Int. J. Inf. Secur.
DOI 10.1007/s10207-017-0362-4

REGULAR CONTRIBUTION

Using targeted Bayesian network learning for suspect identification in communication networks

A. Gruber[1] · **I. Ben-Gal**[1]

© Springer-Verlag Berlin Heidelberg 2017

Abstract This paper proposes a machine learning application to identify mobile phone users suspected of involvement in criminal activities. The application characterizes the behavioral patterns of suspect users versus non-suspect users based on usage metadata such as call duration, call distribution, interaction time preferences and text-to-call ratios while avoiding any access to the content of calls or messages. The application is based on targeted Bayesian network learning method. It generates a graphical network that can be used by domain experts to gain intuitive insights about the key features that can help identify suspect users. The method enables experts to manage the trade-off between model complexity and accuracy using information theory metrics. Unlike other graphical Bayesian classifiers, the proposed application accomplishes the task required of a security company, namely an accurate suspect identification rate (recall) of at least 50% with no more than a 1% false identification rate. The targeted Bayesian network learning method is also used for additional tasks such as anomaly detection, distinction between "relevant" and "irrelevant" anomalies, and for associating anonymous telephone numbers with existing users by matching behavioral patterns.

Keywords Targeted Bayesian network learning · Suspect identification · Behavioral patterns · Privacy · Security · Machine learning · Cyber crimes · Criminal behavior

✉ I. Ben-Gal
bengal@tau.ac.il

A. Gruber
avivgrub@post.tau.ac.il

[1] Department of Industrial Engineering, Tel Aviv University, Ramat Aviv, 69978 Tel Aviv, Israel

1 Introduction

Mobile devices have become essential tools for many human actions—including, evidently, criminal and terrorist activities. Despite the secrecy surrounding the intelligence field, it is well known (as leaked recently by ex-NSA contractor Edward Snowden) that various intelligence agencies and law enforcement authorities make extensive use of monitoring communication channels as part of their daily operations and consider such monitoring as a valuable source of intelligence [36]. Data mining tools can be used to analyze billions of communication records to filter out noise and extract objects (following previous publications, hereinafter we will use the terms "object," "user" and sometimes "device" interchangeably) that will subsequently be investigated in a more detailed and accurate manner [35]. Such large-scale information monitoring poses great challenges regarding the data collection technology used, privacy issues and the methods used to analyze the collected data. For example, it is essential to present the relevant information (such as users' behavioral patterns) to domain experts who are not necessarily data scientists in a clear and intuitive way (graphically if possible) while maintaining a low computational complexity that allows scalability. The issue of privacy is of vital importance in such large-scale monitoring efforts. Users privacy has garnered wide public concern following the exposure of metadata-monitoring programs operated by US defense agencies such as the NSA as part of their increasing endeavors to fight terrorism after the 9/11 attacks [22,36]. Because we rely on metadata in our analysis, let us clearly state that there is no doubt that metadata contains extremely sensitive information and in many cases does not provide the required level of privacy, as has been shown in many studies (e.g., [13,24]). In fact, De Montjoye et al. [13] showed that a few spatiotemporal points of data are sufficient to uniquely iden-

Published online: 08 February 2017

tify 90% of individuals in a credit card behavioral study. Moreover, they showed that even data sets that convey only coarse information at any or all of the dimensions provide little anonymity. Note, however, that many of these studies addressed settings in which private experimental data are to be shared publicly (e.g., see [7,8]). The setting of this study is somewhat different because it considers a situation in which cellular providers or governmental agencies will use only metadata for the initial screening, although they are often authorized to access granular and private user communication content. Thus, a practical implication of this study is not whether to expose or publish private data but whether to try to provide some hierarchical layers of exposure in which metadata is used as a first alternative to a procedure that fully exposes user content. Moreover, although metadata provides less anonymity than anticipated, in many cases it guarantees a higher level of privacy than full content exposure. For example, Mayer et al. [24] specifically addressed privacy protection against government surveillance by comparing the exposure of the full content of communications to the exposure of only the metadata of those communications. The authors specifically discussed the US National Security Agency, which collects telephone metadata nationwide, and investigated the privacy properties of telephone metadata to assess the impact of monitoring policies that distinguish between content and metadata. Although the authors found that telephone metadata can be used to re-identify users as well as to reveal highly sensitive traits, their numerical studies showed that when using the metadata only a portion (although sometimes a significant one) of the telephone numbers could be re-identified.

In this paper, we present a security data mining application that relies on call detail records (CDR) metadata generated by users' cellular devices ("objects" as they are referred to interchangeably) to identify suspects' activities. The proposed method applies a version of the target-based Bayesian network learning (TBNL) model [18] to accomplish the following analytical tasks: (1) to organize and clean the data and select the relevant features that can be used to detect users' anomalies behavior; (2) to determine whether an anomaly is relevant to the suspect classification task (e.g., an anomaly related to a holiday period shared by many users is less relevant for suspect classification); (3) to associate anonymous devices to existing users based on their behavioral patterns; and finally, (4) to classify suspect users and non-suspect users based on selected explanatory variables (features). The applied method supports the selection of various target features and analyzes these features in connection to the abovementioned analytical tasks. It scores the various features by measuring their information-theoretic gain with respect to the class variable while effectively controlling the trade-off between model complexity and classification accuracy [18]. As a result, domain experts can use the exposed relations among the features to support a more detailed feature engineering process. The TBNL, as a classifier, provides these insights along with a descriptive graphical representation. Unlike some "black-box" classifiers (e.g., neural networks) and even some graphical Bayesian networks (such as the Naïve Bayes and TAN methods), the TBNL method yields a descriptive network in which the selected features and their interactions are used to discriminate between positive (e.g., "suspects") and negative (e.g., "non-suspects") values of the class variable. Moreover, the algorithm generates a different model for each class variable while providing an intuitive interface. Consequently, domain experts can better understand the behavioral patterns of the users through which classification is performed [4,16]. The rest of the paper is organized as follows: Sect. 2 provides some background and a literature review. Section 3 describes the modified TBNL method used in this study. Section 4 details the TBNL method's performance within the considered suspect identification use case, and Sect. 5 concludes the paper.

2 Background

This section provides an overview of some previous studies and applications for suspect detection as well as a brief background on Bayesian network learning and classification that is further elaborated in Sect. 3 with respect to the TBNL.

2.1 Machine learning applications for suspect detection

Many of the cybercrime and homeland security detection applications that use machine learning methods involve both *fraud* and *suspect detection*. Phua et al. [31] provided a comprehensive overview of methods for suspect identification (see also [35]), while other works suggest specific modeling techniques such as neural networks and fuzzy rules for detecting frauds (e.g., see [5,26,27]). Despite the similarity between these two areas, they often differ from each other with respect to the tuned models they use. For example, classification accuracy, which is based on balanced true positive/negative rates, is often not an ideal performance measure in homeland security applications (e.g., suspect identification), mainly due to its sensitivity to false positive and false negative errors [33,36]. Moreover, the datasets for suspect detection are relatively small compared to fraud datasets (e.g., credit card fraud) leading to a need for modeling techniques that can use security experts' existing knowledge to enhance the identification rate. Often, these experts need a convenient interface, possibly a graphical one, to gather insights from the data. Overall, these reasons explain why using fraud detection black-box classifiers may not always be suitable for identifying suspects [24,25,32]. In the homeland security domain, several techniques have been suggested to

reduce the false positive rate (FPR), which occurs, for example, due to the imbalanced sizes between the suspect versus non-suspect classes. Jensen et al. [20] suggested using multiple class values as opposed to binary classification. In [2] and in [1], the authors analyzed the relationships between entities using link analysis tools and text mining to locate well-mapped suspects. In [33], Stolofo et al. suggested a cost-based modeling for fraud and intrusion detection that is also used in other cost-sensitive applications [17]. Other works have suggested merging digital forensics techniques, natural language processing and machine learning principles [23] or, for example, using keywords from the text of the webpages of the suspected users [6]. Many of these techniques indeed lead to lower false positive rates—otherwise the number of possible suspects would be too high. Still, these techniques often involve semantic reasoning and require an analysis of the content rather than only the metadata; therefore, they may encounter privacy constraints and higher computational complexity.

