# An Empirical Comparison of Bayesian Network Parameter Learning Algorithms for Continuous Data Streams

**Parot Ratnapinda**[1] **& Marek J. Druzdzel**[1,2]

[1] Decision Systems Laboratory, School of Information Sciences and Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA

[2] Faculty of Computer Science, Białystok University of Technology, Wiejska 45A, 15-351 Białystok, Poland

## Abstract

We compare three approaches to learning numerical parameters of Bayesian networks from continuous data streams: (1) the EM algorithm applied to all data, (2) the EM algorithm applied to data increments, and (3) the online EM algorithm. Our results show that learning from all data at each step, whenever feasible, leads to the highest parameter accuracy and model classification accuracy. When facing computational limitations, incremental learning approaches are a reasonable alternative. Of these, online EM is reasonably fast, and similar to the incremental EM algorithm in terms of accuracy. For small data sets, incremental EM seems to lead to better accuracy. When the data size gets large, online EM tends to be more accurate.

## Introduction

An increasing number of domains involve continuous collection of massive amounts of data. World Wide Web-based systems, for example, often generate records for every user transaction. Real-time monitoring systems obtain sensor readings in fraction of a second increment. A corporate call center may deal with hundreds or even thousands of new cases daily. There exist systems that specialize in continuous data streams and that operate in real-time, for example those mentioned in (Tucker et al. 2003; Olesen, Lauritzen, and Jensen 1992). They all need to learn from the incoming massive amounts of data and systematically update whatever they know about the system that they are monitoring.

There are two fundamental approaches to processing continuous data streams, which we will call *batch learning* and *incremental learning*. In the batch learning approach, we repeatedly add new records to the accumulated data and learn anew from the entire data set. When the number of data records becomes very large, this approach may be computationally prohibitive. In addition, it requires storing and efficiently retrieving the entire data set, which may not be feasible. In the incremental learning approach, we assume that the model learned in the previous step summarizes all the data collected up to that step and use the newly acquired data to refine the model. Incremental learning approach can be divided into two types: *incremental batch learning* and *online learning*. The incremental batch learning or mini-batch learning updates the model by processing the incoming data in chunks, i.e., group of records. The online learning updates the model by processing records one at the time as they arrive.

Our work is in the context of Bayesian network models (Pearl 1988), which have become increasingly popular in modeling and learning tasks. The most flexible algorithm for learning Bayesian network parameters is the EM (Expectation Maximization) algorithm (Dempster, Laird, and Rubin 1977; Lauritzen 1995). While there are several variants of the EM algorithm, two are most notable: the basic EM algorithm (Dempster, Laird, and Rubin 1977) and the *online EM* algorithm (Sato and Ishii 2000; Liang and Klein 2009; Cappe 2010).

The most common mode of operation of the basic EM algorithm is *batch learning*, i.e., learning from an entire data set. The basic EM algorithm can be also applied to incremental batch learning, in which case the existing set of parameters, learned previously from a database of cases, is assigned a level of reliability, expressed by a number called the *equivalent sample size*. Equivalent sample size expresses the number of data records that have been used to learn the existing parameters. While updating the existing parameters, the EM algorithm weights the new cases against the existing parameters according to the relative sizes of the data sets. The computational complexity of the incremental batch learning depends primarily on the size of the set of additional records, i.e., the mini-batch. The *on-line EM* algorithm is a modification of the basic EM algorithm that allows for processing new data into the existing model one record at a time. Its complexity at each time step, both in terms of computation time and memory use, is thus minimal. We should state clearly here that it is based on different principles than incremental EM, so the two algorithms are not equivalent when the increment is equal to one record.

