Pattern Analysis and Applications manuscript No.
(will be inserted by the editor)

Decomposition Methodology for Classification Tasks - A Meta Decomposer Framework

Lior Rokach[1]

Department of Information Systems Engineering
Ben-Gurion University of the Negev
P.O.B. 653, Beer-Sheva 84105, Israel
liorrk@bgu.ac.il

The date of receipt and acceptance will be inserted by the editor

Abstract The idea of decomposition methodology for classification tasks is to break down a complex classification task into several simpler and more manageable sub-tasks that are solvable by using existing induction methods, then joining their solutions together in order to solve the original problem. In this paper we provide an overview of very popular but diverse decomposition methods and introduce a related taxonomy to categorize them. Subsequently we suggest using this taxonomy to create a novel meta-decomposer framework to automatically select the appropriate decomposition method for a given problem. The experimental study validates the effectiveness of the proposed meta-decomposer on a set of benchmark datasets.

1 Introduction

One of the explicit challenges in classification tasks is to develop methods that will be feasible for complicated real-world problems. In many disciplines, when a problem becomes more complex, there is a natural tendency to try to break it down into smaller, distinct but connected pieces. The concept of breaking down a system into smaller pieces is generally referred to as *decomposition*. The purpose of decomposition methodology is to break down a complex problem into smaller, less complex and more manageable, sub-problems that are solvable by using existing tools, then joining them together to solve the initial problem. Decomposition methodology can be considered as an effective strategy for changing the representation of a classification problem. Indeed, Kusiak [38] considers decomposition as the "most useful form of transformation of data sets".

The decomposition approach is frequently used in statistics, operations research and engineering. For instance, decomposition of time series is con-

sidered to be a practical way to improve forecasting. The usual decomposition into trend, cycle, seasonal and irregular components was motivated mainly by business analysts, who wanted to get a clearer picture of the state of the economy [18]. Although the operations research community has extensively studied decomposition methods to improve computational efficiency and robustness, identification of the partitioned problem model has largely remained an ad hoc task [26].

In engineering design, problem decomposition has received considerable attention as a means of reducing multidisciplinary design cycle time and of streamlining the design process by adequate arrangement of the tasks [37]. Decomposition methods are also used in decision-making theory. A typical example is the AHP (Analytic Hierarchy Process) method [62]. In artificial intelligence finding a good decomposition is a major tactic, both for ensuring the transparent end-product and for avoiding a combinatorial explosion [45].

Research has shown that no single learning approach is clearly superior for all cases. In fact, the task of discovering regularities can be made easier and less time consuming by decomposition of the task. However, decomposition methodology has not attracted as much attention in the pattern recognition and machine learning community [11].

Although decomposition is a promising technique and presents an obviously natural direction to follow, there are hardly any works in the pattern recognition literature that consider the subject directly. Instead, there are abundant practical attempts to apply decomposition methodology to specific, real life applications [11]. There are also many discussions on closely related problems, largely in the context of distributed and parallel learning [71] or ensembles classifiers.

Various decomposition methods have been presented [38]. There was also suggestion to decompose the exploratory data analysis process into 3 parts: *model search*, *pattern search*, and *attribute search* [7]. However, in this case the notion of "decomposition" refers to the entire process, while this paper focuses on decomposition of the model search.

In the neural network community, several researchers have examined the decomposition methodology [25]. The *"mixture-of-experts"* (ME) method decomposes the input space, such that each expert examines a different part of the space [46]. However, the sub-spaces have soft "boundaries", namely sub-spaces are allowed to overlap. Each expert outputs the conditional probability of the target attribute given the input instance. A gating network is responsible for combining the various experts by assigning a weight to each network. These weights are not constant but are functions of the input instance x.

Hierarchical mixtures of experts (HME) is well-known extension to the basic mixture of experts [32]. This extension decomposes the space into sub-spaces, and then recursively decomposes each sub-space to sub-spaces.

Variation of the basic mixtures of experts methods have been developed to accommodate specific domain problems. A specialized modular network

called the Meta-p_i network has been used to solve the vowel-speaker problem [24,48]. There have been other extensions to the ME such as nonlinear gated experts for time-series [70]; revised modular network for predicting the survival of AIDS patients [47]; and a new approach for combining multiple experts for improving handwritten numerals recognition [53].

Several taxonomies for decomposition methods have been suggested in the literature [38,66]. However, there is no work that considers the coexistence of these different decomposition methods in order to answer practical questions such as: When should we prefer one decomposition method over the other? Is it possible to solve a given problem using a hybridization of several decomposition methods?

In our previous work [57,43], we presented a preliminary taxonomy for decomposition of classification tasks. In this paper, we first extend this taxonomy and subsequently suggest a systematic way such a taxonomy can be used in practice. More specifically the taxonomy is used as the basis for a new decision tree-based meta-decomposer which aims to automatically identify the best decomposition method for a given database.

2 Decomposition Advantages

Decomposition methods can improve the predictive accuracy of regular methods. In fact in some cases improving performance is the main motivation for decomposition [66]. Although this might look surprising at first, it can be explained by the bias-variance tradeoff. Since decomposition methodology constructs several simpler sub-models instead a single complicated model, we might gain better performance by choosing the appropriate sub-models' complexities (i.e. finding the best bias-variance tradeoff). For instance, a single decision tree that attempts to model the entire instance space usually has high variance and small bias. On the other hand, Naïve Bayes can be seen as a composite of single-attribute decision trees (each one of these trees contains only one unique input attribute). The bias of the Naïve Bayes is large (as it can not represent a complicated classifier); on the other hand, its variance is small. Decomposition can potentially obtain a set of decision trees, such that each one of the trees is more complicated than a single-attribute tree (thus it can represent a more complicated classifier and it has lower bias than the Naïve Bayes) but not complicated enough to have high variance.

There are other justifications for the performance improvement of decomposition methods, such as the ability to exploit the specialized capabilities of each component, and consequently achieve results which would not be possible in a single model. For instance the identification accuracy of a clinical diagnosis can be improved when the problem is decomposed and two neural networks are trained [3].

One of the explicit challenges of the research community is to develop methods that facilitate the use of pattern recognition algorithms for real-world applications. In the information age, data is automatically collected

and therefore the database available for patterns discovery can be quite large, as a result of an increase in the number of records in the database and the number of fields/attributes in each record (high dimensionality).

There are many approaches for dealing with huge databases including: sampling methods; massively parallel processing; efficient storage methods; and dimension reduction. Decomposition methodology suggests an alternative way to deal with the aforementioned problems by reducing the volume of data to be processed at a time. Decomposition methods break the original problem into several sub-problems, each one with relatively small dimensionality. In this way, decomposition reduces training time and makes it possible to apply standard pattern recognition algorithms to large databases [66].

Decomposition methods suggest a conceptual simplification of the original complex problem. Instead of getting a single and complicated model, decomposition methods create several sub-models, which are more comprehensible. This motivation has often been noted in the literature [49, 28, 66]. Smaller models are also more appropriate for user-driven data mining that is based on *visualization techniques*. Furthermore, if the decomposition structure is induced by automatic means, it can provide new insights about the explored domain.

Modularity eases the maintenance of the classification model. Since new data is being collected all the time, it is essential once in a while to execute a rebuild process to the entire model. However, if the model is built from several sub-models, and the new data collected affects only part of the sub-models, a more simple re-building process may be sufficient. This justification has often been noted [38].

If there are no dependencies between the various sub-components, then parallel techniques can be applied. By using parallel computation, the time needed to solve a mining problem can be shortened.

