Batch-Incremental vs. Instance-Incremental Learning in Dynamic and Evolving Data

Jesse Read[1], Albert Bifet[2], Bernhard Pfahringer[2], Geoff Holmes[2]

[1]Department of Signal Theory and Communications
Universidad Carlos III
Madrid, Spain

[2]Department of Computer Science
University of Waikato
Hamilton, New Zealand

Learning in Dynamic and Evolving Data

Data instances arrive

- continually; and
- potentially infinitely.

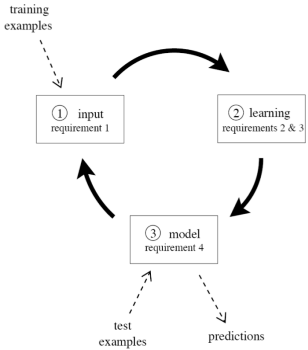
We make a classification for each instance; the true classifications can then be obtained (often via an *automatic* or *collaborative* process).

Applications

- predicting consumer demand
- categorising / filtering news
- labelling / filtering e-mail
- tagging / filtering images, videos, text documents, etc.
- robotics: predicting obstacles, faults, etc.
- social networks

Learning in Dynamic and Evolving Data

1. new training examples incoming at any point
2. must work in finite memory
3. expect *concept drift*
4. must be ready to produce a classification at any point



Instance-Incremental or Batch-Incremental Learning

Instance-Incremental: Update the model with new training examples as soon as they are available.

- Naive Bayes
- Hoeffding Decision Trees
- Neural Networks
- k-Nearest Neighbour (model based on a moving window)

Batch-Incremental: Collect w training examples, then build a batch model with these examples (and drop an old model when memory is full), and repeat.

- Logistic Regression
- Decision Trees
- Support Vector Machines
- etc.

Why This Paper?

Authors tend to take one of the approaches. . .

1. (Instance incremental) we *must* learn in a “true-incremental” fashion, using a classifier naturally suited to the job; or
2. (Batch incremental) “true-incremental” is *not* necessary, we can learn in batches using *any* batch classifier we like.

. . . and then proceed with their paper.

Which approach to use, and why?

Instance-Incremental Learning

Instance-Incremental. . .

The model is updated with new training examples as soon as they are available.

Advantages:

- "naturally suited" for incremental learning
- fast

Disadvantages:

- restricted choice of classifier
- may require massive numbers of instances to learn
- may not adapt naturally to concept drift

Batch-Incremental Learning

Batch-Incremental...

Collect w training examples, then build a batch model with these examples (and drop an old model when memory is full), and repeat.

Advantages:

- use your favourite classifier
- automatically deals with concept drift

Disadvantages:

- the most recent data is not part of model
- have to phase out models over time as memory fills up
- may be slow to learn (running time)
- have to find a good batch size (what is w?)

Experiments: Methods

Instance-Incremental Methods:

NB	Naive Bayes
SGD	Stochastic Gradient Descent
HT	Hoeffding Trees
LB-HT	Leveraging Bagging Ensemble of HT with ADWIN
kNN	*k*-Nearest Neighbour
LB-kNN	Leveraging Bagging Ensemble of kNN with ADWIN

where *Leveraging Bagging* [Bifet et al., 2010] of 10 models with the ADWIN change detector; kNN window (batch) size $-w$ 1000.

Batch-Incremental Methods:

AWE-J48	Accuracy Weighted Ensemble with C4.5 Decision Trees
AWE-SVM	Accuracy Weighted Ensemble with Support Vector Machines
AWE-LR	Accuracy Weighted Ensemble with Logistic Regression

with Accuracy Weighted Ensemble (AWE-$*$) [Wang et al., 2003] of 10 models (batches), batch size $-w$ of 500. All classifiers are from the WEKA/MOA frameworks with default parameters.