The question that we pose in this paper is which of the three approaches is best in practice when learning Bayesian network parameters from continuous data streams. We focus on the impact of choice of each of the learning schemes on (1) computational complexity of learning (speed), (2) accuracy of the learned parameters, and (3) the model's ultimate accuracy. We pose the third question in the context of classification tasks, which is a common application of Bayesian networks. While there exists literature that is related to this question, no comprehensive comparison has been made so far in the context of Bayesian networks. Some papers focus on the comparison of batch learning to incremental learning,e.g., (Carbonara and Borrowman 1998; Wilson and Martinez 2003). They all agree on the obvious truth that the online learning is computationally more efficient than batch learning but they show experimentally that it also achieves accuracy that is similar to that of the batch learning. Cappe (2010), who compares batch EM to online EM, suggests that the decision to select between the two algorithms depends on the size of the data set. His experiments indicate that when the size of the data is less than 1,000 records, EM is preferred over online EM. Holmes and Kirkby (2004) study how mini-batch size affects the performance of incremental learning in terms of classification accuracy and speed. They demonstrate that larger chunk sizes lead to higher classification accuracy.

In this paper, we describe an experiment, in which we use several real data sets from the UCI Machine Learning Repository (Frank and Asuncion 2010) to create gold standard Bayesian network models. We subsequently use these models to generate continuous streams of data. We learn the parameters from these streams of data with three approaches: *batch learning*, *incremental batch learning*, and *online learning*. We measure the time taken by the learning procedure, compare the accuracy of the learned parameters to the original (gold standard) parameters that have generated the data, and test the diagnostic accuracy of the learned models.

Our results show that the batch learning approach leads consistently to the best parameter accuracy and classification accuracy but may take orders of magnitude longer run times than incremental learning. The incremental batch learning approach uses the least computation time but its performance is typically inferior to both batch learning and online learning. The online learning performs typically worse than batch learning but requires only a modest computational and storage effort.

## Bayesian networks

Bayesian networks (Pearl 1988) are probabilistic models that represent joint probability distributions over finite sets of random variables. The structure of the graph of a Bayesian network represents direct probabilistic dependences (or, strictly speaking, independences) among variables. The interaction among every variable $X_i$ and its direct predecessors $Pa(X_i)$ is characterized numerically by a conditional probability table (CPT) representing the conditional probability distributions over the states of $X_i$ given all possible combinations of states of $Pa(X_i)$. Variables without predecessors are specified by prior probability distributions over their states.

The joint probability distribution over the set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ represented by a Bayesian network can be obtained by taking the product of all prior and conditional probability distributions:

$$\Pr(\mathbf{X}) = \Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i | Pa(X_i)) \ .$$

The most important computation performed in Bayesian networks is known as belief updating and amounts to computing the probability distribution over variables of interest given observations of other variables (the evidence). For example, we can use a medical diagnostics model to compute the posterior probability distribution over the modeled diseases given observations of same symptoms.

Bayesian networks have been widely used in classification tasks (Friedman, Geiger, and Goldszmidt 1997). Classification is the task of predicting the class to which an instance belongs based on values described by the attributes of that instance.

## The EM algorithm

The Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) is a widely used method of computing maximum likelihood estimates given incomplete data. One application of the EM algorithm is in learning parameters of Bayesian networks. The EM algorithm consists of two steps: (1) the expectation step (E-step) that uses the current parameters to compute the expected values of the missing data, and (2) the maximization step (M-step), during which the maximum likelihood of the parameters are estimates based on the expected values from the E-step. The EM process repeats until it converges to the maximum likelihood or it reaches a pre-defined improvement threshold.

In the *basic EM algorithm*, during each iteration, we perform the E-step to calculate the expected sufficient statistics across all observation. Then, we do the M-step once at the end to re-estimate the parameters using sufficient statistics from the E-step. Following (Cappe 2010), we describe the basic EM algorithm as follows: Given $n$ observations, $Y_1, \ldots, Y_n$, and an initial parameter guess $\theta_0$, do, for $k \geq 1$.

$$\begin{aligned} \textbf{E-step:} \quad S_{n,k} &= \frac{1}{n}\sum_{t=1}^{n} E_{\theta_{k-1}}\left[ s(X_t, Y_t) \middle| Y_t \right] \\ \textbf{M-step:} \quad \theta_k &= \bar{\theta}\left(S_{n,k}\right) \ . \end{aligned}$$

We define $X_t$ as a random variable corresponding to a variable $Y_t$.

