Improving Supervised Learning by Feature Decomposition

Oded Maimon and Lior Rokach

Tel-Aviv University
Department of Industrial Engineering
Ramat Aviv, Tel Aviv 69978, Israel

email: maimon@eng.tau.ac.il liorr@eng.tau.ac.il

Abstract. This paper presents the Feature Decomposition Approach for improving supervised learning tasks. While in Feature Selection the aim is to identify a representative set of features from which to construct a classification model, in Feature Decomposition, the goal is to decompose the original set of features into several subsets. A classification model is built for each subset, and then all generated models are combined. This paper presents theoretical and practical aspects of the Feature Decomposition Approach. A greedy procedure, called DOT (Decomposed Oblivious Trees), is developed to decompose the input features set into subsets and to build a classification model for each subset separately. The results achieved in the empirical comparison testing with well-known learning algorithms (like C4.5) indicate the superiority of the feature decomposition approach in learning tasks that contains high number of features and moderate numbers of tuples.

1 Introduction and Motivation

Supervised Learning is one of the most important tasks in knowledge discovery in databases (KDD). In supervised problems the induction algorithm is given a set of training instances and the corresponding class labels and outputs a classification model. The classification model takes an unlabeled instance and predicts its class. The classification techniques can be implemented on variety of domains like marketing, finance and manufacturing.

Fayyad et al. (see [13]) claim that the explicit challenges for the KDD research community is to develop methods that facilitate the use of data mining algorithms for real-world databases. One of the characteristics of a real world databases is high volume. The difficulties in implementing classification algorithms as-is on high volume databases derives from the increase in the number of records in the database and from the increase in the number of features or attributes in each record (high dimensionality). High numbers of records primarily create difficulties in storage and computing complexity. Approaches for dealing with high number of records include sampling methods; massively parallel processing and efficient storage methods. However high dimensionality increases the size of the search space in an exponential manner, and thus increases the chance that the algorithm will find spurious models that are not valid in general.

It is well known that the required number of labeled samples for supervised classification increases as a function of dimensionality (see [18]). Fukunaga ([16]) showed that the required number of training samples is linearly related to the dimensionality for a linear classifier and to the square of the dimensionality for a quadratic classifier. In terms of nonparametric classifiers, the situation is even more severe. It has been estimated that, as the number of dimensions increases, the sample size needs to increase exponentially in order to have an effective estimate of multivariate densities (see [17]).

Bellman [5] was the first to define this phenomenon as the "curse of dimensionality", while working on complicated signal processing. Techniques that are efficient in low dimensions fail to provide meaningful results when the number of dimensions goes beyond a 'modest' size. Furthermore smaller data mining models, involving less attributes (probably less than 10), are much more understandable by humans. Smaller models are also more appropriate for user-driven data mining, based on visualization techniques.

Most of the methods for dealing with high dimensionality focus on Feature Selection techniques, i.e. selecting some subset of attributes upon which the induction algorithm will run, while ignoring the rest. The selection of the subset can be done manually by using prior knowledge to identify irrelevant variables or by using proper algorithms.

In the last decade, Feature Selection has enjoyed increased interest by the data mining community. Consequently many Feature Selection algorithms have been proposed, some of which have reported remarkable accuracy improvement. The literature on this subject is too wide to survey here, however, we recommend Langley [23], Liu and Motoda [25] and Maimon and Last [26] on this topic. Despite its popularity, using feature selection methodology for overcoming the high dimensionality obstacles has several shortcomings:

- The assumption that a large set of input attributes can be reduced to a small subset of relevant attributes is not always true; in some cases the target attribute is actually affected by most of the input attributes, and removing attributes will cause a significant loss of important information.
- The outcome (i.e. the subset) of many algorithms for *Feature Selection* (for instance almost any of the algorithms that are based upon the wrapper model methodology) is strongly dependent on the training set size. That is, if the training set is small the size of the reduced subset will be small as well due to the elimination of relevant attributes.
- In some cases, even after eliminating a set of irrelevant attributes, the researcher is left with relatively large numbers of relevant attributes.
- The backward elimination strategy, used by some methods, is extremely inefficient for working with large-scale databases, where the number of original attributes is more than 100.

A number of linear dimension reducers have been developed over the years. The linear methods of dimensionality reduction include projection pursuit (see [14]), factor analysis (see [19]), and principal components analysis (see [12]). These methods are not aimed directly at eliminating irrelevant and redundant attributes, but are rather concerned with transforming the observed variables into a small number of "projections" or "dimensions." The underlying assumptions are that the variables are numeric and the dimensions can be expressed as linear combinations of the observed variables (and vice versa). Each discovered dimension is assumed to represent an unobserved factor and thus provide a new way of understanding the data (similar to the curve equation in the regression models).

The linear dimension reducers have been enhanced constructive induction systems that use a set of existing attributes and a set of predefined constructive operators to derive new attributes (see [31]). These methods are effective for high dimensionality applications only if the original domain size of the input attribute can be in fact decreased dramatically.

One way to deal with the aforementioned disadvantages is to use a very large training set (which should increase in an exponential manner as the number of input attributes increase). However, the researcher rarely enjoys this privilege, and even if it does happen, the researcher will probably encounter the aforementioned difficulties derived from high number of records.

The rest of the paper is organized into three parts. In the first part we introduce the *Feature Decomposition Approach* literally and theoretically. In the second part we develop a heuristic algorithm for implementing the decomposition approach. In the third part we examine the algorithm on several artificial data and real applications.

2 Feature Decomposition Approach

The purpose of decomposition methodology is to break down a complex problem into several manageable problems. In artificial intelligence, according to Michie [30], finding a good decomposition is a major tactic both for ensuring the transparent end-product and for avoiding the combinatorial explosion.

Decomposition methodology can be considered as effective strategy for changing the representation of a learning problem. In fact Kusiak [22] consider decomposition as the "most useful form of transformation of data set".

