Representation Schemes Used by Various Classification Techniques – A Comparative Assessment

Abdallah Alashqur
Applied Science Private University
Amman, JORDAN

Abstract

Data mining technology is becoming increasingly important and popular due the huge amounts of digital data that is stored globally. It provides methods and techniques to analyze these huge data repositories to extract useful information, which then is used to feed the decision making process. *Classification* is one of the data mining approaches to analyzing data. Other popular approaches are *association rule mining* and *clustering*. Various classification techniques have been identified in the literature including decision tree classification, rule-based classification, naïve Bayesian classification, Bayesian belief networks, and rule-based naïve Bayesian classification. One of the main differences between these classification techniques is the representation scheme used by each classification technique. A representation scheme captures the classification criteria and knowledge that a system learns from a pre-classified training set. In this paper we provide a comparative assessment of some these representation schemes and describes the advantages and disadvantages of each classification technique and its underlying representation scheme.

Key Words*: Database, Data Mining, Classification, Machine Learning.*

1. Introduction

With the advent of the Internet, world-wide-web, and mobile computing the volume of digital data stored on servers distributed globally has grown exponentially in recent years. It is becoming very challenging to analyze data that can be measured in Petabyte and even Exabyte in order to efficiently employ knowledge extracted from this data in decision support and other beneficial data processing activities. Classification, among other data mining methods, provides a way for intelligently handling and benefiting from such "Big Data" in decision support systems and other analytic processing.

Classification is concerned with being able to process and map data to appropriate classes where pieces of data that share common characteristics are classified together in one class. In structured data such as relational databases, each record in a dataset needs to be associated with one of the available classes. One column in the data set, usually called a *class label*, is used to identify the class name to which the record belongs. Normally there are few classes and each class is shared by many records in the dataset. The objective of classification is to be able to classify new records whose classes have not yet been known [1,2] in an automated fashion. This is normally achieved by providing the classifier with a pre-classified dataset, which is called *training set*. Machine learning techniques are used by the system to derive (i.e., *learn* or discover) the criteria based on which the records in the training dataset have been classified. Going forward, the system uses the learned criteria to classify new unclassified records [1,3,4]. The learned classification criteria are sometimes called *classification model*. A representation scheme is needed to capture and represent the classification model.

Overall, the classification process passes through two steps. The first step is called the learning step, in which the classification model is built by learning from the training set. After the system constructs the classification model, and after performing some testing, the system becomes ready for the second step. In the second step, which can be called the application step, the classifier applies the classification model that it learned to new records in order to predict their classes. The training set contains records whose classes are known. A training set can be classified manually by giving unclassified records to experts in the application domain. Those experts can then classify the records based on their expert knowledge. Another way to build the training set is to use historical data whose classes have become known facts. For example, in banking loan system, the historical data can show who of the loan applicants actually paid back their loan and who did not. The dataset contains various data about applicants. A column named, say, "Loan Payment", which shows who paid and who did not pay back the loan, can be the class label. In any way, the level of correctness and trustworthiness of classifications in the training set should be extremely high because it is used to train the classification software during the learning step.

Classification techniques differ in the representation scheme they use to represent the classification model. Many classification techniques have been described in the research literature. Examples of these techniques include decision tree classifiers [5,6,7,8], rule-based classifiers [9,10], naïve Bayesian classifiers [11,12], rule-based naïve

Bayesian classifiers [13], and Bayesian Belief Networks [14,15,16]. In decision tree classifiers, the learned model is represented as a decision tree. A rule-based classifier, on the other hand, represents the derived classification model as a collection of rules where each rule has few conditions and a consequence. The consequence of each rule is the class to which the record belongs. Bayesian classifiers are considered to be statistical classifiers in which the classification model is represented by a set of equations that are based on Bayes' theorem. Similarly, each classification technique has its own representation scheme for representing the learned model.

In this paper we briefly describe some popular classification techniques and their underlying representation schemes. We also compare the advantages and disadvantages of these various representation schemes. We mainly focus on the following classification techniques: decision tree classification, rule-based classification, naïve Bayesian classification, rule-based naïve Bayesian classification and Bayesian belief networks. Section 2 of this paper provides a brief description of each of these classification methods. In Section 3, we provide a comparison of the advantages and disadvantages of these techniques. Conclusions are given in Section 4.

2. Classification Methods

In this Section we explore some classification methods. For each classification method, we provide a brief description followed by an example. The example shows how the method is applied to sample data and gives an idea about the complexity associated with each classification method. Section 2.1 describes Decision Tree classification, Rule-Based classifiers are described in Section 2.2. In Section 2.3, Naïve Bayesian classification is explained. A new classification method called Rule-based Naïve Bayesian Classification (RNBC) is described in Section 2.4. And finally, in Section 2.5, we briefly show how Bayesian Belief Networks are used as classifiers.

