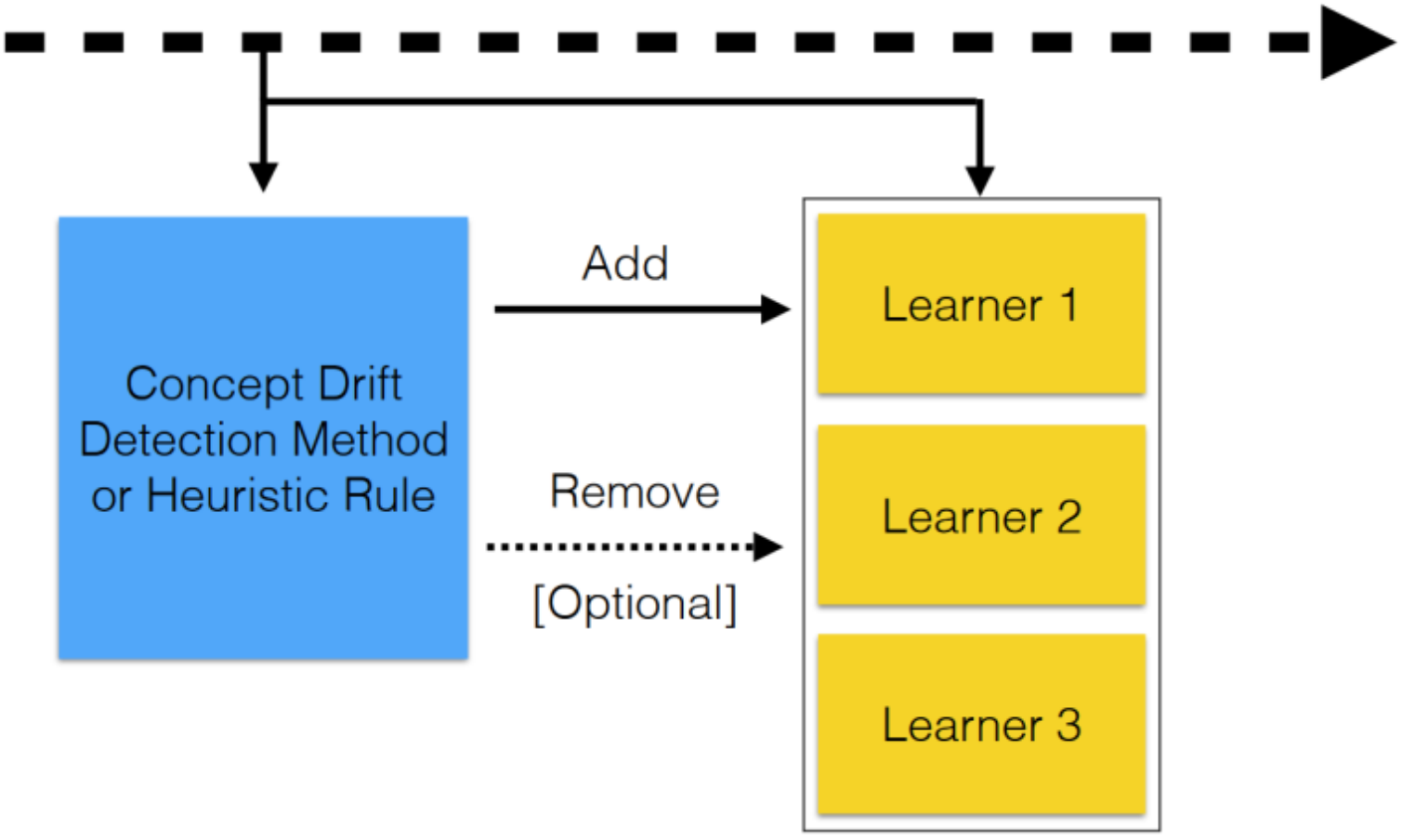
The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 1: forgetting factors



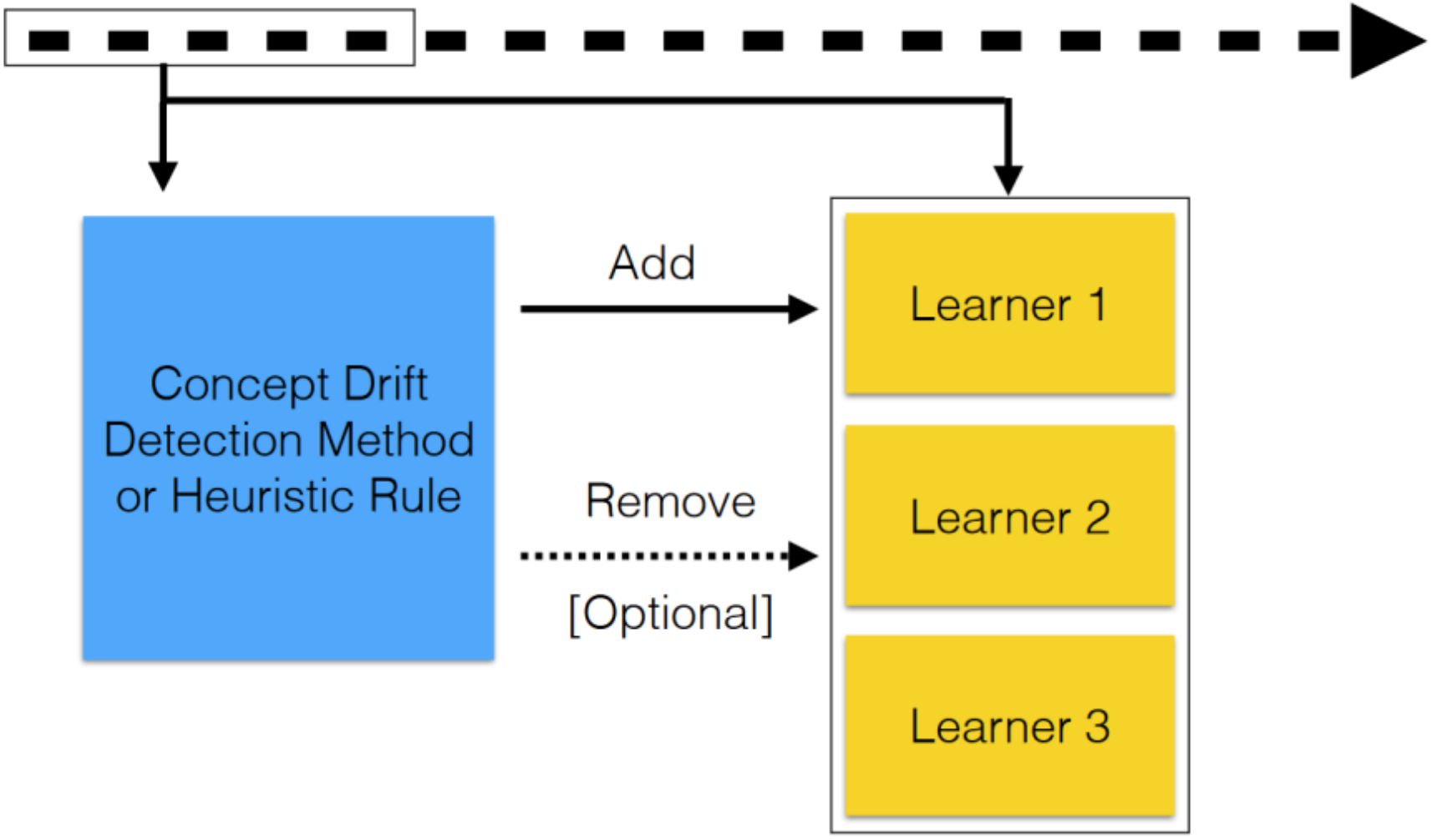
The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 2: adding / removing learners in online learning