It is generally believed that problem decomposition's benefits from: conceptual simplification of the problem, making the problem more feasible by reducing its dimensionality, achieving clearer results (more understandable), reducing run time by solving smaller problems and by using parallel or distributed computation and allowing different solution techniques for individual sub problems.

The decomposition approach is frequently used for operations research and engineering design, however, as Buntine [7] states, it has not attracted as much attention in KDD and machine learning community. Most of the work in machine learning decomposition can be found either in practical attempts in specific real life applications (see [7]) or in treatments of closely related problems mainly in the context of distributed and parallel learning (see [42]) or multiple classifiers (see [1]).

Figure 1.0 illustrates our approach for arranging the different types of decomposition in supervised learning.

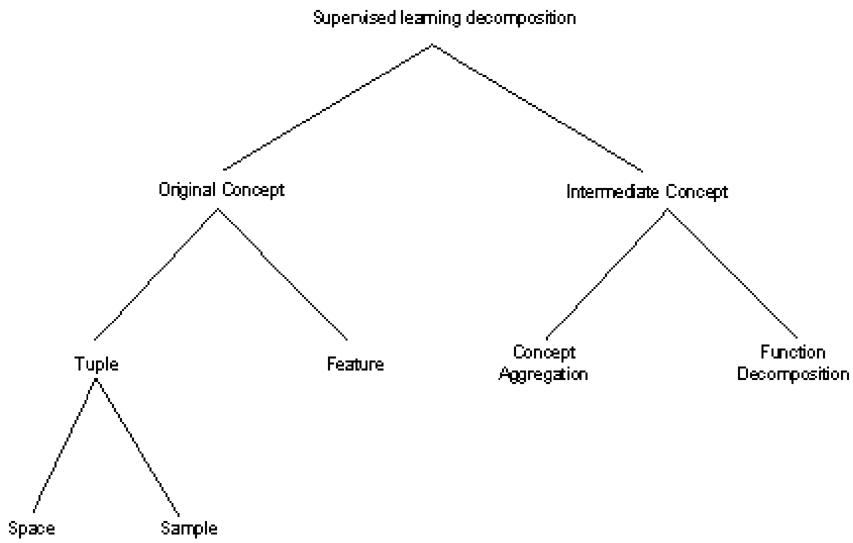

Fig. 1. Decomposition methods in supervised learning

In Intermediate Concept decomposition instead of learning a single complex classification model, several sub problems with different and simpler concepts are defined. The intermediate concepts can be based on aggregation of the original concept's values (concept aggregation) or not necessarily (function decomposition).

Buntine [7] used the concept aggregation to classify free text documents into predefined topics. Buntine [7] suggests breaking the topics up into groups (co-topics), and, instead of predicting the document's topic directly, first classifying the document to one of the co-topics and then using another model to predict the actual topic in that co-topic. Dietterich and Bakiri [9] have developed a general concept aggregation algorithm called Error-correcting output coding (ECOC) which decomposes multi-class problems into multiple two class problems. A classifier is build for each possible binary partition of the classes. Experiments show that ECOC improves the accuracy of neural networks and decision trees on several multi-class problems from the UCI repository.

Function decomposition was originally developed in 50's for improving the design of switching circuits. Recently this approach has been adopted by the machine learning community. Shapiro [38] and Michie [30] used a manual decomposition of the problem and an expert-assisted selection of examples to construct rules for the concepts in the hierarchy. In comparison with standard decision tree induction techniques, structured induction exhibits about the same classification accuracy with the increased transparency and lower complexity of the developed models. Zupan et al. [43] presented a general-purpose function decomposition approach for machine learning. According to this approach, attributes are transformed into new concepts in an iterative manner and create a hierarchy of concepts.

Original Concept decomposition, means dividing the original problem into several sub problems by partitioning the training set into smaller training sets. Each sub problem aims to solve the original problem, i.e. finding a classification model using only portion of the original training set. Then the classification models are combined in some fashion, either at learning or classification time.

There are two obvious ways to partition the data - tuples oriented or feature oriented which will be considered here.

Tuple decomposition by itself can be divided into two different types: sample and space. In sample decomposition (also known as partitioning) the goal is to partition the training set into several sample sets, such that each sub-learning task considers the entire space. In space decomposition on the other hand the original instance space is divided into several subspaces, each considered independently. In the last case no sophisticated combination is required.

For instance Chan and Stolfo [8] have used sample decomposition for reducing running time (as most of the inducers have running time that are quadratic in the sample size) as well as improving accuracy.

Bay [4] has presented a feature decomposition algorithm called MFS which combines multiple Nearest Neighbor classifiers each using only a subset random of features. It has shown that MFS can improve the standard Nearest Neighbor classifiers.

Blum and Mitchell [6] used a sort of feature decomposition for classification of Web pages. They suggested a Co-Training paradigm for learning with labeled and unlabeled data, especially when there is a large sample of data, which only small part of it is labeled. According to Blum and Mitchell paradigm of Co-Training the input space is divided to two different views (i.e. two independent and redundant sets of features). For each view they build a separated classifier that is then used to classify some unlabeled data. The new-labeled data of each classifier is then used to retrain the other classifier.

An important issue in decomposition is how its structure was obtained: It might be obtained manually based on expert's knowledge on a specific domain (like in [6], [38] or [30]) arbitrarily (as in [8]), due to some restrictions (like in the case of distributed learning) or induced without human interaction by a suitable algorithm (like in [43]).

Another important issue that differentiate between various decomposition implementations is whether the decomposition is mutually exclusive or partially overlapping, i.e. whether we utilize a certain data (a value of a certain attribute in a certain tuple) more than once. For instance in the case of sample decomposition, mutually exclusive means that a certain tuple cannot belong to more than one subset. (as in [8]). Bay [4] on the other hand, has used a non-exclusive feature decomposition. According to Bay exclusive feature decomposition may help avoid some of the error correlation problems.

