A Datagenerator for Evaluating Machine Learning Methods

Martin Atzmueller, Joachim Baumeister, Mario Goller, Frank Puppe

Evaluating and comparing machine learning methods concerning the effectiveness and the efficiency is a general research problem. For a careful evaluation the availability of data sets with known characteristics is a key issue. We propose a novel approach to intuitively generate data sets for the evaluation and the comparison of machine learning methods. Three case studies are provided to demonstrate the presented approach.

1 Introduction

The evaluation of machine learning methods is a general research problem: usually the quantitative and qualitative review of a method and its comparison with related methods is of scientific and practical interest. When evaluating a new method we commonly consider the following general issues:

- *Efficiency:* Theoretically, the $\mathcal{O}$–notation is used to describe an algorithm concerning its time and space complexity. In practice, it is also interesting to see how the time-performance of the method and its implementation scales when the number of data samples is increased.
- *Effectiveness:* First, the effectiveness of a method often concerns its accuracy – usually determined by measures like precision or recall. Second, the effectiveness of a method may concern the fraction of correctly learned patterns: Then, it is very important to control the characteristics of the data, i.e., the included patterns. Sometimes the effectiveness is also related to the *robustness*, i.e., how a method performs if the data contains noise.

We see a general problem when one wants to properly evaluate a machine learning method, because a priori often no data base with appropriate samples is available. At best, a data base embodies a sufficiently large collection of data sets with varying sizes and describing characteristics at different complexity levels. Such a setting would allow for a complete and representative evaluation of the method. In the past, data sets were collected from real world applications or were taken from synthetic collections such as the UCI repository [7]. However, existing data sets like the UCI repository as well as samples from real world applications often lack reliability when evaluating a machine learning method. On the one hand, the efficiency of the method cannot be precisely tested if the data sets with different sizes do not show equivalent characteristics. On the other hand, the effectiveness of the method cannot be concisely determined if the complexity of the included patterns is not known beforehand and it therefore cannot be appropriately diversified, respectively.

Salzberg [10] describes a similar problem: It is very difficult to generalize evaluation results using a limited collection of data sets with fixed characteristics. Using well-known evaluation data it also becomes easier to (possibly unintentionally) tune parameters of algorithms in order to obtain better results. Consequently, a collection of data sets with known characteristics allows for the estimation of the performance of a new algorithm since the input characteristics of an algorithm can be controlled.

In this paper, we present a novel approach to generate data sets for the evaluation and comparison of machine learning methods. The approach allows for the precise definition of the characteristics of the data set and its size. The characteristics of the data can be intuitively specified using subgroup descriptions and are later refined using fragments of Bayesian networks. Based on these a (final) data generation model is created: Then, a collection of data sets with varying size can be created according to the given semantics.

The rest of the paper is organized as follows: In Section 2 we describe the process model for data generation. Next, the data generation steps are described in Section 3 in detail. We describe the adaptation techniques in Section 4. Section 5 evaluates the proposed approach with three case studies. We conclude in Section 6 with a summary of the presented work and discuss related approaches.

2 The Data Generation Process

In this section, we first introduce the used knowledge representation and describe the basics of subgroup patterns. After that, we give an informal introduction to Bayesian networks. Finally, we present the process model for generating data which is discussed in more detail in the next section.

2.1 General Definitions

Domain Ontology Let Ω_A the set of all **attributes**. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Furthermore, we assume $\mathcal{V}_A$ to be the (universal) set of **attribute values** of the form $(a = v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value. Let DS be the **data set** containing all available instances. An **instance** $i \in DS$ is given by the tuple $i = ((a_1 = v_1), \ldots, (a_k = v_k))$, $(a_i = v_i) \in \mathcal{V}_A$, $k = |\mathcal{V}_A|$, containing the individual attribute values.

Subgroup Patterns Subgroup patterns [5], often provided by conjunctive rules, describe 'interesting' subgroups of cases, e.g., "the subgroup of 16-25 year old men that own a sports car are more likely to pay high insurance rates than the people in the reference population." Subgroups are then described by relations between independent (explaining) variables (*Sex=male*, *Age=16-25*, *Car type=Sportscar*) and a dependent (target) variable (*Insurance rate*).

A subgroup pattern mainly relies on the following four main properties: the subgroup description language, the subgroup size, the target share of the subgroup, and the target variable. In the context of this work we focus on binary target variables. The description language specifies the individuals from the reference population belonging to the subgroup.

