Uczenie klasyfkaktorów ze zmiennych strumieni danych

Jerzy Stefanowski

Institute of Computing Sciences,
Poznan University of Technology
Poland

Wykład dla PSE – wersja 2020 – wykorzystujący slajdy z
INIT/AERFAI Summer School on Machine Learning 2017

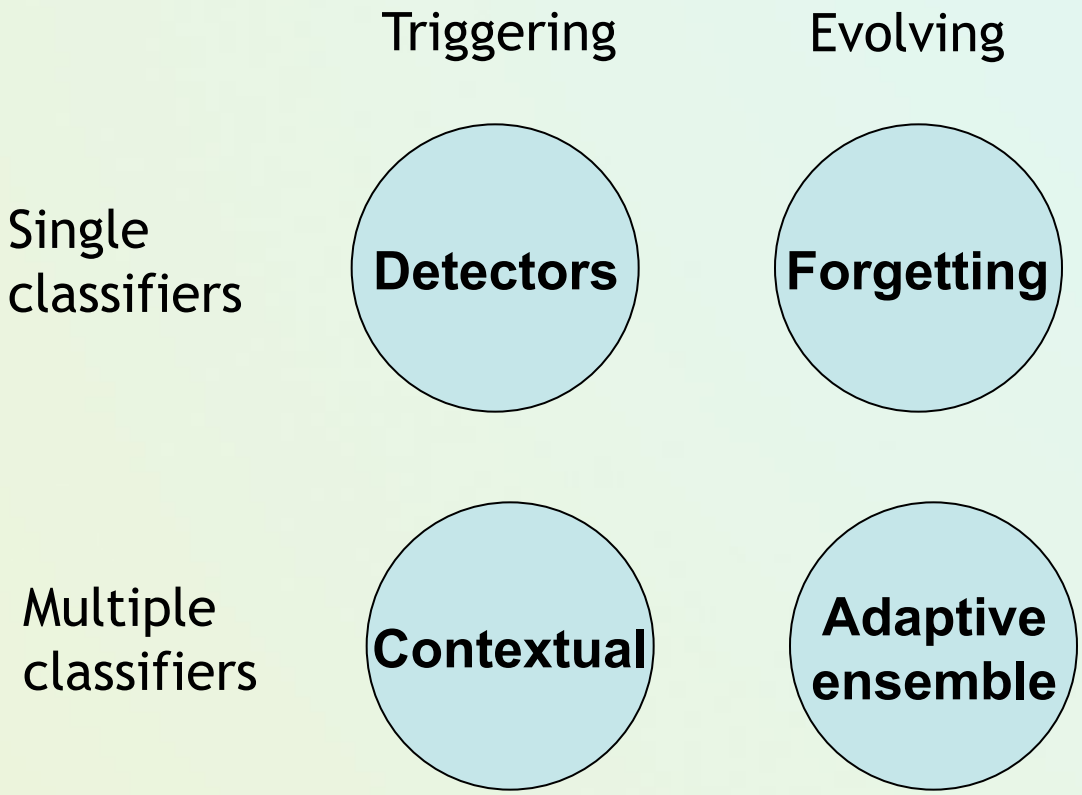
Outline of the talk - part 2

1. Change detection
2. Ensembles for drifting data streams
3. Block-based approach
   → Accuracy Updated Ensemble
4. Strategies for transforming block-based ensembles into instance on-line ones
5. On-line Accuracy Ensemble
6. Learning from imbalanced data
7. Open issues and final remarks

Our research

Ack: Strong co-operation with Dariusz Brzezinski !

Categorization of learning algorithms



Bifet A., Gama. J., Pechenizky M., Zliobaite I.: Handling concept drift. Importance, challenges and solutions. PAKDD Tutorial (2011)

Change detection methods

They are used for explicit drift detection unlike blind adaption or forgetting methods

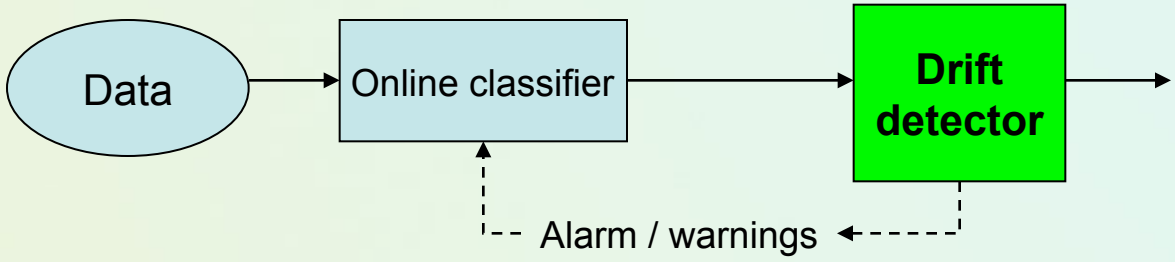
They can:

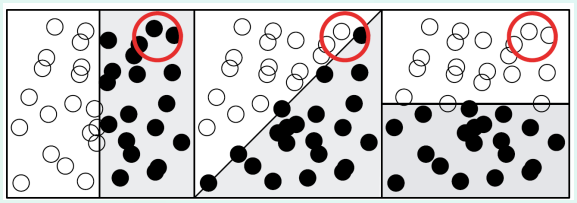
- Indicate change-points or
- Provide small time-windows where the change occurs

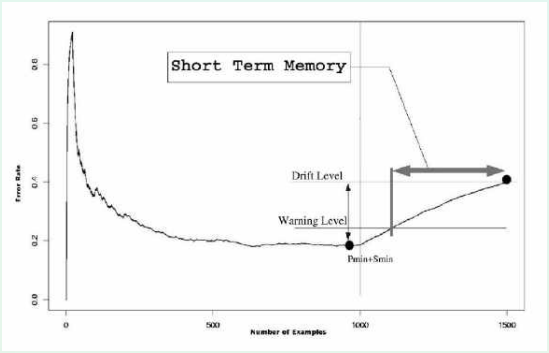
Two different perspectives

- Monitoring the evolution of performance indicators
- Monitoring distributions on two different time-windows

Triggers – the use of drift detectors







Statistical Process Control

DDM, EWMA,…

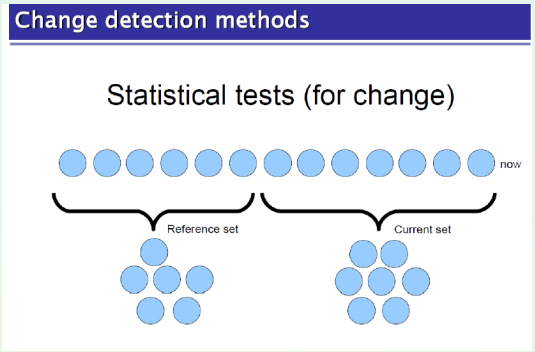
Sequential Analysis

Cumulative Sum Test, Page-Hinkley test

Monitoring distributions over windows

ADWIN

Context approaches



More: J.Gama, I.Zliobaite, M.Pechenizkiy, A. Bouchachia: A Survey on Concept Drift Adaptation. ACM Compt. 2013

Monitor quality of learning with SPC

- When there is a change in the stream the actual classifier does not correspond any more to the actual distribution and its prediction becomes worse
- Monitor the quality of the learning process (estimation of its error rate) using Statistical Process Control techniques
  - Should react to real drift but not to noise or outliers

DDM (Drift Detection Method) is the representative approach

- It estimates the classifier error and its standard deviation (which should decrease with more examples in the stream)
- If the error significantly increases -> indicate drift and the current classifier has to be rebuild

Drift Detection Method - DDM

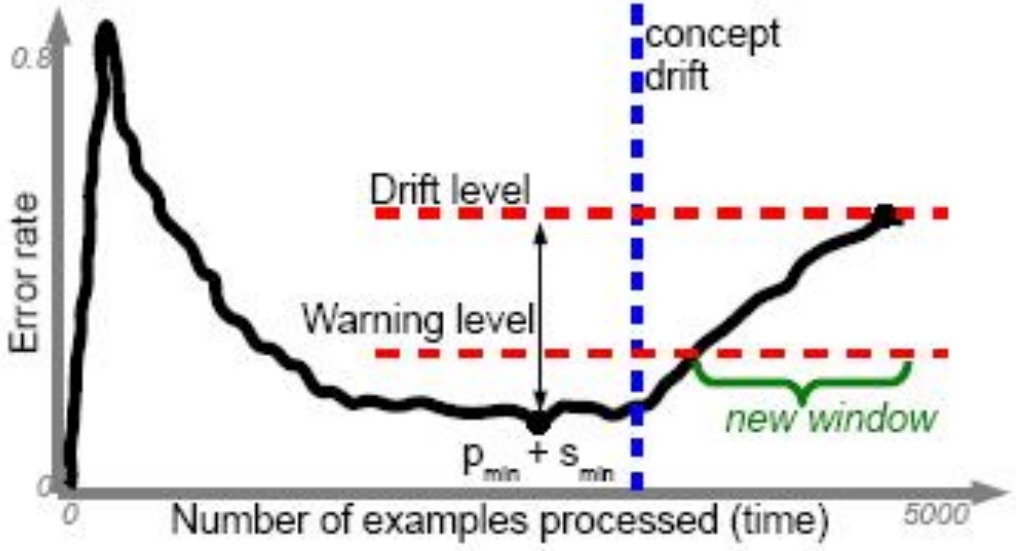
J. Gama, Medas, Gladys, Rodrigues;Learning with Drift Detection. In Proc. of SBIA-, 2004

- Monitor the evolution of the error rate.
- Suppose a sequence of examples in the form < xi,yi>
  - The actual decision model classifies each example in the sequence
  - In the 0-1 loss function, predictions are either True or False
  - The predictions of the learning algorithm are sequences: T, F,T, F,T, F,T,T,T, F, …
  - The Error is a random variable from Bernoulli trials.
  - The Binomial distribution gives the general form of the probability of observing a false prediction (error)
  - It can be approximate by a Gaussian distribution for larger *n*
  - p_i=(#F)/i and s_i=sqrt(p_i(1-p_i)/i)

Monitoring the evolution of the error rate

- Maintains two registers:
  - P_{min} and S_{min} such that $P_{min} + S_{min} = \min(p_i + s_i)$
  - Minimum of the error rate taking into account the variance of the estimator.
  - They are valid after collecting at least 30 examples
- At example j : the error of the learning algorithm will be
  - Out-control if pj + sj > pmin + α smin
  - In-control if pj + sj < pmin + ß smin
  - Warning Level: if pmin + α smin > pj + sj > pmin + smin
- The constants α and ß depend on the desired confidence level.
  - Recommended values are = 2 and = 3.

DDM – idea [Gama e al. 2004]



If the error is

- In-control the current model is still valid. It may Incorporate the example in the decision model
- Warning zone: Maintain the current model and start the window at Lwarning = j
- Out-Control Re-learn a new classifier using as training set the set of examples in the warning window.