2.1 Decision Trees

Decision trees represent a powerful classification technique. Several algorithms exist for building decision trees from training datasets. For example: ID3 (Iterative Dichotomizer 3), CART (Classification and Regression Tree), and C4.5 [1,2,8,17]. One of the differences between these algorithms is the way attributes are chosen as splitting nodes in the tree.

Table 1 shows a training dataset that can be used to build a decision tree We will use the sample data shown in Table 1 to demonstrate how decision trees are used to classify data (we will also use this dataset in later sections to demonstrate how other classification techniques work). The class label in Table 1 is Loan Worthy (LW), which indicates whether a client can or cannot be given a loan based on the given data about the client. The data used to determine loan worthiness are shown in the rest of the columns in the dataset. These are Age (A), Marital Status (M), Home Owner (HO), and Gender (G). The values in the attribute Age are "junior" for ages 22 to 35, "middle" for ages 35 to 50, and "senior" for ages greater than 50. As an example, the first client (that is represented by the first row in the dataset) has the following values for the various attributes: A = junior, M = yes, S = high, HO = yes, and G = M. Based on this data, the decision shown in the LW column is "yes". In other words, based on the given data the client is classified as a person who can be give a loan because most likely he will pay it back. This is based on past experience made by the bank.

Table 1: Class-Labeled Training Records

Age (A)	Married (M)	Salary (S)	Home Owner (HO)	Gender (G)	Loan Worthy (LW)
junior	yes	high	yes	M	**yes**
middle	no	low	yes	M	**no**
senior	no	low	no	M	**no**
senior	no	low	yes	M	**no**
middle	yes	high	yes	F	**yes**
junior	no	high	yes	F	**yes**
junior	yes	low	yes	F	**yes**
middle	yes	high	no	F	**yes**
middle	no	low	no	M	**no**
junior	no	low	no	M	**no**
junior	no	high	yes	M	**no**
senior	yes	low	no	F	**yes**
middle	yes	high	yes	F	**yes**
junior	no	high	yes	F	**yes**
junior	no	high	no	F	**no**
senior	yes	high	yes	F	**yes**
senior	yes	high	no	F	**yes**

In the decision tree construction process, a classification algorithm produces the decision tree based on class-labeled training records (in this paper we use the words "record" and "tuple" interchangeably) shown in Table 1. In a decision tree diagram, each non-leaf node corresponds to a test on an attribute and each edge represents an outcome of the test. If the outcome of a test consists of records that belong to more than one class (i.e., a mixed set of records), further tests are needed to complete the classification along that branch by adding more non-leaf nodes (i.e., test nodes) to the tree. Alternatively, if the outcome of a test consists of records that belong to only one class, a leaf node is added to indicate that the records that satisfy the conjunction of the tests along the path from the root node to the leaf node are placed in one class.

To build a decision tree from training dataset an attribute selection measure is needed. An attribute selection measure is used to make optimal selection of the attribute that best separate the records into distinct classes. This is to insure

we obtain a compact tree in the end. Several such measures have been reported in the literature. In this paper we use one of those measures, namely, Entropy (also called Information Gain). This measure is used by the algorithm ID3 [2]. At each node in the decision tree the classification algorithm selects the attribute with the highest information gain as the splitting attribute.

To find which attribute has the highest entropy, a three-step process is applied. The Function *attribute_selection* shown below incorporates those three steps in the form of three equations (1), (2), and (3). In the first equation, the information needed to classify a record in the dataset D is computed. This equation handles only the distribution of records over classes. Equation 2 examines attribute values in order to measure the impact of each attribute on the class.

Function attribute_selection

```
/* Compute the information needed in
order to classify a record in the
dataset D. */
```

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad (1)$$

```
/* in the above equation i is the
class, pi is the probability a record
falls in the ith class. */
```

FOR EACH *attribute X in the dataset other than the class label attribute DO*

```
/* find the information needed (after
using X to split the dataset D into v
partitions) to classify the dataset D.
each Dj is a partition of the dataset
D. |D| is the number of the rows in
the dataset or partition. */
```

$$Info_X(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \quad (2)$$

```
/* Information gained by branching on
attribute X can be obtained by
subtracting the result of Equation 2
from the result of Equation */
```

$$Gain(X) = Info(D) - Info_X(D) \quad (3)$$

END FOR EACH Loop
EndFunction

Equation 1 of the algorithm is applied to the dataset only one time at the very beginning. Equation 2 and Equation 3 are computed for each attribute. The attribute that results in the highest gain as given by Equation 3, is chosen as the splitting attribute. The function is repeated at each level of the tree until leaf nodes are reached. Leaf nodes represent records that belong to only one class.

