Journal of Applied Logic 13 (2015) 628–642

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Learning discrete Bayesian network parameters from continuous data streams: What is the best strategy?

Parot Ratnapinda [a,b,*], Marek J. Druzdzel [a,c,**]

[a] *Decision Systems Laboratory, School of Information Sciences and Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA*
[b] *Faculty of Science, Information Technology Division, Maejo University, Chiang Mai 50290, Thailand*
[c] *Faculty of Computer Science, Białystok University of Technology, Wiejska 45A, 15-351 Białystok, Poland*

ARTICLE INFO

Article history:
Available online 18 March 2015

Keywords:
Bayesian networks
Parameter learning
EM algorithm

ABSTRACT

We compare three approaches to learning numerical parameters of discrete 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. While the differences in speed between incremental algorithms are not large (online EM is slightly slower), for all but small data sets online EM tends to be more accurate than incremental EM.

1. Introduction

An increasing number of domains produce continuous, 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 increments. A corporate call center may deal with hundreds or even thousands of new cases daily. There exist computer programs that specialize in continuous data streams and that operate in real-time, e.g., [1,8,9]. 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

[*] Corresponding author at: Decision Systems Laboratory, School of Information Sciences and Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA.
[**] Principal corresponding author at: Decision Systems Laboratory, School of Information Sciences and Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA.
E-mail addresses: parotr@gmail.com (P. Ratnapinda), marek@sis.pitt.edu (M.J. Druzdzel).

http://dx.doi.org/10.1016/j.jal.2015.03.007
1570-8683/© 2015 Elsevier B.V. All rights reserved.

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 we use the newly acquired data to refine the model. Incremental learning approaches 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., groups 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 discrete Bayesian network models [18], which are becoming increasingly popular in modeling and learning tasks. While there are other ways of updating Bayesian network parameters (e.g., [16]), the most flexible algorithm for learning discrete Bayesian network parameters is the EM (Expectation Maximization) algorithm [6,13]. While there are several variants of the EM algorithm, two are most notable: the basic EM algorithm [6] and the *online EM* algorithm [2,14,19].

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, captured by a number called the *equivalent sample size* (ESS). 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. As each of the algorithms requires belief updating, their complexity is worst-case NP-hard [4]. Their relative complexity differs, although it is driven largely by the number of data records that they have to process. The computational complexity of the incremental batch EM depends primarily on the size of the set of additional records, i.e., the mini-batch. The *online 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 the lowest of the three.

The question that we pose in this paper is which of the three approaches is best in practice when learning discrete Bayesian network parameters from continuous data streams. We assume these streams to be stationary, i.e., generated by systems whose parameters themselves are not changing over time, although we propose a way of approaching parameter learning when the system is non-stationary. 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., [3,20]. They agree on the obvious truth that the online learning is computationally more efficient than batch learning and show experimentally that it also achieves accuracy that is similar to that of the batch learning. Cappe [2], 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 smaller than 1000 records, batch EM is preferred to online EM. Holmes et al. [10] 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 [7] 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 EM*, *incremental batch EM*, and *online EM*. 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 EM leads consistently to the best parameter accuracy and diagnostic accuracy but may take orders of magnitude more time than incremental approaches. The incremental batch EM algorithm uses the least computation time but its performance is typically inferior to both batch EM and online EM. The online EM performs typically worse than batch EM but requires only a modest computational and storage effort.

2. Bayesian networks

Bayesian networks [18] are probabilistic graphical 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 discrete 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 some symptoms. This probability can be useful, for example, in deciding on the diagnosis or classification of the case.

3. The EM algorithm

The Expectation–Maximization (EM) algorithm [6] is an iterative method for finding maximum likelihood estimates of parameters in statistical models, when these depend on unobserved latent variables or the data contain unobserved measurements of some of the variables. It consists of two steps: (1) the expectation step (E-step) that formulates a function $S_{n,k}$ expressing the expected value of the log-likelihood function in step k, based on n observations and the previous estimates of the parameters θ_{k-1}, and (2) the maximization step (M-step), during which the algorithm computes parameters θ_k that maximize the expected log-likelihood formulated in the preceding E-step. The EM process repeats until it converges to a maximum likelihood (this may be a local maximum) or it reaches a pre-defined improvement threshold.

The task performed by the *basic EM algorithm*, when applied to learning the parameters of Bayesian networks, is estimation of the values of numerical parameters (conditional probabilities) θ of a Bayesian network given a data set (a collection of records) consisting of simultaneous observations of the network variables (with some observations possibly missing). During each iteration, the algorithm performs the E-step to calculate the expected sufficient statistics across all observations. Then, it performs the M-step to re-estimate the parameters using sufficient statistics from the E-step.

We will present a sketch of the EM algorithm using the notation of Cappé [2], who formulated the basic EM algorithm as follows: Let X_t be an unobservable random variable associated with each observation Y_t (there is one latent variable per observed data point). The pair (X_t, Y_t) is referred to as *complete data* [6]. Let $s(X_t, Y_t)$ be a vector of complete-data sufficient statistics belonging to a convex set $\mathbf{S}$. The statistics s and their calculation are problem dependent and have been shown by Cappé to reduce in case of the EM algorithm to scalars and multi-dimensional vectors of parameters. We refer interested readers to a detailed discussion of the algorithm to Cappé [2]. Given n observations, $Y_1, \ldots, Y_n$, and a set of initial values of the parameters θ_0, do, for $k \geq 1$:

$$\textbf{E-step:} \quad S_{n,k} = \frac{1}{n}\sum_{t=1}^{n} E_{\theta_{k-1}}\left[\, s(X_t, Y_t) \middle|\, Y_t\right]$$

$$\textbf{M-step:} \quad \theta_k = \bar{\theta}\left(S_{n,k}\right) \; .$$

The **E-step** formula contains expectation of the sufficient statistic $s(X_t, Y_t)$. θ_k is the estimate of parameters at iteration k. $\bar{\theta}$ is a maximization function that transforms the result of the **E-step**, which was based on θ_{k-1}, into the updated set of parameters θ_k.