2.2 Bayesian networks and classifiers

A Bayesian network (BN) is a probabilistic graphical model that encodes the joint distribution of a set of random variables. A BN is a directed acyclic graph (DAG) consisting of nodes and edges and a set of parameters that function as conditional probability tables. Each node in the graph represents a random variable (feature) of the domain, and each edge between a pair of nodes represents a probabilistic dependency between the associated random variables [29]. One of the advantages of BN models is that they can serve as an intuitive "explanatory" tool that can depict qualitative relationships in the data while maintaining a rigorous well-defined mathematical model that compactly and efficiently represents the domain [3,19]. A BN can be constructed manually, based on knowledge and hypotheses about the relationships among the domain's variables, or it can be obtained "automatically" by applying learning algorithms to datasets. The learning process is computationally expensive because learning an optimal BN is known to be an NP-hard problem [9]. Most BN learning methods divide the learning process into two stages called structure learning and parameter learning. Structure learning remains an NP-hard problem in the number of variables, whereas parameter learning can be performed in polynomial time for a given structure [14]. In the realm of Bayesian classifiers, there are two main approaches to structure learning. One approach is canonical, and the other is target oriented. The canonical methods follow two main phases: in the first phase, a general BN (GBN) is learned without explicitly considering the classification objective, and in the second phase, the GBN can be used for different analyses, including classification tasks. This two-phase approach can in many cases lead to sub-optimal results [15]. In contrast, the target-oriented methods account for the target (class) variable during the learning phase and construct a specific model around it that characterizes the rest of the variables as attributes to support the classification task. Two of the simplest and most popular target-oriented BN approaches are the well-known Naïve Bayes model [14,28] and the tree-augmented network (TAN) proposed by Friedman et al. [15]. These two cornerstone methods for Bayesian classifiers have been widely implemented in many areas. Often, the target-oriented approach is more effective for classification tasks compared to the canonical approach because the complexity of the targeted model is less affected by those connections that are not directly associated with the target variable [21]. However, these target-oriented methods leave very little room for structure learning because the model complexity is often predefined, and it becomes an inherent property of the method (e.g., in a Naïve Bayes classifier all the variables are automatically linked to the class variable and are assumed to be conditionally independent). Thus, these methods are unable to manage the trade-off between classification accuracy and model complexity. This gap is addressed by the applied TBNL model, which offers both a graphical target-oriented interface and a parametric threshold to control model complexity. With respect to the suspect detection task, the TBNL provides security domain experts with the ability to graphically interpret the relations among the explanatory features and the observed anomaly (as seen in Fig. 3 in the sequel) while controlling the number of features that will be included in the model. In fact, the TBNL can be used either as a classification model or as a preprocessing tool to decide which features should be included in the classification model (see Sect. 4). The TBNL is described in the next section.

3 The targeted Bayesian network learning method

3.1 TBNL modeling

The TBNL method (see appendix) is designed to best approximate the expected conditional probability distribution of the class variable based on the conditioning variables in the domain. First, denote a given target or class variable by $X_t \in \mathbf{X}$. Then, the rest of the domain's variables can be denoted by $\overline{\mathbf{X}}_t = \mathbf{X} \backslash \{X_t\}$ or $\overline{\mathbf{X}}_t$, where $\mathbf{X} = (X_1, \ldots, X_N)$ is a vector of N random variables. The TBNL aims at representing the true and unknown distribution $p(X_t) = \sum_{\overline{\mathbf{x}}_t \in \overline{\mathbf{X}}_t} p(X_t | \overline{\mathbf{x}}_t) p(\overline{\mathbf{x}}_t)$ by a BN based distribution $q(X_t) = \sum_{\overline{\mathbf{x}}_t \in \overline{\mathbf{X}}_t} p(X_t \mid \mathbf{z}_t) p(\overline{\mathbf{x}}_t)$, where $\mathbf{z}_t$ denotes a state of the parents of X_t, $\mathbf{Z}_t \subseteq \overline{\mathbf{X}}_t \subset \mathbf{X}$. The method forms a unique dependence structure among related variables. It restricts X_t to have no children while restricting its parents' set to $\mathbf{Z}_t$. Accordingly, $\mathbf{Z}_t$ can be roughly considered as a "diluted"

Markov blanket of X_t (containing its parents and their children). A particular application of the TBNL (namely, the classification and identification of suspected users based on their behavioral characteristics) is presented in this work. The TBNL method enables experts to control the network's complexity at the learning stage of the model, specifically, with respect to the class variable. In particular, we calculate the number of free parameters in the network as a conventional definition for measuring its complexity:

$$k = \sum_{i=1}^{N} (|X_i| - 1) \prod_{j=1}^{L_i} \left| X_i^j \right|, \tag{1}$$

where $|X_i|$ is the number of values that X_i can take, and $\left|X_i^j\right|$ represents all possible values that the parent variable X_i^j in $\mathbf{Z}_i$ can take (X_i has L_i parents). Constructing the TBNL follows the *adding-arrows* principle [34], which generalizes the optimal construction of Bayesian trees [11]. This principle aims to minimize the Kullback–Leibler (KL) divergence between the true probability distribution (p) and the one represented by the BN (q).

$$d_{\mathrm{KL}}(p||q) = \sum_{x_1,\ldots,x_N \in \mathbf{X}} p(x_1, \ldots, x_N) \log \frac{p(x_1, \ldots, x_N)}{q(x_1, \ldots, x_N)}. \tag{2}$$

By substituting an expression of a BN in place of the term q in Eq. (2), the following expression is obtained:
$d_{KL}(p(\mathbf{X})||q(\mathbf{X})) = -H(\mathbf{X}) - \sum_i I(X_i; \mathbf{Z}_i) + \sum_i H(X_i)$, where $H(X_i)$ is the entropy ascribed to the distribution of X_i, $H(\mathbf{X})$ is the joint entropy ascribed to the joint distribution of $\mathbf{X}$ and $I(X_i; \mathbf{Z}_i)$ is the mutual information between the variable X_i and its conditioning variables, namely its parents in the network [34]. Because the two entropy terms are independent of the BN model, the only term that could minimize the above KL divergence is the negative sum of the mutual information weights [34]. This leads to the observation that maximizing the total mutual information weight results in a BN that best approximates $p(\mathbf{X})$, where the edges pointing from parents to a node in the network are weighted by the information gain in the parents about the corresponding domain variable. It is critical to constrain the number of parents of each node; otherwise, the network will always be fully structured, with the maximum number of possible edges. Finding the best set of parents to maximize the BN accuracy subject to a given complexity bound, measured using Eq. (1), is an NP-hard optimization problem because the number of possible edge combinations increases exponentially in the number of domain variables [34]. In cases where subject matter knowledge is available, the problem of finding the most promising parents can be simplified. To extract such knowledge, it is beneficial to provide the experts with a convenient graphical interface in which the relations between features are presented in an intuitive manner. The TBNL method follows this approach. The expression for q in Eq. (2) takes the form of the expected conditional probability distribution of the class variable given the conditioning variables in the network, namely
$q(X_t) = \sum_{\overline{\mathbf{x}}_t \in \overline{\mathbf{X}}_t} p(X_t \mid \mathbf{z}_t)\, p(\overline{\mathbf{x}}_t)$, instead of the joint probability distribution $p(X_t) = \sum_{\overline{\mathbf{x}}_t \in \overline{\mathbf{X}}_t} p(X_t|\overline{\mathbf{x}}_t)\, p(\overline{\mathbf{x}}_t)$ of the entire domain, where $X_t \in \mathbf{X}$ represents the target variable and $\overline{\mathbf{X}}_t = \mathbf{X} \backslash \{X_t\}$ represents the remaining domain variables. This scheme leads to a specific BN structure that depends on the application associated with the target variable. More details of the TBNL algorithm can be found in the Appendix, and it is also presented in [18].

