A Stream of Algorithms for Concept Drift

Mark Maloof

Department of Computer Science
Georgetown University
Washington, DC 20057-1232
http://www.cs.georgetown.edu/~maloof

Heilbronn Annual Conference

21 September 2012

Introduction

- Concept drift is a setting in which a machine learner must acquire a target concept that drifts or changes
- Such learners must balance reactivity and stability
- In 1986, Schlimmer and Granger published a paper on Stagger, which introduced the problem of concept drift and presented the first learning method designed to cope with drift
- Over the past twenty five years, there has been much empirical and theoretical work on drift and on related issues (e.g., recurring contexts)
- This talk is an overview of some of the empirical work that has appeared in the literature over this period

Outline

- Machine Learning
- Concept Drift
- Approaches
  - single-model methods
  - meta-learning methods
  - ensemble methods
- Evaluation
- Problems, Data Sets, and Selected Results
- Recent and Future Directions

Collaborators

- Ryszard Michalski
- Will Headden
- Zico Kolter
- Stephen Bach
- Amin Teymorian

Machine Learning

Or Pick Your Favorite Term...

- Given data, develop or use computational methods to build models that
  1. predict something about new data
  2. provide a better understanding of the data itself
- This talk is about the first effort

Classification

- Let $\mathcal{X}$ be an n-dimensional **input space**
- Let $\mathcal{Y} = \{-1, +1\}$ be **output labels**
- Let $f : \mathcal{X} \to \mathcal{Y}$, where f is an unknown **target concept** (or function)
- We have a sample of **training examples**: $S = \{\vec{x}, y\}_{i=1}^{m}$.
- Use S to find a hypothesis h such that $h \approx f$.
  - hypothesis: also model, **concept description**
- Minimize error on S: $\epsilon = \frac{1}{m} \sum_m \#\{h(\vec{x}) \neq y\}$
- We really care about **generalization**: We really care about the error (or accuracy) on previously unseen examples
- Once the hypothesis achieves acceptable accuracy, we can use it to classify new **observations** (i.e., $\vec{x}$)

Learning Methods

- A learning method or a classification method consists of three components:
  1. concept description language: the language used to build models
  2. learning element: uses training examples to induce a model
  3. performance element: uses the model to output a prediction for an observation
- Critically: the concept description language defines what models the learning element can build

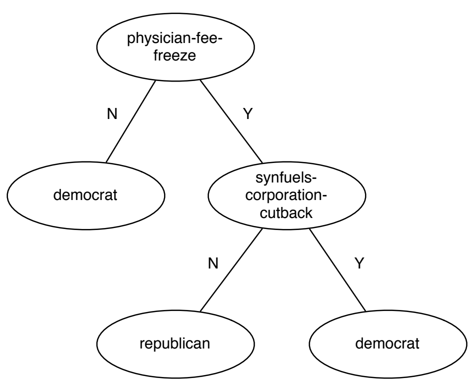
Example of Classification

- Task: predict political party based on voting record
- Data Set: 1984 US Congressional Voting Record

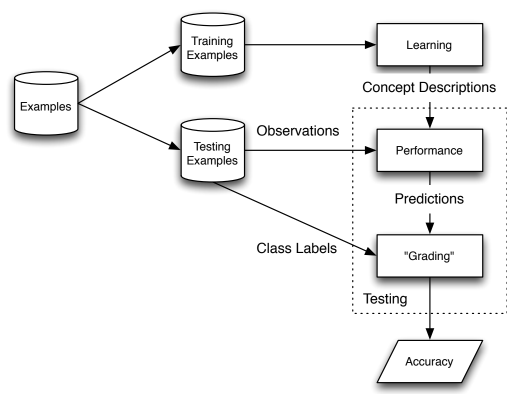
physician-fee-freeze	mx-missile	immigration	(12 others)	crime	party
n	n	y		n	democrat
n	n	y		y	democrat
...	...	...	...	...	...
y	y	n		y	republican
y	n	y		y	republican

- Rule:
  if (physician-fee-freeze = y) **and** (synfuels-corporation-cutback = n)
  then party = republican; **otherwise**, party = democrat