Definition 1 (Subgroup Description) *A subgroup description* $sd = \{e_1, \ldots, e_n\}$ *consists of a set of selection expressions (selectors) that are selections on domains of attributes, i.e.,* $e_i = (a_i, V_i)$*, where* $a_i \in \Omega_A, V_i \subseteq dom(a_i)$*. A subgroup description is defined by the conjunction of its contained selection expressions.*

Then, the subgroup s is the set of instances that fulfill the subgroup description $s = \{i \,|\, i \in DS \wedge sd(i) = \mathit{true}\}$. The parameters subgroup size n and target share p of the subgroup are mainly used to measure the interestingness of the subgroup, where n is the number of instances contained in a subgroup s, i.e., $n = \left|\{i \,|\, i \in s\}\right|$; the target share $p = \frac{1}{n} \cdot \left|\{i \,|\, i \in s \wedge t \in i\}\right|$ specifies the share of subgroup instances that contain the target value $t \in \mathcal{V}_A$, and is usually compared to p_0, the target share of the total population.

Bayesian Networks A Bayesian network consists of a set of attributes and a set of directed edges connecting the attributes (e.g., [4]). For each attribute a the range $dom(a)$ has to contain a finite set of distinct values. A directed acyclic graph is defined by the attributes and the set of edges inducing dependency relations between pairs of attributes. For each attribute a and its parents $pa(a)$ (induced by the edges) a conditional probability table (CPT) is attached. For an attribute with no parents an unconditioned prior probability is used. We will show examples of Bayesian networks in the case studies in Section 5.

2.2 The Process Model

In the following, we define the incremental process model for generating data. We apply the *data generation model* given by a Bayesian network as the underlying knowledge representation: Using the network, we are able to express the dependency relations between the individual attributes capturing the specific data characteristics. Then, we can generate the output data quite easily, e.g., by applying sampling algorithms.

The difficult part is the construction of the Bayesian network itself which is usually non-trivial: The basic elements of the network, i.e., the nodes with the attached conditional probability tables are easy to model at the local (node) level. In contrast, entering all the conditional probabilities is often a difficult problem, e.g., if relations between nodes need to be considered that are not directly connected.

Therefore, we aim to help the user in an incremental process, where the data characteristics can be described from abstract to more specific ones. Parts of the data generation model can be described using subgroup patterns which are structurally mapped to the defined Bayesian network in turn. Then, these patterns (or constraints) can be checked on the model, and inconsistencies of the model can be identified. Alternatively, the Bayesian network can be defined or modified directly with an interactive adaptation step using the given (subgroup pattern) constraints as test knowledge The process consists of the following steps shown in Figure 1.

1. **Define Domain Ontology**: In the process we first define the domain ontology, i.e., set of attributes and values used for the data generation.
2. **Specify Data Generation Model**: The Bayesian network or fragments of the network can be either be specified manually, or it can be generated automatically by applying subgroup patterns that describe the interesting data characteristics: These patterns describe relations between a dependent and several independent variables, with certain characteristics given by the subgroup parameters, i.e., the subgroup size and the target share of the subgroup. The given patterns and network fragments are structurally merged into the *data generation model* for the output generation. The relations between the variables are represented in the network by connections of the individual nodes. However, the strength of these relations may not yet be adequately represented by the conditional probability tables. Thus, using the subgroup parameters, constraints are derived in order to check these relations. Such constraints can also be supplied by the user directly. Thus, after the model has been initialized by the user the model is tested given the available set of constraints.
3. **Adapt Specification/Optimize Model**: If the model fits the constraints, then the process is finished, and the data generation model is ready for use. Otherwise, an optimization step is applied in order to adjust the conditional probability tables contained in the network. Alternatively, the user can also either adapt the patterns/constraints, modify the network structure, or try to edit the conditional probability tables by hand.
4. **Generate Data**: After the data generation model has been fit to the specification of the user, the final data generation step is performed using a sampling algorithm. Given the network we apply the prior-sample algorithm [4] also known as forward-sampling: In a top-down algorithm for every node a value is computed according to the values of its parent nodes.

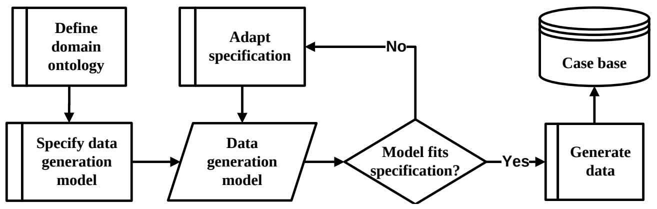


Figure 1: Process model for data generation.