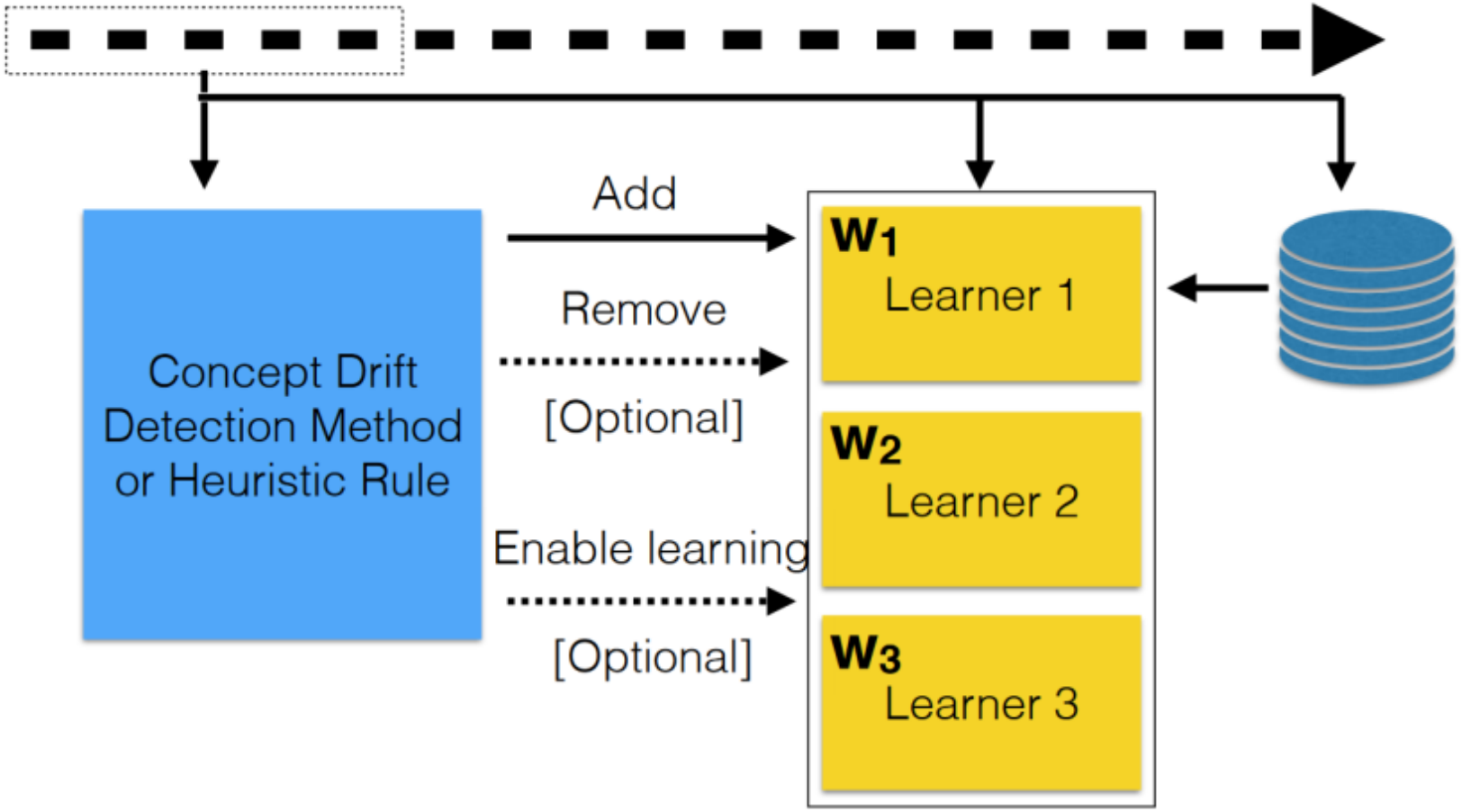
The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 3: adding / removing learners in chunk-based learning



The General Idea of Adaptation Mechanisms

Other strategies / components are also possible



Limited access to true class labels

Most of the works done in data streams assume that true class labels are available for each example or batch of objects immediately after processing.

This would however require extremely high labeling costs - which is far from being a realistic assumption.

We should assume either that we deal with labeling delay or we have a limited labeling budget.

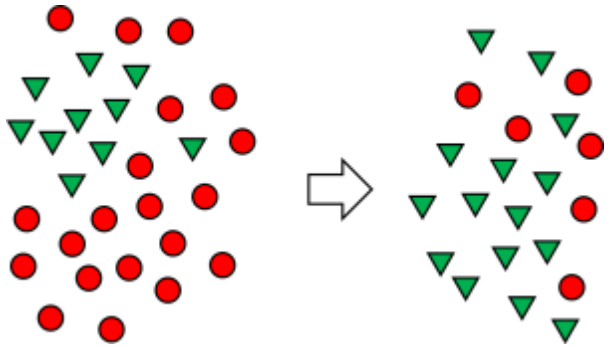
Active learning allows us to select samples to be labeled according to their value to drift detector and / or learner.

Active learning is especially challenging in the presence of concept drift, in order to rapidly adapt to changes.

Learning from imbalanced data streams

The issue of class imbalance is becomes much more difficult in non-stationary streaming scenarios:

- Imbalance ratio, as well as role of classes may evolve
- Class separation may change, as well as class structures
- We work with limited computational resources under time constraints
- Batch cases easier to handle, as one may handle chunks independently
- Online cases highly difficult due necessity of adapting to local changes



Outline

- An introduction to data stream mining
  - What is a data stream?
  - Data stream characteristics
  - Concept drift
- Learning algorithms for data streams
  - Main classifier types for drifting data streams
  - Ensemble learning from drifting data streams
- Limited access to ground truth in data streams
  - How to cope with sparse class labels?
  - Active learning under concept drift
  - Semi-supervised learning from drifting data streams
- Advanced problems
  - Recurrent Concepts
  - Open challenges and future directions

Access to true class labels

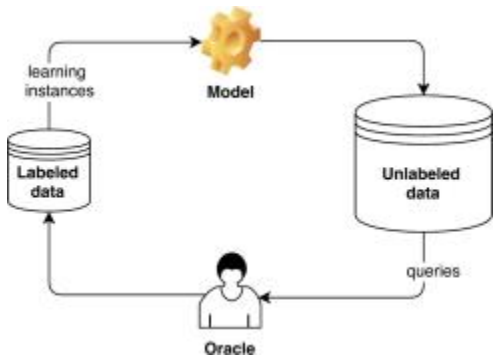
Most of the works done in data streams assume that true class labels are available for each example or batch of objects immediately after processing.