The *online EM algorithm* performs the E-step and follows it by the M-step after each observation. In the E-step, the online EM algorithm uses stochastic approximation instead of sufficient statistics (Cappe 2010).

$$S_n = S_{n-1} + \gamma_n \left( E_{\bar{\theta}(S_{n-1})}\left[ s(X_n, Y_n) \middle| Y_n \right] - S_{n-1} \right) \ .$$

Updating the model after each observation may lead to a poor approximation, which the online EM algorithm avoids by interpolating between $S_{n-1}$ and the expectation values. The value that weights between a previous value and an expected value is a positive step size called $\gamma_n$. We use the generalized version of the online EM algorithm, proposed by Cappe (2010), in the following way. Given $S_0, \theta_0$ and a sequence of step sizes $(\gamma_n)_{n \geq 1}$, do, for $n \geq 1$.

$$\begin{aligned} \textbf{E-step:} \quad S_n = &\quad (1-\gamma_n)S_{n-1} + \gamma_n E_{\theta_{n-1}}\left[s(X_n, Y_n) | Y_n\right] \\ \textbf{M-step:} \quad \theta_n = &\quad \bar{\theta}(S_n)\ . \end{aligned}$$

We have to select a step size $\gamma_n$ for the online EM. To guarantee convergence, the $\gamma_n$ needs to decrease to zero. We use the assumption often used in the stochastic approximation literature that $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$. If we use $\gamma_n = 1/n^\alpha$, then the range of values for $0.5 < \alpha \leq 1$ is valid (smaller $\alpha$ means larger the update). Cappe suggests that the most robust value of $\alpha$ is 0.6 and in our work we have followed this suggestion.

For continuous data streams, in which the data become very large over time, the EM can take a very long time to converge. In Liang and Klein (2009) experiments, one data set needs 100 iterations of the EM algorithm in order to reach a reasonable accuracy. This problem is due to the nature of the EM algorithm — it has to process all data before updating parameters in the M-step. When the data set is large, computing sufficient statistics for all data for the purpose of making just one update may be wasteful.

## Empirical Evaluation

In our experiments, we selected seven data sets from the UCI Machine Learning Repository in order to create gold standard Bayesian network models. We subsequently used these models to generate large data sets (each containing 1,000,000 records) to simulate continuous data streams in our experiments. We re-learned Bayesian network parameters from these data streams using (1) batch learning, (2) incremental batch learning, and (3) online learning.

We implemented the EM and the online EM algorithms in C++. We performed our tests on a Windows 7 computer with 8 GB of memory, and an Intel Core i5-3317U processor running at 1.70 GHz.

### The Data

We selected seven data sets from the UCI Machine Learning Repository (Frank and Asuncion 2010): Adult, Australian Credit, Bank Marketing, Chess (King-Rook vs. King-Pawn), Letter, Mushroom and Nursery using the following selecting criteria:

- The data include a known class variable so that we could test the accuracy of the learned models on a real problem.
- The data set contains a reasonably large number of records. We used the EM algorithm for learning parameters in the gold standard models. The EM algorithm learns parameters more accurately from large data sets and this increased the quality of our initial models. In addition, because we check the accuracy of the models on the original data, the larger the data set, the more reliable our results.
- The majority of the attribute types should be discrete in order to reduce the need for discretization, which would be a confounding factor in our experiments.
- The data set does not contain too many missing values (no more than 1/3 of the data set). Missing values require special treatment in structure learning algorithms, which would be an additional confounding factor in our experiments.
- The selected data sets have a wide range in the number of attributes (8–36), so that we obtain models of different size for testing.

We decided to use real data sets rather than synthetic data sets because we wanted our experiments to be as close as possible to real world applications. We listed all selected data sets in Table 1.