Example of a Decision Tree



Estimating True Accuracy



Estimating True Accuracy

k-fold Cross-validation

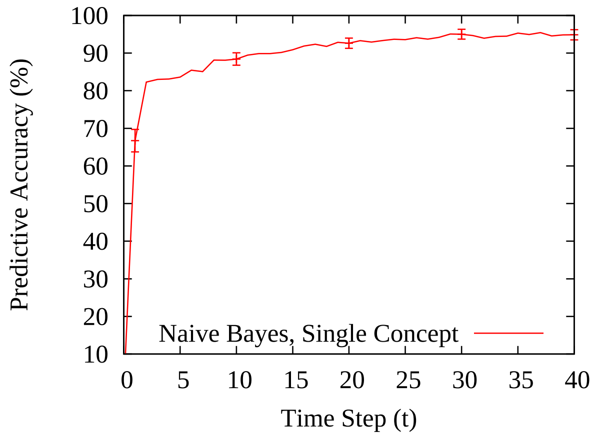
input Examples, Classifier, k
Partition $\leftarrow$ Examples.randomlyPartition(k)
Accuracy $\leftarrow 0$
for $i \leftarrow 1, k$ **do**
    TestExamples $\leftarrow$ Partition[i]
    TrainExamples $\leftarrow \bigcup_{j \neq i}$ Partition[j]
    Classifier.train(TrainExamples)
    Accuracy += Classifier.test(TestExamples)
end for
output Accuracy/k

Estimating True Accuracy

Performance Curve

input Examples, Classifier, k
Partition $\leftarrow$ Examples.randomlyPartition(k)
TestExamples $\leftarrow$ Partition[k]
for $i \leftarrow 1, k-1$ **do**
    TrainExamples $\leftarrow$ TrainExamples $\cup$ Partition[i]
    Classifier.train(TrainExamples)
    Accuracy = Classifier.test(TestExamples)
    output Accuracy
end for

Estimating True Accuracy
Example of a Performance Curve



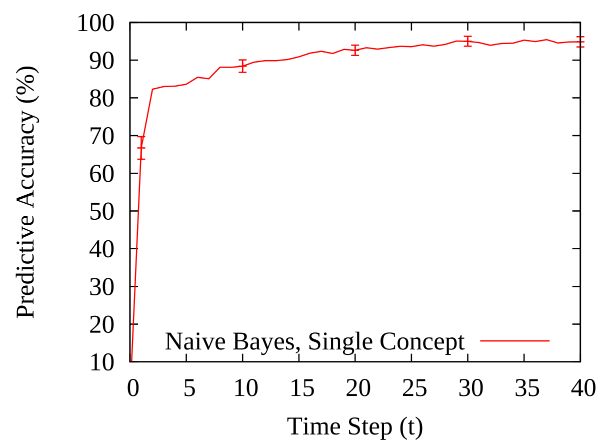
Batch versus On-line Learning

- Batch learning: When one can collect all examples for learning *before* applying the method
- Examples:
  - predict if mushrooms are poisonous (no new mushrooms)
  - predict political party based on last year's votes (all the votes have been cast)
- On-line learning: Examples arrive over time in a **stream**
- Also known as *incremental learning*
- Examples:
  - predict preferences for scheduling meetings
  - predict importance of e-mail
- What happens if the target concept changes?

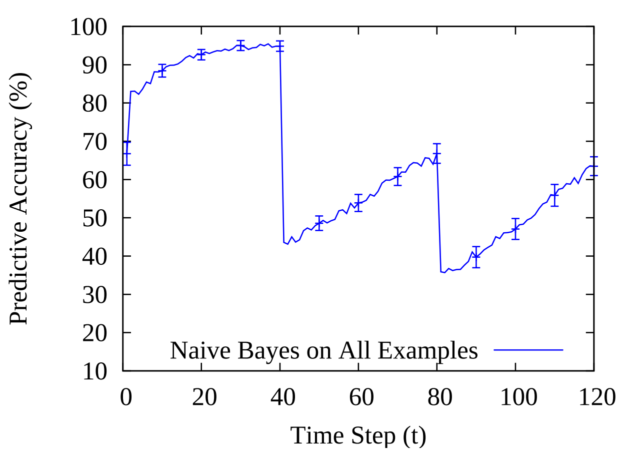
Concept Drift

- Schlimmer and Granger, 1986
- Concretely: An example has a legitimate label at one time and a different legitimate label at another time (cf. noise)
- Bayesian Decision Theory: a change in
  - the prior distribution
  - the class-conditional distribution
  - both distributions
- Geometrically: target concept in the input space changes its
  - size
  - shape
  - location
  - some combination of these
- Also known as *shifting targets*, *non-stationary environments*, *time-changing data streams*, *evolving data streams*