This would however require extremely high labeling costs - which is far from being a realistic assumption.

We should assume either that we deal with labeling delay or we have a limited labeling budget.

Active learning allows us to select samples to be labeled according to their value to drift detector and / or learner.

Access to labels is especially valuable when changes occur and thus active learning should be conducted in a more guided manner.



Model Confidence vs Accuracy.

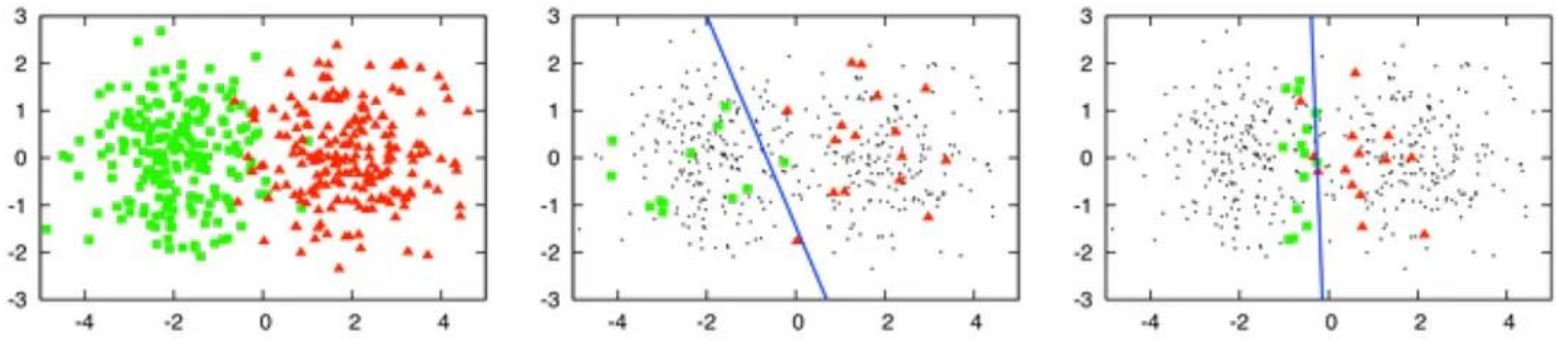
Model Confidence, is an indication of how likely (probability) the predictions of a machine learning algorithm are correct.

Active Learning

Active learning is a supervised machine learning approach that aims to optimize annotation using a few small training samples.

Active learning is the name used for the process of prioritising the data which needs to be labelled in order to have the highest impact to training a supervised model.

Active learning can be used in situations where the amount of data is too large to be labelled and some priority needs to be made to label the data in a smart way.



400 instances sampled from 2 class Gaussians

random sampling
30 labeled instances
(accuracy=0.7)

active learning
30 labeled instances
(accuracy=0.9)

Steps for active learning

The steps to use active learning on an unlabelled data set are:

1. A very small subsample of this data needs to be manually labelled.
2. Once there is a small amount of labelled data, the model needs to be trained on it.
3. After the model is trained, the model is used to predict the class of each remaining unlabelled data point.
4. A score is chosen on each unlabelled data point based on the prediction of the model.
5. Once the best approach has been chosen to prioritise the labelling, this process can be iteratively repeated: a new model can be trained on a new labelled data set, which has been labelled based on the priority score. Once the new model has been trained on the subset of data, the unlabelled data points can be ran through the model to update the prioritisation scores to continue labelling. In this way, one can keep optimising the labelling strategy as the models become better and better.

https://towardsdatascience.com/active-learning-in-machine-learning-525e61be16e5

Prioritisation scores: Least confidence

Takes the highest probability for each data point's prediction, and sorts them from smaller to larger.

$$s_{LC} = \text{argmax}\,(1 - P(\hat{y}|x))$$
$$\hat{y} = \underset{y}{\text{argmax}}\, P(y|x)$$

Data	Class 1	Class 2	Class 3
X1	0.9	0.07	0.03
X2	0.87	0.03	0.1
X3	0.2	0.5	0.3
X4	0	0.01	0.99

In this case, the algorithm would first choose the maximum probability for each data point, hence:

- X1: 0.9
- X2: 0.87
- X3:0.5
- X4:0.99.

The second step is to sort the data based on this maximum probability (from smaller to bigger), hence X3, X2, X1 and X4.

Prioritisation scores: Margin sampling

This method takes into account the difference between the highest probability and the second highest probability.

$$s_{MS} = \underset{x}{\operatorname{argmin}} \left(P(\hat{y}_{max}|x) - P(\hat{y}_{max-1}|x) \right)$$

Data	Class 1	Class 2	Class 3
X1	0.9	0.07	0.03
X2	0.87	0.03	0.1
X3	0.2	0.5	0.3
X4	0	0.01	0.99

The data points with the lower margin sampling score would be the ones labelled the first; these are the data points the model is least certain about between the most probably and the next-to-most probable class.

- X1: 0.9–0.07 = 0.83
- X2: 0.87–0.1 = 0.86
- X3: 0.5–0.3 = 0.2
- X4: 0.99–0.01 = 0.98

The second step is to sort the data based on this (from smaller to bigger), hence X3, X1, X2 and X4.

Prioritisation scores: Entropy

We can define the entropy score prioritisation as follows

$$s_E = \underset{x}{\operatorname{argmax}} \left(-\sum_i P(\hat{y}_i|x) \log P(\hat{y}_i|x) \right)$$

Data	Class 1	Class 2	Class 3
X1	0.9	0.07	0.03
X2	0.87	0.03	0.1
X3	0.2	0.5	0.3
X4	0	0.01	0.99

If we apply the entropy score

- X1: -0.9*log(0.9)-0.07*log(0.07)-0.03*log(0.03) = 0.386
- X2: -0.87*log(0.87)-0.03*log(0.03)-0.1*log(0.1) = 0.457
- X3: -0.2*log(0.2)-0.5*log(0.5)-0.3*log(0.3) =1.03
- X4: -0*log(0)-0.01*log(0.01)-0.99*log(0.99) = 0.056

X4, 0 should be changed for a small epsilon (e.g. 0.00001) for numerical stability

In this case the data points should be shown in the following order: X3, X2, X1 and X4.

Active Learning vs Reinforcement Learning

Active Learning:

- Requires a human expert to label data.
- Is a type of semi-supervised learning.
- Has a training stage where the model learns from labelled and unlabelled data.
- Typical applications: image classification, natural language processing, object detection.