Experiments: Data

Real Datasets, varying domains, types and numbers of attributes:

- 20 NEWSGROUPS 386,000 text records, 19 shifts in concept
- IMDB 120,919 movie plot summaries, predicting the *drama* genre
- COVTYPE 581,012 instances predicting forest cover type
- POKER 1,000,000 hands, predicting the value of each hand
- ELECTRICITY 45,312 instances describing electricity demand

Synthetic Data, with varying concept drift, hundreds of thousands to millions of examples:

- SEA generated from 3 attributes, abrupt drift
- HYP Rotating Hyperplane to produce concept drift
- RBF Generator: fixed number of centroids which move
- LED Generator: predict digit on a LED display

Finding a good batch size (w)

Average Accuracy over all datasets:

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	66.32	80.24	82.33	**82.63**
AWE-J48	70.72	**77.36**	76.90	73.76
AWE-LR	68.77	**69.62**	67.83	65.56
AWE-SVM	67.13	**70.77**	70.07	67.67

Total Time (sec.) over all datasets:

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	2,180	9,993	18,349	71,540
AWE-J48	3,809	6,883	10,865	28,429
AWE-LR	9,659	66,757	10,247	10,112
AWE-SVM	13,860	5,800	6,414	39,298

Total RAM Hours over all datasets:

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
kNN	0.13	1.11	2.98	41.27
AWE-J48	1.96	8.49	21.81	221.66
AWE-LR	12.65	48.07	22.47	67.52
AWE-SVM	3.19	4.12	9.36	255.96

- kNN: more is better, but huge trade off with complexity after $-w$ 1000
- AWE-$*$: $-w$ 500 gives best results

Table: Finding the best window size for `AWE-J48`.

	$-w$ 100	$-w$ 500	$-w$ 1000	$-w$ 5000
20 Newsgroups	94.30	94.74	**95.06**	94.60
IMDB	**55.09**	53.59	53.54	54.33
CovType	55.79	**87.82**	85.58	76.05
Electricity	**78.47**	75.27	74.37	65.10
Poker	76.06	77.89	**79.32**	75.98
CovPokElec	68.03	**81.60**	81.45	74.32
LED(50000)	70.60	71.99	**72.03**	71.37
SEA(50)	84.95	88.03	88.56	**88.68**
SEA(50000)	84.63	87.71	88.16	**88.43**
HYP(10,0.0001)	66.69	71.58	73.41	**78.63**
HYP(10,0.001)	70.95	75.79	77.69	**79.94**
RBF(0,0)	69.42	83.01	84.96	**87.38**
RBF(50,0.0001)	69.12	**79.30**	77.05	60.75
RBF(10,0.0001)	68.49	81.79	**82.78**	80.79
RBF(50,0.001)	**53.78**	50.95	38.55	24.50
RBF(10,0.001)	65.18	76.76	77.92	**79.36**
Average	70.72	**77.36**	76.90	73.76

- best batch size depends on the dataset
- smaller batches much better on a moving concept, e.g. on RBF(50,0.001)