3 Interactive Data Specification

In this section, we describe the interactive data modeling approach: After modifications by the user, the data generation model can always be adjusted in an interactive process. As discussed above, we use a Bayesian network to represent the data generation model: The network can be approximated by patterns, or the user can manually define the Bayesian network. In either case, the network can be adapted with respect to the specification in a final interactive step.

Model Approximation by Patterns A collection of subgroup patterns describe dependencies between a target variable and a set of explaining variables. Using such subgroup patterns a two layered network can be constructed automatically. Either the target variable can be designated as the parent of the explaining variables, or as the child of the set of variables.

For an example, let us consider propositional conjunctive rules with the target variable in the rule head. Using subgroup patterns we can model categoric rules as well as probabilistic rules. For categoric rules we see that these have a target share $p = 1$, i.e., the target concept is always established. For probabilistic rules the target share is given by $p \in [0; 1]$. For example, considering test data for discovering association rules [1], it is easy to see that the target share relates to the concept of *confidence*, because p is equivalent to the conditional probability $p(t \mid sd)$ of the target variable t given the subgroup description sd; the *support* of an association rule is given by the joint probability $supp = p(t \wedge sd)$. Then, we can describe the interesting patterns for data generation quite easily by specifying these parameters.

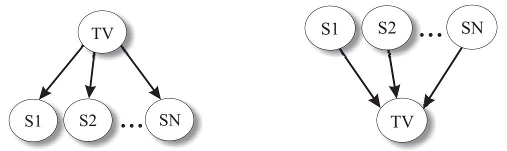
Interactive Modeling of a Bayesian Network In addition to the specification of a set of subgroup patterns the user can also define the Bayesian network directly by connecting the nodes/attributes. Additionally, the conditional probability tables of the nodes in the network can be adapted. If additional nodes are entered manually, then the entries of the conditional probability tables need to be specified. However, this step is usually the most difficult one. Therefore, we provide an interactive adaptation step as discussed below.

In an advanced step the network structure can be enriched using hidden nodes that enable further possibilities for data generation: Hidden nodes are used to add constraining relations of the *active nodes* that are used for data generation. The hidden nodes are used for the data generation step, but they are not visible in the generated data.

Constraint-based Model Adaptation The major step during the construction of the data generation model is the adaptation of the (partially) defined Bayesian network. We can apply constraints represented by joint and conditional probabilities of the network that can be derived from the defined subgroup patterns. Additionally, the user can manually specify such constraints. The set of constraints can then be applied in order to measure the state of the model, and to verify the specification.

The specified constraints are used to optimize the conditional probability tables of the given network since the constraints are defined using the entries contained in the conditional probability tables of the nodes of the network. Then, a hill-climbing constraint satisfaction problem-solver is used to fit the model to the constraints minimizing a global error function. After the CSP-solver has been applied the resulting state of the model can be controlled by the user interactively: The deviations of the defined patterns and the patterns included (implicitly) in the network are compared. Then, the model is tuned if necessary.

For each target variable a constraint is generated using the specified total target share considering the entire population, i.e., the prior probability of the target variable $p(t)$. Constraints for the parameters of the subgroup patterns are based on the contained target variable t and the set of selectors in the subgroup description $sd = \{e_1, e_2, \ldots, e_n\}$. Two constraints are generated for each pattern, i.e., the subgroup size equivalent to the joint probability $p(e_1, e_2, \ldots, e_k), e_i \in sd$, and the target share of the subgroup equivalent to the conditional probability $p(t \mid e_1, e_2, \ldots, e_k), e_i \in sd$. Using the subgroup patterns defined for a target variable the user can select from two basic network structures that are generated automatically if the respective nodes contained in the subgroup description are not already contained in the network: Either the target variable is the parent of the subgroup selectors (Figure 2(a)), or the target variable is the child of the subgroup selectors (Figure 2(b)). The first figure depicts the relation 'IF target *TV* THEN selectors S_i'; the latter models the inverse relation 'IF selectors S_i THEN target *TV*'.



(a) *If* target *TV*, *then* selectors S_i

(b) *If* selectors S_i, *then* target *TV*

Figure 2: Possible network structures for modeling the dependency relations between the target (dependent) variable and the selectors (independent variables).

These options need to be selected based on the desired relations of the data generation model. Both structures have certain advantages and drawbacks concerning the optimization step: Option a) includes a simple definition of the prior probability of the target variable; using the CPTs of the children selectors it is very easy to adapt the parameter *subgroup size*. On the other hand, option b) allows for an easier adaptation of the subgroup target shares (contained as values in the CPT of the target variable). Furthermore, option b) typically results in larger CPTs. This allows for better adaptation possibilities for the optimization algorithm. However, the size of the CPT of the target is exponential with respect to the number of parent selectors.