To apply Equation 1 to the dataset of Table 1, we notice that there is a total of seventeen records, ten of them have class label of "LW = yes" and seven of them are "LW = no". Substituting in Equation 1, we obtain:

Info (D) = – 7/17 log2 (7/17) - 10/17 log2(10/17)
= – 0.412* - 1.280 - 0.588* - 0.675
= 0.997 bits

The *FOR EACH* loop of the algorithm computes the information gain corresponding to each attribute by using Equations (2) and (3). To demonstrate, we show below how these two equations are used to compute the information gain corresponding to the attribute Age.

For Age = junior, there are four records whose class label is "yes" and three records whose class label is "no". For Age = Middle, there are three "yes" records and two "no" records. For Age = senior, there are three "yes" records and two "no" records. The expected information needed to classify a record in the dataset of Table 1 if the records are partitioned according to Age can be obtained by substituting in Equation 2 as shown below.

$Info_A$ (D) = 7/17*(– 3/7 $\log_2$ (3/7) -4/7 $\log_2$ (4/7)) +
5/17 * (– 2/5 $\log_2$ (2/5) - 3/5 $\log_2$ (3/5)) +
5/17* (– 2/5 $\log_2$ (2/5) - 3/5 $\log_2$ (3/5))
= 0.975 bits

Subsequently, the information gain is obtained by substituting in Equation (3) as shown below.

Gain (A) = 0.977 – 0.975 = 0.002

Similarly the FOR-EACH loop computes the Gain for the remaining attributes. The final values of the Gain for these attributes are:

Gain (M) = 0.57, Gain (S) = 0.20,
Gain (HO) = 0.05, Gain (G) = 0.46

Based on the above computations, attribute Married (M) provides us with the highest information gain. Therefore we start with is as the first node (root node) in the tree as shown in Figure 1. The left branch (identified by "M = yes") stemming from that node results in homogeneous classification since all records whose "M = yes" belong to the class LW = yes. Therefore no further splitting nodes are needed along that branch. On the other hand, the right branch in which "M = no," results in inhomogeneous classification since some records are classified as "LW = no" and other records "LW = yes." Therefore we need to identify another attribute from the remaining set of

attributes to branch on. We do that by re-applying the *attribute_selection* function to find the attribute with the highest Information Gain. However, this time we apply it to the subset of records that satisfy the condition "M = no". We exclude Married from the set of attributes used by the function since this attribute has already been used in the decision tree of Figure 1.

If we repeat this process by re-applying the attribute_selection function every time there is a branch with inhomogeneous classification, we will end up with the decision tree shown in Figure 1. A leaf node is labeled with "yes" or "no" to indicate the class to which these records belong. For example the path that starts at the root and goes through the edges identified by conditions (Married = no), (Age = junior), and (Gender = M) ends up at the leaf node labeled with a 'no'. This means that a record that satisfies these three conditions is predicted to belong to the class "LW = yes".

After the decision tree is build, it becomes ready to be used to classify new records. Assume we have the new record <A=junior, M=no, S=low, HO=yes, G=female>. According to the decision tree of Figure 1, we start with the value of M. Since "M = no" in the new record, we branch to the Age node. Because "Age= junior", we branch to the Gender node. Next since "G= f" in the new record, we branch to the Home Owner node. Finally, since "HO = yes" in the new record we branch to the node labeled "yes." In other words, a classification system would predict that this record is classified as "LW = yes" by tracing the decision tree.

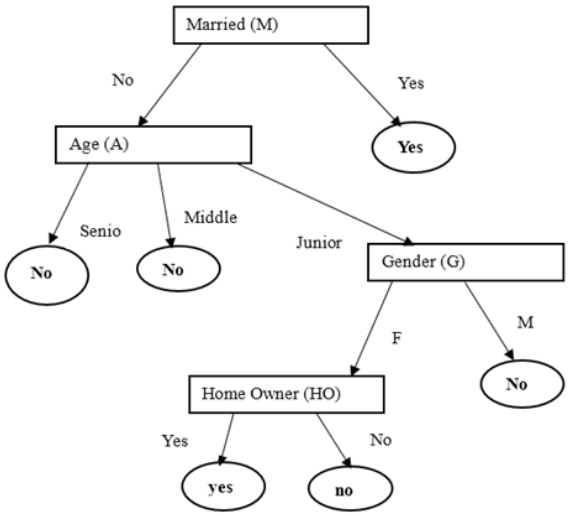

Figure 1: Decision Tree for the data of Table 1

2.2 Rule-Based Classifiers

In general, there are two methods for constructing a rule-based classifier. The first method is called Direct Method. The classification rules, in this method, are *directly* derived from the training dataset. A class of algorithms called sequential covering algorithms use this method. A classification rule in sequential covering algorithms are created sequentially, that is, one at a time. Each time a rule is created, the records that satisfy the rule are taken out from the dataset, and the process is repeated on the records that remain. RIPPER [18] is an example algorithm that follows the direct method. The second method is called *Indirect* Method. The rules in this approach are not derived directly from the data set, but they are extracted from another classification technique similar to decision trees. An example algorithm that uses this method is C4.5 [18]. In C4.5 algorithm, a decision tree is created first, then a set of classification rules are created based on the decision tree.