Table 1: Data sets used in our experiments. #I denotes the number of records, #A denotes the number of attributes, #CV denotes the number of class variables, #FP denotes the number of free parameters, and MV indicates presence of missing values.

| Dataset | #I | #A | #CV | #FP | MV |
|---|---|---|---|---|---|
| Adult | 48842 | 14 | 2 | 3762 | Yes |
| Australian Credit | 690 | 14 | 2 | 388 | No |
| Bank Marketing | 45211 | 16 | 2 | 1180 | No |
| Chess | 3196 | 36 | 2 | 972 | No |
| Letter | 20000 | 16 | 26 | 25279 | No |
| Mushroom | 8124 | 22 | 2 | 4786 | Yes |
| Nursery | 12960 | 8 | 5 | 645 | No |

### Experiments

To learn the gold standard Bayesian networks, we applied the standard Bayesian learning algorithm proposed by Cooper and Herskovits (1992). The Bayesian learning algorithm does not handle missing values and continuous variables. We first discretized continuous attributes using equal frequency discretization with 5 intervals, removed all records with missing values, and used the Bayesian learning algorithm to learn the model structure. Finally, we used the entire data sets (i.e., including the records with missing values) to learn the models' numerical parameters. The models constructed in this way were our gold standard models.

We used the gold standard models to generate data sets of 1,000,000 records each (no missing values). We used these records to simulate data streams in our experiments. We used the structures of the gold standard models as skeletal models for learning parameters.

In our experiments, we compared the following three algorithms for parameter learning from continuous data streams:

1. The basic EM algorithm applied at each step to the entire data set. We referred to it as the *batch learning approach*. We started running the batch learning procedure at 10,000 records. Then we invoked the batch learning algorithm after every 10,000 records. We used uniform priors and equivalent sample size of 1 for all runs.
2. *The batch incremental learning approach* means that the learning happens after each $k$ new instances and these new

records serve to refine the existing model. In our experiments, we set $k = 10,000$. In the first step (the first 10,000 records), we used uniform priors and equivalent sample size of 1. In each subsequent step, we used the existing model as the base model and run the EM algorithm with the equivalent sample size parameter equal to the number of data records that had been used to learn the existing model. For example, when processing the records between 30,000 and 40,000, we set the equivalent sample size to 30,000 (the existing model had been learned from the previous 30,000 records). The basic EM thus combines the new parameters with the existing parameters, according to the relative size of the data sets.

3. *The online learning approach* updates the network parameters each time a new record becomes available. We ran the online EM algorithm until it reached 1,000,000 records. According to Cappe (2010), to get better performance in parameter learning, it is better not to perform the maximization step for the first 20 records. Following his idea, we started the maximization step only after the first 20 records had been processed. We used the learning rate $\alpha = 0.6$ for all runs.

We measured the CPU time consumed by each of the algorithms. We measured the accuracy of parameters in the learned model by comparing them to the parameters in the gold standard models. We tested the classification accuracy of the learned Bayesian network models on the original data sets from the UCI Machine Learning Repository.

## Results

Due to space constraints, we present all results in tabular format along with a handful of representative graphs.

**Speed** Figure 1 show the difference between incremental learning and batch learning grows larger as the number of records increases. The larger the number of attributes in a Bayesian networks, the larger the saving in computation time. We report the run time of each algorithm processing the last 10,000 records (i.e., from 990,000 to 1,000,000) in Table 2. The batch incremental learning approach and the online learning approach use constant amount of time for each run and spend less computation time than the batch learning approach. Times on the order of a second are practically negligible in a system employed in practice — data usually come at a lower speed.

**Parameter accuracy** We measure the accuracy of each algorithm by calculating the Hellinger distance (Kokolakis and Nanopoulos 2001) between the parameters in the learned models and the original parameters in the gold standard models. The Hellinger distance between two probability distributions $P$ and $G$ is computed using the following equation:

$$D_H(P,G) = \sqrt{\sum_i (\sqrt{p_i} - \sqrt{g_i})^2} \; .$$

It is similar to the Kullback-Leibler divergence (Kullback and Leibler 1951), widely used in the Bayesian network

