Big Data Stream Mining

Outline

Bartosz Krawczyk[1] Alberto Cano[1]

[1]Department of Computer Science
Virginia Commonwealth University
Richmond, VA
USA

{bkrawczyk,acano}@vcu.edu

Outline of the tutorial

- Part 1: An introduction to data stream mining
  - What is a data stream?
  - Data stream characteristics
  - Concept drift
- Part 2: Learning algorithms for data streams
  - Main classifier types for drifting data streams
  - Ensemble learning from drifting data streams
- Part 3: 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
- Part 4: Advanced problems
  - Evolutionary algorithms for data stream mining
  - Multi-label data streams
  - Open challenges and future directions

Big Data Stream Mining

Part 1: An introduction to data stream mining

Bartosz Krawczyk[1] Alberto Cano[1]

[1]Department of Computer Science
Virginia Commonwealth University
Richmond, VA
USA

{bkrawczyk,acano}@vcu.edu

Outline

Outline

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.

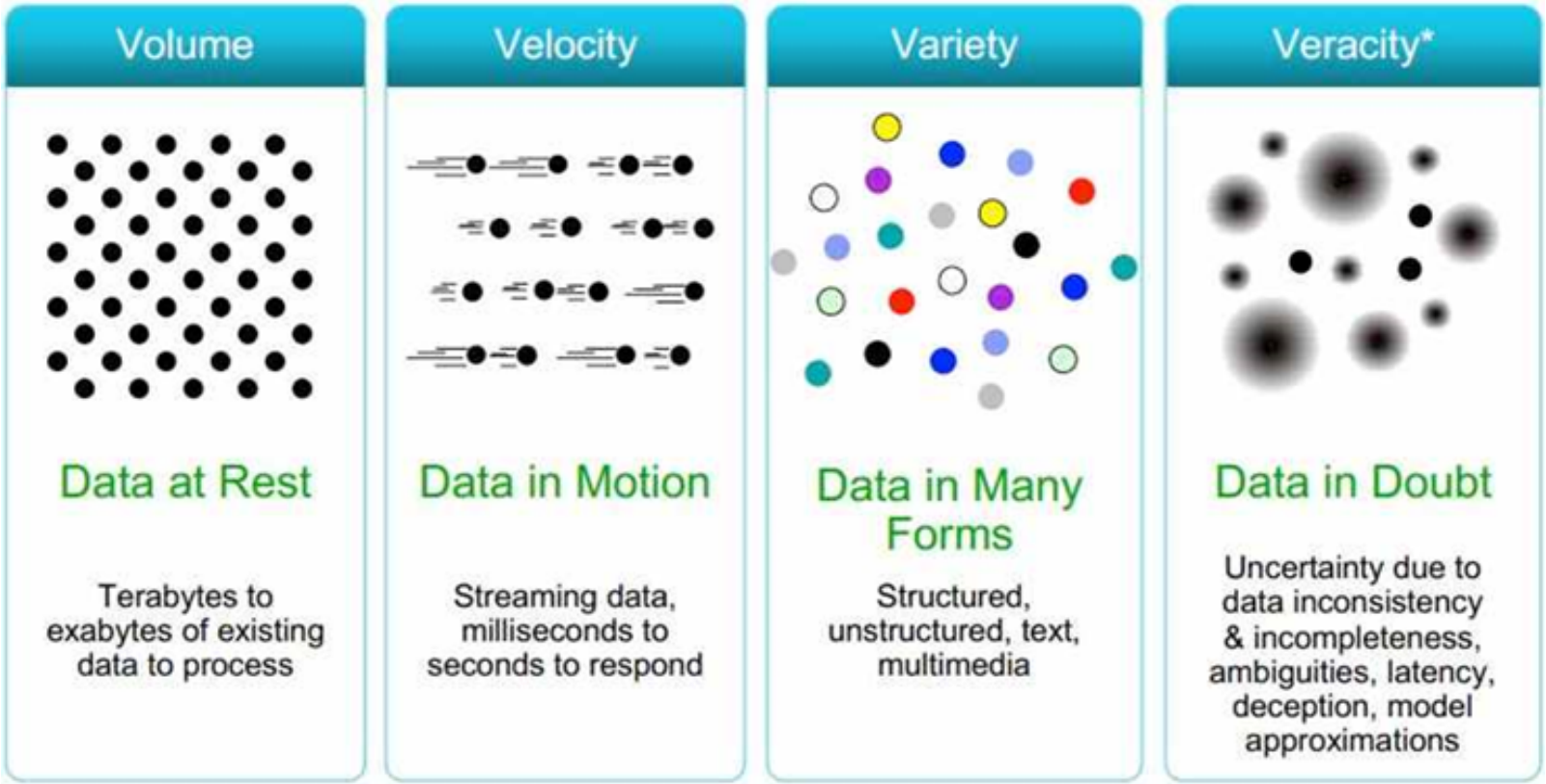
We must deal with new data sources, generating high-speed, massive and heterogeneous data.

According to IDC Report in 2018 close to 5.8 zetabytes of data was generated.

We require novel, efficient and adaptive methods for extracting valuable information from such sources.

Modern data flood

How many V's in Big Data?



There are many V's being constantly added: value, variability and visualization.

What is a data stream?

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

Velocity refers to the speed at which the data is generated and input into the analyzing system.

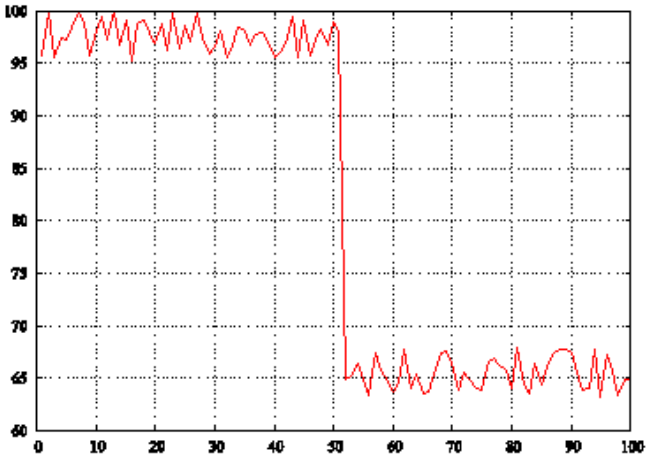
Data streams are also often connected with Volume, forcing us to cope with massive and dynamic problems.

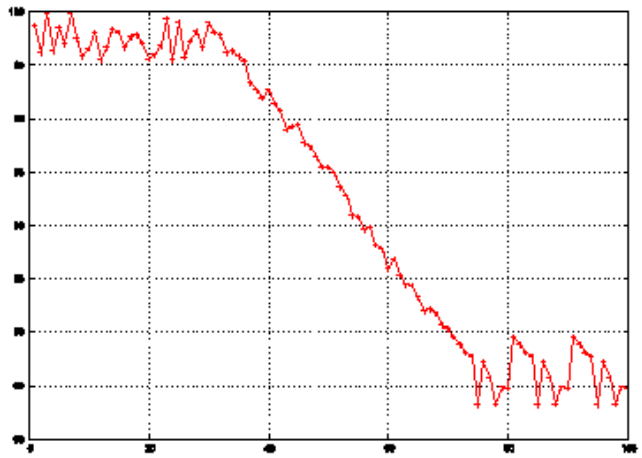
High-speed data streams: arising demands for fast-changing and continuously arriving data to be analyzed in real time.

Outline

Requirements for data stream algorithms

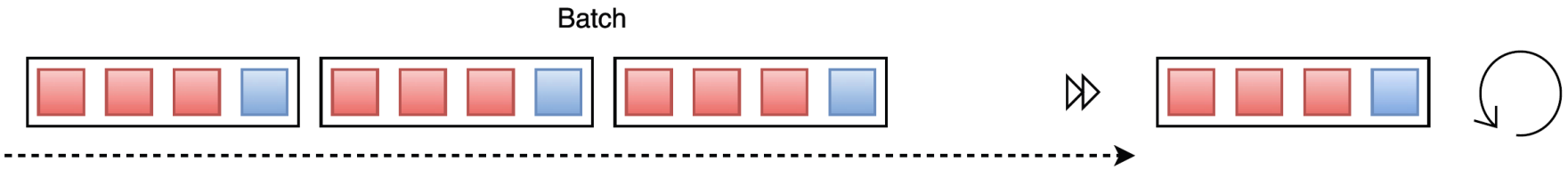
- Incremental processing
- Limited time:
  - Examples arrive rapidly
  - Each example can be processed only once
- Limited memory:
  - Streams are often too large to be processed as a whole
- Changes in data characteristics:
  - Data streams can evolve over time



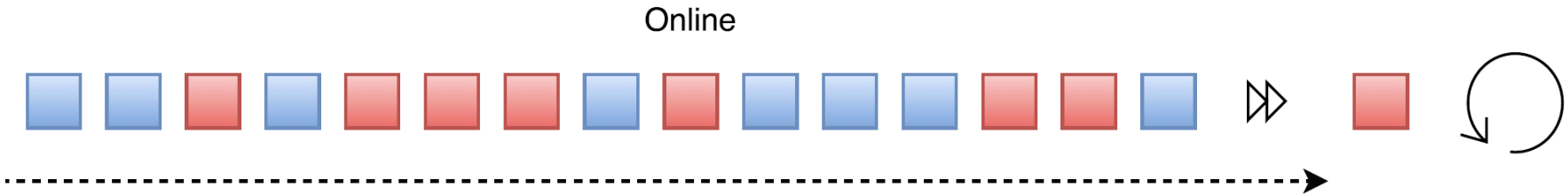


Evaluating data stream algorithms

- 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.

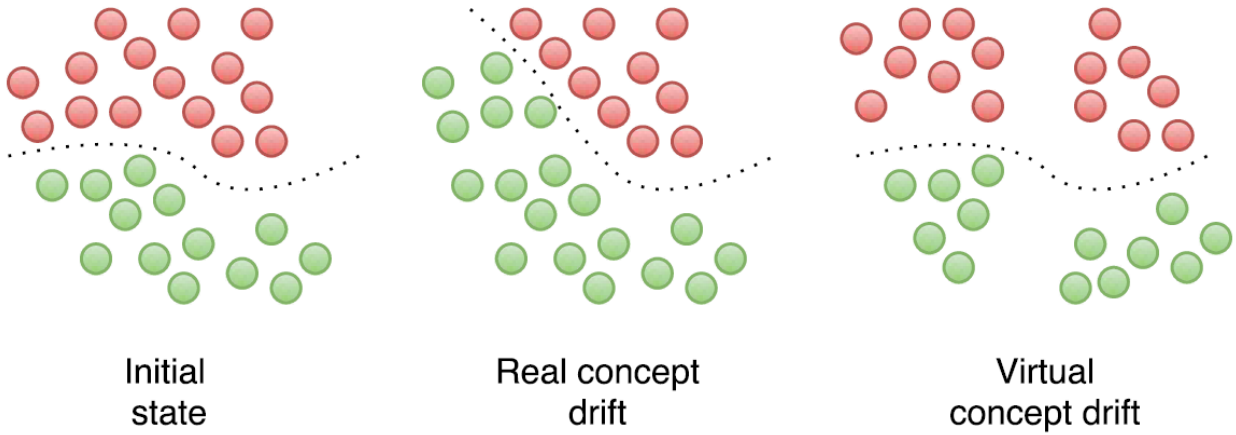
Additional metrics are crucial for evaluating streaming classifiers:

- Memory consumption
- Update time
- Classification time

Outline

Concept drift

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



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

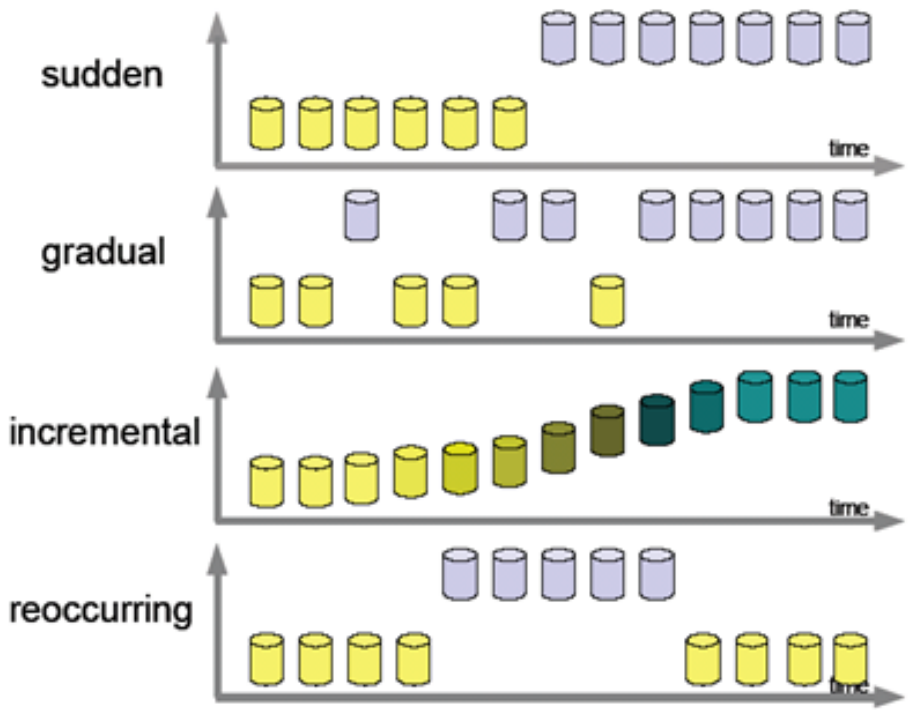
Concept drift

Let us assume that our stream consist of a set of states $\mathbf{S} = \{S_1, S_2, \cdots, S_n\}$, 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 $D_j \cap D_{j+1} \neq \emptyset$
- Reocurring, 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 influence 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)

Handling concept drift

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

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

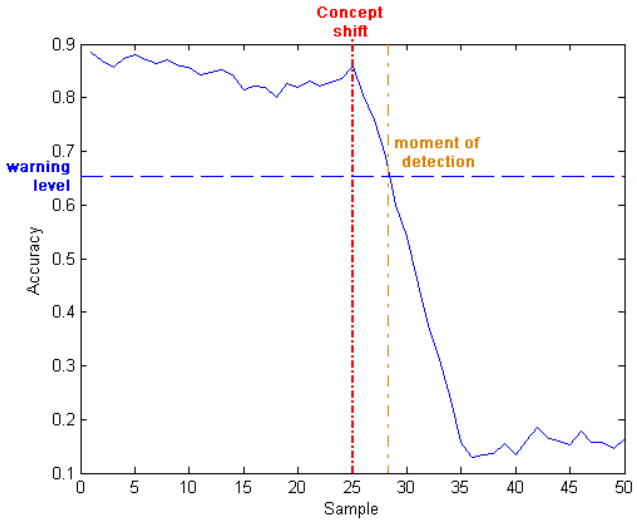
Drift detectors

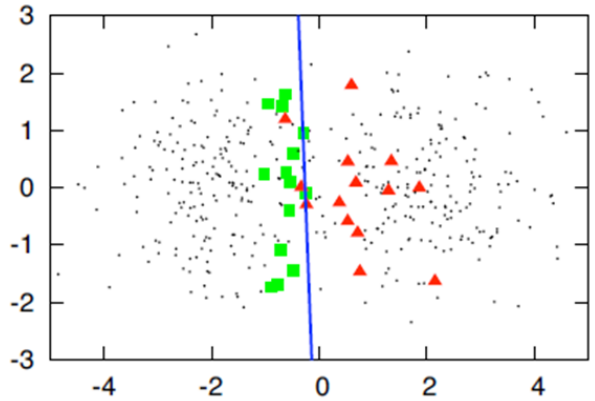
Algorithms that address the question of when drift occurs, being usually a separate tool from the actual classifier.