Reinforcement Learning:

- Does not require a human expert to label data.
- Is a type of unsupervised learning.
- Does not have a training stage; the model learns through trial and error.
- Typical applications: autonomous driving, financial trading, robotics

Active learning for drifting data stream

Active learning assume that we have a realistic labeling budget at our disposal (e.g., 1%, 5%, 10% of instances etc.)

Uniform budget usage is not a good decision, as we should conserve it for the change moment.

Additionally, there are no techniques that allow for saving budget for novel class appearance - yet obtaining labeled instanced from new class is of crucial importance.

Furthermore, in imbalanced data streams we should be interested in getting as much labeled minority instances as possible - but how to predetermine if new instance is in fact a minority one?

Semi-supervised learning for static and streaming data

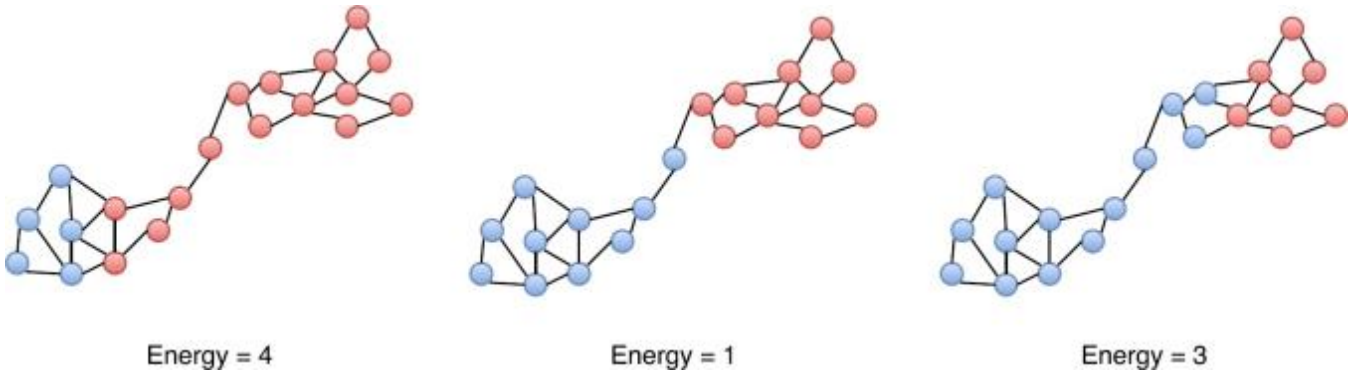
Semi-supervised learning assume that we have a small initial subset of labeled instances and large subset of unlabeled ones.

Labeled instances are used to guide the semi-supervised procedure in order to exploit efficiently the decision space.

Main characteristics of semi-supervised solutions are:

- confidence measure
- addition mechanism
- stopping criteria
- single or multiple learning models

Main approaches based on self-labeling, graph-based solutions and clustering.



Semi-supervised learning for static and streaming data

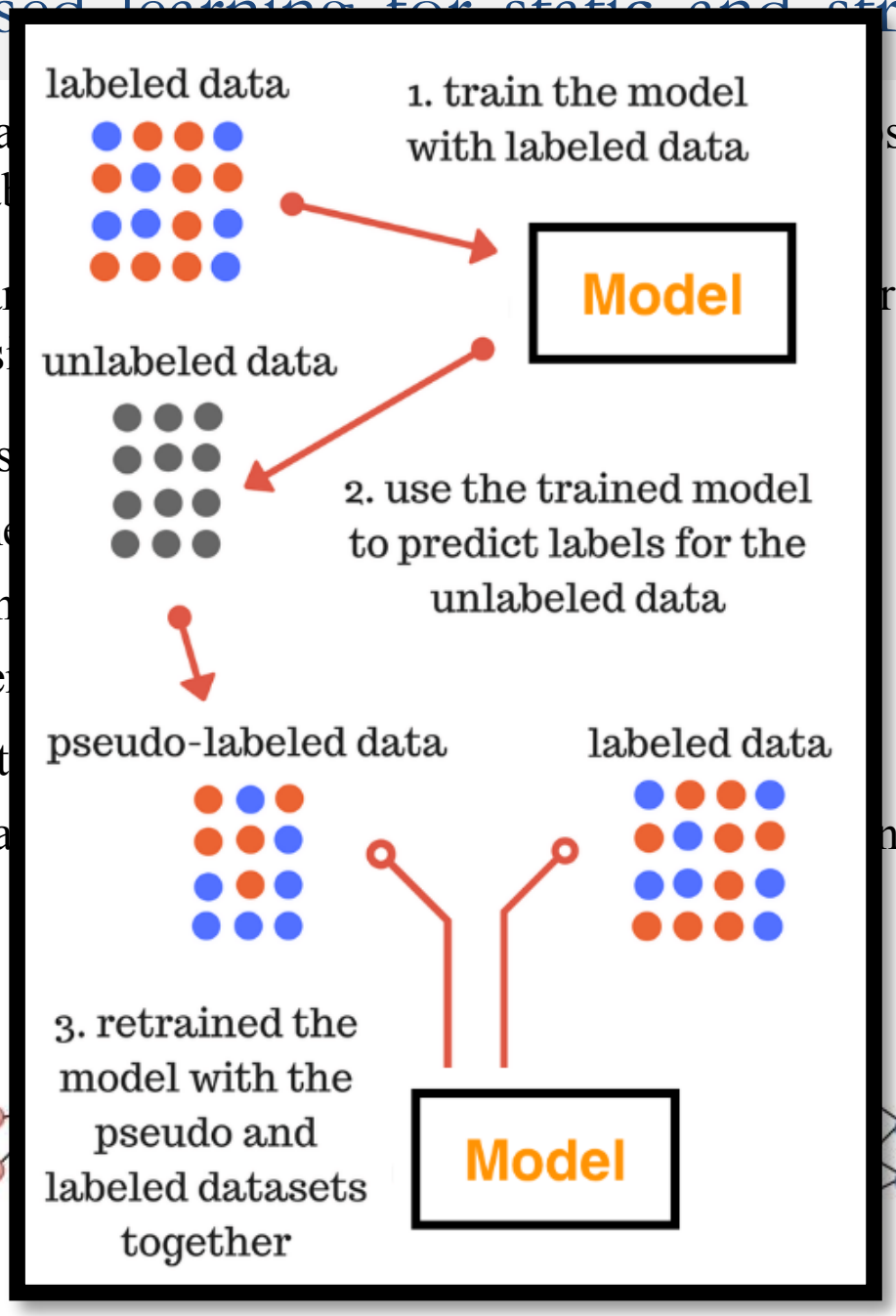
Semi-supervised lea set of labeled instances and large subset of unlab

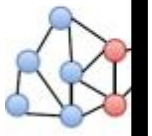
Labeled instances ar re in order to exploit efficiently the decis

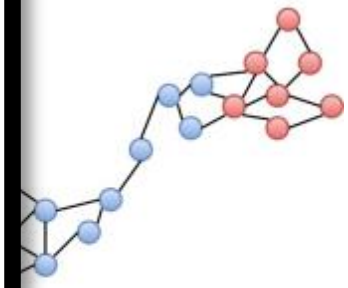
Main characteristics

- confidence me
- addition mech
- stopping crite
- single or mult

Main approaches ba nd clustering.