In its very first step, EM starts with a set of initial model parameters and assigns them a weight called the *equivalent sample size* (ESS). ESS expresses the number of records on which the current parameters are based. When the model is not based on any data, is customary to start with uniform priors, as these are uninformed, have the largest entropy and, hence, carry least information. It is customary to use in such cases the value of ESS $= 1$.

It is possible to use the EM algorithm to refine an existing set of parameters in a model, in which case the ESS should be set to the number of records on which the current parameters are based. Such a refinement, when used for systematic model update, is often called *incremental batch EM* or simply *incremental EM*.

The *online EM* algorithm [2,14,19] is a variation on the basic EM algorithm that performs both the E-step and the M-step after each observation (each record in the data set). In the E-step, the online EM algorithm uses stochastic approximation instead of sufficient statistics:

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

Updating the model after each observation may lead to a poor approximation of the parameters, which the online EM algorithm avoids by interpolating between S_{n-1} and the expected values. The value that weights between a previous value and an expected value is a positive step size called γ_n. We use the generalized version of the online EM algorithm, proposed by Cappe [2], in the following way. Given S_0, θ_0, and a sequence of step sizes $(\gamma_n)_{n \geq 1}$, do, for $n \geq 1$:

$$\textbf{E-step:} \quad S_n = (1 - \gamma_n) S_{n-1} + \gamma_n E_{\theta_{n-1}}\left[\, s(X_n, Y_n) \middle|\, Y_n\right]$$

$$\textbf{M-step:} \quad \theta_n = \bar{\theta}(S_n) \; .$$

γ_n, the step size, is a parameter of the online EM algorithm. If we make the assumptions often used in the stochastic approximation literature that $\sum_n \gamma_n = \infty$ and $\sum_n \gamma_n^2 < \infty$, we can use $\gamma_n = 1/n^{\alpha}$, where α is typically called the learning rate. Under this assumption, the range of values for α between 0.5 and 1.0 is valid. Cappé [2] suggested that the most useful range of values is from 0.6 to 0.9, acknowledging that values close to 0.5 are more robust. In his experiments he used $\alpha = 0.6$.

We should stress here that the online EM algorithm departs from the batch EM algorithm and the incremental EM, so it is not equivalent to the incremental EM algorithm when the increment is equal to one. We illustrate this in Fig. 1, which shows plots of the average error in parameter learning (expressed by Hellinger distance, which we will discuss in more detail later in the paper) (left) and classification accuracy (right) as a function of the number of records for the incremental EM with increment size of 1 and the online EM algorithm (Letter data set of the Irvine Machine Learning Repository [7]). Clearly, both the parameter accuracy and the classification accuracy of the two algorithms are quite different.

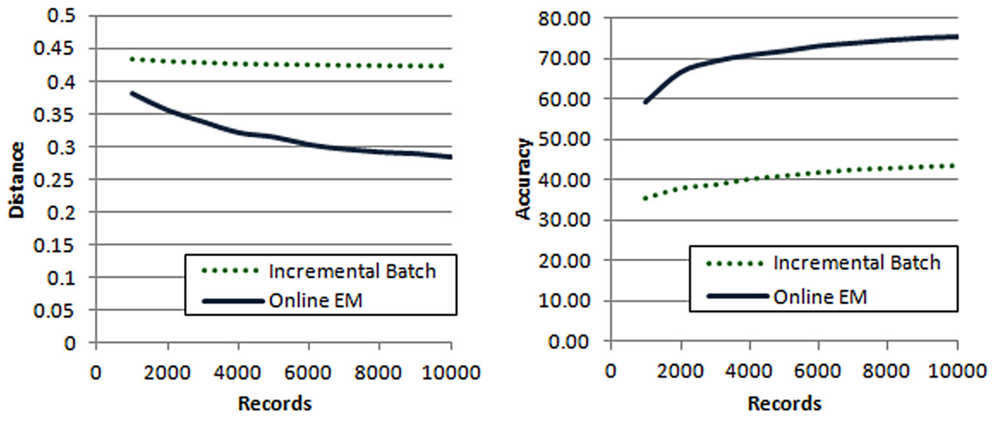

Fig. 1. Average Hellinger distance (left) and classification accuracy (right) as a function of the number of records for the incremental EM with increment size of 1 and the online EM algorithm (Letter data set).

4. Empirical evaluation

An algorithm for learning parameters of a Bayesian network can in the very best case learn the joint probability distribution of the system that has generated the data. In that sense, to evaluate a learning algorithm on a collection of data sets, we need networks that have generated the data in order to be able to evaluate the accuracy of learning. Because real data sets do not usually come with an underlying model that has generated them, we had to perform an extra step for the purpose of our experiments: Creating gold standard models and then generating data from them. Had we based everything on the original data, without the intermediate step of the gold standard model, we would not have a reliable probability distribution to compare to.

