EnBay: A Novel Pattern-Based Bayesian Classifier

EnBay: A novel pattern-based Bayesian classifier

Elena Baralis, Luca Cagliero, and Paolo Garza

E. Baralis and L. Cagliero are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy. E-mail: {elena.baralis, luca.cagliero}@polito.it. P. Garza is with the Dipartimento di Elettronica e Informazione. Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milano, Italy. E-mail: {garza}@elet.polimi.it

Abstract

A promising approach to Bayesian classification is based on exploiting frequent patterns, i.e., patterns that frequently occur in the training dataset, to estimate the Bayesian probability. Pattern-based Bayesian classification focuses on building and evaluating reliable probability approximations by exploiting a subset of frequent patterns tailored to a given test case.

This paper proposes a novel and effective approach to estimate the Bayesian probability. Differently from previous approaches, the *En*tropy-based *Bay*esian classifier, namely EnBay, focuses on selecting the minimal set of long and not overlapped patterns that best complies with a conditional-independence model, based on an entropy-based evaluator. Furthermore, the probability approximation is separately tailored to each class. An extensive experimental evaluation, performed on both real and synthetic datasets, shows that EnBay is significantly more accurate than most state-of-the-art classifiers, Bayesian and not.

Index Terms

H.2.8.b Clustering, classification, and association rules, H.2.8.d Data mining

I. Introduction

Classification aims at defining an abstract model of a set of classes, called classifier, which is built from a set of labeled data, i.e., the training set. The classifier is then used to appropriately classify new data for which the class label is unknown. Different approaches have been proposed to build accurate classifiers, e.g., Bayesian classifiers [32], decision trees [23], SVMs [28], rule-based [9], and associative classifiers [4].

Bayesian classification is an established classification approach, based on Bayes theorem [32]. It predicts the class of a previously unseen test case $T = \{a_1, a_2, \ldots, a_n\}$ by selecting the class c_i that maximizes the following formula

$$P(c_i|T) = \frac{P(T, c_i)}{P(T)} = \frac{P(c_i) \cdot P(T|c_i)}{P(T)} \tag{1}$$

where $P(T|c_i)$ denotes the conditional probability of the test case T given class c_i. Probabilities are estimated from the training set. Since classification focuses on selecting the class that maximizes Formula 1, rather than assigning explicit probabilities to each class, the denominator $P(T)$ in Formula 1 can be omitted, as it does not affect the relative class order.

The hardest task in Bayesian classification is the computation of the joint probability $P(T, c_i)$. Despite its simplicity, the Bayesian approach proves to be computationally intractable without enforcing strong model simplifications [7], [12], [32]. A prominent example of simplification is the Naive Bayes classifier [32], which tackles the issue by assuming that all the attributes are conditionally independent given the class c_i. Hence, the joint probability in Formula 1, based on the generative Naive Bayes model, can be approximated as follows.

$$
\begin{aligned}
P(T, c_i) &= P(a_1, a_2, \ldots, a_n, c_i) \\
&\simeq P(c_i)P(a_1|c_i)P(a_2|c_i)\cdots P(a_n|c_i) \\
&= P(c_i)\prod_{j=1}^{n} P(a_j|c_i)
\end{aligned} \tag{2}
$$

In this case, classification only depends on the values of all $P(a_j|c_i)$ and on $P(c_i)$. Unfortunately, the conditional independence assumption made by Naive Bayes is rarely true on real-life data.

Several techniques have been proposed to overcome the strong independence assumption. The most accurate state-of-the-art Bayesian approaches (e.g., [27], [40]) improve Naive Bayes by learning conditional-dependence models. For instance, Bayesian Networks are conditional-dependence graph-based models that represent conditional dependencies among attributes given the class under exam. To reduce the computational complexity of the structure learning process, they commonly focus their attention on a subset of low order dependencies, thus neglecting some of the higher order ones. To also consider high order correlations during the probability estimate, a parallel research effort has been devoted to exploiting frequent patterns to build an approximation of the joint probability $P(T, c_i)$ [13], [39]. Frequent patterns represent recurrent correlations among training data and can be used to compute a more precise estimate of the probabilities of interest. Given an itemset X=$\{a_1, a_2, \ldots, a_n\}$, its frequency can be considered as a reliable estimate of $P(a_1, a_2, \ldots, a_n)$. Since X involves simultaneously all the a_j of interest, the estimate is intuitively better than that based on the conditional independence assumption made by Naive Bayes. For each probability of interest, the most appropriate set of itemsets used to compute the estimate is selected by means of ad-hoc heuristics. Motivated by probability theory [33], all the previous pattern-based approaches [13], [39] focus on selecting the highest number of long and overlapped itemsets covering T. They are based on the idea that the more long overlapped patterns are included in the product approximation, the more inner attribute dependencies are captured in it. However, the approximation is weakly bound to the class under evaluation, because it is generated once for all the classes. Furthermore, the need of extracting a large number of long and overlapped frequent itemsets causes both the quality of the approximations to be sensitive to the support threshold variations, and the classification algorithms to be unable to cope with large datasets.

We propose a radically different pattern-based approach to estimate the probability approximation, that aims at complying with the independence assumption among the terms that build the approximation. More specifically, the EnBay classifier (*En*tropy-based *Bay*esian classifier) selects the minimal set of long and not overlapped frequent patterns that best complies with a conditional-independence model [21], i.e., a model composed of conditionally mutually independent attribute sets given the class under exam. This choice guarantees the reliability of the enforced independence assumptions among the terms composing the approximation. Hence, the objective of our approach is two-fold: (i) partition the attribute set into a minimal number of large subsets so that their conditional dependence, given an arbitrary class c_i, is minimized, (ii) select frequent itemsets characterized by conditionally independent attribute sets to be included in the approximation of $P(T, c_i)$. Conditional attribute (in)dependence is evaluated,

similarly to [27], [40], by using the entropy-based conditional mutual information measure [25]. Hence, the reliability of the enforced independence assumptions is quantitatively evaluated by using an entropy-based measure. Differently from previous pattern-based approaches [13], [39], our product approximation is specifically tailored to each class. A thorough experimental evaluation showed that EnBay performs significantly better, in terms of classification accuracy, than most state-of-the-art classifiers, Bayesian and not. Finally, EnBay lazily constructs the product approximation for a test case T by exploiting a compact disk-based index structures built over the training data. Long frequent itemsets are dynamically and efficiently extracted by using the disk-based indexing structure to build the probability approximations tailored to T. Itemset suitability for being included in the current product approximation is evaluated on-the-fly during itemset retrieval, i.e., no postpruning phase is needed. Thus, EnBay overcomes the scalability limits of previous pattern-based approaches, which are based on main memory itemset mining algorithms.

The paper is organized as follows. Section II introduces preliminary concepts and notation. Section III describes our novel approach to pattern-based approximation estimate. Section IV thoroughly describes the EnBay classifier. In particular, Section IV-A describes the classifier training phase, while Section IV-B describes the class prediction phase. In Section V, the EnBay classifier is evaluated by means of a large suite of experiments. Finally, Section VI compares our work with previous approaches and Section VII draws conclusions and discusses future work.

II. Preliminary definitions and notation

Pattern-based Bayesian classifiers exploit data correlations discovered by means of frequent itemset mining to compute Bayesian estimates. In this paper, we focus on frequent itemset mining and pattern-based Bayesian classification in the context of structured datasets. A structured dataset is a set of records. A record is a set of items, where an item is a couple (attribute_name, value). While attribute_name is the description of a data feature, value represents the associated information. More formal definitions follow.

Definition 1: **Item.** Let t_i be a label, called attribute, which describes a data feature. Let Ω_i be the discrete domain of attribute t_i. An item (t_i, value_i) assigns the value $\text{value}_i \in \Omega_i$ to attribute t_i.
In the case of continuous attributes, the value range is discretized into intervals, and the intervals are mapped into consecutive positive integers.

Definition 2: **Structured dataset.** Let $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be a set of attributes and $\Omega=\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ the corresponding domains. A structured dataset $\mathcal{D}$ is a collection of records, where each record r is a set of items and contains at most one item for each attribute in $\mathcal{T}$.

An itemset is a set of data items belonging to a structured dataset.

Definition 3: **Itemset.** Let $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be a set of attributes and $\mathcal{I}=\{(t_1, \text{value}_1), (t_2, \text{value}_2), \ldots, (t_n, \text{value}_n)\}$ contain the enumeration of all the items in the corresponding structured dataset. An itemset $X \subseteq \mathcal{I}$ is a set of items.

An itemset of length k is also called k-itemset. Given two itemsets X and Y, $X \subseteq Y$ if and only if $\forall\ (t_i, \text{value}_i) \in X$ then $(t_i, \text{value}_i) \in Y$.

Consider a generic k-itemset and a structured dataset. The observed frequency of the itemset with respect to the dataset is formalized as follows in Definitions 4 and 5.

Definition 4: **Itemset Matching.** Let $\mathcal{D}$ be a structured dataset and X=$\{(t_1, \text{value}_1), \ldots, (t_k, \text{value}_k)\}$ a k-itemset. An itemset X matches a record $r \in D$ if and only if $X \subseteq r$.

Definition 5: **Itemset support.** Let $\mathcal{D}$ be a structured dataset and X be an itemset. The support of X is given by the number of records $r \in \mathcal{D}$ matching X divided by the cardinality of $\mathcal{D}$.

Notice that support may be generalized as a binomial distribution indicating the number of successes in a sequence of boolean itemset matching tests over each record of the training set, given an arbitrary probability of success p.

Itemset mining is commonly driven by a minimum support threshold. Given a structured dataset $\mathcal{D}$ and a minimum support threshold min_sup, itemsets whose support is equal to or exceeds min_sup in $\mathcal{D}$ are said to be frequent. The frequent itemset mining problem addresses the extraction of all frequent itemsets in $\mathcal{D}$.

To perform the classification task, a given record attribute is selected as the target class label (i.e., the class attribute).

Definition 6: **Labeled structured dataset.** Let $\mathcal{D}$ be a structured dataset and $\mathcal{T}$ its corresponding set of record attributes. Let $C \in \mathcal{T}$ be the class attribute, Ω_C=$\{c_1, \ldots, c_k\}$ its domain, and let, for each dataset record r_i such that $r_i \in D$, $c_i \in \Omega_C$ be its class value. $\mathcal{D}$ is a labeled structured dataset.

A record for which the class label is known is called training (labeled) record. A record T for which the class is unknown is a test (unlabeled) record (also denoted as test case).

Given a training (labeled) structured dataset $\mathcal{D}$ (i.e., a structured dataset composed of labeled records), the classification problem consists in learning a model able to assign a class label c_i to each test record T for which the class label is unknown.

III. Product approximation

The hardest task in Bayesian classification is the computation of the joint probability $P(T, c_i)$ in Formula 1 for an arbitrary test record T. This paper proposes a novel, simple, yet effective approach to estimate $P(T, c_i)$ by means of frequent itemsets with the two-fold aim at (i) maximizing the compliance of the approximation with a conditional-independence model to guarantee the reliability of the enforced independence assumptions, and (ii) building and evaluating different class-centric approximations of $P(T, c_i)$ associated with the same test case T, one for each class label c_i.