Energy = 3

Semi-supervised learning for static and streaming data

Two types of methods dedicated to semi-supervised learning:

- transductive - do not generate a model for unseen data, aims at labeling instances
- inductive - train a classifier using unlabeled instances

Semi-supervised learning algorithms usually try to satisfy one of these three assumptions:

- smoothness assumption - if samples are close to each other in high density region, then they may share the same label
- cluster assumption - if samples can be grouped into separated clusters, then points in the same cluster are likely to be in the same class
- manifold assumption - high-dimensionality data can be effectively analyzed in lower dimensions

Active learning framework

Our active learning is guided by an underlying classifier that selects most useful instances for labeling from unlabeled set $\mathscr{U}$:

$$q = \arg\max_{x \in \mathscr{U}} \Psi(h, x). \tag{1}$$

As we work with data streams, we formulate an incremental update of the underlying classification hypothesis under selected training algorithm A and i-th iteration:

$$h_{i+1} = A\left(\{q_k, o(q_k)\}_{k=1}^{i}\right), \tag{2}$$

where

$$q_i = \arg\max_{x \in \mathscr{U}_i} \Psi(h_i, x), \tag{3}$$

$$\mathscr{U}_{i+1} = \mathscr{U}_i \setminus \{q_i\}. \tag{4}$$

Thus classifier in our active learning scenario adapts over time based on previous experience:

$$q_i = \arg\max_{x \in \mathscr{U}_i} \Psi_i(h_i, x), \tag{5}$$

Ensemble active learning

Conduct an active learning process using an ensemble of L classifiers[1]:

$$\Pi = \{\Psi_1, \cdots, \Psi_L\}, \tag{6}$$

This allows for a more robust instance selection for label query.

Krawczyk and Cano: Instead of pooling their decision using voting strategies (like in Query by Committee), propose to select a classifier responsible for a given instance query.

This allows to better utilize a pool of diverse classifiers and select one that can anticipate the direction of changes better than remaining ones.

The idea behind this is similar to dynamic classifier selection - exploiting individual classifier's competencies.

Krawczyk et al. used Multi-Arm Bandit.

---

[1]Bartosz Krawczyk, Alberto Cano: Adaptive Ensemble Active Learning for Drifting Data Stream Mining. IJCAI 2019: 2763-2771

Multi-armed bandit problem

Multi-armed bandit problem is a classic reinforcement learning example where we are given a slot machine with *n* arms (bandits) with each arm having its own rigged probability distribution of success. Pulling any one of the arms gives you a stochastic reward of either R=+1 for success, or R=0 for failure.

Example(s)

If the reward distributions are given in the image below for the $K = 3$ arms, what is the best strategy to maximize your expected reward?

$Poi(\lambda = 1.36)$ $Bin(n = 10, p = 0.4)$ $\mathcal{N}(\mu = -1, \sigma^2 = 4)$

This bandit problem allows us to formally model this tradeoff between:

- Exploitation: Pulling arm(s) we know to be "good".
- Exploration: Pulling less-frequently pulled arms in the hopes they are also "good" or even better.

Multi-armed bandit problem

Multi-armed bandit problem is a classic reinforcement learning example where we are given a slot machine with *n* arms (bandits) with each arm having its own rigged probability distribution of success. Pulling any one of the arms gives you a stochastic reward of either R=+1 for success, or R=0 for failure.

Example(s)

If the reward distributions are given in the image below for the $K = 3$ arms, what is the best strategy to maximize your expected reward?

$Poi(\lambda = 1.36)$ $Bin(n = 10, p = 0.4)$ $\mathcal{N}(\mu = -1, \sigma^2 = 4)$

Regret is the difference between

- The best possible expected reward (if you always pulled the best arm).
- The actual reward you got over T arm-pulls.

Multi-armed bandit problem

Multi-armed bandit problem is a classic reinforcement learning example where we are given a slot machine with *n* arms (bandits) with each arm having its own rigged probability distribution of success. Pulling any one of the arms gives you a stochastic reward of either R=+1 for success, or R=0 for failure.

Example(s)

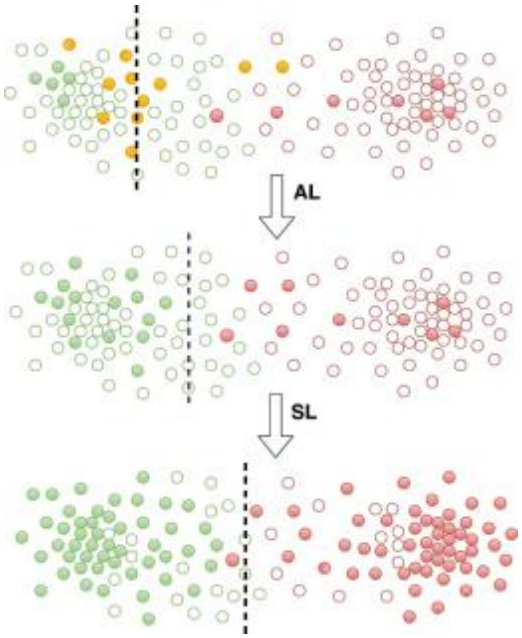
If the reward distributions are given in the image below for the $K = 3$ arms, what is the best strategy to maximize your expected reward?

$Poi(\lambda = 1.36)$ $Bin(n = 10, p = 0.4)$ $\mathcal{N}(\mu = -1, \sigma^2 = 4)$

Epsilon-Greedy: Epsilon-greedy is a simple algorithm that selects the arm with the highest estimated reward with probability 1-epsilon and selects a random arm with probability epsilon.

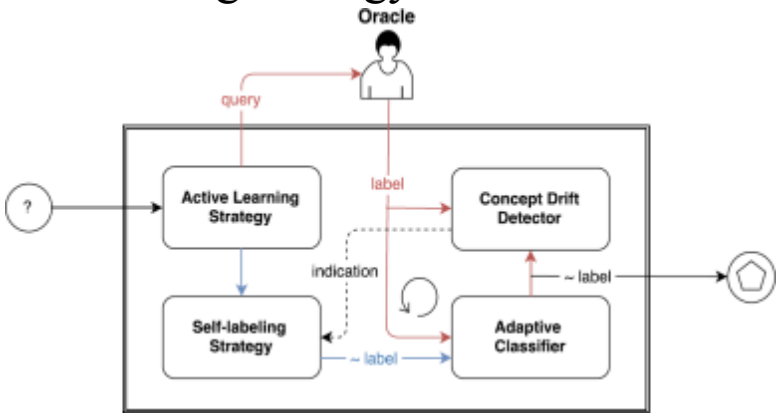
Motivation: Limited Access to Ground Truth

- Active learning allows for an informative selection of instances that will be most useful for adjusting the classifier to the current state of the stream. However, each such query reduces the available budget.
- Self-labeling allows to exploit discovered data structures and improve the competency of a classifier at no cost, yet offers no quality validation.
- These procedures are complimentary - active learning can be interpreted as an exploration step and semi-supervised learning as an exploitation step.



