Tracking Concept Change with Incremental Boosting by Minimization of the Evolving Exponential Loss

Mihajlo Grbovic
mihajlo.grbovic@temple.edu

and

Slobodan Vucetic
slobodan.vucetic@temple.edu

September 6th 2011.

ECML PKDD 2011, Athens, Greece

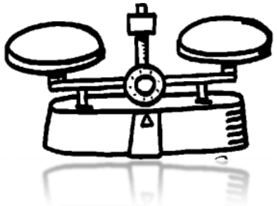
Department of Computer and Information Sciences
Center for Data Analytics and Biomedical Informatics
Temple University
Philadelphia, USA

Telephone: (215) 204-5535

Outline

- Introduction
  - Incremental Learning
  - Motivation for Model Reuse
  - Potential Applications of Incremental AdaBoost
- AdaBoost – Statistical View
  - Fitting an additive model through an iterative optimization of an exponential loss
- Incremental AdaBoost
  - IBoost Methodology
  - IBoost Flowchart
  - IBoost for Concept Change
- Related Work
- Experimental Results

Model Reuse



- **Challenge:** Learn an accurate model using training data set which changes over time

- **Naïve Approach:** Retrain the model from scratch each time the data set is modified (compitationally wasteful)

- **Incremental Learning**: process of updating the existing model when the training data set is changed

  - Particularly appealing for Online Learning, Active Learning, Outlier Removal and Learning with Concept Change

  - Many single-model algorithms are capable of incremental learning (e.g. linear regression, naïve Bayes, kernel perceptrons, SVM)

  - It is still an open challenge how to develop efficient and reliable **ensemble algorithms** for incremental learning

AdaBoost

- **Very popular** because of its ease of implementation and state of the art performance

- **Requires sequential training** of a large number of classifiers which can be costly

- **Rebuilding a whole ensemble** upon slight changes in training data can put an overwhelming burden to the computational resources:
  - e.g. Active Learning Query by Committee AdaBoost algorithm is not suitable for large-scale learning applications

- There exists a **high interest** for modifying boosting for incremental learning applications

  - Online Learning
  - Active Learning
  - Concept Change (Model Reuse)
  - Decremental Learning (Outlier Removal)

AdaBoost (Two Class Case)

- Developed using arguments from the statistical learning theory
- Alternate View: fitting additive model through iterative exponential cost optimization:

$$E_m = \sum_{i=1}^{N} e^{-y_i \cdot F_m(x_i)}$$

, where

$$F_m(x) = \sum_{j=1}^{m} \alpha_j f_j(x)$$

$F_m(x)$: current additive model - a linear combination of m base classifiers produced so far

AdaBoost (Two Class Case)

- Developed using arguments from the statistical learning theory
- Alternate View: fitting additive model through iterative exponential cost optimization:

$$E_m = \sum_{i=1}^{N} e^{-y_i \cdot F_m(x_i)}$$

, where

$$F_m(x) = \sum_{j=1}^{m} \alpha_j f_j(x)$$

$F_m(x)$: current additive model - a linear combination of m base classifiers produced so far

Given: Data set $D = \{(x_i, y_i), i = 1 \ldots N\}$, initial data weights $w_i^0 = 1/N$, number of iterations M

FOR $\boldsymbol{m} = 0$ **TO** $M-1$

(a) Fit $f_{m+1}(x)$ to data by minimizing:

$$J_{m+1} = \sum_{i=1}^{N} w_i^m I(y_i \neq f_{m+1}(x_i)) \qquad (1)$$

(b) Evaluate the quantities:

$$\varepsilon_{m+1} = \sum_{i=1}^{N} w_i^m I(y_i \neq f_{m+1}(x_i)) / \sum_{i=1}^{N} w_i^m \qquad (2)$$

and then

$$\alpha_{m+1} = \ln\left(\left(1 - \varepsilon_{m+1}\right) / \varepsilon_{m+1}\right) \qquad (3)$$

(c) Update the example weights:

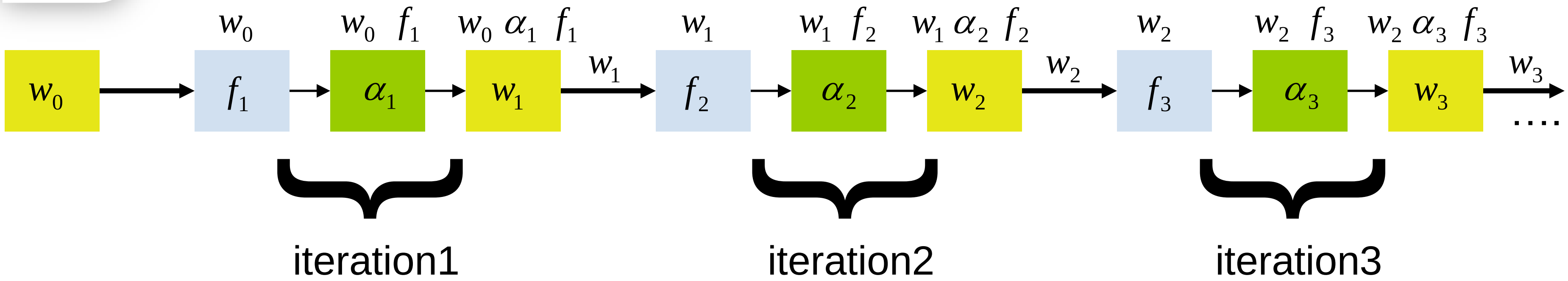
$$w_i^{m+1} = w_i^m e^{\alpha_{m+1} I(y_i \neq f_{m+1}(x_i))} \qquad (4)$$

END

Make predictions for new point x_{test} using:

$$\hat{y} = sign\left(\sum_{m=1}^{M} \alpha_m f_m(x_{test})\right) \qquad (5)$$

AdaBoost (Two Class Case)



Given: Data set $D = \{(x_i, y_i),\ i = 1 \dots N\}$, initial data weights $w_i^0 = 1/N$, number of iterations M

FOR $\boldsymbol{m} = 0$ **TO** $M-1$

(a) Fit $f_{m+1}(x)$ to data by minimizing:

$$J_{m+1} = \sum_{i=1}^{N} w_i^m I(y_i \neq f_{m+1}(x_i)) \quad (1)$$

(b) Evaluate the quantities:

$$\varepsilon_{m+1} = \sum_{i=1}^{N} w_i^m I(y_i \neq f_{m+1}(x_i)) \Big/ \sum_{i=1}^{N} w_i^m \quad (2)$$

and then

$$\alpha_{m+1} = \ln\left(\left(1 - \varepsilon_{m+1}\right) / \varepsilon_{m+1}\right) \quad (3)$$

(c) Update the example weights:

$$w_i^{m+1} = w_i^m e^{\alpha_{m+1} I(y_i \neq f_{m+1}(x_i))} \quad (4)$$

END

Make predictions for new point x_{test} using:

$$\hat{y} = sign\left(\sum_{m=1}^{M} \alpha_m f_m(x_{test})\right) \quad (5)$$

AdaBoost (Derived)

- Given the additive model $F_m(x)$ at iteration $m-1$ the objective is to find an improved one, $F_{m+1}(x) = F_m(x) + \alpha_{m+1} \cdot f_{m+1}(x)$, at iteration m. The cost function can be expressed

as:

$$E_{m+1} = \sum_{i=1}^{N} e^{-y_i (F_m(x_i) + \alpha_{m+1} f_{m+1}(x_i))} = \sum_{i=1}^{N} w_i^m e^{-y_i \alpha_{m+1} f_{m+1}(x_i)}$$

where:

$$w_i^m = e^{-y_i F_m(x_i)} \qquad (6)$$

By rearranging E_{m+1} we can obtain:

$$E_{m+1} = (e^{\alpha_{m+1}} - e^{-\alpha_{m+1}}) \sum_{i=1}^{N} w_i^m I(y_i \neq f_{m+1}(x_i)) + e^{-\alpha_{m+1}} \sum_{i=1}^{N} w_i^m \qquad (7)$$