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 sizeable 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 EM, (2) incremental batch EM, and (3) online EM.

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.

4.1. The data

We selected seven data sets from the UCI Machine Learning Repository [7]: Adult, Credit, Bank Marketing, Chess (King-Rook vs. King-Pawn), Letter, Mushroom and Nursery. We decided to start out with real rather than synthetic data because we wanted the data sets used in our experiments to be as close as possible to real world data. We used the following selection criteria:

- The data include a known class variable so that we could test the classification accuracy of the learned models on a real problem (this has reduced the number of data sets to 214 from the original 299 in the repository as of the time of writing this paper).
- 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. There were only 28 data sets with categorical and 37 with mixed attributes, for a total of 65 data sets.
- The data contain a sufficient number of records for a structure learning algorithm. In addition, because we check the accuracy of the models on the original data, the larger the data set, the more reliable

Table 1
Data sets used in our experiments. #REC denotes the number of observations (records), #ATR denotes the number of attributes, #CLASS denotes the number of classes along with the percentage of records belonging to each of the classes, and MISS indicates percentage of records that contained missing values.

Dataset	#REC	#ATR	#CLASS	MISS
Adult	48,842	14	2 (76%/24%)	8%
Credit	690	14	2 (56%/44%)	–
Bank Marketing	45,211	16	2 (88%/12%)	–
Chess	3196	36	2 (52%/48%)	–
Letter	20,000	16	26 (4.07%/4.03%/4.02%/.../3.67%)	–
Mushroom	8124	22	2 (52%/48%)	31%
Nursery	12,960	8	5 (33.3%/32.9%/31.2%/2.5%/0.02%)	–

our results. Of those selected in the previous steps, there were only 20 data sets with more than 1000 records.

- The data do not contain too many missing values. Because we decided to remove the records with missing values when learning the model structure, we wanted the number of records to remain sufficient for this purpose. We chose a somewhat arbitrary cutoff point that the number of records in the data set with missing values should not be more than 1/3 of the data set.
- The selected data sets vary in the number of attributes, so that we obtain models of different size for testing.

We chose 6 data sets out of 20 after looking at their precise description and contents. We added Credit, even though it had only 690 records (our initial cutoff was 1000), because it was ideal in other respects (missing data and discretization). Some of the above selection criteria were a necessity (e.g., existence of a class variable, sufficient number of records to learn the structure reliably), others caution (e.g., avoiding discretization and dealing with missing data), yet others had to do with the external validity of our experiments (e.g., varying number of attributes). We listed the selected data sets and summarized their properties in Table 1.

4.2. The gold standard models

To learn the gold standard Bayesian networks, we applied the standard Bayesian search learning algorithm proposed by Cooper and Herskovits [5]. This algorithm does not handle missing values and continuous variables. We first discretized continuous attributes using equal frequency discretization with 5 intervals and removed all records with missing values. Only two of the selected data sets, Adult and Mushroom, contained missing values (8% and 31% of all records, respectively). We used the Bayesian learning algorithm to learn the model structure and, subsequently, we used the basic EM algorithm (uniform priors, ESS = 1) on 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 summarized their properties in Table 2.

4.3. Experiments

We used the gold standard models to generate data sets of 1,000,000 records each. We used these records to simulate data streams in our experiments. We used the structures of the gold standard models as skeleton models for learning parameters.

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

Table 2
Characteristics of the gold standard models learned from the data sets used in our experiments. The number of variables in each network is one larger than the number of attributes in the data set (reported in Table 1), #ARC denotes the number of arcs, #FP denotes the number of free parameters in the resulting BN model, μIN denotes the mean node in-degree, CPT$_{max}$ denotes the size of the largest CPT in the model, a measure of model's complexity, μST denotes the mean number of node states, and #ST$_{max}$ denotes the maximum number of node states.

Dataset	#ARC	#FP	μIN	**CPT**$_{max}$	μ#ST	#ST$_{max}$
Adult	18	3762	1.2	2022	8.5	41
Credit	18	388	1.2	98	4.9	14
Bank Marketing	28	1180	1.647	144	4.6	12
Chess	100	972	2.703	256	2.0	3
Letter	43	25,279	2.529	2600	6.1	26
Mushroom	47	4786	2.136	2058	4.5	9
Nursery	10	645	1.111	540	3.6	5

1. The basic EM algorithm applied at each step to the entire data set. We refer to it as the *batch EM*. We started running batch EM at 10,000 records and then invoked it after every 10,000 records. We used uniform priors and ESS = 1 for all runs.
2. In the *incremental batch EM*, learning happens after each k new observations 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 ESS = 1. In each subsequent step, we used the existing model as the base model and ran the EM algorithm with the ESS 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 ESS to 30,000 (the existing model had been learned from the previous 30,000 records).
3. The *online EM* updates the network parameters each time a new record becomes available. In order to get better accuracy in parameter learning Cappe [2] suggests not to perform the maximization step for the first 20 records and we adopted this advice in our experiments. We run the batch EM algorithm for the first 20 data records and use the value $S_{20,k}$ as the initial value of S_0 in the first step (21st data record) of the online EM algorithm.
4. We have noticed fairly quickly that while the online algorithm shows the strongest adaptability of all compared algorithms and an impressive convergence curve, its ultimate performance depends on the starting point: The better the starting point, the smaller the distance from the gold standard. We have decided to test also, in addition to the three basic algorithms, a hypothetical hybrid online algorithm (we call it *Hybrid Online EM* on the plots) that starts from a model trained by the batch EM algorithm on the first 10 K data records but proceeds from this point as the online EM algorithm.