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





Outline

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.

Novelty detection

Novelty detection plays a crucial role in mining data streams.

First applications: novelty as rare, atypical objects.

Novelty detection used for detecting concept drift. Frequent novel data = drift occurred.

Current trends: novelty detection = evolving class structure. Initial set of classes in not the definite one and new classes may appear with the progress of stream:

$$P_{S_j}(y = M_i) = 0 \text{ and } P_{S_{j+1}}(y = M_i) > 0. \quad (1)$$

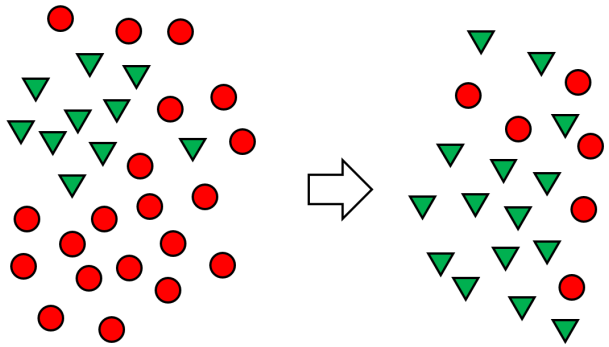
Previously known classes may start to appear less frequently and finally stop appearing at all.

$$P_{S_j}(y = M_i) > P_{S_{j+1}}(y = M_i). \quad (2)$$

Learning from imbalanced data streams

The issue of class imbalance is becomes much more difficult in non-stationary streaming scenarios [1] [2]:

- 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



[1] Bartosz Krawczyk: Learning from imbalanced data: open challenges and future directions. Progress in AI 5(4): 221-232 (2016)

[2] Alberto Fernandez, Salvador Garcia, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, Francisco Herrera: Learning from Imbalanced Data Sets. Springer 2018

Ending notes

Thank you for your attention! Q & A time!

Next: **Part 2: Learning algorithms for data streams**

Big Data Stream Mining

Part 2: Learning algorithms for data streams

Bartosz Krawczyk[1] Alberto Cano[1]

[1]Department of Computer Science
Virginia Commonwealth University
Richmond, VA
USA

{bkrawczyk,acano}@vcu.edu

Outline

Outline

Three main families of classifiers for data streams

All classifiers for data stream mining can be categorized into three groups:

- Sliding window solutions
- Online learners
- Ensemble learners

Sliding window

Assumption: recently arrived data are the most relevant - contain characteristics of the current context. However, their relevance diminishes with the passage of time.

There are two most popular strategies employed:

- Instance selection with a sliding window that cut offs older examples
- Instance weighting that assigns relevance level to each example present in the window

Size of the window has crucial impact. Shorter window - focus on the current concept, prone to local overfitting. Wider window- global outlook on the stream, may consist of instances from mixed concepts.

There is a number of proposals on applying windows with dynamic size or multiple windows at the same time.

Online learners

Online learners for data streams must fulfill the following requirements:

- Each object must be processed only once in the course of training
- The system should consume only limited memory and processing time
- The training process can be paused at any time, and its accuracy should not be lower than that of a classifier trained on batch data collected up to the given time

Some of standard classifiers like Naïve Bayes or Neural Networks can work in online mode.

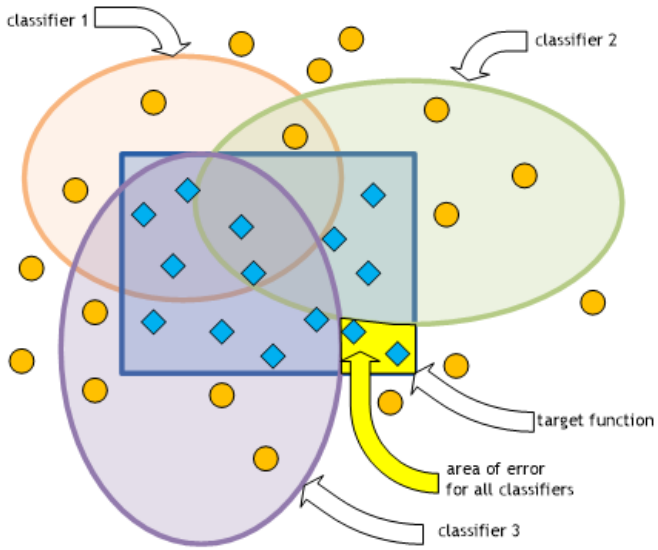
More sophisticated: Concept-adapting Very Fast Decision Trees, online Support Vector Machines, Mondrian Forests or weighted learners.

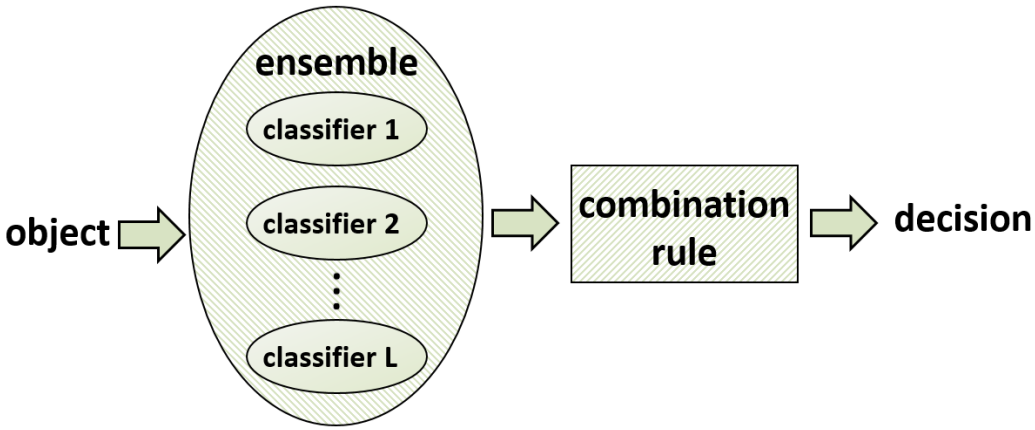
Outline

Advantages of ensembles

Ensemble learning is a well-established area in static machine learning due to the following reasons:

- Classifiers combination can improve the performance of the best individual ones and it can exploit unique classifier strengths
- Avoiding the selection of the worst classifier
- Usually they offer more flexible decision boundary and at the same time they do not suffer from overfitting
- Can be simply and efficiently applied to distributed environments





Advantages of ensembles for stream mining

Ensemble learning can be seen as a natural choice for mining non-stationary data streams[1]:

- It can use the changing concept as a way to maintain diversity
- It has flexibility to incorporate new data:
  - Adding new components
  - Updating existing components
- It offers natural forgetting mechanism via ensemble pruning
- It reduces the variance of base classifiers, thus increasing the stability
- It allows to model changes in data as weighted aggregation of base classifiers

[1] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Wozniak: Ensemble learning for data stream analysis: A survey. Information Fusion 37: 132-156 (2017)

Ensembles for stream mining - taxonomy

Ensembles according to processing modes:

- Block ensembles
- Online ensembles

Ensembles according to their method for adapting to drifting streams:

- Dynamic combiners: base classifiers learned in advance, combination rule adapts to changes
- Ensemble updating: all / some base classifiers updated with incoming examples
- Dynamic ensemble line-up: new classifiers added for incoming data, weakest ones removed from the committee

Dynamic combiners

Based on assumption that concept drift can be modeled as varying classifier combination scheme, e.g., with weights assigned to each classifier.

In order to work we require an efficient pool of initial classifiers with high diversity to capture different properties of the analyzed stream.

Classifier combination block is subject to identical limitations as standard classifiers in regard to time and memory consumption.

Untrained combiners - less accurate, low computational complexity, fast adaptation.

Trained combiners - more accurate, increased complexity, require additional data for training (big limitation for streams).

Ensemble updating

This approach assumes that our ensemble consist of classifiers that can be updated in batch or online modes.

At the beginning we train a set of classifiers that will be continually adapted to the current state of the data stream.

This requires a diversity assurance method, usually realized as initial training on different examples (online Bagging) or different features (online Random Subspaces or online Random Forest).

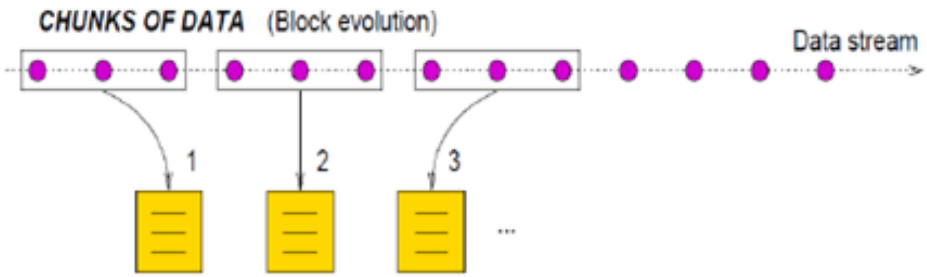
Additional diversity may be assured by using incoming examples to update only some of the classifiers in a random or guided manner.

Dynamic ensemble line-up

This approach assumes that we have a flexible ensemble set-up and add new classifiers for each incoming chunk of data.

Generic scheme:

- Train single initial classifier or K initial classifiers (subject to training data availability)
- For each incoming chunk of data:
  - Train a new component classifier
  - Test other classifiers against the recent chunk
  - Assign weight to each classifier
  - Select top L classifiers (remove the weaker classifiers)



Dynamic ensemble line-up

Advantages of this approach:

- Use static learning algorithms
- May have smaller computational costs than on-line ensembles
- Allows naturally to employ a weighted combination scheme

Classifier combination plays a crucial role.

Most approaches use weighted voting, where weights reflect the usability for the current state of stream or time spent in the ensemble.

Recent proposals use more sophisticated combination based on continuous outputs (support functions) for each class.

Outline

Kappa Updated Ensemble

Ensemble classification algorithm for drifting data streams[2]

Main contributions:

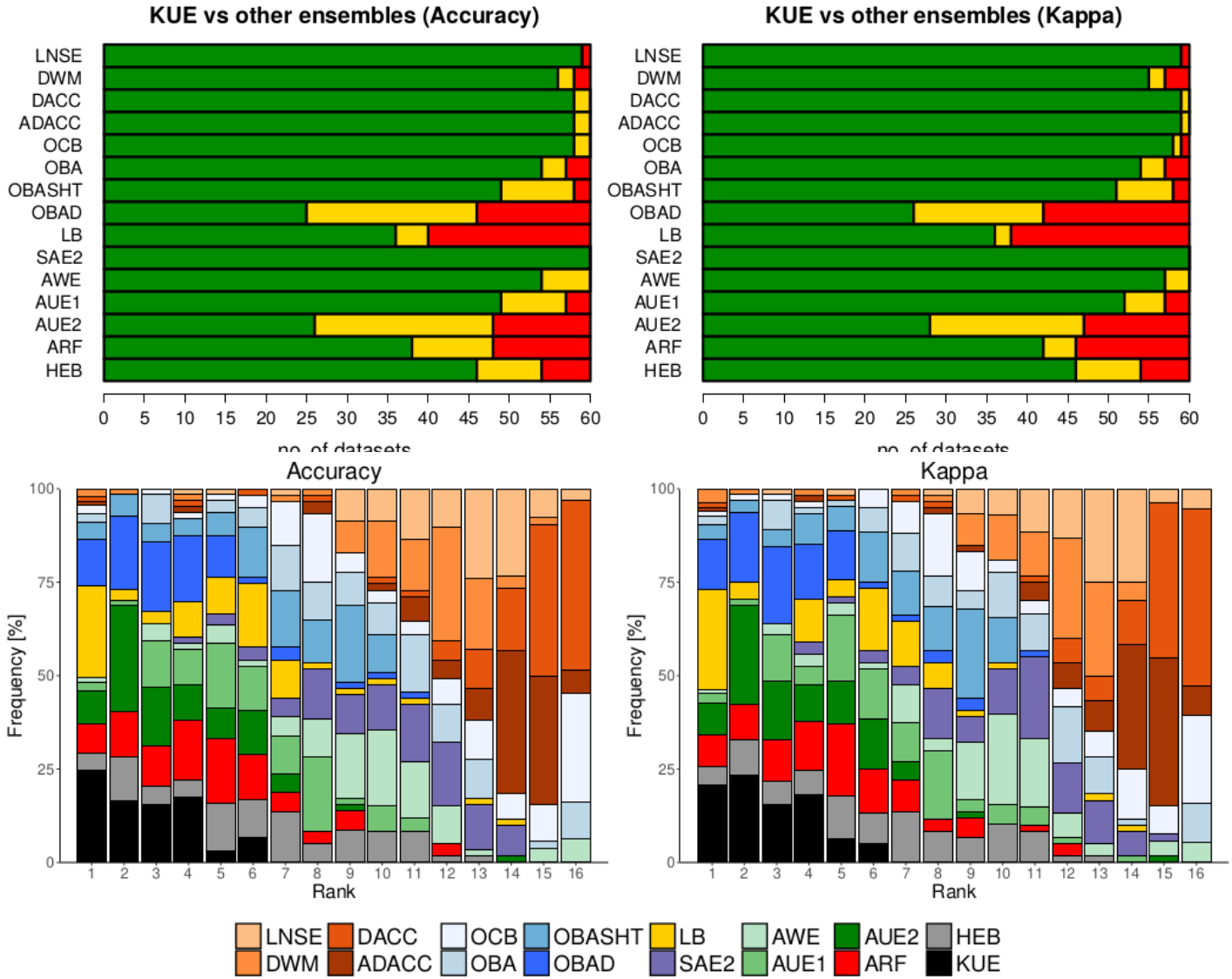
- Kappa statistic for selecting and weighting base classifiers
- Robustness to drifting imbalance ratio distributions
- Hybrid architecture updates base classifiers in an online manner while changes ensemble setup in block-based mode
- Diversification online bagging with random feature subspaces
- Abstaining mechanism reduces impact of non-competent classifiers

$$Kappa = \frac{n\sum_{i=1}^{c} x_{ii} - \sum_{i=1}^{c} x_{i.}x_{.i}}{n^2 - \sum_{i=1}^{c} x_{i.}x_{.i}} \cdot 100$$