- classifier $f_{m+1}(x)$ can be trained by minimizing (7) assuming α_{m+1} is fixed, as $f_{m+1}(x) = \arg\min_{f(x)} J_{m+1}$, where J_{m+1} is defined as (1)

- α_{m+1} can be determined by minimizing (7) assuming $f_{m+1}(x)$ is fixed. By setting $\partial E_{m+1} / \partial \alpha_{m+1} = 0$
  - the closed form solution can be derived as (3), where ε_{m+1} is defined as in (2)

- Before continuing to round $m+1$ the example weights w_i^m are updated as (4) by making use of (6)

Proposed Method (IBoost)

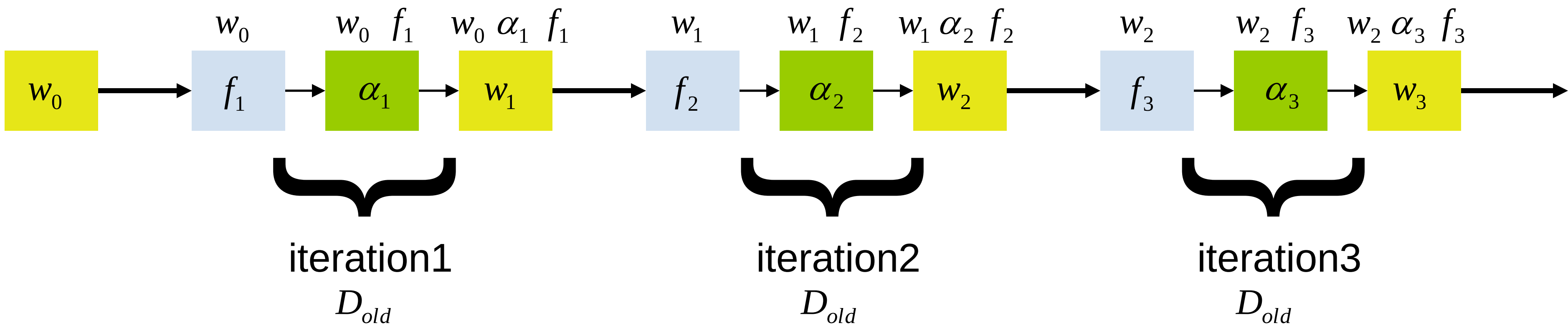
- Assume an $AdaBoost$ committee with m base classifiers $F_m(x)$ has been trained on data set D_{old}
- We wish to train a committee upon the data set changed to D_{new} by addition of N_{in} examples D_{in}, and removal of N_{out} examples, $D_{out} \subset D$
- The new training data set is $D_{new} = D_{old} - D_{out} + D_{in}$

Option 1: discard $F_m(x)$ and train a new ensemble from scratch

Option 2: reuse the existing ensemble

Proposed Method (IBoost)

- What prevents AdaBoost to be incrementally updated?

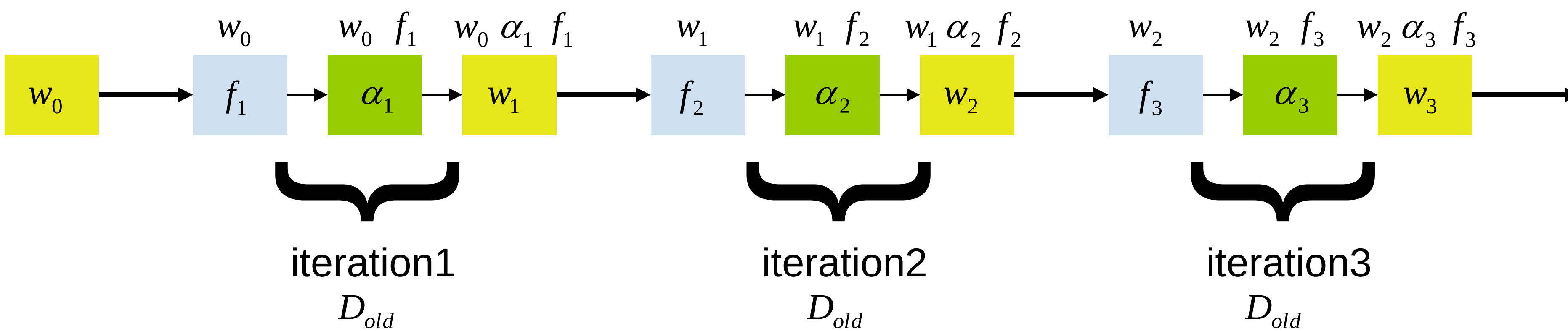


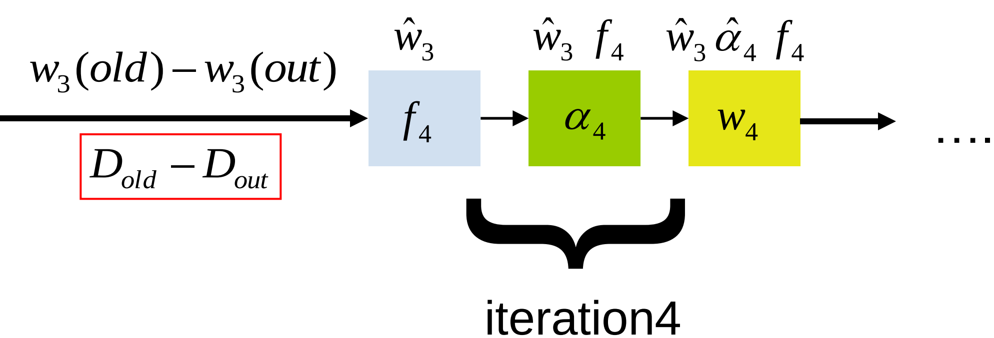
$$\xrightarrow[\boxed{D_{old} - D_{out}}]{w_3(old) - w_3(out)}$$

$$\hat{y} = sign(\sum_{m=1}^{3} \alpha_m f_m(x_{test}))$$

Proposed Method (IBoost)

- ❑ What prevents AdaBoost to be incrementally updated?

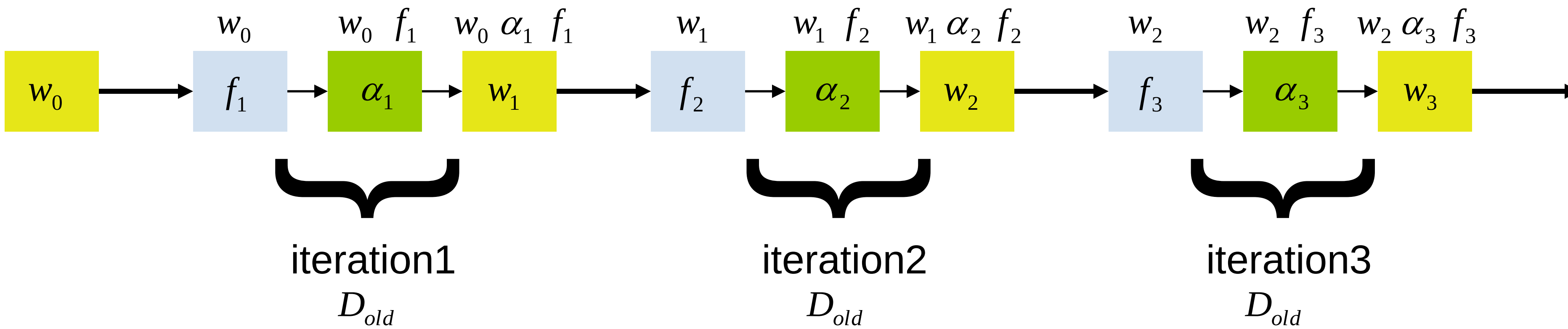




$$\hat{y} = sign\left(\sum_{m=1}^{4} \hat{\alpha}_m f_m(x_{test})\right)$$

Proposed Method (IBoost)

- What prevents AdaBoost to be incrementally updated?

