Fast Online Estimation of the Joint Probability Distribution

J.P. Patist

Vrije Universiteit Amsterdam, Computer Science,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
jpp@cs.vu.nl

Abstract. In this paper we propose an algorithm for the on-line maintenance of the joint probability distribution of a data stream. The joint probability distribution is modeled by a mixture of low dependence Bayesian networks, and maintained by an on-line EM-algorithm. Modeling the joint probability function by a mixture of low dependence Bayesian networks is motivated by two key observations. First, the probability distribution can be maintained with time cost linear in the number of data points and constant time per data point. Whereas other methods like Bayesian networks have polynomial time complexity. Secondly, looking at the literature there is empirical indication [1] that mixtures of Naive-Bayes structures can model the data as accurate as Bayesian networks. In this paper we relax the constraints of the mixture model of Naive-Bayes structures to that of the mixture models of arbitrary low dependence structures. Furthermore we propose an on-line algorithm for the maintenance of a mixture model of arbitrary Bayesian networks. We empirically show that speed-up is achieved with no decrease in performance.

1 Introduction

In recent years, the emergence of applications involving massive data sets such as customer click streams, telephone records, multimedia data, has resulted in extensive research on analyzing data streams within the field of data mining [2]. The computational analysis of data streams is constrained by the limited amount of memory and time.

Furthermore the process, which generates the data stream, is changing over time. Consequently the analysis should cope with changes in the data distribution.

Probability distribution estimation is used widely as a component of descriptive analysis, outlier -and change detection, etc. Thus, it is a necessarily component in almost any data stream analysis system. Popular models used for representing the joint probability distribution are Bayesian networks, dependency models, mixture models, markov models and wavelets.

In this paper we restrict ourselves to the following models: dependency models, Bayesian networks and mixture of Bayesian networks. The optimal dependency tree can be found efficiently using the The Chow-Liu [3] algorithm. Bayesian

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 689–696, 2008.

network exhibit more expressive power than dependency tree, but learning is more expensive. So, on the one hand, we have a simple model which can be maintained efficiently and on other hand a powerfull model but with larger maintenance cost. Our solution, namely mixture models, provides a powerfull model and a less demanding estimation procedure. The parameters are learned by an online EM-algorithm and the structure is adjusted by randomly adding components and dependencies.

The paper is structured as follows. First we report on related work on online EM-algorithms and joint probability distribution estimation. Hereafter we define the problem as well the batch and online algorithm. Experiments on synthetic data show the performance of the system compared to two baseline models, namely dependency trees and Bayesian networks.

2 Related Work

In reporting on related work we restrict ourselves to the topics of the estimation of the joint probability distribution in the context of data stream mining and on-line or one-pass versions of the EM-(Expectation Maximization) algorithm [4].

A major part of the research on on-line versions of the EM-algorithm like [5] is based on stochastic approximation and can be seen as special types of the Robbins-Monro algorithm.

In [6] a one-pass EM algorithm is proposed for the estimation of Gaussian mixtures in large data sets. Data points are compressed into sufficient statistics or discarded based on their clusterness. The method seems scalable in the number of records and more accurate than sampling.

In [7] a method is proposed to sequentially update the dependency structure of a Bayesian network. The Bayesian network is updated using a search buffer for possible neighborhood structures.

In [8] a density estimation procedure is proposed based on kernel estimation. A buffer is maintained in which kernels over data point are stored. The buffer is compressed whenever it reached the maximum buffer size. The procedure has linear time complexity.

In [9] the wavelet density estimation technique is adapted to the context of data streams. The wavelet density estimators require fixed amount of memory and is updated in an on-line manner.

In [1] the joint probability distribution is estimated using mixtures of Naive-Bayes models. The Naive-Bayes basis model is a Bayesian network with the constraint of independence between all 'non-target' variables given the 'target' variable. The mixture model was comparable in accuracy to Bayesian networks.