The usage of frequent itemsets for estimating the joint discrete probability distribution has been already investigated in [13], [39]. These works propose to exploit frequent patterns to build an approximation expressed by a product form, i.e., the product approximation, of the joint probability. Section III-A introduces the concept of product approximation and motivates the proposed approach by comparing it with previous pattern-based Bayesian approaches. Section III-B presents our approach to estimate reliable product approximations and provides its main theoretical foundations. Finally, its effectiveness in Bayesian classification is shown in Section V.

A. Issues and motivations

Consider, as a running example, a structured dataset $\mathcal{D}$ composed of 5 attributes A_1, A_2, ..., A_5 and the class attribute C. Let c_i be an arbitrary class label and T=$\{a_1, a_2, \ldots, a_5\}$ a test record where, for the sake of simplicity, an item (A_i, a_i) is represented by a single character a_i. The joint probability $P(T, c_i)$ in Formula 1 may be estimated by using either conditional-dependence or conditional-independence models. In the former case, conditional dependencies among items are made explicit given class c_i. In the latter case, instead, conditional dependencies remain implicit. For instance, in Formula 3 a probability estimate of $P(T, c_i)$ based on a conditional-independence model is reported.

$$\begin{aligned} P(T, c_i) &= P(a_1, a_2, \ldots, a_5, c_i) \\ &\simeq P(c_i)P(a_1a_2a_3|c_i)P(a_4a_5|c_i) \end{aligned} \tag{3}$$

The probabilities used in the generated product approximations are estimated by considering the support values of the corresponding itemsets. In general, the probability $P(a_1, a_2, \ldots, a_n)$ is approximated by the support of the frequent itemset $\{a_1, a_2, \ldots, a_n\}$. Notice that probabilities are exclusively conditioned by class c_i and, thus, all the other underlying independencies (e.g., the independence between $\{a_1, a_2, a_3\}$ and $\{a_4, a_5\}$ given class c_i) are assumed. Differently, by using a conditional-dependence model, the joint probability $P(T, c_i)$ in Formula 1 may be rewritten by means of the following rule chain.

$$\begin{aligned} P(T, c_i) &= P(a_1, a_2, \ldots, a_5, c_i) \\ &= P(c_i)P(a_1|c_i)P(a_2|a_1c_i)\cdots P(a_5|a_1a_2a_3a_4c_i) \end{aligned} \tag{4}$$

State-of-the-art pattern-based Bayesian approaches [13], [39] mostly rely on conditional-dependence models and relax the strong Naive independence assumption by approximating at best the above rule chain. Formulas 5 and 6 are examples of valid approximations of the product form in rule chain 4.

$$\begin{aligned} P(T, c_i) &= P(a_1, a_2, \ldots, a_5, c_i) \\ &\simeq P(c_i)P(a_1a_2a_3|c_i)P(a_4a_5|a_1c_i) \end{aligned} \tag{5}$$

$$\begin{aligned} P(T, c_i) &= P(a_1, a_2, \ldots, a_5, c_i) \\ &\simeq P(c_i)P(a_1a_4|c_i)P(a_2a_5|a_1c_i)P(a_3|a_2a_5c_i) \end{aligned} \tag{6}$$

Analouglsy to the previous case, they approximate the joint probability, while assuming that all the necessary attribute independence assumptions are true (e.g., the independence assumption between the 2-itemset $\{a_4, a_5\}$ and the items a_2 and a_3 in Formula 5).

Although the discovery of the best set of patterns to include in a product approximation is known to be NP-hard, in [33] it is argued that the more itemsets are involved in the product approximation, the weaker is the conditional independence assumption. Hence, the approach first proposed in [39] incrementally adds new terms to the product approximation that are characterized by (i) a minimal number of new attributes with respect to the previously added ones (possibly one) and (ii) a maximal number of already selected attributes (possibly all). For example, the product approximation in Formula 6 would be preferable with respect to the one in Formula 5. However, the above approach has two main drawbacks. (i) A unique product approximation is selected for all classes (i.e., the "structure" of the product approximations is the same for all classes). Hence, the approximation is only weakly bound to the class under evaluation. (ii) The quality of the selected product approximation is sensitive to support threshold variations. In particular, discovering a set of long and overlapped high-quality itemsets suitable for the best coverage of a given test case [39] may require enforcing very low support thresholds during the itemset mining phase. Thus, the mining task may become unfeasible when coping with large datasets.

Differently from the state-of-the-art pattern-based Bayesian approaches, the EnBay classifier estimates and evaluates on-the-fly a potentially different product approximation for every class. Each product form complies with a conditional-independence model, like the one in Formula 3, which guarantees the reliability of the attribute independence assumptions. For instance, the discovery of a conditional independence (or a weak dependence) between the attribute sets $\{A_1, A_2, A_3\}$ and $\{A_4, A_5\}$ under class c_i may prompt the selection of the frequent itemsets $\{a_1, a_2, a_3, c_i\}$ and $\{a_4, a_5, c_i\}$ to be combined in a product form approximation for class c_i and test case T as in Formula 3.

B. A novel approach to product approximation

Consider a labeled structured dataset $\mathcal{D}$. Let T be a test case for $\mathcal{D}$. Let minsup be a non-negative integer number. We denote as FI = $\{X_1, X_2, \ldots, X_m\}$ the set of itemsets, mined from the subset of dataset $\mathcal{D}$ labeled with class c_i, whose support value exceeds min_sup. Let sup(X_i) be the support of X_i in the same data subset. We are interested in including in the product approximation of $P(T, c_i)$ a subset of disjoint frequent itemsets in FI that fully covers the given test case T with a minimal amount of terms, so that their corresponding attribute sets compose a conditional-independence model, i.e., they are maximally independent given class c_i. We formalize this task as a set covering problem.

Problem statement: Let FI be the set of frequent itemsets extracted by enforcing a minimum support threshold min_sup and $\mathcal{A}_{\mathrm{X}}$ be the attribute set covered by a generic itemset X. Let $W_{\mathcal{A}_{\mathrm{X_r}}\mathcal{A}_{\mathrm{X_j}}}$ be an estimate of the class-centric influence between the attribute sets $\mathcal{A}_{\mathrm{X_r}}$ and $\mathcal{A}_{\mathrm{X_j}}$ given class c_i. The set covering problem addressed by this paper entails the selection of the subset $\mathcal{F}$=$\{X_1, X_2, \ldots, X_k\} \subseteq$ FI that optimizes the following multi-objective optimization problem:

$$\begin{aligned} \underset{\mathcal{F}}{\text{minimize}} \quad & F(\mathcal{F}) = [F_1(\mathcal{F}), F_2(\mathcal{F}), F_3(\mathcal{F})]^T \\ \text{subject to} \quad & \mathcal{F} \subseteq \text{FI}, \\ & X_i \cap X_j = \emptyset \;\; \forall X_i, X_j \in \mathcal{F} \mid X_i \neq X_j, \\ & X_1 \cup X_2 \cup \ldots \cup X_k = T \end{aligned}$$

where $F_1(\mathcal{F})$=size($\mathcal{F}$)=$|\mathcal{F}|$, $F_2(\mathcal{F})$=$\sum_{r=1}^{k} W_{\mathcal{A}_{\mathrm{X_r}} \bigcup_{\mathrm{j=1,j\neq r}}^{\mathrm{k}} \mathcal{A}_{\mathrm{X_j}}, \mathrm{c_i}}$, $F_3(\mathcal{F})$=-$\sum_{r=1}^{k} W_{\mathcal{A}_{\mathrm{X_r}} \mathcal{A}_{\mathrm{X_r}}, \mathrm{c_i}}$ and, given a precedence operator $\prec$ in order of importance, $F_1 \prec F_2 \prec F_3$ holds.

Arranging the considered objective functions in order of importance is an established optimization method [6], [37]. In particular, lexicographical ordering is a technique that requires the decision-maker to establish the priority of each objective function. Then, solutions are first compared with respect to the most important one. In case of ties, the algorithm proceeds to compare the solutions but now with respect to the next most important objective. Hence, the search space for the least important objective functions is reduced.

In our context, the goal is to:

A) first, minimize the number of enforced (and potentially unreliable) independence assumptions by including the minimal number of terms (objective function F_1),
B) secondly, minimize the class-conditional dependency among the considered terms (objective function F_2), and
C) lastly, maximize the inner attribute dependencies which remain implicit in the current product approximation (objective function F_3).

Since, in general, set covering problems are NP-hard, we tackle the issue by adopting a heuristics that performs iterative itemset selection until the test case is fully covered. More specifically, to drive the covering process of a given test case and build the pattern-based model $P(T, c_i)$, at each iteration, the greedy itemset selection strategy is driven by the following rules.

Longest disjoint pattern. The selection of the longest disjoint candidate frequent itemset attempts to minimize the objective function F_1 by enforcing the minimal number of independence assumptions. It also allows significantly reducing the complexity of the local search (see Section IV).

Minimal dependency with already selected patterns. The candidate itemset with minimal class-centric influence of its attribute set with respect to the attributes of itemsets already in $\mathcal{F}$ is selected. This rule aims at minimizing the objective function F_2 and, thus, guaranteeing the reliability of the independence assumptions in $P(T, c_i)$.

Maximal inner pattern dependency. The candidate characterized by maximal influence $W_{\mathcal{A}_{\mathrm{X}} \mathcal{A}_{\mathrm{X}}, \mathrm{c_i}}$ among its own attributes is selected. This rule attempts to minimize the objective function F_3. It aims at considering strong underlying dependencies among attributes when building the product approximation.

Candidate frequent itemsets that satisfy the first rule are selected first. Selecting the itemsets that cover the largest part of the given test case is an attempt to minimize the number of product approximation terms. In case of ties, candidates that satisfy the first two rules are preferred. Hence, candidates whose influence with the already selected

itemsets is minimal are preferred at this stage. Finally, at equal terms, the last rule is used to discriminate among candidates by maximizing the innner dependencies which are intrinsically represented by the eligible candidates.

Discovering the independencies (or weakest dependencies) holding among attributes is the cornerstone to build reliable product approximations. A common way to evaluate dependencies in Bayesian classification is Bayesian Network learning. Bayesian Networks are graph-based structures whose nodes represent data attributes while arcs represent attribute dependencies. Bayesian Network classifiers focus on overcoming the (unreliable) Naive independence assumption by taking attribute dependencies into account during classification. Since Bayesian Network structure learning is NP-hard [7] several model simplifications have been proposed (e.g., [16], [27], [40]). Although the approximations make the learning process tractable, the corresponding probability estimates are prone to errors, because either some of the high level dependencies are neglected, or the estimates are affected by noise. Instead of learning and evaluating a complete (possibly sub-optimal) structure, we exploit a smart subset of weak (possibly null) attribute dependencies to drive the construction of the most reliable per-class product approximations.