Decomposition methodology suggests the ability to use different inducers for individual sub-problems or even to use the same inducer but with a different setup. For instance, it is possible to use neural networks having different topologies (different number of hidden nodes). The researcher can exploit this freedom of choice to boost classifier performance.

The first three advantages are of particular importance in commercial and industrial data mining. However, as it will be demonstrated later, not all decomposition methods display the same advantages.

3 The Elementary Decomposition Taxonomy

Finding an optimal or quasi-optimal decomposition for a certain supervised learning problem might be hard or impossible. For that reason the *elementary decomposition methodology* have been proposed [43]. The basic idea is to develop a meta-algorithm that recursively decomposes a classification problem using elementary decomposition methods. We use the term "elementary decomposition" to describe a type of simple decomposition that

can be used to build up a more complicated decomposition. Given a certain problem, we first select the most appropriate elementary decomposition to that problem. A suitable decomposer then decomposes the problem, and finally a similar procedure is performed on each sub-problem. This approach agrees with the "no free lunch theorem", namely if one decomposition is better than another in some domains, then there are necessarily other domains in which this relationship is reversed.

For implementing this decomposition methodology, one might consider the following issues:

- What type of elementary decomposition methods exist for classification inducers?
- Which elementary decomposition type performs best for which problem? What factors should one take into account when choosing the appropriate decomposition type?
- Given an elementary type, how should we infer the best decomposition structure automatically?
- How should the sub-problems be re-composed to represent the original concept learning?
- How can we utilize prior knowledge for improving decomposing methodology?

Figure 1 suggests an answer to the first issue. This figure illustrates a novel approach for arranging the different elementary types of decomposition in supervised learning.

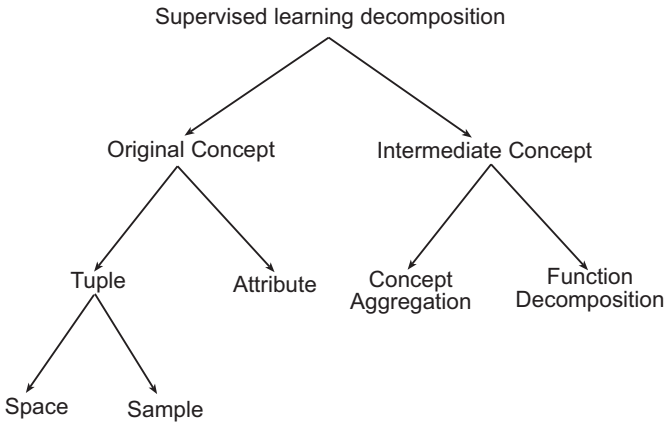

Fig. 1 Elementary Decomposition Methods in Classification.

3.1 Intermediate Concept Decomposition

In *intermediate concept* decomposition, instead of inducing a single complicated classifier, several sub-problems with different and more simple con-

cepts are defined. The intermediate concepts can be based on an aggregation of the original concept's values (*concept aggregation*) or not (*function decomposition*).

Classical concept aggregation replaces the original target attribute with a function, such that the domain of the new target attribute is smaller than the original one.

Concept aggregation has been used to classify free text documents into predefined topics [11]. This paper suggests breaking the topics up into groups (co-topics). Instead of predicting the document's topic directly, the document is first classified into one of the co-topics. Another model is then used to predict the actual topic in that co-topic.

The *Error-Correcting Output Coding* (ECOC) is a general concept aggregation algorithm which decomposes multi-class problems into multiple, two-class problems [15]. A classifier is built 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. The idea to decompose a K class classification problems into K two class classification problems [2]. Each problem considers the discrimination of one class to the other classes. The last method can be extended for manipulating the data based on the class relations among training data [41]. By using this method, they divide a K class classification problem into a series of $K(K-1)/2$ two-class problems where each problem considers the discrimination of one class to each one of the other classes. They have examined this idea using neural networks.

The round-robin classification problem (pairwise classification) is a technique for handling multi-class problems, in which one classifier is constructed for each pair of classes [20]. Empirical study has showed that this method can potentially improve classification accuracy.

Function decomposition was originally developed in the Fifties and Sixties for designing switching circuits. It was even used as an evaluation mechanism for checker playing programs [63]. This approach was later improved [8]. Recently, the machine learning community has adopted this approach. A manual decomposition of the problem and an expert-assisted selection of examples to construct rules for the concepts in the hierarchy was studied [45]. In comparison with standard decision tree induction techniques, structured induction exhibits about the same degree of classification accuracy with the increased transparency and lower complexity of the developed models.

A general-purpose function decomposition approach for machine learning has also been developed [73]. According to this approach, attributes are transformed into new concepts in an iterative manner and create a hierarchy of concepts. It is also possible to use a different function decomposition known as bi-decomposition [40].

3.2 Original Concept Decomposition

Original Concept decomposition means dividing the original problem into several sub-problems by partitioning the training set into smaller training sets. A classifier is trained on each sub-sample seeking to solve the original problem. Note that this resembles ensemble methodology but with the following distinction: each inducer uses only a portion of the original training set and ignores the rest. After a classifier is constructed for each portion separately, the models are combined in some fashion, either at learning or classification time.

There are two obvious ways to break up the original dataset: tuple-oriented or attribute (feature) oriented. 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 sub-spaces. Each sub-space is considered independently and the total model is a (possibly soft) union of such simpler models.

Space decomposition also includes the divide and conquer approaches such as mixtures of experts, local linear regression, CART/MARS, adaptive subspace models, etc., [31, 32, 54, 27].

Feature set decomposition (also known as attribute set decomposition) generalizes the task of feature selection which is extensively used in data mining. Feature selection aims to provide a representative set of features from which a classifier is constructed. On the other hand, in feature set decomposition, the original feature set is decomposed into several subsets. An inducer is trained upon the training data for each subset independently, and generates a classifier for each one. Subsequently, an unlabelled instance is classified by combining the classifications of all classifiers. This method potentially facilitates the creation of a classifier for high dimensionality data sets because each sub-classifier copes with only a projection of the original space.

In the literature there are several works that fit the feature set decomposition framework. However, in most of the papers the decomposition structure was obtained ad-hoc using prior knowledge. Moreover, it was argued that: "*There exists no algorithm or method susceptible to perform a vertical self-decomposition without a-priori knowledge of the task!*" [61].

The feature set decomposition algorithm known as MFS (Multiple Feature Subsets) combines multiple nearest neighbor classifiers, each using only a subset of random features [4]. Experiments show MFS can improve the standard nearest neighbor classifiers. This procedure resembles the well-known bagging algorithm [10]. However, instead of sampling instances with replacement, it samples features without replacement.

Additional alternative is to group the features according to the attribute type: nominal value features, numeric value features and text value features

[38]. A similar approach was used for developing the linear-bayes classifier [21]. The basic idea consists of aggregating the features into two subsets: the first subset containing only the nominal features and the second subset only the continuous features.

An approach for constructing an ensemble of classifiers using rough set theory was presented by Hu [29]. Although Hu's work refers to ensemble methodology and not decomposition methodology, it is still relevant for this case, especially as the declared goal was to construct an ensemble such that different classifiers use different attributes as much as possible. According to Hu, diversified classifiers lead to uncorrelated errors, which in turn improve classification accuracy. The method searches for a set of reducts, which include all the indispensable attributes. A reduct represents the minimal set of attributes which has the same classification power as the entire attribute set.

The feature set can be decomposed according to the target class [68]. For each class, the features with low correlation relating to that class have been removed. This method has been applied on a feature set of 25 sonar signals where the target was to identify the meaning of the sound (whale, cracking ice, etc.).