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 also tested the classification accuracy of the learned Bayesian network models on the original data sets from the UCI Machine Learning Repository.

In all parameter accuracy measurements, we used Hellinger distance [11], which expressed the distance between two probability distributions P and G by means of the following equation:

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

It is similar to the Kullback–Leibler (KL) divergence [12], widely used in the Bayesian network 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 KL divergence that it is undefined for zero probabilities.

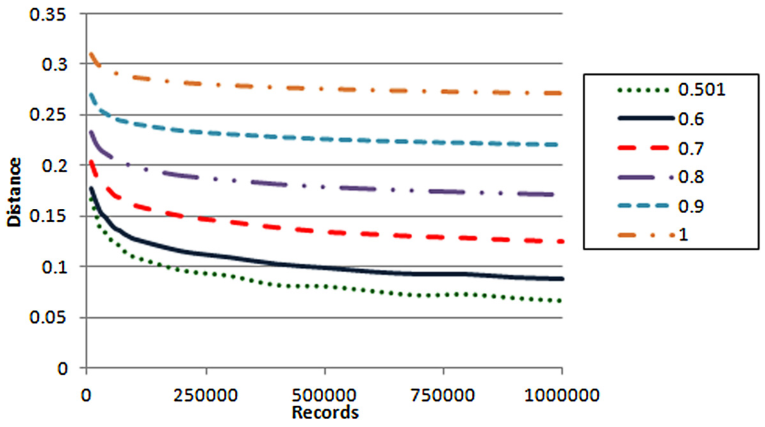

Fig. 2. Average Hellinger distance as a function of the number of records in the data stream for different learning rates α (Adult data set).

In a preparatory step, we measured the impact of various values of the learning rate α on the accuracy of the parameters as measured by the Hellinger distance between the parameters in the learned models and the original parameters in the gold standard models. As we mentioned in Section 3, Cappé [2] suggested that the most useful range of values for α is from 0.6 to 0.9 and used $\alpha = 0.6$ in all his experiments. We tested the full range of possible values of α empirically on each of the models by comparing the Hellinger distance between the gold standard and the learned parameters for various values of α. Fig. 2 shows a typical plot of Hellinger distance as a function of the number of records (Adult data set). We noticed that $\alpha = 0.6$, which performs quite well, does not seem to be the best value. Based on the observation that Hellinger distance for online EM increases with the value of α, we recommend picking a value of α as close as possible to 0.5. For the purpose of our experiments, we selected $\alpha = 0.501$.

4.4. Results

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

4.4.1. Computation time

Fig. 3 shows that the difference in terms of computation time between incremental EM and batch EM algorithms grows larger as the number of records increases. 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 3. While the comparison of computation times may not be completely fair, as the algorithms do somewhat different work, we wanted to give the reader an idea of the absolute computation time that one has to face in practice when applying each of the algorithms. The incremental batch EM and the online EM use constant amounts of time for each run and spend less computation time than the batch EM. For the last 10,000 records (Table 3), the batch EM algorithm processes 100 times more records than the incremental EM. The observed computation times differ indeed by two orders of magnitude.

The learning time for the incremental batch EM algorithm and, to a lesser degree, for the online EM algorithm decreases as more records have been processed. Given that at each step both algorithms deal with the same network and the same number of records, here is what must be happening: The EM algorithm is iterative and stops when the adjustments to parameters become smaller than a pre-set threshold value. (In our implementation, we use the threshold of the ratio of the previous to the current log-likelihoods equal to 1.0001.) As we proceed with learning and processing records, the network parameters become more and more precise and it takes fewer iterations to reach the termination threshold.

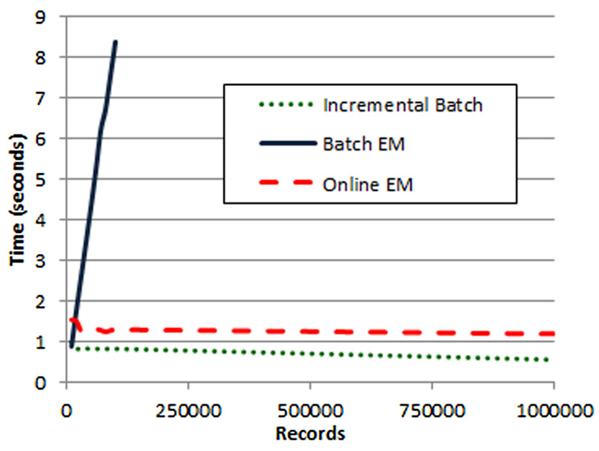

Fig. 3. Time taken to learn the parameters of the Adult data set. Time taken by the batch algorithm is linear in the number of records, we omit its run time after 100,000 records in order to show the details for the incremental batch EM and the online EM algorithms.

Table 3
Computation time required to process the 1,000,000th record shown in seconds. Please note that the incremental batch learning EM and the online EM algorithms process only the last 10,000 records (i.e., from 990,000 to 1,000,000).

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

Table 4
Final Hellinger distance and its slope for the last 100 K records.