3.2 The TBNL applications

In this paper, we demonstrate how a TBNL application can be constructed to support suspect identification, anomaly detection and object classification. Each of the analytical tasks described in Sect. 1 can be represented by a proper random variable, which is then defined as the target variable for the TBNL model (as seen in Fig. 3). Moreover, note that—unlike other graphical Bayesian classifiers such as Naïve Bayes and the TAN—the TBNL method considers a flexible trade-off between the performance indicators derived from the target variable and the model complexity defined by the number of parameters, as seen in Eq. (1). In particular, by setting proper threshold values, the TBNL provides a convenient graphical presentation that can support reasoning and causality analysis by the subject matter experts. The algorithm applied to realize the TBNL method attempts to find the most relevant variables (features) with respect to the class variable. It does so by first selecting the parent nodes that maximize the information about that variable. Then, given those parents, it attempts to maximize the total information among them. The algorithm applies this concept by using a recursive procedure: for any given variable, it selects the set of parents that maximize the information over all the combinatorial options under a given set of constraints. The information maximization is greedy in the sense that, at each step, the procedure adds to the set of parents by selecting the variable with the largest conditional information gain given the existing parents. It stops when one or more of the constraints are violated or when it runs out of candidates. The applied algorithm first finds the parents of the class variable. Then, it applies the same procedure to each of the selected parents. The candidate parents for the class variable are all the variables in the space, whereas the candidate parents for any other arbitrary variable (which must be a parent of the class variable) are the members in the set of their parents that do not form cycles. Such a network is referred to as a "nuclear family" network because the variables compris-

ing it are those that are directly connected to the class variable (and possibly also to other variables). Although this procedure does not lead to a classical form of a Markov blanket [29], the rest of the variables are considered to be less relevant for classification purposes (See Fig. 3). Other constraints are implemented either as restrictions on the graph structure and its complexity (e.g., the maximum number of parents allowed for each variable) or as threshold conditions that are related to information theory measures and calculated from the network (e.g., the maximum allowed information weight about variables represented as nodes in the network). Other construction parameters are (1) an upper-bound parameter on the relative percentage information gained about any variable and (2) a minimum step-size parameter on the information gain for each additional parent. Note that when the constraint values are relaxed, one might obtain a complete network in which all the possible edges are drawn in any possible direction as long as the network configuration still maintains the DAG structure (all these configurations score an identical total information). On the other hand, under tighter constraint values, the order of the added edges becomes important to obtain a network with a lower complexity. By setting a low upper-bound parameter, one can generate a TBNL network with low complexity, resulting in a network model that is focused on the most relevant features for the classification task. After this set of relevant features has been selected, the user can either continue with a TBNL-based classification or consider this step as a preprocessing step for selecting features and then continue to perform classification with other classification models (such as linear regression, SVM, Naïve based or TAN) based on the selected features. Both options are implemented in Sect. 4; the former option is defined as TBNL classification and the latter as TBNL-based feature selection ('TBNL-based FS'). The TBNL is found to be an appealing learning model in cases where a graphical representation is helpful for modeling an unbalanced and noisy dataset because it provides an interactive interface through which subject matter experts can integrate their knowledge [18]. Such a case study is considered in this work by applying the TBNL method to a suspect identification task based on real-world cellular network data. It demonstrates a complete data mining process (which is described in the next section) while relying on feature selection, graphical representation and control of the accuracy–complexity trade-off.

3.3 TBNL and a KDD scheme

When applying the TBNL to a real use case, one must consider the overall operational procedure, which involves preprocess steps such as data acquisition and feature selection as shown in Fig. 1. In general, the figure follows the state-of-the-art knowledge discovery in databases (KDD) approach but with specific modifications for the TBNL implementation in this case. The process comprises four main stages, and each stage can be revisited several times, as needed. The first stage includes an Extract, Transform, Load (ETL) process for data cleaning and aggregation. Aggregation is a critical step in suspect identification because it determines the proper granularity level for each entity, resulting in a feature vector of corresponding characteristics for each entity type. The second stage determines the model design and derives the descriptive patterns that characterize the aggregated entity types. In this stage, a data scientist and a subject matter expert design the measures and the dimensions by which the data are sliced. The third stage includes mutual information (MI) and correlation tests between the various features and the selected target variable (designated by a class label that can change according to the analytic task) and among the features themselves, the goal of which is to discard redundant features. This stage is considered as a preprocessing step to the feature selection step. The last stage is data mining itself, in which the TBNL method is applied either for feature selection alone or both feature selection and classification. Note that in both cases the feature selection is performed with respect to the information gained about the target variable from the features defined in the analytic task using Eq. (2). At the same time, the complexity is constrained and shown graphically using Eq. (1) as well as the other required computations as described above.

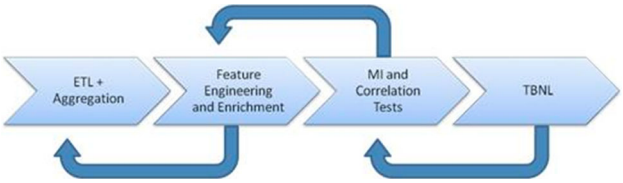

Fig. 1 TBNL-specific KDD implementation

The next section describes the implementation of the TBNL for various analytical tasks including suspect identification and comparisons to the conventional classifiers.

4 Suspect identification by the TBNL model

The four suspect detection tasks listed in Section 1 were defined by a leading homeland security company that preferred to remain anonymous. The list contains a set of preliminary tasks that eventually lead to the classification of users as suspects or non-suspects. Moreover, for the final classification task, the company was required to maintain a minimum recall of 50% along with an FPR no higher than 1%. Requiring such a low false detection rate is common in situations with unbalanced classes that might lead to a large number of suspects, and it is compatible with other analytical approaches—including those that consider the unbalanced

expected costs of statistical errors [17,20,24,33]. In the following section, we describe the data preprocessing and the initial steps performed in accordance with the KDD approach in Fig.1.