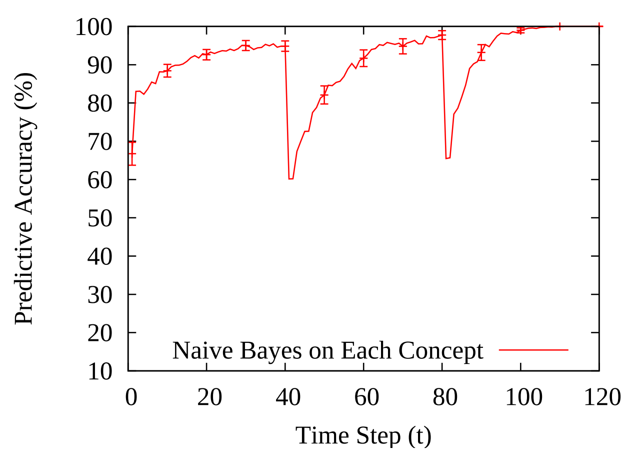
Insights into Performance
Classifier Trained on Examples from a Single Target Concept



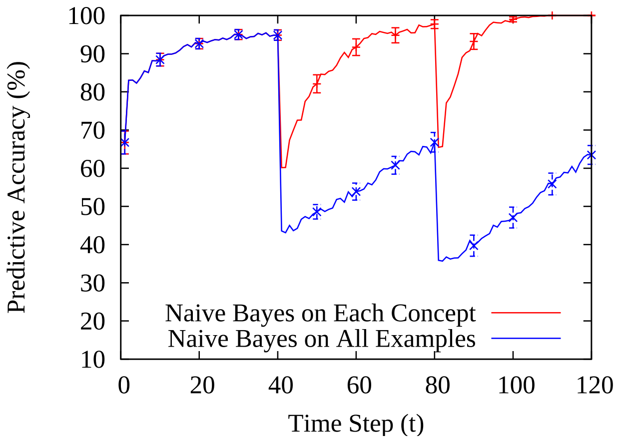
Insights into Performance
Classifier Trained on All Examples Over Three Different Target Concepts



Insights into Performance
Classifier Trained on Examples from Each Target Concept



Insights into Performance
Overlay of the Previous Two Plots



Similar Settings

- Consider the concept of "warm"
- Concept Drift
  - "warm" is different now
- Recurring Concepts
  - it's "warm" again
- Changing and Recurring Contexts
  - "warm" is different in winter than in summer
  - it's summer again
- Covariate Shift
  - "warm" is different than it was during training
- Change-point Detection
  - "warm" changed in December
- Don't be fooled by sampling or ordering effects!

Approaches to Concept Drift

- Single-model Methods
  - build and use one model for prediction
  - may maintain and develop partial models
- Meta-learning Methods
  - typically use a standard learning algorithm
  - build and use one model for prediction
  - mechanisms let the standard method cope with drift
- Ensemble Methods
  - typically use a standard learning algorithm
  - build and use multiple models for prediction
  - mechanisms let the collection of the standard methods cope with drift

Single-model Methods

- Early approaches to concept drift used single models
  - Stagger (Schlimmer and Granger, 1986)
  - Winnow (Littlestone, 1988)
  - Weighted Majority (Littlestone and Warmuth, 1994)
  - CAP (Mitchell et al., 1994)
  - FLORA (Widmer and Kubat, 1996)
  - AQ-PM Systems (Maloof and Michalski, 2000, 2004)
  - Concept-drifting Very Fast Decision Tree (Hulten et al., 2001)

CVFDT
Overview

- Concept-drifting Very Fast Decision Tree (Hulten et al., 2001)
- Basic idea: extends Hoeffding Trees (VFDT)
  - grows a tree down from leaf nodes
  - objective: leaf nodes correspond perfectly to one class label
  - splits a leaf using "entropy" and the Hoeffding bound
    - "entropy" measures the disorder of the class labels
    - Hoeffding bound indicates whether the entropy has deviated sufficiently from the expected value
  - when drift occurs, builds alternative subtrees and uses them when they are more accurate than the primary subtree
- Implemented in the Very Fast Machine Learning (VFML) Toolkit