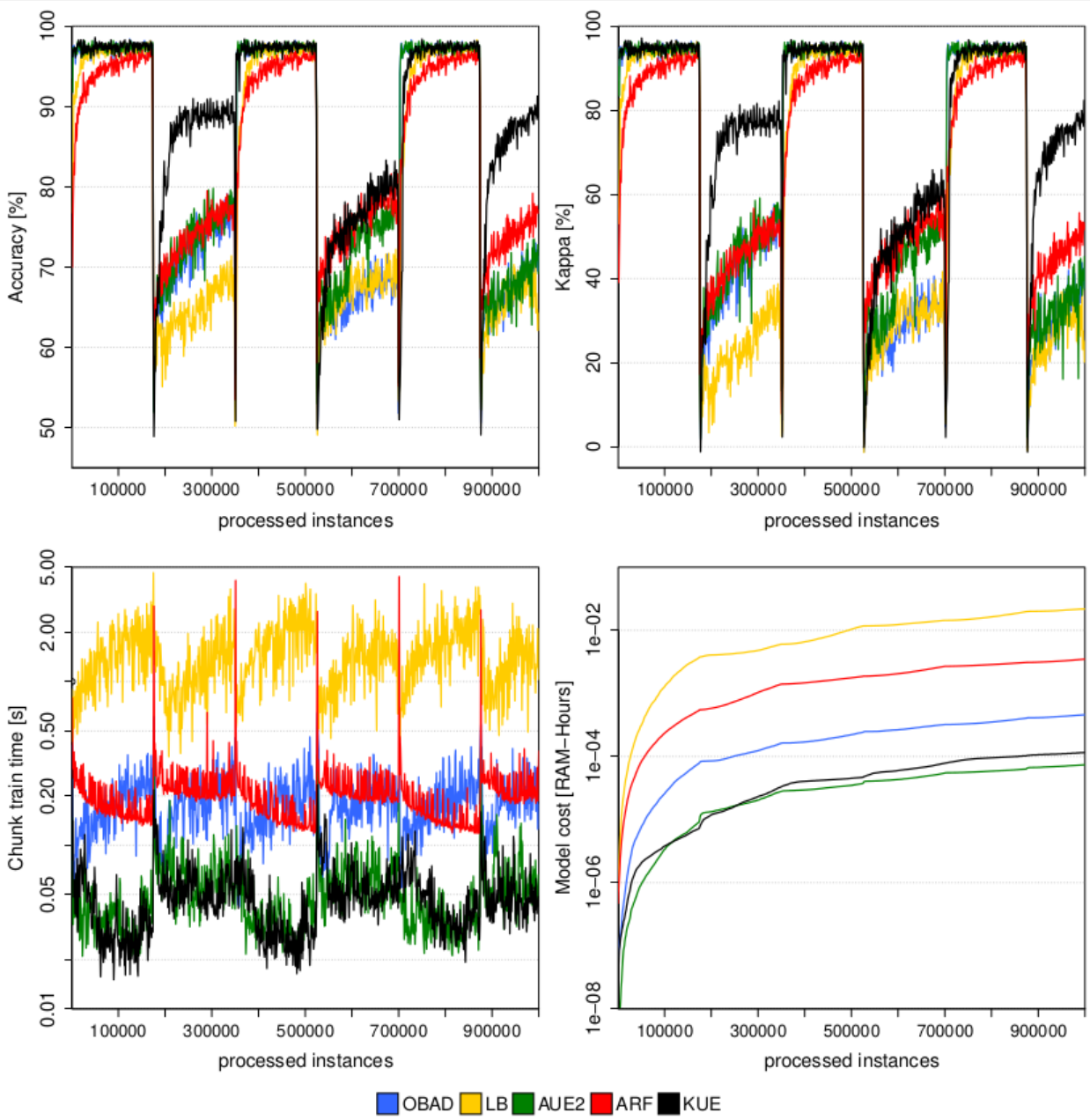
[2] A. Cano, B. Krawczyk: Kappa Updated Ensemble for Drifting Data Stream Mining. Machine Learning, In Press (2019)