4.1 Initial study and data preprocessing

The initial dataset consisted of approximately 13 million CDRs of telephone communications and text messages generated by approximately 10,000 users (also interchangeably termed "objects" or "devices") monitored over 20 months. Each data record represents a communication event/interaction associated with a single user, regardless of whether the user was the initiator (caller/sender) or the receiver of the communication interaction. As a result of operational constraints and the objective of balancing the learned classes, the monitoring period was not uniformly distributed over all the users. In fact, the dataset was biased toward suspect users (e.g., it contained 6,175 suspects—a proportion far beyond the ordinary proportions of suspects in a population) obtained by using manual queries, tagging and pre-sampling processes. Note that such a procedure requires adjusting the trained classifiers when they are applied to datasets with non-balanced classes, either by using a Bayesian approach with a proper a posteriori distribution per class, as implemented here, or by applying other adjustment procedures (e.g., see [1,16]). First, an ETL and various aggregation processes were executed according to the KDD process shown in Fig. 1. Each monitored object in the database was labeled either as "suspect" or as "non-suspect" and associated with the main participating user in each CDR. It should be noted that while suspect users were fully monitored throughout the period, the dataset also contains users who were not monitored and who appear only as the "other party" communication partners of the monitored users. As indicated above, the ETL records contained only the metadata of the interactions; they included no information concerning the communication content itself. Specifically, the metadata consisted of a timestamp for the communication event, its type (call or text), its duration or size (depending on its type) and the ID codes for both the caller and the receiver. These IDs were generated specifically for the study by performing a one-time pseudonym mapping from the phone numbers to create the list of IDs. Thus, in this study any details on the names, identities or even the geographic associations that are generally available for landline numbers were unavailable. Note, however, that in a continuous implementation setting, some sort of encryption mapping is probably needed, which most often cannot mask the identity of the user given the metadata features. For example, as shown by De Montjoye et al. [13] only a few spatiotemporal points of data were sufficient to uniquely identify most users. The preprocessing stage also involved the implementation of data format standards and decomposition of the data fields to allow faster query processing. Auxiliary dimensions and relevant indices were defined and calculated to support the desired retrieval patterns, including various time-based slices, communication events (call vs. text), group association of CDR parties, listing of calendar events and an indication of the communication direction. This raw data processing allowed efficient and convenient data slicing as well as fast statistical calculations over the large data volumes. At this stage, to create a baseline, an initial classification study was carried out using the raw CDR details. An initial TBNL classifier was trained over the CDR features to indicate whether a tested CDR was associated with a suspected object. The corresponding performance at this initial stage achieved an average accuracy of 71% with a recall level of 95%, but it resulted in an FPR of over 50%, which did not meet the company requirements of a maximum 1% FPR as discussed above. Next, a feature engineering scheme (see Fig. 1) was implemented to extract a relevant feature vector to be used as an input to the learning algorithms. Various features were extracted and aggregated to enrich the learning dataset. In this stage, a transformation from CDR-level records to user-level records was performed. While the initial dataset contained a log of the detailed communication events and their corresponding characteristics, the aggregative dataset was associated with objects and their characteristics over the monitoring period. The ultimate objective was to obtain a user-descriptive dataset that would support the identification of suspects by analyzing the behavioral patterns of users. The initial feature vector consisted of metadata features for each object such as the number of contacts, average length of calls and the variance of contacts as well as data slice communication features such as the number of communications of each (call/text), the communication direction (incoming/outgoing), tagged object groups (e.g., caller/recipient) and descriptive statistics for these features (e.g., rate, average, standard deviation, minimum and maximum). Following the feature selection procedure described below, each augmented communication record contained approximately 30 discrete features. Discretization of continuous values followed the method proposed by Ching et al. [10], which attempts to maximize the normalized information score between the discretized feature and the class variable. After the features were defined, various time slices and augmentations were calculated for each feature over the CDR values. For example, interactions over any single day were divided into four quarters (after midnight, morning, afternoon and night) and the following augmented measures were calculated for each quarter of the day: the average number of incoming (or outgoing) calls, the variance in the number of incoming/outgoing text messages and so on. Some of these measures were defined in advance by company domain experts—for example, the requirement to separate weekdays and weekend days that are known to have different usage pat-

terns. Other measures and aggregations were found through experimentation based on the obtained classification results. For example, a weekly aggregation was found to yield better anomaly detection results, as described in the next section. Corresponding to the third stage of the KDD process shown in Fig. 1, pairwise mutual information (MI) and correlation tests were conducted between pairs of features. Features that were not correlated with the class variable or that were correlated to the class variable but were found to be redundant were removed from the dataset. This stage was repeated several times until the number of features stabilized. To neutralize the "seasonality" effects caused by the different monitoring periods and, in particular, the difference in the number of objects among different period durations, the feature values were normalized with respect to the monitoring time. For example, rather than measuring the total number of interactions in a monitoring period, the rate of interactions was taken into account. Although the dimension space of the various features increases exponentially with the number of characteristics and the quantity of statistical measures, the TBNL model was then used to select the most informative features. It used a features selection scheme by which information redundancy was measured and only features with high information gain on the class variable were selected. This scheme can result in a model that contains a relatively small number of features (as seen in Fig. 3) while controlling model complexity and maintaining a desired accuracy level. Each of the following subsections 4.2–4.5 corresponds to the final TBNL stage of the KDD process shown Fig. 1. In particular, each subsection briefly addresses one of the analytical tasks described in Sect. 1. For each task, a specific class variable and a set of features are defined and learned by a task-specific TBNL model. Due to space limitations, we provide only brief descriptions on some of these tasks.

4.2 Per-user anomaly detection

For this task, the objective was to construct a TBNL model to detect user anomalies. The target variable was defined to represent a deviation from the "normal" behavior of a user, aggregated over a given time period. Several different time periods were evaluated. Ultimately, a weekly period was found to yield the best modeling results. In particular, a month-aggregated dataset proved too small; it yielded biased models that underperformed in the testing phase. Similarly, a day-aggregated dataset was sparse: the aggregated values for many days for most users were too dull due to little daily activity. The week-aggregated dataset represents an overall usage behavior for a specific week. Note that the data were aggregated separately for suspect and non-suspect users, to enable a detailed and specific examination of these groups with respect to their normal/abnormal behavioral patterns. By learning a TBNL model based on these data features, it is possible to identify behavioral patterns on a weekly basis, from which anomalies can be detected. A dynamic likelihood threshold was specified to control the true positive/true negative rates. Per-user anomaly detection results were presented to security domain experts and found to be very accurate and valuable based on subjective judgments regarding anomalous per-user behavior. Following this aggregation, another ETL preprocessing step was conducted to remove outlier and anomalous records such as augmentations that resulted in smaller than expected values. Figure 2 shows a description of the week-aggregated anomaly dataset obtained by this process.

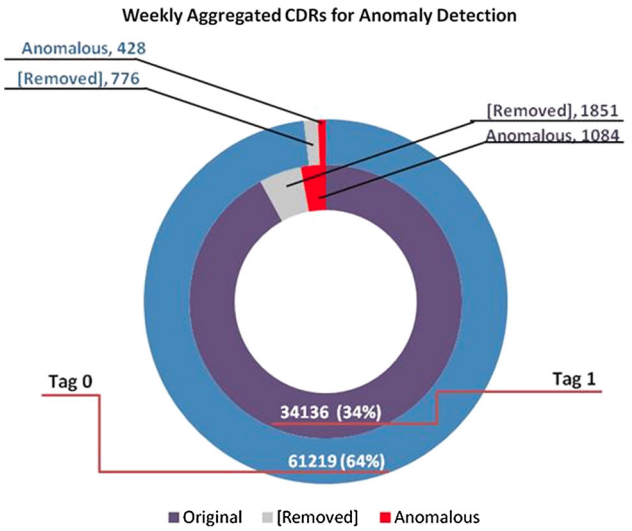

Fig. 2 Statistics on the week-aggregated dataset

4.3 Relevance of detected anomalies

For this task, the operational requirement was to distinguish between "relevant" versus "irrelevant" anomalies. An example of an irrelevant anomaly could be a sharp increase in the number of text messages a user sends when a substantial number of users simultaneously exhibit a similar behavioral pattern. This situation may occur, for example, during holidays, sport events or general emergency situations (e.g., a large storm in a certain area), all of which can cause many users to increase their text messaging interactions over a short time period, thus increasing their text-to-call ratio dramatically. A trained classifier would detect an anomaly for each of these users in that period, although, in fact, this pattern of behavior is not an anomaly during that particular time window. To avoid mass reporting of anomalies by the classifier in such cases, this task requires filtering out those seemingly irrelevant events and retaining only the relevant events. To address such scenarios, the TBNL model was augmented by a week index variable to determine whether a detected anomaly was shared by other users during a given