CVFDT
Hoeffding Tree

- Concept description:
  - leaf nodes have counts for each attribute/value pair by class
  - internal nodes have a single attribute with edges to children for each attribute value
- Learning:
  - use an example's attributes and values to sort it to a leaf node and update the node's counts
  - use "entropy" to measure each attribute's ability to split the node so the children correspond more closely to the class labels
  - if the difference between the top two attributes is significant enough based on the Hoeffding bound, then split using the best attribute and create new leaf nodes
- Performance:
  - use an observation's attributes and values to sort it to a leaf node
  - return the majority class of the examples processed by that node

CVFDT
Concept Description

- A primary Hoeffding Tree
- A collection of alternative Hoeffding Trees for each node of the primary Hoeffding Tree
  - modification: internal nodes have counts for each attribute/value pair by class
- A window of the most recent examples

CVFDT I
Learning

1. Input Example
2. Remove oldest example from Window and from the primary and alternative trees
3. Add Example to Window
4. Use the Example's attributes and values to sort it from the primary Hoeffding tree's root to a leaf
5. For each node along the path, update node counts and recursively grow each node's alternative trees
6. At the leaf node:
   6.1 use "entropy" to measure each attribute's ability to split the node so the children correspond more closely to the class labels
   6.2 if the difference between the top two attributes is significant enough based on the Hoeffding bound, then split using the best attribute and create new leaf nodes

CVFDT II
Learning

7. Check the validity of the splits of internal nodes for the primary and alternative trees, and if there is a better splitting attribute for a node, then start building an alternative tree at that node with the attribute
8. Periodically, test whether an alternative subtree is more accurate on a sequence of examples, and use it instead of the primary subtree; if necessary, prune poor alternative subtrees

CVFDT
Performance

1: Input Observation
2: Use the Observation's attributes and values to sort it from the primary Hoeffding tree's root to a leaf node
3: Predict the majority class of the examples processed by that node

Meta-learning Methods

- MetaL(B) and MetaL(IB) (Widmer, 1997)
- SPLICE (Harries et al., 1998)
- $\xi\alpha$-estimators for SVMs (Klinkenberg and Joachims, 2000)
- Drift Detection Method (Gama et al., 2004)
- Early Drift Detection Method (Baena-García et al., 2006)
- ADWIN (Bifet and Gavaldà, 2007)
- EWMA for Concept Drift Detection (Ross et al., 2012)

Drift Detection Method (DDM)
Overview

- Gama, Medas, Castillo, and Rodrigues (2004)
- Basic idea:
  - monitor the accuracy of a classifier
  - if the average accuracy is within one standard deviation of the mean, then continue learning
  - if the average accuracy is within two standard deviations of the mean, then start accumulating examples
  - if the average accuracy exceeds three standard deviations of the mean, then train a new classifier with the accumulated examples
- Implemented in MOA (Massive Online Analysis)

Drift Detection Method (DDM)
Concept Description

- Some Classifier, batch or on-line
- Parameters for Binomial distribution:
  - n: number of examples processed
  - p: success probability (accuracy) over n examples
- s: standard deviation of accuracies over n examples
- p_{min}: minimum accuracy over n examples
- s_{min}: minimum standard deviation over n examples
- ps_{min}: minimum $p + s$
- Context: Examples collected during a period of instability

Drift Detection Method (DDM)
Learning

```
input Example
update p based on Classifier's prediction for Example
increment n, and compute s
if  n > 30  then
  if  p + s ≤ ps_min  then
    p_min ← p, s_min ← s, ps_min ← p + s
  end if
  if  p + s ≤ p_min + 2.0 × s_min  then                ▷ in-control
    Context.clear()
  else if  p + s ≤ p_min + 3.0 × s_min  then           ▷ warning
    Context.add(Example)
  else if  p + s > p_min + 3.0 × s_min  then           ▷ out-of-control
    initialize(n, p, s, p_min, s_min, ps_min)
    Classifier.retrain(Context)
  end if
end if
Classifier.train(Example)
```

Drift Detection Method (DDM)
Performance

```
input Observation
output Classifier.classify(Observation)
```

Ensemble Methods

- Blum's Weighted Majority and Winnow (Blum, 1997)
- Streaming Ensemble Algorithm (Street and Kim, 2001)
- Accuracy-Weighted Ensemble (Wang et al., 2003)
- Dynamic Weighted Majority (Kolter and Maloof, 2003, 2007)
- Additive Experts (Kolter and Maloof, 2005)
- Accuracy Classifier Ensemble (Nishida and Yamauchi, 2007)
- Paired Learners (Bach and Maloof, 2008)
- Adaptive-Size Hoeffding Tree (Bifet et al., 2009)
- Bayesian Conditional Model Comparison (Bach and Maloof, 2010)
- Diversity for Dealing with Drifts (Minku and Yao, 2012)