In the context of class-centric model learning, the dependence between distinct attributes A_j and A_k can be evaluated by using the conditional mutual information [16], which is an established entropy-based measure [25] defined as follows:

$$M(A_j, A_k|c_i) \quad \simeq \quad \sum_{a_j, a_k} P(a_j, a_k, c_i) log \frac{P(a_j, a_k|c_i)}{P(a_j|c_i)P(a_k|\mathrm{c_i})}$$

By definition, instead, $M(A_j, A_j|c_i)$=0. In [40] the influence $W_{k\mathcal{A},\mathrm{c_i}}$ of a single attribute A_k on a subset of the dataset attributes $\mathcal{A}$ is computed as the sum of the corresponding mutual conditional informations.

$$\begin{aligned} W_{k\mathcal{A},\mathrm{c_i}} \quad &= \quad \sum_{j \ | \ A_j \in \mathcal{A}} M(A_j, A_k|c_i) \\ &\simeq \quad \sum_{j \ | \ A_j \in \mathcal{A}}^{M(A_j, A_k|c_i) > \phi(A_j, A_k)} M(A_j, A_k|\mathrm{c_i}) \end{aligned} \tag{7}$$

To filter out unreliable dependencies, only the terms that satisfy a minimum relevance threshold $\phi(A_j, A_k)$, derived by the Minimum Description Length (MDL), are considered. In [40] $\phi(A_j, A_k)$ is defined as

$$\phi(A_j, A_k) = \frac{log|D|}{2 \cdot |D|} |A_j| \cdot |A_k|$$

where $|D|$ and $|A_x|$ are the training dataset and the A_x attribute domain cardinalities, respectively.

For our purposes, we need to define the influence $W_{\mathcal{A}_\mathrm{q}\mathcal{A}_\mathrm{r},\mathrm{c_i}}$ of an attribute set $\mathcal{A}_\mathrm{q}$ on an arbirary attribute set $\mathcal{A}_\mathrm{r}$. It can be evaluated by extending Formula 7 as follows.

$$W_{\mathcal{A}_\mathrm{q}\mathcal{A}_\mathrm{r},\mathrm{c_i}} \quad = \quad \sum_{k | A_k \in \mathcal{A}_\mathrm{q}} W_{k\mathcal{A}_\mathrm{r},\mathrm{c_i}} \tag{8}$$

Intuitively, the influence of attribute set $\mathcal{A}_{\mathrm{q}}$ on set $\mathcal{A}_{\mathrm{r}}$ is evaluated as a linear combination of the influence of every attribute in $\mathcal{A}_{\mathrm{q}}$ on $\mathcal{A}_{\mathrm{r}}$.

EnBay adopts the influence estimator reported in Formula 8 to evaluate the class-conditional independence assumed when including a long pattern in the product approximation.

IV. The EnBay classifier

The EnBay classifier exploits frequent itemsets to build accurate Bayesian estimates. Consider a labeled structured dataset $\mathcal{D}$, a minimum support threshold min_sup, and a test case T. EnBay exploits the novel approach presented in Section III-B to estimate, for each class in $\mathcal{D}$, a product approximation tailored to test case T. EnBay is a lazy classifier, i.e., it does not create a classification model. It evaluates and selects on-the-fly a set of eligible itemsets when a new test case T has to be classified. To allow efficient pattern retrieval, EnBay first performs data storage in a disk-based compact data representation. In particular, it builds smart, tree-based data index structures [3] on the training data, separately for each class. To classify an unlabeled instance T, it visits the tree-based data index to construct and evaluates the class-centric product approximations tailored to T. It performs at the same time frequent itemset mining and on-the-fly itemset evaluation by means of the entropy-based heuristics discussed in Section III-B.

In this section the EnBay classifier is thoroughly described. Section IV-A presents the training phase of EnBay, i.e., the data index generation. Section IV-B thoroughly describes the class prediction phase of EnBay, based on on-the-fly frequent pattern evaluation.

A. EnBay classifier training

To allow efficient per-class pattern retrieval, EnBay training phase builds a separate disk-based FP-tree-based data representation [3] to compactly store the training data belonging to each class. An effective indexing technique [3] is then exploited to efficiently retrieve frequent patterns on-the-fly from the stored representation. The FP-tree is a prefix tree data structure frequently used in the context of itemset mining. It is used to compactly represent in main memory transactional datasets. Each transaction is represented by a single path in the tree, but the common prefix of two or more transactions is represented by a common prefix-path, thus reducing memory consumption. In [3], an on-disk persistent representation of FP-trees, namely Materialization Trees, is proposed. By exploiting the set of provided access primitives, it is possible to selectively retrieve from disk and load in main memory only a subset of the represented tree, driven by the specification of the subset of items to analyze. The Materialization Tree disk-based data representation allows EnBay to cope with large datasets.

Algorithm 1 shows the pseudo-code of EnBay training phase. EnBay training phase addresses the construction of the Materialization Trees, per-class index structures stored on disk, which allow efficient frequent pattern retrieval. The Materialization Tree for a given class c_i is populated by incrementally inserting all the records in $\mathcal{D}$ labeled with c_i into the corresponding FP-tree-like structure [3]. The common prefix of two records is represented as a single path [20]. A minimum support threshold may be enforced to early prune infrequent items. In this case, items

not meeting the required minimum support threshold are not included in the Materialization Tree. When no support threshold is enforced (i.e., min_sup = 0), the data representation is complete. We discuss in Section V-C the effect of minimum support threshold enforcement on the classification performance of EnBay.

Algorithm 1 EnBay Training Phase($\mathcal{D}$, min_sup)

Input: the training set $\mathcal{D}$ and the minimum support threshold min_sup
Output: MAT=$\{MT_i\}\ \forall\ c_i \in C$, a Materialization Tree for each class belonging to the training class set C
1: **for all** c_i in C **do**
2: r_{c_i} = set of all training records belonging to class c_i
3: /* per-class index population */
4: MT_i = materialize(r_{c_i}, min_sup)
5: MAT = MAT $\cup$ $\{MT_i\}$
6: **end for**
7: **return** MAT

B. Class prediction

EnBay classification of a given test case T follows the procedure described in Algorithm 2. For each class c_i belonging to the training dataset, the corresponding Materialization Tree MT_i is visited to build the class-centric product approximation $\mathcal{F}_i$ and compute the joint probability $P(T, c_i)$. Disjoint frequent itemsets in MT_i are iteratively added to $\mathcal{F}_i$ until the test case is fully covered (Lines 8-16). For a given class c_i, at each iteration EnBay builds the FP-tree projection proj_Tree$_{uncov}$ that exclusively contains items belonging to the current test case T not yet included in the product approximation $\mathcal{F}_i$ (Line 10).

The onTheFlyBestPatternExtraction procedure (see Algorithm 3) extracts the currently best itemset to be included in $P(T, c_i)$. Looking for the longest matching patterns allows a significant reduction of the candidate search space. In particular, when either the longest pattern is straightforwardly derivable (Lines 1-7), or a longer pattern can never be generated from the current projected FP-tree (Lines 19-22), the recursive visit is stopped early. During the extraction process, Procedure patternComparison compares every extracted itemset with the best current itemset (Lines 5 and 13), based on the three heuristic rules described in Section III-B. Itemset evaluation is performed on-the-fly, i.e., without the need to temporarily store frequent candidates.

Consider, as an example, a test case T=$\{a_1, a_2, a_3, a_4, a_5, a_6\}$. Longest frequent itemsets matching T are selected first. Let $\{a_1, a_2, a_3\}$ and $\{a_1, a_2, a_4\}$ be the longest frequent candidates, i.e., no other frequent itemset of length 3 of higher is contained in proj_Tree$_{uncov}$ for an arbitrary class c_1. To select the best among the two longest candidates, the dependence between each candidate and the already selected patterns in $\mathcal{F}_1$ should be considered. However, since $\mathcal{F}_1$ is initially empty, at the first iteration this rule does not discriminate. Hence, the candidate itemset that maximizes the inner dependence among its own attributes (e.g., $\{a_1, a_2, a_3\}$) is selected and included in $\mathcal{F}_1$. Once an itemset is included in a product approximation, all its items are marked as covered. Itemset selection iterates until all the items in T have been matched. In particular, the set $\{a_4, a_5, a_6\}$ of uncovered items in T is considered and the projection proj_Tree$_{uncov}$ is updated accordingly. Then, the onTheFlyBestPatternExtraction procedure selects

Algorithm 2 EnBay class prediction(MAT, T, min_sup)

Input: the set MAT=$\{MT_i\}$ of Materialization Trees, the minimum support threshold min_sup, and the test instance T
Output: the class prediction c for test instance T

1: **for all** c_i in C **do**
2: /* initialize the set of itemsets which are used to compute the product form estimation for class c_i */
3: $\mathcal{F}_i = \emptyset$;
4: /* generate the projected FP-tree involving test instance T items only */
5: proj_Tree$_i$ = projectedTree(MT_i,T);
6: /* initialize the set of items of T not already covered */
7: uncov = T;
8: **while** uncov is not empty **do**
9: /* generate the projected FP-tree involving uncov items only */
10: proj_Tree$_{uncov}$ = projectedTree(proj_Tree$_i$,uncov);
11: /* extract the best pattern from proj_Tree$_{uncov}$ */
12: bestPattern = onTheFlyBestPatternExtraction(proj_Tree$_{uncov}$, null, min_sup, $\mathcal{F}$, null);
13: $\mathcal{F}_i = \mathcal{F}_i \cup \{\text{bestPattern}\}$;
14: /* remove the items of bestPattern from the set of uncovered items */
15: uncov= uncov $\setminus$ bestPattern;
16: **end while**
17: /* compute the product form estimation for class c_i */
18: /* P($l|c_i$) is the support of l in MT_i */
19: P(T,c_i) = P(c_i) $\cdot \prod_{l \in \mathcal{F}_i}$ P($l|c_i$);
20: **end for**
21: **return** the class c_i with maximal P(T,c_i);

the next candidate to add to $\mathcal{F}_1$. In this case, let $\{a_4, a_5\}$ and $\{a_4, a_6\}$ be the longest frequent patterns. The one with minimal influence of its attribute set with respect to the attributes of $\{a_1, a_2, a_3\}$ in class c_1 is selected (e.g., $\{a_4, a_5\}$). Finally, the last iteration covers the remaining uncovered item in T by adding the frequent item $\{a_6\}$ to $\mathcal{F}_1$. If a_6 is not frequent, it cannot be added to $\mathcal{F}_1$. This issue (i.e., partial coverage of a test case) would raise the zero frequency count problem discussed below.

Once all product approximations $\mathcal{F}_i$ for a test case T are available, the Bayesian probabilities P(T, c_i) are computed for every class (Line 19 in Algorithm 2). Finally, the class c_i maximizing P(T, c_i) is returned as class prediction for T.