Early Drift Detection Method (EDDM)

- EDDM is a modification of DDM to improve the detection of gradual drift
- It uses the same warning and alarm mechanism – however instead of the classifier error, the authors propose to use the distance between two error and its standard deviation
- It better identify slow gradual drifts but may be more sensitive to noise

Manuel Baena-García, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method, In Fourth International Workshop on Knowledge Discovery from Data Streams, 2006.

Sequential Analysis (CSUM)

- The CUSUM test is used to detect significant increases (or decreases) in the successive observations of a random variable x
- It gives an alarm when the mean of the input data is significantly different from zero
- For detecting increases:

$$g_0 = 0$$

$$g_t = \max\left(0; g_{t-1} + (x_t - \alpha)\right)$$

- If $g_t > \lambda$ then alarm and $g_t = 0$

More discussion with motivation for PH Test: João Gama, Raquel Sebastião, Pedro P. Rodrigues, On evaluating stream learning algorithms, Machine Learning

Page Hinkley algorithm

- The PH test is a sequential adaptation of the detection of an abrupt change in the average of input normal signal
- It considers a cumulative variable m_T , defined as the cumulated difference between the observed values and their mean till the current moment:

$$m_{t+1} = \sum_{1}^{t} (x_t - \bar{x}_t + \alpha)$$

where *a* corresponds to the magnitude of allowed changes

- Additionally the minimal m_t is defined as

$$M_T = \min(m_i; i = 1, \ldots t)$$

- The PH test monitors the difference $PH_t = m_t - M_T$
- If this difference is greater than threshold (λ) - alarms a change
  - Gama et al proposed to consider error rate as an observed value
  - Alternatively it may be a ratio between errors estimated on two windows (larger and short)

Detection by model changes

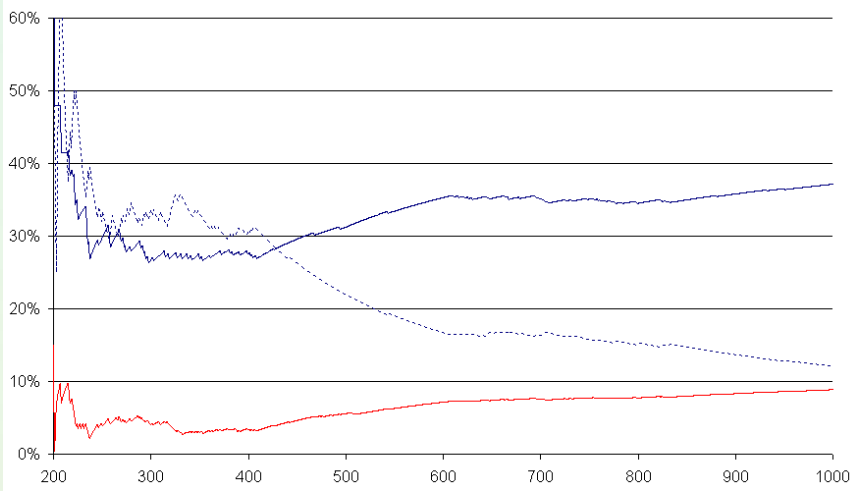
- Partially labeled examples
- Store assignments of unlabeled examples to particular leaves
- Leaf l covering n_l examples – the statistic

$$P(l) = \frac{n_l}{N} \quad \sum_l P(l) = 1$$

- Data distribution – approximated by distribution of leaf statistics
- Strong change of leaf statistics may indicate a concept drift

- Looking rather for trend changes – increase of differences / not simple thresholds
- When a trend in differences is identified, then

1. Construct a new training set (get labels)
2. Verify error estimation
3. Retrain older classifier and induce a new one



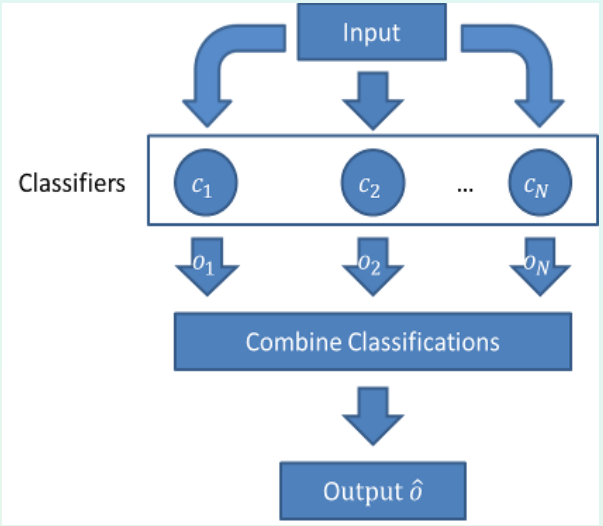
M.Kmieciak, J.Stefanowski: Handling Sudden Concept Drift in Enron Message Data Streams. Control and Cybernetics 2011.

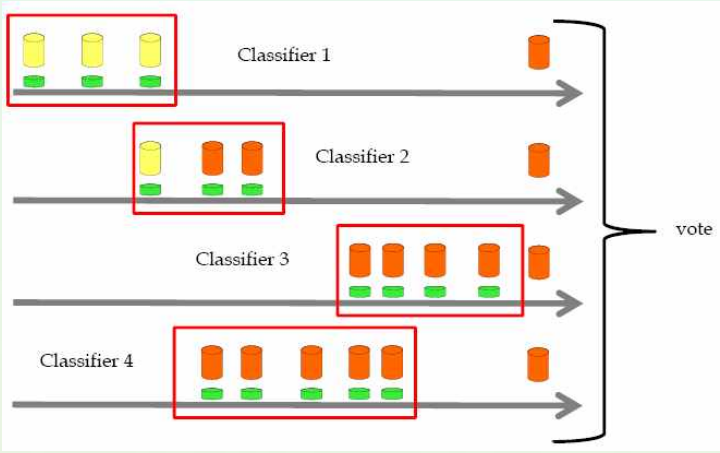
Why could we apply multiple classifiers (ensembles) to concept-drifting data?

- Many proposals for static data
- Natural for non-stationary frameworks
  - Modular construction
    - Flexibility to incorporate new data
      - adding new components
      - updating existing components
    - Natural forgetting
      → pruning ensembles
    - Continuous adapting aggregation (voting weights) technique
    - Reduce the variance of the error comparing to single classifiers
      → stability
  - Another motivation

Stability-plasticity dilemma

During changes data generated as a mixture of distributions
→ may be modeled as a weighted combination





Ensembles for concept drifting streams

- Different taxonomies → Kuncheva (2004)

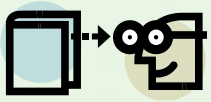
  Dynamic combiners → component classifiers learn in advance, adapting by changing the combination rule [weighted majority]

  Updating training data → recent data use to on-line update of component classifiers [Oza]

  Updating ensemble members→ update on-line or retrain in a batch mode

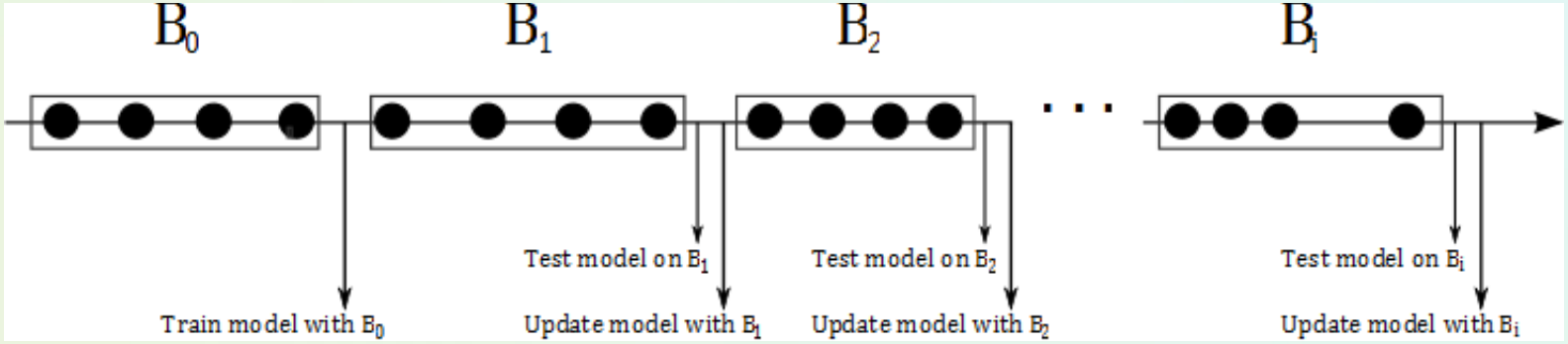
  Structural changes of the ensemble → replace "the loser" and add new component

- Trigger vs. Adaptive one → Active vs. Passive
- This presentation
  - On-line ensembles → learn incrementally after processing single examples
  - Block-based ensembles → learn after processing blocks of data
    - Solutions for stationary vs. concept drifting streams

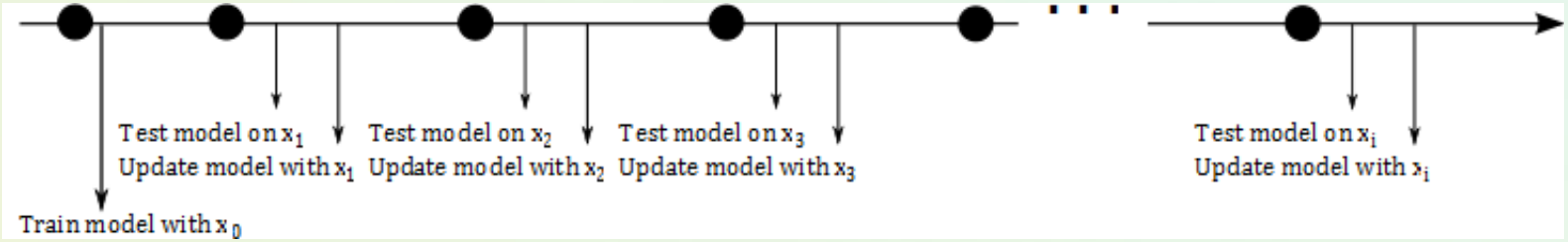


Different processing schemes

Block processing (data chunks)