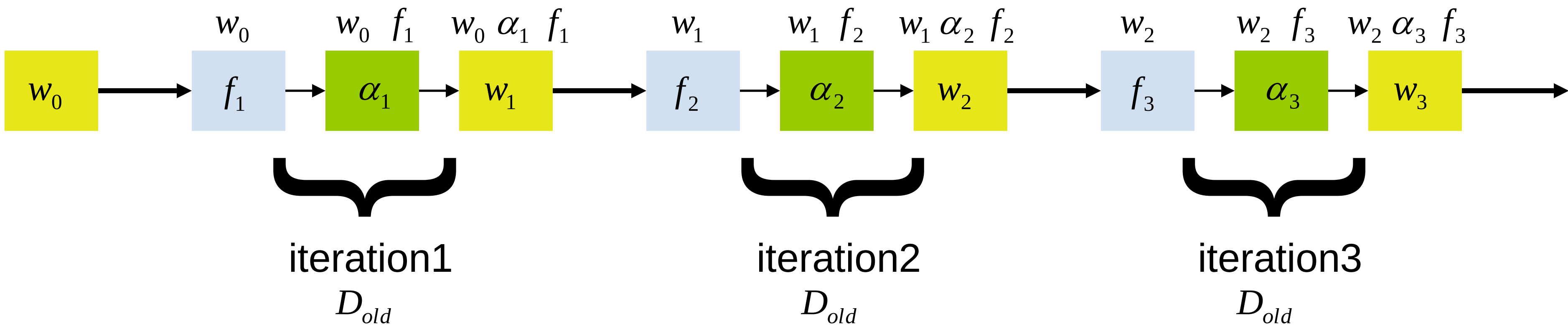


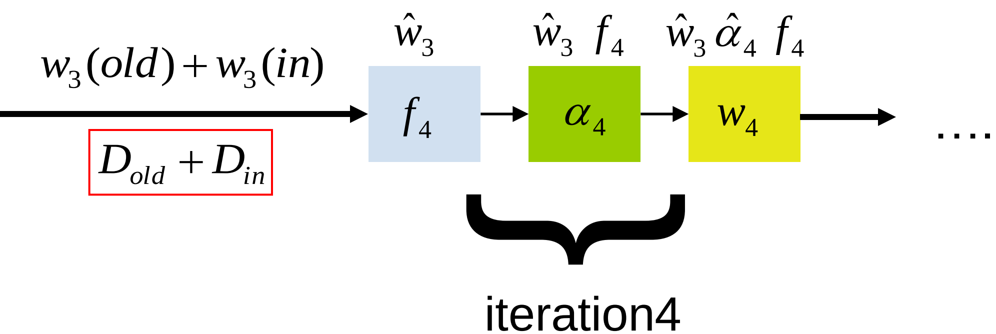
$$\xrightarrow[\boxed{D_{old} + D_{in}}]{w_3(old) + w_3(in)}$$

$$\hat{y} = sign(\sum_{m=1}^{3} \alpha_m f_m(x_{test}))$$

Proposed Method (IBoost)

- ❑ What prevents AdaBoost to be incrementally updated?





$$\hat{y} = sign(\sum_{m=1}^{4} \hat{\alpha}_m f_m(x_{test}))$$

Proposed Method (IBoost)

- Upon change of data set the cost function changes:

$$E_m^{old} = \sum_{i \in D_{old}} e^{-y_i \cdot F_m(x_i)} \rightarrow E_m^{new} = \sum_{i \in D_{new}} e^{-y_i \cdot F_m(x_i)}$$

- One could make several choices regarding reuse of the current ensemble $F_m(x)$:

1) Update $\alpha_t,\ t = 1 \ldots m$, to better fit the new data set

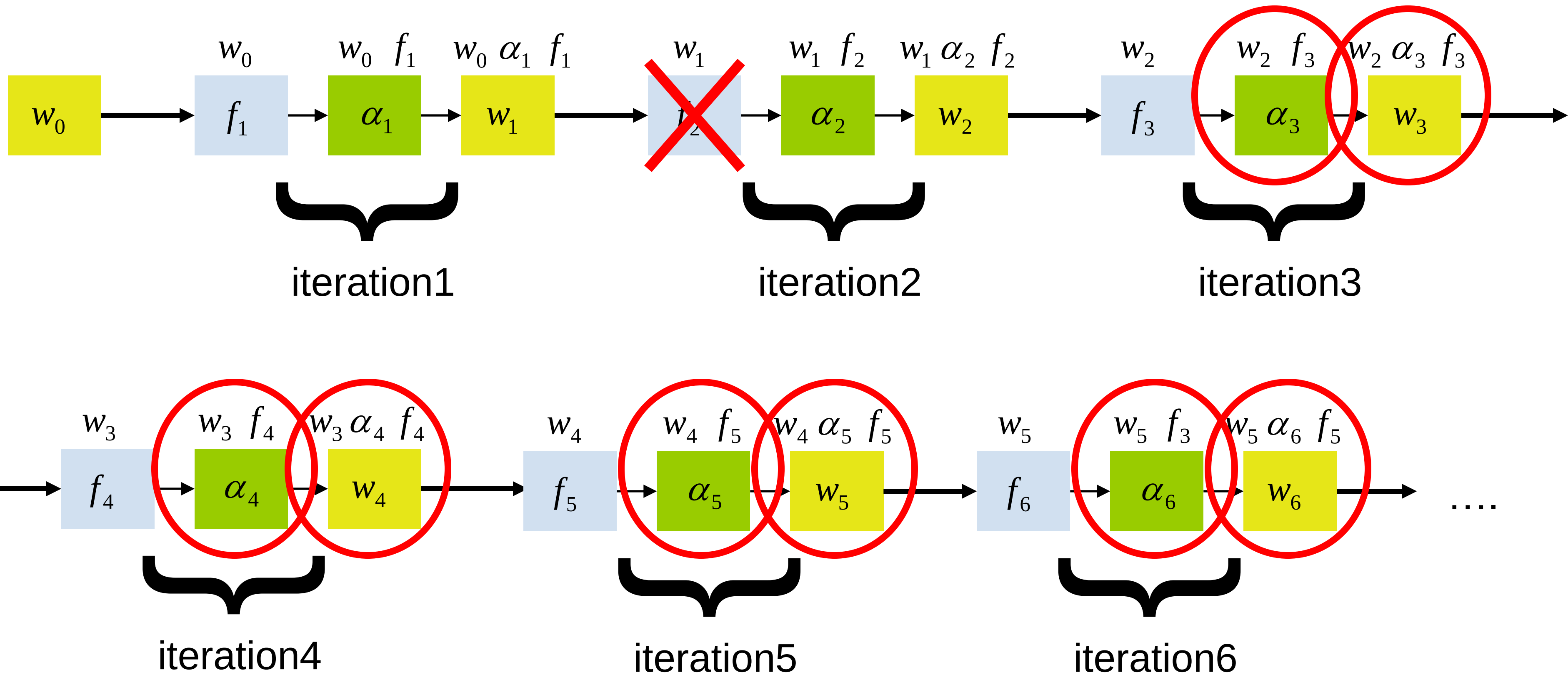
2) Remove base classifiers which no longer fit well to the data

3) Add a new base classifier f_{m+1} and its α_{m+1}

Require actions which keep the E_m^{new} minimized (confidence parameter updates and example weight updates/recalculation)

To avoid an unbounded growth: budget M

Proposed Method (IBoost)



Proposed Method (IBoost)