Hybrid framework for drifting data stream mining on a budget

- Developed a hybrid framework that uses active learning for creating a meaningful input for self-labeling strategy[2].



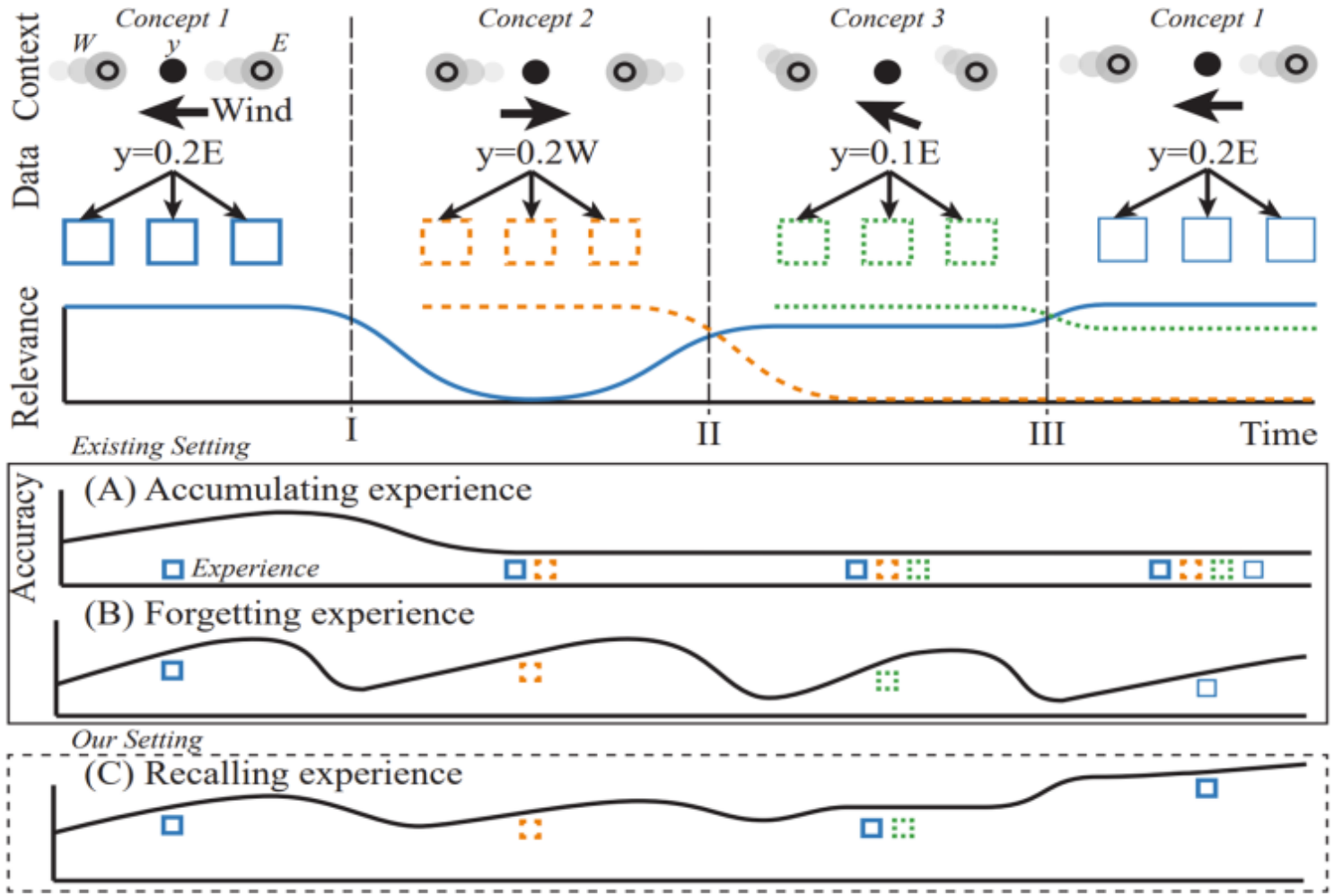
- Drifting data self-labeling were proposed, divided into two groups:
  - blind self-labeling strategies relied on adaptation of uncertainty threshold in a similar manner to their active learning counterparts.
  - informed self-labeling strategies utilized input from the drift detector to adapt their actions depending on the current state of the stream.

[2]Lukasz Korycki, Bartosz Krawczyk: Combining Active Learning and Self-Labeling for Data Stream Mining. CORES 2017: 481-490

Outline

- An introduction to data stream mining
  - What is a data stream?
  - Data stream characteristics
  - Concept drift
- Learning algorithms for data streams
  - Main classifier types for drifting data streams
  - Ensemble learning from drifting data streams
- Limited access to ground truth in data streams
  - How to cope with sparse class labels?
  - Active learning under concept drift
  - Semi-supervised learning from drifting data streams
- Advanced problems
  - Recurrent Concepts
  - Open challenges and future directions