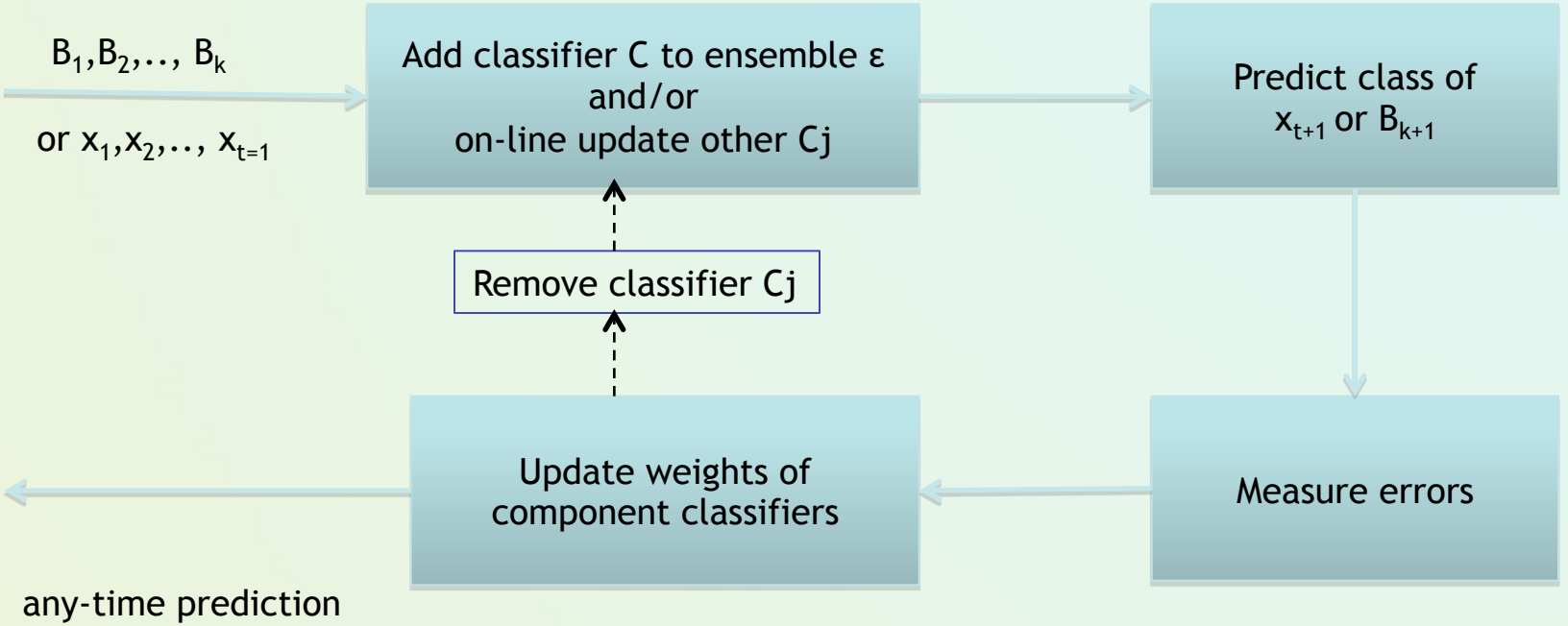
Online processing (instances)



Completely labeled examples or partly …?

Test-then-train scenario

Adaptive Approaches



- Continuously adapt the ensemble and its parameters
- Passive / adaptive approaches → the most popular
- A few ensembles with trigger techniques
  - Adaptive Classifier Ensemble ACE (Nishida et al. 2009)
  - JIT ensemble (C. Alippi et al. 2012), BWE (2011), …

Taxonomy of adaptive ensembles

Block-based ones:

Streaming Ensemble Algorithm (SEA) – Street & Kim 2001

Accuracy Weighted Ensemble (AWE) Wang et al 2003

Learn++.NSE – Polikar et al.

Recurring concepts

- CCP – Katakis et al. 2010
- RCD – Goncalvas et al. 2013
- FAE – Diaz et al 2015

Block processing also in some semi-supervised or novel class detection – Masud et al. 2009; Farid et al 2013

On-line (instance based)

WinNow, Weighted Majority Alg. – Littlestone 1988, L & Warmuth

Dynamic Weighted Majority (DWM) – Kolter & Maloof 2003 → AddExp (2005)

On-line bagging and on-line boosting [Oza]

- BagADWIN,
- Leverage bagging – Bifet et al. 2007

DDD – Minku 2011 (diversity & on-bagging)

Hoeffding Option Trees (HOT)

UFFT (Gama at al. 2005)

ADACC – Jaber 2013

Boosting classifier for drifting concepts – Scholtz & Klinkenberg 2007

Hybrid approaches

ACE – Nishida 2009, AUE → OAUE

Recent surveys on ensembles and data streams

Just published in 2017:

- Bartosz Krawczyk, Leandro L. Minku, Joao Gama, Jerzy Stefanowski, Michał Wozniak: Ensemble learning for data stream analysis: a survey, Information Fusion. 37 (2017), 132-156.
- H.Gomes, J.Barddal, F.Enembreck, A.Bifet: A survey on ensemble learning for data stream classification. ACM Computing Surveys, vol. 50 (2), March 2017

Earlier surveys on classification in streams:

- Gregory Ditzler, Manuel Roveri, Cesare Alippi, Robi Polikar. Learning in non-stationary environments: A survey. IEEE Computational Intelligence Magazine, 10(4):12–25, (2015).
- Vincent Lemaire, Christophe Salperwyck,Alexis Bondu.: A survey on supervised classification on data streams. In Proc. Business Intelligence eBIS 2014, vol. 205 of Lecture Notes Business Information Processing, 2015

Older – Ludmila Kuncheva: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. Workshop SUEMA at ECAAI 2008.

Majority vote algorithm (Littlestone, Warmuth, 1994)

Train the base classifiers in the ensemble *D*1 ,…, *DL*. **(Pick your experts / horse racing)** - they are not re-train in next step

1. Set all weights to 1, *wj* = 1/*L*, *i* = 1, …, *L*. Choose ß∈ [0,1].
2. For a new ***x***, calculate the support for each class as the sum of the weights for all classifiers that voted for that class. (Sum up the weights for those experts that predicted a win and compare with the sum of weights of those who predicted a loss.). Make a decision for the most supported class.
3. Observe the true label of x and update the weights of the classifiers that were wrong using wj=βwj.
4. Normalize the weights and continue from 2.

Winnow variant [Littlestone]

- Observe the true label of **x** and update the weights of **all** the classifiers **if the ensemble prediction was wrong.**

The author showed theoretical bounds for an ensemble error.

Dynamic Weighted Majority (DWM)

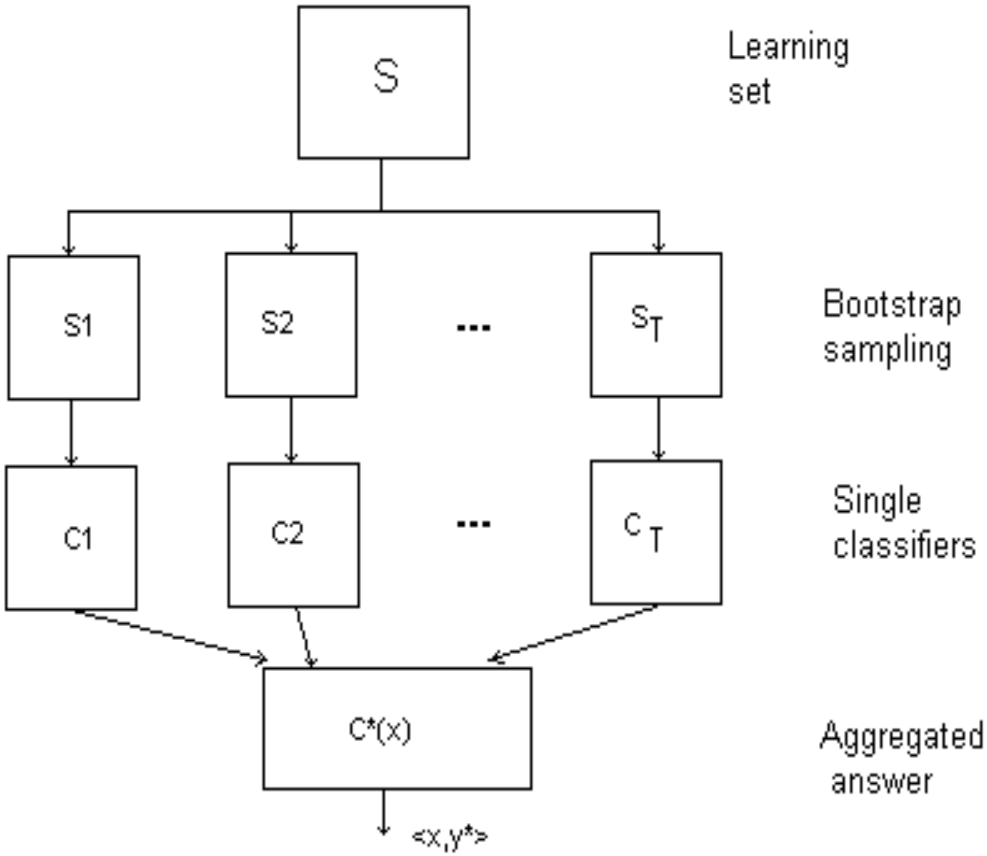
Kolter and Maloof (ICDM03, ICML05)

The Dynamic Weighted Majority algorithm (DWM) is an ensemble method, based on Majority Voting Algorithm, for tracking concept drift

- Maintains an ensemble of base learners (experts),
- Predicts using a weighted-majority vote of these experts
- Dynamically creates and deletes experts in response to changes in performance (parameters)

From static Bagging to online versions

Bootstrap aggregating – L.Breiman [1996]



input S – learning set, T – no. of bootstrap samples, LA – learning algorithm

output C* - multiple classifier

for i=1 **to** T **do**

begin

S_i:=bootstrap sample from S;

C_i:=$LA(S_i)$;

end;

$$C^*(x) = \text{argmax}_y \sum_{i=1}^{T} (C_i(x) = y)$$

Bootstrap sampling

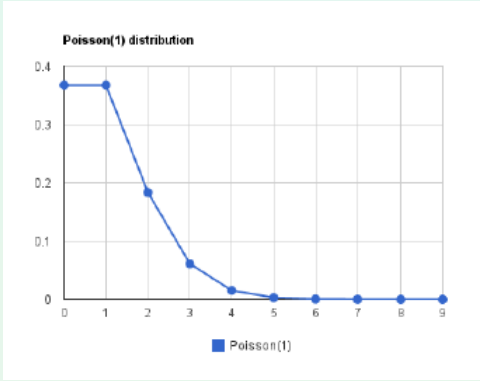
- Bagging = **B**ootstrap **agg**regation
  - Generates individual classifiers on bootstrap samples of the training set
- As a result of the sampling-with-replacement procedure, each classifier is trained on the average of 63.2% of the training examples.
  - For a dataset with N examples, each example has a probability of $1-(1-1/N)^N$ of being selected at least once in the N samples. For $N \rightarrow \infty$, this number converges to (1-1/e) or 0.632

- How to simulate sampling with replacement in streams?