3 Problem Description

We define a data stream s as a possibly infinite sequence of random variables $X_i^d, \ldots, X_n^d$ from the discrete nominal domain N^d. The objective is to have an any time up-to-date accurate estimate of the underlying joint probability

distribution of s. Furthermore, the estimation is constrained by time and space and must have low time and space complexity.

We translate this problem to finding the probability mass function F which maximized $L(X^t, F)$ indexed by t. The likelihood function L is defined as: $L(X_{i..n}, F) = \prod_i^n F^{t=i}(X_i)$.

In this paper, Bayesian networks, mixture models, dependency trees and mixture of independence models constitute for F. In the on-line setting we want to optimize $L(X_{i..n}, F^t)$, the likelihood over the sequence $X_i, \ldots, X_n$, where F^t is the probability mass function at time t.

4 Mixture Models

A mixture model is a convex combination of k probability mass function: $P(X_i) = \sum_{s=1}^k \alpha_s P_s(X_i)$. A d-dimensional mixture of arbitrary Bayesian networks is a probability mass function on N^d that is given by a convex combination of k Bayesian networks: $P_s(X_i) = \prod_j^d P_s(X_i^j | \texttt{parents}(X_i^j))$, where $\texttt{parents} \subset \cup_d X^d$ and the underlying dependency structure of the data distribution does not contain loops. The Likelihood of a data record i given a Bayesian network is equal to the product of local probabilities. X_i^j is the j-th attribute of record i. Note that dependency trees are Bayesian models only with an extra constraint on the dependency structure.

Given a set $D = \{X_1, \ldots, X_n\}$ of independent and identically distributed samples from P(X), the learning task is to estimate the parameter vector $\Theta = \{\alpha_s, CPT_{k,i}\}_{s=1}^k$ components that maximizes the log-likelihood function $L(\Theta; D) = \sum_{i=1}^k \log P(X_i)$. Maximization of the data log-likelihood $L(\Theta)$ is usually achieved by running the EM-algorithm. The standard batch-EM algorithm starts with parameters set to some initial values, and then iteratively repeats two steps (the E-step and the M-step) trying to improve the value of $L(\Theta)$ by adjusting. It terminates after a pre-specified number of iterations, or when the improvement rate drops below a certain threshold. In the E-step, the EM-algorithm finds the contributions $q_i(s)$ of the points X_i to all mixture components, $q_i(s) = p(s_j | X_i)$.

4.1 Estimating Mixture Models Using On-Line EM

In [10] improvements are investigated such as block-EM. In block-EM both the E and M-step are performed over blocks of data. Per block the sufficient statistics are stored and replaced when it is used to update the model. This enables the model to incorporate information faster.

In this paper we extend the work of [10] by exponentially weighting sufficient statistics. We update the model after observing a buffer of b data points. Then the formulas become: $S_{b,\alpha_k^{(t+1)}} = \sum_{x \epsilon b} P(s_k | x^{t+1})$, $S_{b,\theta_{v,k}^{(t+1)}} = \sum_{x \epsilon b} P_v(s_k | x_v^{t+1})$ and $\alpha_{k,b} = (1-\lambda)\alpha_{k,t} + \lambda S_{\alpha_{b,k}}^{t+1}$, $\quad \theta_{k,b} = (1-\beta)\theta_{k,t} + \beta S_{\theta_{b,k}}^{t+1}$. S_α, S_θ are sufficient statistics for α and θ(=CPT).

The advantage of updating the model after b points is two-fold. Firstly, it saves computational effort by not having to update the model parameters after each point, but only after each b points. Secondly, because of the more reliable estimates of the expectations improvement of the performance is more stable and faster.

