Fundamenta Informaticae 47 (2001) 91–117
IOS Press

Local Feature Selection with Dynamic Integration of Classifiers

Seppo Puuronen[†*]
Department of Computer Science and Information Systems
University of Jyväskylä
P.O. Box 35, FIN-40351 Jyväskylä, Finland
sepi@cs.jyu.fi

Alexey Tsymbal[†]
Department of Computer Science and Information Systems
University of Jyväskylä
P.O. Box 35, FIN-40351 Jyväskylä, Finland
alexey@cs.jyu.fi

Abstract. Multidimensional data is often feature space heterogeneous so that individual features have unequal importance in different sub areas of the feature space. This motivates to search for a technique that provides a strategic splitting of the instance space being able to identify the best subset of features for each instance to be classified. Our technique applies the wrapper approach where a classification algorithm is used as an evaluation function to differentiate between different feature subsets. In order to make the feature selection local, we apply the recent technique for dynamic integration of classifiers. This allows to determine which classifier and which feature subset should be used for each new instance. Decision trees are used to help to restrict the number of feature combinations analyzed. For each new instance we consider only those feature combinations that include the features present in the path taken by the new instance in the decision tree built on the whole feature set. We evaluate our technique on data sets from the UCI machine learning repository. In our experiments, we use the C4.5 algorithm as the learning algorithm for base classifiers and for

[*]We would like to thank the UCI machine learning repository of databases, domain theories and data generators for the data sets, and the machine learning library in C++ for the source code used in this study. We are grateful to the anonymous referees for their valuable comments and constructive criticism.
Address for correspondence: Department of Computer Science and Information Systems, University of Jyväskylä, P.O. Box 35, FIN-40351 Jyväskylä, Finland
[†]I would like to thank the COMAS Graduate School of the University of Jyväskylä for the financial support during this work.

the decision trees that guide the local feature selection. The experiments show some advantages of the local feature selection with dynamic integration of classifiers in comparison with the selection of one feature subset for the whole space.

Keywords: feature selection, ensemble of classifiers, dynamic integration, data mining, machine learning

1. Introduction

Current electronic data repositories contain enormous amount of data including also unknown and potentially interesting patterns and relations, which are tried to be revealed using knowledge discovery and data mining methods [12]. One approach commonly used is supervised machine learning, in which a set of training instances is used to train one or more classifiers that map the feature space of the instances into the set of class values [2]. Each training instance is usually represented by a vector of the values of the features and the class label of the instance. An induction algorithm is used to learn a classifier, which maps the space of the feature values into the set of the class values. The learned classifiers are later used to classify new instances with unknown class values.

Data, which is the subject of analysis and processing in knowledge discovery and data mining, is usually multidimensional, and presented by a number of features. One significant problem in data mining is the presence of irrelevant features. When several such features are present in the training instances, many machine learning algorithms become confused and this may result in severe degradation of classification accuracy. A natural solution to this problem is to identify the irrelevant features, and discard them before further use of the training set [17].

The multidimensional data is sometimes feature-space heterogeneous so that individual features have unequal importance in different sub areas of the whole feature space. Many methods have been proposed for the purpose of feature selection, but most of them ignore the fact that some features may be relevant only in a context (i.e. in some region of the feature space). This ignorance may result in discarding the features that are highly relevant in a restricted area of the feature space because this relevance is swamped by their irrelevance in all the other areas of the feature space. Ignorance may also consist in retaining the features that are relevant in most of the space, but still unnecessarily confuse the classifier in some sub areas of the feature space [11].

In this paper we describe a technique that searches for a division of the feature space trying to identify the best subsets of features for each instance. To make the feature selection local, we apply the recently developed technique for dynamic integration of classifiers to determine which classifier and which feature subset is applied for each new instance [22]. Our technique can be applied also in the case of implicit heterogeneity when the regions of heterogeneity cannot be easily defined by a simple dependency. We make experiments with well-known data sets of the UCI machine learning repository [6] using ensembles of simple base classifiers using only one or two features. Our results are promising and show some advantages of the local feature selection in comparison with the selection of one feature subset for the whole space.