Accuracy-Weighted Ensemble (AWE)
Overview

- Wang, Fan, Yu, and Han (2003)
- Concept description:
  - Maintains a fixed-capacity, weighted ensemble of batch learners
- Learning:
  - Accumulates a preset number of examples from the data stream into a batch
  - Trains and tests a new learner using cross-validation to calculate its mean square error (MSE)
  - Tests ensemble members on the batch to calculate their MSEs
  - Sets the learners' weights to the differences between their MSEs and the expected MSE of random classification
  - Discards all learners with weight ≤ 0
  - Keeps the highest weighted learners, up to the capacity of the ensemble
- Performance:
  - Predicts using a weighted-majority vote

Accuracy-Weighted Ensemble (AWE)
Comments

- SEA (Street and Kim, 2001) is like AWE without weighting and with "quality" instead of MSE as measure for replacing members of the ensemble
- ACE (Nishida and Yamauchi, 2007) is like AWE with an on-line learner learning from each new example instead of learning from a new batch of examples
- Re-weighting the members of the ensemble:
  - mostly it helps, sometimes it doesn't
- Using on-line learners as members of the ensemble:
  - generally, it helps, sometimes significantly ($\approx$ 10–12%)
  - but increases running time

Dynamic Weighted Majority (DWM)
Overview

- Kolter and Maloof (2003, 2007)
- Concept description:
  - a resizable, weighted ensemble of on-line learners.
- Learning:
  - if the global prediction for an example is incorrect:
    - adds a new learner to the ensemble with a weight of 1
    - reduces the weights of learners that predict incorrectly by some percentage
  - trains ensemble members on the example
- Performance:
  - predicts using a weighted-majority vote

Dynamic Weighted Majority (DWM)
Concept Description

- A vector of weights for a set of on-line learners

Dynamic Weighted Majority (DWM)
Learning

input Example
$\beta \leftarrow 0.5$
prediction = classify(Example)
for $i = 1, \ldots, \vec{w}$.length **do**
  if Learners[i].classify(Example) $\neq$ Example.class() **then**
    $w_i \leftarrow \beta w_i$
  end if
  Learners[i].train(Example)
end for
$\vec{w}$.normalize()
if prediction $\neq$ Example.class() **then**
  Learners.add(new Learner())
  $\vec{w}$.add(new Double(1.0))
end if

Dynamic Weighted Majority (DWM)
Performance

input Observation
for $i = 1, \ldots, \vec{w}$.length **do**
  prediction = Experts[i].classify(Observation)
  WeightedVoter.add(prediction, w_i)
end for
output WeightedVoter.winner()

Evaluations

- Types of evaluations:
  - direct comparisons
  - indirect comparisons
  - characterizations
- Use synthetic and real-world data sets
  - test the research hypothesis
  - benchmarks
- Synthetic data sets:
  - sequence of target concepts
  - time-dependent instance generator
  - protocol for training and testing learners
  - issue: **realism**
- Real-world data sets:
  - based on or inspired by a real-world application
  - gather and process data
  - determine training and testing protocol
  - issue: **ground truth**

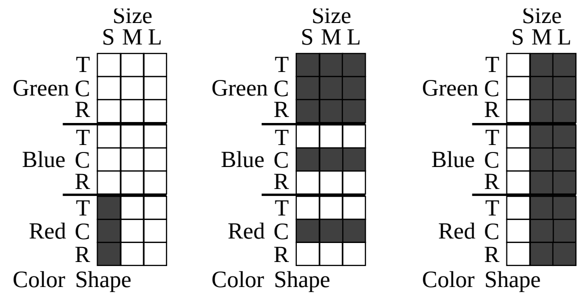
Problems and Data Sets

- Synthetic Data Sets
  - Stagger Concepts (Schlimmer and Granger, 1986; Widmer and Kubat, 1996)
  - SEA Concepts (Street and Kim, 2001)
  - CVFDT Concepts (Hulten et al., 2001)
  - AWE Concepts (Wang et al., 2003)
  - AddExp Concepts (Kolter and Maloof, 2005)
  - Also: Shifting hyperplanes, sine functions, and Gaussians
- Real-world Data Sets
  - Calendar Scheduling (Mitchell et al., 1994)
  - Electricity Pricing (Harries et al., 1998)
  - Web Page Caching (Hulten et al., 2001)
  - Credit Card Fraud (Wang et al., 2003)
  - Spam Filtering (Delany et al., 2004)