4.2 Adapting Model Structure

To facilitate faster adaptation to changes in the true data distribution we use four different techniques: adding and deleting components and, adding and deleting dependencies in basis models. Components are added by probability dependent on the number of components in the mixture model and is bounded by a maximum. In practice the formula equals: $P(\text{add component}|\text{ncomp=n}) = \eta(1 - \frac{n}{n_{\max}})$ It follows that if $n = n_{\max}$, the probability of adding a component is 0. The new component is a full-independence model with uniform random parameters. The prior of the new component is set equal to prior$\alpha_{\min}$. The priors of the remaining basis models are adjusted relatively, such that $\sum \alpha_i = 1$.

A Component is deleted when its prior drops below some threshold t. To ensure that components are not deleted immediately after insertion, deletion is constrained by $\min_{age}$. After deletion the priors $\alpha.$ are normalized.

The structure of individual basis models is changed by removing and adding dependencies in the Bayesian model. Dependencies are added randomly from the set of eligible dependencies. The maximal number of parents is 1. The probabilities of the new dependency is set such the marginal distribution is equal the distribution before adding. Assuming dependencies which are not supported by the data can be harmful.

We represent every conditional probability table (CPT) by a mixture model of two components, $CPT_{X|Y} = P(X|Y) = \lambda\hat{P}(X) + (1 - \lambda)\hat{P}(X|Y)$. This mixture representation is only used in the on-line mixture model. Parameters are estimated using the EM-algortihm. We interchange $P(X|Y)$ by $P(X)$, when $\lambda > h$. This corresponds to the deletion of the dependency $X|Y$, the prior is set to zero and the dependency deleted.

4.3 Bayesian Networks and Dependency Trees

Dependency trees and Bayesian networks are use as a reference to our methods. Learning in Bayesian networks is divided into two tasks: structure learning and parameter estimation. Structure learning is the learning of the optimal topology of the network. All parameters can be estimated by determining frequencies of combinations of variable values. Examples of structure learning algorithms are: $K2$, $MCMC$, etc.. We use the $K2$ algorithm [11]. Dependency trees can be constructed more efficiently than Bayesian networks. A dependency tree is a Bayesian network with the extra constraint that each variable has only one parent. The optimal tree can be constructed in polynomial time [3].

4.4 Time Complexity

The complexity of the Chow-Liu algorithm is $O(nd^2 + d^2c^2)$, of $K2$ it is $O(n \max_p^2 d^2 c)$, Batch EM for mixture of Bayesian networks: $O(iknd + \frac{ikcnd}{b})$ and On-line EM for mixture of Bayesian networks: $O(knd + \frac{kcnd}{b})$. The variables n, d, i, c, max_p correspond to the number of data points, the dimensionality, the number of iterations of the EM-algorithm, the maximum number of values per variable and the maximum number of parents.

The advantage of the on-line EM-algorithm is the spread of the computational effort over the data points, resulting in an $O(kdc/b)$ average time complexity per data point. The parameter b corresponds to the number of data points between two model updates. In the experiments $b = 50$.

5 Experiments

In the first experiment we investigate the performance of the different methods on different kind of stationary artificial data sets. In the second experiment we explore the performance on dynamic data.

The models that are compared are: dependency trees, Bayesian networks and the proposed on-line EM-algorithm for arbitrary low complexity Bayesian networks as well as the batch EM variant. Performance is measured by the log likelihood on a test set.

We generated 50 different data sets differing in cardinality, complexity and dimensionality. The cardinality, the number of different values per variable is, 2, 5 and 10 values and equal for all variables. The complexity of the underlying Bayesian models is expressed in the number of dependencies. The number of dependencies in the experiments is set to 1, 3 and 5. If the complexity is 1, a variable has one parent, if no loops are present, consequently when 3 then three parents. The adjacency matrix and the probabilities of the conditional probability tables are generated at random.

The Bayesian networks are build using the Murphy toolbox [12] for Matlab using K2. The K2-algorithm was constraint by a maximum of 3 parents. We called the algorithm 10 times with different order on the variables and selected the network with the highest log likelihood on the training set.

In case of the batch-EM algorithm, the algorithm is stopped when 100 iterations of E and M-steps are performed or when the log likelihood did not improve significantly. The structure as well the number of the basis models is fixed after initialization.