4 Constraint-Based Adaptation of a Bayesian Network

In the following we discuss how the constraint-based adaptation of the data generation model is implemented. We first describe how we compute arbitrary probabilities in the Bayesian network. After that, we show how the CPT entries of the network are adapted using a CSP-solver.

Computing Arbitrary Joint Probabilities In order to check and to optimize the constraints specified for the data generation model we need to compute joint probabilities of node values corresponding to (arbitrary) combinations of subgroup selectors. Additionally, conditional probabilities of a target variable given a set of selectors also need to be calculated. It is easy to see that if we can compute arbitrary joint probabilities in the Bayesian network, then we can also compute arbitrary conditional probabilities. In principle, arbitrary joint probabilities can be estimated using a sampling method, e.g., the forward sampling algorithm discussed in Section 2.2, by counting the respective joint occurrences. However, this is only an approximation of the real probabilities and also requires many generated samples. Therefore,

Algorithm 1 Computing arbitrary joint probabilities.

```
Require: Joint probability $W$, where $var(W)$ are the variables of $W$
 1: Group nodes by layers: Nodes with a longest path to a root node of size $k$ are sorted into layer $k$.
 2: Current network layer $n := 0$.
 3: repeat
 4:     for all nodes $K$ of layer $n$ do
 5:         if $K \in val(W)$
            then $val^+ = \{v \mid v \in values(K) \wedge v \in W\}$,
            else $val^+ = values(K)$
 6:         for all $v_K \in val^+$ do
 7:             Compute the prior probability $p(v_K)$ of the relevant node values $val^+$:
                $p(v_K) = \sum_i ( p(v_K \mid v_{par}(K)_i) \cdot \prod_j p(v_j) )$, where $v_{par}$ is a combination of the node values of the parent nodes and $v_j \in v_{par}$ is a single value of such a parent node.
 8:         Let $con_{children}(K)$ be the set of child nodes of node $K$ that are linked to a node contained in $W$, and that are not connected to each other. For every value combination $v_{children}(K)$ of the nodes contained in $con_{children}(K)$, we compute:
            $P(v_K, v_{children}(K)) = P(v_K) \cdot \prod_i P(v_i \mid v_K)$, where $v_i \in v_{children}(K)$
 9:         n = n + 1
10: until the next network layer is empty or the desired joint probability can be computed.
```

we propose to compute arbitrary joint probabilities directly using the atomic CPT values, and cache the used calculation rules. Then, as long as the structure of the network does not change, the modification of individual CPT values and their effect on specific joint and conditional probabilities can be immediately calculated.

The proposed approach is a top-down method shown in Algorithm 1: Starting with the root nodes of the network, in a recursive process the probabilities of the children nodes can be computed given the prior probabilities of their parents. Calculation rules (for computing the individual probabilities) created by the algorithm are stored for later reference. Then, these can be reused, e.g., for the automatic adaptation of CPT-entries or for computing joint probabilities for nodes in diverging connections in the algorithm. The computation works in a breadth-first manner and terminates early, if the desired joint probability can be already computed.

Automatic Adaptation of the CPT Entries As discussed above the *explicit constraints* specified by a set of subgroup patterns refer to probabilities that can be calculated using atomic CPT entries of the Bayesian network. In addition, the network includes also implicit constraints, i.e., the completeness relation for a given CPT: $\sum_i p(x_i \mid parents(X)_j) = 1$, for a given node X with values x_i and parents $parents(X)$, with the corresponding values $parents(X)_j$. The explicit constraints can also be disabled during the optimization step by the user on demand, e.g., if it is clear that they cannot be fulfilled. Additionally, a priority category can be assigned to each constraint, i.e., *low*, *normal*, *high*, that is considered when selecting the adaptation strategy.

In order to optimize and to adapt the network, we need to compare the values of the explicit constraints with their current values. Then, we apply a hill-climbing constraint satisfaction problem-solver to solve possibly occurring inconsistencies. For the adaptation of the data generation model, we use the created calculation rules for the joint probabilities (see Algorithm 1) corresponding to the constraints. After that, we apply a recursive approach terminating in the adaptation of CPT entries. Let us consider three probabilities p_1, p_2, p_3 in a Bayesian network. We distinguish the following three elementary cases for computing p_1 given p_2 and p_3:

1. $p_1 = p_2 + p_3$,
2. $p_1 = p_2 \cdot p_3$, and
3. $p_1 = p_2 / p_3$.