In Section 2 we consider the problem of integrating multiple classifiers and discuss published research about the ensembles of classifiers. In Section 3 we consider our technique for dynamic integration of classifiers. Section 4 discusses local feature selection based on the dynamic classifier integration, and

in Sections 5-8 our experiments on a number of data sets are considered. We conclude briefly in Section 9 with a summary and further research topics.

2. An Ensemble of Classifiers

In this section we consider how to use an ensemble of classifiers and discuss related work published. The use of an ensemble of classifiers concerns two basic questions: (1) what kind of a set of classifiers should be generated?; and (2) how should the predictions of the base classifiers of an ensemble be integrated?. We discuss these topics including several insights that might have effect for future research directions related to these challenging questions.

2.1. Generation of an Ensemble of Classifiers

Several independent researchers have shown in experiments and analytically, that the more diverse and accurate the predictions of the base classifiers combined are, the better integration results can be achieved [19, 28, 35]. To generate a set of diverse and still accurate classifiers, several approaches have been tried.

One way of generating a set of diverse classifiers is to use learning algorithms with heterogeneous representation and search biases [19], such as decision trees, neural networks, instance-based learning, etc. For example, in Puuronen *et al.* [22] we use C4.5 decision tree learning, PEBLS instance learning algorithm, and Bayesian learning to generate base classifiers in the ensembles.

Another approach is to use classifiers with homogeneous representations that differ in their method of search or in the instances on which they are trained. Several different techniques have been proposed. The base classifiers may be learned from different subsets of the training data. For example, two well-known ensemble methods of this type are bagging and boosting [25]. Bagging builds each base classifier from a set of instances that are drawn from the training set with replacement using bootstrap sampling. On each draw, each training instance has an equal probability of being drawn. In this approach the construction of base classifiers is independent of one another. In contrast, boosting builds its base classifiers sequentially. On each trial a boosting technique draws instances from the training set following a probability distribution that insures that instances misclassified by a classifier constructed on a previous trial are more likely to be drawn for the current classifier. In Tsymbal & Puuronen [32] and Tsymbal [33] we have considered a combination of the bagging and boosting approaches for generating base classifiers with our approach for dynamic integration of classifiers.

Another technique for building classifiers with homogeneous representations consists of using different subsets of features for each classifier. For example, in Oza & Tumer [21] base classifiers are built on different feature subsets, where each feature subset includes features relevant for distinguishing one class label from the others (the number of base classifiers is equal to the number of classes). Building the base classifiers on different feature subsets to provide local feature selection using the dynamic classifier integration is the focus of this paper.

Also, natural randomisation in the process of model search (e.g., random weight setting for neural networks) can be used to build an ensemble of classifiers with homogeneous representations, or the randomisation can be injected artificially. For example, in Heath *et al.* [13] a randomised decision tree induction algorithm, which generates different decision trees every time it is run, was used for that purpose.

Sometimes, a combination of the above techniques can be useful in order to receive desired characteristics of an ensemble. For example, a combination of boosting and wagging (which is a kind of bagging technique) is considered in Webb [36].

2.2. Integration of Multiple Classifiers

Integration of multiple classifiers to improve classification results is currently an active research area in machine learning and neural networks communities. Dietterich [10] showed that this area is currently one of the four most important directions in machine learning research and presented a thorough overview of different approaches. Integration of an ensemble of classifiers has been shown to yield higher accuracy than the most accurate base classifier alone in different real-world tasks [28].