Online Bagging [N.Oza, S.Russel]

Use incremental learning of component classifiers

Basic idea: each incoming example is presented to a component classifier r times, where r is defined by Poisson(1) distribution



Create *k* component classifiers

for each example *x*

    for *i* = 1 to *k*

    send *r* copies of *x* to *i*th classifier with prob. P(*r*)

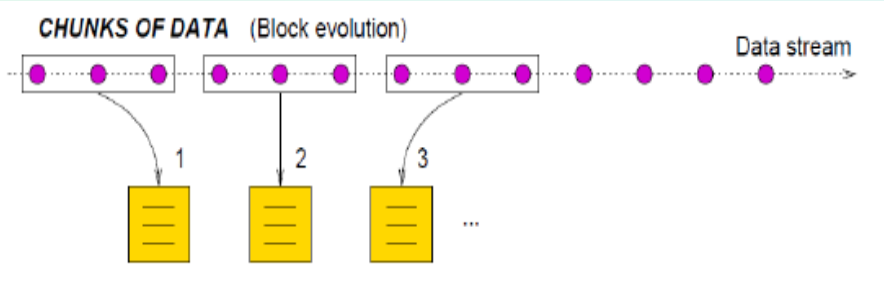
predict(*x*) = majority vote of *k* classifiers

Many extensions of online bagging

- ADWIN Bagging [Bifet]
  - ADWIN used as a change detector. When a change is detected, the worst classifier is removed and a new classifier is added.
- Levaraging Bagging [Bifet et al]
  - Introducing more randomization while constructing online bagging
  - Input – change resampling r = Poisson(λ)
  - Output: Random Output Codes
- DDD [Minku, Yao]
  - Uses several diversified online bagging ensemble and change them when the drift is detected

Block-based ensemble

- The origins → SEA (Streaming Ensemble Algorithm)



- Generic schema
  - train K classifiers from K blocks
  - for each subsequent chunk - block

    train a new component classifier

    test other classifiers against the block

    assign weight to each classifier

    select top K classifiers (remover the weaker classifiers)

Some advantages:

- When examples comes in blocks (chunks)
- Use static learning algorithms
- May have smaller computational costs than on-line ensembles

Accuracy Weighted Ensemble - AWE

H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining Concept-Drifting Data Streams using Ensemble Classifiers", KDD'03.

Idea: **Weight classifiers according to the current data distribution**

- Formal proof that classifiers weighted this way are equally or more accurate than single classifiers built upon all
- Weights approximated by computing the classification error on the most recent data block (chunk)

$$w_{ij} = MSE_r - MSE_{ij}, MSE_{ij} = \frac{1}{|B_i|} \sum_{\{\mathbf{x},y\} \in B_i} (1 - f_y^j(\mathbf{x}))^2, MSE_r = \sum_y p(y)(1 - p(y))^2$$

- The new candidate – 10-fold cross validation on the latest block
- Remove classifiers with weights smaller than MSE_r
- Originally used with J48 (classical non-incremental) trees / also other static learners

Limits of block-based algorithms

- Accuracy is highly dependent on data block B_i size
  → needs experimental efforts to tune
- Too slow reacting to sudden concept drifts
  - Small block size may help, but not for stability periods and comput. costs
- Sudden concept drifts can sometimes mute all base classifiers
- Component classifiers often trained only ones, never change

Refer to on-line ensembles:

- They better react to sudden drifts but less to gradual drifts
- Component classifiers update over time
- However, usually more computationally costly

What to do?

Drift Detection and Ensembles

Adaptive Classifier Ensemble (ACE)

Nishida K, Yamauchi K, Omori T

Motivations – to overcome limitations of adaptive block based ensembles (AWE);

To force component retraining when drift is detected

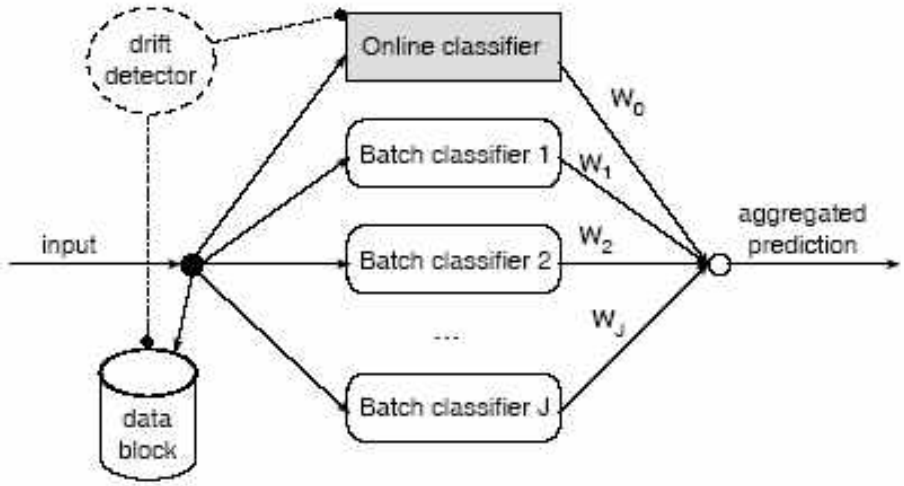


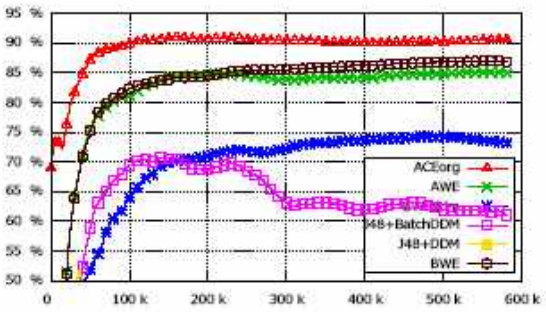
Figure 2.8: Basic architecture of ACE

The drift detection – based on confidence intervals

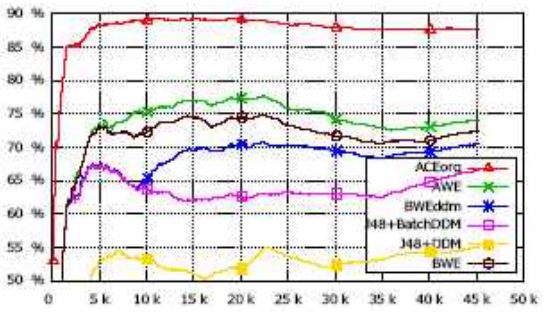
Aggregation – non-linear weights

Experiments – a bit limited

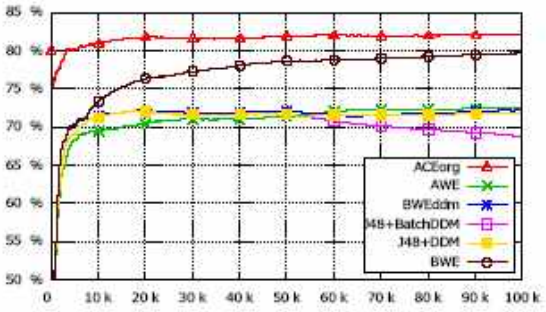
Experimental evaluation of ACE



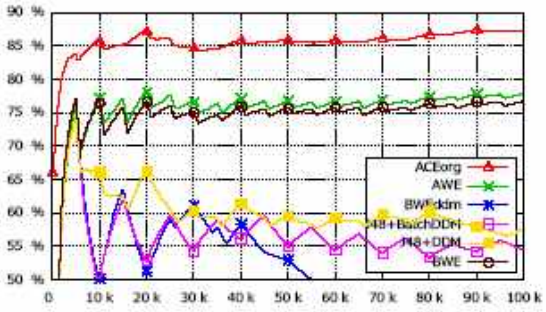
(a) CovType



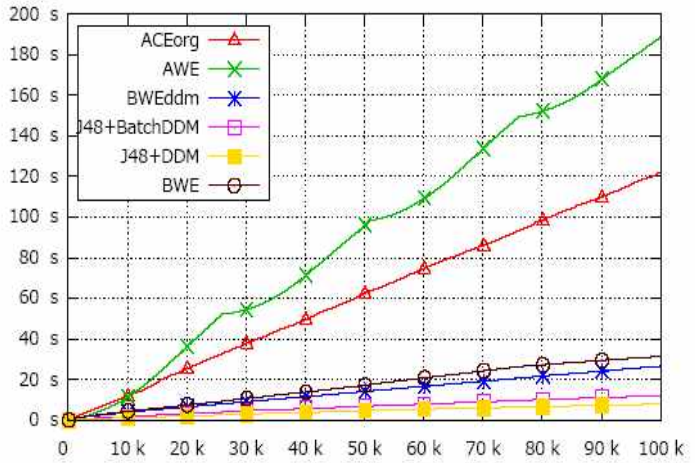
(b) Electricity



(c) Hyperplane



(d) RBFSudden



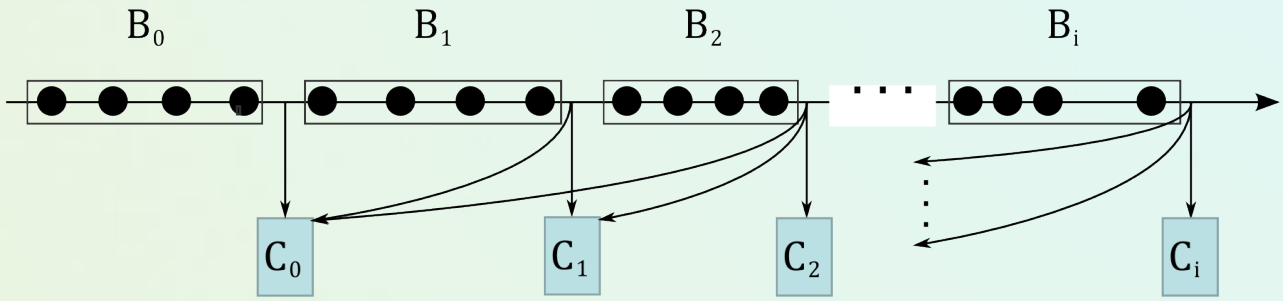
- Deckert 2011, 2012 – ACE outperforms AWE and its variants, also block based detectors like BWE
- However, it may be computationally costly

More: Deckert M., Stefanowski J. Comparing block ensembles for data streams with concept drift. In Workshop on Mining Complex and Stream Data, ADBIS 2012

Our approaches to stream ensembles

- Hypothesis
  - ***Could we combine best properties of both (block, on-line) approaches to sufficiently adapt to several types of concept drifts with satisfactory memory and time?***
- Our proposals
  - Block-based AUE
  - On-line OAUE
- Strong co-operation with Dariusz Brzeziński

Accuracy Updated Ensemble - Motivations



- Keep the block schema of constructing new classifiers, substituting the worst ones, periodical evaluations of components (weighting)
- Incremental updating of component classifiers
  - Improves reaction to various drifts, and reduces the influence of the block size
- Analysis of changes in weighting component classifiers and the role of the new introduced classifier
- Additional reducing computational costs
- Hope to improve predictive peformance

Accuracy Updated Ensemble - AUE

- Incremental updating component classifiers (Hoeffding Trees)
- New weighting of classifiers
  - Another non-linear weights (better differentiate classifiers and resign from extra pruning with MSE_r)

$$w_{ij} = \frac{1}{MSE_{ij} + MSE_r + \varepsilon}$$

- Newest classifier treated as the best one (MSE_{ij}=0) – gets a highest weight and always substitute the worst classifiers
- Reducing computational costs
  - Resign from an extra buffer of classifiers
  - Manage memory limits – prune trees
  - No extra cross-validation for a new component

D. Brzezinski, J. Stefanowski: **Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm**. IEEE Transactions on Neural Networks and Learning Systems, 25 (1), 81-94 (2014).

D.Brzeziński, JStefanowski: Accuracy updated ensemble for data streams with concept drift, Proceeding of HAIS 2011.