Using these relations, we can describe any probability computation recursively, terminating when we reach a CPT entry. Based upon the structure of the probability computation we adapt the value of the left hand side, modifying the values on the right hand side, i.e., $p_1 \uparrow = \frac{p_2 \uparrow}{p_3 \downarrow}$ or $p_1 \uparrow = p_2 \uparrow \cdot p_3 \uparrow$. The arrows indicate the direction of the adaptation. The current *step size* defines the value to increase or to decrease a CPT entry in every adaptation step. The constraint-based adaptation method is shown in Algorithm 2.

Algorithm 2 CSP-Solver for the constraint-based adaptation of the data generation model.

```
Require: Step size $step$, maximum number of iterations $k$
 1: Group constraints such that constraints referring to a common set of nodes (and subgroups) are grouped into one constraint group $G \in \mathcal{G}$.
 2: repeat
 3:     for all constraint groups $G \in \mathcal{G}$ do
 4:         for all constraints $c \in G$ do
 5:             Determine the direction of the adaptation $d$, according to the current constraint value and the specified (goal) value defined by the constraint $c$
 6:             for all operands $o$ of the right hand side of $c$, set the adaptation direction to $d$ do
 7:                 If $o$ is a CPT entry: Modify the value according to $d$ and step size $step$. Update the global improvement $I$ and revert the modification.
                    Otherwise, if $o$ is not a CPT entry, then apply step 6 recursively for the constraint $o$
 8:                 Re-apply the modification with the best improvement value $I$
 9: until all constraints are fulfilled or the maximum number of iterations $k$ has been reached.
```

To compute the global improvement of the constraints we first apply a localized error function $v(c)$ for a constraint c:

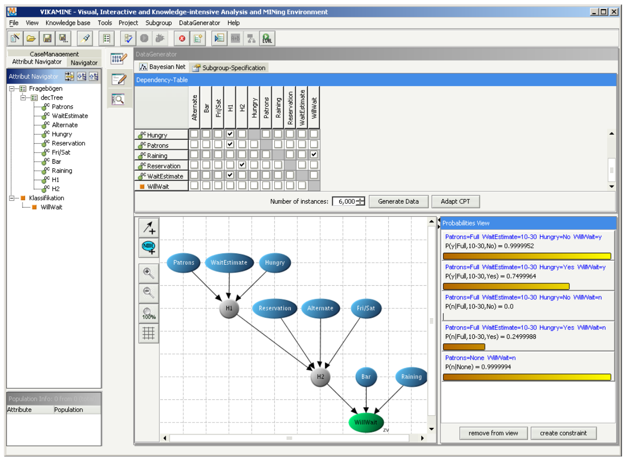
$$v(c) = (goal\ value(c) - current\ value(c))^2 \cdot priority(c)\,.$$

The sum of these local errors is compared to the global error in the previous adaptation step. An improvement can be measured, if $error_{previous} > error_{current}$. In consequence, we choose the adaptation with the maximum improvement value. Besides this hill-climbing algorithm other algorithms, e.g., simulated annealing, could be used alternatively. In our own work simulated annealing showed comparable results with respect to the hill-climbing approach.

5 Case Studies

In this section, we demonstrate the proposed approach by three case studies that consider different machine learning and data mining techniques and the corresponding process of data generation. We describe appropriate modeling guidelines in order to show how to use the given process model in an effective way. In the case studies we generate nominal data. However, generating numeric data is supported as well, since we can either generate data using discretized intervals in the Bayesian network for numeric variables, or common distributions (normal, uniform) can be specified for a node of the Bayesian network.

The process and the case studies were implemented with the data mining tool VIKAMINE [3], that was initially developed for the efficient and interactive mining of subgroup patterns but also offers an independent component for the generation of synthetic data. A screen shot of VIKAMINE is shown in Figure 3.



	Alternate	Bar	Fri/Sat	H1	H2	Hungry	Patrons	Raining	Reservation	WaitEstimate	WillWait
Hungry				✓							
Patrons				✓							
Raining											✓
Reservation					✓						
WaitEstimate				✓							
WillWait											

Patrons=Full WaitEstimate=10-30 Hungry=No WillWait=y
P(y|Full,10-30,No) = 0.9999952

Patrons=Full WaitEstimate=10-30 Hungry=Yes WillWait=y
P(y|Full,10-30,Yes) = 0.7499964

Patrons=Full WaitEstimate=10-30 Hungry=No WillWait=n
P(n|Full,10-30,No) = 0.0

Patrons=Full WaitEstimate=10-30 Hungry=Yes WillWait=n
P(n|Full,10-30,Yes) = 0.2499988

Patrons=None WillWait=n
P(n|None) = 0.9999994

Figure 3: Generating synthetic data with the data mining tool VIKAMINE . The left pane shows the used attributes, and the right pane shows an editor for the data generation model.