Kappa Updated Ensemble

- Standard data streams

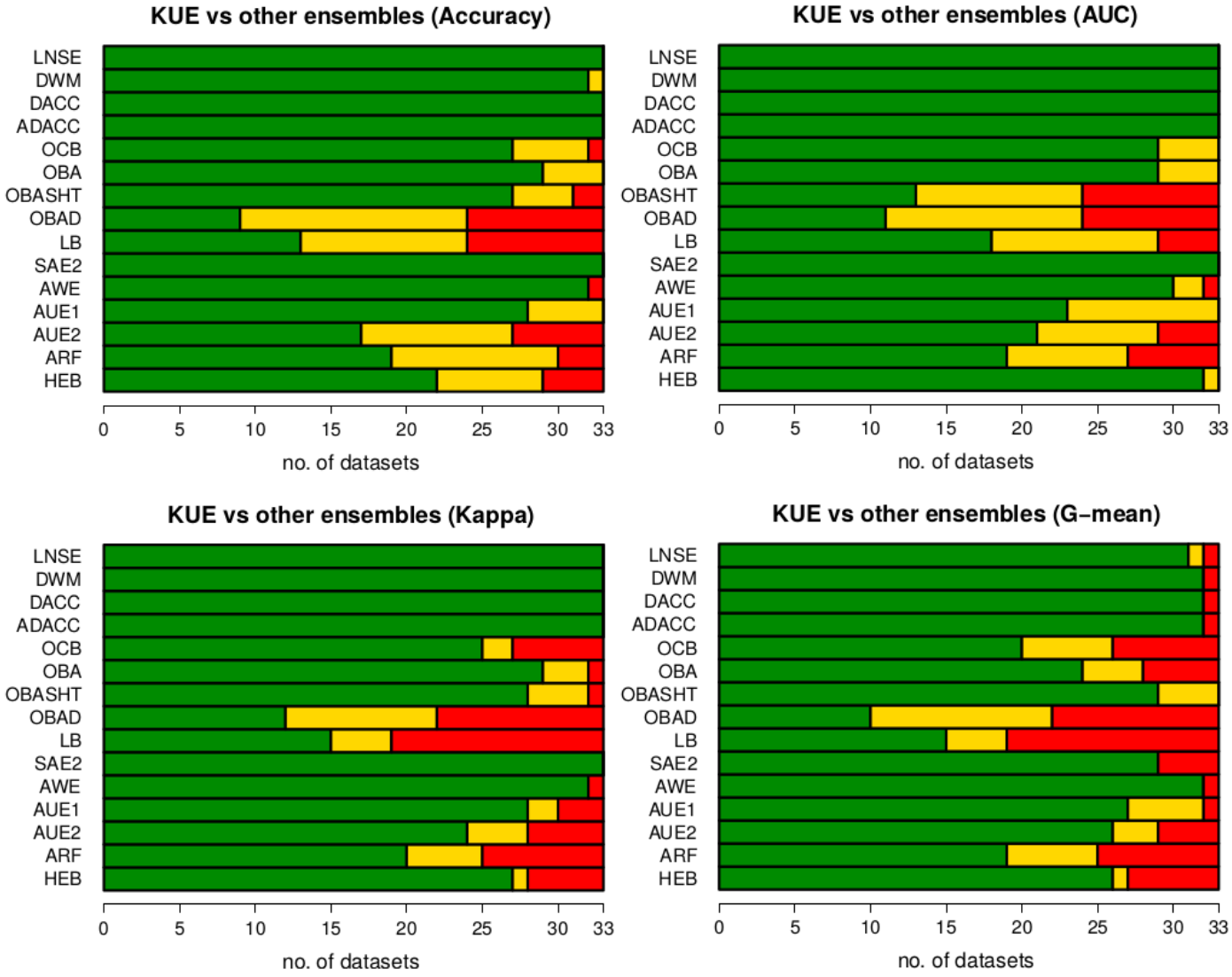


Kappa Updated Ensemble

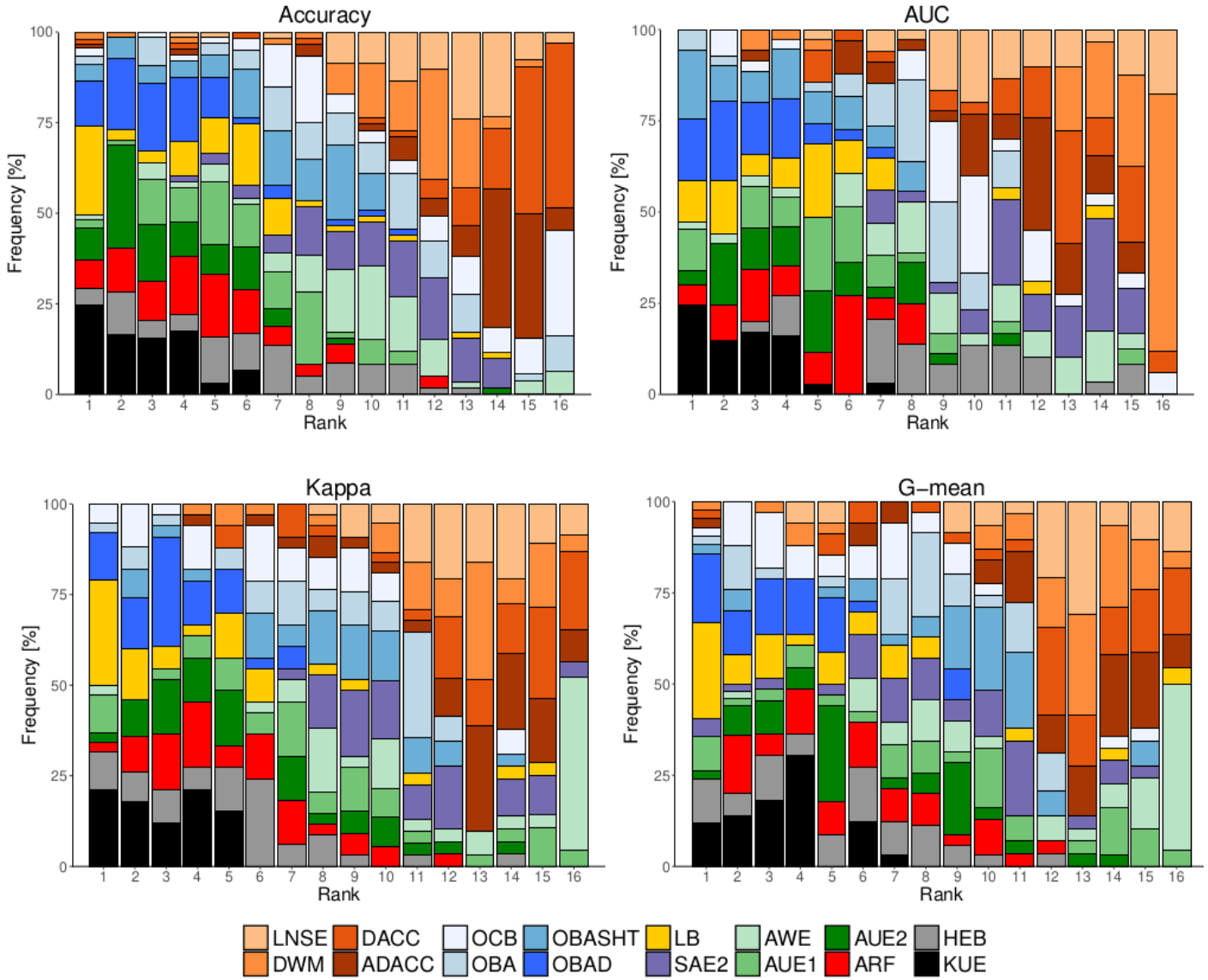


Kappa Updated Ensemble

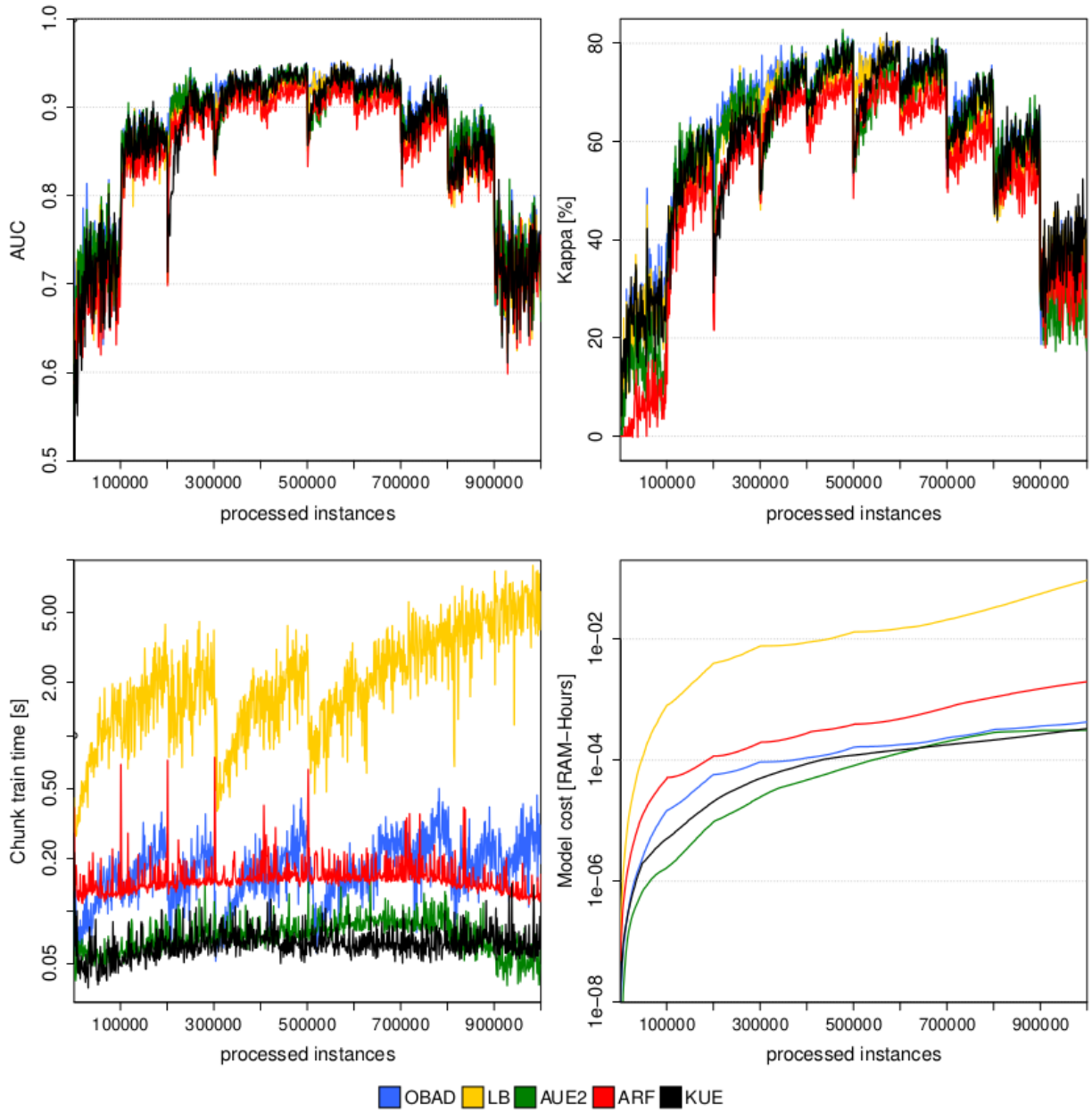
- Imbalanced data streams



Kappa Updated Ensemble

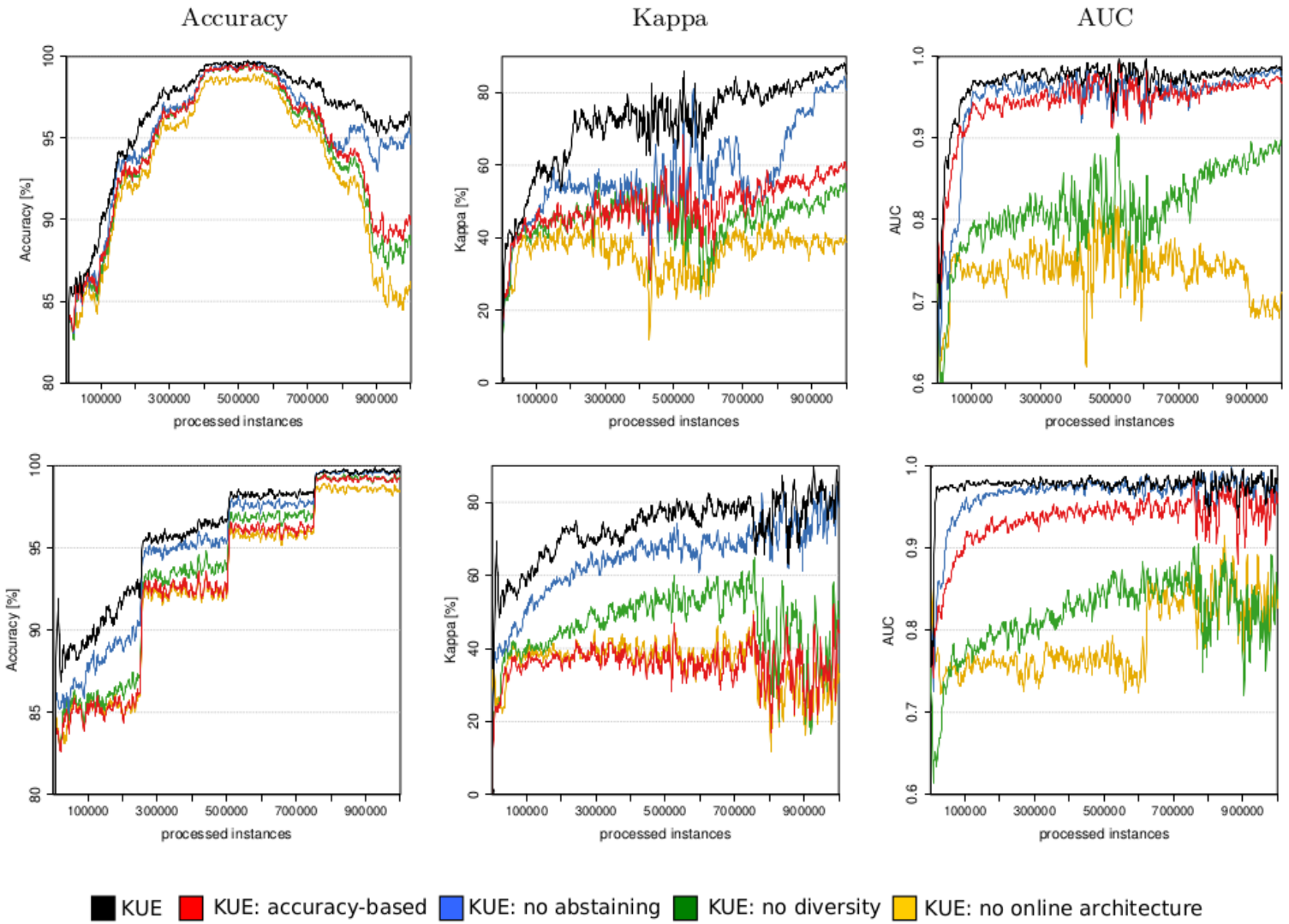


Kappa Updated Ensemble



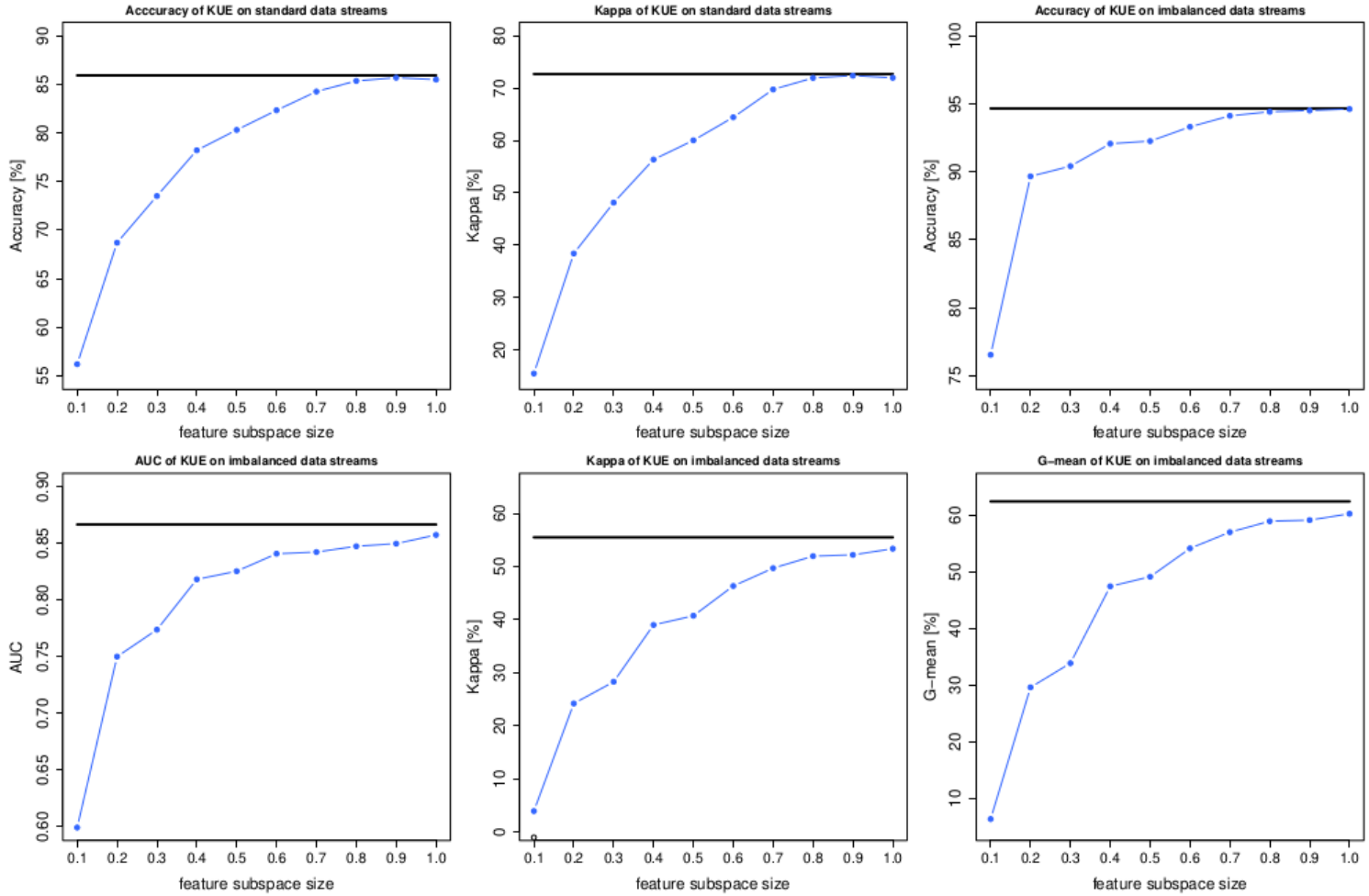
Kappa Updated Ensemble

- Contribution by individualized KUE mechanisms



Kappa Updated Ensemble

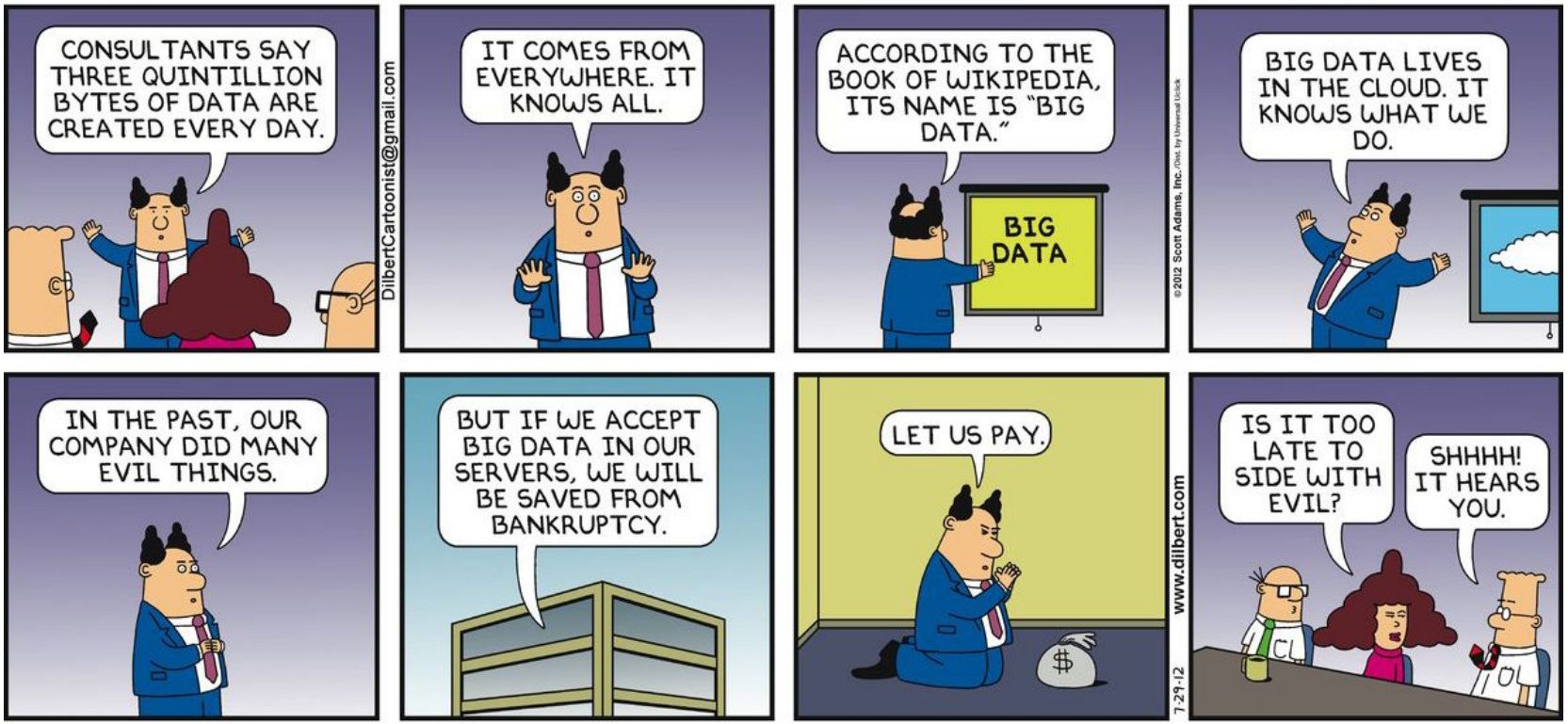
- Random feature subspaces vs fixed size feature subspace



Ending notes

Thank you for your attention! Q & A time!

Next: **Part 3: Limited access to ground truth in data streams**



Big Data Stream Mining

Part 3: Limited access to ground truth in data streams

Bartosz Krawczyk[1] Alberto Cano[1]

[1]Department of Computer Science
Virginia Commonwealth University
Richmond, VA
USA

{bkrawczyk,acano}@vcu.edu

Outline

Outline

Access to true class labels

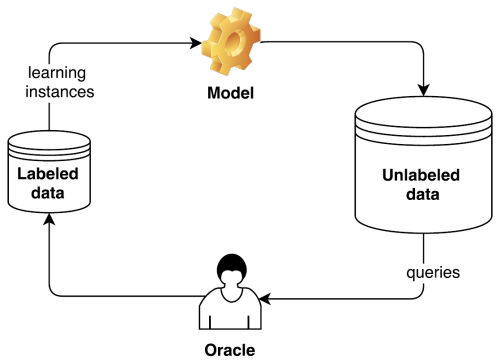
Most of the works done in data streams assume that true class labels are available for each example or batch of objects immediately after processing.

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.



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

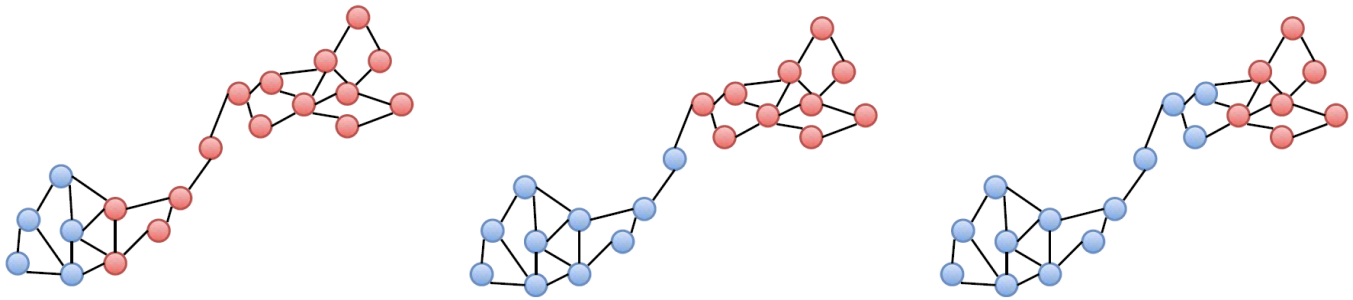
Semi-supervised learning assume that we have a small initial subset of labeled instances and large subset of unlabeled ones.

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

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

Outline

Active learning framework

Our active learning is guided by a 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

We propose to conduct 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.

Instead of pooling their decision using voting strategies (like in Query by Committee), we 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.

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

Multi-armed Bandit approach

We realize the continuous classifier selection for active learning via Multi-armed Bandit optimization.

Each classifier is treated as an individual machine that is being played to maximize a cumulative reward.

This is formulated as a regret function - difference between reward obtained using a selected strategy and a reward obtained using a hypothethical optimal strategy:

$$\min_s \mathscr{R}_s = \sum_{k=1}^{T} r_k^{opt} - \sum_{k=1}^{T} r_k^s \iff \max_s \sum_{k=1}^{T} r_k^s, \tag{7}$$

Therefore, choosing a proper reward function allows us to track the effectiveness of a classifier in guiding the active learning process.

Reward function

Most active learning algorithms are based on classifier's uncertainty - selecting instances that are close to current decision boundary.

This is not feasible for drifting data streams, as boundaries change dynamically in the presence of concept drift - e.g., new concept may appear in the region of high certainty.

We propose to measuring the increase in generalization capabilities of the classifier according to a metric m on a separate validation set V for each selected instance:

$$r_m(h_i, h_{i-1}, V)) = m(h_i(V), o(V)) - m(h_{i-1}(V), o(V)). \quad (8)$$

This allows us to measure how a given instance will increase the generalization capabilities of a given classifier.

Classifier that displays increased generalization capabilities is more likely to quickly adapt to concept drift.

Thus, it should be selected by Multi-armed Bandit algorithm to guide the current active learning query.

Outline

Used optimization strategy

We propose to select classifiers for guiding the active learning procedure based on their generalization capabilities.

In order to optimize the classifier selection in the proposed ensemble active learning approach, we need an efficient Multi-armed Bandit strategy.

Recent works point to Upper Confidence Bound (UCB1) as an effective tool for this task.

It approaches the minimal regret bound of $\Omega(\log T)$ when the constant variance of each bandit (in our case classifier) is assumed:

$$b = \arg \max_{l \in \{1, \cdots, L\}} \left(\bar{r}_l + \sqrt{\frac{2 \log T}{|P_l|}} \right). \tag{9}$$

Used optimization strategy

UCB1 is not suitable for drifting data streams, as one cannot assume an identical variance of each underlying classifier.

We propose to use a tuned version of UCB1 that takes into account individual variances of each bandit (classifier in our case):

$$b = \arg \max_{l \in \{1,\cdots,L\}} \left(\bar{r}_l + \sqrt{\frac{\log T}{|P_l|} \min \left(\frac{1}{4}, \underset{k \in P_j}{\text{var}}(r_k) + \sqrt{\frac{2 \log T}{|P_l|}} \right)} \right). \tag{10}$$

Outline

Practical considerations

Validation set: used ensemble classifiers must be capable of evaluating the generalization metric. In practice, this can be obtained from any streaming ensemble classifier (out-of-bag instances or different chunks).

Classifier outputs: EAL-MAB requires for the base classifiers in ensemble to return continuous outputs (e.g., support functions) and not discreet labels. In practice, this is realized by most of online / streaming single classifiers.

Usage of labeling budget: EAL-MAB runs on each new chunk of data for T iterations to select instances, one per iteration. Thus, the given budget B for a window size of ω is equal to the number of iterations that EAL-MAB will perform: $T = B \times \omega$.

Usage of metric m: EAL-MAB may use any metric suitable for data streams. We propose to use prequential accuracy.

Results according to prequential accuracy

Comparison of EAL-MAB and reference active learning algorithms over different ensemble architectures and base classifiers over 84 cases (12 benchmark datasets and 7 different budgets).