Feature set decomposition has been used for radar volcanoes recognition [14]. In this case, a feature set of 119 features was manually decomposed into 8 subsets. Features that are based on different image processing operations were grouped together. As a consequence, for each subset, four neural networks with different sizes were built.

The feature set decomposition was proved to be beneficial in many other applications, such as text-independent speaker identification [13], truck backer-upper problem [30] and quality problem in manufacturing plants [42].

The co-training paradigm for learning with labelled and unlabelled data can be considered as a feature set decomposition for classifying Web pages [9]. Co-training is useful when there is a large data sample, of which only a small part is labelled. In many applications, unlabelled examples are significantly easier to collect than labelled ones. This is especially true when the labelling process is time-consuming or expensive, such as in medical applications. According to the co-training paradigm, the input space is divided into two different views (i.e. two independent and redundant sets of features). For each view, a different classifier is built to classify unlabelled data. The newly labelled data of each classifier is then used to retrain the other classifier. It has been shown, both empirically and theoretically, that unlabelled data can be used to augment labelled data.

Another alternative for decomposing the feature set is as follows [39]: All input features are initially grouped by using a hierarchical clustering algorithm based on pairwise mutual information, with statistically similar features assigned to the same group. As a consequence, several feature subsets are constructed by selecting one feature from each group. A neural

network is subsequently constructed for each subset. All networks are then combined.

In the statistics literature, the most well-known decomposition algorithm is the MARS algorithm [19]. In this algorithm, a multiple regression function is approximated using linear splines and their tensor products. It has been shown that the algorithm performs an ANOVA decomposition, namely the regression function is represented as a grand total of several sums. The first sum is of all basic functions that involve only a single attribute. The second sum is of all basic functions that involve exactly two attributes, representing (if present) two-variable interactions. Similarly, the third sum represents (if present) the contributions from three-variable interactions, and so on.

Other works on feature set decomposition have been developed by extending the Naïve Bayes classifier. The Naïve Bayes classifier [16] uses the Bayes' rule to compute the conditional probability of each possible class, assuming the input features are conditionally independent given the target feature. Due to the conditional independence assumption, this method is called "Naïve". Nevertheless, a variety of empirical researches show surprisingly that the Naïve Bayes classifier can perform quite well compared to other methods, even in domains where clear feature dependencies exist [16]. Furthermore, Naïve Bayes classifiers are also very simple and easy to understand [36].

Recently, a new general framework that searches for helpful feature set decomposition structures has been proposed [58]. This framework nests many algorithms, two of which are tested empirically over a set of benchmark datasets. The first algorithm performs a serial search while using a new Vapnik-Chervonenkis dimension bound for multiple oblivious trees as an evaluating schema. The second algorithm performs a multi-search while using wrapper evaluating schema. This work indicates that feature set decomposition can increase the accuracy of decision trees.

4 The Decomposer's Characteristics

The following sub-sections present the main properties that characterize decomposers. These properties can be useful for differentiating between various decomposition frameworks.

4.1 The Structure Acquiring Method

This important property indicates how the decomposition structure is obtained:

– Manually (explicitly) based on an expert's knowledge in a specific domain [9, 45]. If the origin of the dataset is a relational database, then the schema's structure may imply the decomposition structure.

- Predefined due to some restrictions (as in the case of distributed data mining)
- Arbitrarily [17, 12] - The decomposition is performed without any profound thought. Usually, after setting the size of the subsets, members are randomly assigned to the different subsets.
- Induced without human interaction by a suitable algorithm [73].

Some may justifiably claim that searching for the best decomposition might be time-consuming, namely prolonging the induction process. In order to avoid this disadvantage, the complexity of the decomposition algorithms should be kept as small as possible. However, even if this cannot be accomplished, there are still important advantages, such as better comprehensibility and better performance that makes decomposition worth the additional computational complexity.

Furthermore, it should be noted that in an ongoing induction effort (like in a churning application) searching for the best decomposition structure might be performed in wider time buckets (for instance, once a year) than when training the classifiers (for instance once a week). Moreover, for acquiring decomposition structure, only a relatively small sample of the training set may be required. Consequently, the execution time of the decomposer will be relatively small compared to the time needed to train the classifiers.

Usually in real-life applications the decomposition is performed manually by incorporating business information into the modelling process. The following quotation provides a practical example [6]:

> It may be known that platinum cardholders behave differently from gold cardholders. Instead of having a data mining technique figure this out, give it the hint by building separate models for the platinum and gold cardholders.

Decomposition can be also useful for handling missing data. In this case we do not refer to sporadic missing data but to the case where several attribute values are available for some tuples but not for all of them. For instance: "Historical data, such as billing information, is available only for customers who have been around for a sufficiently long time" or "Outside data, such as demographics, is available only for the subset of the customer base that matches"). In this case, one classifier can be trained for customers having all the information and a second classifier for the remaining customers [6].

4.2 The Mutually Exclusive Property

This property indicates whether the decomposition is mutually exclusive (*disjointed decomposition*) or partially overlapping (i.e. a certain value of a certain attribute in a certain tuple is utilized 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 [17, 12]. Other have

used non-exclusive feature decomposition [4]. Similarly CART and MARS perform mutually exclusive decomposition of the input space, while HME allows sub-spaces to overlap.

The partially overlapping decompositions are potentially more accurate than mutually exclusive decompositions, because the latter forms a restriction on the problem space which might skip on accurate models. Still mutually exclusive decomposition has some important and helpful properties:

- A greater tendency in reduction of execution time than non-exclusive approaches. Since most learning algorithms have computational complexity that is greater than linear in the number of attributes or tuples, partitioning the problem dimensionality in a mutually exclusive manner means a decrease in computational complexity [51].
- Since mutual exclusiveness entails using smaller datasets, the models obtained for each sub-problem are smaller in size. Without the mutually exclusive restriction, each model can be as complicated as the model obtained for the original problem. Smaller models contribute to comprehensibility and ease in maintaining the solution.
- Mutually exclusive decomposition may help avoid some error correlation problems that characterize non-mutually exclusive decompositions [4]. However, mutually exclusive training sets do not necessarily result in low error correlation [66]. This point is true when each sub-problem is representative (i.e. represent the entire problem, as in sample decomposition).
- Reduced tendency to contradiction between sub-models. When a mutually exclusive restriction is unenforced, different models might generate contradictive classifications using the same input. Reducing inter-models contraindications help us to grasp the results and to combine the sub-models into one model. The resulting predictions of ensemble methods are usually inscrutable to end-users, mainly due to the complexity of the generated models, as well as the obstacles in transforming theses models into a single model [56]. Moreover, since these methods do not attempt to use all relevant features, the researcher will not obtain a complete picture of which attribute actually affects the target attribute, especially when, in some cases, there are many relevant attributes.
- Since the mutually exclusive approach encourages smaller datasets, they are more feasible. Some inducers can process only limited dataset size (for instance when the program requires that the entire dataset will be stored in the main memory). The mutually exclusive approach can make certain that inducers are fairly scalable to large data sets [12, 51].
- We claim that end-users can grasp mutually exclusive decomposition much easier than many other methods currently in use. For instance, boosting, which is a well-known ensemble method, distorts the original distribution of instance space, a fact that non-professional users find hard to grasp or understand.