Data set	Incremental batch	Batch	Online
Adult	0.07642	**0.01786**	0.06608
	−0.00006	−0.00073	−0.00268
Credit	0.02750	**0.00527**	0.01660
	−0.00012	−0.00018	−0.00310
Bank Marketing	0.05668	**0.00710**	0.03072
	−0.00012	−0.00030	−0.00048
Chess	0.02984	**0.00818**	0.02116
	−0.00003	−0.00053	−0.00065
Letter	0.13040	**0.04762**	0.09783
	−0.00009	−0.00043	−0.00257
Mushroom	0.09482	0.04188	**0.02109**
	−0.00005	−0.00108	−0.00028
Nursery	0.06070	**0.01341**	0.04219
	−0.00002	+0.00001	−0.00313

4.4.2. Parameter accuracy

We show the final average Hellinger distance for all data sets in Table 4. 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.

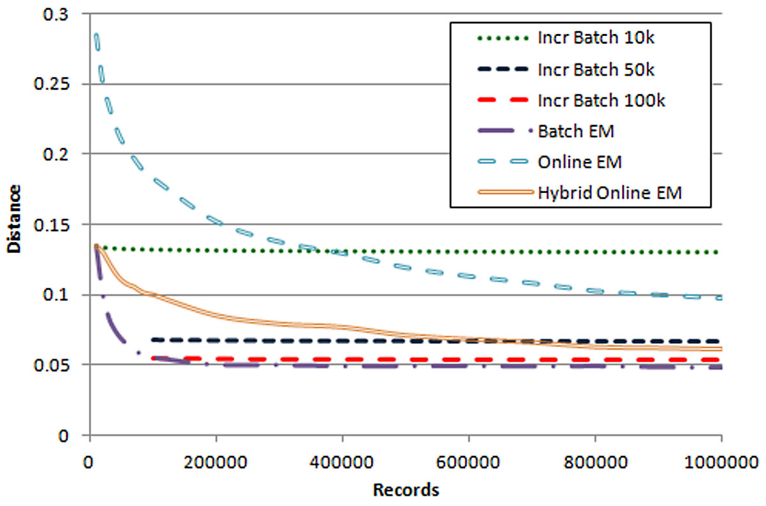

Fig. 4. A typical plot of the average Hellinger distance as a function of the number of records in the data stream (Letter data set).

In six of the seven cases, batch EM resulted in the smallest Hellinger distance, i.e., the highest accuracy of retrieving the original parameters from data. The online EM performed best only on the Mushroom data set. The negative slopes of the curves (the second number in the table) indicate that each of the algorithms improves its accuracy over time. However, this improvement is typically smaller for the incremental batch EM algorithm. Interestingly, in an experiment reported by Liang and Klein [14], consisting of four different unsupervised learning tasks in the domain of natural language processing, the online EM algorithm performed often better than the batch EM algorithm. We have no explanation for this difference except for perhaps the nature of the task.

We show a typical plot of Hellinger distance as a function of the number of records in Fig. 4. Hellinger distance for both batch EM and online EM decreases with the number of records. We added plots for the incremental batch EM algorithm with different step sizes. We confirm the findings of Holmes et al. [10] that the larger the step size, the more accurate the incremental batch EM algorithm. Based on our results, we do recommend choosing as big of a step size as computationally feasible. Incremental batch EM typically seems to reach a plateau beyond which it hardly improves the accuracy of parameters. A plausible explanation of this phenomenon is based on the fact that the incremental batch EM algorithm combines the existing parameters with the new data records using the ESS parameter. As we proceed with learning, the current parameters are based on an increasing number of records (and, hence, an increased value of the ESS). As ESS becomes larger, the relative role of the increment, which remains constant, decreases. At some point, ESS becomes much larger than the number of new records in the increment and the role of the increment is minimal. When comparing the performance of the online EM against the hybrid online EM algorithm, we can see that a better starting point makes a big positive difference in the ultimate performance of the online algorithm.

4.4.3. Classification accuracy

Evaluation of the ability of a parameter learning algorithm to retrieve the joint probability distribution that has generated the data, described in the previous section, is an important performance criterion. Another, weaker criterion of interest may be evaluation of the classification accuracy of the learned model.

In testing the classification accuracy of the learned models on the original UCI Machine Learning Repository data sets, we used a simple criterion, which is that the model guesses the most likely class to be the correct class for each record. Table 5 shows the final accuracy for each of the data sets. The difference in accuracy seems minimal, which is consistent with the finding of Onisko and Druzdzel [17], who show experimentally that Bayesian network classifiers are generally not too sensitive to precision of their parameters.

Table 5
Final classification accuracy.

Data set	Incremental batch	Batch	Online	Gold
Adult	84.21%	**84.22%**	84.20%	84.23%
Credit	**85.51%**	**85.51%**	**85.51%**	85.51%
Bank Marketing	89.48%	**89.54%**	89.52%	89.55%
Chess	**94.21%**	**94.21%**	**94.21%**	94.21%
Letter	83.97%	**87.57%**	86.01%	87.61%
Mushroom	99.85%	**99.90%**	**99.90%**	99.90%
Nursery	**94.74%**	94.64%	94.63%	94.65%