5.1 Association Rules

The first case study refers to the specification of data characteristics concerning association rule learners [1]. As shown in Section 3 we can easily describe the parameters *confidence* and *support* by specifying the constraints for the corresponding probabilities. For example, we can use the following association rule scenario from the well known *Wal-Mart* domain: *The husband and father usually bought the diapers at the end of the working week when beer often becomes a priority; and so beer became the product most often associated with the sale of diapers.* We can describe this relation by the rule: $Beer \Rightarrow Diapers$. The support of this rule is equivalent to the joint probability $p(Beer, Diapers)$ and the confidence corresponds to the conditional probability $p(Diapers \mid Beer)$. There are no explicit probability values given in our example, so we use some suitable values to associate the common relations, for example we set $p(Diapers \mid Beer) = 0.6$.

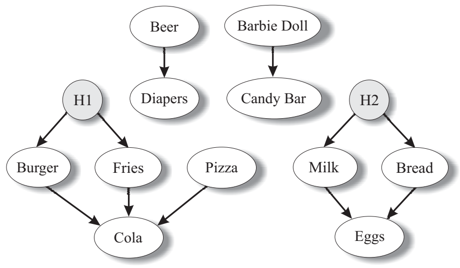


Figure 4: A Bayesian network representing Wal-Mart data for the generation of association rules.

After the specification of the basic constraints we refine the data generation model by the construction of the appropriate network structure. We can provide the attributes, that are a part of the rule conditions as parent nodes and the variable in the conclusion of the rule as the child node. Figure 4 shows a Bayesian network containing several subnetworks according to some specified association rules. When we create many of these patterns we have to set a low support in order to minimize the joint probability of different association rules. To generate a low support value we minimize the prior probability of the parent nodes, because all root nodes are independent. If a condition of an association rule contains more than one attribute, then we have to ensure that these attribute values frequently occur together. This can be obtained by adding a hidden node (e.g., H1 or H2) as a parent of the co-occurring attribute nodes. Additionally, we have to pay attention to the corresponding conditional probabilities, e.g., $p(Beer \mid Diapers)$ when we specify $p(Diapers \mid Beer)$, in order to avoid a high value for $p(Beer \mid Diapers)$. This example shows that adapting the CPT entries in an arbitrary way can lead to some "noisy" undesired dependencies between the attribute values.

5.2 Decision Trees

In the following, we provide a model for a decision tree [8]. Every path in a decision tree represents a categoric rule which predicts the class of an unclassified instance, for example $a_1 \wedge a_2 \wedge \ldots \wedge a_n \Rightarrow c$. To specify such a rule in our generation model we create a constraint which refers to the conditional probability $p(c \mid a_1, a_2, \ldots, a_n)$, where the a_i are the attribute values of the condition and c is the predicted class value.

We use the exemplary decision tree model from [9] to demonstrate how to specify the data generation model. The example describes the problem of whether to wait for a table at a restaurant given some evidential attributes like *hungry*, *raining*, and *alternate restaurant*. For example we have the rule *Patrons=full* $\wedge$ *WaitEstimate=10-30* $\wedge$ *Hungry=no* $\Rightarrow$ *WillWait=yes* or a simple one like *Patrons=none* $\Rightarrow$ *WillWait=no*. For each decision rule we specify a corresponding subgroup pattern, where the target variable is equal to the current class attribute *WillWait*. For categoric rules the conditional probability (target share) for a class value, given the rules condition variables as selectors, is always equal to the probability value 1. If we want to include noise into the generated data, then we use a lower probability value, e.g., 0.95, enforcing some misclassified instances.

In the next step we construct the Bayesian network: In the simplest case we can do this automatically to obtain a two-layer network with the class attribute as the child node (see Figure 2(b)). This can lead to a very large CPT, because of the exponential growth of the CPT entries according to the number of parent values. We avoid this by splitting the structure into several network levels connected by hidden-nodes. We can then use these hidden-nodes to create direct dependencies between root nodes and leaf nodes, e.g., *Patrons* and *WillWait*. The modeling process is finished by adapting the CPT entries

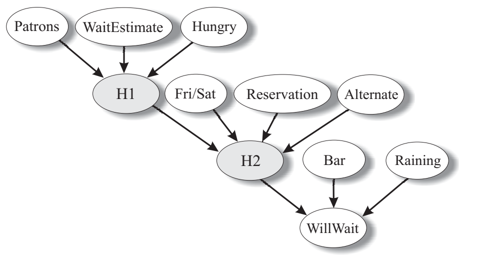


Figure 5: A Bayesian network to generate decision tree data.

according to the specified conditional probabilities in order to represent all rules that the decision tree includes. Figure 5 shows the Bayesian network for the restaurant domain.