A classification rule is of the form:

R_i: *(Condition*$_1$ *AND Condition*$_2$, *... AND Condition*$_n$*)* → *Class*$_x$

Which means that if the attributes of a new record satisfy the conditions: $Condition_1$, $Condition_2$, …, AND $Condition_n$ then it is classified as $Class_x$.. R_i is the rule-ID. The left hand side of a rule is a conjunction of *conditions* on some or all of the attributes in the dataset except the attribute that represents the class label. The right hand side is the class, which in our example is either "LW = yes" or "LW = no".

Using the indirect method, where classification rules can be derived from a decision tree, we show three of the rules that can be derived from the decision tree of Figure 1.

R1: (M = no) AND (Age = Junior) AND (G = f) AND (HO = yes) → LW = yes

R2: (M = no) AND (Age = Junior) AND (G = f) AND (HO = no) → LW = no

R3: (M = no) AND (Age = Junior) AND (G = m) → LW = no

If the attribute values of a record satisfy the conditions of a rule R, the rule is said to *cover* the record. Hence rule R3 covers the 10th and 11th records in the dataset of Table 1. In addition, a record is said to *satisfy* a certain rule if all the conditions in the left hand side of the rule are satisfied by the record's values. The rule coverage is defined as the percentage of tuples in the dataset that satisfy the rule [2]. This can be expressed by the following equation.

$$Coverage \text{ of rule } (R) = \frac{Number\ of\ tuples\ satisfying\ R}{Total\ number\ of\ tuples}$$

Substituting in this equation for rule R3 above, we obtain: Coverage of rule (R3) = 2/17 = 0.18.

If no record satisfies more than one rule, the rules are said to be *mutually exclusive*. The set of rules is said to be *exhaustive* if each tuple in the dataset is covered by at least one rule. If a record is satisfied by multiple rules, a conflict

resolution method is normally employed to select the rule to apply [18]. One of the resolution methods is to assign weights to the rules, and the rule with the highest weight is applied.

2.3 Naïve Bayesian Classification

Naïve Bayesian classification [2,19] is a classification technique that is based on Bayesian Theorem. If a new tuple T is to be classified, Bayesian Theorem is used to compute the probability that it belongs to class C_i by using Equation (1) as shown below.

$$P(C_i|T) = \frac{P(T|C_i)P(C_i)}{P(T)} \quad (1)$$

In the above equation *P* denotes *probability* and the notation $P(T|C_i)$ represents the conditional probability of T given that the class is known to be C_i. C_i is a class that belongs to the set of classes *{ C1, C2, C3, ...}* for the dataset. Equation (1) is computed for every one of these classes and the class whose $P(Ci|T)$ is largest is identified as the tuple's class. When computing $P(C_i|T)$ for every C_i, the denominator $P(T)$ is constant across all classes. Hence it can be removed from the equation. Thus, Equation (2) below can be used to find the class that has the highest probability.

$$P(C_i|T) \sim P(T|C_i)P(C_i) \quad (2)$$

Where the symbol "~" denotes that the left hand side is *proportional* to the right hand side.

Naïve Bayesian classification is based on the assumption of *class-conditional independence* (this is the reason it is called "naïve"). This assumption basically means that attribute values of the tuple *T* are independent of each other. As a consequence, if *T* is the n-tuple $<t_1, t_2, \ldots t_n>$, then $P(T|Ci)$ in Equation (2) above can be computed by using Equation (3) below.

$$P(T|C_i) = \prod_{k=1}^{n} P(t_k|C_i) = P(t_1|C_i) \times P(t_2|C_i) \times \ldots \times P(t_n|C_i) \quad (3)$$

The justification for Equation (3) is based on probability theory, where the *joint* probability of *independent* events can be computed by multiplying the probabilities of these events. Therefore, to compute $P(C_i|T)$ based on Equation (2) we need to compute $P(C_i)$ and compute $P(T|C_i)$ based on Equation (3) and multiply the two results. This is performed for each class C_i and the class with the highest value is chosen as the class for the new tuple T.

As an example, assume a new tuple T for a new loan applicant is inserted in Table 1. The values of the attributes for the new tuple T are: ("junior", "no", "low" "no", "M"). These values are in the same order as the table's attributes. Below we use Naïve Bayesian classification to predict the value of the class label attribute Loan Worthy (LW).