In this paper we present new mutually exclusive feature decomposition where the attributes set is decomposed by an algorithm and not manually by the researcher. This method facilitates the creation of a classification model for high dimensionality databases. The idea we propose can be summarized as follows:

- The original input attribute set is decomposed to mutually exclusive subsets.
- An inducer algorithm is run upon the training data for each subset independently.
- The generated models are combined in order to make classification for new instances.

Variety of methods (like boosting or bagging), which also provide an improvement in classification performance by combining several simple classifiers produced by the same method, can be found in the literature. However, the resulting predictions are usually inscrutable to end-users ([34]), mainly due to the complexity of the generated models as well as the obstacles in transforming theses models into a single model. Moreover these methods do not attempt to use all relevant features, consequently in some cases especially when there are a lot of relevant attribute the researcher will not get the complete picture, of what attributes actually have an effect on the target attribute.

Other methods that deals directly with high dimensional data like support vector machines also suffer from inscrutability. The method proposed here improves classification performance, without jeopardizing the model's comprehensibility. Furthermore the *Feature Decomposition Approach* assists the researcher in discovering and understanding the relations between the different attributes, so it might lead to more comprehensible description of the learned concepts.

3 Notation

Throughout this paper we will use the following notation:

- $A = \{a_1, ..., a_i, ..., a_n\}$ - a set of n input attributes, where a_i is an attribute No. i.
- y - Represents the class variable or the target attribute.
- $G_k = \{a_{k,j(i)} \,|j = 1, ..., l_k\}$ - indicates a subset k that contains l_k input attributes where i is the original attribute index in the set A.
- $R_k = \{i/a_{k,j(i)} \in G_k, j = 1, ..., l_k\}$ denotes the correspondence indexes of subset k in the complete attribute set A.
- $Z = \{G_1, ...G_k..., G_\omega\}$ - A decomposition of the attribute set A into ω mutually exclusive subsets $G_k = \{a_{k,j(i)} \,|j = 1, ..., l_k\} \quad k = 1, ..., \omega\}$.
- $V_i = \{v_{i,j} \,|j = 1, ..., q_i\} \forall i = 1, ..., n$ - The domain of the attribute a_i with finite cardinality denoted q_i. We assume that each domain is a set of nominal values. For numeric attributes having continuous domains, each value represents an interval between two continuous values. In similar way V_y represents the domain of the target attribute.
- The *instance space* is defined as Cartesian product of all the input attributes domains: $X = V_1 \times V_2 \times ... \times V_n$. The *labeled instance space set* U is defined as Cartesian product of all input attribute domains and target attribute domain, i.e.:$U = X \times V_y$.
- $S = (< x_1, y_1 >, ..., < x_m, y_m >)$ - indicate the training set, containing m tuples (sometime called "instances," "records," "rows" or "observations") where $x_q \in X$and $y_q \in V_y$. We assume that the tuples are generated according to some fixed and unknown joint probability distribution D over U. Note that this is a generalization of the deterministic case when a supervisor classifies a tuple using a function $y = f(x)$.
- $x_{q,i}$ - value of attribute i in an observation q. $\forall q,i$: $x_{q,i} \in V_i$. For discretized attributes, this means that any continuous value can be related to one of the intervals. Null (missing) values are not allowed (unless they are encoded as a special value.

– We use the notation I to represent a probabilistic induction algorithm (by probabilistic learning algorithm we refer to an algorithm that generates classifiers that also provide estimates to the conditional probability of the target attribute given the input attributes), and by $I(S, G_k)$ to represent a probabilistic classifier which was induced by activating the learning method I on a training set S using the input attributes in the subset G_k. Using $I(S, G_k)$ one can estimate the probability $\hat{P}_{I(S,G_k)}(y = v_{y,j} \left| a_i = x_{q,i} \; i \in R_k \right.)$ of an observation x_q. Note we add a hat to the conditional probability estimation in order to distinguish it form the actual conditional probability.

4 The Problem Formulation

The problem of decomposing the input attribute set is that of finding the best decomposition, such that if a specific induction algorithm is run on each attribute subset data, then the combination of the generated classifiers will have the highest possible accuracy.

In this paper we focus on Naive Bayesian combination, which is an extension of the *Naive Bayesian* classifier. The *Naive Bayesian* classifier ([10]) uses Bayes rule to compute the probability of each possible value of the target attribute given the instance, assuming the input attributes are conditionally independent given the target attribute. The predicted value of the target attribute is the one, which maximizes the calculated probability, i.e.:

$$v_{MAP}(x_q) = \underset{v_{y,j} \in V_y}{\arg\max} \; \hat{P}_I(y = v_{y,j}) \cdot \prod_{i=1}^{n} \hat{P}_I(a_i = x_{q,i} \left| y = v_{y,j} \right.)$$

Despite its naive assumption, a variety of empirical research shows surprisingly that the Naive Bayesian classifier can perform quite well compared to other methods even in domains where clear attribute dependencies exist. Furthermore, Naive Bayesian classifiers are also very simple and easy to understand (see [20]).

In the Feature Decomposition approach with Naive Bayesian combination we use a similar idea, namely the prediction of a new instance x_q is based on the product of the conditional probability of the target attribute, given the values of the input attributes in each subset. Mathematically it can be formulated as follows:

$$v_{MAP}(x_q) = \underset{v_{y,j} \in V_y}{\arg\max} \frac{\prod_{k=1}^{\omega} \hat{P}_{I(S,G_k)}(y = v_{y,j} \left| a_i = x_{q,i} \; i \in R_k \right.)}{\hat{P}_{I(S,G_k)}(y = v_{y,j})^{\omega - 1}}$$

In fact extending the Naive Bayesian classifier by joining attributes is not new. Kononenko [21] have suggested the Semi-Naive Bayesian Classifier that use a conditional independence test to join attributes. Domingos and Pazzani [10] used estimated accuracy (as determined by leaveoneout cross validation on the training set). In both cases, the suggested algorithm finds the single best pair of attributes to join by considering all possible joins. Friedman et al [15] have suggested *Tree Augmented Naive*

Bayesian classifier (TAN) which extends Naive Bayes taking into account dependencies among input attributes. Selective Bayesian Classifier ([24]) preprocesses data using a form of feature selection to delete redundant attributes.