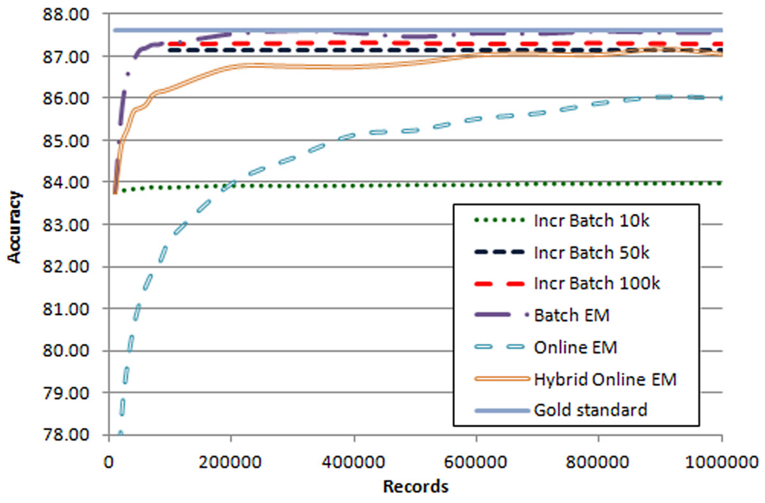

Fig. 5. A typical plot of the classification accuracy as a function of the number of records (Letter data set).

However, here again the batch EM algorithm resulted in the best classification accuracy on all data except for the Nursery data set. We show a typical plot of models' classification accuracy as a function of the number of records in Fig. 5.

Prompted by reviewers' questions, we would like to stress that classification accuracy by itself is not the best criterion for comparing models. In some applications, the precise numerical probabilities are more important than the qualitative, ordinal relationship among the posterior probabilities. When we want to support decisions that include valuations of outcomes (utilities), for example, we need to weigh the utilities of outcomes by the probabilities of these outcomes. The first criterion, i.e., precision of learning the parameters and Hellinger distances between the learned and the gold standard probability distributions is, in our opinion, more critical.

4.4.4. Missing values

In order to check whether missing data have impact on our results, we used the two data sets of 1,000,000 records with 10 percent and 30 percent values missing. We obtained qualitatively similar results with a slight overall increase of the Hellinger distance and a decrease of classification accuracy (Tables 6 and 7). Here also, batch EM was typically better than online EM, which was better than the incremental batch EM.

4.4.5. Original rather than generated data sets

To probe the question whether generation of data streams (as opposed to taking the original Irvine repository data) made a difference in our results, we repeated our experiments on the original records belonging to the four largest data sets (Adult, Bank, Letter and Nursery). Each of these sets has a sizeable number of records (reported in Table 1), although not on the order of a million. We run the incremental batch EM algorithm in increments of 1000 (as opposed to the 10,000 in the main experiment). The observed

Table 6
Final Hellinger distance with values missing at random. Baseline denotes no missing values, 10% denotes 10 percent values missing at random, and 30% denotes 30 percent values missing at random.

Data set	Incremental batch			Batch			Online		
	Baseline	10%	30%	Baseline	10%	30%	Baseline	10%	30%
Adult	0.076	0.084	0.103	0.018	0.021	0.027	0.066	0.072	0.084
A. Credit	0.028	0.036	0.037	0.005	0.005	0.009	0.017	0.018	0.023
Bank M.	0.057	0.068	0.085	0.007	0.008	0.019	0.031	0.033	0.042
Chess	0.030	0.036	0.046	0.008	0.008	0.018	0.022	0.023	0.034
Letter	0.130	0.146	0.178	0.048	0.049	0.062	0.098	0.108	0.136
Mushroom	0.095	0.104	0.104	0.042	0.043	0.062	0.021	0.024	0.034
Nursery	0.061	0.058	0.062	0.014	0.011	0.018	0.042	0.044	0.047

Table 7
Final classification accuracy with values missing at random. Baseline denotes no missing values, 10% denotes 10 percent values missing at random, and 30% denotes 30 percent values missing at random.

Data set	Incremental batch			Batch			Online		
	Baseline	10%	30%	Baseline	10%	30%	Baseline	10%	30%
Adult	84.2%	84.2%	84.0%	84.2%	84.2%	84.2%	84.2%	84.2%	84.2%
A. Credit	85.5%	85.5%	85.5%	85.5%	85.5%	85.5%	85.5%	85.5%	85.5%
Bank M.	89.5%	89.3%	89.3%	89.5%	89.6%	89.5%	89.5%	89.4%	89.5%
Chess	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%	94.2%
Letter	84.0%	83.5%	81.1%	87.6%	87.5%	87.3%	86.0%	85.8%	84.4%
Mushroom	99.9%	99.9%	99.8%	99.9%	99.9%	99.9%	99.9%	99.9%	99.9%
Nursery	94.7%	94.5%	94.6%	94.6%	94.6%	94.7%	94.7%	94.7%	94.8%

results are qualitatively similar to those obtained on the generated data. We show plots of the Hellinger distance for each of the data sets in Fig. 6.

4.4.6. Dealing with non-stationary systems

There is a fundamental question that one needs to ask when processing continuous streams of data: Does the system from which the data originate remain constant over time? In our experiments, we assumed that it does and, hence, we learned from all accumulated records. To our knowledge, the Irvine Repository data sets were not time series. Real systems, however, can evolve over time (e.g., [15]). 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. The difficulty in practice is that with real data we may not know whether they are non-stationary and we may not know the nature of the process and the speed with which the system that generates the data is changing. It may be a good idea to assume conservatively that the system is changing and not look too much back at data records.

Is interesting to investigate how much precision is lost by making a conservative assumption that the system is non-stationary. To test this, we assumed that the test records are time series and treated them as such. We had each algorithm look at a sliding window of 100,000 most recent records. And so, suppose that we process the record number 130,000. Batch EM algorithm learned from the records 30,001 through 130,000, incremental batch EM algorithm learned from records 120,001 through 130,000, using the existing network with ESS = 90,000, the online EM used only the record 130,000 and ESS = 99,999. We show the Hellinger distances for each of the data sets (an equivalent of Table 4) in Table 8.