There are two classes as identified by the class label, namely LW = "yes" and LW = "no".
Substituting in Equation (2) for each of the two classes, we get:

P (LW = "yes") ~ P (T| LW = "yes") P(LW = "yes") (4)
P (LW = "no") ~ P (T| LW = "no") P(LW = "no") (5)

In the first step below, we compute P(LW = "yes") and P(LW = "no") used in Equations (4) and (5).

P(LW = "yes") = 10/17 = 0.59
P(LW = " no") = 7 / 17 = 0.41

In the second step below, we compute P (T| LW = "yes") and P (T| LW = "no") by substituting the individual probability values for each attribute value into Equation 3.

P (A = 'junior'| LW = 'yes') = 4/10 = 0.4
P (M = 'no'| LW 'yes') = 2/10 = 0.2
P (S = 'low' | LW = ' yes') = 2/10 = 0.2
P (HO= 'no' | LW = 'yes') = 3/10 = 0.3
P (G ='m' | LW = ' yes ') = 1/10 = 0.1

P(T| LW = "yes") = 0.4 * 0.2 *0.2 *0.3*0.1 = 0.00048

P (A= 'junior' | LW = " no ") = 3/7 = 0.429
P (M = ' no' | LW = " no ") = 7/7 = 1
P (S = ' low' | LW = " no ") = 5/7 = 0.714
P (HO= ' no' | LW = " no") =4/7 = 0.571
P (G='m' | LW " no ") =6/7 = 0.857

P(T| LW = " no ") = 0.429*1*0.714*0.571*0.857 = 0.14988

In the final step, we perform the multiplication represented by Equations (4) and (5) to obtain the final result.

P (LW = "yes") ~ P(T| LW = " yes") * P(LW = "yes")= 0.0005* 0.59 = 0.000282

P (LW = "yes") ~ P(T| LW = "no") * P(LW = "no") = 0.1499 * 0.41 = 0.0616

Since P (LW = "no") > P (LW = "yes"), the class of the new row is identified as LW = "no".

2.4 Rule-based Naïve Bayesian Classifier (RNBC)

In naïve Bayesian classification, every time a tuple is to be classified, the whole dataset should be scanned in order to apply a set of statistical equations. The final result of these

equations determines the class to which the record should belong. If the dataset is very large as often is the case, the need to scan the whole dataset whenever a new record is inserted is considered a disadvantage from a performance perspective. This is because of the high cost of scanning a large dataset. To alleviate this problem, a new approach was introduced in [13] in which a probabilistic model based on Naïve Bayesian classification is used for building a set of classification rules. This approach is called Rule-based Naïve Bayesian Classification (RNBC).

A three step-methodology is used in RNBC to build the set of classification rules [13]. The Bayesian equations are used only at the beginning of the process in order to create the set of classification rules. These rule cover all possible classification cases. Once this rule set is compiled, any new record can be classified by searching the set of rules to find the rule that is *satisfied* by the attribute values of the record. That rule is then applied to infer the record's classification. In other words, the Bayesian equations don't have to be applied against a large dataset every time a new record is to be classified. The three-step methodology for building such a rule-based classifier [13] is summarized below.

Step 1. Generate records that contain all possible combinations of attribute values that exist in the dataset.
Step 2. For each generated record, compute the probability of each class.
Step 3. Generate the classification rules, one rule for each generated record. The class identified by each rule is the class with the highest probability.

We can demonstrate how those three steps work by showing an example based on Table 1. The following are the set or permissible values for each attribute (i.e., the domain of the attribute).

Age = {junior, middle, senior}, Married = {yes, no},

Salary = {high, low}, Home Owner = {yes, no},
Gender = {male, female}

The number of permissible values for each attribute is shown below.

|Age| = 3, |Married| = 2, |Salary | = 2
|Home Owner| = 2, |Gender| = 2

The total number of records that represent all possible combinations of attribute values can be computed by multiplying the number of permissible values in each attribute as shown below.

Total Number of possible records = 3 * 2 * 2 * 2 * 2 = 48

Step 1 in the methodology generates the 48 possible records. In step 2, the system applies Naïve Bayesian classification equations to each of the 48 records just as shown in Section 2.3 of this paper. For instance, one of those 48 records is the one shown in the example of Section 2.3, which is the record <"junior", "no", "low" "no", "M">. The computations performed in Section 2.3 concluded that the class of this record is LW = "no". Similar computations are performed in Step 2 for the remaining 47 records to determine their classes.

In Step 3, the results of Step 2 are summarized in the form of IF-Then Rules. For example, the rule that corresponds to the record <"junior", "no", "low" "no", "M"> is shown below.

IF (A = junior) AND (M = no) AND (S = Low) AND (HO = no) AND (G = Male) **THEN** LW = "no".

From that point on, whenever a new record is to be added, all the system has to do is to find the rule whose conditions are satisfied by the data in the new record. That rule is *fired* to infer the class.