4.3 The Inducer Usage

This property indicates the relation between the decomposer and the inducer used. Some decomposition implementations are "inducer-free", namely they do not use intrinsic inducers at all. Usually the decomposition procedure needs to choose the best decomposition structure among several structures that it considers. In order to measure the performance of a certain decomposition structure, there is a need to realize the structure by building a classifier for each component. However since "inducer-free" decomposition does not use any induction algorithm, it uses a frequency table of the Cartesian product of the feature values instead. Consider the following example. The training set consists of four binary input attributes (a_1, a_2, a_3, a_4) and one target attribute (y). Assume that an "inducer-free" decomposition procedure examines the following feature set decomposition: (a_1, a_3) and (a_2, a_4). In order to measure the classification performance of this structure, it is required to build two classifiers; one classifier for each subset. In the absence of an induction algorithm, two frequency tables are built; each table has $2^2 = 4$ entries representing the Cartesian product of the attributes in each subset. For each entry in the table, we measure the frequency of the target attribute. Each one of the tables can be separately used to classify a new instance x: we search for the entry that corresponds to the instance x and select the target value with the highest frequency in that entry. This "inducer-free" strategy has been used in several places. For instance the extension of Naïve Bayes suggested can be considered as a feature set decomposition with no intrinsic inducer [16]. The function decomposition algorithm developed by using sparse frequency tables also fits this strategy [73].

Other implementations are considered as an "inducer-dependent" type, namely these decomposition methods use intrinsic inducers, and they have been developed specifically for a certain inducer. They do not guarantee effectiveness in any other induction method. For instance, some works have been developed specifically for neural networks [41] or decision trees [58].

The third type of decomposition method is the "inducer-independent" type. These implementations can be performed on any given inducer, however, the same inducer is used in all subsets. As opposed to the "inducer-free" implementation, which does not use any inducer for its execution, "inducer-independent" requires the use of an inducer. Nevertheless, it is not limited to a specific inducer like the "inducer-dependent".

The last type is the "inducer-chooser" type, which, given a set of inducers, the system uses the most appropriate inducer on each sub-problem.

4.4 Exhaustiveness

This property indicates whether all data elements should be used in the decomposition. For instance, an exhaustive feature set decomposition refers to the situation in which each feature participates in at least one subset.

4.5 Combiner Usage

This property specifies the relation between the decomposer and the combiner. Some decomposers are combiner-dependent. That is to say they have been developed specifically for a certain combination method like voting or Naïve Bayes. Other decomposers are combiner-independent; the combination method is provided as input to the framework. Potentially there could be decomposers that, given a set of combiners, would be capable of choosing the best combiner in the current case.

4.6 Sequentially or Concurrently

This property indicates whether the various sub-classifiers are built sequentially or concurrently. In sequential framework the outcome of a certain classifier may effect the creation of the next classifier. On the other hand, in concurrent framework each classifier is built independently and their results are combined in some fashion. Some refers to this property as "The relationship between modules" [65] and distinguishes between three different types: successive, cooperative and supervisory. Roughly speaking the "successive" refers to "sequential" while "cooperative" refers to "concurrent". The last type applies to the case in which one model controls the other model, for instance, one neural network is used to tune another neural network.

The original problem in *intermediate concept decomposition* is usually converted to a sequential list of problems, where the last problem aims to solve the original one. On the other hand, in *original concept decomposition* the problem is usually divided into several sub-problems which exist on their own. Nevertheless, there are some exceptions. For instance, the "windowing" concept [52] is considered to be sequential.

Naturally there might be other important properties which can be used to differentiate a decomposition scheme. Table 1 summarizes the most relevant research performed on each decomposition type.

Table 1 Summary of Decomposition Methods in the Literature.

Paper	Decomposition Type	Mutually Exclusive	Structure Acquiring Method
[2]	Concept	No	Arbitrarily
[11]	Concept	Yes	Manually
[45]	Function	Yes	Manually
[73]	Function	Yes	Induced
[1]	Sample	No	Arbitrarily
[17]	Sample	Yes	Arbitrarily
[59]	Sample	Yes	Induced
[54]	Space	No	Induced
[35]	Space	Yes	Induced
[4]	Attribute	No	Arbitrarily
[38]	Attribute	Yes	Manually

5 Meta Decomposer Framework

As stated above, our ultimate goal is to develop a mechanism that combines all decomposition methods such that given an unseen dataset; the most appropriate decomposition (if any) could be selected. There are two alternatives to achieve this automatic and systematic decomposition procedure:

- The wrapper approach – Given a certain dataset, use each elementary decomposition and select the one that appears to give the highest success rate. The main advantage of this approach is its ability to predict quite well the performance of each examined method. The main disadvantage of this method is it's prolonged processing time. For some inducers the induction times may be very long, particularly in large real-life datasets. Several researchers have implemented this method for selecting induction algorithms or dimension reduction algorithms and showed that it produces superior results [64,34].
- The meta-learning approach – Based on datasets characteristics, the meta-decomposer decides whether to decompose the problem or not and what elementary decomposition to use. The idea of the meta-decomposer approach can be summarized as follows: If a certain decomposition method outperforms other decomposition methods in a particular dataset, then one should expect that this method will be preferable when other problems with similar characteristics are presented. For this purpose one can employ meta-learning. Meta-learning is concerned with accumulating experience on the performance of multiple applications of a learning system. One possible output of the meta-learning process is a meta-classifier that is capable to indicate which learning method is most appropriate to a given problem. In this paper the meta-learning focuses on explaining what causes a decomposition method to be successful or not in a particular problem. Thus, in this case the meta-classifier is used a meta-decomposer which attempts to select the most appropriate decomposition method . This goal can be accomplished by performing the following phases: In the first phase one should examine the performance of all investigated decomposition methods on various datasets. Upon examination of each dataset, the characteristics of the dataset are extracted. The dataset's characteristics, together with the indication of the most preferable decomposition method, (in this dataset) are stored in a meta-dataset. This meta-dataset reflects the experience accumulated across different datasets. In the second phase, an inducer can be applied to this meta-dataset to induce a meta-decomposer that can map a dataset to the most appropriate decomposition method (based on the characteristics of the dataset). In the last phase, the meta-decomposer is actually used to match a new unseen dataset to the most appropriate decomposition method.

This paper adopts the second alternative and examines it on real world problems. Previous works have already considered this approach for select-

ing the most appropriate induction algorithm given dataset characteristics (see for instance [22,69,5]). However, applying this methodology for selecting the most appropriate decomposition given a certain dataset, has not yet been considered. The main disadvantages of the meta-learning process concern the assumption that datasets with similar properties behave the same. Furthermore, in meta-learning, the amount of data available (dataset descriptions and different performances) is usually quite small, thus the meta-decomposer is based on small meta datasets. Nevertheless, the main advantage of this approach is that after the meta-learning phase is completed, it can be used to select the best decomposition in negligible processing time.

Figures 2 and 3 present the schematic framework of the meta-decomposer. Figure 2 presents the meta-data generation phase. Figure 3 presents (A) the meta-learning phase and (B) the usage of the meta-decomposer. As it can be seen in Figure 2, the Dataset-Generator component is responsible to extend the original datasets repository into a much bigger repository by manipulating the original datasets.