The online-EM algorithm is initialized in the same way as the batch variant. Note that in the case of the online variant we use a basis model which represents the conditional probability table as a mixture of the full probability table and the marginal probabilities. The online-EM calculates sufficient statistics over blocks of 50 data points. Every 100 data points components and dependencies are pruned and added. Components are pruned when the $prior < \frac{1}{50k}$. Components are added by probability $p = (ncomp_{max} - ncomp_{model})/(2 \cdot ncomp_{max})$. The

learning rate is uniformly decreased from 1 to 0.01. Dependencies are removed when the prior of the marginal distribution is larger than 0.5. Dependencies are added at random when possible, thus not creating loops and not exceeding the maximum number of parents.

In the experiments the performance is determined using different sizes of data streams and different number of components. The number of components used are: 5, 10, 15 and 20 components and the sizes are: 1.000, 10.000, 20.000, 30.000, 40.000, 50.000 data points. The data dimensionality is 10 or 20.

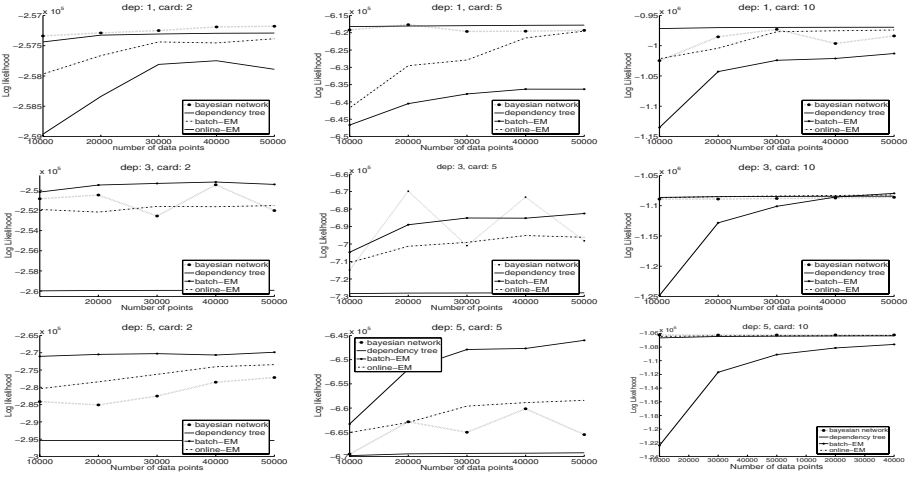

Fig. 1. The performance of Bayesian networks, dependency trees and mixture models using batch EM an on-line EM on 9 different data sets as a function of different data sizes. The data sets have 10 dimensions and differed on the number of different values per variable and the complexity of the underlying Bayesian network. The mixture models contained 20 basis model components.

6 Results

In Figure 1. is shown the typical behavior of the different algorithms on the data stream. We generated more data sets than shown, however these figures show the typical behavior. The figure shows the results of the mixture models with 20 components. A common picture, except for batch EM, is, the more components the better the performance. In case of batch-EM, the more components the slower the improvement. The online-EM did not seem to suffer from this. The batch EM-variant performed best on data streams with complexity of at least 3, on which the on-line variant and Bayesian networks are comparable.

Whenever the cardinality of the variables was 10 the 'convergence' of the basic-EM was relatively slow. This seems to hold for all complexity data sets with cardinality of 10. Both mixture models seem to perform relatively worst in