Experiments: Results

	NB	kNN	HT	AWE-J48	LB-HT	SGD	AWE-LR	AWE-SVM	LB-kNN
20 Newsgroups	68.1 **8**	94.9 **2**	94.3 **6**	94.7 **4**	94.4 **5**	94.9 **2**	88.4 **7**	95.6 **1**	DNF
IMDB	60.4 **6**	60.8 **5**	63.5 **2**	53.6 **9**	61.8 **4**	63.8 **1**	54.0 **8**	54.5 **7**	62.4 **3**
CovType	60.5 **9**	92.2 **2**	80.3 **7**	87.8 **4**	88.6 **3**	60.7 **8**	84.5 **5**	84.2 **6**	92.4 **1**
Electricity	73.4 **6**	78.4 **4**	79.2 **3**	75.3 **5**	88.8 **1**	57.6 **9**	70.5 **7**	68.6 **8**	80.8 **2**
Poker	59.5 **9**	69.3 **5**	76.1 **3**	77.9 **2**	95.0 **1**	68.9 **6**	60.9 **7**	60.4 **8**	70.3 **4**
CovPokElec	24.2 **9**	78.4 **5**	79.3 **3**	81.6 **2**	92.4 **1**	68.1 **8**	70.1 **6**	69.8 **7**	79.1 **4**
LED(50000)	54.0 **8**	63.2 **7**	68.7 **6**	72.0 **4**	73.2 **1**	11.8 **9**	73.0 **2**	72.8 **3**	69.8 **5**
SEA(50)	85.4 **9**	86.8 **6**	86.4 **7**	88.0 **4**	88.2 **3**	85.4 **8**	89.4 **2**	89.6 **1**	88.0 **5**
SEA(50000)	85.4 **8**	86.5 **6**	86.4 **7**	87.7 **5**	88.8 **3**	85.2 **9**	89.0 **2**	89.2 **1**	87.7 **4**
HYP(10,0.0001)	91.2 **3**	83.3 **7**	89.0 **4**	71.6 **9**	88.1 **5**	79.5 **8**	93.7 **1**	93.4 **2**	87.1 **6**
HYP(10,0.001)	70.9 **9**	83.3 **5**	78.8 **6**	75.8 **7**	84.8 **4**	71.1 **8**	91.8 **2**	92.0 **1**	86.9 **3**
RBF(0,0)	51.2 **6**	89.0 **3**	83.2 **4**	83.0 **5**	89.7 **2**	16.6 **9**	46.9 **8**	50.5 **7**	90.6 **1**
RBF(50,0.0001)	31.0 **8**	89.4 **2**	45.5 **7**	79.3 **3**	76.7 **4**	16.6 **9**	54.9 **6**	57.9 **5**	90.5 **1**
RBF(10,0.0001)	52.1 **7**	89.3 **2**	79.2 **5**	81.8 **4**	85.5 **3**	16.6 **9**	51.0 **8**	52.8 **6**	90.7 **1**
RBF(50,0.001)	29.1 **8**	84.0 **1**	32.3 **7**	51.0 **4**	55.7 **3**	16.6 **9**	46.5 **6**	50.4 **5**	82.1 **2**
RBF(10,0.001)	52.0 **6**	88.3 **2**	76.4 **5**	76.8 **4**	81.8 **3**	16.6 **9**	49.4 **8**	50.7 **7**	88.9 **1**
Avg. Rank	7.44 **8**	4.00 **3**	5.12 **6**	4.69 **4**	2.88 **2**	7.56 **9**	5.31 **7**	4.69 **4**	2.69 **1**
Avg. Accuracy	59.3 **7**	82.3 **2**	74.9 **4**	77.4 **3**	83.3 **1**	51.9 **8**	69.6 **6**	70.8 **5**	
Tot. Time (s)	260	18349	417	6883	9877	42	66757	5800	166312
Tot. RAM-Hrs	0.01	0.80	0.54	3.55	50.39	0.00	37.83	3.49	77.90

(Format: Accuracy **Rank**); RAM-Hrs = hours with 1 GB in memory

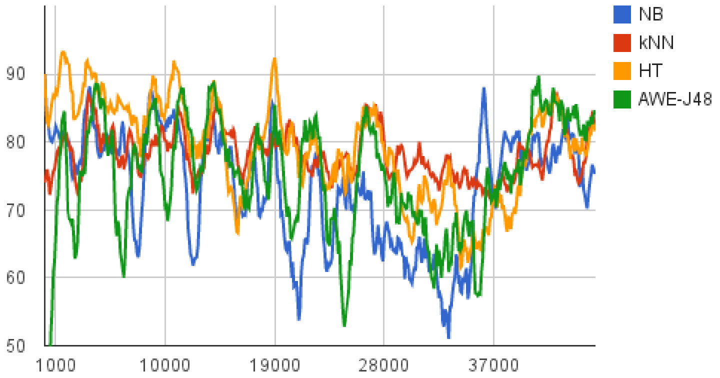
Summary of Results

- Naive Bayes, SGD, HT are fast
- Batch- SVM, J48, LR perform better, slower
- kNN is the best single model
- Hoeffding Trees (HT) accurate on stable concepts; but
- batch Decision Trees (AWE-J48) are better on dynamic contexts
- Leveraging Bagging + ADWIN recovers HT losses, but at a large computational cost
- Leveraging Bagging + ADWIN with kNN is the best but slowest method
- Each method except Naive Bayes is in top-2 at least once