A tie was considered when McNemar's test rejected the significance of difference between tested algorithms.

Adaptation to concept drift

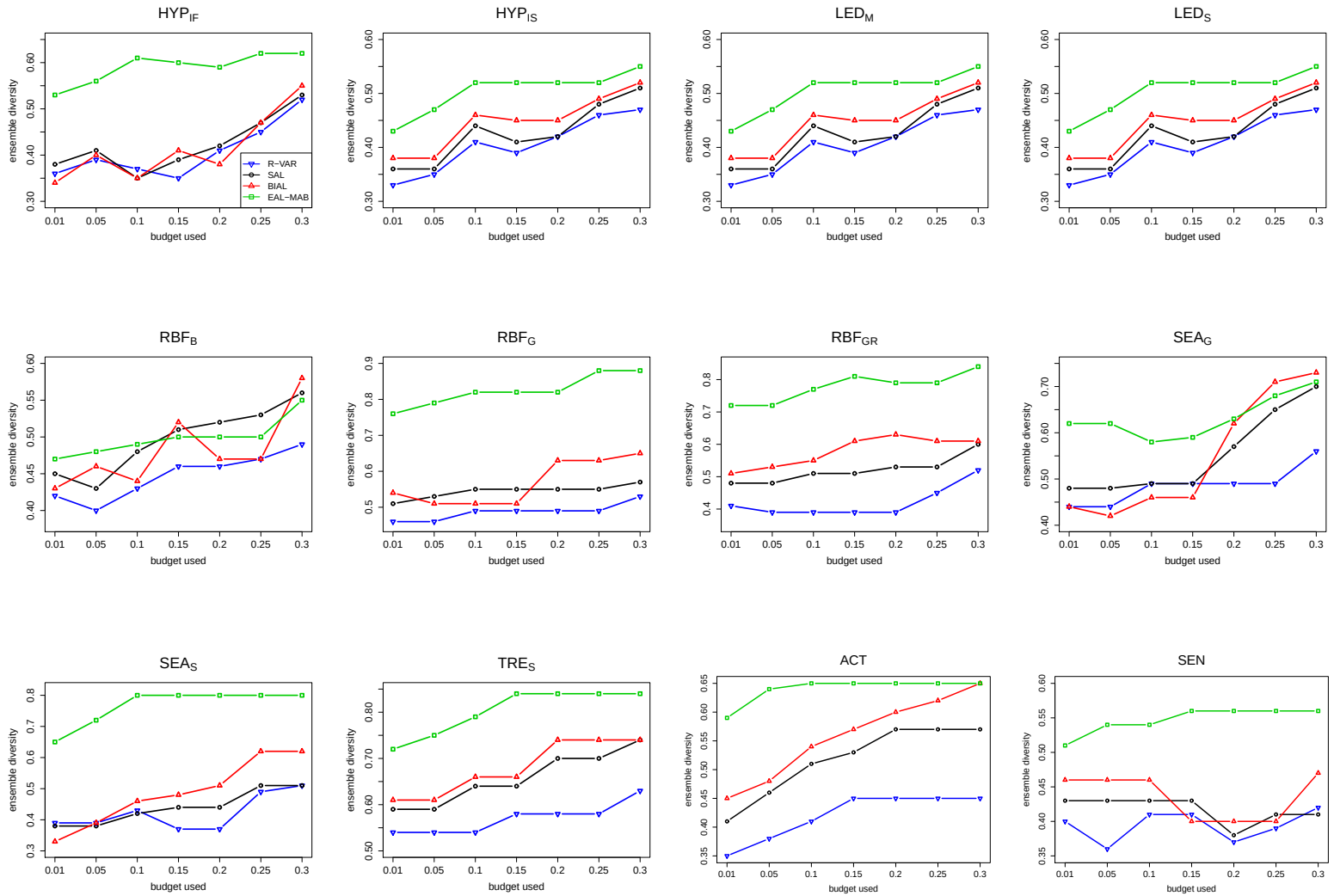
We measure percentage of drifting instances that were selected for label query by active learning algorithms.

Instances from the new concept (after drift) should be queried most frequently in order to maximize the classifier adaptation.

Dataset	R-VAR	SAL	BIAL	EAL-MAB
HYP_{IF}	17.23±5.21	19.54 ±4.12	20.46± 4.51	**26.12**±3.18
HYP_{IS}	18.65± 4.26	22.54±3.95	21.89±4.26	**28.81**±3.52
LED_{M}	32.73±2.19	38.45±3.11	39.99±3.82	**43.26**±3.18
LED_{S}	27.41±1.86	29.45±2.11	29.88± 3.28	**33.47**±1.68
RBF_{B}	21.09±2.76	24.98±2.98	**29.72**±3.07	26.54±3.01
RBF_{G}	36.44±4.98	38.72±6.11	40.07±5.28	**45.28**±5.39
RBF_{GR}	38.56±6.21	40.03± 7.01	41.13±6.38	**47.20**±6.94
SEA_{G}	11.87±3.98	17.43±2.51	**18.82**±2.99	15.82±2.32
SEA_{S}	10.02±7.32	15.77±6.21	16.61±5.84	**25.06**±5.11
TRE_{S}	38.23±4.98	31.44±2.66	32.80±2.29	**43.19**±3.36
ACT	–	–	–	–
SEN	–	–	–	–

Diversity analysis

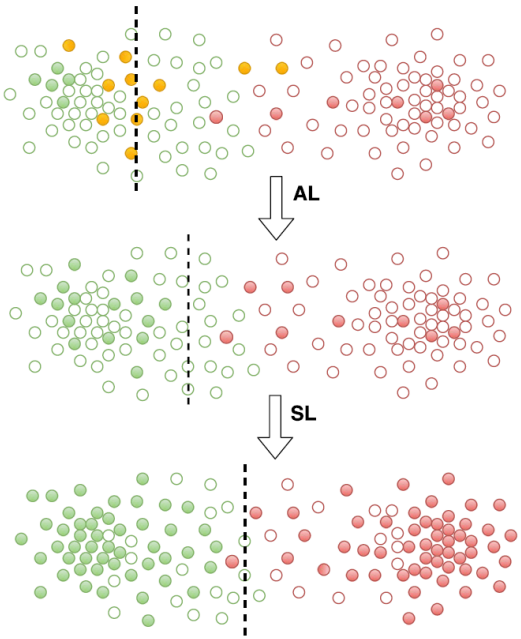
We measure diversity of ensembles measured with kappa interrater agreement metric with respect to varying budget sizes



Outline

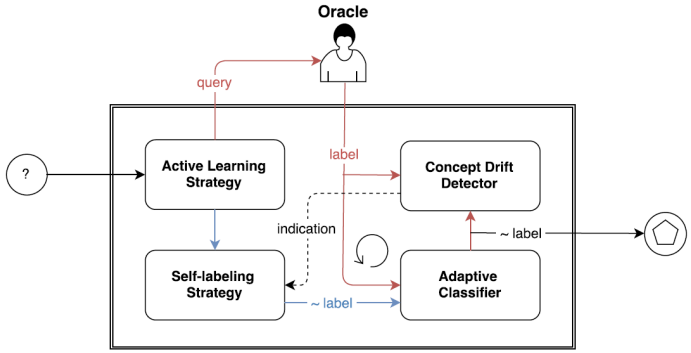
Motivation

- 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

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



- Seven strategies for 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

Continuous DDM strategy

- DDM assumes that changes can be detected by tracking the actual error rate p along with its standard deviation s and comparing it with the registered error for the stable period. The algorithm makes decisions based on the condition:

$$p + s > p_{min} + \alpha s_{min}, \tag{11}$$

  where p_{min} and s_{min} are the mean error and its standard deviation registered for a stable concept after at least 30 samples. The α parameter is used to determine thresholds for *warning* ($\alpha = 2$, the confidence interval is 95%) and *change* ($\alpha = 3$, the confidence interval is 99%) states
- We simply extract the tracked, continuous error measure $\varepsilon = p + s$.
- The threshold should be higher during a concept drift and lower during a stable period:

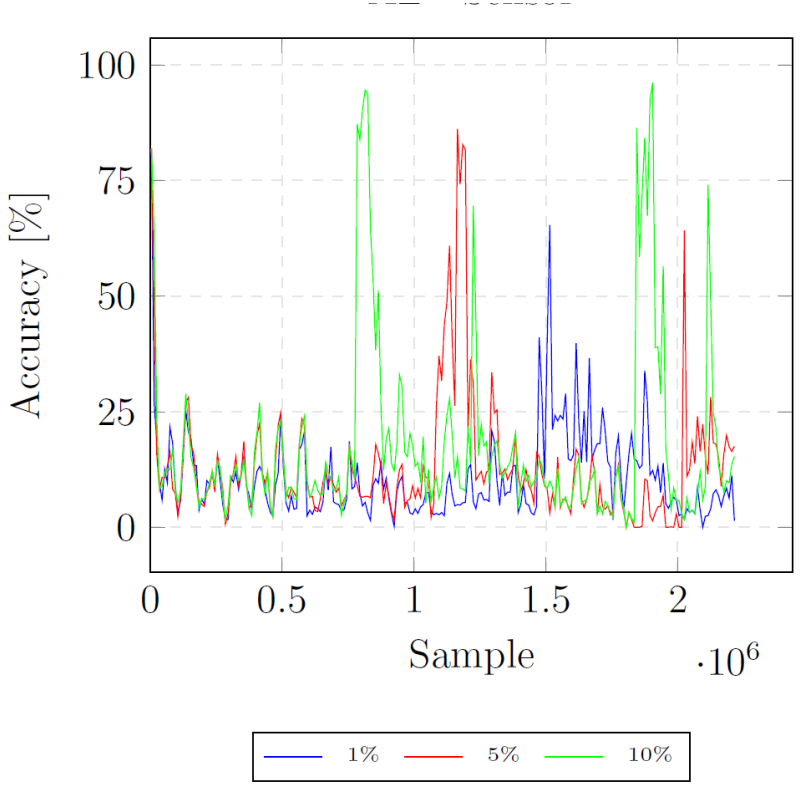
$$p(\hat{y}|X) \geq \tanh 2(\varepsilon + \frac{1}{c}). \tag{12}$$

- We add $1/c$ to additionally penalize a situation when a classifier simply guesses labels for $\varepsilon = 1 - 1/c$.

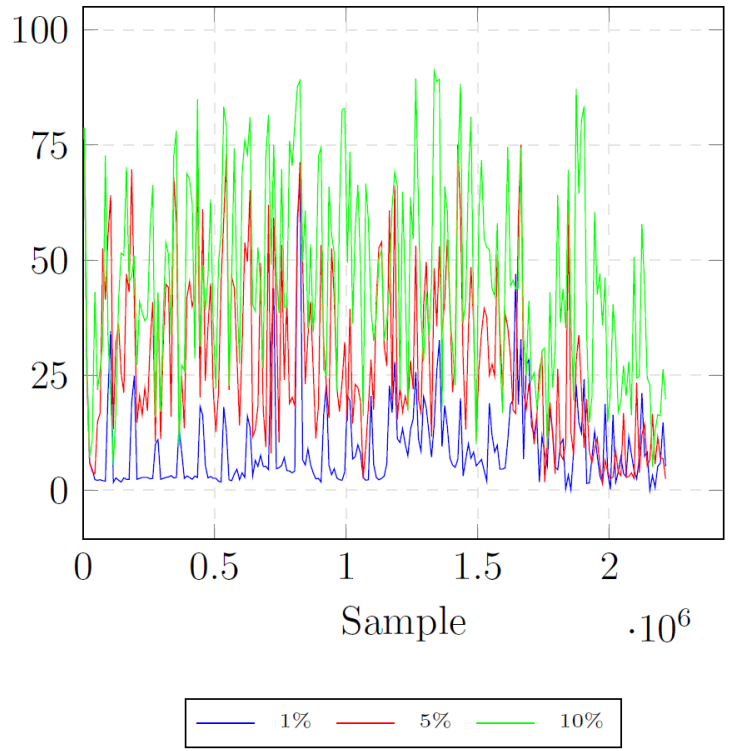
Hybrid self-labeling ensembles

- Our hybrid solution can be easily incorporated into an ensemble learning scheme.
- For active learning part we incorporate our previously discussed online Query by Committee solution that uses online Bagging and our classifier update strategy.
- While active learning is based on collective decision of classifiers, we propose to assign a self-labeling module to each base learner independently.
- This allows to efficiently utilize and maintain the diversity of base models, as each classifier uses different subset of instances that in turn will lead to different self-labeling outcomes.
- We add a continuous pruning of weakest subset of learners to avoid situations where classifiers propagate self-labeling errors.

Hybrid self-labeling ensembles - results

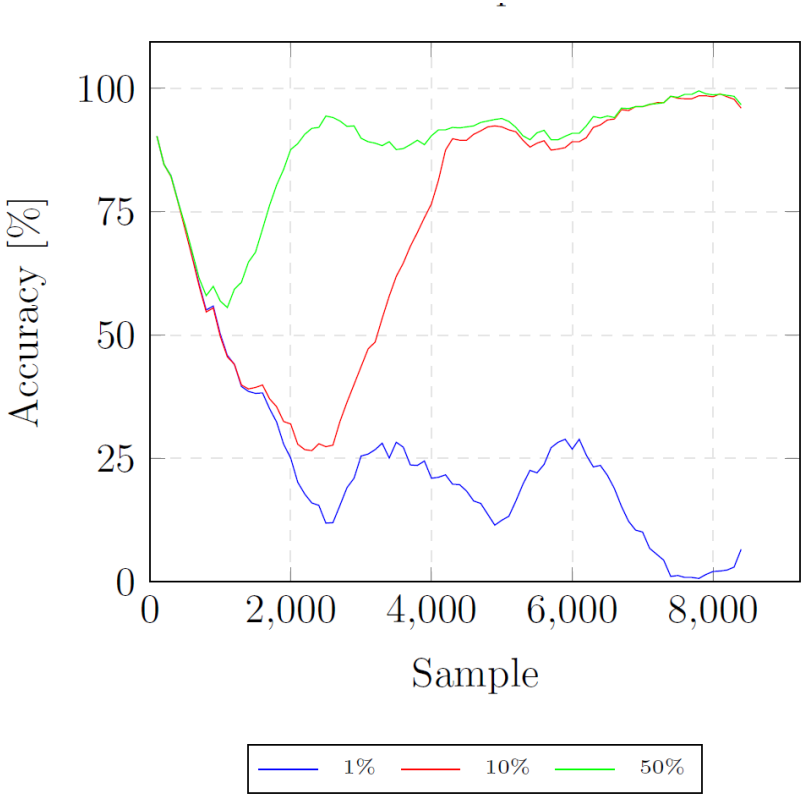


(a) Sensors - QBC

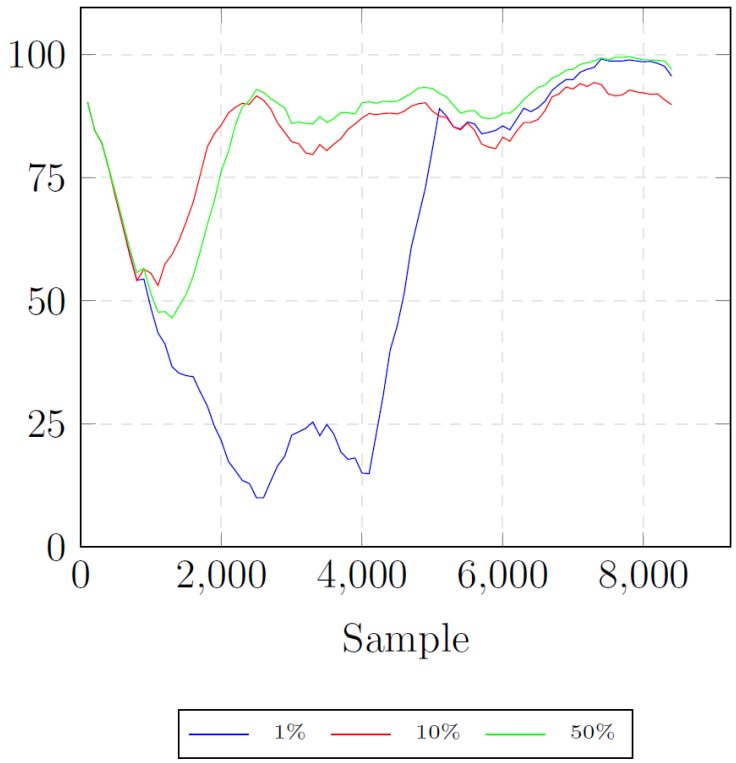


(b) Sensors - hybrid

Hybrid self-labeling ensembles - results



(a) SPAM - QBC



(b) SPAM - hybrid

Ending notes

Q & A time!