(indexed) week. Again, a weekly augmentation was found to yield better anomaly detection results with respect to other time period augmentations, as discussed above. Specifically, the TBNL model was trained with a new week index variable such that the probability parameters of the class variable could be conditioned on this week index variable. As a result, the TBNL classifier could represent two states: one for a global anomaly detection with respect to the entire period (unconditioned on the week index) and a second state for anomaly detection at that slice-specific week. At the testing phase, both model states were used to identify whether an observed record was, respectively, globally anomalous (termed a "global anomaly") and/or specifically anomalous for that week (termed a "slice-specific anomaly"). Table 1 summarizes the definition of "relevant" and "irrelevant" anomalies under the different scenarios. The abovementioned example of a high text-to-call ratio during holidays is represented by the third case in the table, where a global anomaly is considered irrelevant due to specific behavior during that week. Table 2 shows some cases classified by the logic rules described in Table 1. For example, observations 5 and 6 are interpreted as "irrelevant" anomalies according to Table 1.

Note that a similar slice-specific approach can be implemented using other variables (not necessarily related to time periods) such as age group, geographic location, education level, weekend or holiday flags and so on. The proposed implementation is in fact agnostic to the time period used as well as to the aggregation type; thus, the slice-specific feature can be adjusted by the requirements of domain experts.

Table 1 Truth table of global anomaly and slice-specific anomaly interpretation

Global anomaly	Slice-specific anomaly	Interpretation
No	No	Normal
No	Yes	Noise
Yes	No	Irrelevant
Yes	Yes	Relevant

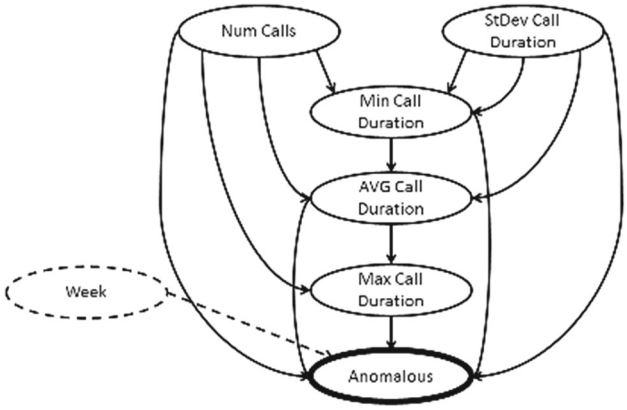

Fig. 3 An augmented TBNL network used for week-based anomaly detection. The tailored model is used to indicate the relevance of the detected anomalies to the classification task

The considered augmented model can be conveniently constructed by using the TBNL model while adding a slicing node as a parent node to the class variable. As indicated above, in the current task, a week-slice index variable ranging from 1 to 52 was selected as the parent of the class variable. All the rest of the network was constructed according to the proposed TBNL method (see the Appendix) to enable a direct comparison based on aggregated weekly slices. Figure 3 shows the augmented BN structure obtained at this stage.

4.4 Association of anonymous devices to existing users

In this task, the objective was to associate anonymous phone numbers with already known users. Such a scenario can happen, for example, when a suspect is using an anonymous pay card or a prepaid SIM in another cellular device. The association was obtained by using the TBNL to classify a

Table 2 Tested instances for determining whether an instance is relevant, intelligence-wise

ID	No. of text messages	No. of calls	Text/calls ratio	Day	Global anomaly	Slice-specific anomaly	Relevant anomaly
1	65	75	0.87	0.44	0	0	0
2	81	46	1.76	0.86	0	1	0
3	52	75	0.69	0.46	0	0	0
4	35	24	1.46	0.97	0	1	0
5	41	40	1.03	0.51	1	0	0
6	82	43	1.91	0.31	1	0	0
7	48	89	0.54	0.91	1	1	1
8	1	11	0.09	0.88	0	1	0
9	4	9	0.44	0.37	0	0	0
10	11	17	0.65	0.54	1	1	1

Table 3 Confusion matrix of new-to-existing device associations. The rows and columns represent the observed and classified objects, respectively. Entries in the table represent the number of classifications of a new device in the corresponding column, where the true label of the device is given by the corresponding row

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
A	21	9	9	0	2	0	0	0	0	1	4	2	0	1	0	6	0
B	9	16	9	1	1	1	0	0	1	0	2	4	0	5	0	6	3
C	4	4	31	2	9	5	0	1	3	1	14	0	0	0	0	2	0
D	0	1	0	55	0	0	0	0	0	0	1	0	0	0	0	0	7
E	4	3	7	0	25	2	0	0	0	0	20	1	0	1	0	3	0
F	2	3	2	1	0	53	1	0	12	0	0	1	0	1	0	1	0
G	0	0	0	0	0	0	51	0	0	0	0	0	0	0	0	0	0
H	1	0	0	1	1	1	0	74	0	0	0	5	0	4	0	0	0
I	0	0	0	0	0	16	0	0	94	1	0	0	0	0	0	0	5
J	0	0	0	0	1	0	0	0	0	85	0	0	0	0	0	0	0
K	5	3	5	0	12	0	0	0	0	0	41	0	0	0	0	1	0
L	1	7	0	0	1	1	0	0	0	0	4	38	0	11	0	3	0
M	1	0	1	6	0	0	1	0	0	1	0	0	68	0	0	1	0
N	0	5	1	0	0	0	1	0	1	0	1	8	0	38	0	1	1
O	0	0	0	0	0	0	0	0	0	2	0	0	0	1	49	0	0
P	1	3	1	0	0	2	0	0	1	0	2	3	0	0	0	37	0
Q	0	0	0	3	0	0	0	0	5	1	0	0	0	2	0	0	93

small set of unknown devices to the most similar device from a given set of existing (known) devices. Specifically, the TBNL classifier learned the behavioral patterns of each of the existing devices. The behavioral pattern was derived by predetermined aggregations resulting in a characteristic feature vector for each known device. In the testing phase, the CDRs of new (unknown) devices were aggregated, resulting in a corresponding usage feature vector for each new device. The TBNL was then used to classify the unknown device, to one of the existing learned devices, by pattern-matching likelihood scoring. A match score was computed for each pair of feature vectors between the unknown device and the existing devices. The evaluation was performed using a 10-fold cross-validation procedure (i.e., splitting the data randomly into to ten equal-size partitions and using each partition in turn as a test set for the model that was trained over the rest of the partitions). Seventeen users were considered, marked by the letters A–Q. For evaluation purposes, the class label of each (new) device in the test set was hidden, and a confusion matrix was then calculated according to the TBNL's classifications versus the actual classes.