However these methods have not noticeably improved accuracy. The reasons for the moderate results are two-fold. First both algorithms used a limited criterion for joining attributes. Second and more importantly, attributes are joined by creating a new attribute, whose values are the Cartesian product of its ingredients' values, specifically the number of attributes that can be joined together is restricted to a small number.

Duda and Hart ([11]) showed that Bayesian classifier has highest possible accuracy (i.e. Bayesian classifier predicts the most probable class of a given instance based on the complete distribution). However in practical learning scenarios, where the training set is very small compared to the whole space, the complete distribution can hardly be estimated directly.

According to the decomposition concept the complete distribution is estimated by combining several partial distributions. Bear in mind that it is easier to build a simpler model from limited number of training instances because it has less parameters. This makes classification problems with many input attributes more feasible.

This problem can be related to the bias-variance tradeoff (see for example [15]). The bias of a learning algorithm for a given learning problem is the persistent or systematic error that the learning algorithm is expected to make. A concept closely related to bias is variance. The variance captures random variation in the algorithm from one training set to another. This variation can result from variation in the training sample, from random noise in the training data, or from random behavior in the learning algorithm itself. The smaller each subset is, the less probabilities we have to estimate and potentially less variance in the estimation of each one of them. On the other hand, when there are more subsets, we expect that the approximation of the full distribution using the partial distributions is less accurate (i.e. higher bias error). Formally the problem can be phrased as follows:

Given a learning method I, and a training set S with input attribute set $A = \{a_1, a_2, ..., a_n\}$ *and target attribute* y *from a distribution D over the labeled instance space, the goal is to find an optimal decomposition* Z_{opt} *of the input attribute set A into* ω *mutually exclusive subsets* $G_k = \{a_{k,j(i)} \,|j = 1, ..., l_k\} \quad k = 1, ..., \omega$ *not necessarily exhaustive such that the generalization error of the Naive Bayesian combination of the induced classifiers* $I(S, G_k), k = 1, ..., \omega$ *will be minimized over the distribution D.*

It should be noted that the optimal is not necessarily unique. Furthermore it is not obligatory that all input attributes will actually belong to one of the subsets. Because of that the problem can be treated as an extension of the feature selection problem, i.e. finding the optimal decomposition of the form $Z_{opt} = \{G_1\}$,as the non-relevant features are in fact *NR=A-*G_1.

Moreover, when using a Naive Bayes for combining the models as described above, the Naive Bayes Classifier can be treated as specific decomposition: $Z = \{G_1, G_2, ..., G_n\}$, where $G_i = \{A_i\}$.

5 The Single Oblivious Decision Tree for Feature Selection

Oblivious Decision Trees are decision trees, in which all nodes at the same level test the same attribute. Despite its restriction, oblivious decision trees found to be effective as a feature selection procedure. Almuallim and Dietterich [2]) as well as Schlimmer [37] have proposed forward feature selection procedure by constructing oblivious decision trees.

Recently Maimon and Last [26] have suggested a new algorithm for constructing oblivious decision trees, called *IFN* that is based on information theory. Figure 2 represents a typical oblivious decision tree with three input attributes: glucose level, age and blood pressure of a certain patient and the Boolean target attribute representing whether that patient suffer from diabetes. Each layer is uniquely associated with an input attribute by representing the interaction of that attribute and the input attributes of the previous layers. The first layer (layer No. 0) includes only the root node and is not associated with any input attribute. The oblivious decision tree can also be used to predict values of target attributes in a disjunctive manner, similar to the decision-tree. The principal difference between the oblivious decision tree and a regular decision tree structure (see [33]) is the *constant ordering* of input attributes at every terminal node of the oblivious decision tree, the property which is necessary for minimizing the overall subset of input attributes (resulting in dimensionality reduction). The arcs that connect the terminal nodes and the nodes of the target layer are labeled with the number of records that fit this path. For instance there are twelve patients in the training set whose their glucose level is less than 107 but who still suffer from diabetes.

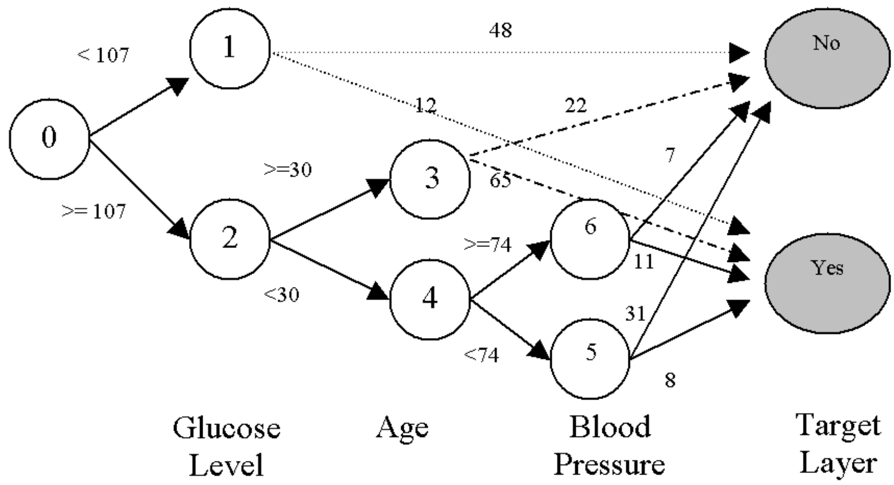

Fig. 2. Illustration of Oblivious Decision Tree

6 The DOT Algorithm

As the oblivious decision tree approach and more specifically the IFN algorithm were found to be effective in discovering the relevant attributes and their relations to the target attribute, we further extended it aiming to approach the optimal feature decomposition (Z_{opt}). For our purpose each subset is represented by a different oblivious decision tree, while each attribute in this subset is located in a different layer. Attributes in the same tree should be dependent, while independent attributes should be in different trees as the independent assumption suggests. However it does not mean that attributes in different trees are necessarily independent. In some cases assigning dependent attributes into different groups contributes to the overall classification accuracy.