STAGGER Concepts

- Schlimmer and Granger, 1986; Widmer and Kubat, 1996
- Three discrete attributes with three possible values each
  - `size` $\in$ {`small`, `medium`, `large`}
  - `color` $\in$ {`green`, `blue`, `red`}
  - `shape` $\in$ {`triangle`, `circle`, `rectangle`}
- Class label is `true` or `false`
- For each of 120 time steps:
  - train on 1 random example
  - test on 100 random examples

Stagger Concepts

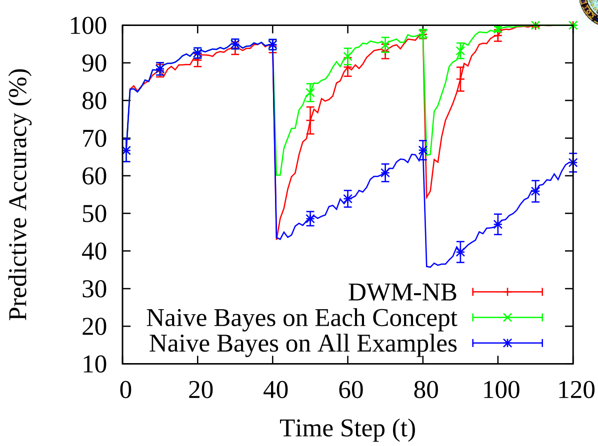


Time steps 1–40: `red` $\wedge$ `small`

Time steps 41–80: `green` $\vee$ `circle`

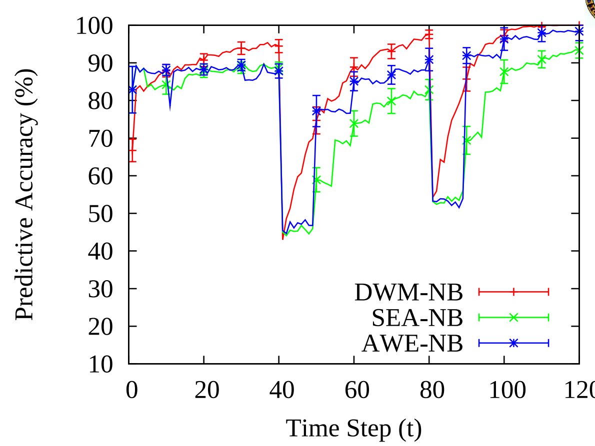
Time steps 81–120: `medium` $\vee$ `large`

Stagger Concepts



Bach and Maloof (2008)

Stagger Concepts



Bach and Maloof (2008)

Stagger Concepts

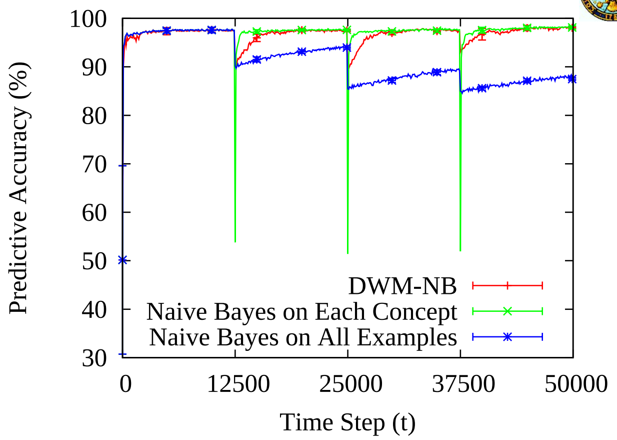
Learner	AUC
NB, on each concept	0.914±0.007
DWM-NB	0.868±0.007
AWE-NB	0.808±0.010
SEA-NB	0.732±0.011
NB, on all examples	0.516±0.011

Measures are normalized areas under the performance curve after the first drift point with 95% confidence intervals (Bach and Maloof, 2008).