Table 3 shows the confusion matrix resulting from the associations of an apparently unknown device to an existing device based on the TBNL classifications. Each column in the matrix represents the classification results, and each row represents the actual (hidden) class of the tested device. A correct association was achieved when a device was classified correctly to the actual class label. The diagonal entries in the confusion matrix represent all the correct associations of devices in this study. The ratio of the sum of the diagonal entries to the sum of all the entries in the matrix yields an accuracy level of 71.8%. This result could be primarily benchmarked to a random classification of seventeen potential classes, in which one would expect an accuracy level of less than 6%. Thus, the TBNL evaluation in this limited study shows an improvement of approximately 1200% that a new unknown device will be correctly matched to its corresponding devices using the pattern matching score derived by the TBNL classifier.

4.5 Classification of suspect users

In this stage, after preprocessing the dataset, identifying user-based relevant anomalies, and associating unknown users to known users, the main classification task was executed. The parameters of the TBNL were defined by a cross-validation procedure such that they yielded the desired ratio between the model's accuracy and its FPR versus the model's complexity. The default parameter values resulted in several classification TBNL models; one is shown in Fig. 4. The figure emphasizes again the importance of a graphical model, particularly for the TBNL in such a use case: the figure provides an intuitive understanding of the main factors, interactions and patterns that can be used to identify suspects by distinctive rules. For example, one can see that the minimum duration of an outcall (denoted by "min_call_duration_OUT") is influenced by the percentage of interactions during the after midnight hours (denoted by "Inter_prc_q1"), while the maximum call duration (denoted by "max_call_duration") is influenced by the percentage of interactions during the morning hours (denoted by "Inter_prc_q2"). These patterns can be used by security domain experts to provide intuitive communication profiles

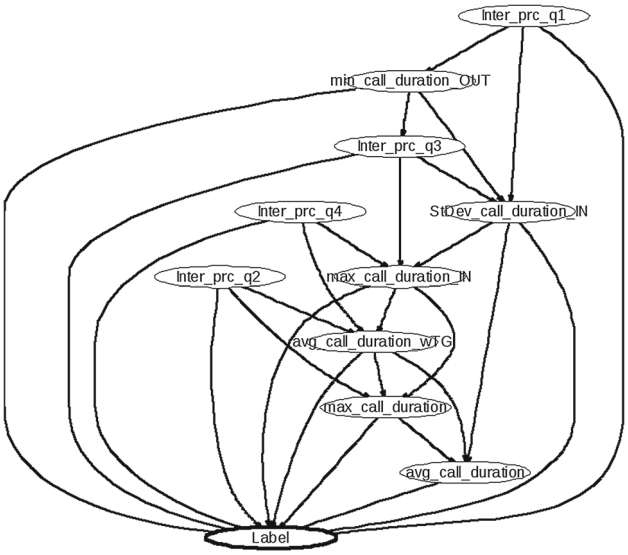

Fig. 4 A TBNL graph for suspect identification

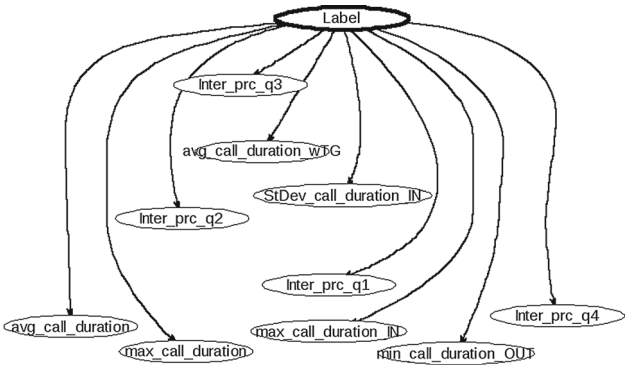

Fig. 5 A Naïve Bayes graph obtained using TBNL-selected features

to distinguish between suspects and non-suspects. Moreover, note that the TBNL graphical model is more informative than other graphical models such as the popular Naïve Bayes model shown in Fig. 5 (or the TAN model), in which all the features simply appear as child nodes of the class variable.

The process of investigating the TBNL models over the cross-validation process is combined with a forward feature selection procedure, which relies on the total information gain in each feature with respect to the class variable. The graphical output of the TBNL for a given accuracy–complexity trade-off such as the one presented in Fig. 4 enables a domain expert to better define the potential interactions among the features and include them in the classification model—either the TBNL itself or for use by another classifier that relies on the TBNL feature selection (for example, by including the relevant features and interaction terms in the exponent of a logistic regression model, and denoting this model by 'LR (TBNL based FS)'). After the features and their potential interactions have been exposed in the TBNL graph, domain experts can use it to find interesting

Table 4 Most influential features according to the TBNL model

Variable (feature)	Information gain (%)
Average call duration with suspects	38.49
Maximum call duration	29.26
Text-to-call ratio in the 2nd day-quarter	4.48
Average call duration	2.74
Percentage of contacts in the 3rd day-quarter	1.90
Minimum duration of a call out	1.32
Percentage of contacts in the 4th day-quarter	1.10
Percentage of contacts in the 1st day-quarter	1.05
Maximum duration of an incoming call	1.04
Standard deviation duration of an incoming call	0.75

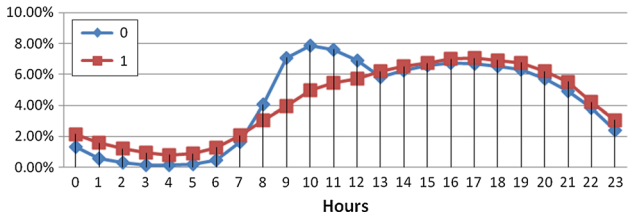

Fig. 6 Text-to-call ratio over 24 h for both suspects (labeled by "1") and non-suspects (labeled by "0"). Note the difference between the two populations during the late morning hours

and significant patterns in the data. For example, the interaction between 'maximum call duration' to the 'percentage of interactions during the morning hours' leads the security experts to measure the 'text-to-call ratio' in the second day-quarter, which was found to provide important information on suspects' behavior. Moreover, note that by increasing the information-theoretic threshold parameter of the TBNL method, the number of edges between the features grows, revealing more relevant interactions, up to a level that satisfies the required accuracy and provides an explanatory and intuitive description for the security experts. Table 4 shows the most influential features and interactions in terms of their mutual information scores about the class variable.

One example of a pattern that was exposed by the TBNL feature selection is shown in Fig. 6, which depicts the text-to-call ratio over 24 hours for both suspects (labeled by "1") and non-suspects (labeled by "0"). This pattern was exposed by the high information gain percentage of 4.48% for the "Text-to-call ratio in the 2nd day-quarter" feature, as shown in Table 4. This is the third important feature found by the TBNL model, right after the average call duration with other suspects' feature (which had an information gain percentage of 38.49%) and the "maximum call duration" feature (which had an information gain percentage of 29.26%). These two leading variables represent behavioral patterns of relatively short calls for the suspects' population. An intuitive analysis

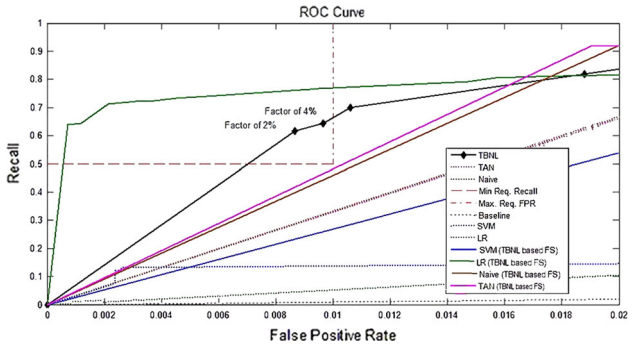

Fig. 7 A suspect detection ROC curve: Recall versus FPR

of Fig. 4 and Table 4 leads to the hypothesis made by the security expert that non-suspect (conventional) users communicate verbally (speak) more during the morning hours when they commute (drive) to work, while suspect users—who do not necessarily work on regular hours (and often use more than one phone: one for personal calls and the other for criminal activities)—continue to interact through texting during those hours. The expert's hypothesis was verified by drawing the proposed feature, "text-to-call ratio," of suspect users versus non-suspect users over the day. Figure 6 reflects the inherent differences between the behavioral patterns of the two populations. Clearly, these patterns are similar most of the time, despite a very noticeable difference in the 2nd day-quarter (especially from 9 am to 11 am) between the two populations. These behavioral patterns of relatively short calls of suspects emphasize that the TBNL can be used as a descriptive quantitative model to extract patterns and reveal insights for suspect identification by domain experts.