DOT (Decomposed Oblivious decision Trees) is an algorithm that instantiates this idea. The DOT learning algorithm begins with a single tree with a single node (the root node), representing an empty subset of input attributes. In each iteration the algorithm decides which attribute should be added to the current tree as a new layer, and to what nodes on the previous layer it will be connected (the splitted nodes). The nodes of a new layer are defined as all Cartesian product combinations of the splitted nodes with the values of the new input attribute, which have at least one observation associated with it.

Obviously DOT can construct up to n trees (in the extreme case where each tree may represent one input attribute) and up to n attributes in each tree (in the case where we have one tree that include all input attributes).

Each valid instance (instance that belongs to the instance space), is associated in each tree with exactly one path between the root node and the target layer. We make the following two assumptions:

- There are no missing values. Either they are replaced with valid values or the observation should be ignored.
- All attributes have discretely valued domains. Continuous attributes should be made discreet before using them in the algorithm.

The selection of the a new input attribute is made according the following criteria:

- The selected attribute should maximize the total significant decrease in the conditional entropy, as a result of adding it as a new layer. In order to calculate the total significant decrease in the conditional entropy, we estimate for each node in the last layer, the decrease in the conditional entropy as a result of splitting that node according the candidate attribute values. Furthermore the decrease in the conditional entropy is tested for significance by a likelihood-ratio test ([3]). The null hypothesis (H_0) is that a candidate input attribute and a target attribute are conditionally independent, given the node (implying that their conditional mutual information is zero). If H_0 holds, the test statistic is distributed as χ^2 with degrees of freedom equal to the number of independent proportions estimated from the observations associated with this node. Finally all significant decreases of specific nodes are summed up to achieve the total significant decrease.
- The attribute is conditionally dependent on the splitted nodes (nodes with significant decrease in the conditional entropy) given the target attribute. For testing the

conditional independency we use a χ^2 test as described in [41]. The null hypothesis (H_0) is that a candidate input attribute and all splitted nodes are conditionally independent, given the target attribute.

– Adding the attribute to the current tree should decrease the generalization error bound of the combined trees so far. The bound is discussed in the following section. In this case there are two purposes for using it; First of all as adding a new attribute to the current tree increases this tree's complexity. Under the Occam's-razor assumption that simplicity is a virtue in conflict with training accuracy, we verify whether the decrease in the entropy worth the addition in complexity. Moreover the addition of the attribute to the current tree, contributes to the performance of the combined trees structure as a whole.

If no input attribute was selected an attempt is made to construct a new tree (a new subset) with the input attributes that were not used by previous trees. This procedure continues until there is no more unused attributes or until the last tree is left empty.

7 Generalization Error Bound for Multiple Oblivious Decision Trees

Over-fitting and under-fitting are well known problems in learning. A conventional approach to overcome these problems is to use a generalization error bound in terms of the training error and concept size.

As stated before, a oblivious decision tree can be considered as restricted decision tree, for that reason we might use one of the generalization error bound that were developed for decision trees in the literature (see [28]). However there are several reasons to develop a specific bound. First, we can utilize the fact that oblivious structure is more restricted, in order to develop a tighter bound. Second we are required to extend the bound for several oblivious trees combined using Naive Bayesian combination.

Due to the fact that we use probabilistic classification tree (represented by real value weights on the terminal nodes), the hypothesis space is infinite. A typical bound for infinite hypothesis space which makes use of VC (Vapnik and Chervonenkis) dimension, and holds with probability at least $1 - \delta$, has the following form ([39]):

$$|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \sqrt{\frac{d \cdot (\ln \frac{2m}{d} + 1) - \ln \frac{\delta}{4}}{m}} \quad \begin{array}{l} \forall h \in H \\ \forall \delta > 0 \end{array}$$

Where d is the VC-Dimension of hypothesis class H, $\hat{\varepsilon}(h)$ is the error of h on training set of cardinality m.

The VC dimension for a set of indicator functions is defined as the maximum number of data points that can be shattered by the set of admissible functions. By definition, a set of m points is shattered by a concept class if there are concepts (functions) in the class that split the points into two classes in all of the 2^m possible ways. The VC dimension might be difficult to compute accurately. Here we introduce an upper bound and a lower bound of the VC dimension. The functions that we consider as a hypothesis class are computed by classes of multiple mutually exclusive oblivious decision trees. We characterize such a class by two vectors and one scalar: $\boldsymbol{L} = (l_1, ..., l_\omega)$, $\boldsymbol{T} = (t_1, ..., t_\omega)$ and n. Where l_k is the numbers of layers (not including the root and target layers) in the tree k, t_k is the number of terminal nodes in the tree k and n is the number of input attributes.

For the sake of simplicity the bound described in this section is developed assuming that the input attributes and the target attribute are both binary. However the bound can be extended for other cases in a straightforward manner. Note that each oblivious decision tree with non-binary input attributes can be converted to a corresponding binary oblivious decision tree by using appropriate artificial attributes.

Before presenting the main theorem we first consider two lemmas:

Lemma 1. *The VC dimension of oblivious decision tree on n binary input attributes with l layers and t terminal nodes is not greater than:* $t + \log_2(\frac{n!}{(n-l)!} \cdot \frac{(2t-4)!}{(t-2)! \cdot (t-2)!})$

Proof. Any oblivious decision tree can be converted to a suitable classification tree with leafs labeled $\{0,1\}$ according to the highest weight of each of the terminal node in the original tree. Because the probabilistic oblivious tree and its corresponded classification tree shattered the same subsets, their VC dimensions are identical.