	NB	kNN	HT	AWE-J48	LB-HT	SGD	AWE-LR	AWE-SVM	LB-kNN
Avg. Rank	7.44 **8**	4.00 **3**	5.12 **6**	4.69 **4**	2.88 **2**	7.56 **9**	5.31 **7**	4.69 **4**	2.69 **1**
Avg. Accuracy	59.3 **7**	82.3 **2**	74.9 **4**	77.4 **3**	83.3 **1**	51.9 **8**	69.6 **6**	70.8 **5**	
Tot. Time (s)	260	18349	417	6883	9877	42	66757	5800	166312
Tot. RAM-Hrs	0.01	0.80	0.54	3.55	50.39	0.00	37.83	3.49	77.90

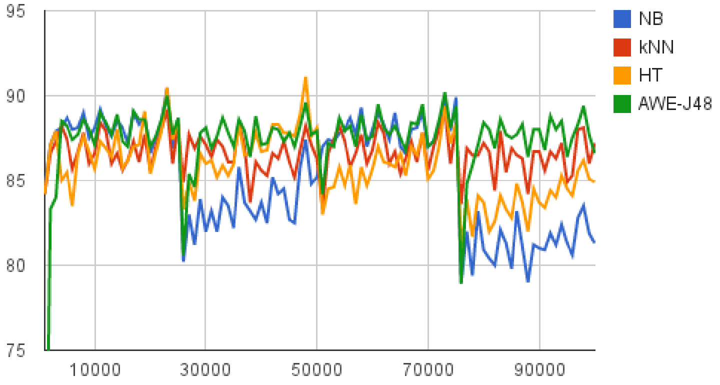
(Format: Accuracy **Rank**); RAM-Hrs = hours with 1 GB in memory

Accuracy and Running Time over Time



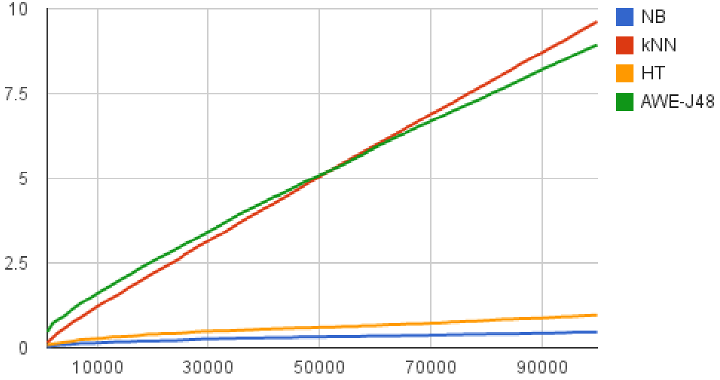
Classification accuracy over time on the Electricity dataset.

Accuracy and Running Time over Time



Classification accuracy over time on the SEA dataset.

Accuracy and Running Time over Time



Cumulative running time over time on the SEA dataset.

Conclusions: Instance-incremental vs. Batch-incremental

- Not sure? kNN is a safe bet!
- A good model of 1000 instances can be better than one of millions.
- If you go instance-incremental, find the concept drift!
- If you can spare the resources, go ensemble.
- And, as always – choose your method according to your data.

Conclusions: Instance-incremental vs. Batch-incremental

- Not sure? kNN is a safe bet!
- A good model of 1000 instances can be better than one of millions.
- If you go instance-incremental, find the concept drift!
- If you can spare the resources, go ensemble.
- And, as always – choose your method according to your data.

The End

Questions ??

Bifet, A., Holmes, G., and Pfahringer, B. (2010).
Leveraging bagging for evolving data streams.
In *ECML/PKDD (1)*, pages 135–150.

Wang, H., Fan, W., Yu, P. S., and Han, J. (2003).
Mining concept-drifting data streams using ensemble classifiers.
In *KDD '03*, pages 226–235, New York, NY, USA. ACM.