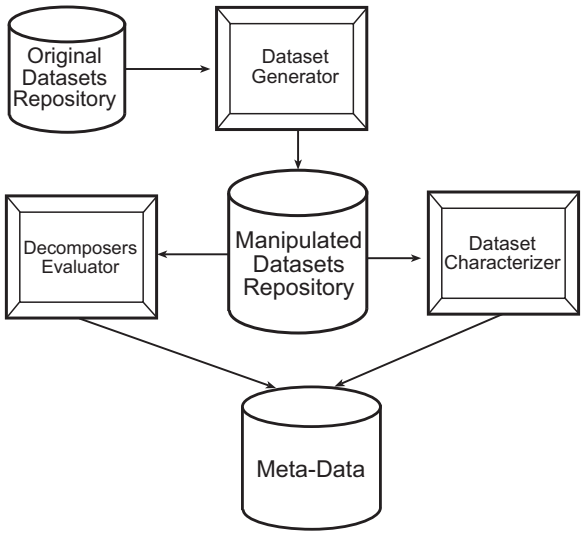

Fig. 2 Meta-Data Generation Phase.

5.1 Dataset Characterizer

It appears that datasets can be described using a vector of numeric values using certain features. It is possible to categorize the characteristic measures into the following types [22]:

– Simple measures (e.g. number of attributes, number of classes, proportion of binary, categorical or ordinal attributes).

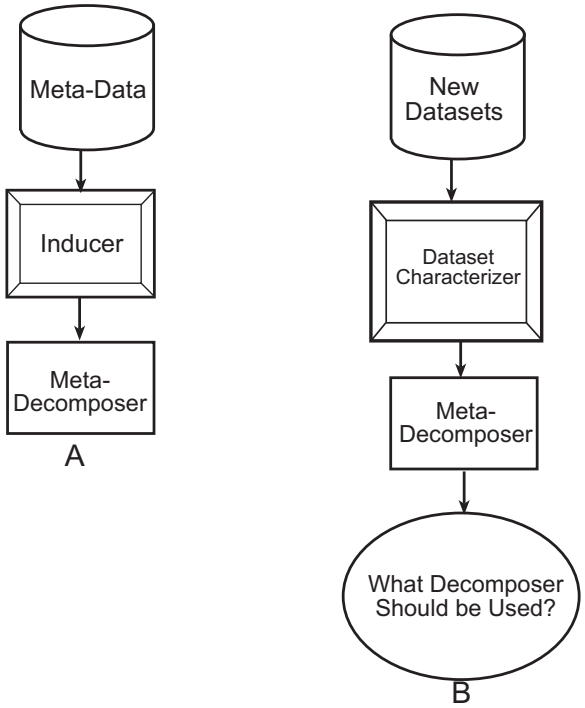

Fig. 3 (A) Meta Induction Phase Figure and (B) Meta-Decomposer Usage.

- Statistical measures (e.g. standard deviation ratio).
- Information based measures (e.g. mean entropy).

The Meta attributes used here are:

1. Number of instances in the training dataset.
2. Number of attributes.
3. Ratio of number of instances to the number of attributes - Potential for overfitting. If this value is small, inducers may find a classifier that adapts too well to the specificities of the training data, which may be caused by noisy or irrelevant attributes, and thus result in poor generalization performance.
4. Number of classes — The domain size of the target attribute.
5. Proportion of the binary attributes.
6. Proportion of the categorical attributes.
7. Proportion of the ordinal attributes.
8. Default accuracy — accuracy obtained when using the most frequent class in the training set, without building any classifier.
9. Mean of means — the mean of all numerical attributes means.
10. Mean of standard deviation — the mean of all numerical attributes standard deviations.

11. Mean Kurtosis - The mean of all numerical attributes kurtosis measure. Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution.
12. Mean Skewness — The mean of all numerical attributes skewness measure. Skewness is a measure of symmetry, or more precisely, the lack of symmetry.
13. Mean of entropies — mean of all attributes simple entropy. Entropy measures the amount of uncertainty of a random variable. The Shanon entropy measure resembles in many ways to the classic variance measurement. Both of them are measures of quantifying uncertainty changes. However, entropy is different from variance by its metric-free nature: It is dependent only on the probability distribution of a random variable and not on its values. However,the entropy measure is not expressive as much as the cumulative effect of all statistical measures. For this purpose it is possible to use one of the generalized entropy measures available in the literature. For this purpose we can use two generalizations proposed in the literature: Renyi [55] and Tsallis [67].
14. Average absolute correlation between numeric attributes: indicate robustness to irrelevant attributes.
15. Proportion of numeric attributes with outliers: Indicate robustness to outlying values. A certain attribute is considered to have outliers if the ratio of the variances of mean value and the α-trimmed mean (where $\alpha = 0.05$) is smaller than 0.7.
16. Average Gain Ratio — Average information gain ratio of the target attribute obtained by splitting the data according to each attribute. Useful as an indicative to the amount of relevant information embodied in the input attributes.

5.2 Dataset Manipulator

As stated above one of the main drawbacks of the meta-learning methodology is the necessity to induce from a very limited meta-dataset. The purpose of the Dataset Manipulator component is to extend the original repository of datasets into a much bigger repository and by that overcoming the limited meta-dataset problem.

Obviously the manipulation operators should efficiently affect the dataset characteristics in order to explore new options that are not represented in the original repository. The following simple operators can be suitable to this task:

- Projection – Randomly choose a subset of the original input attributes set and project the data according to it. This operation can have different levels of manipulation by setting a parameter that represents the subset size as a portion of the original attribute set. Note that this operation is disabled if the parameter is set to 100%.

- Selection – Randomly select a sub-sample of the original dataset. This operation can have different levels of manipulation by setting a parameter that represents the sub-sample size as a portion of the original dataset. Note that this operation is disabled if the parameter is set to 100%.
- Distortion – Changing the distribution of the target attribute by reassigning some of the instances to a different class. This operation can have different levels of manipulation by setting a parameter that represents the portion of instances that remain untouched. Note that this operation is disabled if the parameter is set to 100%.

Each manipulated dataset is obtained by performing these three operations. Note that there is no meaning to the order of manipulative operations.

5.3 Decomposer Evaluator

The aim of this component is to evaluate the performance of each decomposition method on each dataset in the manipulated repository. In this paper we use the C4.5 algorithm [52] as the internal inducer. Furthermore, we check the performance of C4.5 without performing any decomposition at all. Each manipulated dataset is represented in the meta-dataset as a single tuple, where its target value is chosen to be the method's name having the highest performance (from accuracy perspective) for this dataset. The performance is evaluated by averaging the results obtained by 10-fold-cross-validation procedure repeated 5 times.

6 Experimental Study

In this section we examine the suggested meta-decomposition approach. For this purpose we use 25 datasets from the UCI repository [44].

For each manipulation operation we have examined four different levels of manipulation: 100%, 90%, 75% and 50%. This results in 64 different combinations (manipulated datasets) for every dataset in the original repository. Because the original dataset repository contains 25 datasets, the manipulated repository consists of 1600 datasets.

In this paper we compare only three decomposition methods: Feature set decomposition using DOG algorithm[58], Space Decomposition using K-Classifier algorithm[60] and Sample Decomposition using Cluster Based Concurrent algorithm[59]. All algorithms were executed using their default parameters. Obviously there are several limitations in the methodology presented above. First, the results could be altered if the algorithms' parameters are tuned differently. Second, there is no guarantee that these algorithms precisely represent the corresponded decomposition method, namely if different decomposition algorithms had been employed here the results could be different.