SEA Concepts

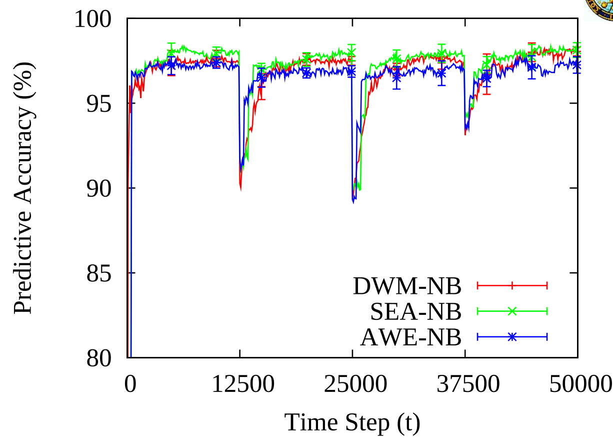
- Street and Kim, 2001
- Three numeric attributes $x_i \in [0, 10]$
- Class label is `true` or `false`
- $x_1 + x_2 \leq \theta$, where $\theta \in \{8, 9, 7, 9.5\}$
- x_3 is an irrelevant attribute
- 10% class noise
- 50,000 time steps total
- 12,500 time steps for each target concept
- 2,500 testing examples for each target concept

SEA Concepts



Bach and Maloof (2008)

SEA Concepts



Bach and Maloof (2008)

SEA Concepts

Learner	AUC
SEA-NB	0.971±0.002
NB, on each concept	0.971±0.001
DWM-NB	0.969±0.002
AWE-NB	0.966±0.002
NB, on all examples	0.890±0.002

Measures are normalized areas under the performance curve after the first drift point with 95% confidence intervals (Bach and Maloof, 2008).

Calendar Scheduling

- Mitchell, Caruana, Freitag, McDermott, and Zabowski, 1994
- Predict the location, duration, start time, and day of week for meetings
- Real-world data created by Tom Mitchell (CMU)
- 1,685 examples collected over 16 months
- 34 attributes (e.g., type of meeting, type of attendees)
- Learners process examples incrementally in temporal order

Calendar Scheduling

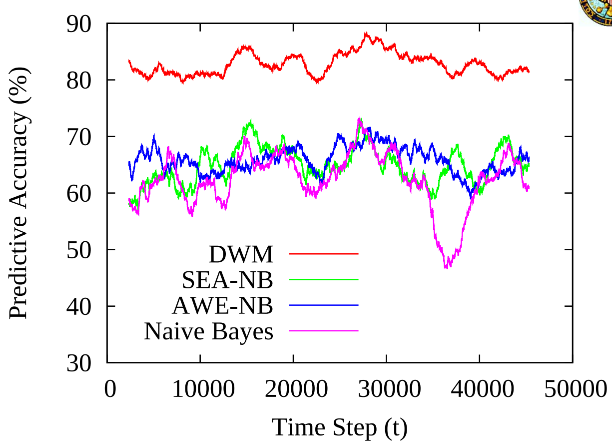
Task	BWM	DWM-NB	NB	SEA-NB	AWE-NB
Location	74.00	66.90	62.93	58.95	58.36
Duration	73.00	65.90	63.25	59.29	59.56
Start Time	50.00	38.87	33.24	27.07	27.31
Day of Week	56.00	51.70	52.29	40.95	40.58
Average	63.00	55.84	52.92	46.57	46.17

Blum (1997); Bach and Maloof (2008)

Electricity Pricing

- Harries, Sammut, and Horn, 1998
- Predict whether the price of electricity goes up or down
- Real-world data collected from New South Wales, Australia
- 45,312 examples collected at 30-minute intervals
- Five attributes
  - `Day of Week` $\in \{1 \ldots 7\}$
  - `Period of Day` $\in \{1 \ldots 48\}$
  - `Demand in New South Wales` $\in \mathbb{Z}^+$
  - `Demand in Victoria` $\in \mathbb{Z}^+$
  - `Electricity transferred` $\in \mathbb{Z}$
- Class label is `Up` or `Down`.
- Learners process examples incrementally in temporal order

Electricity Pricing



Bach and Maloof (2008)

Web Page Caching

- Hulten, Spencer, and Domingos (2001)
- Predict whether to store a requested Web page in a cache
- Real-world data set collected at the University of Washington
- 82.8 million HTTP requests collected over one week
- Peak rate of 17,400 requests per second
- 23,000 active clients
- Processing resulted in 1.89 million examples
- Results
  - VFDT: 72.7%
  - CVFDT: 72.3%

Recent and Future Directions

- Adversarial learning (Lowd and Meek, 2005)
- Statistical relational learning (Neville et al., 2005)
- Semi-supervised approaches (Jia et al., 2009; Zhang et al., 2009; Masud et al., 2011; Li et al., 2012)
- Class imbalance, skew (Gao et al., 2008; Ditzler et al., 2010)
- Big Data
  - First International Workshop on Big Data, Streams and Heterogeneous Source Mining, August 12, 2012, in Beijing
  - Special Session on Scalable Big Data Mining at Pacific Rim International Conference, September 3–7, 2012, Kuching, Malaysia
- Stronger temporal models?