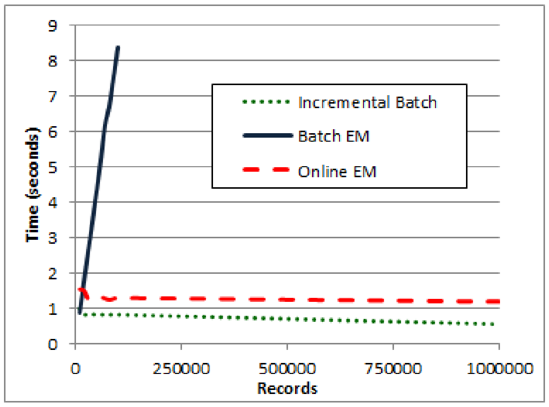

Figure 1: Adult data set computation time. Time taken by the batch algorithm is linear in the number of records, we omit its run time after 100,000 records in order to show the details for the incremental batch EM and the online EM algorithms

Table 2: Computation time required to process the 1,000,000th record shown in seconds. Please note that the incremental batch learning and the online learning algorithms process the last 10,000 records (i.e., from 990,000 to 1,000,000).

| Data set | Incremental Batch | Batch | Online |
|---|---|---|---|
| Adult | 0.55 | 84.22 | 1.18 |
| Australian Credit | 0.55 | 78.66 | 0.66 |
| Bank Marketing | 0.73 | 112.41 | 0.94 |
| Chess | 1.93 | 276.55 | 1.89 |
| Letter | 1.03 | 157.72 | 4.99 |
| Mushroom | 1.61 | 165.47 | 1.86 |
| Nursery | 0.36 | 51.72 | 0.48 |

community, in the sense of amplifying large absolute differences in small probabilities, under-appreciated by Euclidean distance. At the same time, it is free of a disturbing property of the latter of being undefined for zero probabilities.

We show the final average Hellinger distance for all data sets in Table 3. Because the shape of the distance curves as a function of the number of records seems quite regular, we also added a second number that indicates the slope of the curve at the last 100,000 records. When equal to 0.01, for example, it leads to an absolute reduction of Hellinger distance of 0.01 per 100,000 records.

In six of the seven cases, the batch learning approach resulted in the smallest Hellinger distance, i.e., the highest accuracy of retrieving the original parameters from data. The online learning performed best only on the Mushroom data set. This result differs somewhat from the results obtained by Liang and Klein (2009), who observed that online EM is often more accurate than batch EM on unsupervised tasks. The slopes of the curves (the second number in the table) indicate that each of the algorithms leads to an improvement in accuracy over time. However, this improvement is typically smaller for the incremental batch algorithm.

We show a typical plot of Hellinger distance as a func-

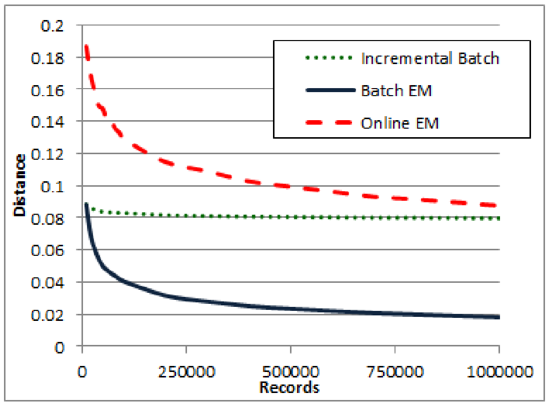

Figure 2: Average Hellinger distance as a function of the number of records in the data stream for the Adult data set

Table 3: Final Hellinger distance

| Data set | Incremental Batch | Batch | Online |
|---|---|---|---|
| Adult | 0.07971 | **0.01806** | 0.08759 |
| | -0.00014 | -0.00091 | -0.00202 |
| Australian Credit | 0.02750 | **0.00527** | 0.01805 |
| | -0.00012 | -0.00018 | -0.00003 |
| Bank Marketing | 0.05650 | **0.00690** | 0.04494 |
| | -0.00011 | -0.00043 | -0.00188 |
| Chess | 0.03140 | **0.00777** | 0.03744 |
| | -0.00003 | +0.00010 | -0.00077 |
| Letter | 0.13554 | **0.06567** | 0.17683 |
| | -0.00006 | -0.00133 | -0.00329 |
| Mushroom | 0.09207 | 0.04195 | **0.03096** |
| | -0.00005 | -0.00036 | -0.00100 |
| Nursery | 0.02957 | **0.01371** | 0.04725 |
| | -0.00005 | -0.00128 | -0.00244 |

tion of the number of records for the Adult data set (Figure 2). Hellinger distance for both batch EM and online EM decreases with the number of records. Interestingly, incremental batch learning typically seems to reach a plateau beyond which it hardly improves the accuracy of parameters.