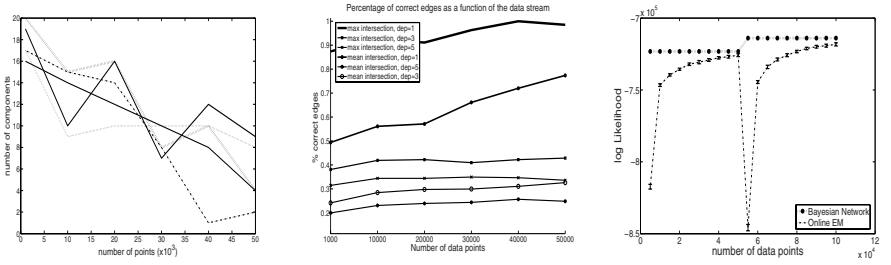

Fig. 2. From left to right: 1. The number of components as a function of the size of the data stream. The data is generated from a dependency tree. 2. The maximum and mean number of correct dependencies per component in the case of 3 different data streams. 3. The Log Likelihood as a function of a changing data stream as well the standard deviation over 10 runs initialized by 10 different structures.

the case complexity 1. The online variant approaches the highest accuracy when the cardinality is 5 and 10 and complexity is 1. This is due random search over the dependencies.

In Figure 2.1. is shown the number of components as a function of the data stream from data sets with complexity 1. There is a clear linear trend in the number of components and the size of the data stream. Probably the priors of the better fitting mixture components are growing faster than others.

In Figure 2.2. is shown the number of correctly estimated dependencies, we see an large improvement when the complexity is 1. When the number of correct dependencies is one, every basis component has the same dependency structure as the underlying Bayesian network. We plotted the maximum and average number of correct dependencies over the total number of basis models. The maximum approaches almost 1, and the average to 75 percent correct.

In the second experiment the data distribution is changed abruptly at data point 50.000. As we can see in Figure 2.3. the Log Likelihood drops and recovers. The top line is the performance of a Bayesian network build on 50.000 data points. The data is generated from Bayesian networks of complexity 2.

7 Conclusion and Future Work

We proposed a fast online-EM algorithm for the mixture of arbitrary Bayesian networks. The method iteratively changes the structure of the basis components in search of better structures. It outperforms the batch variant with respect to speed and in some cases improves on accuracy. Our method is comparable to the Bayesian network structure finding algorithm K2. When the dimensionality increases the Bayesian network is outperforming both methods. Our method outperforms the other methods with respect to speed in case the number of components are of reasonable size. In case of dynamic data streams the online-EM algorithm recovers from a change in the underlying data stream.

Random structure does not always effectively search the space. If the space grows the usefulness of random search will decrease. Thus, possible future work is a more directed randomized search procedure at the same computational cost, constraining the space of structures or using the Chow-Liu algorithm.

References

1. Lowd, D., Domingos, P.: Naive bayes model for probability estimation. In: Twenty-Second International Conference on Machine Learning, pp. 529–536 (2005)
2. Aggarwal, C.: Data Streams: Models and Algorithms. Springer, Heidelberg (2007)
3. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. Transactions on Information Theory, 462–467 (1968)
4. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B, 1–38 (1977)
5. Sato, M.A., ishii, S.: On-line EM algorithm for the normalized gaussian network. Neural Computation 12(2), 407–432 (1999)
6. Bradley, P., Fayyad, U., Reina, C.: Scaling EM(expectation maximization) clustering to large databases. In: Technical Report MSR-TR-98-35, Microsoft Research (1998)
7. Friedman, N., Greiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning, 103–130 (1997)
8. Zhou, A., Cai, Z., Wei, L., Qian, W.: M-kernel merging: Towards density estimation over data streams. In: DASFAA 2003: Proceedings of the Eighth International Conference on Database Systems for Advanced Applications, pp. 285–292. IEEE Computer Society, Washington (2003)
9. Heinz, C., Seeger, B.: Wavelet density estimators over data streams. In: The 20th Annual ACM Symposium on Applied Computing (2005)
10. Thiesson, B., Meek, C., Heckerman, D.: Accelerating em for large databases. Machine Learning 45(3), 279–299 (2001)
11. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning, 309–347 (1992)
12. Murphy (2004), `http://www.cs.ubc.ca/~murphyk/software/bnt/bnt.html`