The zero frequency count problem. Zero frequency count takes place when a given class and feature value (i.e., item) never occur together in the training set. The resulting zero probability will wipe out the information in all other probabilities when they are multiplied in the product form. To address this issue, a small-sample correction is incorporated into all probabilities, e.g., the Laplace estimate [12] or the M-estimate correction [26]. M-estimate leads, in most cases [26], to a more accurate classification than the Laplace estimate. Hence, it has been adopted in EnBay to address the zero-count problem.

Multiple support thresholds. Frequently, when training data are unevenly distributed among classes, the use of a common support threshold for all classes may be less effective. On the one hand, by setting a high threshold, the extraction task is simpler but rare class predictions are not accurate. On the other hand, by setting a low support threshold, data overfitting may occur. Furthermore, the computational task becomes heavier, because a huge number

Algorithm 3 onTheFlyBestPatternExtraction(Tree, α, min_sup, $\mathcal{F}$, bestPattern)

Input: a FP-tree Tree, a suffix pattern α, a minimum support threshold min_sup, the set of already selected itemsets $\mathcal{F}$, the best pattern bestPattern mined so far
Output: the best pattern (itemset) to include in the product form estimation set

```
if Tree contains a single path p then
    /* the longest pattern extractable from Tree is the path in Tree union α */
    longestPattern = p ∪ α;
    /* check if longestPattern is better than the current best pattern */
    if patternComparison(longestPattern, bestPattern, F)==true then
        bestPattern=longestPattern;
    end if
else
    /* recursive extraction of frequent itemsets from Tree */
    for all item i in the header table of Tree do
        /* generate a new pattern β by joining suffix α and item i */
        β = i ∪ α;
        if patternComparison(β, bestPattern, F)==true then
            bestPattern=β;
        end if
        /* construct β's conditional pattern base and then β's conditional FP-tree */
        condPatterns = generateConditionalPatterns(Tree, β);
        /* the (recursive) itemset extraction step continues only if potential longer patterns can be mined from Tree_β */
        if length(longest pattern in condPatterns)≥length(bestPattern) then
            Tree_β = createFP-tree(condPatterns, min_sup);
            bestPattern = onTheFlyBestPatternExtraction(Tree_β, β, min_sup, F,
            bestPattern);
        end if
    end for
end if
return bestPattern;
```

of frequent itemsets satisfy the support threshold. Different support thresholds for each class may be exploited to tune EnBay accuracy performance on datasets characterized by an unbalanced distribution among classes [36]. For each class c_i, the corresponding minimum support threshold becomes $\text{min\_sup}_i = \text{min\_sup} \cdot \text{sup}(c_i)$, where $\text{sup}(c_i)$ is the frequency of class c_i, and min_sup is a *global* support value[1]. This formula, first proposed in [36], may yield more accurate predictions for rare classes, without requiring to manage an excessive number of patterns for frequent classes. An experimental evaluation of the effectiveness of multiple support threshold enforcement on datasets characterized by an unbalanced data distribution is reported in Section V-C.

V. Experimental Results

We performed a large variety of experiments to evaluate (i) the classification accuracy, (ii) the training and classification time, and (iii) the scalability of the EnBay algorithm. We also investigated the tuning of the most significant EnBay algorithm parameters and their influence on the classification performance. We ran experiments

[1]This value is not what is commonly defined as global support. It is a parameter exploited in the formula computing the local support threshold for each class [36].

on 35 datasets of the UCI Machine Learning Repository [5], whose main characteristics are listed in Table I. Continuous attribute values cannot be directly employed in classification by means of itemsets or association rules. Hence, for EnBay and all the classifiers that can not handle continuous attributes, we performed attribute discretization as a preprocessing step, while for the others the original dataset version has been considered. Automatic continuous attribute discretization is performed by applying the entropy-based method proposed in [14]. The adopted discretization technique exploits a heuristics based on the class information entropy of candidate partitions to identify the best discretization thresholds. The discretization code[2] is taken from the MLC++ Machine Learning library [31]. To study the scalability of EnBay, synthetic datasets have been generated by means of the IBM data generator [22]. Experiments were performed on a 3.2-GHz Pentium IV PC with 2.0 GBytes of main memory, running Kubuntu 6.06.

A. *Classification accuracy*

Accuracy measures the ability of the classifier to correctly classify unlabeled data. It is the ratio between the number of correctly classified data and the number of given data (including correct and wrong classifications). Accuracy has been computed by using a 10-fold cross validation test. Tables II- IV report the comparison between EnBay and a selection of classifiers belonging to a specific category, i.e., Bayesian classifiers, associative classifiers, and other (neither Bayesian, nor associative) classifiers. In particular, we considered the Weka [43] implementations of Naive Bayes, Bayes Network, HNB, AODE, LBR, K-NN, and SVMs, and the publicly available implementations of FBN [40], C5.0 [23], Ripper [9], and L^3 [4]. The experimental results show that EnBay on average performs better than all the other considered classifiers. The statistical significance of EnBay accuracy improvement is discussed in Section V-B.

For EnBay, two different configurations have been evaluated, *standard* and *tuned*. In the standard configuration the same parameter setting is used for all datasets, while in the tuned configuration algorithm parameter setting is separately tuned for each tested dataset. In EnBay standard configuration, the minimum support threshold is set to 1.5% and is fixed (i.e., it is identical for all classes), while the M-estimate correction [26] has been applied with equivalent sample size m equal to 1. In EnBay tuned configuration, while keeping the same M-estimate setting, both multiple and fixed minimum support thresholds have been tuned to their best value for each dataset. In particular, separately for each dataset, we selected the setting that achieves the best accuracy value by testing several values of fixed and multiple support thresholds in the range [0.5%, 25%]. The tuned EnBay results reported throughout the paper correspond to the support threshold settings shown in Column 13 of Table II. For the other tunable classifiers we also reported the results achieved by both configurations. For the tuned configuration, each algorithm parameter setting has been separately tuned for each dataset.

In Tables II-IV, for each dataset the accuracy of the classifier(s) that achieve(s) the highest accuracy value, by considering standard and tuned versions separately, is written in boldface. For those algorithms for which no

[2]The MLC++ discretize utility was used. Data was discretized by invoking the following command: discretize -O DISC_TYPE=entropy -O DATAFILE=<input file> -O DUMPSTEM=<output file>.

TABLE I

UCI DATASETS CHARACTERISTICS

Dataset	Transactions	Attributes		Classes	Items
		Cont.	Categ.		
Anneal	998	6	32	6	71
Audiology	226	0	69	24	178
Australia	690	6	8	2	49
Auto	205	15	10	7	106
Balance-scale	625	4	0	3	23
Breast-w	699	10	0	2	29
Cleve	303	5	8	2	27
Diabetes	768	8	0	2	15
Flare	1066	0	10	2	31
German	1000	7	13	2	58
Glass	214	9	0	7	20
Heart	270	6	7	2	18
Hepatitis	155	6	13	2	33
Horse	368	7	21	2	61
Hypo	3163	7	18	2	53
Ionosphere	351	34	0	2	144
Iris	150	4	0	3	12
Labor	57	8	8	2	29
Letter rec.	20,000	16	0	26	155
Lymph	148	3	15	4	59
Mushroom	8124	0	22	2	116
Pima	768	8	0	2	15
Satimage	6435	36	0	6	448
Segment	2310	19	0	7	171
Sick	4744	7	22	2	58
Sonar	208	60	0	2	42
Soybean l.	683	0	35	19	99
Soybean s.	47	0	35	4	72
Splice	3175	0	60	3	240
Vehicle	846	18	0	4	71
Voting	433	0	16	2	48
Waveform	5000	21	0	3	106
Wine	178	13	0	3	37
Yeast	1484	8	0	10	21
Zoo	101	0	16	7	34

TABLE II

ACCURACY COMPARISON WITH STATE-OF-THE-ART BAYESIAN CLASSIFIERS

Dataset	Naive	BayesNet		HNB	FBN	AODE	LBR	LB	LBChi^2	BCEP	EnBay		
		K2 P=1 SimpleEst α=0.1	Tuned	Mutual Information preprocessing				sup=1% τ=0.04	sup=1% τ=0.04 p=0.005	sup=1% μ=2 GR Tuned	Standard	Tuned sup % -type	Tuned acc %
Anneal	86.7	92.7	92.9	94.3	95.7	92.8	92.5	-	-	-	**98.9**	1.5-fixed	**98.9**
Audiology	70.8	**78.3**	**78.3**	74.8	74.3	70.8	71.2	-	-	-	74.2	1.5-fixed	74.2
Australia	85.4	85.2	85.2	84.2	85.5	85.9	85.1	85.7	**86.1**	86.4	85.2	7-fixed	**87.4**
Auto	67.8	74.6	74.6	85.9	**87.8**	80.5	81.0	-	-	-	85.7	1.5-fixed	85.7
Balance-scale	70.6	70.6	70.6	69.1	70.6	69.9	70.6	-	-	-	**72.1**	1-fixed	**72.2**
Breast-w	96.9	97.0	97.0	96.6	96.7	97.1	97.0	96.9	97.1	97.3	**97.2**	1-fixed	**97.4**
Cleve	**83.5**	83.2	**84.2**	83.5	83.5	83.2	83.2	82.2	82.6	82.4	82.5	10-fixed	82.6
Diabetes	77.7	77.7	77.7	76.6	**78.6**	77.7	77.7	76.7	-	76.8	77.5	10-fixed	78.1
Flare	97.4	96.7	**99.5**	99.5	99.2	99.1	99.3	81.5	81.8	80.6	99.4	1.5-fixed	99.4
German	72.5	75.6	75.6	76.3	75.5	**77.3**	75.4	74.8	75	74.7	75.0	7-fixed	75.7
Glass	72.4	73.4	74.3	**79.4**	69.6	76.2	74.3	-	-	73.7	78.0	1.5-fixed	78.0
Heart	84.1	84.1	84.1	81.9	84.1	**84.4**	84.1	82.2	81.9	81.9	83.1	5-fixed	84.2
Hepatitis	83.9	85.2	85.2	**87.7**	**87.7**	84.5	83.9	84.5	84.4	83.3	86.5	5-fixed	87.4
Horse	79.6	79.1	81.3	80.2	77.4	80.2	**82.1**	-	-	-	80.0	1.5-fixed	80.0
Hypo	97.2	97.7	97.7	97.5	97.7	98.0	**98.2**	-	-	-	97.9	1.5-fixed	97.9
Ionosphere	90.6	91.7	91.7	93.7	**94.6**	93.2	90.9	-	-	93.2	93.0	1.5-fixed	93.0
Iris	94.7	94.7	94.7	94.7	94.7	94.0	94.7	-	-	-	**95.2**	2-fixed	**95.3**
Labor	91.2	93.0	93.0	93.0	91.2	89.5	91.2	-	-	90	**96.5**	10-fixed	**98.4**
Letter rec.	74.0	74.6	74.6	89.8	90.9	88.7	85.1	76.4	85.9	84.3	**91.5**	1.5-fixed	**91.5**
Lymph	78.8	84.5	85.1	83.8	83.8	**86.5**	84.5	84.6	85.2	83.1	83.4	7-fixed	83.6
Mushroom	95.8	98.1	98.1	**100**	**100**	**100**	99.9	-	-	**100**	99.9	3-multi	**100**
Pima	78.1	78.1	78.1	75.9	**78.6**	78.3	78.0	75.8	75.6	75.7	77.5	10-fixed	78.4
Satimage	81.8	82.5	82.5	88.6	88.4	**89.4**	86.6	83.9	83.8	87.4	87.0	1.5-fixed	87.0
Segment	91.3	92.6	92.6	96.9	96.1	95.9	94.0	94.2	93.3	95.2	**97.0**	1.5-fixed	**97.0**
Sick	86.1	97.0	97.1	97.5	97.3	97.4	97.6	-	-	97.3	**97.7**	2-fixed	**98.1**
Sonar	84.1	84.6	85.1	85.1	83.2	**86.5**	85.1	-	-	78.4	84.9	10-fixed	85.6
Soybean l.	89.1	91.5	91.5	91.9	93.2	88.6	88.3	-	-	-	**95.2**	1-fixed	**95.3**
Soybean s.	98.0	**100**	**100**	**100**	**100**	**100**	**100**	-	-	-	**100**	1.5-fixed	**100**
Splice	95.5	95.7	95.7	96.0	95.0	95.8	95.5	94.6	94.1	94.1	**96.3**	2-fixed	**97.2**
Vehicle	62.9	62.8	62.8	71.7	73.0	70.4	71.3	68.8	71.9	68.1	72.9	1.5-fixed	**73.6**
Voting	90.1	89.9	90.1	94.5	94.9	94.7	94.3	94.7	93.3	89.8	94.5	7-fixed	**95.3**
Waveform	81.8	81.8	81.8	**86.2**	82.5	86.0	84.2	79.4	79.1	82.7	83.5	2-fixed	83.9
Wine	96.9	98.9	98.9	98.3	98.9	98.3	98.9	-	-	97.5	**99.5**	1.5-fixed	**99.5**
Yeast	58.9	58.8	58.8	59.4	58.8	59.0	59.2	58.2	58.2	58.2	59.3	1.5-multi	**60.0**
Zoo	89.1	97.0	97.0	**98.0**	96.0	96.0	96.0	-	-	94.7	97.7	1.5-fixed	97.7
Average	83.87	85.67	85.92	87.50	87.29	87.02	86.60	-	-	-	**87.88**		**88.27**
Avg.LB	-	-	-	-	-	-	-	81.95	-	-	**84.96**		**85.54**
Avg.LBChi^2	-	-	-	-	-	-	-	-	82.90	-	**85.40**		**85.98**
Avg.BCEP	-	-	-	-	-	-	-	-	-	84.88	**87.56**		**88.08**