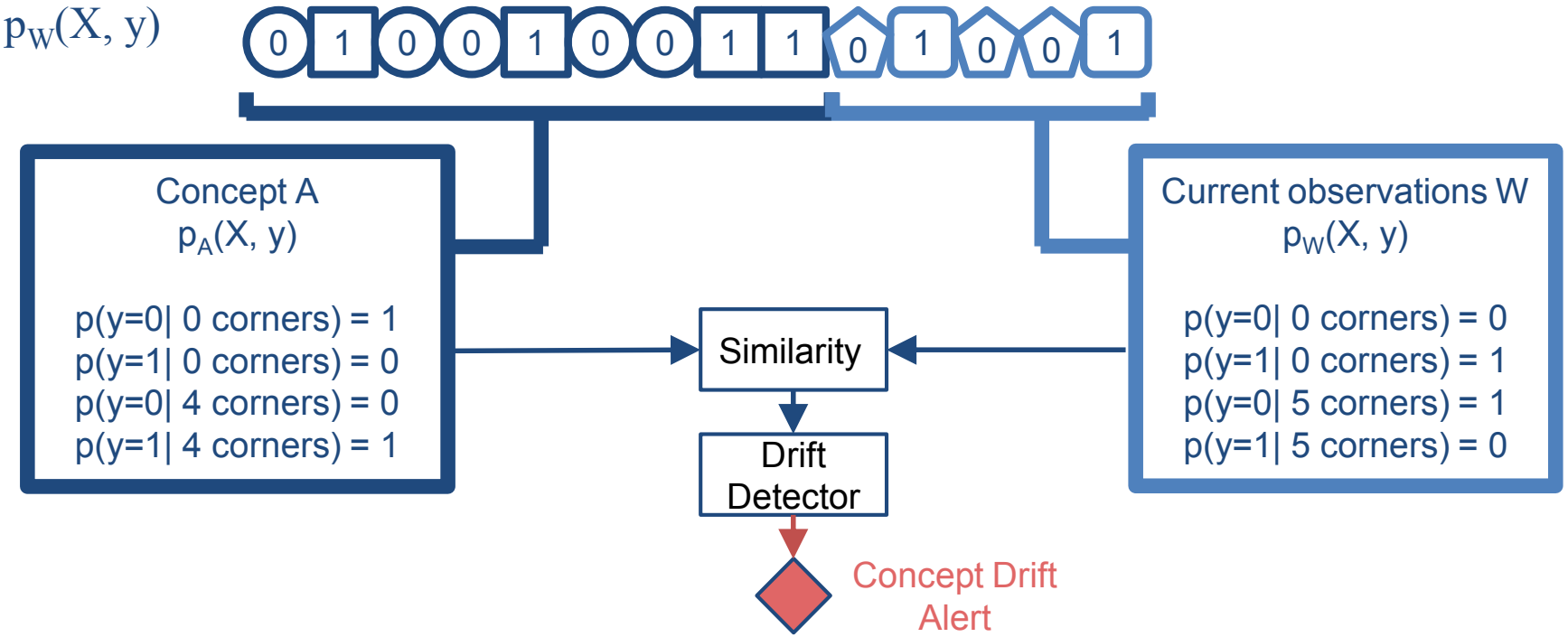
Relevance Experience



B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

Concept Drift Detection

Methods for detecting a difference in the distribution of current observations: $p_A(X, y) \neq p_W(X, y)$



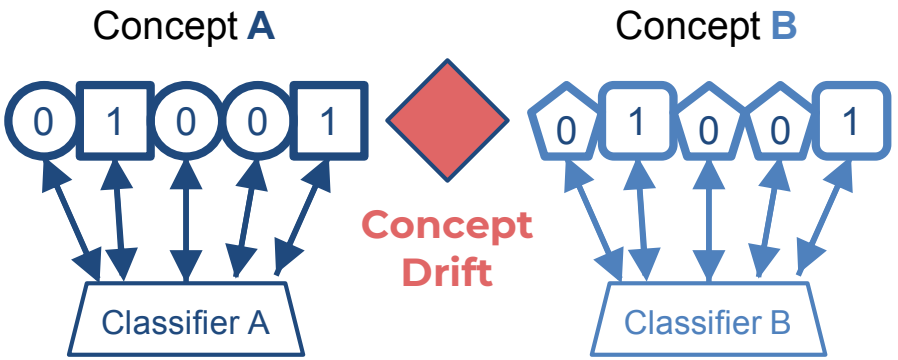
B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

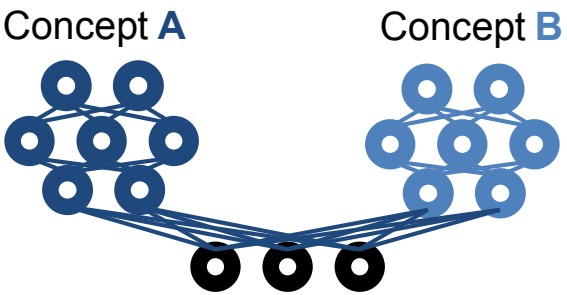
Forgetting experience

Previous observations describe the old distribution and may conflict with the new concept.

Conflict: $p_A(y|X) \neq p_W(y|X)$

The forgetting function modifies the active classifier so it does not use experience irrelevant to the new concept.