2.5 Bayesian Belief Networks (BBN)

Unlike Naïve Bayesian classifier, a BBN does not assume that attributes in a dataset are mutually independent [16]. The dependencies between attributes are represented in the form of a directed acyclic graph (dag), with nodes representing attributes and edges between nodes represent dependencies [14,15,16]. Figure 2 shows a BBN in which six attributes from the medical domain are represented. These attributes are Sodium Intake (SI), Obesity (O), High Blood Pressure (HBP), Diabetes (D), Headache (H), and Eye Disease (ED).

A node in a BBN is assumed to be conditionally independent of all of its non-descendants if its parents are known [15,16]. A probability table associating each node to its immediate parent is added to each node. The table is called conditional probability table (CPT). These tables are shown in Figure 2. For example, the first row in the CPT associated with High Blood Pressure (HBP) node indicates that P(HBP| SI=yes AND O = yes) = 0.8. The CPT associated with nodes that have no parents such as SI represent the prior probabilities.

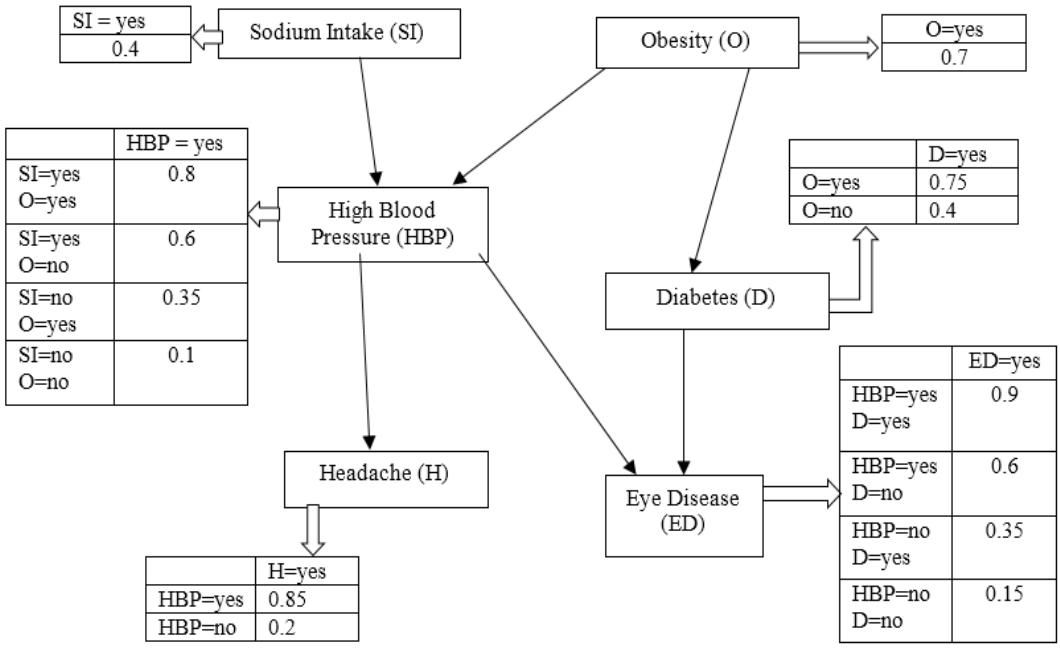

Figure 2. A Bayesian Belief Network from the Medical Field

Computations using BBN normally use the probability Chain Rule. This rule can be stated as follows. Let E1, E2, E3, .. En be different events that are not necessarily independent, then

P(E1,E2) = p(E1|E2) p(E2)
We can extend this to three events:
P(E1, E2, E3) = P(E1| E2, E3) P(E2, E3)
= P(E1| E2, E3) P(E2| E3) P(E3)

and in general, the chain rule for n variables is represented as follows:
P(E1, E2, ..., En) = P(E1| E2, ..., En) P(E2| E3, ..., En) P(En-1|En) P(En)

The chain rule is used for finding Joint probability in terms of conditional probabilities. When applying this rule to a BBN we need to take into consideration the assumption that a node in a BBN is conditionally independent of all of its non-descendants, given its parents. Consider the BBN shown in Figure 3 with nodes V, W, X, Y, and Z

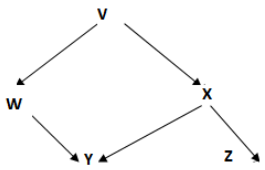

Figure 3: Dependencies between variables

Based on the dependencies that exist in Figure 3, we can perform the following simplifications to the probability of conjunction of events.

P(V, W, X, Y, Z) = P(Z | V, W, X, Y) P(V, W, X, Y)
/*from probability properties */
= P(Z | X) P(V, W, X, Y)
/* Z is independent of V,W,Y. therefore we can omit them.*/
= P(Z | X) P(Y | V, W, X) P(V, W, X)
= P(Z | X) P(Y | W, X) P(X | V, W) P(V, W)
= P(Z | X) P(Y | W, X) P(X | V) P(W | V) P(V)

Below we show how computations can be performed for three different cases based on the BBN and the CPT tables of Figure 2.