Figure 4 presents the decision tree obtained by employing the C4.5 algorithm on meta-data (the tree has been slightly modified for legibility). It can be seen that space decomposition is useful when there are numeric attributes. This makes sense as the employed algorithm uses the K-Means algorithm which is more suitable to numeric instance space. The Instance-Attribute Ratio Meta attribute is used to differentiate between ATTRIBUTE (Attribute Decomposition) and NONE (not performing decomposition at all) in the second and third leaves. In this case, if the Instance-Attribute Ratio is below a 78.77 (namely there are many attributes relatively to the dataset size), then attribute decomposition should be applied. Another interesting observation is that both MeanEntropy2 and MeanTsallisEntropy2 (Tsallis' Entropy Measure) are found to be relevant. This indicates that the new proposed measure is not redundant.

```
num prop <= 0
  MeanEntropy2 <= 0.03 ⟶ ATTRIBUTE (362.0/57.0)
  MeanEntropy2 > 0.03
     Cat prop <= 0.67
        MeanTsallisEntropy2 <= 0.57
           Instance-Attribute Ratio <= 78.77
                    ⟶ ATTRIBUTE (336.0/135.0)
           Instance-Attribute Ratio > 78.77
                    ⟶ NONE (213.0/56.0)
        MeanTsallisEntropy2 > 0.57
                    ⟶ SAMPLE (425.0/235.0)
     Cat prop > 0.67 ⟶ NONE (328.0)
num count > 0 ⟶ SPACE (280.0/151)
```

Fig. 4 Meta-Decomposer Decision Tree.

6.1 Evaluating the Meta-Decomposer

To evaluate whether meta-decomposer brings some benefits, we have carried out the leave-one-out procedure. According to this procedure the meta-decomposer has been generated by learning from a partial meta-dataset obtained by removing one original dataset (and all its derived datasets). Then the obtained meta-decomposer has been evaluated on the dataset that has been left out. This procedure has been repeated 25 times, each time leaving out a different dataset. Table 2 presents the obtained results. The first column shows the dataset name. The second and third columns correspondingly present the actual best method and the method anticipated

by the meta-decomposer to be the most suitable method. The fourth, fifth and sixth columns present the accuracy obtained using C4.5, the actual best method and the anticipated method respectively. As it can be seen in almost two thirds of the datasets, the meta-decomposer was able to predict which decomposition method (if at all) outperforms other methods. In the remaining cases the performance of the anticipated method has been slightly less accurate than the actual best method with no statistical significance. Moreover, employing the meta-decomposer together with its corresponded decomposition method can improve on average the accuracy of C4.5 in 6.86 % ± 3.65% (while the best improvement that might be obtained by selecting the most suitable decomposition method is 7.07% ± 3.64%).

Table 2 Performance Results of Meta-Decomposer Procedure.

Dataset Name	Actual Best Method	Anticipated Best Method	Accuracy C4.5	Accuracy Actual Best Method	Accuracy Anticipated Best Method
AUST	Attribute	None	85.36±5.1	86.52±2.5	85.36±5.1
AUDIOLOGY	Attribute	ATT	75±6.95	78.5±6.54	78.5±6.54
BCAN	Attribute	None	92.99±2.87	97.42±1.17	92.99±2.87
HEPATITIS	Attribute	Attribute	70.32±8.46	80±6.89	80±6.89
IRIS	None	None	96±3.33	95.33±5.05	96±3.33
KR-VS-KP	Space	None	99.44±0.55	99.62±0.43	99.44±0.55
LABOR	Attribute	Attribute	87.72±12.72	98.24±4.52	98.24±4.52
LED17	Attribute	Attribute	59.09±6.9	73.64±5.5	73.64±5.5
LETTER	Space	Space	74.96±0.8	77.46±0.64	77.46±0.64
LCAN	Attribute	Attribute	38.71±17.82	93.55±10.05	93.55±10.05
MONKS1	Attribute	Attribute	75.81±8.2	98.39± 2.3	98.39± 2.3
MONKS2	Sample	Sample	61.54±8.6	65.36 ±5.7	65.36 ±5.7
MONKS3	Attribute	None	93.44±3.7	93.442±3.3	93.44±3.7
MUSH	None	Attribute	100±0	100±0	100±0
NURSE	None	Space	97.45±0.4	90.65±0.6	90.65±0.6
OPTIC	Attribute	Attribute	62.42±2	91.73±1.4	91.73±1.4
SONAR	Attribute	Attribute	69.71±5.4	77.12±8.7	77.12±8.7
SOYBEAN	Space	Attribute	92.83±1.52	94.9±4.61	92.9±2.56
SPI	Attribute	Attribute	91.2±1.9	95.8±0.9	95.8±0.9
TTT	None	Space	85.7±1.65	79.33±4	79.33±4
VOTE	None	None	96.21±2.45	90.52±1.23	96.21±2.45
WINE	Attribute	Attribute	85.96±6.9	96.63±3.9	96.63±3.9
ZOO	Attribute	Attribute	93.07±5.8	98.02±3.02	98.02±3.02
ARCENE	Sample	None	75 ±9.2	79±8.1	75 ±9.2
DEXTER	Attribute	Attribute	78.33 ±3.6	89.33±2.7	89.33±2.7
MADELON	Attribute	Attribute	69.8±4.7	71.4 ±2.6	71.4 ±2.6

7 The Relation to Other Methodologies

The main distinction between existing approaches, such as ensemble methods and distributed data mining to decomposition methodology, focuses on the following fact: the assumption that each model has access to a comparable quality of data is not valid in the decomposition approach [68]. In decomposition methodology classifiers may have significant variations in their overall performance. Furthermore when individual classifiers have substantially different performances over different parts of the input space, combining is still desirable [68]. Nevertheless neither simple combiners nor more sophisticated combiners are particularly well-suited for the type of problems that arise.

The ensemble methodology is closely related to the decomposition methodology. In both cases the final model is a composite of multiple models combined in some fashion. However, it is possible to distinguish between these methodologies in the following way [65]: the main idea of ensemble methodology is to combine a set of models, each of which solves the same original task. The purpose of ensemble methodology is to obtain a more accurate and reliable performance than when using a single model. On the other hand, the purpose of decomposition methodology is to break down a complex problem into several manageable problems, enabling each inducer to solve a different task. Therefore, in ensemble methodology, any model can provide a sufficient solution to the original task. On the other hand, in decomposition methodology, a combination of all models is mandatory for obtaining a reliable solution.

Distributed data mining (DDM) deals with mining data that might be inherently distributed among different, loosely coupled sites with slow connectivity, such as geographically distributed sites connected over the Internet [33]. Usually DDM is categorized according to data distribution:

- Homogeneous – In this case, the datasets in all the sites are built from the same common set of attributes. This state is equivalent to the sample decomposition discussed above, when the decomposition structure is set by the environment.
- Heterogeneous – In this case, the quality and quantity of data available to each site may vary substantially. Since each specific site may contain data for different attributes, leading to large discrepancies in their performance, integrating classification models derived from distinct and distributed databases is complex.

DDM can be useful also in the case of "mergers and acquisitions" of corporations. In such cases, since each company involved may have its own IT legacy systems, different sets of data are available.

In DDM the different sources are given, namely the instances are predecomposed. As a result, DDM is mainly focused on combining the various methods. Several researchers discuss ways of leveraging distributed techniques in knowledge discovery, such as data cleaning and preprocessing, transformation, and learning.

The JAM system is a meta-learning approach [50]. The meta-learning approach is about combining several models (describing several sets of data from several sources of data) into one high-level model.

Another meta-learning concept is the *knowledge probing* [23]. In knowledge probing, supervised learning is organized into two stages. In the first stage, a set of base classifiers is constructed using the distributed data sets. In the second stage, the relationship between an attribute vector and the class predictions from all of the base classifiers is determined.

A closely related field is *Parallel Data Mining* (PDM). PDM deals with mining data by using several tightly-coupled systems with fast interconnection, as in the case of a cluster of shared memory workstations [71].