1. Update α_t, $t=1\ldots m$, to better fit the new data set (so that they minimize E_m^{new} for fixed base classifiers f_t, $t=1\ldots m$)

$$\alpha_j^{new} = \alpha_j^{old} + \eta \sum_{i \in D_{new}} y_i f_j(x_i) e^{-y_i \sum_{k=1}^{m} \alpha_k^{old} f_k(x_i)} \quad (8)$$

batch

$$\alpha_j^{new} = \alpha_j^{old} + \eta \cdot y_i f_j(x_i) e^{-y_i \sum_{k=1}^{m} \alpha_k^{old} f_k(x_i)} \quad (9)$$

stochastic

2. Potentially remove base classifiers

- that are underperforming: $\alpha<0$
- when budget is full: $\min(\alpha)$

3. Update example weights (three scenarios)

1) If α were unchanged since the last iter. use:

$$w_i^m = e^{\sum_{t=1}^{m} \alpha_t I(y_i \neq f_t(x_i))}, i \in D_{in} \quad (10)$$

2) If α were updated, use:

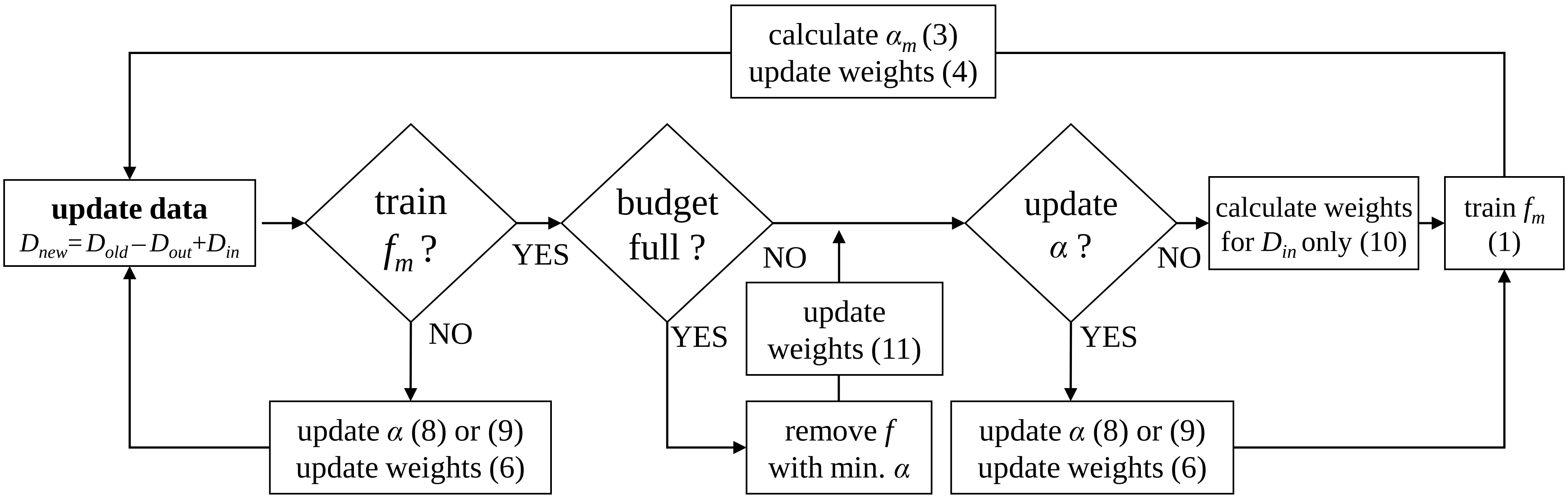
$$w_i^m = e^{-y_i F_m(x_i)} \quad (6)$$

3) If any base classifier f_j was removed, use:

$$w_i^m = w_i^{m-1} e^{-\alpha_j I(y_i \neq f_j(x_i))} \quad (11)$$

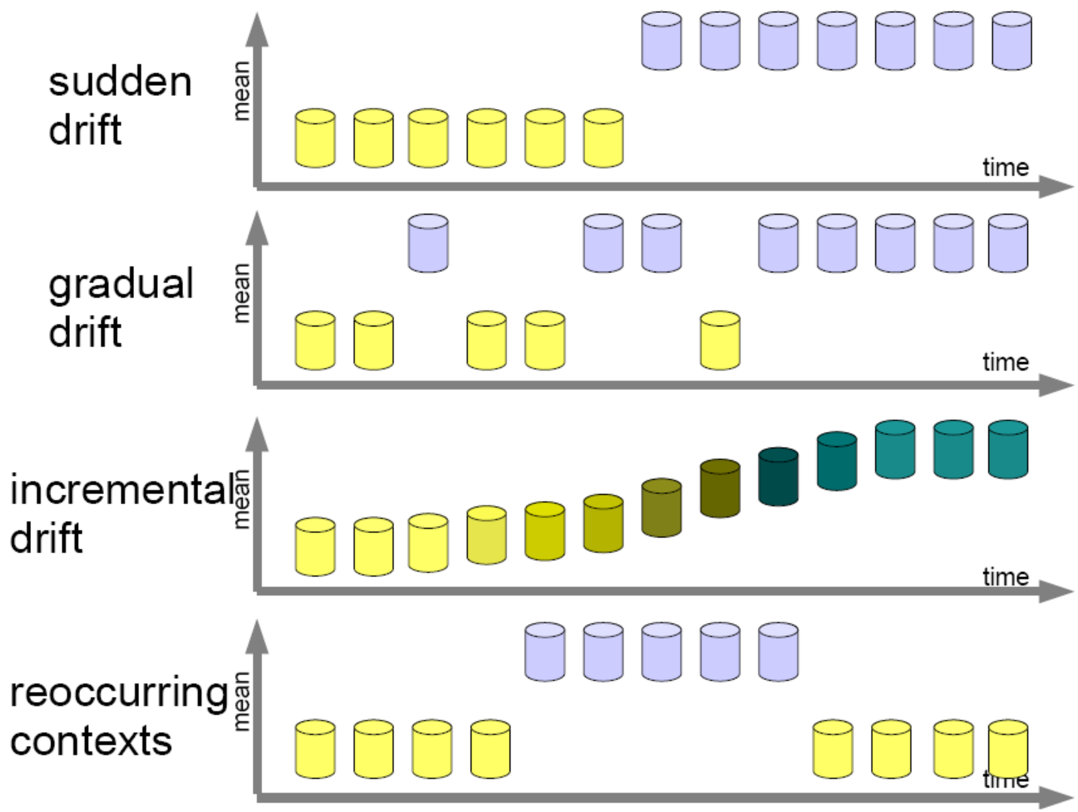
4. Add a new base classifier f_{m+1} and calculate its α_{m+1}

Proposed Method (IBoost)



Concept Change (Drift)

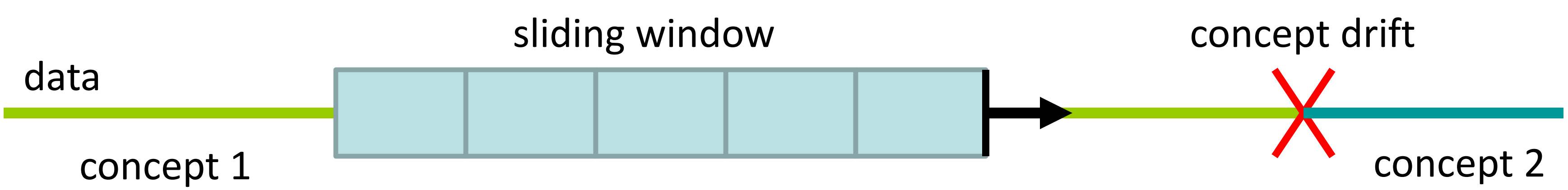
- Data stream in which the properties of the target value *y* change over time
- The change can happen in unforeseen ways and at a random time
- Drift Types:



Concept Change (Drift)

- General Approach: Online Learning using a sliding window

- Window size n presents a tradeoff between accuracy on the current concept and fast recovery from distribution changes



Concept Change (Drift)

- Popular: Adaptive supervised learning techniques (Adaptive Ensembles)

- Upadate criterion: How often to update the model?