The challenging problem of the integration is to decide which one(s) of the classifiers to rely on or how to combine the results produced by the base classifiers. The problem of integration can be defined as follows. Let us suppose that there is a training set $\mathbf{T}$ and a set of classifiers $\mathbf{C}$. Let the training set $\mathbf{T}$ be $\{(\mathbf{x}_i, y_i),\ i = 1, ..., n\}$, where n is the number of the training instances, $\mathbf{x}_i = \{x_j\}, j = 1, ..., l$ is the vector of the attribute values of the i-th training instance (the values of the attributes are allowed to be continuous or nominal), and $y_i \in \{c_1, ..., c_k\}$ is the classification of the i-th instance, where k is the total number of the classes. Let the set of classifiers $\mathbf{C}$ be $\{C_1, ..., C_m\}$, where m is the number of the *base classifiers*. Each base classifier is either derived using some learning algorithm or a hand-crafted classifier constructed using some heuristic knowledge. A new instance $\mathbf{x}^*$ is an assignment of values to the vector of the attributes $\{x_j\}$. Let each base classifier be able to classify the new instance. Then the problem of integration of the classifiers is to derive a technique to classify the new instance $\mathbf{x}^*$ using the classifiers of the set $\mathbf{C}$.

Most existing classifier integration methods can be considered as instantiations of the *stacked generalization* (or *stacking*) framework [37] presented in Figure 1. The goal of stacking is to combine the predictions of the base classifiers based on information learned about their particular biases with respect to the training data. The basic idea is to have another layer of induction over the base classifications with the goal of learning how to correct for inappropriate biases [37].

In the basic form of the stacked generalization layered architecture, the base classifiers $\mathbf{C}$ form the first layer, and a single combining classifier M forms the second layer. The combining classifier M is trained using the predictions of the base classifiers $\{C_i(\mathbf{x}_j)\}$ and the training set T [37]. When a new instance $\mathbf{x}^*$is classified, then first, each of the base classifiers C_i produces its classification result $C_i(\mathrm{x}^*)$, and then the combining classifier M produces the final classification using these results. The technique for dynamic classifier integration used in this research can be also considered as an instantiation of the stacking. However, as the meta-level data, spatial information about errors of the base classifiers is used instead of the predictions of the base classifiers themselves.

Techniques using two basic approaches have been suggested as a solution to the integration problem. In the first approach, the classifications produced by the base classifiers are combined. Several effective techniques for *combination* of classifiers have been proposed. One of the most popular and simplest techniques to combine the results of the base classifiers is simple voting (also called majority voting and select all majority (SAM)) [5]. In the voting technique, the classification of each base classifier is considered as an equally weighted vote for that particular classification. Classification that receives the biggest amount of votes is selected as the final classification (ties are solved arbitrarily). Often, weighted voting is used: each vote receives a weight, which is usually proportional to the estimated generalization

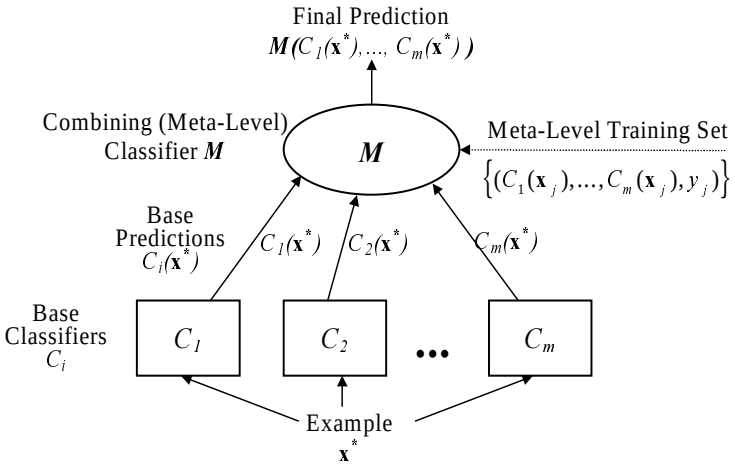

Figure 1. The stacked generalization framework.

performance of the corresponding classifier. Weighted voting works usually much better than simple majority voting [5].