Next: **Part 4: Advanced problems and open challenges in data stream mining.**

Big Data Stream Mining

Part 4: Advanced problems

Bartosz Krawczyk[1] Alberto Cano[1]

[1]Department of Computer Science
Virginia Commonwealth University
Richmond, VA
USA

{bkrawczyk,acano}@vcu.edu

Outline

1. Evolutionary algorithms for drifting data streams
2. Multi-label data streams
3. Open challenges and future directions

Outline

Evolutionary algorithms for drifting data streams

- Evolutionary algorithms were traditionally perceived to be too slow for real-time data streams
- Advances in high-performance computing architectures (GPUs and MapReduce) now allow fast and efficient distributed computing
- Evolutionary algorithms are intrinsically parallel and easy to speed up
- Evolutionary algorithms are designed to evolve solutions to fit the objective function. Self-adapting heuristic to model concept drift.
- Genetic Programming evolves a population of trees that can represent interpretable classification rules describing the stream
- Concept drift may be assessed by tracking how the classification rules change to reflect changes in the data properties

Genetic Programming on GPUs for Drifting Data Streams

Evolving Rule-Based Classifiers with Genetic Programming on GPUs for Drifting Data Streams[1]

Main contributions:

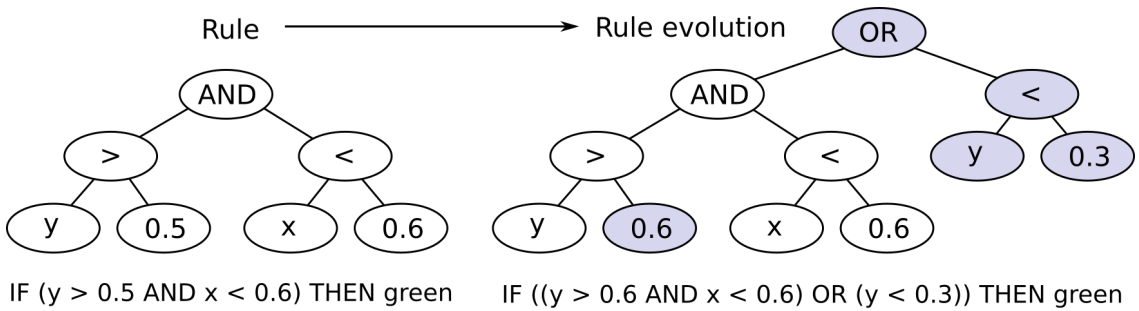
- Exploit of genetic programming for automatic rule adaptation to stream changes with no need for explicit drift detection
- Rule diversification and stream sampling strategies to allow for both fast adaptation and maintaining previously learned knowledge
- Efficient implementation on GPUs for obtaining competitive runtimes on data streams
- Learning from partially labeled data streams with very limited access to ground truth

[1] A. Cano and B. Krawczyk: Evolving Rule-Based Classifiers with Genetic Programming on GPUs for Drifting Data Streams. Pattern Recognition, vol. 87, 248-268 (2019)

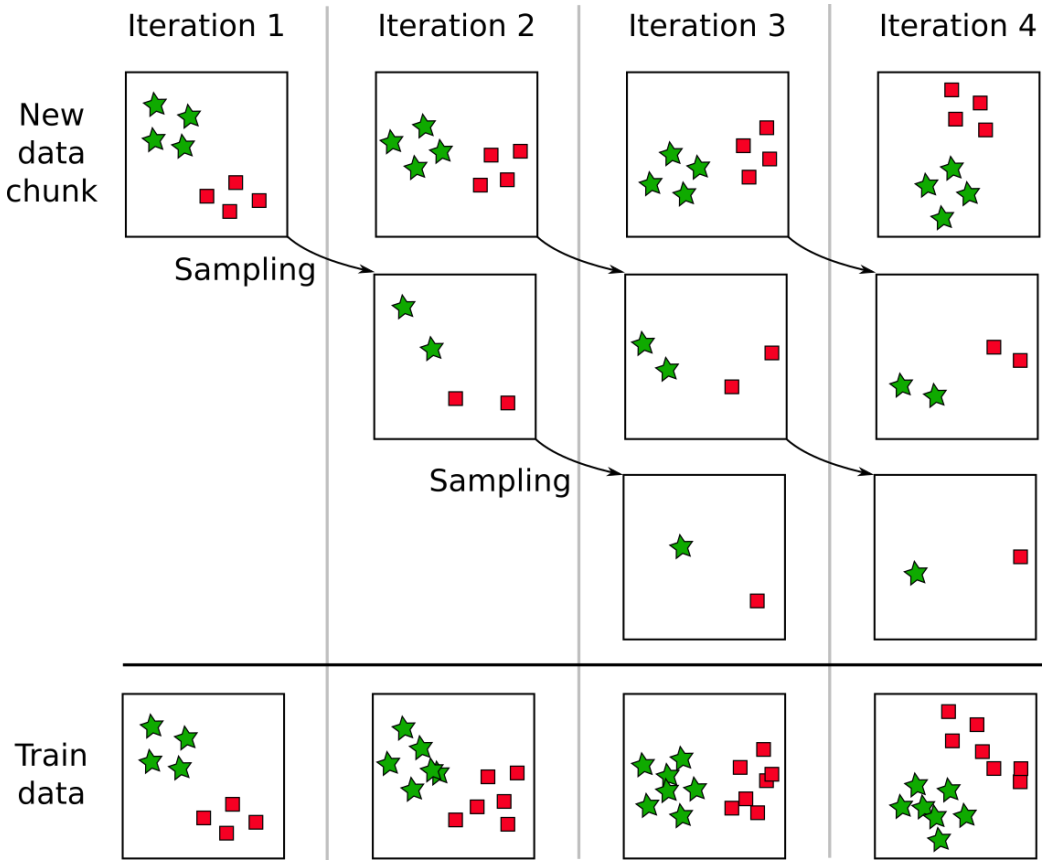
- Context-free grammar to generate classification rules

$G = (V_N, V_T, P, S)$
$V_N = \{Comparison,\ Operator,\ Attribute,\ Value\}$
$V_T = \{AND,\ OR,\ NOT,\ <,\ >,\ =,\ \neq,\ attributes,\ values\}$
$P = \{\langle S\rangle \rightarrow AND\ \langle S\rangle\ \langle Comparison\rangle$
$\langle S\rangle \rightarrow OR\ \langle S\rangle\ \langle Comparison\rangle$
$\langle S\rangle \rightarrow NOT\ \langle S\rangle$
$\langle Comparison\rangle \rightarrow \langle Operator\rangle\ \langle Attribute\rangle\ \langle Value\rangle$
$\langle Operator\rangle \rightarrow > \mid < \mid = \mid \neq$
$\langle Attribute\rangle \rightarrow$ *random attribute in dataset's features*
$\langle Value\rangle \rightarrow$ *random value within attribute's valid domain*
$\}$

- Genetic operators: crossover and mutation



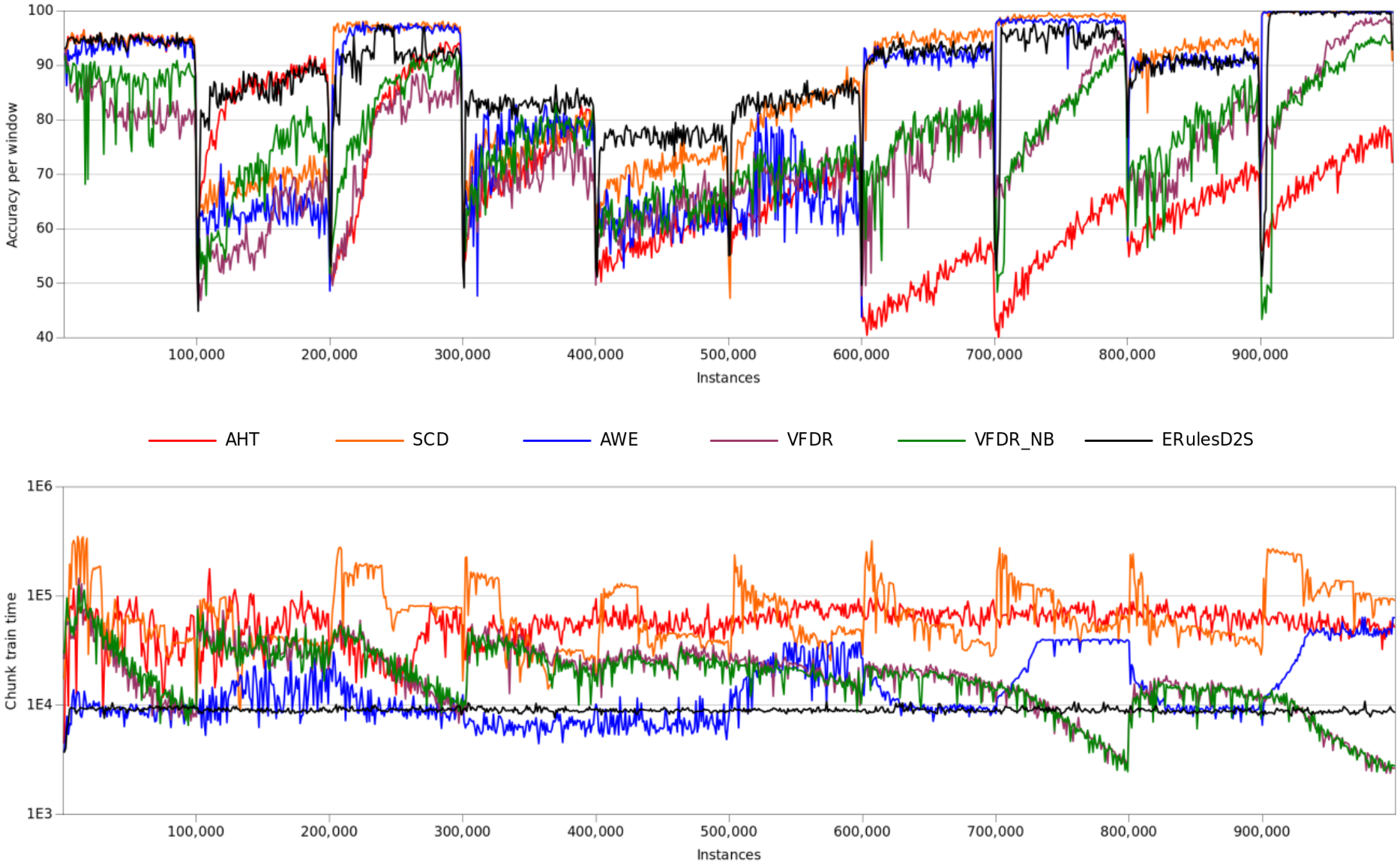
- Sampling sliding window



- Parameter configuration: accuracy and runtime

Windows	Sampling factor	Rules	Pop	Gen	Accuracy	Train Time	Test Time	RAM Hours
10	0.25	10	25	50	82.36	0.663	0.038	4.7E-4
10	0.5	10	25	50	82.41	0.656	0.036	3.2E-4
10	0.25	5	25	50	81.91	0.356	0.021	2.3E-4
10	0.5	5	25	50	82.08	0.350	0.018	1.6E-4
5	0.25	10	25	50	82.37	0.578	0.034	3.9E-4
5	0.5	10	25	50	82.59	0.596	0.034	2.8E-4
5	0.25	5	25	50	82.22	0.343	0.017	2.1E-4
5	**0.5**	**5**	**25**	**50**	**82.25**	**0.338**	**0.020**	**1.8E-4**
5	0.5	5	50	50	82.28	0.651	0.020	4.2E-4
5	0.5	5	15	25	81.66	0.208	0.017	6.4E-5
5	0.5	10	15	25	82.20	0.384	0.034	1.6E-4
5	0.5	3	25	50	81.38	0.199	0.010	7.4E-5