TABLE III

ACCURACY COMPARISON WITH STATE-OF-THE-ART ASSOCIATIVE CLASSIFIERS

Dataset	iCAEP	CPAR	CMAR	Harmony	DeEPs		L^3			EnBay	
	minrate=5	δ=0.05	δ=4	Tuned	α=0.12	Dyn.	Standard	Tuned		Standard	Tuned
	minimp=0.01	mingain=0.7	conf. dif. 20%					sup %-rule type			
Anneal	95.1	98.4	97.3	95.6	94.4	95.0	97.9	1-g	98.1	**98.9**	**98.9**
Audiology	-	-	-	-	-	-	68.3	1-g	68.3	**74.2**	**74.2**
Australia	86.1	**86.2**	86.1	-	84.8	**88.4**	85.2	18-g	87.0	85.2	87.4
Auto	-	82.0	78.1	61.5	67.7	72.7	81.0	1-g	82.4	**85.7**	**85.7**
Balance-scale	-	-	-	-	-	-	69.6	1-g	69.6	**72.1**	**72.2**
Breast-w	**97.4**	96.0	96.4	96.1	96.4	96.4	95.7	4-s	96.4	97.2	**97.4**
Cleve	-	81.5	82.2	-	81.2	84.2	80.5	21-g	**85.5**	**82.5**	82.6
Diabetes	-	75.1	74.5	-	76.8	76.8	**78.0**	1-s	**78.7**	77.5	78.1
Flare	-	-	-	-	83.5	83.5	98.3	1-s	99.1	**99.4**	**99.4**
German	73.1	73.4	74.9	-	74.4	74.4	73.0	4-s	74.3	**75.0**	**75.7**
Glass	-	74.4	70.1	49.8	58.5	67.4	76.2	9-s	**78.0**	**78.0**	**78.0**
Heart	80.3	82.6	82.2	58.4	81.1	82.2	81.5	1-s	**84.4**	**83.1**	84.2
Hepatitis	83.3	79.4	80.5	86.0	81.2	82.5	83.9	1-s	85.8	**86.5**	**87.4**
Horse	-	**84.2**	82.6	84.6	**84.2**	**85.3**	81.5	1-s	81.5	80.0	80.0
Hypo	96.4	98.1	98.4	-	98.4	98.3	**98.9**	1-s	**98.9**	97.9	97.9
Ionosphere	90.6	92.6	91.5	**93.5**	86.2	91.2	92.0	45-g	**93.5**	**93.0**	93.0
Iris	93.3	94.7	94.0	96.0	**96.0**	**96.7**	92.7	1-s	93.3	95.2	95.3
Labor	89.7	84.7	89.7	-	87.7	87.7	91.2	1-g	94.7	**96.5**	**98.4**
Letter rec.	-	-	-	76.8	**93.6**	**93.6**	84.0	1-s	84.0	91.5	91.5
Lymph	79.8	82.3	83.1	-	75.4	75.4	78.4	15-g	**86.5**	**83.4**	83.6
Mushroom	-	-	-	99.9	**100**	**100**	**100**	1-s	**100**	99.9	**100.0**
Pima	72.3	73.8	75.1	73.8	76.8	77.1	**78.3**	4-s	**78.5**	77.5	78.4
Satimage	-	-	-	-	**88.5**	**88.5**	85.5 ($L=5$)	1-s	85.5 ($L=5$)	87.0	87.0
Segment	-	-	-	-	95.0	96.0	94.3	1-s	94.3	**97.0**	**97.0**
Sick	-	96.8	97.5	-	94.0	96.6	95.5	1-g	95.6	**97.7**	**98.1**
Sonar	-	79.3	79.4	-	84.2	**87.0**	79.3	45-g	81.7	**84.9**	85.6
Soybean l.	-	-	-	-	90.1	90.1	92.6 ($L=4$)	1-s	92.6 ($L=4$)	**95.2**	**95.3**
Soybean s.	-	-	-	-	**100**	**100**	97.9	1-s	97.9	**100.0**	**100.0**
Splice	-	-	-	-	69.7	69.7	93.5	1-s	93.5	**96.3**	**97.2**
Vehicle	62.8	69.5	68.8	-	71.0	**74.6**	71.4	5-s	73.4	**72.9**	73.6
Voting	-	-	-	-	**95.2**	95.2	94.7	40-s	95.2	94.5	**95.3**
Waveform	81.7	80.9	83.2	80.5	**84.4**	**84.4**	82.1	2-s	82.5	83.5	83.9
Wine	98.9	95.5	95.0	94.9	95.6	96.1	97.8	1-g	98.3	**99.5**	**99.5**
Yeast	-	-	-	-	58.9	**60.2**	59.0	1-s	59.0	**59.3**	60.0
Zoo	-	95.1	97.1	96.0	97.2	97.2	91.1	1-g	94.1	**97.7**	**97.7**
Average	-	-	-	-	-	-	85.74		86.92	**87.88**	**88.27**
Average DeEPs	-	-	-	-	84.91	86.19	-		-	**88.77**	**89.18**
Average Harmony	-	-	-	82.89	-	-	-		-	**89.81**	**90.06**
Average CMAR	-	85.07	85.12	-	-	-	-		-	**87.36**	**87.84**
Average iCAEP	85.39	-	-	-	-	-	-		-	**87.03**	**87.58**

implementation is publicly available (e.g., CMAR, BCEP), we showed the accuracy on each dataset reported in the referenced paper and, for each classifier, we also separately averaged the results on its dataset subset.

We initially performed a comparison with the state-of-the-art Bayesian classifiers. Then, since our approach is pattern-based, we also compared it with well-known associative classifiers. Finally, for the sake of completeness, we performed a comparison with a selection of other well-known classifiers.

Comparison with Bayesian classifiers. In Table II, we compared the accuracy of the EnBay algorithm with that of the main state-of-the-art Bayesian classifiers and some of the most popular Bayesian classifiers, i.e., Hidden Naive Bayes (HNB) [27], Full Bayesian Network Classifier (FBN) [40], Aggregating One-Dependence Estimators (AODE) [42], Bayes Network (BayesNet) [16] Lazy Bayesian Rules Classifier (LBR) [46], Large Bayes (LB) [39], Large Bayes χ^2 (LBχ^2) [38], BCEP [13], and Naive Bayes [32].

HNB, FBN, Bayes Network, AODE, and BayesNet are Bayesian Network classifiers that aim at overcoming the Naive independence assumption by generating and evaluating conditional-dependence models. LBR is an hybrid approach that combines Naive Bayes and rule-based classification. Finally, LB, LBχ^2, and BCEP are pattern-based Bayesian approaches that exploit either frequent itemsets or emerging patterns to also consider high order item correlations in the probability estimates.

On average, EnBay performs better than all the other Bayesian approaches, both the traditional (e.g., standard about +4.01% against Naive, and tuned about +4.40%) and the most recent ones (e.g., standard about +0.38% against HNB, +0.59% against FBN, and +2.68% against BCEP, while tuned about +0.77% against HNB, +0.98% against FBN, and +3.20% against BCEP).

Learning the most reliable attribute (in)dependencies, together with the use of a pattern-based approximation, turns out to be more effective than traditional Bayesian Network (e.g., HNB, FBN) and pattern-based Bayesian (e.g., LB, BCEP) approaches in overcoming the naive independence assumption. The improvement with respect to previous pattern-based Bayesian classifiers is particularly significant (standard EnBay about +3.0% against LB, +2.5% against LBχ^2, and +2.7% against BCEP). The use of a product approximation tailored to each class, which may capture different facets of the analyzed data, and the high effectiveness of the proposed approximation in covering the given test cases are the main reasons of EnBay performance.