Experimental evaluation

15 Datasets

- 11 synthetic and 4 real ones

45 000 - 10 000 000 instances

Different drift scenarios

- incremental, gradual, sudden, recurring, mixed, blips, no drift
- Fast vs. slow rate

MOA generators →

Hyperplane, RBF, SEA, Tree. LED

AUE → Implementation for MOA

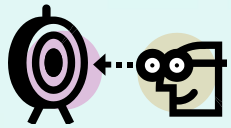
Base classifiers → Hoeffding Tree with NB leaf predictions (n_{min}=100, δ=0.01, φ=0.05)

Evaluate: time, memory, predictive accuracy

Table 3.1: Characteristic of datasets

Dataset	Instances	Attributes	Classes	Noise	Drifts	Drift type
$\texttt{Hyp}_S$	1M	10	2	5%	1	incremental
$\texttt{Hyp}_F$	1M	10	2	5%	1	incremental
$\texttt{RBF}_B$	1M	20	4	0%	2	blips
$\texttt{RBF}_{GR}$	1M	20	4	0%	4	gradual
$\texttt{RBF}_{ND}$	1M	20	2	0%	0	none
$\texttt{SEA}_S$	1M	3	4	10%	3	sudden
$\texttt{SEA}_F$	2M	3	4	10%	9	sudden
$\texttt{Tree}_S$	1M	10	4	0%	4	sudden recurring
$\texttt{Tree}_F$	100k	10	6	0%	15	sudden recurring
$\texttt{LED}_M$	1M	24	10	10%	3	mixed
$\texttt{LED}_{ND}$	10M	24	10	20%	0	none
Elec	45k	7	2	-	-	unknown
CovType	581k	53	7	-	-	unknown
Poker	1M	10	10	-	-	unknown
Airlines	539k	7	2	-	-	unknown

Component Analysis of AUC



- The role of an additional buffer (up to 20 classifiers)

AUE without it practically the same accuracy but reduces comput. approx. 5 training times, 2 memory

- New candidate classifier weighting schemas

Non-linear weight and 'perfect' classifiers the best accuracy with reduced training time ; some differences for no-drift and blips data (preference to older components)

- Refraining from some component updates

Best accuracy → update all components / for drift datasets

Details in : D.Brzezinski, Block-based and on-line ensembles for concept-drifting data streams, Ph.D. Thesis, 2015

	All	$\theta_{0.5\%}$	$\theta_{1\%}$	$\theta_{2\%}$	$\theta_{3\%}$	$\theta_{5\%}$	b_4	b_5	b_6	b_7	b_8
Hyp_S	88.44	88.51	88.58	88.43	87.74	**89.49**	87.72	87.96	88.23	88.39	88.48
RBF_B	**94.81**	94.34	93.99	92.57	88.61	78.30	93.83	94.12	94.31	94.62	94.60
RBF_{GR}	**94.10**	93.79	93.39	91.54	86.49	79.80	93.27	93.61	93.74	93.97	94.01
RBF_{ND}	92.58	92.99	92.59	91.55	89.40	77.08	92.03	92.49	92.82	92.97	**93.12**
SEA_S	**89.21**	89.16	89.12	88.61	88.00	87.19	88.67	89.01	89.04	89.07	89.19
Elec	**71.83**	70.61	70.52	70.60	70.81	70.93	69.29	69.29	69.29	69.29	69.29
CovType	**84.79**	84.57	84.19	83.36	82.57	81.17	83.29	83.60	83.99	84.21	84.57
Poker	**60.77**	59.67	59.68	59.69	59.79	59.85	59.82	59.82	59.82	59.82	59.82

Comparative Study

AUE compared against 11 other algorithms:

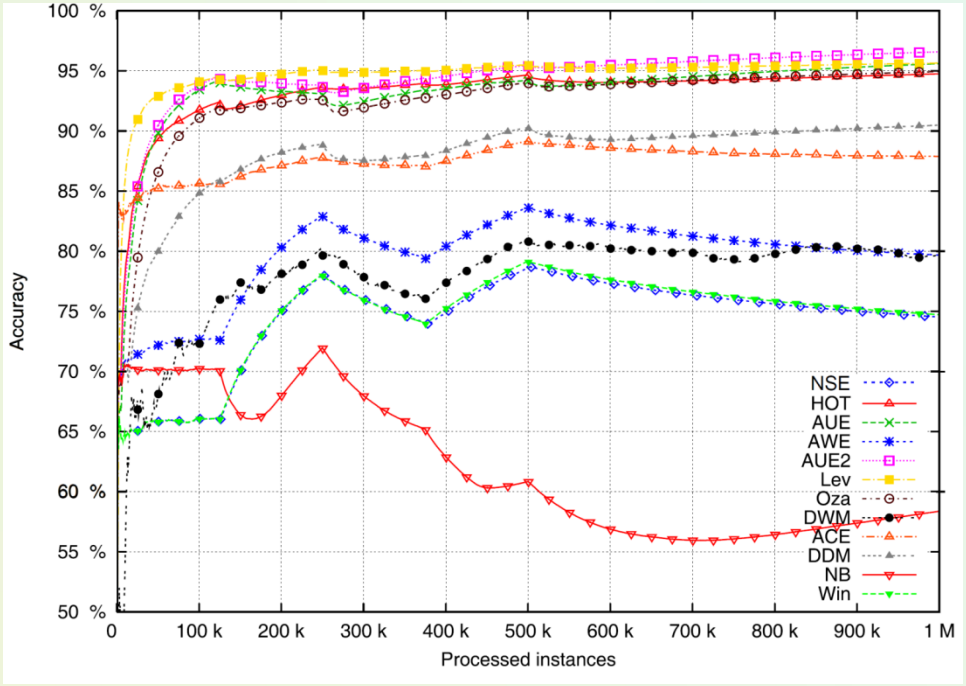
- Accuracy Weighted Ensemble (AWE)
- Hoeffding Option Trees (HOT)
- Adaptive Classifiers Ensemble (ACE)
- AUE (previous 2011 version)
- Online Bagging, Leveraging Bagging
- Dynamic Weighted Majority (DWM)
- Learn^{++}.NSE
- Single HT with a window (Win)
- Naive Bayes (NB)

Mostly MOA implementations, ACE and Learn^{++}.NSE adapted from other versions

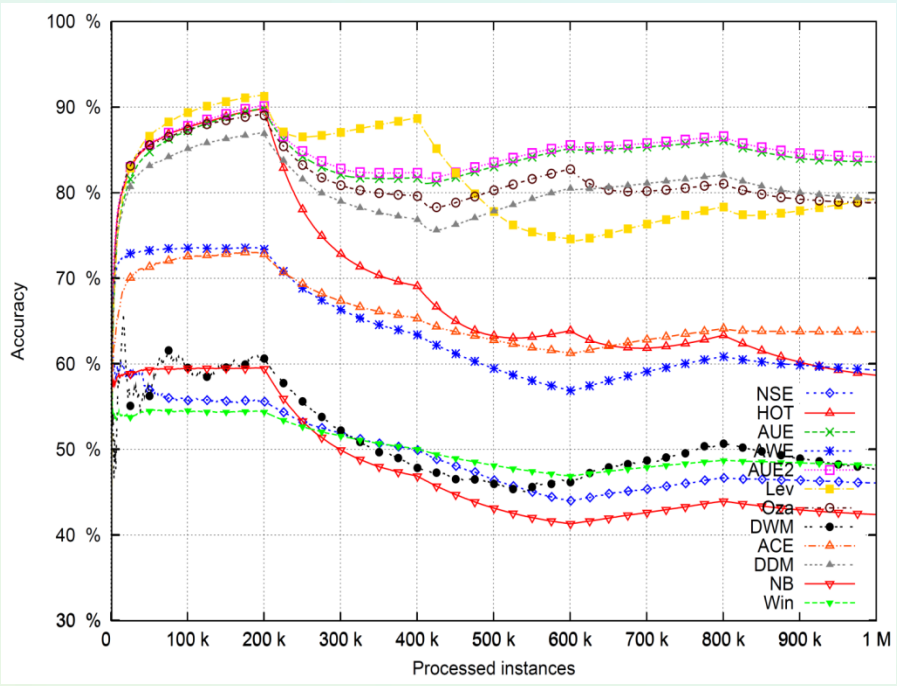
❑ The same 15 datasets

- Evaluate: time, memory, predictive accuracy

Reacting to different changes



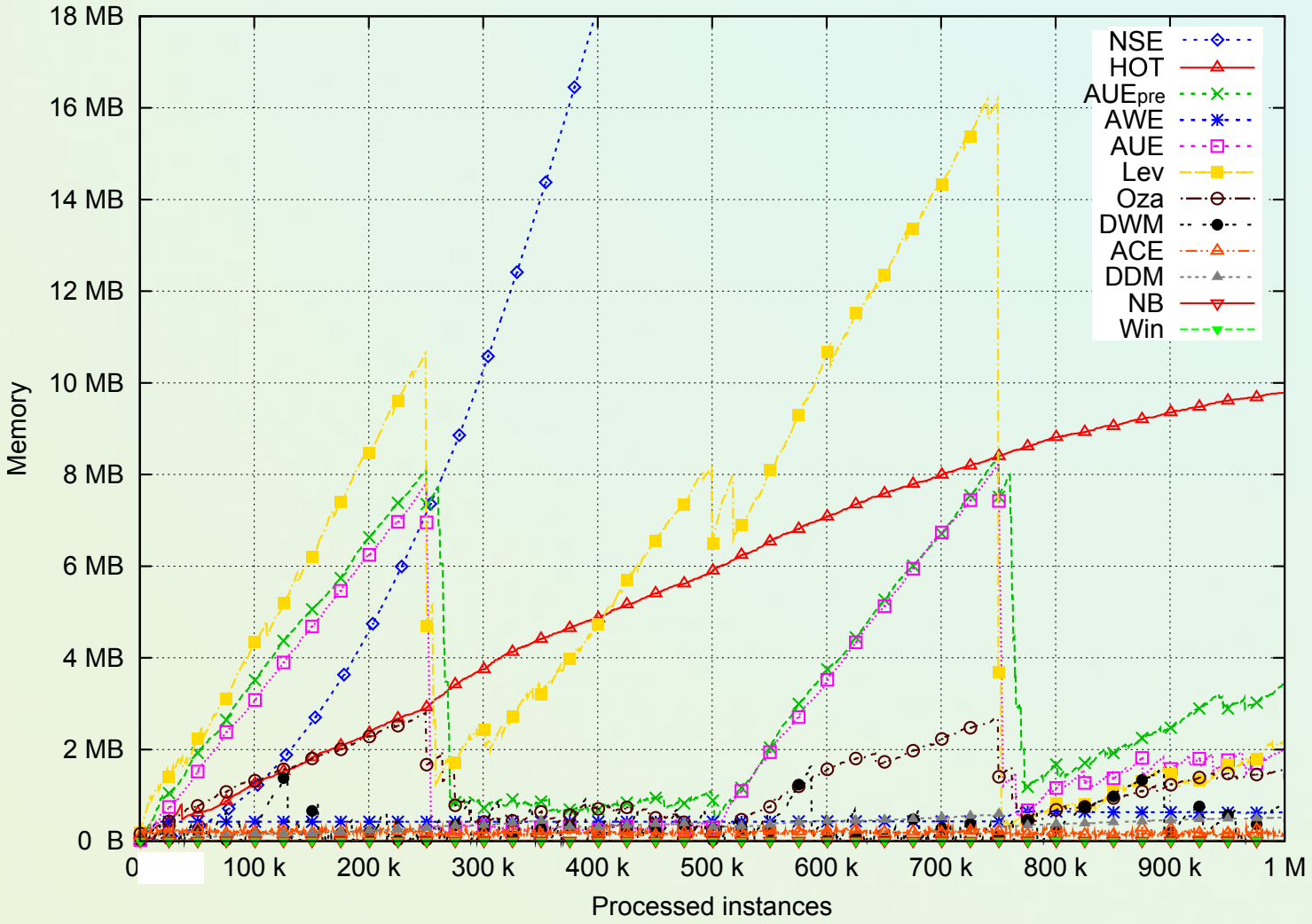
Classification accuracy on RBF_{GR} (slow, gradual changes)