5.3 Subgroup Patterns

The last use case refers to the evaluation of subgroup discovery algorithms [5]. Because we use subgroup patterns as the underlying knowledge representation for the data characteristics, we can build a generation model for subgroup patterns quite easily. For a given subgroup pattern we define the target share and the subgroup size as parameters. Similarly to the specification of association rules we create a Bayesian network that contains several subnetworks for each subgroup. In this example we want to specify some subgroup patterns for a simple insurance domain at which we focus on *insurance rate* as the main target variable.

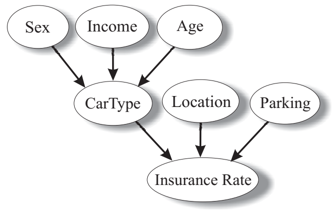


Figure 6: A Bayesian network describing subgroup patterns.

We use attributes like *Age*, *Sex*, *Income* and *CarType* to describe interesting subgroup patterns, for example: *Sex=m* $\wedge$ *Age=16-25* $\wedge$ *CarType=Sportscar* $\Rightarrow$ *InsuranceRate=high (90%)*, and

subgroup description	p_0	p	*size*
Sex=m $\wedge$ Income=high $\wedge$ Age=25-40	0.1	0.5	0.05
Sex=m $\wedge$ Income=very high $\wedge$ Age=25-40	0.1	0.8	0.05
Sex=w $\wedge$ Age=> 60	0.1	0.05	0.1

Table 1: Exemplary subgroup patterns for the target variable *CarType* = *Sportscar*

Sex=m $\wedge$ *Income=high* $\wedge$ *Age=25-40* $\Rightarrow$ *CarType=Sportscar (80%)*, as shown in Table 1 in more detail.

If a selector of one subgroup also occurs as a target variable in another subgroup, then we need to merge these two networks. In Figure 6 we show such a structure where the attribute *CarType* is a selector and also a target variable.

5.4 Evaluation

We tested different versions of the model and the generated data, respectively, using the data mining toolkit WEKA [11] for the decision tree and association rule data. The subgroup patterns were tested using the VIKAMINE toolkit. For each case study we were able to discover the modeled patterns in our experiments. Using the association rule and the subgroup data we measured minor deviations from the specified parameters that can be explained by the sampling algorithm. In Figure 7 a part of the decision tree is shown, that was learned by WEKA using the generated data.

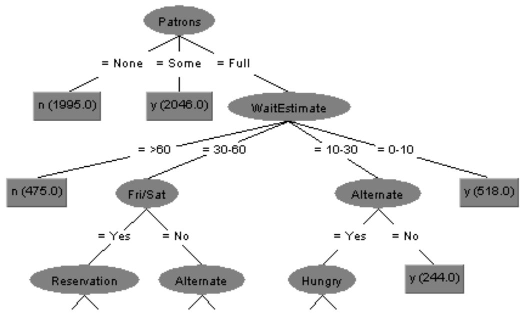


Figure 7: A part of the *restaurant* decision tree learned by WEKA based on the generated data.

In Table 2 we show the parameters of the discovered subgroups of the generated data, for an example dataset with 5000 instances. Compared to the patterns shown in Table 1 there are only minor deviations, due to the sampling-based data generation approach.

subgroup description	p_0	p	*size*
Sex=m $\wedge$ Income=high $\wedge$ Age=25-40	0.09	0.47	0.04
Sex=m $\wedge$ Income=very high $\wedge$ Age=25-40	0.09	0.79	0.03
Sex=w $\wedge$ Age=> 60	0.09	0.05	0.1

Table 2: Examples of discovered subgroup patterns for the target variable *CarType* = *Sportscar*

Furthermore, we applied the presented method in more extensive tests and we were able to generate multiple sets of valid evaluation data. The modeled patterns of interest were always correctly included in the generated data sets.