More sophisticated combination techniques, which can be considered as instantiations of the stacking framework, include the SCANN method based on the correspondence analysis and the nearest neighbor search in the correspondence analysis results [19, 20]. Another technique combines minimal nearest neighbor classifiers within the stacked generalization framework [28]. Two effective classifier combination techniques called "arbiter" and "combiner" are also based on the stacked generalization [9]. Hierarchical classifier combination has also been considered. Experimental results of Chan [9] and Chan & Stolfo [8] showed that the hierarchical (multi-level) combination approach where the data set was distributed among a number of sites was often able to sustain the same level of accuracy as a global classifier trained on the entire data set.

There are still many open questions in integration of classifiers; even with such widely used architecture as stacked generalization. For example, there does not exist clear guidelines how to decide which base classifiers should be used, which features should be used to form the meta-level training set for the combining classifier, and which learning algorithm should be used to build the combining classifier. Different combining classifier algorithms have been considered by various researchers, including the boosting algorithm using weighted voting, ID3 for combining nearest neighbor classifiers [28], and the nearest neighbor classification [20] to search in the space of correspondence analysis results (not directly on the predictions).

Techniques of the second approach try to select the best base classifier from the ensemble. One of the most popular and simplest *selection* techniques is the cross-validation majority (CVM) [26, 15]. In the CVM, the cross-validation accuracy for each base classifier is estimated using the training set, and then the classifier with the highest accuracy is selected (ties are solved using voting). More sophisticated selection approaches include estimation of the local accuracy of the base classifiers by considering errors made in instances with similar attribute patterns [18], learning a number of meta-level classifiers ("referees") that predict whether the corresponding base classifiers are correct or not for new instances (for example each "referee" is a C4.5 tree that recognizes two classes) [16]. Todorovski & Dzeroski [31]

train a meta-level decision tree, which selects dynamically a base classifier to be applied for the considered instance, using the confidence level of the base classifiers in correct classification of the instance. Applications of classifier selection in medical diagnostics have been considered in Skrypnyk *et al.* [29], Terziyan *et al.* [30], and Tsymbal *et al.* [34]. They predict the local accuracy of the base classifiers by analyzing the accuracy in near-by instances of the training set.

The approaches of classifier selection techniques can be divided into two subsets: *static* and *dynamic* selection. The static approaches propose one "best" method for the whole data space, while the dynamic approaches take into account each new instance to be classified. The CVM is an example of the static approach, while the other selection techniques above and the one used in this article are examples of the dynamic approach.

Techniques for combination of classifiers can be static or dynamic as well. For example, widely used weighted voting [5] is a static approach. The weights for each base classifier's vote do not depend on the instance to be classified. In contrast, Dynamic Voting (DV) technique, which is in focus of this paper, is an example of dynamic combination approach. The weights for each base classifier's vote depend on the instance to be classified (the weights are proportional to estimated classifiers' local accuracies). Usually better data mining results can be achieved if the classifier integration is done dynamically taking into account characteristics of each new instance [18].

3. Dynamic Integration of Classifiers

In this section we discuss two variations of stacked generalization that use a metric to locally estimate the errors of the base classifiers. Rather than trying to train a meta-level classifier that predicts a class using the predictions of the base classifiers as in stacked generalization we propose to train a meta-level classifier that will predict the errors of the base classifiers for each new input instance. These errors are then utilized to produce a final classification. Based on our competence area assumption we use each base classifier just in the sub area of the instance space where it is the most reliable one. Thus we try to achieve overall results that are better than those of the best individual classifier alone.