Classification accuracy for $Tree_S$ dataset (fast, sudden)

Memory requiremnts

RBF-F synthetic data



Comparative study – classification accuracy

Ranks in Friedman test

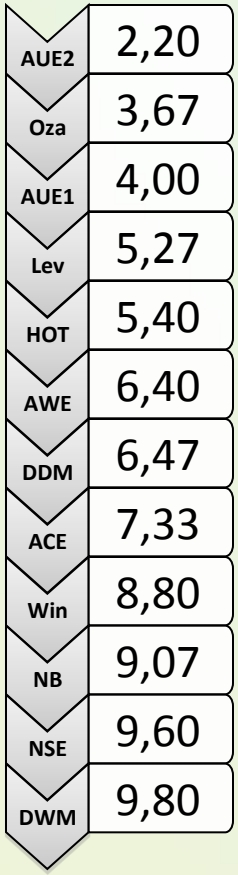


Table 1: Average classification accuracy

Data	ACE	AWE	AUE1	AUE2	HOT	DDM	Win	Lev	NB	Oza	DWM	NSE
Hyp_S	80,65	**90,43**	88,59	88,43	83,23	87,92	87,56	85,36	81,00	89,89	71,20	86,83
Hyp_F	84,56	89,21	88,58	**89,46**	83,32	86,86	86,92	87,21	78,05	89,32	76,69	85,39
RBF_B	87,34	78,82	94,07	94,77	93,79	88,30	73,07	**95,28**	66,97	93,08	78,11	73,02
RBF_GR	87,54	79,74	93,37	94,43	93,24	87,99	74,67	**94,74**	62,01	92,56	77,80	74,49
RBF_ND	84,74	72,63	92,42	**93,33**	91,20	87,62	71,12	92,24	72,00	91,37	76,06	71,07
SEA_S	86,39	87,73	89,00	**89,19**	87,07	88,37	86,85	87,09	86,18	88,80	78,30	86,23
SEA_F	86,22	86,40	88,36	**88,72**	86,25	87,80	85,55	86,68	84,98	88,37	79,33	85,07
Tree_S	65,77	63,74	84,35	**84,94**	69,68	80,58	50,15	81,69	47,88	81,67	51,19	49,37
Tree_F	45,97	45,35	**52,87**	45,32	40,34	42,74	41,54	33,42	35,02	43,40	29,30	33,90
LED_M	64,70	67,11	67,29	67,58	66,92	67,17	65,52	66,74	67,15	**67,62**	44,43	62,86
LED_ND	46,33	51,27	50,68	51,26	51,17	51,05	47,07	50,64	**51,27**	51,23	26,86	47,16
Elec	75,83	69,33	70,86	77,32	**78,21**	64,45	70,35	76,08	73,08	77,34	72,43	73,34
Cov	67,05	79,34	81,24	85,20	**86,48**	58,11	77,19	81,04	66,02	80,40	80,84	77,16
Poker	67,38	59,99	60,57	66,10	74,77	60,23	58,26	**82,62**	58,09	61,13	74,49	59,56
Airlines	66,75	63,31	63,92	**67,37**	66,18	65,79	64,93	63,10	66,84	66,39	61,00	63,83

Block to online transformation: Why

- Complementary approaches:
  - Block-based algorithms react well to gradual changes
  - Online algorithms offer quicker reactions to sudden drifts
- Online learners are of more value in some scenarios
  + in some environments class labels available after each example
- Block-based inspirations:
  - Component evaluation and their weighting,
  - Ensemble periodically updated with a new *candidate* classifier trained on last *d* examples
- Can block-based algorithms be adapted to work in online environments?
- Could be inspirations for a new on-line algorithm?

Block to online transformation: How

- We modify a generic block based training schema:
  - Weight classifiers, remove the worst
  - Keep periodically adding a new *candidate* classifier trained on last *d* examples
- Three **transformation** strategies:
  - Windowing technique
  - Additional online (instance) ensemble member
  - Drift detector with on-line classifier

Main Observations

- Weighting after each example improves accuracy AWE and AUE (2,3%) but highly increases training time (15x)

  => too expensive, look for other solution
- Additional on-line classifier improves more AWE than AUE

  => Perhaps specialized weight depending on ensemble

  ⇒ AUE updates earlier components
- Drift Detector slightly for AUE, not for AWE

  => needs for other solutions

- We should compare it against typical on-line ensembles

Incremental updating component classifiers and weighting with incoming examples are profitable

Online Accuracy Updated Ensemble

- Advantages of periodical adaptation mechanisms and on-line classifiers with weighting after each example

  → minimize computational costs of the transformation strategies

Basic characteristics of a new online examples

- On-line learning classifiers
- Periodical adding a new classifiers with highest weight
- No blocks → sliding window
- New weighting function → on-line more efficient per example

Moreover,

- Perform comparable to state-of-the-art algorithms, wrt. accuracy, memory and time

Non-linear weight calculated after each example

$$MSE_i^t = \begin{cases} MSE_i^{t-1} + \frac{e_i^t}{d} - \frac{e_i^{t-d}}{d}, & t - \tau_i > d \\ \frac{t - \tau_i - 1}{t - \tau_i} \cdot MSE_i^{t-1} + \frac{e_i^t}{t - \tau_i}, & 1 \leq t - \tau_i \leq d \\ 0, & t - \tau_i = 0 \end{cases}$$

$$e_i^t = (1 - f_{iy}^t(\mathbf{x}^t))^2$$

$$MSE_r^t = \begin{cases} MSE_r^{t-1} - r^{t-1}(y^t) - r^{t-1}(y^{t-d}) + r^t(y^t) + r^t(y^{t-d}), & t > d \\ \sum_y r^t(y), & t = d \end{cases}$$

$$r^t(y) = p^t(y)(1 - p^t(y))^2$$

$$w_i^t = \frac{1}{MSE_r^t + MSE_i^t + \epsilon}$$

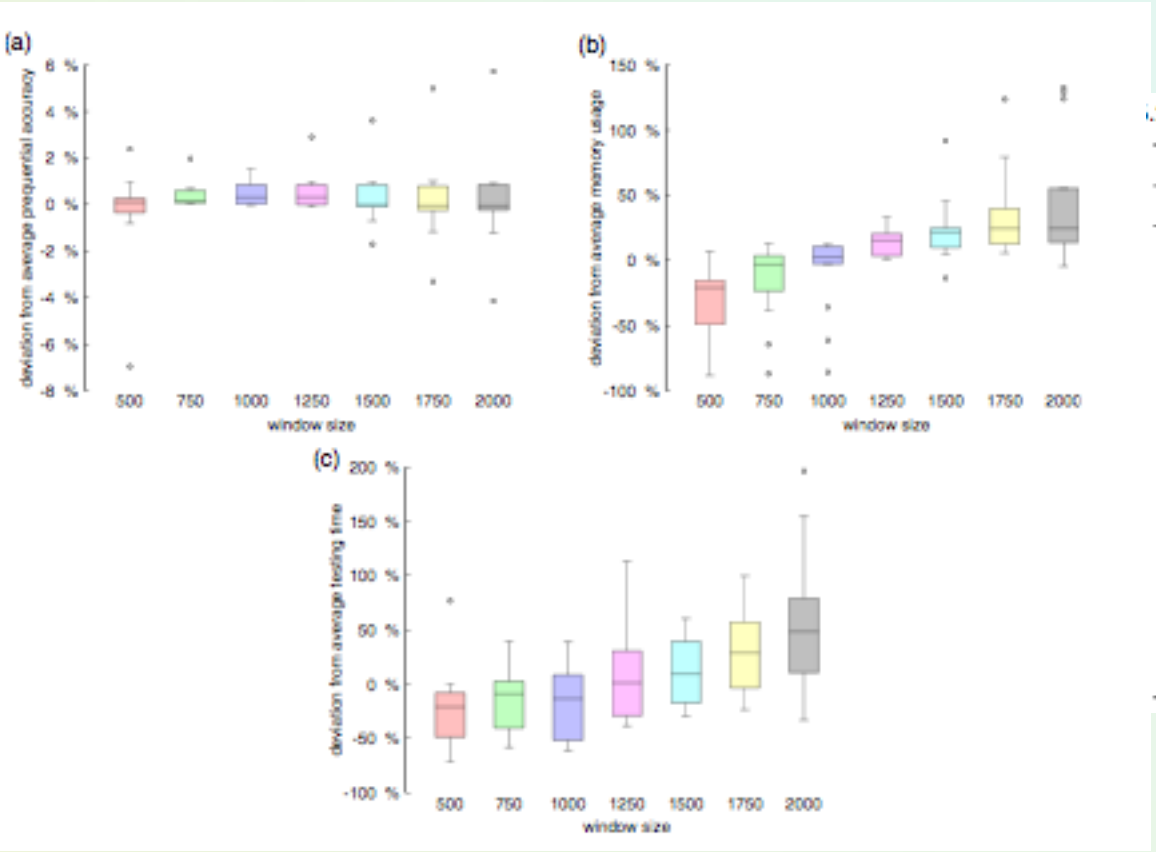
Experimental analysis

- 5 on-line algorithms: ACE, DWM, Lev, Bag, OAUE
- 15 datasets → the same as before
- Different types of drifts
- Evaluation wrt: time, memory, and accuracy

- Studying the impact of OAUE elements
  - Different size of sliding windows
  - Non-linear vs. linear weighting functions