The hypothesis space size of a classification oblivious tree with l layers, t terminal nodes and n input attributes to choose from is not greater than:

$$\frac{n!}{(n-l)!} \cdot 2^t \cdot \frac{(2t-4)!}{(t-2)! \cdot (t-2)!}$$

The first multiplier indicates the number of combinations for selecting with order l attributes from n. The second multiplier corresponds to the different classification options of the terminal nodes. The third multiplier represents the number of different binary tree structures that contains t leaves. The last multiplier is calculated using Wallace tree structure (see [40]). Note that in the case of binary tree there are exactly one more leaves than inner nodes. Furthermore the tree string always begins with an inner nodes (when $l \geq 1$) and end with at least two leaf nodes.

Based on the familiar relation $VC(H) \leq \log_2(|H|)$ for finite H, we proved the Lemma.

Lemma 2. *Consider ω mutually exclusive oblivious decision trees that are combined with Naive Bayesian and that has a fixed structure containing $\boldsymbol{T} = (t_1, ..., t_\omega)$ terminal nodes. The number of dichotomies it induces on a set of cardinality m is at most:*

$$2\left(\frac{em}{1+\sum\limits_{i=1}^{\omega} t_i}\right)^{1+\sum\limits_{i=1}^{\omega} t_i}.$$

Proof. For proving the lemma, we use a similar lemma introduced by Schmitt [36] for the number of dichotomies that a higher order threshold neuron with k monomials induces on a set of cardinality m.

A definition of a higher-order threshold neuron has the form:

$$w_1 M_1 + w_2 M_2 + ... + w_k M_k - t$$

Where $M_1, M_2, ..., M_k$ are monomials.

ω binary and oblivious decision trees which are combined with Naive Bayesian, can be converted to a higher order threshold neuron, where the set of terminal nodes constitutes the neuron's monomials and the log-odds in favor of $y = 1$ in each terminal node is the corresponded neuron's weight. Furthermore to be able to use the sign activation function we use a threshold that equals to the sum of all other monomials.

Theorem 1. *The VC-Dimension of ω mutually exclusive oblivious decision trees on n binary input attributes that are combined using Naive Bayesian combination and that have $\boldsymbol{L} = (l_1, ..., l_\omega)$ layers and $\boldsymbol{T} = (t_1, ..., t_\omega)$ terminal nodes is not greater than:*

$$\begin{cases} F + \log U & \omega = 1 \\ 2(F+1)\log(2e) + 2\log U & \omega > 1 \end{cases}$$

and at least:

$$F - \omega + 1$$

where:

$$U = \frac{n!}{(n - \sum\limits_{i=1}^{\omega} l_i)!} \cdot \prod_{i=1}^{\omega} \frac{(2t_i - 4)!}{(t_i - 2)! \cdot (t_i - 2)!} \; ; \; F = \sum_{i=1}^{\omega} t_i$$

Proof. We first discuss the upper bound. If $\omega = 1$ we can use Lemma 1 directly. For the case $\omega > 1$ we first bound the number of dichotomies induced by ω mutually exclusive oblivious decision trees on an arbitrary set of cardinality m. Because the biggest shattered set follows this bound as well, we derive the statement of theorem.

There are at most:

$$\frac{n!}{(n - \sum\limits_{i=1}^{\omega} l_i)!} \cdot \prod_{i=1}^{\omega} \frac{(2t_i - 4)!}{(t_i - 2)! \cdot (t_i - 2)!}$$

different structures for ωmutually exclusive oblivious trees on n binary input attributes with $\boldsymbol{L} = (l_1, ..., l_\omega)$ layers and $\boldsymbol{T} = (t_1, ..., t_\omega)$ terminal nodes.

According to Lemma 2 a fixed structure and variable weights can induce at most:

$$2 \left(\frac{em}{1 + \sum\limits_{i=1}^{\omega} t_i} \right)^{1 + \sum\limits_{i=1}^{\omega} t_i}$$

dichotomies on a given set of cardinality m. Enumerating over all structures, we derive that there are at most:

$$\frac{n!}{(n-\sum\limits_{i=1}^{\omega} l_i)!} \cdot \prod_{i=1}^{\omega} \frac{(2t_i-4)!}{(t_i-2)! \cdot (t_i-2)!} \cdot 2 \left(\frac{em}{\sum\limits_{i=1}^{\omega} t_i} \right)^{\sum\limits_{i=1}^{\omega} t_i}$$

dichotomies on a given set of cardinality m that are induce by the class considered. If the above class shatters the given set, we must have:

$$2^m \leq \frac{n!}{(n-\sum\limits_{i=1}^{\omega} l_i)!} \cdot \prod_{i=1}^{\omega} \frac{(2t_i-4)!}{(t_i-2)! \cdot (t_i-2)!} \cdot 2 \left(\frac{em}{1+\sum\limits_{i=1}^{\omega} t_i} \right)^{1+\sum\limits_{i=1}^{\omega} t_i}$$

However the last inequality will not be true If we choose $m \geq 2F \log(2e) + 2 \log U$ where F and U defined above. The lower bound is true due to the fact that any set of ω trees with fixed structure has the above VC dimension. The result can be achieved by setting in each tree (beside one) a neutralized terminal node (i.e. a terminal node with posteriori probabilities that are equal to the apriority probabilities). This concludes our proof.

Our preliminary experiments have shown that estimating the generalization error by using the lower bound of the VC Dimension provides a better performance.

8 Making a Classification

The oblivious decision trees constructed by DOT algorithm can be used for classification of unlabeled instances, by performing the the following steps:

- In each tree locate the relevant node (final or unsplitted) that satisfies the unseen instance.
- Extract from the located node, the frequency vector (how many instances relate to each possible value of the target attribute.)
- Transform the frequency vector to probability vector. Using the frequency as is will typically over-estimate the probability so we use the Laplace correction to avoid this phenomenon (See [10]). According to Laplace's law of succession, the probability of the event $a_i = v_{i,j}$ which has been observed m_i times of m instances is $(m_i + 1)/(m + d_i)$ where d_i denotes the domain size of attribute a_i.
- Combine the probability vectors using Naive Bayesian combination.
- Select the target value maximizing the Naive Bayesian combination.