Comparison with associative classifiers. Since EnBay is pattern-based, in Table III we also compared it with the well-known associative classifiers L^3 [4], DeEPs [34], CMAR [35], Harmony [41], CPAR [44], and iCAEP [45]. EnBay performs better than all the other approaches. Based on experimental results on common datasets, tuned EnBay performs significantly better than the instance-centric rule-based (tuned) Harmony classifier (+7.4%). The standard EnBay outperforms both the standard and the tuned versions of DeEPs (+2.3% and +0.95% respectively). Both standard and tuned EnBay perform better than the corresponding L^3 configuration settings (+0.7% against standard and +0.5% against tuned). The way in which EnBay exploits frequent patterns in class prediction differs significantly from the approach adopted in associative classifiers. More specifically, the majority of the traditional associative classifiers exclusively consider a very limited number of high-confidence rules covering the test case (e.g., around 5 rules in the case of CMAR). Since the selected classification rule coverage is commonly limited

to a portion of the given test case, some of the possibly relevant facets of the analyzed data are not considered. Differently, the product approximation adopted in EnBay yields a complete and reliable test data coverage.

Comparison with other well-known classifiers. Table IV reports the comparison with a selection of other, neither Bayesian nor associative, state-of-the-art classifiers. More specifically, EnBay is compared with rule-based (Ripper [9]), K-Nearest Neighbor (K-NN [10]), decision tree (C5.0 [23]), and Support Vector Machines (SVMs [28]) classifiers. For lack of space, we reported only the results achieved by the tuned versions of the EnBay competitors. EnBay performs significantly better than most of the other classifiers (e.g., standard +4.0% against standard K-NN, tuned about +2.2% against tuned K-NN) and is competitive with SVMs (standard +3.6% against standard SVMs, tuned +0.8% against tuned SVMs), whose performance tuning may be hardly manageable by not expert users.

B. Accuracy performance validation

To validate the statistical significance of EnBay accuracy improvement, we used the 10-fold cross-validated paired t-test and the Wilcoxon signed rank test [11]. Both tests were applied at significance level p = 0.05 on all the evaluated datasets. EnBay is compared with all the Bayesian classifiers considered in the previous section (Naive Bayes, Bayes Network, HNB, FBN, AODE, LBR), a state-of-the-art associative classifier (L^3 [4]) a decision tree (C5.0), a rule-based classifier (Ripper), K-NN, and SVMs. For the algorithms whose implementation is not publicly available we could not perform the t-test. For each comparison, Table V reports the number of datasets on which EnBay performs statistically better/worse than the classifier reported in the corresponding column. Comparisons between both standard and tuned classifier versions were performed. When comparing EnBay with parameter-free algorithms, standard and tuned versions of the classifiers are coincident.

EnBay performs better than all the other considered classifiers in both tests. When comparing standard EnBay and Naive Bayes, the paired t-test considers statistically significant the accuracy improvements achieved on 9 datasets in favor of EnBay against 1 in favor of Naive Bayes. On the subset of relevant datasets, EnBay achieves an average accuracy improvement with respect to Naive of +9.1%. The above results highlight the effectiveness of EnBay in overcoming the Naive independence assumption. Similar considerations hold for the other comparisons reported in Table V. EnBay is statistically better than HNB in 8 cases and worse in 5. When the tuned version of EnBay is used, the gap further increases: 10 against 4. For the algorithms for which we could not perform the t-test, we provide in Table VI a summarized comparison based on Tables II, III, and IV. For each comparison, Win/Draw/Loss are the number of datasets where EnBay accuracy is higher/equal/lower than the compared classifier.

C. Tuning EnBay performance

The results reported in Tables II-IV show that, when using the standard configuration, EnBay yields good accuracy results. Its performance may be further improved by (a) tuning the support threshold and (b) enforcing different support thresholds for every class. In this case, a classifier tuning session is performed, in which the best (tuned) configuration is experimentally selected by generating several classifiers with different configurations. In particular,

TABLE IV

ACCURACY COMPARISON WITH WELL-KNOWN CLASSIFIERS

Dataset	K-NN	SVMs	C5.0	Ripper	EnBay	
	Tuned	Tuned	Tuned	Tuned	Standard	Tuned
Anneal	94.7	87.0	95.1	95.8	**98.9**	**98.9**
Audiology	73.9	78.8	**81.4**	76.1	74.2	74.2
Australia	85.7	85.7	85.1	87.0	85.2	**87.4**
Auto	73.2	59.5	85.4	76.1	**85.7**	**85.7**
Balance-scale	90.2	**100.0**	79.8	81.6	72.1	72.2
Breast-w	96.9	96.7	95.7	96.9	97.2	**97.4**
Cleve	82.5	**82.8**	78.9	81.8	82.5	82.6
Diabetes	75.1	77.3	75.0	76.6	77.5	**78.1**
Flare	**99.5**	**99.5**	**99.5**	**99.5**	99.4	99.4
German	74.6	73.3	72.5	74.7	75.0	**75.7**
Glass	70.6	73.8	70.6	72.0	**78.0**	**78.0**
Heart	81.1	84.1	78.9	82.6	83.1	**84.2**
Hepatitis	85.8	**87.7**	83.2	80.6	86.5	87.4
Horse	55.2	68.5	71.7	**87.8**	80.0	80.0
Hypo	97.2	98.3	**99.3**	99.2	97.9	97.9
Ionosphere	85.8	**95.4**	90.6	91.2	93.0	93.0
Iris	96.7	**98.0**	96.0	96.0	95.2	95.3
Labor	93.0	96.5	84.2	93.0	96.5	**98.4**
Letter rec.	95.9	**98.0**	88.6	86.4	91.5	91.5
Lymph	83.8	**85.8**	79.1	81.8	83.4	83.6
Mushroom	**100**	**100.0**	**100**	**100.0**	99.9	**100.0**
Pima	75.1	77.3	75.0	76.6	77.5	**78.4**
Satimage	**90.9**	86.6	87.1	87.0	87.0	87.0
Segment	97.0	96.4	**97.3**	96.1	97.0	97.0
Sick	95.3	96.8	**98.8**	98.4	97.7	98.1
Sonar	86.1	**89.9**	74.0	83.2	84.9	85.6
Soybean l.	89.6	89.6	86.3	85.0	95.2	**95.3**
Soybean s.	**100**	**100.0**	97.9	97.9	**100.0**	**100.0**
Splice	86	83.7	94.1	94.7	96.3	**97.2**
Vehicle	72.1	**80.9**	72.9	70.6	72.9	73.6
Voting	93.3	**97.0**	**97.0**	95.9	94.5	95.3
Waveform	83.3	**87.2**	76.9	80.4	83.5	83.9
Wine	97.2	96.6	93.3	94.9	**99.5**	**99.5**
Yeast	58.3	55.1	59.6	59.6	59.3	**60.0**
Zoo	97.0	**98.0**	96.0	90.1	97.7	97.7
Average	86.06	87.48	85.62	86.48	87.88	**88.27**

TABLE V

STATISTICAL TESTS OF SIGNIFICANCE. X/Y INDICATES THAT ENBAY STANDARD IS SIGNIFICANTLY BETTER X TIMES (WORSE Y TIMES) THAN THE CLASSIFIER IN COLUMN

	paired t-test, p=0.05										
	Naive	BayesNet	HNB	FBN	AODE	LBR	L^3	K-NN	SVMs	C5.0	Ripper
Standard EnBay	12/2	10/1	8/5	8/5	9/6	7/3	9/2	11/4	8/5	11/4	10/3
Tuned EnBay	13/1	10/0	10/4	9/3	10/5	10/2	12/3	9/3	7/5	12/1	12/2
	Wilcoxon signed rank test, p=0.05										
	Naive	BayesNet	HNB	FBN	AODE	LBR	L^3	K-NN	SVMs	C5.0	Ripper
Standard EnBay	9/1	8/1	5/3	8/4	5/2	5/2	6/0	9/3	6/4	10/1	8/1
Tuned EnBay	9/0	7/0	7/2	8/3	6/2	7/2	9/2	7/2	5/3	11/1	8/0

TABLE VI

ENBAY VS THE OTHER CLASSIFIERS. WIN/DRAW/LOSS ARE THE DATASETS WHERE, BASED ON TABLES II, III, AND IV, ENBAY HAS HIGHER/EQUAL/LOWER ACCURACY THAN THE CLASSIFIER IN COLUMN

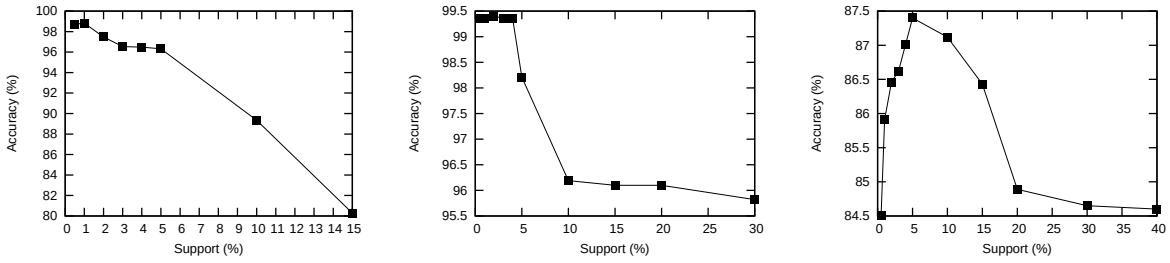
Win/Draw/Loss	LB	LBχ^2	BCEP	iCAEP	CPAR	CMAR	Harmony	DeEPs	
							Tuned	$\alpha = 0.12$	Dyn.
Standard EnBay	15/0/3	13/1/3	21/0/5	13/1/2	20/1/3	20/1/3	11/2/3	24/2/8	19/2/13
Tuned EnBay	17/0/1	15/1/1	23/1/2	14/2/0	21/1/2	21/1/2	12/1/3	25/3/6	20/3/11

for each dataset, the best overall (i.e., tuned) configuration is selected by (i) enabling/disabling multiple support thresholds and (ii) tuning the support threshold to its best value by varying it in the range [0.5%, 25%].

As shown in Tables II-IV, tuning may provide a significant accuracy improvement. Furthermore, since the Materialization Trees exploited for the probability estimations may be generated once with min_sup=0 for the entire tuning session, tuning is efficient and has a low impact on the final classifier training time. In the following, the effect of each EnBay parameter on the classification accuracy is discussed separately.

Tuning the minimum support threshold. The effect of minimum support threshold on the classification accuracy has been studied by enforcing a wide range of different support values on all the evaluated datasets. Figures 1(a)-1(c) show the accuracy variation with respect to the minimum support threshold for three different datasets, taken as representatives of the different behaviors shown by our experiments on all datasets. For most of the considered datasets the highest accuracy values were achieved when low support threshold values (i.e., around 1.5%) were enforced, while for a few others (e.g., see Figure 1(c)) the best value is obtained by enforcing higher support threshold values (from 5% to 10%). When minimum support thresholds significantly lower than 1.5% have been enforced, most of the datasets do not show any significant accuracy improvement. In a few cases, they may even yield an accuracy reduction (Figure 1(c)) in particular for the datasets characterized by better accuracy results for high minimum support threshold values. This is likely due to training data overfitting.

Effect of multiple support thresholds. To evaluate this effect, EnBay has been run on all datasets with (a) a



(a) Anneal. (b) Flare. (c) Hepatitis.