    → Linear one better on fastest drifting streams

Sliding window *d* in OAUE



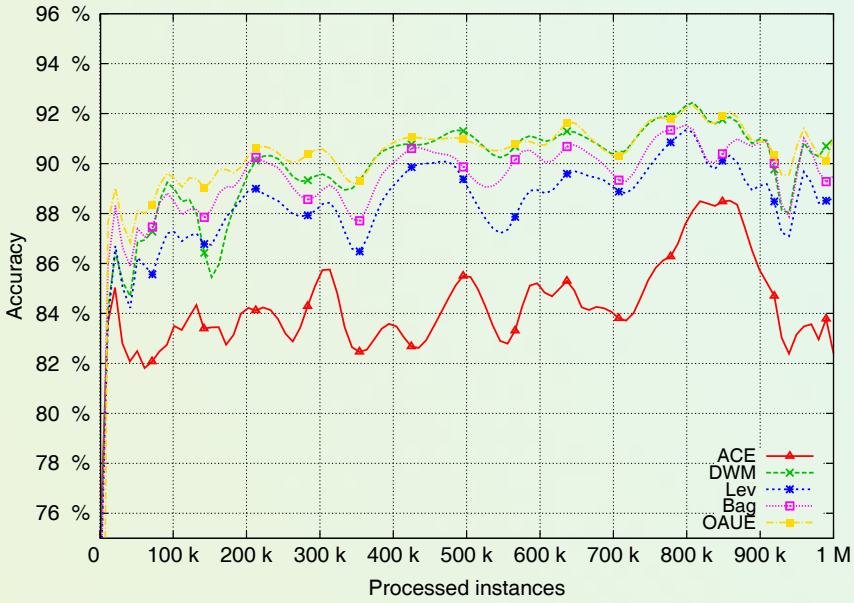
.2: Average prequential accuracy [%] of OAUE for different window

	Window size						
	500	750	1000	1250	1500	1750	2000
Airlines	67.50	66.93	67.03	67.12	66.72	66.33	66.23
CovType	90.07	90.85	90.91	91.08	91.43	91.51	91.58
Hyper_F	90.55	90.43	90.42	90.26	90.30	90.24	90.19
Hyper_S	89.05	89.04	88.94	89.00	88.98	88.92	88.97
LED_M	53.40	53.40	53.38	53.24	52.65	52.40	52.38
LED_{ND}	51.54	51.48	51.40	51.39	51.35	51.27	51.28
PAKDD	80.24	80.23	80.23	80.20	80.20	80.20	80.17
Poker	81.54	87.92	88.87	90.18	90.81	92.01	92.65
Power	15.73	15.58	15.54	15.34	15.27	15.23	14.87
RBF_B	96.78	97.59	97.83	97.84	97.96	98.00	97.90
RBF_{GR}	96.69	97.27	97.38	97.46	97.56	97.53	97.43
SEA_G	88.95	88.85	88.81	88.79	88.70	88.67	88.62
SEA_S	89.41	89.32	89.31	89.28	89.23	89.22	89.15
Tree_{SR}	46.23	46.05	45.86	45.21	44.39	43.66	43.28
Wave	84.34	85.25	85.47	85.58	85.53	85.50	85.49
Wave_M	83.86	84.75	84.85	84.87	84.86	84.73	84.66

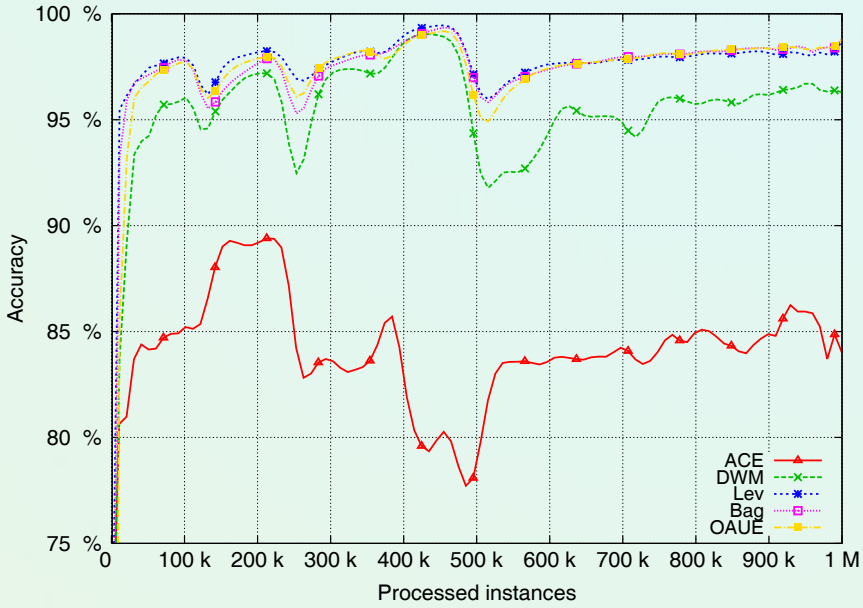
- Window size *d* → no impact on accuracy, but time and memory are proportional

More → D. Brzezinski, J. Stefanowski, 2014. Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams. Information Sciences, 265, 50-67.

Reacting to different changes

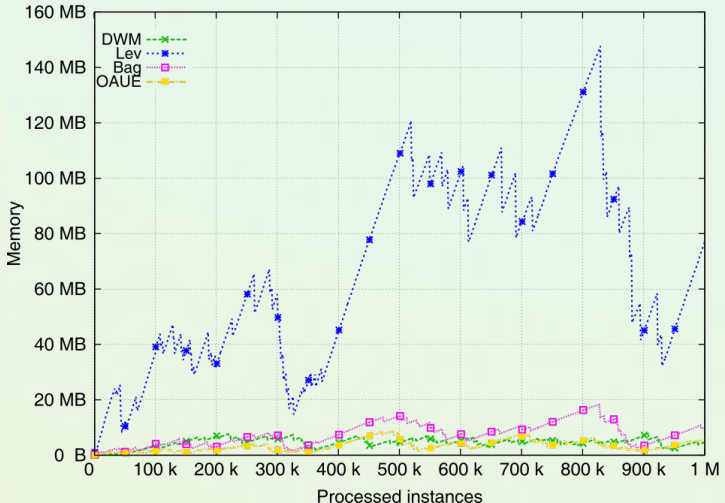


Classification accuracy on $Hyper_F$ (fast, suddenl changes)



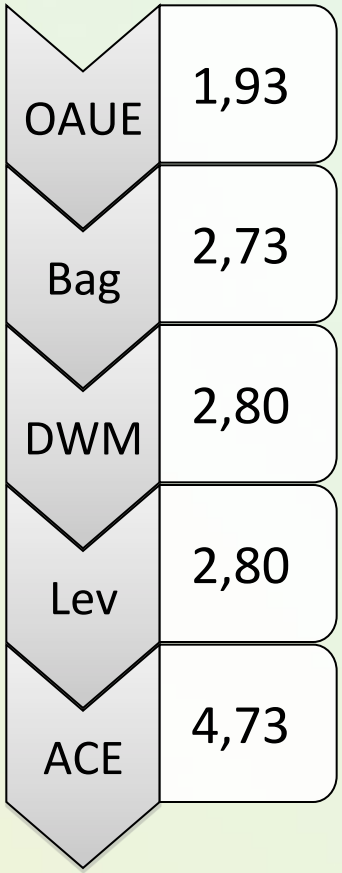
Accuracy on RBF_{GR} (slow, gradual changes)

Memory $Hyper_F$



Comparison of on-line classifiers

Ranks in Friedman test



Average prequential classification accuracy

Data	ACE	DWM	Lev	Bag	OAUE
Airlines	64,86	64,98	62,84	64,24	**67,02**
CovType	69,47	89,87	**92,11**	88,84	90,98
Hyp_F	84,34	89,94	88,49	89,54	**90,43**
Hyp_S	79,62	88,48	85,43	88,35	**88,95**
LED_M	46,45	53,34	51,31	53,33	**53,40**
LED_ND	39,80	51,48	49,98	**51,50**	51,48
PAKKDD	-	**80,24**	79,85	80,22	80,23
Poker	79,79	91,29	**97,67**	76,92	88,89
RBF_B	84,78	96,00	**98,22**	97,87	97,87
RBF_GR	84,16	95,49	**97,79**	97,54	97,42
SEA_G	85,97	88,39	**89,00**	88,36	88,83
SEA_S	85,98	89,15	89,26	88,94	**89,33**
Tree_SR	43,39	42,48	47,88	**48,77**	46,04
Wave	-	84,02	83,99	**85,51**	85,50
Wave_M	-	83,76	83,46	**84,95**	84,90

	ACE	DWM	Lev	Bag	OAUE
Memory	--	1.81	3.56	2.6	2.0
Time	2.5	1.81	4.81	3.2	2.6

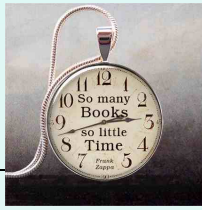
Ranks in Friedman test

Conclusions AUE → OAUE

- AUE → a hybrid ensemble for block streams
- Leaving pure block-based solutions → Novelty:
  - Incremental update of component classifiers
  - New weighting function
  - Strategies for creating strong components
- Experiments
  - Improve reaction to various drifts
  - Comparative study → 'best' average accuracy
    + faster and less memory consuming than the most competitive ensembles
- However → if not block-based processing of streams, …

More: D. Brzezinski, J. Stefanowski: Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm. IEEE Transactions on Neural Networks and Learning Systems, (2014)

Conclusions → towards on-line



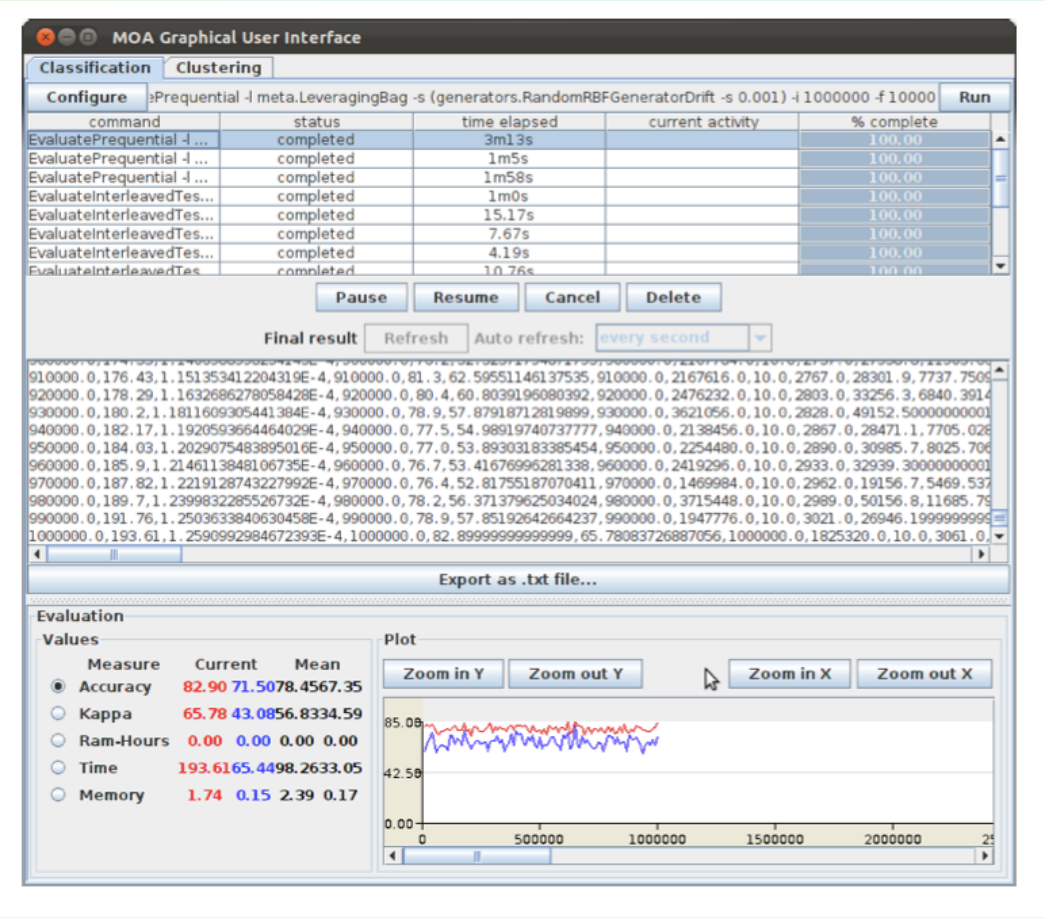
- In environments, where labels are available after each instance
  → the AUC new elements may be insufficient, so …
- Online generalization → OAUE
  - Trains (update) and weight component classifiers with each incoming example
  - Efficient (time & memory) formula for estimating errors
  - Overcome limits of too simple transformation strategies
- Experiments:
  - Parameters (d) of AUE not so influential
  - Comparative study → OAUR provides best averaged classification accuracy with quite good time & memory costs

More: D. Brzezinski, J. Stefanowski: . Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams. Information Sciences, Volume 265 (2014).