The main goal of PDM techniques is to scale-up the speed of the data mining on large datasets. It addresses the issue by using high performance, multi-processor computers. The increasing availability of such computers calls for extensive development of data analysis algorithms that can scale up as we attempt to analyze data sets measured in terabytes on parallel machines with thousands of processors. This technology is particularly suitable for applications that typically deal with large amounts of data, e.g. company transaction data, scientific simulation and observation data. Another important example of PDM is the SPIDER project that uses shared-memory multiprocessors systems (SMPs) to accomplish PDM on distributed data sets [72].

8 Conclusion and Future Work

In this paper we have reviewed the necessity of decomposition methodology in pattern recognition, machine learning and data mining. We have suggested an approach to categorize elementary decomposition methods. We also discussed the main characteristics of decomposition methods and demonstrated these characteristics on various methods presented in the literature.

Finally we proposed a meta-decomposition approach and validated its prediction capabilities. We have shown empirically that the proposed meta-decomposer usually select the most appropriate decomposition method. In fact using the meta-decomposer achieved an accuracy improvement of 6.8% while using the posteriori best method has provided only slightly better results (7.1%).

Additional research issues in meta decomposer approach include: Extending the meta-learning schema to investigate other decomposition methods presented in Section 3, more precisely: Function Decomposition and Concept Aggregation; Checking whether the meta-learning results are still valid when different decomposition algorithms implementations are used, namely examine the robustness of the meta-decomposer; Examine the effectiveness of recursively using the meta-decomposer; and finally how can we utilize prior knowledge for decomposition methodology.

9 Originality and Contribution

This paper introduces a novel taxonomy for decomposition methods as applied to classification tasks. The proposed taxonomy refers to elementary decomposition methods that can be used as building blocks to construct a more complicated decomposition. The taxonomy is illustrated using an extensive review of existing decomposition methods.

The taxonomy is subsequently used as the basis for a new meta-decomposition methodology which is designed to automatically select the best decomposition method for a given database. For this purpose we examine a meta-decomposer framework that contains two phases. In the first phase the meta-dataset is generated by evaluating the performance of several decomposition methods on a given set of datasets. Every dataset is characterized by a set of well-known measures, such as entropy and skewness. In order to significantly extend the dataset variety, we suggest to use a randomize dataset manipulator. In the second phase a decision tree-based meta-decomposer is trained by using the generated meta-dataset. Then when a new dataset is provided, we employ the meta-decomposer to select the most promising decomposition method.

10 Acknowledgment

The author would like to thank Prof. Oded Maimon for his helpful and inspiring comments.

References

1. Ali K. M., Pazzani M. J., Error Reduction through Learning Multiple Descriptions, Machine Learning, 24: 3, 173-202, 1996.
2. Anand R, Methrotra K, Mohan CK, Ranka S. Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Networks, 6(1): 117-125, 1995.
3. Baxt, W. G., Use of an artificial neural network for data analysis in clinical decision making: The diagnosis of acute coronary occlusion. Neural Computation, 2(4):480-489, 1990.
4. Bay, S., Nearest neighbor classification from multiple feature subsets. Intelligent Data Analysis, 3(3): 191-209, 1999.
5. Bensusan H. and Kalousis A., Estimating the Predictive Accuracy of a Classifier, In Proc. Proceedings of the 12th European Conference on Machine Learning, pages 25-36, 2001.
6. Berry M., and Linoff G., Mastering Data Mining, John Wiley & Sons, 2000.
7. Bhargava H. K., Data Mining by Decomposition: Adaptive Search for Hypothesis Generation, INFORMS Journal on Computing Vol. 11, Iss. 3, pp. 239-47, 1999.
8. Biermann, A. W., Faireld, J., and Beres, T., 1982. Signature table systems and learning. IEEE Trans. Syst. Man Cybern., 12(5):635-648.

9. Blum A., and Mitchell T., Combining Labeled and Unlabeled Data with Co-Training. In Proc. of the 11th Annual Conference on Computational Learning Theory, pages 92-100, 1998.
10. Breiman L., Bagging predictors, Machine Learning, 24(2):123-140, 1996.
11. 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.
12. Chan P.K. and Stolfo S.J, On the Accuracy of Meta-learning for Scalable Data Mining, J. Intelligent Information Systems, 8:5-28, 1997.
13. Chen K., Wang L. and Chi H., Methods of Combining Multiple Classifiers with Different Features and Their Applications to Text-Independent Speaker Identification, International Journal of Pattern Recognition and Artificial Intelligence, 11(3): 417-445, 1997.
14. Cherkauer, K.J., Human Expert-Level Performance on a Scientific Image Analysis Task by a System Using Combined Artificial Neural Networks. In Working Notes, Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms Workshop, Thirteenth National Conference on Artificial Intelligence. Portland, OR: AAAI Press, 1996.
15. Dietterich, T. G., and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2:263-286, 1995.
16. Domingos, P., & Pazzani, M., On the Optimality of the Naive Bayes Classifier under Zero-One Loss, Machine Learning, 29: 2, 103-130, 1997.
17. Domingos, P., Using Partitioning to Speed Up Specific-to-General Rule Induction. In Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models, pp. 29-34, AAAI Press, 1996.
18. Fischer, B., "Decomposition of Time Series - Comparing Different Methods in Theory and Practice", Eurostat Working Paper, 1995.
19. Friedman, J. H., "Multivariate Adaptive Regression Splines", The Annual Of Statistics, 19, 1-141, 1991.
20. Fürnkranz, J., More efficient windowing, In Proceeding of The 14th national Conference on Artificial Intelegence (AAAI-97), pp. 509-514, Providence, RI. AAAI Press, 1997.
21. Gama J., A Linear-Bayes Classifier. In C. Monard, editor, Advances on Artificial Intelligence – SBIA2000. LNAI 1952, pp 269-279, Springer Verlag, 2000
22. Giraud–Carrier C., Vilalta R., Brazdil P., Introduction to the Special Issue of on Meta-Learning, Machine Learning, 54 (3), 197-194, 2004.
23. Guo Y. and Sutiwaraphun J., Knowledge probing in distributed Data Mining, in Proc. 4h Int. Conf. Knowledge Discovery Data Mining, pp 61-69, 1998.
24. Hampshire, J. B., and Waibel, A. The meta-Pi network - building distributed knowledge representations for robust multisource pattern-recognition. Pattern Analyses and Machine Intelligence 14(7): 751-769, 1992.
25. Hansen J., Combining Predictors. Meta Machine Learning Methods and Bias, Variance & Ambiguity Decompositions. PhD dissertation. Aurhus University. 2000.
26. He D. W., Strege B., Tolle H., and Kusiak A., Decomposition in Automatic Generation of Petri Nets for Manufacturing System Control and Scheduling, International Journal of Production Research, 38(6): 1437-1457, 2000.
27. Holmstrom, L., Koistinen, P., Laaksonen, J., and Oja, E., Neural and statistical classifiers - taxonomy and a case study. IEEE Trans. on Neural Networks, 8,:5–17, 1997.