6 Discussion and Summary

In this paper, we presented a novel approach for the generation of synthetic data to be used for the evaluation of machine learning methods. We motivated that the availability of synthetic data sets with controllable characteristics and sizes is very important for a careful evaluation and comparison of new methods. The presented approach allows for a precise definition of the patterns included in the data sets to be generated and is therefore suitable to generate data sets at different levels of complexity. However, the concise specification of the data generation model is a difficult task using Bayesian networks as the underlying knowledge representation. For this reason, we introduced a process model in which the user first specifies simple subgroup patterns that are later mapped to a Bayesian network. Constraints are applied to guide the adjustment of the network that is finally applied for the data generation. This step can be performed interactively and full tool support is given.

With respect to related approaches, Salzberg [10] provides a detailed discussion of the use of synthetic data for the evaluation of machine learning methods: In order to reliably evaluate a (new) machine learning method in comparison with existing algorithms the characteristics of the data sets need to be considered. The UCI website [7] lists three simple data generators, i.e., the *quadruped animals data generator*, the *DGP/2* generator, and the *waveform data generator*. These generate data consisting of continuous attributes having randomly assigned values. So, the data generators provide no real control of the data generation process. Agrawal and Srikant [1] also describe a data generator to create synthetic transaction data. The generator is a special-purpose tool, and can only be used to generate data with specific parameters, e.g., by specifying the maximal size or the average size of the transactions. The SCDS/DATGEN [6] generator uses a limited domain representation to generate the data, including attribute types and additional constraints. However, no fine-grained specification of the relations between the attributes is possible. Of course, as a general data generator the *HUGIN* [2] system may be used. This system uses Bayesian networks, but no approximate specification of the network is possible as described in our approach. In consequence, the user is left with high knowledge acquisition costs.

In contrast to the approaches for generating synthetic data mentioned above the presented approach combines two central ideas from the SCDS and HUGIN systems: We can use *simple* patterns to develop the data generation model, but we can also use the advanced representation of Bayesian networks with instant consistency tests. The mentioned systems only use one of these mechanisms, i.e., rules (SCDS) or Bayesian networks (HUGIN). We combine both representations and provide the opportunity of an interactive adaptation of the data specification.

The presented approach was implemented within the data mining toolkit VIKAMINE [3]. We demonstrated the feasibility of the approach by three case studies that described the process for generating synthetic data for association rule learners, for decision tree learning, and for subgroup discovery. In our opinion, the presented approach is a promising step for improving the general quality of evaluating machine learning methods aiming for a standardized and transparent setting of test data. Furthermore the generation process can be used for complementing existing evaluation methodologies using real data sets.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. 20th Intl. Conf. on Very Large Data Bases*, pages 487–499, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann.

[2] S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. HUGIN: A Shell for Building Bayesian Belief Universes for Expert Systems. In *Readings in Uncertain Reasoning*, pages 332–337. Kaufmann, San Mateo, CA, 1990.

[3] Martin Atzmueller and Frank Puppe. Semi-Automatic Visual Subgroup Mining using VIKAMINE. *Journal of Universal Computer Science (JUCS), Special Issue on Visual Data Mining*, 11(11):1752–1765, 2005.

[4] Finn Verner Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, England, 1996.

[5] Willi Klösgen. *Handbook of Data Mining and Knowledge Discovery*, chapter 16.3: Subgroup Discovery. Oxford University Press, New York, 2002.

[6] Gabor Melli. SCDS/DATGEN Data Set Generator. www.datasetgenerator.com (currently unavailable - available via www.archive.org), 1995.

[7] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

[8] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2002.

[10] Steven L. Salzberg. On Comparing Classifiers: A Critique of Current Research and Methods. *Data Mining and Knowledge Discovery*, 1:1 – 12, 1999.

[11] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, 2000.

Contact

Martin Atzmueller
Department of Computer Science 6
(Applied Computer Science and Artificial Intelligence) University of Würzburg
e-mail: atzmueller@informatik.uni-wuerzburg.de

Bild

Martin Atzmueller studied computer science at the University of Würzburg and received his diploma in 2002. At present he is a research assistant at the Department of Artificial Intelligence and Applied Computer Science at the University of Würzburg. His research interests include knowledge-intensive data mining, text mining, visualization, and case-based reasoning.

Bild

Joachim Baumeister is a research assistant at the University of Würzburg since 1999. His research focuses on practical knowledge engineering techniques including validation, restructuring and semi-automatic learning of diagnostic knowledge.

Bild

Mario Goller studied computer science at the University of Würzburg and received his diploma in 2005. Currently, he works as a software-developer at Innovations Softwaretechnologie GmbH.

Bild

Frank Puppe is a full professor at the University of Würzburg since 1992. His research interests comprise all aspects of knowledge-based systems and their practical use. He has authored and co-authored 5 books and more than 100 articles on the topic.