9 Evaluation of DOT on variety Classification Problems

In order to validate the DOT algorithm and to illustrate the usefulness of the decomposition approach in classification problems, we apply our method to various domains and compare the results to other methods. In this paper we decided to compare the DOT algorithm to the following algorithms: IFN, Naive Bayes and C4.5. We have special interest in both IFN and Naive Bayes as they represent specific points in the search space of the DOT algorithm. We also compare the DOT to C4.5, a state-of-the-art heuristic machine learning algorithm, because its similarity to IFN (in the model structure) and due to its popularity in many other comparative studies.

The *DOT* approach has been applied to 15 representative public domain data sets from the UCI Machine learning and KDD Repository ([29]). Although we recognize the limitations of using this repository for comparing algorithms (see [35]), we use it here because it is considered as an objective method for comparing algorithms since the published results can be validated.

The databases passed a simple preprocessing stage. In this stage missing values are replaced by a distinctive value, and numeric attributes are made discrete by dividing their original range to ten equal-size intervals (or one per observed value, whichever was least). We could improve the results by using more sophisticated discretization. However, since the main purpose of this trial is to verify whether the *DOT* is capable of approximating optimal Feature Decomposition, we decided that any non-relevant differences between the algorithms should be neutralized. Furthermore after *DOT* create a decomposition, one may use any other state-of-the-art data mining algorithms for constructing the appropriate models.

Table 1 shows the average accuracy and sample standard deviation obtained by using 10-fold-cross-validation. One tailed paired t-test with confidence level of 95% was used in order to verify whether the differences in accuracy are statistically significant. The results of our experimental study are very encouraging. In fact there was no significant case where Naive Bayes or IFN were more accurate than *DOT*, on the other hand *DOT* was more accurate than Naive Bayes and IFN in 8 databases, and 7 databases respectively. Furthermore *DOT* was significantly more accurate than C4.5 in 9 databases, and less accurate in only 2 databases.

The contribution of feature decomposition in real life applications can be found in [27].

10 The link between Error Reduction and the Problem Complexity.

In order to understand when the suggested approach introduces considerable performance enhancement, we checked the correlation between error reduction and the problem complexity.

There are two obvious alternatives for measuring the error reduction achieved by using Feature Decomposition approach: measuring the error difference between IFN and DOT or measuring the error ratio (i.e. the error of DOT divided by the error of single IFN). Following [1] we use error ration because it manifests the fact that it becomes

Table 1. Summary of experimental results

Database	Naive Bayes	C4.5	IFN	D-IFN
Aust	84.93±2.7	85.36±5.1	84.49±5.1	84.49±2.9
Bcan	97.29±1.6	$^{+}$92.43±3.5	$^{+}$94.39±3.5	97.29±1.6
LED17	$^{+}$63.18±8.7	$^{+}$59.09±6.9	$^{+}$55.55±6.3	73.64±5.5
LETTER	$^{+}$73.29±1	$^{+}$74.96±0.8	$^{+}$69.56±0.7	79.07±0.9
Monks1	$^{+}$73.39±6.7	$^{+}$75.81±8.2	$^{+}$75.00±10.7	92.74±11
Monks2	$^{+}$56.21±6.1	$^{+}$52.07±8.6	63.87 ±6.4	62.13±6.4
Monks3	93.44±3.7	93.44±3.7	92.38±3.3	92.62±3.3
MUSH	$^{+}$95.48±0.9	100±0	100±0	100±0
Nurse	$^{+}$65.39±24	$^{-}$97.65±0.4	92.47±0.5	92.67±0.6
OPTIC	91.73±1.3	$^{+}$62.42±2	$^{+}$48.90±2.5	91.73±1.4
Sonar	75.48±7.3	69.71±5.4	76.48±6.8	77.12±8.7
SPI	$^{+}$94.2±0.9	$^{+}$91.2±1.9	$^{+}$87.00±2.6	95.8±0.9
TTT	$^{+}$69.27±3.2	$^{-}$85.31±2.7	73.19±3.9	73.33±4
Wine	96.63±3.9	$^{+}$85.96±6.9	$^{+}$91.45±5	96.63±3.9
Zoo	89.11±7	93.07±5.8	90.89±9.7	92.71±7.3

gradually more difficult to achieve error reduction as the error of single IFN converge to zero.

In order to estimate the problem complexity we used the following ratio $\log(|H|)/m$. Where m is the training set size and $|H|$is the hypothesis space size of all possible functions of the investigated problem. Note that the problem complexity increases as the problem dimensionality grows.

The estimated linear correlation coefficient (r) is 0.9. This result is quite encouraging as it evidently indicates when should we potentially use Feature Decomposition.

11 Summary and Future Work

In this paper we presented a new concept of Feature Decomposition for classification problems and proposed the DOT algorithm for discovering the appropriate decomposition. The algorithm has been implemented on variety of datasets, and has generated models with higher classification accuracy than other comparable methods.

Finally, the issues to be further studied include: considering other search methods for the problem defined, examining how the Feature Decomposition concept can be implemented using other classification methods like Neural Networks, examining other techniques to combine the generated classifiers, and exploring different decomposition paradigms other than Feature Decomposition.

Along with performing more empirical, we hope to better understand when it will be appropriate to use attributes decomposition.

References

1. Ali K. M., Pazzani M. J., Error Reduction through Learning Multiple Descriptions, Machine Learning, 24(3): 173-202, 1996.