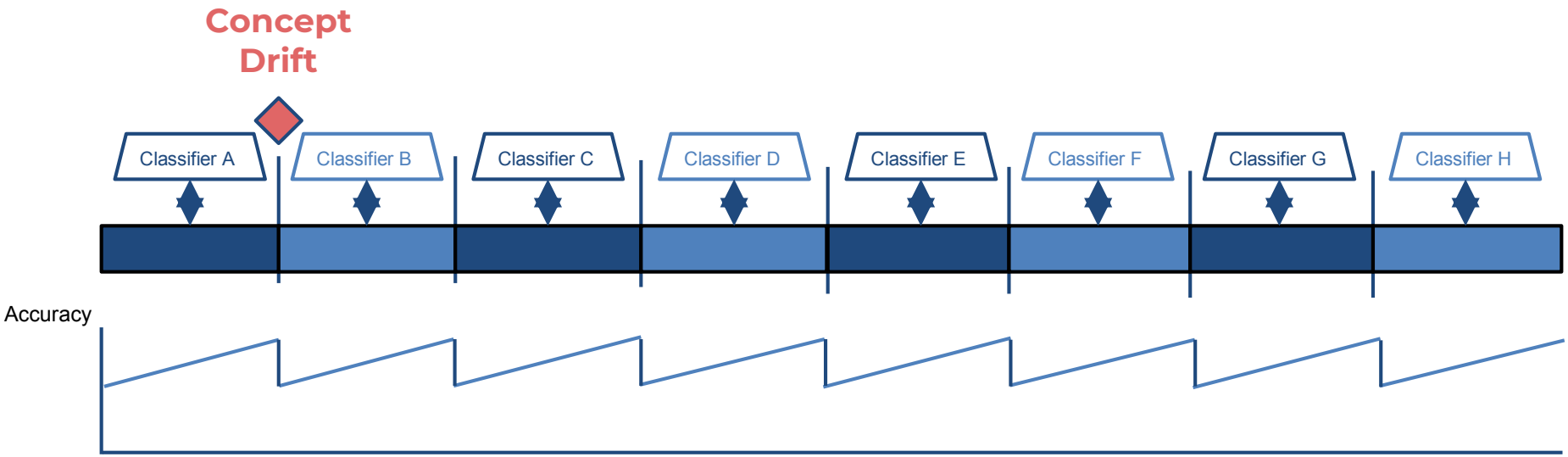




B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

Adaptive Learning

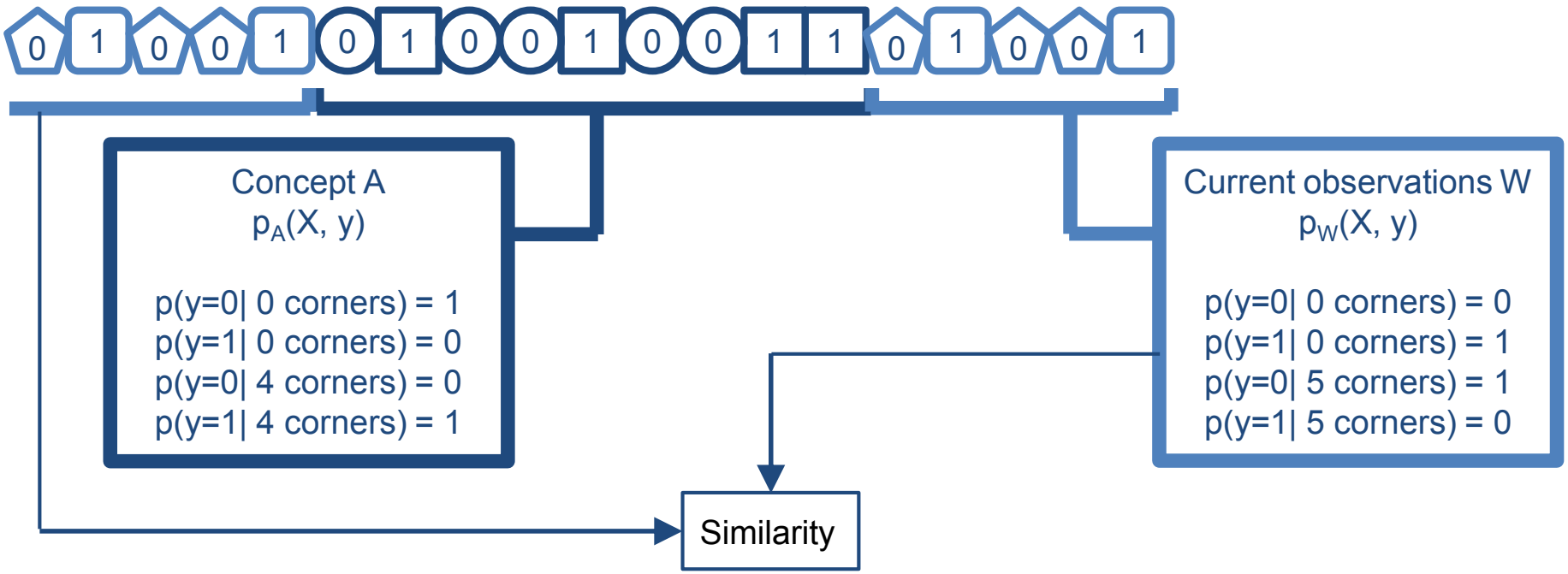
Forgetting avoids experience conflict, but may lead to catastrophic forgetting



B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

Concept Re-identification

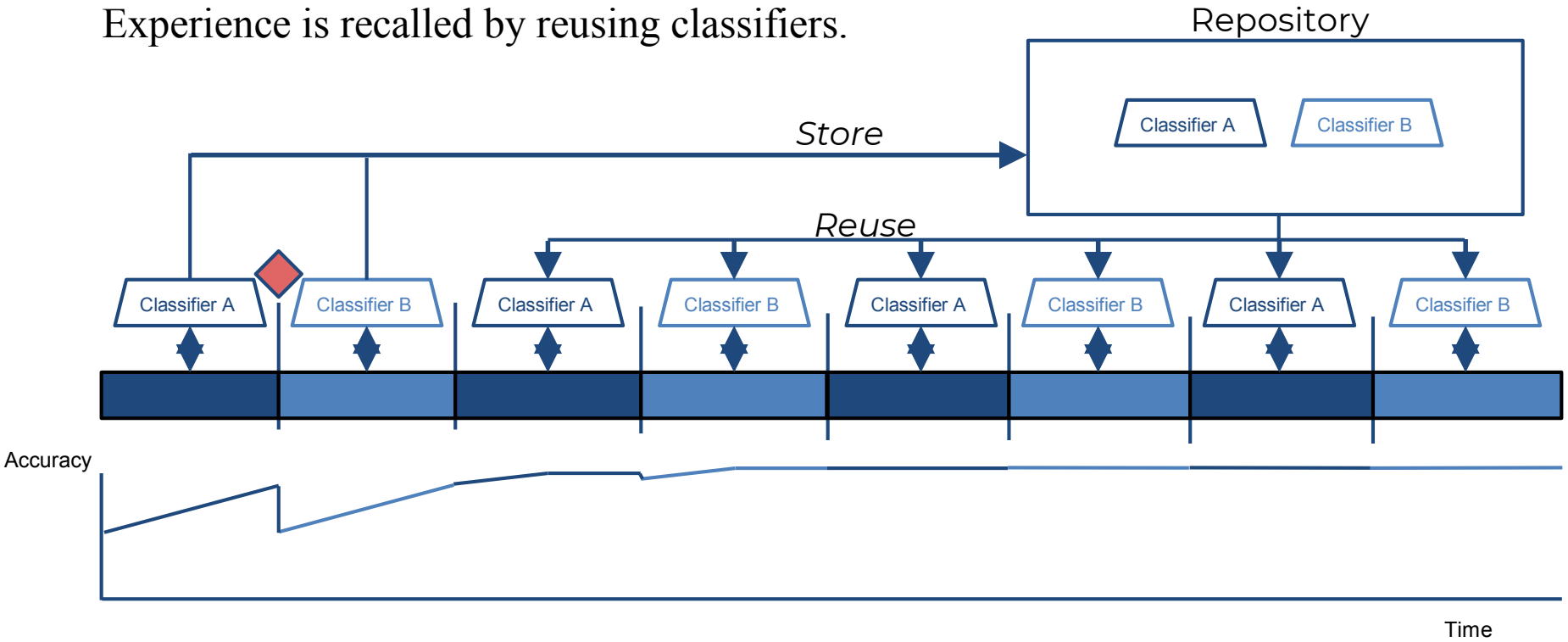
Methods for detecting a similarity in the distribution of previous observations: $p_i(X, y) \sim p_W(X, y)$



B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

Recalling experience - reuse

A repository is used to store and query trained classifiers.
Experience is recalled by reusing classifiers.



B. Halstead, Y. S. Koh, P. Riddle, M. Pechenizkiy, A. Bifet and R. Pears, "Fingerprinting Concepts in Data Streams with Supervised and Unsupervised Meta-Information," 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 2021, pp. 1056-1067, doi: 10.1109/ICDE51399.2021.00096.

Open challenges and future directions

- Interpretability vs accuracy of drifting data streams: explaining the concept drift
  - Explainable artificial intelligence (XAI) for non-stationary data
  - Understanding what and why changed and how can we use this knowledge to improve adaptation
- Learning for extremely sparsely labeled data streams
  - Learning from data streams without any access to class labels
  - Merging unsupervised methods with supervised predictors
- Multi-view asynchronous data streams
  - Transferring useful information among multiple data streams
  - Using different views on data streams to extract more information-rich representation and better detect drifts
- Robustness to adversarial attacks
  - "Fake" and malicious concept drifts
  - Appearance of artificial classes to increase the class imbalance and learning difficulty

Resources

Tutorial
https://riverml.xyz/
https://streamlearningtutorial2020.netlify.app/

Part of these are from:
MINKU, L.; WANG, S.; BORACCHI, G. . "Learning Class Imbalanced Data Streams", *World Congress on Computational Intelligence (WCCI)*, July 2018

http://bigdataieee.org/BigData2020/files/IEEE_BigData_2020_Tutorial7.pdf