Finally, the proposed TBNL is compared to popular state-of-the-art classifiers. Figure 7 shows the ROC curves of three graphical models: the TBNL, the TAN and the Naïve Bayes classifiers. In addition to these graphical models, it also benchmarks the ROC curves of two non-graphical cornerstone classifiers: the SVM and the logistic regression (LR) classifiers. All the compared classifiers are analyzed in two configurations. The first configuration consists of a conventional implementation of each classifier using the straightforward procedures available in R. The second configuration is an implementation of these classifiers following the TBNL's feature selection procedure as described above. The second configuration is denoted in the figure as (''TBNL-based FS''). The vertical axis in Fig. 7 describes the detection rate (Recall), while the horizontal axis describes the FPR—both based on a fivefold cross-validation procedure. Note that the horizontal axis is magnified to represent the "required classification area" with a false positive rate less than 1% and a recall of over 50%. The only two classifiers that met the company requirement were the TBNL itself (denoted by "TBNL") and the logistic regression model whose exponent was based on the TBNL feature selection procedure (denoted by "LR (TBNL-based FS)"), which provides the best classification result. Namely, in this model the TBNL is used to find and engineer the features and their relevant interactions (as detailed above) and these features and interactions are then applied to the exponent of the Logit function to better map the result to a [0–1] probabilistic measure by using the revised LR model. None of the other classifiers meet the industrial requirement. Sorted by their performance in descending order they are: TAN based on the TBNL feature selection procedure, denoted by "TAN (TBNL-based FS)"; Naïve Bayes based on the TBNL feature selection procedure, denoted by "Naïve (TBNL-based FS)"; TAN, denoted by "TAN"; SVM based on the TBNL feature selection procedure, denoted by "SVM (TBNL-based FS)"; and SVM, denoted by "SVM"; and the linear regression classifier, denoted by "LR." The lower line in the figure represents a baseline of 45°, which is equally balanced between the recall and the FPR. Note that the TBNL outperformed the TAN, the Naïve Bayes, the SVM and the logistic regression (LR) classifiers when using the set of all available features. The performance of all these classifiers improved when they followed the TBNL feature selection procedure, which is based on a complexity–accuracy trade-off and a graphical interface to select the final set of features and interactions. The LR TBNL-based FS model then obtained the best classification results by far—dominating all the other tested classifiers. Both the TAN and the Naïve Bayes classifiers detected at least 3705 of the 6175 suspects, along with approximately 80 out of 4210 users that were falsely classified as suspects. In comparison, the TBNL classifier obtained a recall value of 60% with an FPR of less than 1%. The TAN classifier obtains a similar recall value, but with a 2% FPR. The Naïve Bayes classifier's performance is even worse. Figures 4 and 5 above show the corresponding graphs of the TBNL and the Naïve Bayes models, respectively.

5 Conclusions

The results obtained by the considered use case show that the TBNL method obtained a 50% recall with a false positive rate of no more than 1%. Note that these results were obtained without accessing the contents of the CDRs; only their metadata were used and analyzed to characterize users' behavioral patterns. The added value of the TBNL lies in its capability to efficiently manage the trade-off between model complexity and accuracy as well as in its ability to provide an informative graphical interface that allows security domain experts to investigate and find the behavioral patterns that can distinguish suspect from non-suspect users. Regarding this particular use case, the most influential characteristics for the classification task were found to be the durations of the

CDRs and their derivatives in various crossovers, such as the average call duration with other suspects and the distribution of calls and text messages throughout the four quarters of the day. We used primary statistical metrics; however, we do not claim that the feature engineering task was optimal, and we leave this discussion for future research. The applied algorithm considers such new features during the BN learning stage while providing an intuitive presentation that subject matter experts can grasp easily.

Acknowledgements This research was partially supported by the Israeli Chief Scientist Magneton program no. 44596, "Target-Based Bayesian Network Modeling for Homeland Security applications" (Principle Investigator: Prof. Irad Ben-Gal). We are grateful for the support of our colleagues from the industry in this project, as well as for Shai Yanovski's participation in the project.

Appendix: the TBNL algorithm

The TBNL algorithm (for more details see [18]) uses a recursive procedure that can be applied to any node that represents a variable. The procedure, called *AddParents* (described below) adds edges from candidate nodes to the node to which the procedure is currently applied: each time, it adds the edge from the node with the highest Information Gain (IG) value. Essentially, *AddParents* is a greedy, forward feature selection procedure, which is similar to the feature selection scheme used by the *adding-arrows* principle [34]. The main difference is that the TBNL algorithm starts with the class variable and then proceeds recursively to the selected parent nodes. In particular, the TBNL algorithm starts by applying the *AddParents* procedure to the target node to select its parents. Then, *AddParents* is applied to each parent sequentially to select its own parents from the set of the target node's parents. Thus, any node in the network can be a parent of the target node (i.e., corresponding to a limited form of a Markov blanket) while still maintaining the DAG structure. The input parameters of the *AddParents* procedure are as follows: X_i represents the current node; $\mathbf{T}_i$ represents the set of the candidate parents of X_i; $\mathbf{C}$ represents the set of arbitrary constraints on the network such as the number of permitted parameters; η_i represents a constraint of the maximum allowed MI concerning X_i; and β_i represents the minimum IG "step size" when adding a parent to X_i in the network. After applying the *AddParents* procedure to node X_i, the output is the set of parent nodes $\mathbf{Z}_i$ if one of the following conditions is fulfilled: 1) any of the $\mathbf{C}$ constraints is not met; 2) $I\left(X_i; \overline{\mathbf{Z}}_i | \mathbf{Z}_i\right) / H\left(X_i\right) < \beta_i$; or 3) the set of candidate parents $\mathbf{T}_i$ is empty. The *AddParents* procedure is shown next. The last two code lines imply that it is a quasi-recursive procedure; namely, the TBNL algorithm actually calls *AddParents* only once. Then, having obtained $\mathbf{Z}_i$, it iteratively calls $\mathbf{Z}_j = AddParents\left(X_j, \overline{\mathbf{Z}}_i, \mathbf{C}, \eta_j, \beta_j\right)$ for each $X_j \in \mathbf{Z}_i$. Note that the order of the iterations is well defined: the output parents from each iteration directly affect the input of the next step. Such a procedure generates different outputs than those that would have been obtained had the algorithm iterated the procedure only after obtaining the full set of parents. Thus, the TBNL calls $\mathbf{Z}_t = AddParents\left(X_t, \overline{\mathbf{X}}_t, \mathbf{C}, \eta_t, \beta_t\right)$, which ultimately results in a DAG $\mathcal{G} = \{\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_N\}$.