Common solutions of the classification problem are based on the assumption that the entire space of features for a particular domain area consists of null-entropy areas, in other words, that instances of one class are not uniformly distributed, but concentrated in some sub areas [2]. In fact, the accuracy of a classification model depends on many factors as the size and shape of decision boundaries, the number of null-entropy areas (problem-depended factors), the completeness and cleanness of the training set (sample-depended factors) and certainly, on the individual peculiarities of the learning algorithm (algorithm-depended factors). Instances misclassified by a classifier are not uniformly distributed and the whole space can be regarded to consist of a set of null-entropy areas with categories "a classifier gives correct classification" and "a classifier gives incorrect classification". The dynamic approach to integration of multiple classifiers attempts to create a meta-model according to this vision. This meta-model is then used to predict the errors of the base classifiers in new instances.

Our approach contains two phases: the learning phase and the application phase. In the learning phase a performance matrix is formed and stacked. It includes information about the performance of each base classifier in each instance of the training set. In the application phase the combining classifier is used to predict the performance of each base classifier for a new instance. We discuss two variations of the application phase: dynamic selection (DS) and dynamic voting (DV). In DS, a classifier with the

best predicted performance is selected to produce the final classification. In DV, a weight is given to each base classifier and each classifier gives its vote. A classification with the highest vote is then selected.

Koppel & Engelson [16] have proposed building a referee predictor for each base classifier. These predictors are able to predict whether the corresponding classifiers can be trusted for each new instance. In their approach the final classification is the one produced by the base classifier which can be trusted most according to the predictions of referees. In the referee induction they used the decision tree induction algorithm C4.5. To train each referee for each base classifier the training set was divided into two parts: correctly and incorrectly classified instances. We apply the same basic idea that each classifier is most reliable inside a particular sub area of the instance space. The main difference between these two approaches is the combining algorithm. Instead of the C4.5 decision tree induction algorithm [16], we use the weighted nearest neighbor classification (WNN) [1]. WNN simplifies the learning phase because there is no need to learn the m referees, only the performance matrix for the base classifiers is needed. In our technique during the application phase the nearest neighbors of a new instance are sought out and the performance prediction for each base classifier is calculated using the performance matrix. In this calculation the corresponding performance values for each classifier are summed up using weights that depend on the distances between the new instance and its nearest neighbors.

The use of WNN as a meta-level classifier conforms to the assumption that each base classifier is most reliable inside certain sub areas of the instance space. This assumption is also supported by the experiences, that classifiers usually work well not only in certain points of the space, but also in sub areas around the points. Thus if a classifier does not work well for instances near a new instance, then it is quite probable that it will not work well for the new instance. Of course, the training set needs to be a good representative of the whole space to be able to make this assumption highly reliable.

Our approach can be considered as a particular case of the stacked generalization architecture Figure 1. Instead of the base classifier predictions, we use information about the performance of the base classifiers for each training instance. For each training instance we calculate a vector of the classifier's errors containing m values. These values can be binary (i.e. a classifier gives correct or incorrect result) or the values can represent corresponding misclassification costs. This information about the performance of base classifiers is then stacked (as in stacked generalization) and is further used together with the initial training set as meta-level knowledge for estimating errors in new instances. Cross-validation [26, 15] is used to calculate the base classifiers' errors. Using cross-validation, the vector of classification errors (or misclassification costs) is formed for each of the n training instances. Those n vectors (i.e. the matrix of the classifier errors $\mathbf{P}_{n\times m}$) together with the set of the attributes of the training instances $\mathbf{T}_{n\times l}$ (l is the number of attributes) are used as a training set for the meta-level classifier $\mathbf{T}^*_{n\times(l+m)}$.

It is necessary to note that cross-validation usually gives pessimistically biased estimates because the classifiers are trained only on a part of the training set, and classifiers trained on the whole training set are used in real problems. Although we obtain biased estimations, these estimations are similarly biased for all the base classifiers and have a small variance. That is why cross-validation is used to generate the meta-level error information of the base classifiers.