We notice a significant loss of accuracy in case of some but not all data sets (compare Tables 4 and 8). While Hellinger distance increases almost three times for the Bank Marketing data set, it increases only 2% for the Nursery data set. This is the loss of precision that we would suffer by assuming conservatively that a process is non-stationary while in reality it is stationary. The computational savings for the batch algorithm (this is the only algorithm that would benefit from reducing the data set to the most recent records) are roughly proportional to the ratio of the total number of records to the sliding window size. A nice side effect

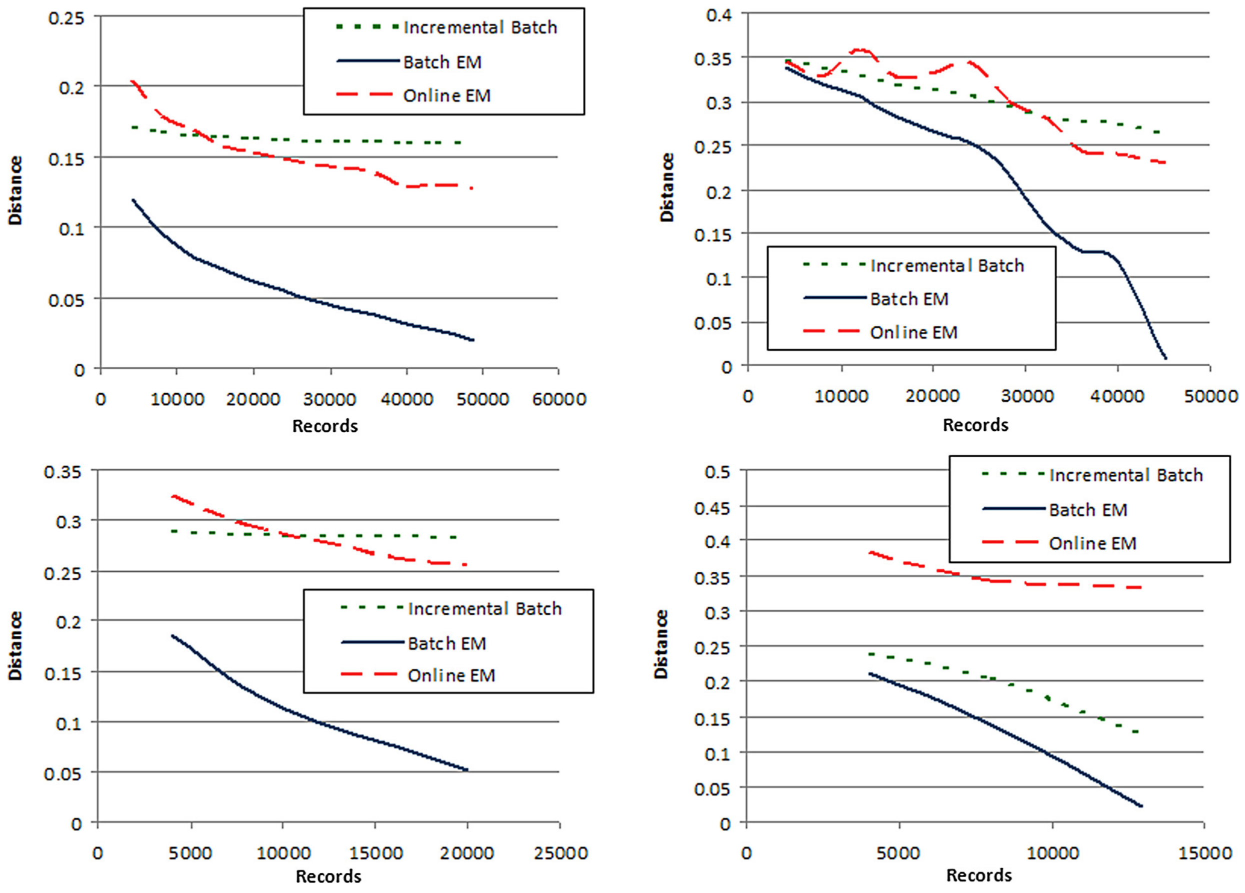

Fig. 6. Plot of the average Hellinger distance as a function of the number of records in the data stream for the original Irvine Repository data (clockwise: Adult, Bank, Letter and Nursery data sets).

Table 8
Final Hellinger distance and its slope for the last 100 K records for the sliding window experiment.

Data set	Incremental batch	Batch	Online
Adult	0.07270	**0.03960**	0.06697
	−0.00048	0.00015	−0.00203
Credit	0.03056	**0.00527**	0.01660
	−0.00030	0.00174	−0.00356
Bank Marketing	0.05170	**0.02118**	0.03576
	−0.00069	−0.00020	−0.00028
Chess	0.02832	**0.01602**	0.02025
	−0.00028	−0.00021	−0.00022
Letter	0.12672	**0.05697**	0.08392
	−0.00049	0.00231	−0.00428
Mushroom	0.09180	0.07256	**0.02302**
	−0.00037	0.00053	−0.00054
Nursery	0.05897	**0.01368**	0.04482
	−0.00069	−0.00090	−0.00469

of this assumption is that the incremental batch EM algorithm keeps improving with new records rather than reaching a plateau after some number of records (see the discussion in Section 4.4.2).