Genetic Programming on GPUs for Drifting Data Streams

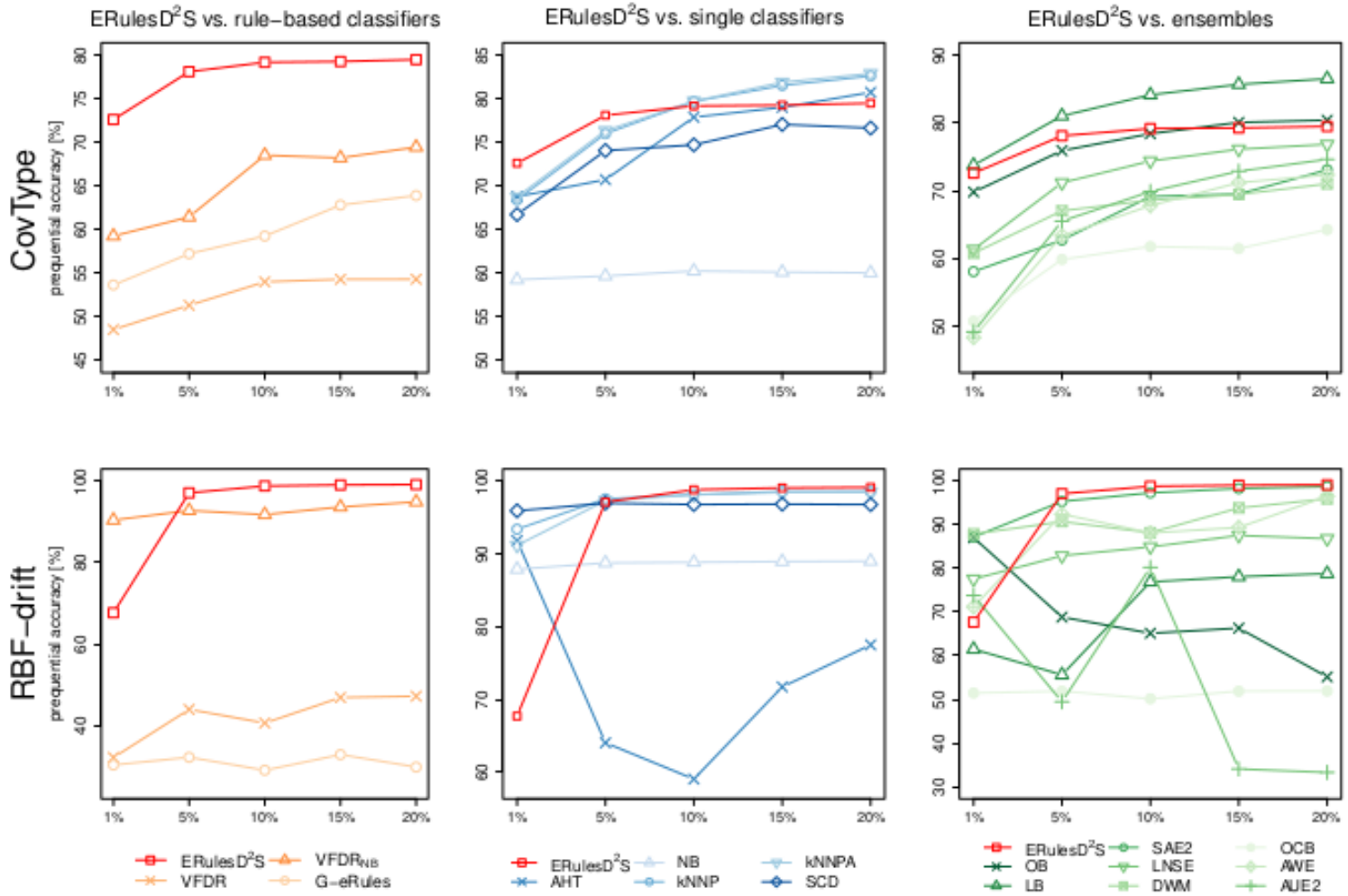


- Accuracy and complexity of the rule base

Dataset	Atts	Number of rules			Number of conditions			Accuracy			
		ERulesD^{2}S	VFDR	G-eRules	ERulesD^{2}S	VFDR	G-eRules	ERulesD^{2}S	VFDR	VFDR$_{NB}$	G-eRules
RBF	10	10	392	28	70	219	54	83.49	77.53	81.71	53.68
	100	10	142	34	70	503	67	98.34	87.10	97.78	58.69
	1000	10	342	55	74	884	109	99.97	77.04	99.28	59.03
	10000	10	368	78	69	2328	143	99.97	59.83	86.97	56.64
RBF-drift	10	20	184	94	140	45	185	77.63	58.58	76.42	31.13
	100	20	244	110	140	251	184	98.63	63.34	96.18	33.86
	1000	20	552	188	149	723	291	99.48	50.60	98.81	35.50
	10000	20	814	95	147	1282	190	99.66	30.55	87.45	43.18
HP-drift-n	10	20	73	21	88	106	42	82.91	77.39	83.83	49.95
	100	20	56	16	88	260	30	80.14	76.63	82.17	50.05
	1000	20	273	19	86	467	32	82.22	69.86	73.94	49.96
	10000	20	338	13	87	853	24	75.73	48.00	47.60	49.85
RT-drift	10	20	114	572	140	262	907	58.85	44.35	58.10	48.38
	100	20	107	397	139	365	1164	49.78	39.47	46.45	42.47
	1000	20	320	314	157	516	937	55.21	55.32	57.00	34.00
	10000	20	369	309	163	1356	1409	43.34	36.80	16.69	31.11

Genetic Programming on GPUs for Drifting Data Streams

- Partially labeled data streams (1%, 5%, 10%, 15%, 20%)



Outline

Multi-label data streams

- Data may simultaneously be associated to multiple labels $Y \in \{0,1\}^{|L|}$

$$\mathrm{x} = (x_1, \ldots, x_D) \rightarrow \mathrm{y} = (y_1, \ldots, y_L)$$

- Concept drift may also happen in the distributions of the labelsets
- Label cardinality, density, and sparsity become an issue
- Problem transformation
  - Binary Relevance: decompose into L binary classification problems
  - Label Powerset: transform into a 2^L multi-class classification problem
- Algorithm adaptation

MLSAMPKNN

Multi-label punitive kNN with self-adjusting memory for drifting streams[2]

Main contributions:

- Self-adjusting window for varying forms of concept drift
- Punitive system to identify and remove instances negatively impact the classifier
- Computationally efficient nearest neighbor search
- Robustness to label noise and label imbalance

[2] M. Roseberry, B. Krawczyk, and A. Cano: Multi-label Punitive kNN with Self-Adjusting Memory for Drifting Data Streams. ACM Transactions on Knowledge Discovery from Data, In Press (2019)

MLSAMPKNN

- Self-adjusting memory

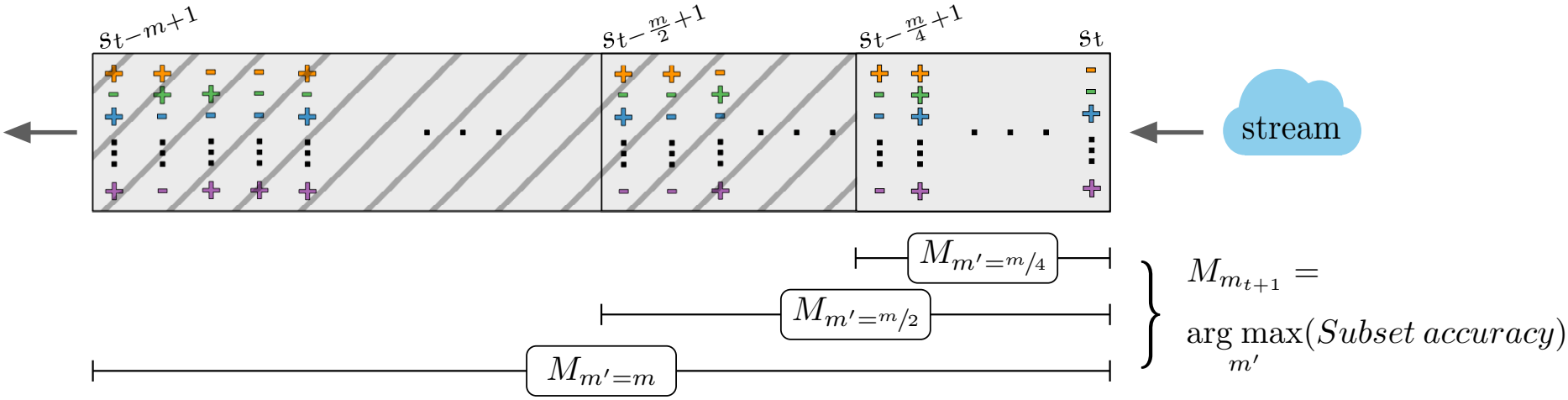
At time t, the window M contains m instances, formally:

$$M_m = \{s_{t-m+1}, \ldots, s_t\}$$

Several different sized windows $M_{m'}$ where $m' \leq m$ are evaluated based on their subset accuracy (exact match of all labels), formally:

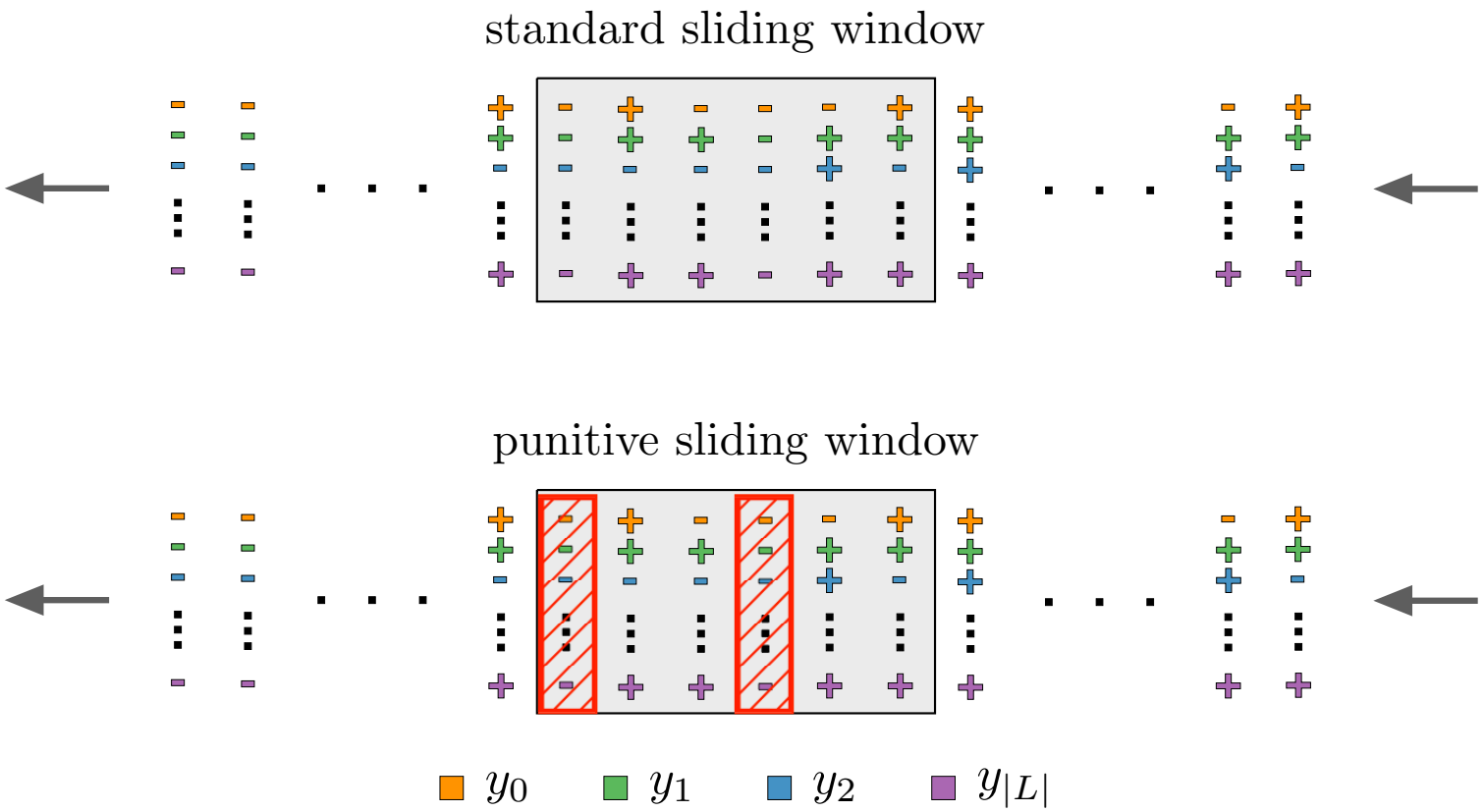
$$Subset\ accuracy = \frac{1}{m'} \sum_{i=0}^{m'} \mathbb{1} \mid Y_i = Z_i$$

The window $M_{m'}$ with the highest subset accuracy is used going forward.



MLSAMPKNN

- Punitive removal: keeps record of the errors made by each instance and removes any instance with errors exceeding a given threshold



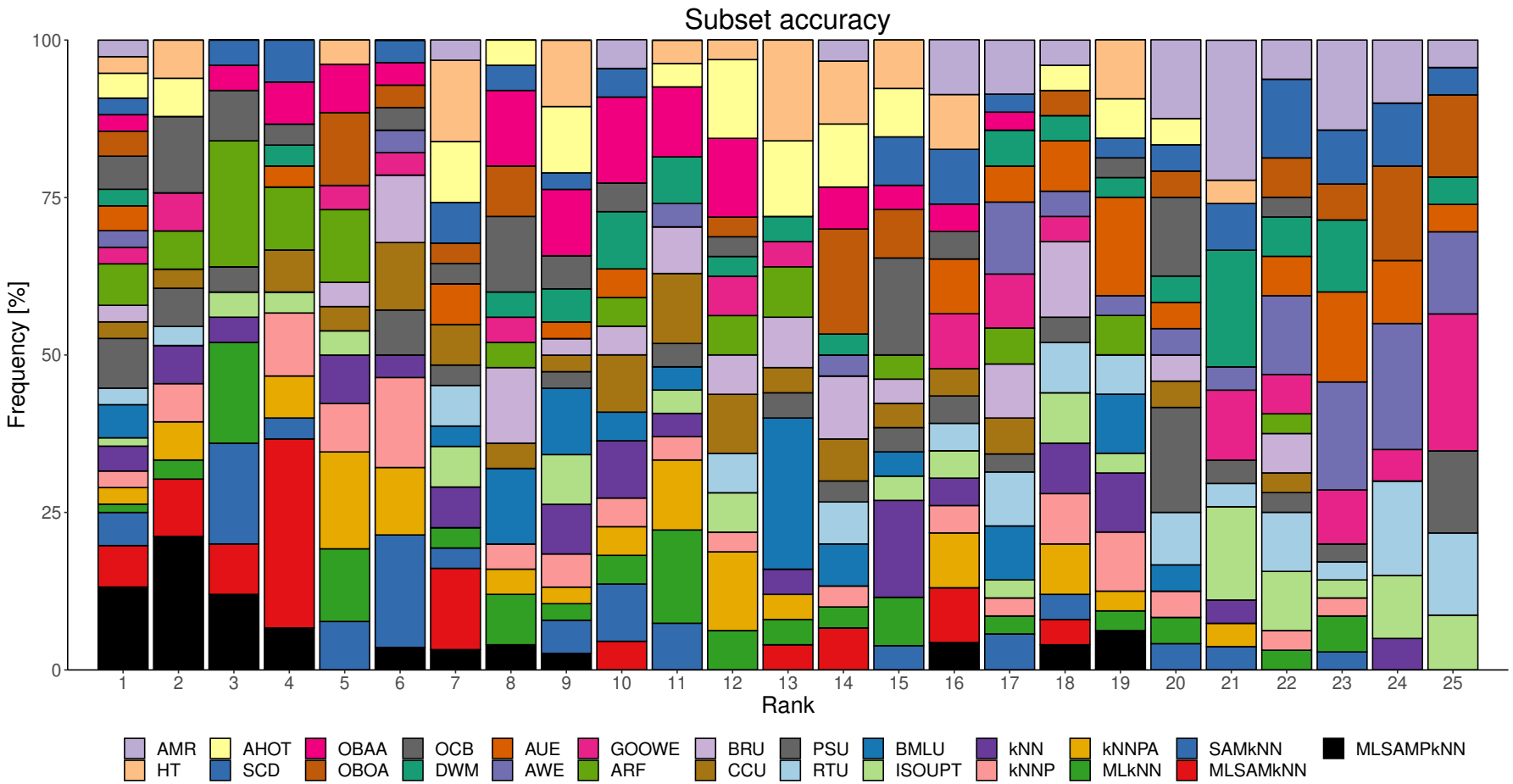
MLSAMPKNN

- Sensitivity analysis: influence of the punitive penalty and the number of neighbors

Subset accuracy	$k = 3$	$k = 5$	$k = 10$	F-measure	$k = 3$	$k = 5$	$k = 10$
p = 1	**0.3557**	0.3433	0.3356	p = 1	**0.4319**	0.4150	0.4094
p = 2	0.3493	0.3469	0.3311	p = 2	0.4221	0.4164	0.4064
p = 3	0.3472	0.3457	0.3314	p = 3	0.4194	0.4131	0.4072