    - depends on the properties of the data stream

    - depends on computational resources

    - one solution: after enough incoming data examples are missclassified

IBoost Variant for Concept Change

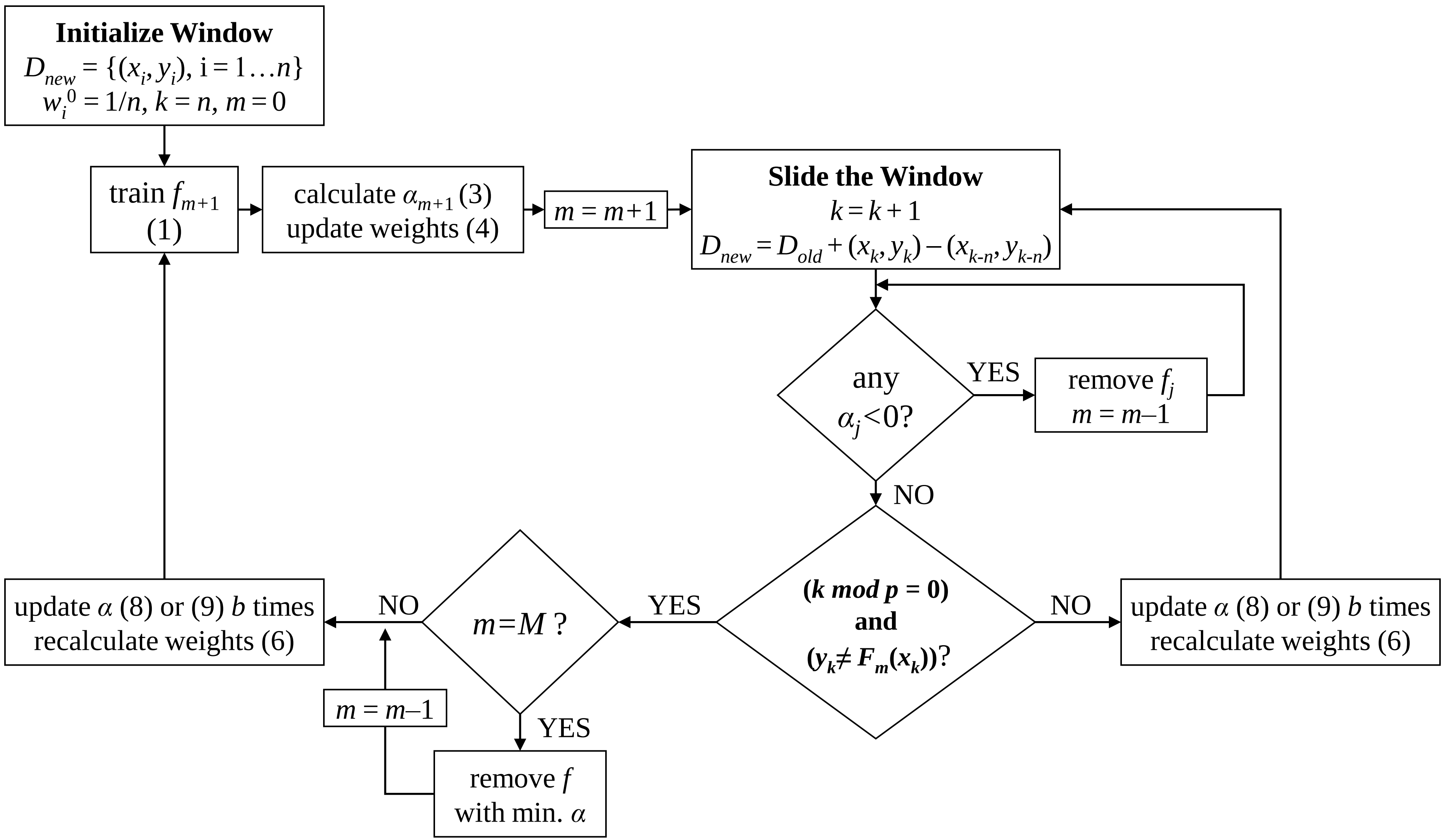
Input:

1. data stream $D = \{(x_i, y_i), i = 1 \ldots N\}$
2. window size n
3. budget M
4. frequency of model addition p
5. number of gradient descent updates b

Parameters (M, n, p and b) are intuitive and easy to select for a specific application:

- n is a tradeoff between accuracy on the current concept and fast recovery
- Larger p values can speed-up the process with slight decrease in performance
- Larger M imporves accuracy at cost of prediction, model update and storage
- b is a tradeoff between accuracy, concept change recovery and time

IBoost Variant for Concept Change



Related Work

IBoost will be compared to:

- Non-incremental *AdaBoost* (retrained)
- *Online Coordinate Boost* (**OCB**)
- ***OnlineBoost***
- Two *OnlineBoost* modifications for concept change (***NSOnlineBoost*** and **FLC**)
- *Fast and Light Boosting* (**FLB**)
- *Dynamic Weighted Majority* (**DWM**)
- *AdWin Online Bagging* (**AdWin Bagg**)

Characteristics	*IBoost*	*Online Boost*	*NSO Boost*	*FLC*	*AdWin Bagg*	*OCB*	*DWM*	*FLB*
Change Detector Used				•	•			•
Online Base Classifier Update		•	•	•	•		•	
Classifier Addition and Removal	•		•		•		•	•
Sliding Window	•		•	•	•			•

Related Work

❑ *OnlineBoost*

- Initial base models $f_j, j = 1 \ldots m$: assigned weights $\lambda_j^{sc} = 0$ and $\lambda_j^{sw} = 0$
- A new example (x_i, y_i) : assigned an initial example weight $\lambda_d = 1$
- *Poisson* distribution used : update each f_j $k = Poisson\ (\lambda_d)$ times using (x_i, y_i)
- If $f_j(x_i) = y_i$: update $\lambda_d = \lambda_d / 2(1 - \varepsilon_j)$ and $\lambda_j^{sc} = \lambda_j^{sc} + \lambda_d$
- Otherwise: $\lambda_d = \lambda_d / 2\varepsilon_j$ and $\lambda_j^{sw} = \lambda_j^{sw} + \lambda_d$, where $\varepsilon_j = \lambda_j^{sw} / (\lambda_j^{sw} + \lambda_j^{sc})$
- Update the next base model f_{j+1}, etc.
- Parameters α obtained using (3), predictions are made using (5)

Related Work

❑ *Online Coordinate Boost* (**OCB**)