Fig. 7 shows a typical plot of Hellinger distance as a function of the number of records (similarly to Fig. 4, we use the Letter data set). Plots in this experiment showed the same order of accuracy (i.e., batch EM

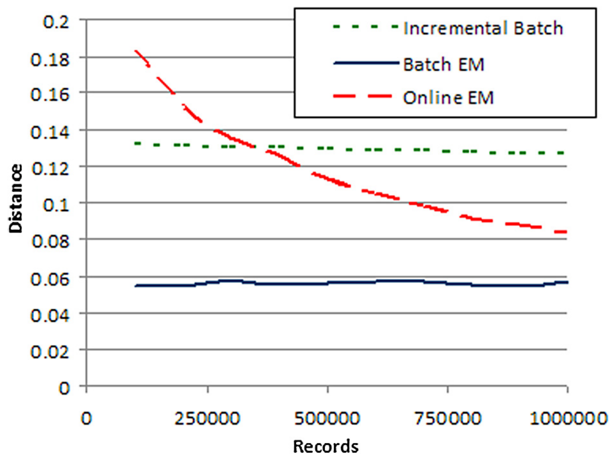

Fig. 7. A typical plot of the average Hellinger distance as a function of the number of records in the data stream using a sliding window of 100,000 records (Letter data set).

being most accurate and incremental batch being least accurate). They are, however, more flat, which is not surprising, given that the number of records that the model is trained on is (except for the first 100,000 records) always equal to the sliding window of most recent 100,000 records.

We have conducted this experiment for three different data sets: (1) data set with no missing values, (2) data set with 10% missing values, and (3) data set with 30% missing values. Since no new insights emerged from this experiment, given space constraints, we are not reporting the exact numerical results.

5. Discussion

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

Our results indicate that while batch learning, i.e., using at each step the basic EM algorithm applied to the entire data set, makes the largest computational and storage demands, it also offers the highest resulting parameter accuracy. Online EM requires less computation time and minimal amount of storage while achieving better accuracy than the incremental EM algorithm for all but small data sets.

We advise to use batch EM 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 EM 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 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 Oniśko and Druzdzel [17] that the accuracy of discrete Bayesian network classifiers is quite insensitive to precision of their numerical parameters.

Acknowledgements

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 https://dslpitt.org/genie/. We would like to thank Mark Voortman and Martijn de Jongh for their help in implementing the online EM algorithm. While we take full responsibility for any remaining errors and shortcomings of the paper, we thank anonymous reviewers for the 26th International Florida Artificial Intelligence Research Conference (FLAIRS-2013), where we presented an early version of this paper, and reviewers for the *Journal of Applied Logic* for valuable suggestions, questions, and corrections that led to additional experiments and significantly improved the quality of the paper.

References

[1] J.M. Agosta, T.R. Gardos, M.J. Druzdzel, Query-based diagnostics, in: M. Jaeger, T.D. Nielsen (Eds.), Proceedings of the Fourth European Workshop on Probabilistic Graphical Models, PGM-08, Aalborg, Denmark, 2008, pp. 1–8.
[2] O. Cappe, Online expectation maximisation, in: K.L. Mengersen, C.P. Robert, D.M. Titterington (Eds.), Mixtures: Estimation and Applications, John Wiley & Sons, Ltd., 2011, pp. 31–53.
[3] L. Carbonara, A. Borrowman, 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, Springer-Verlag, London, UK, 1998, pp. 264–272.
[4] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artif. Intell. 42 (1990) 393–405.
[5] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learn. 9 (1992) 309–347.
[6] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc., Ser. B 39 (1977) 1–38.
[7] A. Frank, A. Asuncion, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2010.
[8] J. Gama, Knowledge Discovery from Data Streams, Chapman & Hall/CRC, 2010.
[9] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Trans. Neural Netw. 22 (2011) 1901–1914.
[10] G. Holmes, R. Kirkby, D. Bainbridge, Batch incremental learning for mining data streams, Working Paper, Department of Computer Science, University of Waikato, 2004, http://researchcommons.waikato.ac.nz/handle/10289/1749.
[11] G. Kokolakis, P. Nanopoulos, Bayesian multivariate micro-aggregation under the Hellinger's distance criterion, Res. Off. Stat. 4 (2001) 117–126.
[12] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1951) 79–86.
[13] S.L. Lauritzen, The EM algorithm for graphical association models with missing data, Comput. Stat. Data Anal. 19 (1995) 191–201.
[14] P. Liang, D. Klein, 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, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 611–619.
[15] A. Łupińska-Dubicka, M.J. Druzdzel, 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, 2012, pp. 219–226.
[16] K. Olesen, S. Lauritzen, F. Jensen, aHUGIN: a system creating adaptive causal probabilistic networks, in: Proceedings of the Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence, UAI-92, Morgan Kaufmann, San Mateo, CA, 1992, pp. 223–229.
[17] A. Oniśko, M.J. Druzdzel, Impact of precision of Bayesian network parameters on accuracy of medical diagnostic systems, Artif. Intell. Med. 57 (2013) 197–206.
[18] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.
[19] M.-A. Sato, S. Ishii, On-line EM algorithm for the normalized Gaussian network, Neural Comput. 12 (2000) 407–432.
[20] D.R. Wilson, T.R. Martinez, The general inefficiency of batch training for gradient descent learning, Neural Netw. 16 (2003) 1429–1451.