**Classification accuracy** In testing the classification accuracy of the learned models on the original UCI Machine Learning Repository data sets, we used the simplest possible criterion, which is that the model guesses the most likely class to be the correct class for each record. Table 4 shows the final accuracy for all data sets. While the difference in accuracy is minimal, the batch learning approach resulted in the best classification accuracy on all data except for the Nursery data set. We show two typical plots of model's classification accuracy as a function of the number of records in Figures 3 and 4.

## Discussion

Our paper addresses the problem of learning Bayesian network parameters from continuous data streams. We have described an experiment that focuses on a comparison of three approaches: (1) batch learning, (2) incremental batch learning, and (3) online learning, in terms of the computational

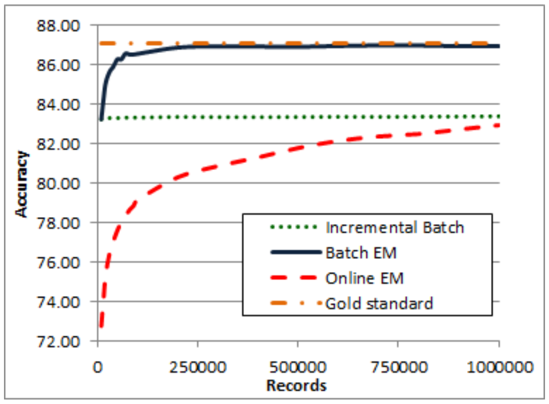

Figure 3: Classification accuracy as a function of the number of records for the Letter data set

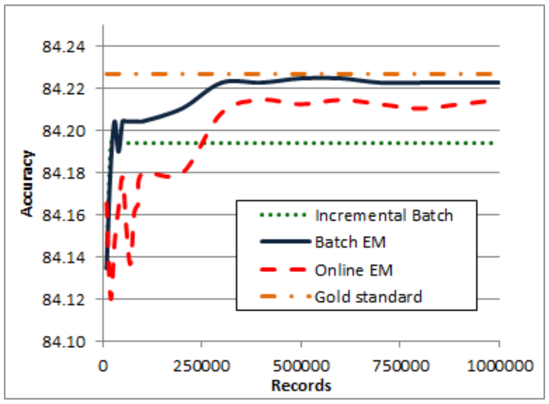

Figure 4: Classification accuracy as a function of the number of records for the Adult data set

efficiency, the resulting accuracy of parameters, and the resulting classification accuracy.

Our results indicate that while batch learning, i.e., the basic EM algorithm, makes the largest computational demands and requires access to the entire data set, i.e., makes high demands on storage, it also offers the highest resulting parameter accuracy. Online EM requires less computation time and no storage while achieving similar results to incremental EM algorithm in terms of accuracy. Incremental EM has better accuracy when the data sets is small.

We advise to use batch learning applied to the entire data set whenever computation time and memory space permit. When the computation becomes too long or the complete data set uses too much storage, we recommend switching over to the online algorithm as a safer choice. It seems that both the batch and the online EM algorithms make significant improvements in accuracy in the beginning. A possible hybrid strategy is to start with the batch EM algorithm and transform it to the online EM algorithm when necessary. An alternative strategy for all systems in which real-time response is critical, is to use the online EM algorithm during daily operations and the batch EM during maintenance hours. This should ensure that the starting points of

Table 4: Final classification accuracy

| Data set | Inc. Batch | Batch | Online | Gold |
|---|---|---|---|---|
| Adult | 84.19% | **84.22%** | 84.21% | 84.23% |
| Australian Credit | **85.51%** | **85.51%** | **85.51%** | 85.51% |
| Bank Marketing | 89.22% | **89.55%** | 89.32% | 89.55% |
| Chess | **94.21%** | **94.21%** | **94.21%** | 94.21% |
| Letter | 83.40% | **86.95%** | 82.95% | 87.09% |
| Mushroom | 99.85% | **99.90%** | 99.85% | 99.90% |
| Nursery | **94.61%** | 94.54% | 94.54% | 94.65% |

the online algorithm for real-time operations is always the best possible.