Fig. 1. Accuracy of EnBay when varying the minimum support threshold.

single support threshold (denoted as "fixed" in Tables II-IV), and (b) multiple support thresholds tailored to each class (denoted as "multi" in Tables II-IV). Both the fixed and the global (for multiple support thresholds) minimum support thresholds have been varied on a significant value range for all considered datasets to analyze their effect on the classification accuracy. The best setting depends on the data distribution. More specifically, the positive impact of the multiple support thresholds is particularly evident when classes are unevenly distributed on training data, while, as expected, balanced datasets do not show any relevant accuracy improvement. However, multiple support thresholds should be carefully adopted in presence of rare and very small classes, as the enforcement of multiple support thresholds may lead to data overfitting. In some cases, fixed minimum support threshold may provide better performance results even in presence of unevenly distributed training data.

D. Classifier characteristics

EnBay does not create a classification model. To improve the efficiency of the on-the-fly prediction process, during the training phase EnBay generates a set of disk-resident Materialization Trees, one for each training class. The disk-resident Materialization Trees are generated by enforcing a minimum support threshold equal to zero for every class, i.e., a complete representation of the entire training dataset is generated. Materialization Trees are then visited instead of the source dataset to evaluate the Bayesian probabilities.

We describe the characteristics of the EnBay classifier, in terms of (a) Materialization Tree size, (b) Materialization Tree generation time, and (c) classification time. The experimental results reported in Table VII refer to the standard EnBay configuration.

Materialization Tree size. Since Materialization Trees are created without enforcing any minimum support threshold, the average disk occupancy of the Materialization Trees is a relevant issue. To evaluate the compactness of the Materialization Tree structure with respect to the source dataset, we evaluated its compression factor, defined as follows: CF $=(1 - \frac{\text{MAT\_size}}{\text{Data\_size}})\%$. Table VII reports in column (5) the average per-class compression factor values provided by the Materialization Trees on each dataset. The compression factor strongly depends on the data distribution. It ranges from a few per cents on very sparse datasets (e.g., 4.6% for mushroom) to even more than

80% on denser datasets (e.g., 84.1% for Satimage). The average compression factor on all the evaluated datasets is roughly 47.4%. Hence, the adopted data representation is suitable for efficiently storing possibly large amounts of data characterized by diverse data distributions.

Materialization Tree generation time. Column (4) of Table VII reports the time spent by EnBay to generate all the Materialization Trees. On most UCI datasets (19) index generation takes from 0.1 to 0.5 seconds, while a few datasets (7 datasets) take from 0.5 to 2 seconds. On 9 small datasets (e.g., labor, heart) the Materialization Tree generation step requires even less than 0.1 seconds. The EnBay Materialization Tree generation time is affected by both (i) the dataset size (column (2) of Table VII) and (ii) the number of classes (column (3)). However, the effect of the dataset cardinality is dominant.

Classification time. In Table VII, Column (6) reports the average classification time per test case. The time spent in disk-based tree visits for longest frequent pattern extraction and evaluation significantly affects the prediction time. The classification time scales roughly linearly with the number of records. The per-class Bayesian estimate increases the EnBay computational load when the dataset is characterized by a large number of classes (e.g., Letter rec.). The average classification time is still acceptable, between 20 and 80 ms for the majority of the evaluated datasets (19 datasets). Other approaches, which require a more limited number of disk accesses (e.g., L^3 [4], SVMs [28]), take on average a few milliseconds. However, as shown in Section V-E, they show strong limitations in coping with larger datasets and they are less accurate than EnBay.

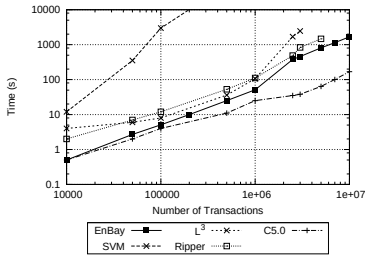
E. Scalability

We analyzed the scalability of the standard EnBay configuration by varying (a) the number of dataset records and (b) the number of attributes on synthetic datasets. The datasets were generated by means of the IBM data generator [22] with classification function 2. For each dataset, two class labels are available and each record is labeled by applying the classification function on the first nine attributes of the dataset. The scalability analysis has been performed on the training and test phases separately.

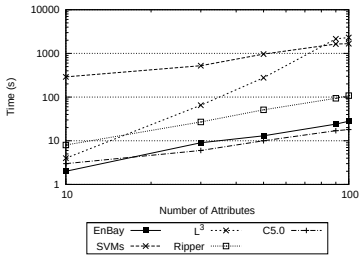
To perform a scalability comparison, we selected the classifiers for which an efficient implementation was available. In particular, we considered the L^3 [4], Ripper [9], C5.0 [23], and SVMs [28] classifiers as representative of different classification approaches (associative, rules-based, decision trees, and SVMs). We analyzed scalability with the standard configuration settings.

Training time. Since, as already discussed, EnBay does not generate any classification model, we compared the training time of the other classification algorithms with the time spent by EnBay in the preprocessing step required for per-class Materialization Trees creation. The Materialization Trees are generated without enforcing any support threshold.

Dataset cardinality. Figure 2(a) shows the training time by varying the number of transactions from 10,000 to 10,000,000 on datasets with the first nine attributes. Most algorithms show a good scalability with the number of dataset records. Almost all the tested algorithms scale approximatively linearly with the number of records. All the approaches are significantly faster than SVMs, whose complexity scales superlinearly with the number of



(a) Training time time when varying the number of transactions (number of attributes=9).



(b) Training time when varying the number of attributes (number of transactions=$50,000$).

Fig. 2. EnBay Training Time Scalability.

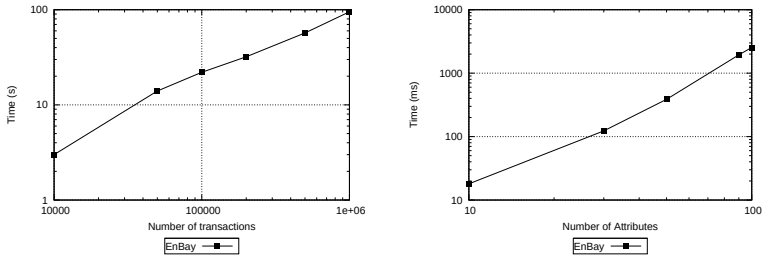
transactions. EnBay is slower than C5.0, while it is faster than Ripper and L^3. Except for EnBay and C5.0, all the other approaches are not able to cope with datasets characterized by more than 5,000,000 records. By exploiting the disk-based tree materializations EnBay efficiently manages large structured training sets not manageable by most of the alternative approaches. Furthermore, it significantly outperforms, in terms of classification accuracy, the other scalable approaches, e.g., C5.0 (see Section V-A).

Number of attributes. Figure 2(b) plots the training time when varying the number of attributes from 10 to 100 on datasets with 50,000 records. EnBay scales well with the number of attributes. Unlike EnBay, the L^3 [4] associative classifier does not show a good scalability with the number of attributes, due to the non-linear dependence of the number of frequent itemsets with the number of attributes. The Materialization Tree data representation, instead, allows efficiently handling datasets characterized by a large number of attributes. All the considered approaches, except L^3, scale almost linearly with respect to the number of attributes.

Classification time. Since our classifier is based on a lazy approach, its per-data classification time is affected by the cardinality of the training dataset. Hence, we also analyzed its scalability with respect to the characteristics of the training dataset in terms of per-data classification time. Since all the algorithms tested in the scalability experiments are eager classifiers, their prediction time is only marginally affected by the dataset cardinality. Hence, their corresponding plots were omitted.

Dataset cardinality. Figure 3(a) shows the average per-data classification time when varying the number of transactions from 10,000 to 1,000,000 on the IBM datasets characterized by the first nine attributes. The prediction time of EnBay scales well with the number of transactions. In particular, it depends on the size of the Materialization Tree.

Number of attributes. Figure 3(b) plots the average per-data prediction time when varying the number of attributes from 10 to 100 on datasets with 50,000 transactions. EnBay scales more than linearly with the number of attributes due to the non-linear dependence of the number of frequent itemsets with the number of record attributes. The



(a) Average test data prediction time when varying the number of transactions (number of attributes=9).

(b) Average test data prediction time when varying the number of attributes (number of transactions=$50,000$).

Fig. 3. EnBay Classification Time Scalability.

classification time scales more than linearly with the number of attributes. However, its per-data classification time is still acceptable even when the number of attributes is high (e.g., around 2 s with 90 attributes and $50,000$ records).

VI. Related Work

Several research efforts have been devoted to improving Naive Bayes [32] classification performance. In order to relax the naive independence assumption, the mostly used approach is to represent attribute dependencies in graph-based models, called Bayesian Networks, in which nodes represent attributes, while oriented arcs are weighted by conditional probabilities for each node given its parent. Since Bayesian Network learning is NP-hard [7] many approaches propose to impose model restrictions to make the learning problem tractable. For instance, the Tree Augmented Naive-Bayes (TAN), adopted in [8], [12], [16], allows at most one parent node for each attribute node (in addition to the class node). Thus, the influence of the other attributes is ignored. Subsequent approaches (e.g., [24], [27], [40], [42]) introduced significant improvements. For instance, in [42] an ensemble of TANs is learned, each one rooted in a different attribute. Then, the class label is predicted by aggregating the classifications of all qualified TANs. In [40] a Full Bayesian Network structure is learnt first, while assuming that all the attributes are dependent. Next, decision trees are built on top of it. Differently, in [24] a Markov network model is used to represent attribute dependencies, which are evaluated, similarly to [17], by maximizing the conditional log-likelihood objective function. However, accomplishing the maximization task may become computationally intensive. To avoid the complexity of the structure learning process, many approaches focus on inheriting the structural simplicity of the Naive Bayes classifier [15], [18], [19], [27], [30], [46]. For instance, authors in [19], [30], [46] propose hybrid approaches that combine Naive Bayes and decision trees classifiers. While in [30], [46] Naive Bayes classification is applied at the level of the decision tree leaves and performed on a subset of training data, in [19] the dataset attributes are split into two groups: one group assigns class probabilities based on Naive Bayes, the other based on a decision table. Next, the resulting probability estimates are combined. Despite their simplicity, the above-mentioned classifiers still show some limitations in coping with highly correlated data. In [15], [18] attributes are weighted

differently according to their contribution to classification. A similar approach is adopted in Hidden Naive Bayes (HNB) [27], which is, to the best of our knowledge, the latest and most effective Bayesian Network classifier. In [27], the authors proposed to create, for each attribute, a hidden parent that represents the influence of all the others. The influence is computed, similarly to [40], as a linear combination of the conditional mutual information between attribute couples. Thus, higher order item correlations are neglected.