Software platforms

Massive Online Analysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams.

Albert Bifet and cooperators (Waikato University)



Classifiers for imbalanced data streams

- Class imbalance → still a challenge for static machine learning
- Learning from imbalanced, evolving data streams → even more difficult
  - Interaction of class imbalance and various drifts
  - Changes of the imbalance ratio over time
  - More complex changes of class distributes (data difficulty factors and …)
  - Appearing of new sub-concepts
- Still less attention in DS, limited research …
  - Many proposals adapts stream classifiers (chunk-based or online), e.g. online bagging variants
  - Use under-sampling or over-sampling to deal with class imbalances

Classifiers for imbalanced streams

- Uncorrelated bagging [J.Gao et al 2008-2014]
- The selective recursive approach (SERA) and recursive ensemble approach (REA) [Chen, He 2009-2014]
- Chunk-based with Hellinger distance [Chawla et al. 2009,2012]
- Extensions of Learn++.CDS → combination concept drift with re-sampling (SMOTE) [Polikar, Ditzler 2013]
- Ensemble Classifier for Skewed Data Streams (ECDS) [Zhang et al. 2011]
- Resampling in online learning ensembles from imbalanced streams [Wang, Minku, Yao 2013-2016]
- …

Gao et al. proposal of under-bagging

- Stream dived into blocks B1, B2, …Bj
- Bj contains **much less** minority examples than majority ones
- While building a new classifier from the current block Bj take all minority examples from the latest blocks Bk (k<j) and under-sample the majority examples from Bj -> it gives a new training set Ts
- *m* learning sets (for bagging) are sampled from Ts:
  - Minority examples are copies to all of these *m* sets
  - Majority examples are randomly propagated to one of *m* learning sets (disjoint)
- From each *m* set a new component classifier is constructed and added to the ensemble
  - A final prediction – voting like in bagging
- It resembles under-bagging ensembles from the static framework
- May also limit the number of previous blocks and components
- Not useful for more complex changes of the minority class

Selectively Recursive Approach (SERA)

S.Chen, H.He: SERA selectively recursive approach towards nonstationary imbalanced stream mining. Conf. JCNN 2009.

- Still block based ensemble but uses minority examples in a different way than in [Gao et al]
  - In the latest blocks it looks for the limited number the most similar (Mahalanobis distance) examples to ones in the current block Bj
  - Combine them with the majority examples in Bj like in an over-sampling way
- Construct bagging ensemble

REA extension

- Adding past minority examples being k-nearest neighbours of ones in the current block (may help with small sub-concepts)
- Use non-linear weighting function for components in the ensemble
- Weights reflect mean square errors on testing examples from the latest block (resembles AWE)

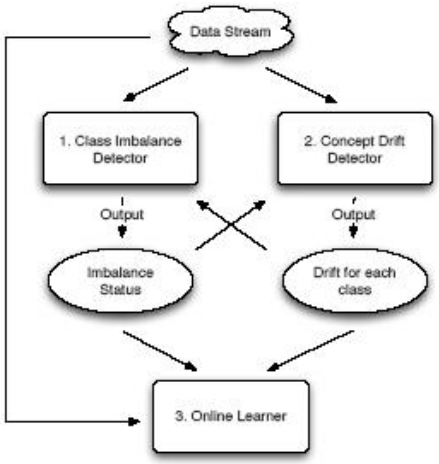
Online learning of multiple, drifting classes

Researchers: Shuo Wang, Leandro Minku, Xin Yao

- A learning framework for online class imbalance learning. IEEE CIEL 2013 and other papers

Another perspective – dynamic changes:

- Stream may contain several classes
- Classes may change their roles – majority becomes minority in a longer period of time,
- The number of minority classes may change – which classes are harder to be recognized?
- The imbalance ratio is evolving – what is the current imbalance ratio?



They promote their framework with modified online bagging and detectors

Class imbalance and drift detectors

- An identification of class imbalances (immediate accesss to true labels of examples)
- Monitor $w_k^{(t)}$ - the size percentage of each class c_k at time t:

$$w_k^{(t)} = \eta \cdot w_k^{(t-1)} + (1-\eta)[(x^t, c_k)]$$

- Three class labels – minority Y_{min}, majority Y_{maj} and normal-classes Y_{nom}
- Decide which class is imbalanced
  - For two classes the difference of their percentage sizes w > $\delta 1$
  - And the difference of their Recall (sensitivity) R> $\delta 2$
- The classes may be moved from the previous roles (second phase)

```
for i = 1 to |Y| − 1 do
  for j = i + 1 to |Y| do
    if w_j − w_i > δ1 and R_j − R_i > δ2
      Y_min ← Y_min ∪ {c_i}
      Y_maj ← Y_maj ∪ {c_j}
    end if
  end for
end for
```

Re-sampling online bagging

Choosing - **online bagging**

Depending on the class of new coming example (x, c_k) – tune the parameter λ of Poisson distribution depending on w_k percentage size of the class

$c_k \in Y_{min}$ then $\lambda = 1/w_k$ -> indirectly increases the chance for more copies / over-sampling of the minority class

$c_k \in Y_{maj}$ then $\lambda = 1 - w_k$ -> decreases the chance for drawing more copies / under-sampling of the majority class

$c_k \in Y_{nor}$ standard $\lambda = 1$

Input: label sets Y_{min}, Y_{maj} and Y_{nom}, an ensemble with M base learners, and current training example (x, c_k).

for each base learner f_m $(m = 1, 2, \ldots, M)$ **do**
  if $c_k \in Y_{min}$
    set $K \sim Poisson(1/w_k)$
  else
    set $K \sim Poisson(1)$
  end if
  update f_m K times
end for

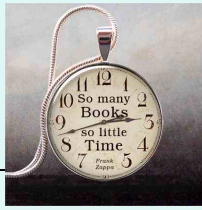
Over-sampling online bagging

Input: label sets Y_{min}, Y_{maj} and Y_{nom}, an ensemble with M base learners, and current training example (x, c_k).

for each base learner f_m $(m = 1, 2, \ldots, M)$ **do**
  if $c_k \in Y_{maj}$
    set $K \sim Poisson(1 - w_k)$
  else
    set $K \sim Poisson(1)$
  end if
  update f_m K times
end for

Under-sampling online bagging

Open issues



- Specific focused or general purpose techniques for handling drifts
  - Better understanding what forms of drift are handled by each detector or adaptation technique
- Provide insights into changes
  - Interpretability, local vs. global change
- Including additional knowledge in drift adaptations
  - Seasonal effects or temporal relationships
- Novel class detection or more structural changes
- Detectors for imbalanced changes (also other data changes than the global imbalance ratio)
- Evaluation issues
  - New measures, adaptability, multiple-criteria point of view
  - New testing procedures (controlled permutations, ..)
  - Unavailability of suitable public benchmark data sets

Open issues

Classification challenges

- Availability of ground truth in on-line
  - Delayed labels, obtained on demands, uncertain
- Semi-supervised, unsupervised approaches
- Complex and heterogeneous data representations
- Structured output or specific classification problems

Big Data special requirements

- Need more efficient time- and storage algorithms
- New platforms, e.g. SAMOA project

Knowledge Challenges

- Discover novel knowledge about how a domain evolves
- Understand how things change
- Monitor existing knowledge

Some References

- D. Brzezinski, J. Stefanowski, 2014. **Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm**. IEEE Transactions on Neural Networks and Learning Systems, Volume 25 (1), 81-94.
- D. Brzezinski, J. Stefanowski, 2014. **Combining Block-based and Online Methods in Learning Ensembles from Concept Drifting Data Streams**. Information Sciences, Volume 265, 50-67.
- D. Brzezinski, J. Stefanowski, 2013. **Classifiers for Concept-drifting Data Streams: Evaluating Things That Really Matter**. Proc. ECML PKDD 2013 Workshop Real-World Challenges for Data Stream Mining.
- D. Brzezinski, J. Stefanowski, 2012. **From Block-based Ensembles to Online Learners in Changing Data Streams: If- and How-To**. Proc. ECML PKDD 2012 Workshop on Instant Interactive Data Mining.
- D. Brzezinski, J. Stefanowski, 2011. **Accuracy Updated Ensemble for Data Streams with Concept Drift**. Proc. 6th International Conf. HAIS 2011, Part II, Volume 6679 of LNCS, Springer, 155-163.
- D. Brzezinski, J. Stefanowski, 2015. **Prequential AUC for Classifier Evaluation and Drift Detection in Evolving Data Streams**. New Frontiers in Mining Complex Patterns, LNCS Volume 8983, 87-101.
- Dariusz Brzezinski, Jerzy Stefanowski: Prequential AUC: Properties of the Area Under the ROC Curve for Data Streams with Concept Drift, Knowledge and Information Systems Journal (2017)
- M. Deckert, J. Stefanowski: **Comparing Block Ensembles for Data Streams with Concept Drift**. In: New Trends in Databases and Information Systems, Springer Comput. Intelligence, vol. 185, 2012, 69-78
- M. Deckert, J. Stefanowski: **RILL: Algorithm for learning rules from streaming data with concept drift**. Proc. ISMIS 2014, vol. 8502 of LNAI, 20-29.
- M.Kmieciak, J.Stefanowski: **Handling Sudden Concept Drift in Enron Message Data Streams**. Control and Cybernetics, vol. 40 no. 3, 2011, 667-695.

Check them at: www.cs.put.poznan.pl/jstefanowski or www.cs.put.poznan.pl/dbrzezinski

Thank you for your attention

Questions or comments ?

Contact, remarks:
Jerzy.Stefanowski@cs.put.poznan.pl
or www.cs.put.poznan.pl/jstefanowski