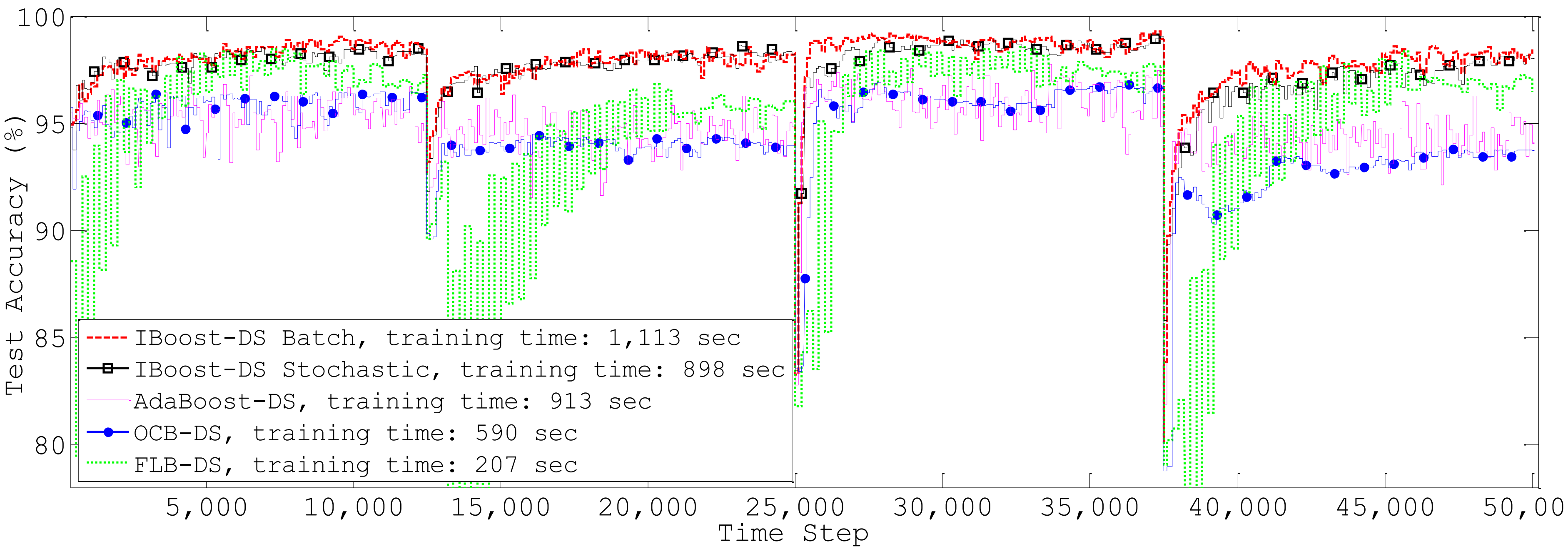
- Base models f_j, $j = 1 \ldots m$, trained offline using some initial data
- Parameters α_j, $j = 1 \ldots m$, and sums of weights of correctly and incorrectly classified examples (λ_j^{sc} and λ_j^{sw}, respectively) also provided
- A new example (x_i, y_i) : find the appropriate updates $\Delta\alpha_j$ for α_j such that the *AdaBoost* loss with the addition of (x_i, y_i) is minimized
- $\Delta\alpha_j$ cannot be found in the closed form, closed form solution that minimize the approximate loss is derived
- Such optimization requires keeping and updating the sums of weights ($\lambda_{(j,l)}^{sc}$ and $\lambda_{(j,l)}^{sw}$) which involve two weak hypotheses j and l and introduction of the order parameter o

Data Sets

Data Set	Drift Type	Train Size	Test Type	Test Size
SEA	Sudden	50,000	Hold Out	10,000
Santa Fe	Incremental	10,000	Hold Out	2,475
LED	Rigorous	1,000,000	Test Then Train	-
RBF	Gradual	1,000,000	Test Then Train	-

Results

SEA data Set: 4 concepts (**Sudden Drifts**)
Test on hold-out data from current concept



Decision Stumps $M = 200$, $n = 200$

Results

SEA data Set – Decision Stumps

Algorithm		window size $n = 200$					budget $M = 200$				
		budget M					window size n (n_{ocb}, n_{fb})				
		20	50	100	200	500	100	200	500	1,000	2,000
Stochastic	*test accuracy* (%)	94.5	96.4	96.7	**97.1**	97.5	96.9	97.1	97.3	97.5	98
IBoost	*recovery* (%)	92.5	93.1	93.3	**93.5**	93.4	93.4	93.5	92.4	90.1	89.6
$b = 5$	*time* (*s*)	39	90	183	**372**	751	221	372	396	447	552
Batch	*test accuracy* (%)	95.9	97.4	97.8	**97.9**	98	97.2	97.9	98.1	98.3	98.5
IBoost	*recovery* (%)	91.5	92.1	92.9	**92.5**	93.4	92.8	92.5	91.2	88.8	88.4
$b = 5$	*time* (*s*)	77	188	401	**898**	2.1K	801	885	1K	1.7K	2.3K
	test accuracy (%)	94.5	95	95	**94.9**	94.9	92.8	94.9	96.7	97	97.5
AdaBoost	*recovery* (%)	92	92.1	92.2	**91.9**	91.9	91.7	91.9	89.9	88.1	86.3
	time (*s*)	91	192	432	**913**	2.1K	847	913	1K	1.3K	1.8K
	test accuracy (%)	92.7	93.9	94.3	**94.4**	94.1	91.3	94.4	95.4	95.8	96.8
OCB	*recovery* (%)	84.3	86.4	89.8	**91.2**	91.2	88.7	91.2	90.1	84.4	93.5
	time (*s*)	47	120	259	**590**	2K	584	590	567	560	546
	test accuracy (%)	82.6	89.4	92.9	**94.4**	94.9	94.7	94.4	90.5	87.5	83.4
FLB	*recovery* (%)	82.3	85.3	86.1	**84.7**	84.9	85.2	84.7	83.8	83.5	81.9
	time (*s*)	73	104	156	**207**	435	183	207	262	390	456

Results

SEA data Set – Decision Stumps

Algorithm		window size $n = 200$					budget $M = 200$				
		budget M					window size n				
		20	50	100	200	500	100	200	500	1,000	2,000
Stochastic	*test accuracy* (%)	94.5	96.4	96.7	**97.1**	97.5	96.9	97.1	97.3	97.5	98
IBoost	*recovery* (%)	92.5	93.1	93.3	**93.5**	93.4	93.4	93.5	92.4	90.1	89.6
$b = 5$	*time* (*s*)	39	90	183	**372**	751	221	372	396	447	552
Batch	*test accuracy* (%)	95.9	97.4	97.8	**97.9**	98	97.2	97.9	98.1	98.3	98.5
IBoost	*recovery* (%)	91.5	92.1	92.9	**92.5**	93.4	92.8	92.5	91.2	88.8	88.4
$b = 5$	*time* (*s*)	77	188	401	**898**	2.1K	801	885	1K	1.7K	2.3K
	test accuracy (%)	94.5	95	95	**94.9**	94.9	92.8	94.9	96.7	97	97.5
AdaBoost	*recovery* (%)	92	92.1	92.2	**91.9**	91.9	91.7	91.9	89.9	88.1	86.3
	time (*s*)	91	192	432	**913**	2.1K	847	913	1K	1.3K	1.8K

Recovery (%): average test accuracy on the first 600 examples after introduction of new concept

- IBoost Batch was more accurate but significantly slower than IBoost Stochastic
- Fastest recovery for all three concept changes was achieved by IBoost Stochastic

- Increase in budget M: resulted in larger training times and accuracy gain for all algorithms
- Increase in window size n: improves performance at cost of increased training time and slower recovery increases recovery performance gap between IBoost and AdaBoost, while reduces the test accuracy gap