Differently from all the previously discussed research works, this paper addresses the usage of frequent patterns in Bayesian classification. The idea behind pattern-based Bayesian classification is to exploit frequent patterns, mined by means of traditional algorithms (e.g., Apriori [1]), to estimate reliable Bayesian probabilities. The Large Bayes classifier [39] made the first attempt to relax the strong independence assumption by using long and frequent itemsets to estimate probabilities through a product form approximation [33]. Patterns are first discovered and selected based on an interestingness measure, derived from cross-entropy evaluators [29], to construct a reliable probability approximation tailored to a given test case. Next, the product approximation is evaluated against each class. The most likely class labels the test case. In [38] the same authors proposed to use the chi-square test in the Large Bayes classifier to better capture interesting patterns to involve in the product approximation. The work in [13] extended [38], [39] by proposing the use of emerging patterns [34] in the product approximation to discover more effectively multi-attribute dependencies among classes. However, all the previous pattern-based Bayesian approaches build a unique product approximation for every test case. Thus, the estimates are only weakly bound to the considered classes. Furthermore, the need of extracting a large number of long and overlapped frequent itemsets causes both the quality of the approximations to be sensitive to the support threshold variations, and the classification algorithms to be unable to cope with large datasets.

Unlike [13], [38], [39], EnBay adopts a novel, simple, yet effective approach that combines the generation of conditional-independence discriminative models with the construction of per-class reliable probability approximations.

VII. Conclusions and future work

This paper presents EnBay, a pattern-based Bayesian classifier that exploits frequent itemsets to approximate the joint Bayesian probability. EnBay exploits a novel and more effective probability approximation estimate complying with a conditional-independence model. Long, frequent, and disjoint itemsets to be included in the class-based approximations are selected, by means of an entropy-based heuristics, so that the corresponding attribute sets are conditionally mutually independent given the class under evaluation. To successfully cope with large datasets, the use of efficient disk-based index structures allows the retrieval and on-the-fly evaluation of patterns without the need of temporarily storing patterns.

Experiments performed on real and synthetic datasets show the effectiveness and the efficiency of the proposed approach. EnBay is significantly more accurate than most state-of-the-art classifiers, Bayesian and not, and achieves good scalability results.

Future developments of this work will address the domain of noisy data, for which the reliability of the probability

estimate becomes particularly relevant, and the integration of generalized itemset mining algorithms (e.g., [2]) to further enhance classification accuracy.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, pages 207–216, 1993.

[2] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, and P. Garza. Support driven opportunistic aggregation for generalized itemset extraction. In *5th International Conference on Intelligent Systems*. IEEE, 2010.

[3] E. Baralis, T. Cerquitelli, and S. Chiusano. A persistent hy-tree to efficiently support itemset mining on large datasets. In *ACM Symposium on Applied Computing 2010*, 2010.

[4] E. Baralis, S. Chiusano, and P. Garza. A lazy approach to associative classification. *IEEE TKDE*, 20(2):156–171, 2008. Software downloaded from http://dbdmg.polito.it/twiki/bin/view/Public/L3.

[5] C. Blake and C. Merz. Uci repository of machine learning databases. Available at http://archive.ics.uci.edu/ml.

[6] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno-Perez. Dynamic lexicographic approach for heuristic multi-objective optimization. In *Proceedings of the Workshop on Intelligent Metaheuristics for Logistic Planning (CAEPIA-TTIA 2009)*, pages 153–163. Springer, Springer, 2010.

[7] D. Chickering. Learning bayesian networks is NP-Complete. *Learning from data: Artificial Intelligence and Statistics*, pages 121 – 130, 1996.

[8] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968.

[9] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995. Software downloaded from http://www.cs.cmu.edu/∼wcohen.

[10] P. Cover and P. Hart. Fast effective rule induction. *IEEE Transaction on INformation Theory*, 13, 1967.

[11] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7), 1998.

[12] P. Domingos and M. J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[13] H. Fan and K. Ramamohanarao. A bayesian approach to use emerging patterns for classification. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 39–48, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[14] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

[15] J. T. A. S. Ferreira, D. G. T. Denison, and D. J. Hand. Weighted naive bayes modelling for data mining, 2001.

[16] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[17] D. Grossman and P. Domingos. Learning bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 46–, New York, NY, USA, 2004. ACM.

[18] M. Hall. A decision tree-based attribute weighting filter for naive bayes. In M. Bramer, F. Coenen, and A. Tuson, editors, *Research and Development in Intelligent Systems XXIII*, pages 59–70. Springer London, 2007.

[19] M. Hall and E. Frank. Combining naive bayes and decision tables. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, pages 318–319. AAAI Press, 2008.

[20] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *In SIGMOD'00, Dallas, TX*, May 2000.

[21] T. Havrnek. Statistics of multidimensional contingency tables and the guha method. *International Journal of Man-Machine Studies*, 10(1):87 – 93, 1978.

[22] IBM. IBM Quest Synthetic Data Generation Code. http://www.almaden.ibm.com/.

[23] J. Quinlan. *C4.5: Programs for Machine Learning*. The Morgan Kaufmann, 1993. Software downloaded from http://www.rulequest.com.

[24] A. Jakulin and I. Rish. Bayesian learning of markov network structure. In *ECML'06: 17th European Conference on Machine Learning*, pages 198–209, 2006.

[25] E. T. Jaynes. Information theory and statistical mechanics. ii. *Phys. Rev.*, 108(2):171–190, Oct 1957.

[26] L. Jiang, D. Wang, and Z. Cai. Scaling up the accuracy of bayesian network classifiers by m-estimate. In *ICIC (2)*, pages 475–484, 2007.

[27] L. Jiang, H. Zhang, and Z. Cai. A novel bayes model: Hidden naive bayes. *IEEE TKDE*, 21(10):1361–1371, 2009.

[28] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *ECML '98*, 1998.

[29] R. Johnson and J. Shore. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26(1):26–37, 1980.

[30] R. Kohavi. Scaling up the accuracy of naive bayes classifiers: a decision-tree hybrid. In *KDD '96*, pages 740–743, 1996.

[31] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. Mlc++: a machine learning library in c++. In *IN TOOLS WITH ARTIFICIAL INTELLIGENCE*, pages 740–743. IEEE Computer Society Press, 1994.

[32] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML*, pages 4–15, 1998.

[33] P. Lewis. Approximating discrete probability distributions to reduce storage requirements. *Information and Control*, 2:214–225, 1959.

[34] J. Li, G. Dong, K. Ramamohanaro, and L. Wong. Deeps: A new instance-based discovery and classification system, 2001.

[35] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *ICDM*, pages 369–376, 2001.

[36] B. Liu, Y. Ma, and C. K. Wong. Improving an association rule based classifier. In *Principles of Data Mining and Knowledge Discovery*, pages 504–509. Springer-Verlag, 2000.

[37] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, Apr. 2004.

[38] D. Meretakis, H. Lu, and B. Wüthrich. A study on the performance of large bayes classifier. In *ECML*, pages 271–279, 2000.

[39] D. Meretakis and B. Wüthrich. Extending naïve bayes classifiers using long itemsets. In *KDD'99*, pages 165–174, 1999.

[40] J. Su and H. Zhang. Full bayesian network classifiers. In *ICML'06: Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 897–904, 2006. Software downloaded from http://www.cs.unb.ca/profs/hzhang/FBC.rar.

[41] J. Wang and G. Karypis. On mining instance-centric classification rules. *IEEE TKDE*, 18(11):1497–1511, 2006.

[42] G. I. Webb, J. R. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

[43] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005. Software downloaded from http://www.cs.waikato.ac.nz/ml/weka/.

[44] X. Yin and J. Han. CPAR: Classification based on predictive association rules, 2003.

[45] X. Zhang, G. Dong, and K. Ramamohanarao. Information-based classification by aggregating emerging patterns, 2000.

[46] Z. Zheng and G. I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, 2000.

Elena Baralis is full professor at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since January 2005. She holds a Master degree in Electrical Engineering and a Ph.D. in Computer Engineering, both from Politecnico di Torino. Her current research interests are in the field of database systems and data mining, more specifically on mining algorithms for very large databases and sensor/stream data analysis. She has published over 80 papers in international journals and conference proceedings. She has served on the program committees or as area chair of several international conferences and workshops, among which VLDB, IEEE ICDM, ACM SAC, DaWak, ACM CIKM, PKDD.

Luca Cagliero is a post-doc fellow at the Dipartimento di Automatica e Informatica of the Politecnico di Torino since March 2012. He holds a Master degree in Computer and Communication Networks and a PhD in Computer Engineering from Politecnico di Torino. His current research interests are in the fields of Data Mining and Database Systems. In particular, he has worked on structured and unstructured data mining by means of itemset and association rule mining algorithms.

Paolo Garza received the masters and PhD degrees in computer engineering from the Politecnico di Torino. He has been an assistant professor (with non-tenure track position) at the Dipartimento di Elettronica e Informatica, Politecnico di Milano, since June 2010. His current research interests include data mining and database systems. In particular, he has worked on the classification of structured and unstructured data, clustering, and itemset mining algorithms.

TABLE VII

CHARACTERISTICS OF THE ENBAY CLASSIFIER

Dataset	Standard EnBay				
	Num. trans.	Num. classes	Index generation time(s)	CF(%)	Average classification time per record (ms)
Anneal	998	6	0.28	33.7	135.4
Audiology	226	24	0.21	22.9	25.9
Australia	690	2	0.10	11.5	61.1
Auto	205	7	0.15	40.2	74.4
Balance-scale	625	3	0.08	39.2	35.0
Breast-w	699	2	0.10	60.7	26.0
Cleve	303	2	0.06	29.9	33.0
Diabetes	768	2	0.10	74.1	28.2
Flare	1066	2	0.15	33.9	29.1
German	1000	2	0.16	80.7	262.4
Glass	214	7	0.13	59.9	45.9
Heart	270	2	0.04	46.3	25.4
Hepatitis	155	2	0.06	13.4	27.3
Horse	368	2	0.11	51.3	103.8
Hypo	3163	2	0.49	44.9	112.1
Ionosphere	351	2	0.12	76.7	121.7
Iris	150	3	0.06	80.7	23.5
Labor	57	2	0.04	8.3	19.1
Letter rec.	20000	26	1.98	72.5	2,356.3
Lymph	148	4	0.09	7.2	46.9
Mushroom	8124	2	1.03	4.6	221.4
Pima	768	2	0.10	74.1	20.6
Satimage	6435	6	1.48	84.1	1,931.5
Segment	2310	7	0.43	77.1	231.8
Sick	4744	2	0.56	69.4	61.3
Sonar	208	2	0.10	77.0	95.0
Soybean l.	683	19	0.53	17.7	316.0
Soybean s.	47	4	0.09	3.2	34.2
Splice	3175	3	0.96	57.6	1,520.9
Vehicle	846	4	0.20	55.9	77.9
Voting	433	2	0.09	2.9	47.1
Waveform	5000	3	0.77	69.9	641.8
Wine	178	3	0.07	50.1	25.3
Yeast	1484	10	0.32	85.2	77.9
Zoo	101	7	0.15	47.7	58.2