FIRST: Given the BBN shown in Figure 2, the computations below find P(HBP = yes).

P(HBP = Yes) = $\sum_{x\in\{Yes,No\}} \sum_{y\in\{Yes,No\}}$
P(HBP=Yes | SI=x, O=y) * P(SI = x, O=y)
= $\sum_{x\in\{Yes,No\}} \sum_{y\in\{Yes,No\}}$ P(HBP=Yes | SI=x, O=y) * P(SI=x) * P(O=y)

= P (HBP=Yes | SI=Yes, O=Yes) P (SI=Yes) P (O=Yes) +
P (HBP=Yes | SI=Yes, O=No) P (SI=Yes) P (O=No) +
P (HBP=Yes | SI=No, O=Yes) P (SI=No) P (O=Yes) +
P (HBP=Yes | SI=No, O=No) P (SI=No) P (O=No)

= (0.8 * 0.4 * 0.7) + (0.6 * 0.4 * 0.3) + (0.35 * 0.6 * 0.7) + (0.1 * 0.6 *0.3) = 0.46

SECOND: Given the BBN shown in Figure 2, the following shows how P(HBP = yes | H = Yes) can be computed.

$P(H = Yes) = \sum_{x \in \{Yes,No\}} P(H=Yes \mid HBP=x) * P(HBP = x)$
$= (0.85 * 0.461) + (0.2 * 0.539)$
$= 0.49965$

$P(HBP= Yes \mid H = Yes) = \frac{P(H = Yes \mid HBP = Yes) * P(HBP = Yes)}{P(H = Yes)}$
$= \frac{0.85*0.461}{0.49965} = 0.784$

THIRD: Below we compute P(HBP = yes | H = Yes, SI = yes, O = yes) based on the BBN shown in Figure 2.

P(HBP = Yes | H = Yes, O = Yes, SI = Yes)

$= \frac{P(H= Yes \mid HBP=Yes, O = Yes, SI = Yes)}{P(H = Yes \mid O = Yes, SI = Yes)} * P(HBP = Yes \mid O = Yes, SI = Yes)$

$= \frac{P(H = Yes \mid HBP = Yes)\ P(HBP = Yes \mid O = Yes, SI = Yes)}{\sum_{x \in \{Yes,No\}} P(H=Yes \mid HBP=x)\ P(HBP=x \mid O=Yes, SI=Yes)}$

$= \frac{0.85 * 0.8}{0.85 * 0.8 + 0.2 * 0.2} = 0.94$

In the above computations, BBN allows us to predict the values of some attributes given that we know values of some other attributes.

3. Comparison of classification techniques

In this section we provide a comparison between the different classifications techniques discussed in this paper. Table 2 shows the main advantages and disadvantages of each classification technique and its underlying representation scheme.

Table 2: A comparison between the various classification techniques

Classification method	Advantages	Disadvantages
Decision Tree	• Easy to construct with the exception of the attribute selection problem. • Easy to use at the time of classifying new records by simply traversing the tree based on the record's values.	• Requires complex computations to select the branching attribute at each node of the tree. • The tree can be very large in situations when the number of attributes and distinct values is high.
Rule-Based	• IF-THEN rules are easy to understand by humans. • Rule based systems represent a mature technology that can be used in this type of classification.	• Constructing the rules is not straight forward since it can be direct or indirect. • Needs a resolution technique when multiple rules apply to a new record.
Naïve Bayesian	• Based on simple probability principles • It can be used as a standard to compare other classification methods.	• It ignores interdependencies that may exist between attributes. This is why it is called "Naïve". • Computations have to be repeated every time there is a new record whose class needs to be discovered.
Rule-based Naïve Bayesian Classification.	• It combines techniques of both Rule-Based approach and the Naïve Bayesian classification approach. • It overcomes the main shortcoming of Naïve Bayesian Classification of having to perform all computations every time a new record is to be classified.	• It is a relatively new approach that yet needs to be tried in real life classification problems. • The number of rules can be very large if the number of distinct values in each column in the training set is large.
Bayesian Belief Networks	• It does take into consideration the interdependencies that may exist among attributes, unlike Naïve Bayesian classification. • As a consequence, its classification and prediction is very precise given that the conditional probability tables (CPT) are precise.	• The computations can be very lengthy and complex. • It is hard to build the conditional probability tables and to ensure that they contain precise values. The BBN may have to be trained.