Results

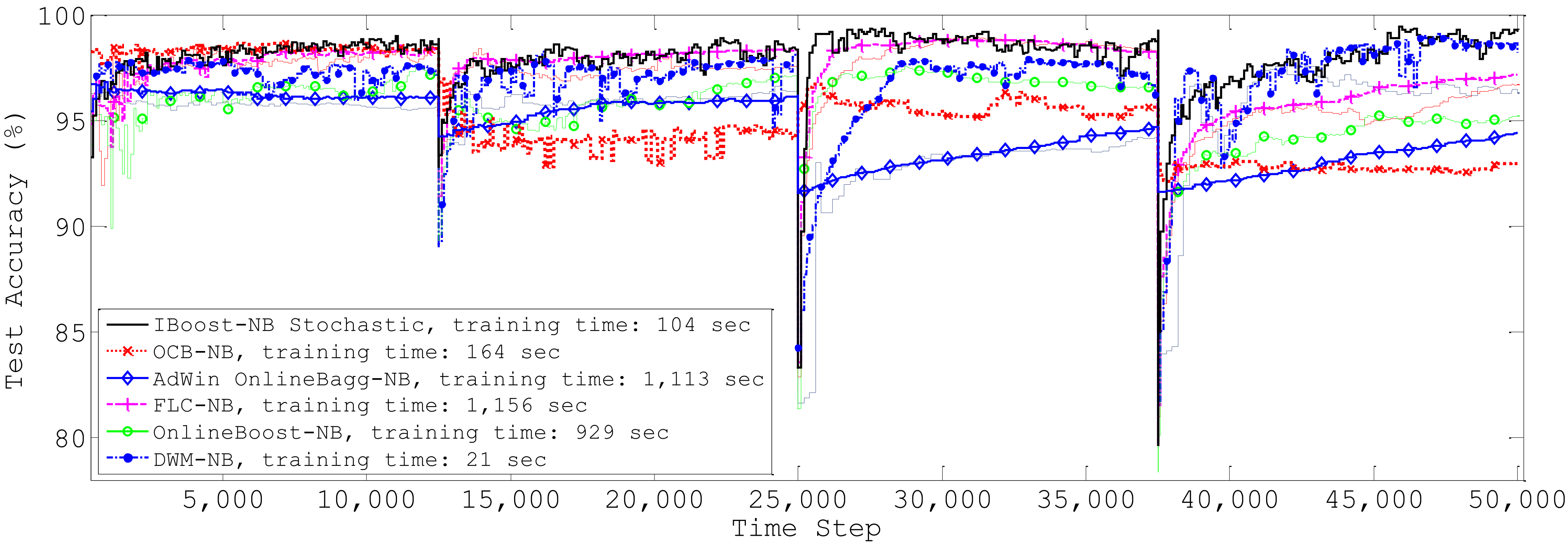
SEA data Set – Decision Stumps

Algorithm	$M = 200$ $n = 200$	$p = 1$			$b = 1$		
		$b = 1$	$b = 5$	$b = 10$	$p = 10$	$p = 50$	$p = 100$
IBoost Stochastic	test accuracy (%)	96.7	**97.1**	97.4	96.5	94.7	93.1
	recovery (%)	93.1	**93.5**	93.7	92.8	92.7	92.1
	time (s)	201	**372**	635	104	45	22
IBoost Batch	test accuracy (%)	97.6	**97.9**	98.2	97.1	95.6	93.7
	recovery (%)	92.3	**92.5**	92.9	92.6	91.6	91.4
	time (s)	545	**898**	1.6K	221	133	96

- Bigger values of b improved the performance at cost of increasing the training time
- Bigger values of p degraded the performance (some just slightly, e.g. $p = 10$) coupled with big time savings

Results

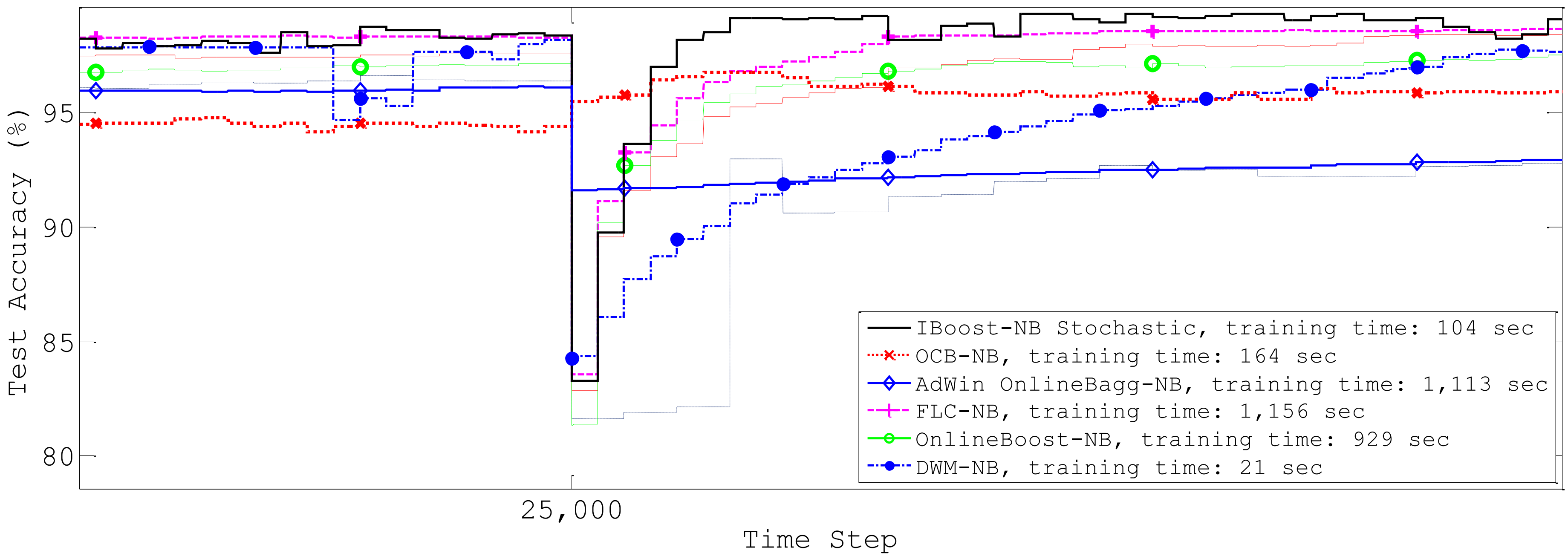
SEA data Set: 4 concepts (**Sudden Drifts**)
Test on hold-out data from current concept



Naïve Bayes $M = 50,\ n = 200,\ p = 1,\ b = 5$

Results

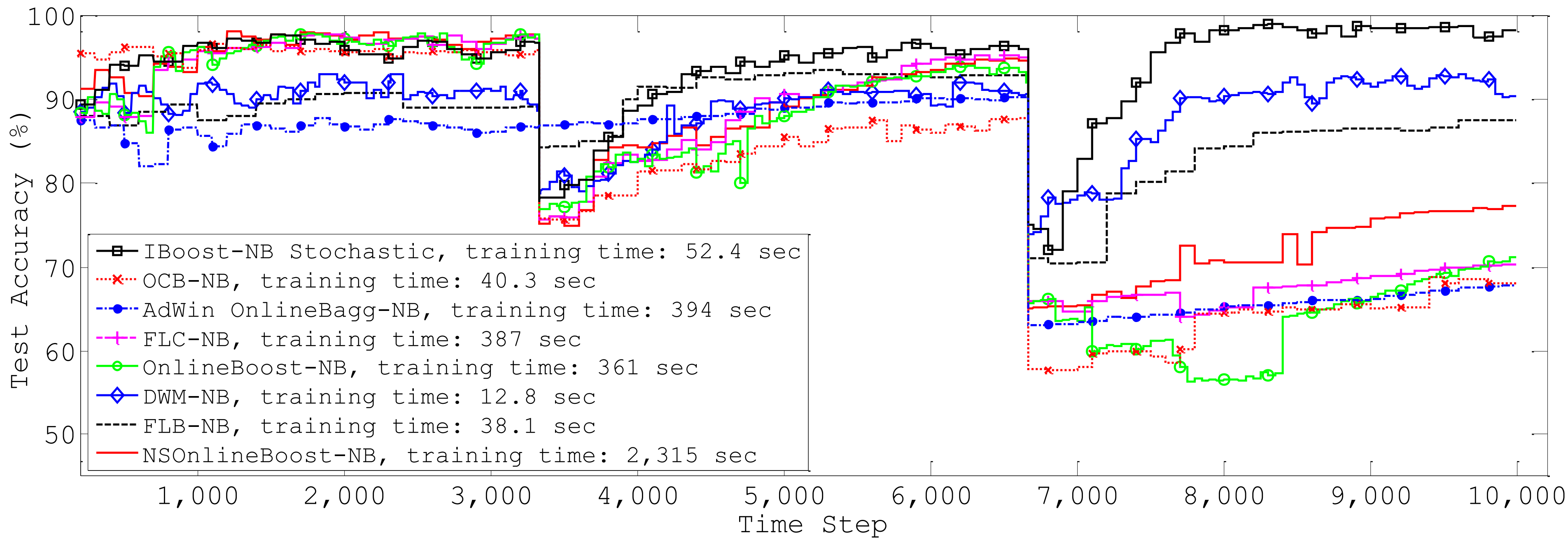
SEA data Set: 4 concepts (**Sudden Drifts**)
Test on hold-out data from current concept