2. Almuallim H. and Dietterich T.G., Learning boolean concepts in the presence of many irrelevant features. Artificial Intelligence, 69(1-2):279–306, 1994.
3. Attneave, F., Applications of Information Theory to Psychology. Holt, Rinehart and Winston, 1959.
4. Bay, S. Nearest neighbor classification from multiple feature subsets. Intelligent Data Analysis, 3(3): 191-209, 1999.
5. Bellman, R., Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.
6. Blum , A. and Mitchell, T., "Combining Labeled and Unlabeled Data with Cotraining", COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 1998.
7. Buntine, W., "Graphical Models for Discovering Knowledge", in U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp 59-82. AAAI/MIT Press, 1996.
8. Chan, P.K. and Stolfo, S.J., A Comparative Evaluation of Voting and Meta-learning on Partitioned Data, Proc. 12th Intl. Conf. on Machine Learning ICML-95, 1995.
9. Dietterich, T. G., and Bakiri, G., Solving multiclass learning problems via errorcorrecting output codes. Journal of Artificial Intelligence Research, 2:263-286, 1995.
10. Domingos, P., and Pazzani, M., "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," Machine Learning, 29: 103-130, 1997.
11. Duda, R., and Hart, P., Pattern Classification and Scene Analysis, New-York, NY: Wiley, 1973.
12. Dunteman, G.H., Principal Components Analysis, Sage Publications, 1989.
13. Fayyad, U., Piatesky-Shapiro, G., and Smyth P., "From Data Minig to Knowledge Discovery: An Overview," in U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pp 1-30, MIT Press, 1996.
14. Friedman, J.H., and Tukey, J.W., "A Projection Pursuit Algorithm for Exploratory Data Analysis," IEEE Transactions on Computers, 23 (9): 881-889, 1974.
15. Friedman, J.H., "On bias, variance, 0/1 - loss and the curse of dimensionality," Data Mining and Knowledge Discovery, 1 (1) : 55-77, 1997.
16. Fukunaga, K., Introduction to Statistical Pattern Recognition. San Diego, CA: Academic, 1990.
17. Hwang J., Lay S., and Lippman A., "Nonparametric multivariate density estimation: A comparative study," IEEE Trans. Signal Processing, vol. 42, pp. 2795– 2810, Oct. 1994.
18. Jimenez, L. O., and Landgrebe D. A., "Supervised Classification in High- Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data." IEEE Transaction on Systems Man, and Cybernetics — Part C: Applications and Reviews, 28:39-54, 1998.
19. Kim J.O., and C.W. Mueller, Factor Analysis: Statistical Methods and Practical Issues. Sage Publications, 1978.
20. Kononenko, I., "Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition". In Current Trends In reply to: Knowledge Acquisition, IOS Press, 1990.
21. Kononenko, I., "Seminaive Bayesian classifier," in Proceedings of the Sixth European Working Session on Learning, SpringerVerlag, pp. 206-219, 1991.
22. Kusiak, A., Decomposition in Data Mining: An Industrial Case Study, IEEE Transactions on Electronics Packaging Manufacturing, Vol. 23, No. 4, 2000, pp. 345-353
23. Langley, P., "Selection of relevant features in machine learning," in Proceedings of the AAAI Fall Symposium on Relevance. AAAI Press, 1994.
24. Langley, P. and Sage, S., Oblivious decision trees and abstract cases. Working Notes of the AAAI-94 Workshop on Case-Based Reasoning, Seattle, WA: AAAI Press, 113-117.

25. Liu and H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, 1998.
26. Maimon, O., and M. Last, Knowledge Discovery and Data Mining: The Info-Fuzzy network (IFN) methodology, Kluwer Academic Publishers, 2000.
27. Maimon, O. and Rokach, L., "Data Mining by Attribute Decomposition with semiconductors manufacturing case study" in D. Bracha, Editor, Data Mining for Design and Manufacturing: Methods and Applications, Kluwer Academic Publishers, 2001.
28. Mansour, Y., and McAllester, D., Generalization Bounds for Decision Trees, COLT 2000: 220-224.
29. Merz, C.J, and Murphy. P.M., UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
30. Michie, D., "Problem decomposition and the learning of skills," in Proceedings of the European Conference on Machine Learning, Springer-Verlag, PP. 17-31, 1995.
31. Pfahringer, B., "Controlling constructive induction in CiPF," in Proceedings of the European Conference on Machine Learning, Springer-Verlag, pp. 242-256. 1994.
32. Pickard, L., B. Kitchenham, and S. Linkman., "An investigation of analysis techniques for software datasets", in Proc. 6th IEEE Intl. Software Metrics Symposium. Boca Raton, FL: IEEE Computer Society, 1999.
33. Quinlan, J.R., C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
34. Ridgeway, G., Madigan, D., Richardson, T. and O'Kane, J. (1998), "Interpretable Boosted Naive Bayes Classification", Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp 101-104.
35. Salzberg. S. L. (1997), "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach". Data Mining and Knowledge Discovery, 1, 312-327, Kluwer Academic Publishers, Boston.
36. Schmitt , M. On the complexity of computing and learning with multiplicative neural networks, to appear in Neural Computation, 2001.
37. Schlimmer, J. C. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In Proceedings of the 1993 International Conference on Machine Learning, pages 284–290, San Mateo, CA, 1993. Morgan Kaufmann.
38. Shapiro, A. D., Structured induction in expert systems, Turing Institute Press in association with Addison-Wesley Publishing Company, 1987.
39. Vapnik, V.N., 1995. The Nature of Statistical Learning The-ory. Springer-Verlag, New York.
40. Wallace, C. S., 1996. MML Inference of Predictive Trees, Graphs and Nets. In Computational Learning and Probabilitic Reasoning, A. , Gammerman (ed), Wiley, pp43-66.
41. Walpole, R. E., and Myers, R. H., Probability and Statistics for Engineers and Scientists, pp. 268-272, 1986.
42. Zaki, M. J., and Ho, C. T., Eds., Large- Scale Parallel Data Mining. New York: Springer-Verlag, 2000.
43. Zupan, B., Bohanec, M., Demsar, J., and Bratko, I., "Feature transformation by function decomposition," IEEE intelligent systems & their applications, 13: 38-43, 1998.