Thus, the algorithm for dynamic classifier integration includes the learning phase and two variations of the application phase (dynamic selection, DS, and dynamic voting, DV). In the learning phase first, the training set $\mathbf{T}$ is partitioned into folds. Then the cross-validation technique is used to estimate the errors of the base classifiers E_j on the training set and to form the meta-level training set $\mathbf{T}^*$ that contains features of the training instances $\mathbf{x}_i$, and the estimations of the errors of the base classifiers on those instances E_j. The last part of the learning phase includes training of the base classifiers C_j on the whole training set.

In the DS version of the application phase the classification error E_j^* is predicted for each base classifier C_j using the WNN procedure and a classifier with the smallest error (with the smallest global error in the case of ties) is selected to make the final classification. In the DV version of the application phase each base classifier C_j has a weight W_j, and the final classification is derived by weighted voting.

4. Local Feature Selection

Data, analyzed and processed in knowledge discovery and data mining is usually multidimensional presented by a number of features (attributes). Commonly there are present also many irrelevant features. When several such features exist, many of the data mining techniques are confused that may result in severe degradation of accuracy. In this section the feature selection problem is discussed based on local considerations of the relevance of each feature and this forms background for the experimental work that is considered in next sections.

The problem caused by irrelevant features is usually solved by identifying and discarding them retaining all information needed for further processing [17]. The task to identify irrelevant features has mostly been tried to solve globally by covering the whole domain space by a single subset of features. However, the feature space is often heterogeneous, where the features that are important are different in different sub areas of the instance space [3].

This may result in two kinds of problematic situations. First, when some feature is relevant only inside a small sub area, it may globally be evaluated as irrelevant or almost irrelevant and discarded from the set of relevant features even if it is highly relevant inside the sub area. Second, when some feature is almost globally relevant, it can still be totally irrelevant inside some small sub area and may confuse the classifiers totally inside it [11]. Both of these situations are hard to solve using a single set of relevant features.

There are two main approaches to solve the above problem [4]. First, a more complex global model can be generated to include the necessary information, for example by adding extra artificial features. Second, the problem can be divided into subproblems and the solution of the whole classification task can be guided by the heterogeneity of the instance space. This decomposition approach is potentially highly beneficial, because most widely used techniques rely on measures that are computed over all the features and all the examples at hand, and are inevitably diffused by an averaging effect over the entire problem [4]. We apply this second approach in this paper trying to fit a simple model to local regions instead of trying to build a single global model for the whole instance space.

For example, Atkeson *et al.* [4] showed that global learning methods can often be improved by localizing them using locally weighted training criteria. Domingos [11] introduced a new feature-selection algorithm that uses a clustering-like approach to select sets of locally relevant features for nearest-neighbor classifiers. Here we consider a local feature selection technique that can be used with different types of classifiers. Previously we considered an application of preliminary version of this technique to heterogeneous medical databases [29].

Local feature selection methods include methods, defining feature subsets that vary with *class*, *feature value*, and *instance location* in the instance space [4]. In the first two approaches relevant features are considered to be identical for all instances of the same class or for all instances with the same feature values. For example, Howe & Cardie [14] presented a technique called class distribution weighting (CDW), where they allow feature weights to vary at the class level. However, those approaches are

also restricted to only a part of all feature selection problems. We consider a more general and flexible approach when relevant features can be different for sub areas of the instance space.

Differentiation between feature subsets can be based on a heuristic measure, as the quality of the class separation, or on the actual accuracy obtained by applying the classifier using those features [17]. A feature selection algorithm applying a heuristic measure acts as a *filter* extracting features from a feature set. A feature selection algorithm applying the actual accuracy acts as a *wrapper* around the main algorithm. We consider a dynamic feature selection technique that uses the wrapper approach [17] because it has been found to yield the best results. The evaluation function in our case is the local classification accuracy obtained by applying the classification algorithm with different feature subsets. We apply a technique for dynamic integration of classifiers [22]. It allows us to determine what classifier and with what feature subset should be applied for each new instance. This technique can also be applied in the case of implicit heterogeneity when the regions of heterogeneity cannot be defined by a simple dependency. In our method the classifiers included in the ensemble (the base classifiers) are built based on various subsets of the original feature set, as discussed in Skrypnyk *et al.* [29].