Naïve Bayes $M = 50, n = 200, p = 1, b = 5$

Results

SantaFE data Set: 3 concepts (**Incremental Drifts**)
Test on hold-out data from current concept



Naïve Bayes $M = 50,\ n = 200,\ p = 1,\ b = 5$

Results

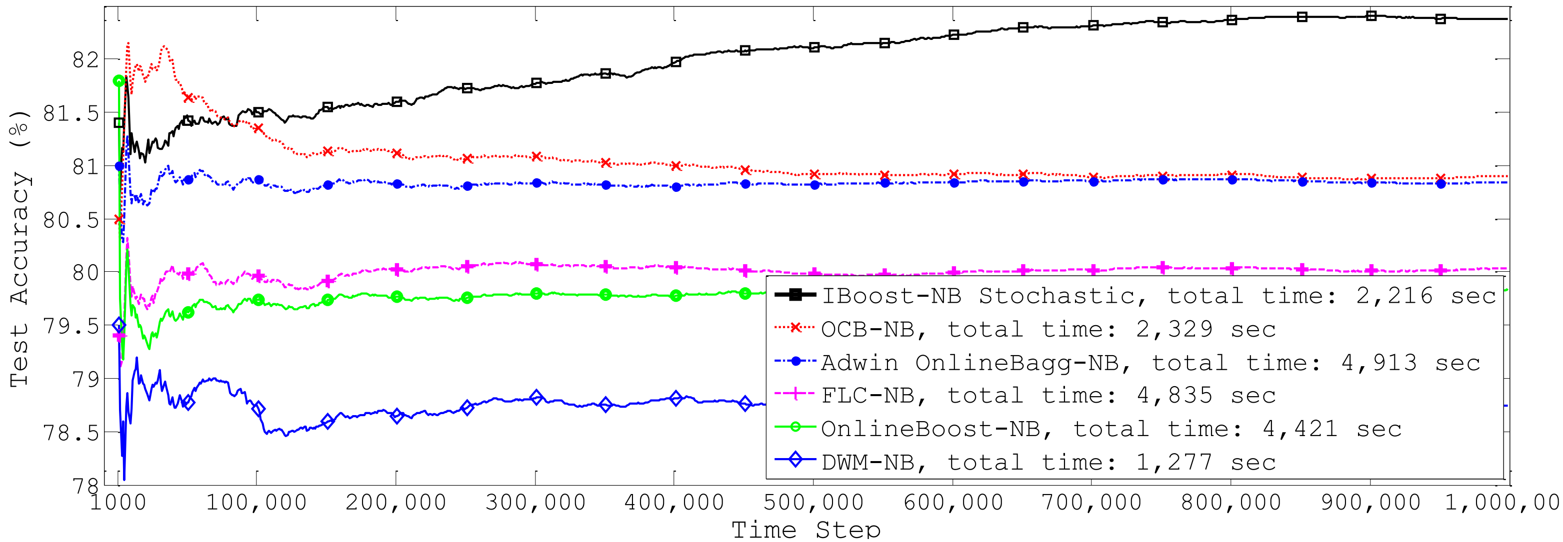
Performance summary on SEA and Santa Fe data sets based on the test accuracy (%)

Data Set	*IBoost Stoch.*	*Online Boost*	*NSOBoost*	*FLC*	*AdWinBagg*	*OCB*	*DWM*	*FLB*
SEA	**97.95**	95.6	96.9	97.35	94.5	95.2	96.9	94.9
Santa Fe	**94.1**	81.8	85.1	83.4	80	80.6	88.8	87.6

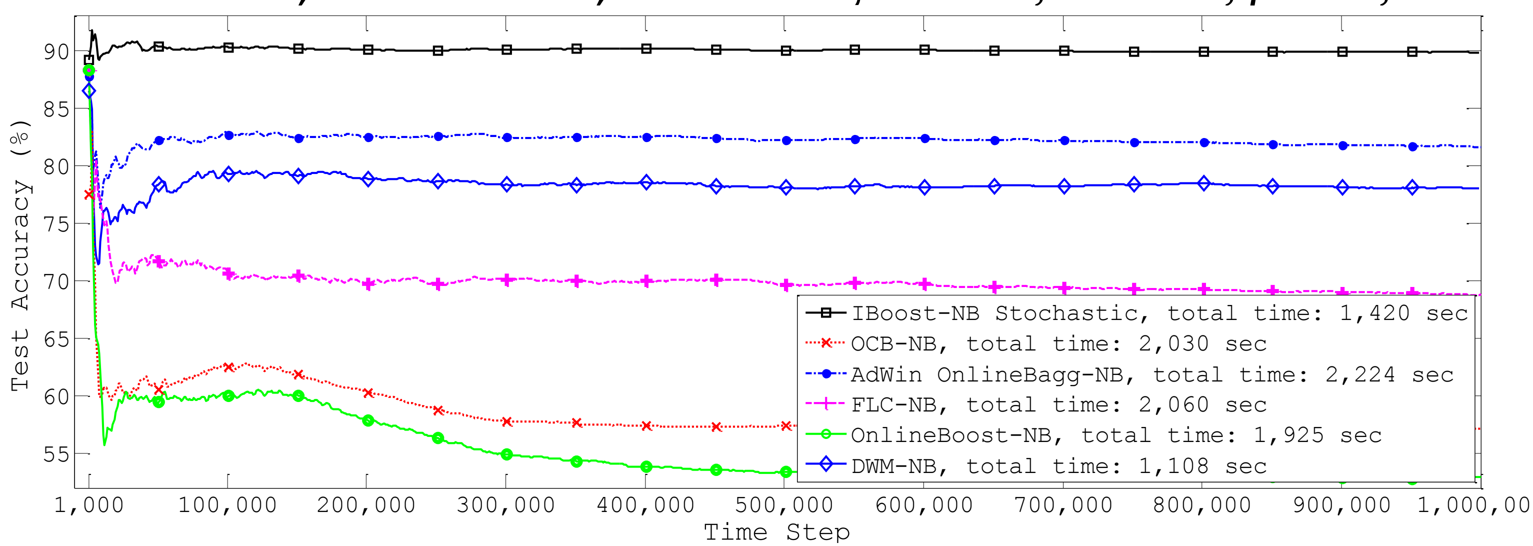
Results

Test Method: Test then Train

LED Data Set, 10% noise, 4 drifting attributes, $M = 20$, $n = 200$, $p = 10$, $b = 1$



RBF Data Set, 10 centroids, drift 0.001, $M = 20$, $n = 200$, $p = 10$, $b = 1$



Remarks

- RBF and LED Data Generated Using: MOA (massive online analysis)
  http://moa.cs.waikato.ac.nz/

- Experiments Performed in Matlab

- Code available soon

Conclusion

- We proposed an extension of AdaBoost to incremental learning
- The new algorithm was evaluated on concept change applications
- Experiments show that IBoost is more accurate, resistant, efficient than the original AdaBoost and previously proposed algorithms
- Future Work:
  - Extend IBoost to perform multi-class classification
  - Combine it with the powerful AdWin change detection technique
  - Experiment with Hoeffding Trees as base classifiers

THANK YOU