Input:
$\mathcal{G}$: a DAG; X_i: the current node; $\mathbf{T}_i$: the set of candidate parents of X_i; $\mathbf{C}$: a set of constraints on the BN; η_i: maximum MI about X_i; β_i: minimum IG about X_i given $\mathbf{Z}_i$
Output:
$\mathbf{Z}_i$: the set of parents of X_i
Procedure:
Set $\mathbf{Z}_i = \{\emptyset\}$; Set $\mathrm{T}'_i = \mathrm{T}_i$
Stop=FALSE
While $\mathrm{T}'_i \neq \{\emptyset\}$AND NOT (Stop)
    $X_j = \text{argmax}_{j':X_{j'} \in \mathrm{T}'_i}\{ I\left(X_i; X_{j'} | \mathbf{Z}_i\right) / H\left(X_i\right) \}$
    Add an arrow $E_{ji} = V_j \text{—>} V_i$ in $\mathcal{G}$
    If $\mathbf{C}$ is not met
        Stop=TRUE
    If $I\left(X_i; \mathbf{Z}_i \cup \{X_i\}\right) \geq \eta_i$
        Stop=TRUE
    If $I\left(X_i; X_j | \mathbf{Z}_i\right) / H\left(X_i\right) < \beta_i$
        Stop=TRUE
    If Stop
        Remove E_{ji}
    Else
        $\mathbf{Z}_i = \mathbf{Z}_i \cup \{X_i\}$
        $\mathbf{T}'_i = \mathbf{T}'_i \setminus \{X_j\}$
End {While}
For each $X_k \in \mathbf{Z}_i$
    $\mathbf{Z}_k = \text{AddParents}\left(\mathcal{G}, X_k, \mathbf{Z}_i, \mathbf{C}, \eta_k, \beta_k\right)$
Return $\mathbf{Z}_i$

References

1. Ben-Akiva, M., Bierlaire, M.: Discrete choice methods and their applications to short term travel decisions. In: Handbook of Transportation Science, pp. 5–33. Springer, New York (1999)
2. Ben-Dov, M., Wu, W., Feldman, R., Cairns, P.A.: Improving Knowledge Discovery by Combining Text-Mining & Link Analysis Techniques. Lake Buena Vista, Florida: Workshop on Link Analysis, Counter-terrorism, and Privacy, in conjunction with SIAM International Conference on Data Mining (2004)
3. Ben-Gal, I.: Bayesian networks. In: Ruggeri, F., Faltin, F., Kenett, R. (eds.) Encyclopedia of Statistics in Quality and Reliability. Wiley, New Jersey (2007)
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)

5. Bolton, R.J., Hand, D.J.: Statistical fraud detection: a review. 2002. Stat. Sci. **17**, 235 (2002)
6. Bouchard, M., Joffres, K., Frank, R.: Preliminary analytical considerations in designing a terrorism and extremism online network extractor. In: Computational Models of Complex Systems, pp. 171–184. Springer International Publishing (2014)
7. Boulton, G.: Open your minds and share your results. Nature **486**(7404), 441–441 (2012)
8. Burns K.: In US cities, open data is not just nice to have; it's the norm. The Guardian, 21 Oct 2013. www.theguardian.com/local-government-network/2013/oct/21/open-data-us-san-francisco (2013)
9. Chickering, D.M., Geiger, D., Heckerman, D.: Learning Bayesian networks: the combination of knowledge and statistical data. Mach. Learn. **20**, 197–243 (1995)
10. Ching, J.Y., Wong, A.K.C., Chan, K.C.C.: Class-dependent discretization for inductive learning from continuous and mixed mode data. IEEE Trans. Pattern Anal. Mach. Intell. **17–7**, 641–650 (1995)
11. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inf. Theory IT-14, 462–467 (1968)
12. Claeskens, N.L., Hjort, G.: The focused information criterion. J. Am. Stat. Assoc. **98**, 900–945 (2003)
13. De Montjoye, Y.A., Radaelli, L., Singh, V.K.: Unique in the shopping mall: On the reidentifiability of credit card metadata. Science **347**(622), 536–539 (2015)
14. Duda, R.R., Hart, P.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
15. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**, 131–163 (1997)
16. Ganganwar, V.: An overview of classification algorithms for imbalanced datasets. Int. J. Emerg. Technol. Adv. Eng. **2**(4), 42–47 (2012)
17. Grau, J., Ben-Gal, I., Posch, S., Grosse, I.: VOMBAT: prediction of transcription factor binding sites using variable order Bayesian trees. Nucleic Acids Res **34**(suppl 2), W529–W533 (2006)
18. Gruber, A., Ben-Gal, I.: Efficient Bayesian network learning for optimization in systems engineering. Qual. Technol. Quant. Manag. **9–1**, 97–114 (2012)
19. Heckerman, D.: A tutorial on learning with Bayesian networks.: MS TR-95-06 (1995)
20. Jensen, D., Rattigan, M., Blau, H.: Information awareness: a prospective technical assessment. In: Proceedings of SIGKDD03 , pp. 378–387 (2003)
21. Kelner, K., Lerner, B.: Learning Bayesian network classifiers by risk minimization. Int. J. Approx. Reason. **53**, 248–272 (2012)
22. Kreykes, B.D.: Data mining and counter-terrorism: the use of telephone records as an investigatory tool in the war on terror. ISJLP **4**, 431 (2008)
23. Marturana, F., Tacconi, S.: A machine learning-based triage methodology for automated categorization of digital media. Digit. Investig. **10**, 193–204 (2013)
24. Mayer, J., Mutchler, P., Mitchell, J.C.: Evaluating the privacy properties of telephone metadata. Proc. Nat. Acad. Sci. **113**(20), 5536–5541 (2016)
25. Mena, J.: Homeland security techniques and technologies. Charles River Media **198**(254), 262–263 (2007)
26. Meng, G., Dan, L., Ni-hong, W., Li-chen, L.: A network intrusion detection model based on K-means algorithm and information entropy. Int. J. Secur. Appl. **8**(6), 285–294 (2014)
27. Nhauo, D., Sung-Ryul, K.: Classification of malicious domain names using support vector machine and Bi-gram method. Int. J. Secur. Appl. **7**(1) January, 51 (2013)
28. Ng, A., Jordan, M.: On discriminative versus generative classifiers: a comparison of logistic regression and naive Bayes. Adv Neural Inf. Process. Syst. v2. pp. 841–848 (2002)
29. Pearl, J.J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
30. Pearl, J.J.: Causality: Models, Reasoning, and Inference. University Press, Cambridge (2000)
31. Phua, C., Lee, V., Smith, K., Gayler, R.: A comprehensive survey of data mining-based fraud detection research. www.bsys.monash.edu.au/people/cphua (2005)
32. Shmueli, E., Tassa, T.W., Shapira, B., Rokach, L.: Data mining for software trustworthiness. Inf. Sci. **191**, 98–127 (2012)
33. Stolfo, S.J., Fan, W., Lee, W., Prodromidis, A., Chan, P.K.: Cost-based modeling for fraud and intrusion detection: results from the JAM project. In: DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. IEEE Proceedings, vol. 2, pp. 130–144. (2000)
34. Williamson, J.J.: Approximating discrete probability distributions with Bayesian networks. Hobart Tasmani, Proceedings of the International Conference on Artificial Intelligence in Science and Technology (2000)
35. Van Renesse, R., Birman, K., Vogels, W.: Astrolabe: a robust and scalable technology for distributed system monitoring management, and data mining. ACM Trans. Comput. Syst. **21**, 164–206 (2003)
36. Zhu, D., Premkumar, G., Zhang, X., Chu, C.H.: Data mining for network intrusion detection: a comparison of alternative methods. Decis. Sci. **32**, 635–660 (2001)