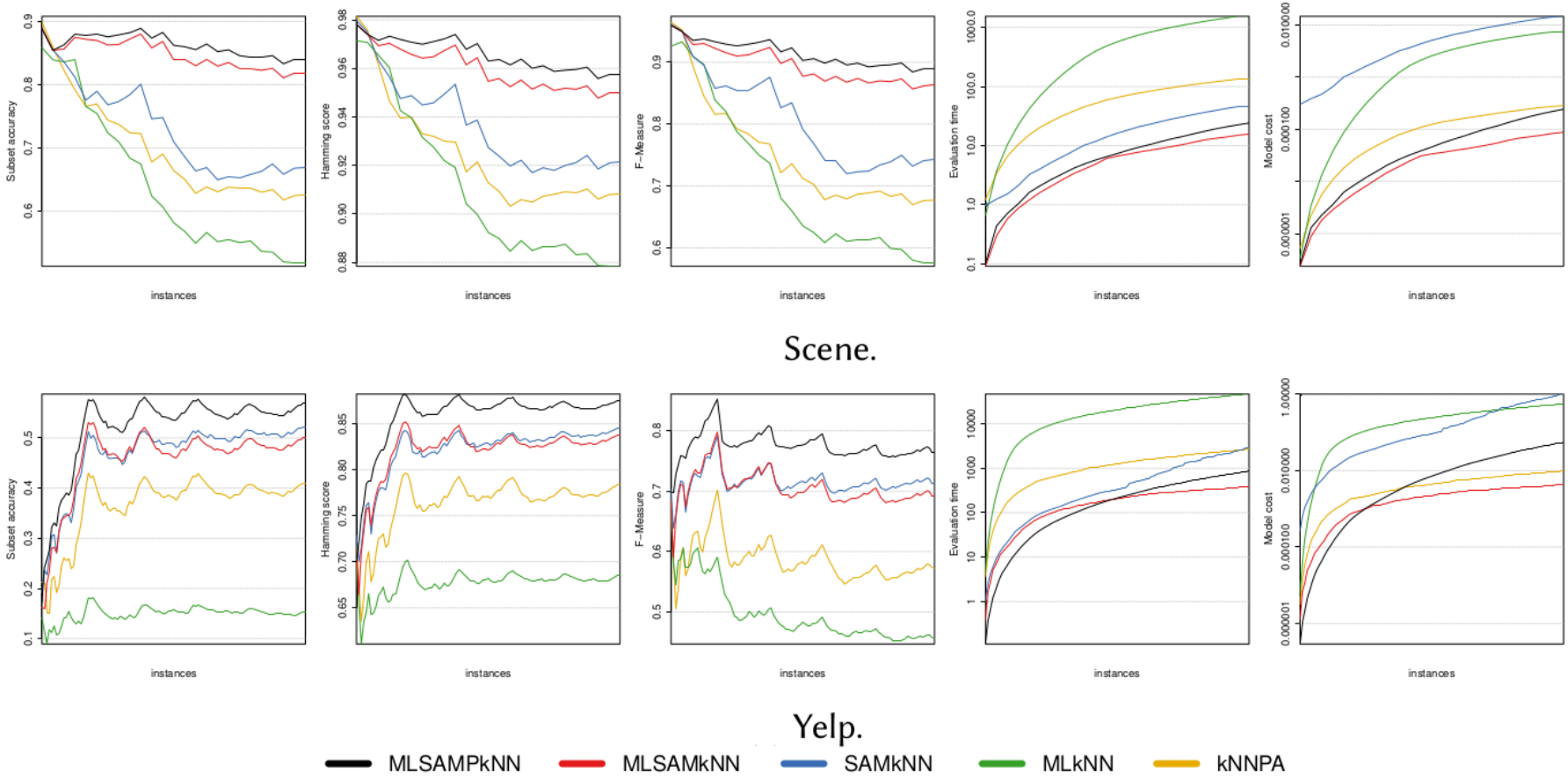
MLSAMPKNN

- Distribution of the algorithm ranks

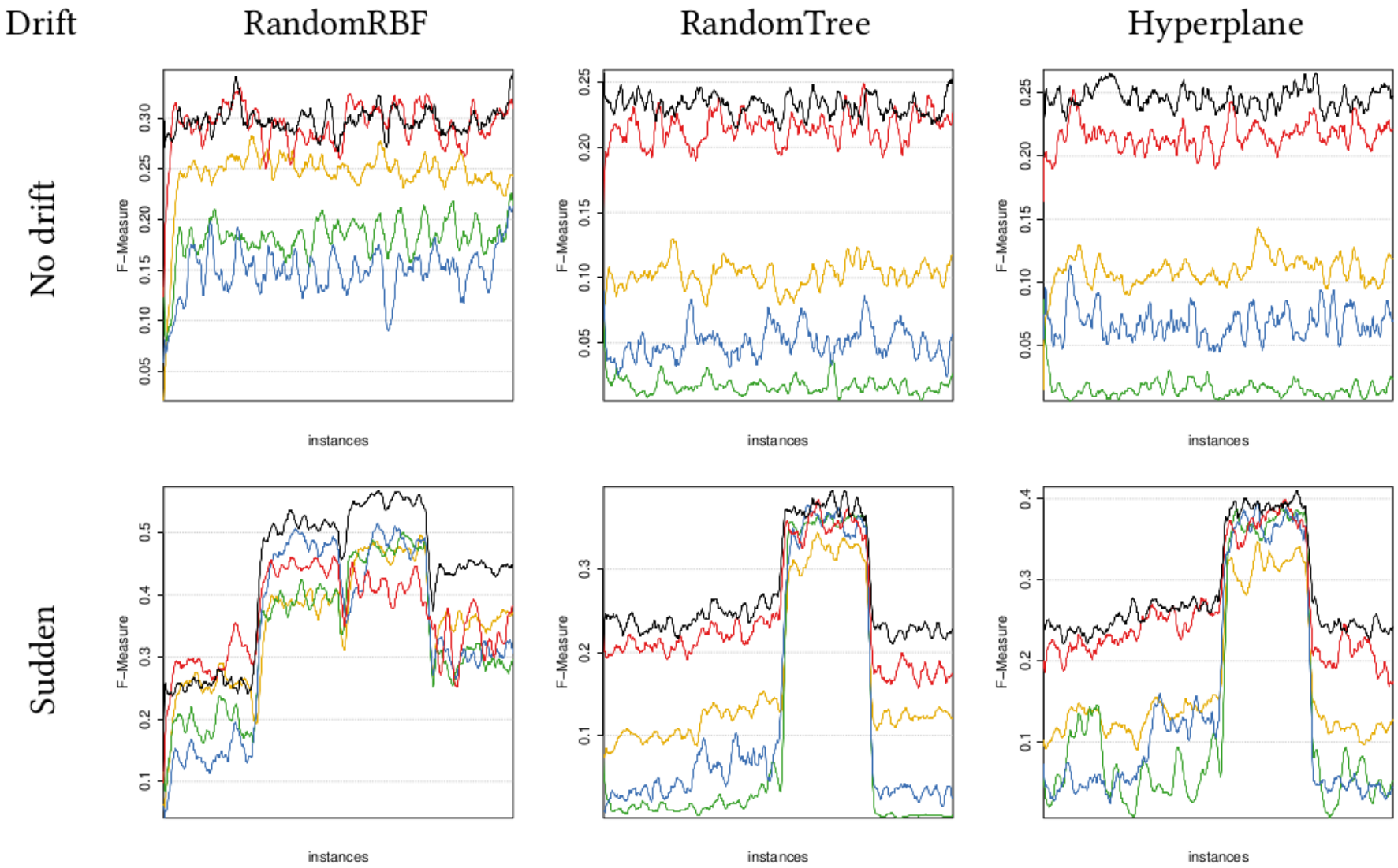


MLSAMPKNN

- Comparison of KNN-based methods

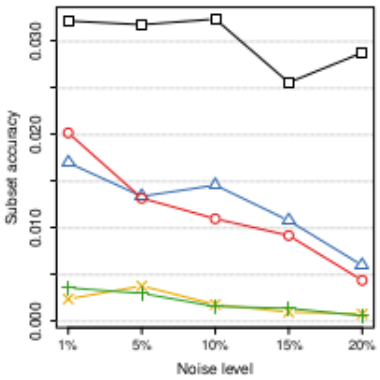


MLSAMPKNN

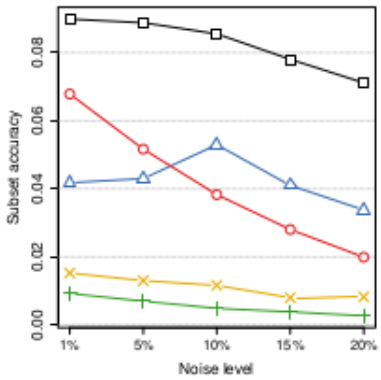

Drift
RandomRBF
RandomTree
Hyperplane
No drift
Sudden
F−Measure
instances

MLSAMPKNN

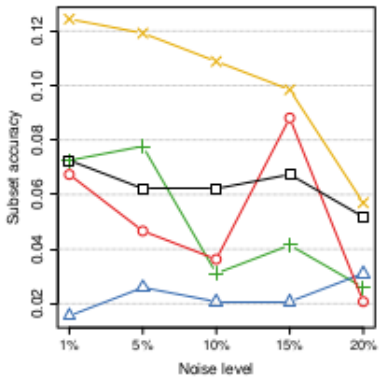
- Robustness to noise in labels (1%, 5%, 10%, 15%, 20%)



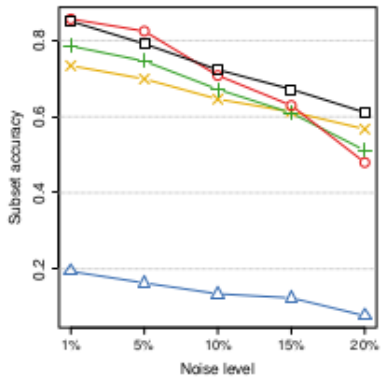
(a) Corel5k.



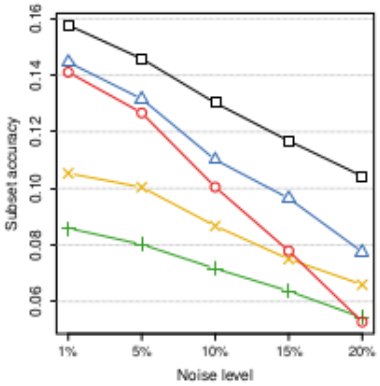
(b) Corel16k.



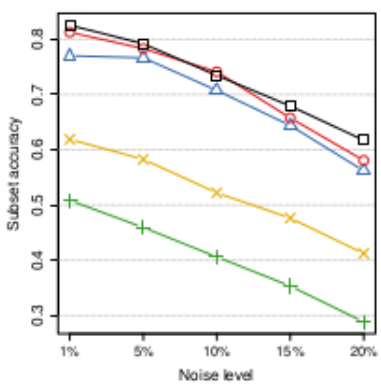
(c) Flags.



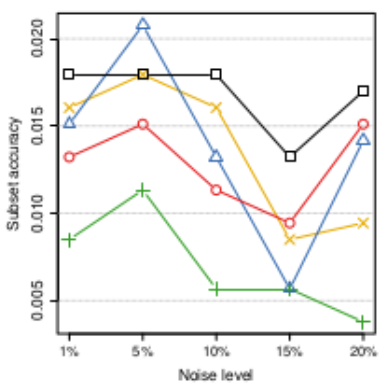
(d) Genbase.



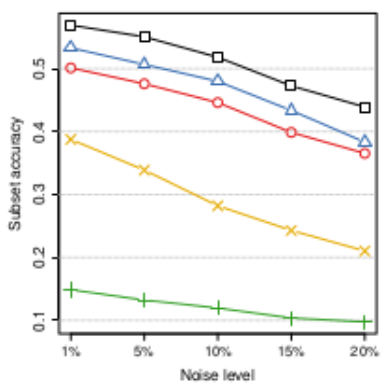
(e) Mediamill.



(f) Scene.



(g) Water-Qual.

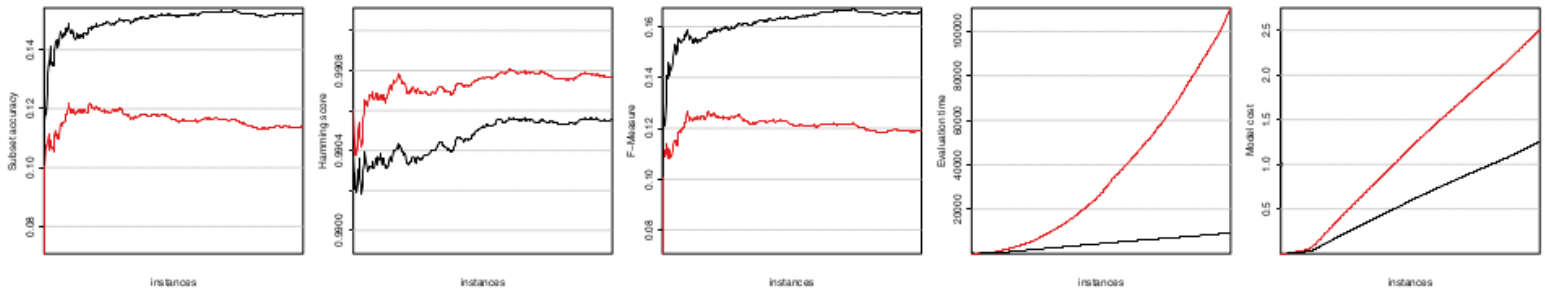


(h) Yelp.

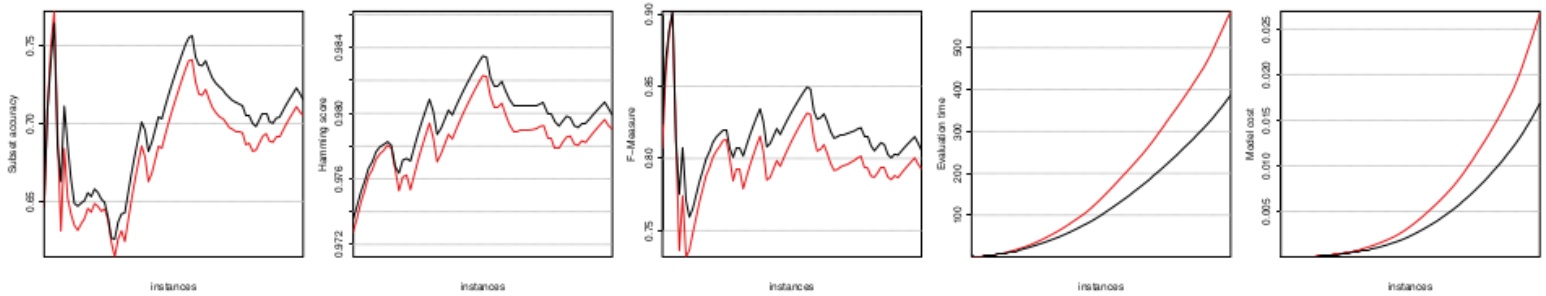
MLSAMPkNN MLSAMkNN SAMkNN MLkNN kNNPA

MLSAMPKNN

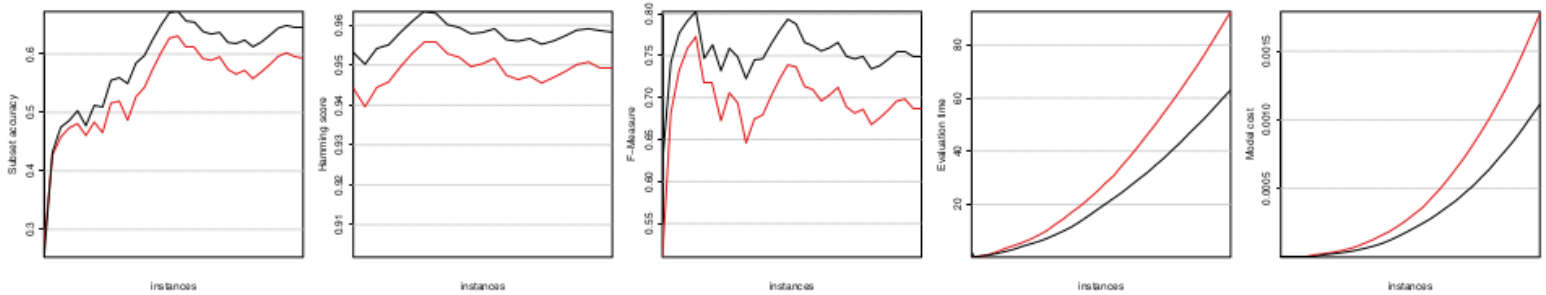
- Contribution of the punitive system



(a) Bookmarks.



(b) Eukaryote.



(c) Human.

Outline

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

Ending notes

Thank you for your attention! Q & A time!