28. Hrycej T., Modular Learning in Neural Networks. New York: Wiley, 1992.
29. Hu, X., Using Rough Sets Theory and Database Operations to Construct a Good Ensemble of Classifiers for Data Mining Applications. ICDM01. pp 233-240, 2001.
30. Jenkins R. and Yuhas, B. P. A simplified neural network solution through problem decomposition: The case of Truck backer-upper, IEEE Transactions on Neural Networks 4(4):718-722, 1993.
31. Johansen T. A. and Foss B. A., A narmax model representation for adaptive control based on local model -Modeling, Identification and Control, 13(1):25-39, 1992.
32. Jordan, M. I., and Jacobs, R. A., Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6, 181-214, 1994.
33. Kargupta, H. and Chan P., eds, Advances in Distributed and Parallel Knowledge Discovery , pp. 185-210, AAAI/MIT Press, 2000.
34. Kohavi R. and John G., The Wrapper Approach, In Feature Extraction, Construction and Selection: A Data Mining Perspective, H. Liu and H. Motoda (eds.), Kluwer Academic Publishers, 1998.
35. Kohavi R., Becker B., and Sommerfield D., Improving simple Bayes. In Proceedings of the European Conference on Machine Learning, 1997.
36. Kononenko, I., Comparison of inductive and Naive Bayes learning approaches to automatic knowledge acquisition. In B. Wielinga (Ed.), Current Trends in Knowledge Acquisition, Amsterdam, The Netherlands IOS Press, 1990.
37. Kusiak, E. Szczerbicki, and K. Park, A Novel Approach to Decomposition of Design Specifications and Search for Solutions, International Journal of Production Research, 29(7): 1391-1406, 1991.
38. Kusiak, A., "Decomposition in Data Mining: An Industrial Case Study", IEEE Transactions on Electronics Packaging Manufacturing, Vol. 23, No. 4, pp. 345-353, 2000.
39. Liao Y., and Moody J., Constructing Heterogeneous Committees via Input Feature Grouping, in Advances in Neural Information Processing Systems, Vol.12, S.A. Solla, T.K. Leen and K.-R. Muller (eds.),MIT Press, 2000.
40. Long C., Bi-Decomposition of Function Sets Using Multi-Valued Logic, Eng. Doc. Dissertation, Technischen Universitat Bergakademie Freiberg 2003.
41. Lu B.L., Ito M., Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification, IEEE Trans. on Neural Networks, 10(5):1244-1256, 1999.
42. Maimon O. and Rokach L., "Data Mining by Attribute Decomposition with semiconductors manufacturing case study", In "Data Mining for Design and Manufacturing: Methods and Applications", Kluwer Academic Publishers, pp. 311-336, 2001.
43. Maimon O. and Rokach L., "Decomposition Methodology for Knowledge Discovery and Data Mining: Theory and Applications", World Scientific, 2005.
44. 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.
45. Michie, D., Problem decomposition and the learning of skills, in Proceedings of the European Conference on Machine Learning, pp. 17-31, Springer-Verlag, 1995.
46. Nowlan S. J., and Hinton G. E. Evaluation of adaptive mixtures of competing experts. In Advances in Neural Information Processing Systems, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., vol. 3, pp. 774-780, Morgan Kaufmann Publishers Inc., 1991.

47. Ohno-Machado, L., and Musen, M. A. Modular neural networks for medical prognosis: Quantifying the benefits of combining neural networks for survival prediction. Connection Science 9, 1, 1997, 71-86.
48. Peng, F. and Jacobs R. A., and Tanner M. A., Bayesian Inference in Mixtures-of-Experts and Hierarchical Mixtures-of-Experts Models With an Application to Speech Recognition, Journal of the American Statistical Association, 1995.
49. Pratt, L. Y., Mostow, J., and Kamm C. A., Direct Transfer of Learned Information Among Neural Networks, in: Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA, 584-589, 1991.
50. Prodromidis, A. L., Stolfo, S. J. and Chan, P. K., Effective and efficient pruning of metaclassifiers in a distributed data mining system. Technical report CUCS-017-99, Columbia Univ., 1999.
51. Provost, F.J. and Kolluri, V., A Survey of Methods for Scaling Up Inductive Learning Algorithms, Proc. 3rd International Conference on Knowledge Discovery and Data Mining, 1997.
52. Quinlan, J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, 1993.
53. Rahman, A. F. R., and Fairhurst, M. C. A new hybrid approach in combining multiple experts to recognize handwritten numerals. Pattern Recognition Letters, 18: 781-790,1997.
54. Ramamurti, V., and Ghosh, J., Structurally Adaptive Modular Networks for Non-Stationary Environments, IEEE Transactions on Neural Networks, 10 (1):152-160, 1999.
55. R'enyi A., Probability Theory, North-Holland, Amsterdam, 1970.
56. Ridgeway, G., Madigan, D., Richardson, T. and O'Kane, J., Interpretable Boosted Naive Bayes Classification, Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp 101-104, 1998.
57. Rokach L., Maimon O., "Decomposition Methodology for Classification Tasks", Proceedings of the IEEE International Conference on Granular Computing, Beijing, July 2005, IEEE Computer Society Press, ISBN: 0-7803-9017-2, pp 636-641.
58. Rokach L. and Maimon O., "Feature set decomposition for decision trees", Intelligent Data Analysis, 6(2):1-28, 2005.
59. Rokach L., Maimon O., Arad O., "Improving Supervised Learning by Sample Decomposition", International Journal of Computational Intelligence and Applications, 5(1):37-54, 2005.
60. Rokach L., Maimon O. and Lavi I., "Space Decomposition In Data Mining: A Clustering Approach", Proceedings of the 14th International Symposium On Methodologies For Intelligent Systems, Maebashi, Japan, Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 24–31.
61. Ronco, E., Gollee, H., and Gawthrop, P. J., Modular neural network and self-decomposition. CSC Research Report CSC-96012, Centre for Systems and Control, University of Glasgow, 1996.
62. Saaty, X., The analytic hierarchy process: A 1993 overview. Central European Journal for Operations Research and Economics, Vol. 2, No. 2, p. 119-137, 1993.
63. Samuel, A., Some studies in machine learning using the game of checkers II: Recent progress. IBM J. Res. Develop., 11:601-617, 1967.
64. Schaffer, C., Selecting a classification method by cross-validation. Machine Learning 13(1):135-143, 1993.

65. Sharkey, A., On combining artificial neural nets, Connection Science, Vol. 8, pp.299-313, 1996.
66. Sharkey, A., Multi-Net Iystems, In Sharkey A. (Ed.) Combining Artificial Neural Networks: Ensemble and Modular Multi-Net Systems. pp. 1-30, Springer
-Verlag, 1999.
67. Tsallis C., Possible Generalization of Boltzmann-Gibbs Statistics, J. Stat.Phys., 52, 479-487, 1988.
68. Tumer, K., and Ghosh J., Linear and Order Statistics Combiners for Pattern Classification, in Combining Articial Neural Nets, A. Sharkey (Ed.), pp. 127-162, Springer-Verlag, 1999.
69. Vilalta R., Giraud–Carrier C., Brazdil P., "Meta-Learning", in O. Maimon and L. Rokach (Eds.), Handbook of Data Mining and Knowledge Discovery in Databases, pp. 731-748, Springer, 2005.
70. Weigend, A. S., Mangeas, M., and Srivastava, A. N. Nonlinear gated experts for time-series - discovering regimes and avoiding overfitting. International Journal of Neural Systems 6(5):373-399, 1995.
71. Zaki, M. J., Ho C. T., Eds., Large- Scale Parallel Data Mining. New York: Springer- Verlag, 2000.
72. Zaki, M. J., Ho C. T., and Agrawal, R., Scalable parallel classification for Data Mining on shared- memory multiprocessors, in Proc. IEEE Int. Conf. Data Eng., Sydney, Australia, WKDD99, pp. 198– 205, 1999.
73. Zupan, B., Bohanec, M., Demsar J., and Bratko, I., Feature transformation by function decomposition, IEEE intelligent systems & their applications, 13: 38-43, 1998.