4. Conclusions

We showed in this paper that each classification technique has an underlying representation scheme used to represent and capture the knowledge about how it should classify new records. The underlying representation scheme for a decision tree classifier for example is the tree structure. The classification knowledge and criteria is embedded in the tree as nodes and branching conditions. Similarly in Naïve Bayesian classifier, the classification knowledge is captured in a set of classification equations that the system use to classify new records. We have explored and compared several important classification techniques by briefly describing the classification technique followed by a simple example. Though the main difference between these classification techniques is the representation scheme used to represent the knowledge it learns from the training set, other differences exist. For example they differ in the way the model is constructed. Because the models are different, they consequently differ in the way they are used by the system to classify new records. In the comparison we provided between these classification techniques, the focus was on the main advantages and disadvantages of each classification technique and its underlying representation scheme.

Acknowledgement

The author is grateful to the Applied Science Private University, Amman, Jordan, for the full financial support granted to this research project (Grant No. DRGS-2015-2016-4).

References

[1] Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A., A Survey of Evolutionary Algorithms for Decision-Tree Induction. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, n. 3, p. 291-312, May 2012.

[2] Han, J., M. Kamber and J. Pei Data Mining: Concepts and Techniques, 3rd Ed. 2011. Morgan Kaufmann.

[3] M. Basgalupp , A. de Carvalho , R. C. Barros , D. Ruiz and A. Freitas "Lexicographic multi-objective evolutionary induction of decision trees", Int. J. Bio-Inspired Comput., vol. 1, no. 1–2, pp.105 -117 2009

[4] M. J. Aitkenhead "A co-evolving decision tree classification method", Expert Syst. Appl., vol. 34, no. 1, pp.18 -25, 2008

[5] Z. Fu , B. L. Golden , S. Lele , S. Raghavan and E. Wasil "Diversification for better classification trees", Comput. Oper. Res., vol. 33, no. 11, pp.3185 -3202 2006

[6] R. C. Barros , D. D. Ruiz and M. Basgalupp "Evolutionary model trees for handling continuous classes in machine learning", Inf. Sci., vol. 181, pp.954 -971 2011

[7] M. Kretowski and M. Grzes "Mixed decision trees: An evolutionary approach", Proc. Int. Conf. Data Warehousing Knowl. Discov., pp.260 -269 2006

[8] J. Gray and G. Fan "Classification tree analysis using TARGET", Comput. Statist. Data Anal., vol. 52, pp.1362 -1372, 2008

[9] D. Kalles and A. Papagelis "Lossless fitness inheritance in genetic algorithms for decision trees", Soft Comput.—Fusion Found., Method. Appl., vol. 14, pp.973 -993 2010

[10] A. Freitas , D. C. Wieser and R. Apweiler "On the importance of comprehensible classification models for protein function prediction", IEEE/ACM Trans. Comput. Biol. Bioinformat, vol. 7, no. 1, pp.172-182, 2010

[11] C. Bratu , C. Savin and R. Potolea "A hybrid algorithm for medical diagnosis", Proc. Int. Conf. Comput. Tool (EUROCON), pp.668 -673 2007

[12] C. To and T. Pham "Analysis of cardiac imaging data using decision tree based parallel genetic programming", Proc. 6th Int. Symp. Image Signal Process. Anal., pp.317 -320 Sep. 16–18, 2009

[13] A. Alashqur, "A Novel Methodology for Constructing Rule-Based Naïve Bayesian Classifiers" International Journal of Computer Science & Information Technology (IJCSIT), Vol 7, No 1, Pages 139-151, February 2015.

[14] Nee O and Hein A, "Clinical Decision Support With Guidelines And Bayesian Networks," in Decision Support Systems Advances. In: Devlin G, Editor INTECH, 2010. pages 117-136

[15] R. Daly, Q. Shen, S. Aitken, "Learning Bayesian Networks: Approaches And Issues" The Knowledge Engineering Review, 26 (2) (2011), pages 99–157

[16] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, first edition, 2012. ISBN-13: 978-0521518147.

[17] Vikas Chaurasia, Saurabh Pal," Early Prediction of Heart Diseases Using Data Mining Techniques, Carib.j.SciTech, Vol.1, 208-217, 2013.

[18] Tan, P-N, M. SteinBach, and V. Kumar, 2005. Introduction to Data Mining, Addison Wesley. ISBN-10: 0321321367.

[19] 19 Mozina, M., J. Demsar, M. Kattan and B. Zupan, 2004. Nomograms for visualization of naive Bayesian classifier. Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, September 20-24, 2004, Pisa, Italy, pp: 337-348.

Abdallah Alashqur obtained his Masters and Ph.D. degrees from the University of Florida in 1985 and 1989, respectively. After obtaining his Ph.D. degree, Dr. Alashqur worked in the IT Industry (in the USA) for seventeen years before returning to academia. Currently he is an associate professor in the Faculty of Information Technology at the Applied Science University in Amman, Jordan. His research interests include Data Mining and Database Systems.