A Stream of Algorithms for Concept Drift

Mark Maloof

Department of Computer Science
Georgetown University
Washington, DC 20057-1232
http://www.cs.georgetown.edu/~maloof

Heilbronn Annual Conference

21 September 2012

References I

S. H. Bach and M. A. Maloof. Paired learners for concept drift. In *Proceedings of the Eighth IEEE International Conference on Data Mining*, pages 23–32, Los Alamitos, CA, 2008. IEEE Press.

S. H. Bach and M. A. Maloof. A Bayesian approach to concept drift. In *Advances in Neural Information Processing Systems 23*, pages 127–135. Curran Associates, Red Hook, NY, 2010. URL http://books.nips.cc/nips23.html. Proceedings of the 24th Annual Conference on Neural Information Processing Systems.

M. Baena-García, J. Campo-Ávila, R. Fidalgo-Merino, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. In *Proceedings of the ECML PKDD International Workshop on Knowledge Discovery from Data Streams (IWKDDS-2006)*, 2006. URL http://www.machine-learning.eu/iwkdds-2006/.

A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 443–448, 2007. Short paper.

A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148, New York, NY, 2009. ACM Press.

A. Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. In A. Macintosh, R. Ellis, and T. Allen, editors, *Applications and Innovations in Intelligent Systems XII*, pages 3–16. Springer, Berlin-Heidelberg, 2004. URL http://www.comp.dit.ie/sjdelany/publications/AI%202004%20(crc).pdf. Proceedings of AI 2004.

G. Ditzler, R. Polikar, and N. Chawla. An incremental learning algorithm for non-stationary environments and class imbalance. In *Proceedings of the Twentieth International Conference on Pattern Recognition*, pages 2997–3000, Los Alamitos, CA, 2010. IEEE Press.

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In A. L. C. Bazzan and S. Labidi, editors, *Advances in Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer, Heidelberg-Berlin, 2004. SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29–October 1, 2004. Proceedings.

J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6):37–49, 2008.

References II

M. Harries, C. Sammut, and K. Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998. URL citeseer.nj.nec.com/harries98extracting.html.

G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, New York, NY, 2001. ACM Press.

Y. Jia, S. Yan, and C. Zhang. Semi-supervised classification on evolutionary data. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1083–1088, Menlo Park, CA, 2009. AAAI Press.

R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 487–494, San Francisco, CA, 2000. Morgan Kaufmann.

J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 123–130, Los Alamitos, CA, 2003. IEEE Press.

J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456, New York, NY, 2005. ACM Press.

J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, Dec 2007.

P. Li, X. Wu, and X. Hu. Mining recurring concept drifts with limited labeled streaming data. *ACM Transactions on Intelligent Systems and Technology*, 3(2), 2012.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, New York, NY, 2005. ACM Press. Research Track Poster Paper.

M. A. Maloof and R. S. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154: 95–126, 2004.

References III

M. A. Maloof and R. S. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.

M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874, 2011.

L. L. Minku and X. Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012. URL http://www.computer.org/csdl/trans/tk/2012/04/ttk2012040619.html.

T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, July 1994.

J. Neville, Ö. Şimşek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 449–458, New York, NY, 2005. ACM. doi: 10.1145/1081870.1081922. URL http://doi.acm.org/10.1145/1081870.1081922.

K. Nishida and K. Yamauchi. Adaptive classifiers-ensemble system for tracking concept drift. In *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, pages 3607–3612, Los Alamitos, CA, 2007. IEEE Press.

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33:191–198, 2012.

J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 502–507, Menlo Park, CA, 1986. AAAI Press.

W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, New York, NY, 2001. ACM Press.

H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235, New York, NY, 2003. ACM Press.

G. Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286, 1997.

References IV

G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23: 69–101, 1996. URL http://www.springerlink.com/content/q534525304wn2752/?p=41c43a2766df45be90992e91688ecaa7\&pi=5.

P. Zhang, X. Zhu, and L. Guo. Mining data streams with labeled and unlabeled training examples. In *Proceedings of the Ninth IEEE International Conference on Data Mining*, pages 627–636, Los Alamitos, CA, 2009. IEEE Press.