We observed that classification accuracy did not change much with refinement of parameters. Our experiments have confirmed an earlier finding of Onisko and Druzdzel (2013) that Bayesian networks are quite insensitive to precision of their numerical parameters.

There is a fundamental question that one needs to ask when processing continuous streams of data: Does the system generating the data remain constant over time? In our work, we assumed tentatively that it does and, hence, we learned from all accumulated records. Real systems, however, can evolve over time (e.g., (Łupińska-Dubicka and Druzdzel 2012)). In all such cases, we need to assign more recent parameters a higher weight. It may be natural in such cases to discard older records altogether and, hence, avoid the problem of excessive data storage or prohibitive computation. Our approach and results hold for all such cases. One might treat the data sets in our experiment as belonging to the sliding windows from which the parameters are learned.

## Acknowledgments

We acknowledge the support from XDATA program of Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0332, the National Institute of Health under grant number U01HL101066-01, and the Government of the Kingdom of Thailand. Implementation of this work is based on SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`. We would like to express our thanks to the DSL members for their help. Special thanks go to Mark Voortman for his help in implementing the online EM algorithm. We thank anonymous reviewers for valuable suggestions that improved the clarity of the paper.

## References

Cappe, O. 2010. Online Expectation-Maximisation. *ArXiv e-prints:1011.1745* 1–20.

Carbonara, L., and Borrowman, A. 1998. A comparison of batch and incremental supervised learning algorithms. In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, PKDD '98, 264–272. London, UK, UK: Springer-Verlag.

Cooper, G. F., and Herskovits, E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9(4):309–347.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1):1–38.

Frank, A., and Asuncion, A. 2010. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Friedman, N.; Geiger, D.; and Goldszmidt, M. 1997. Bayesian network classifiers. *Machine Learning* 29(2-3):131–163.

Holmes G, Kirkby R, B. D. 2004. Batch incremental learning for mining data streams. Working paper, Department of Computer Science, University of Waikato. http://researchcommons.waikato.ac.nz/handle/10289/1749.

Kokolakis, G., and Nanopoulos, P. 2001. Bayesian multivariate micro-aggregation under the Hellinger's distance criterion. *Research in Official Statistics* 4(1):117–126.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* 22:79–86.

Lauritzen, S. L. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis* 19(2):191–201.

Liang, P., and Klein, D. 2009. Online EM for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, 611–619. Stroudsburg, PA, USA: Association for Computational Linguistics.

Łupińska-Dubicka, A., and Druzdzel, M. J. 2012. A comparison of popular fertility awareness methods to a DBN model of the woman's monthly cycle. In *The Sixth European Workshop on Probabilistic Graphical Models (PGM 2012)*, 219–226.

Olesen, K.; Lauritzen, S.; and Jensen, F. 1992. aHUGIN: a system creating adaptive causal probabilistic networks. In *Proceedings of the Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)*, 223–229. San Mateo, CA: Morgan Kaufmann.

Oniśko, A., and Druzdzel, M. J. 2013. Impact of precision of Bayesian networks parameters on accuracy of medical diagnostic systems. *Artificial Intelligence in Medicine* To appear.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Sato, M.-A., and Ishii, S. 2000. On-line EM algorithm for the normalized gaussian network. *Neural Computation* 12(2):407–432.

Tucker, P. A.; Maier, D.; Sheard, T.; and Fegaras, L. 2003. Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering* 15(3):555–568.

Wilson, D. R., and Martinez, T. R. 2003. The general inefficiency of batch training for gradient descent learning. *Neural Networks* 16(10):1429–1451.