In this paper we consider an advanced version of the method presented in Skrypnyk *et al.* [29]. We previously analyzed all possible combinations of features to build the ensemble of base classifiers. However, this can be computationally very expensive and leads to overfitting. The big number of feature subsets and, consequently, integrated classifiers dramatically increases the number of degrees of freedom in the training process, leading to increased variance of predictions and an increased risk of overfitting the data. To reduce the risk, we propose to limit the number of feature subsets in our local wrapper technique. The base classifiers in the ensemble can be built using combinations of only potentially locally relevant features, discarding features that are definitely irrelevant at that sub area.

Some recursive partitioning techniques or some heuristic measures can be used to discard features that are locally irrelevant with a high probability. Thus, we propose to combine our wrapper-based method with a filter approach, using it in advance for restricting the possible feature combinations. In Cardie & Howe [7] a decision tree is proposed for local feature selection, where only those features are considered to be locally relevant for a test case, which lie on the path taken by the test case in the decision tree. We propose to combine this decision-tree filter feature selection [7] with our method.

$\mathbf{T}_i$ *i*-th fold of the training set $\mathbf{T}$
$\mathbf{T}^*$ meta-level training set for the combining algorithm
$c(\mathbf{x})$ class of the instance x
$\mathbf{F}$ set of feature subsets
$\mathbf{C}$ set of base classifiers C_j built on feature subsets $\mathbf{F}_j$
m,m' number of base classifiers before & after feature filtering
$C_j(\mathbf{x})$ class predicted by C_j on instance $\mathbf{x}$
$E_j(\mathbf{x})$ estimation of error of C_j on instance $\mathbf{x}$
$E_j^*(\mathbf{x})$ prediction of error of C_j on instance $\mathbf{x}$
$\mathbf{W}$ vector of weights for the base classifiers $\mathbf{C}$
nn number of near-by instances for local error prediction
$W_{NN\,i}$ weight of *i*-th near-by instance
GTree guiding tree for local feature filtering

procedure *learning_phase*(**T,C**)
begin
  `generate` **F**
  `partition` **T** `into` *v* `folds`
  loop for $\mathbf{T}_i \subset \mathbf{T}$, $i = 1, ..., v$
    loop for *j* **from** *1* **to** *m* C_j= `train`($\mathbf{F}_j$,**T**-$\mathbf{T}_i$)
    loop for $\mathbf{x} \in \mathbf{T}_i$
      loop for *j* **from** *1* **to** *m*
        `compare` C_j(**x**) `with` *c*(**x**) `and derive` E_j(**x**)
    `collect` (**x**,E_1(**x**),...,E_m(**x**)) `into` $\mathbf{T}^*$
  loop for *j* **from** *1* **to** *m* C_j= `train`($\mathbf{F}_j$,**T**)
  GTree= `C4.5_train`(**T**)
end

function *DS_application_phase*($\mathbf{T}^*$,**C,x**) **returns** `class of` **x**
begin
  `find path of` **x** `in` *GTree*
  `discard classifiers with features not in the path`
  loop for *j* **from** *1* **to** *m'* $E_j^* \leftarrow \frac{1}{nn}\sum_{i=1}^{nn} W_{NN_i} \cdot E_j(\mathbf{x}_{NN_i})$
  $l \leftarrow \arg\min_j E_j^*$ {`number of classifier with min.` E_j^* }
  {`with the least global error in the case of ties`}
  return C_j(**x**)
end

function *DV_application_phase*($\mathbf{T}^*$,**C,x**) **returns** `class of` **x**
begin
  `find path of` **x** `in` *GTree*
  `discard classifiers with features not in the path`
  loop for *j* **from** *1* **to** *m'* $W_j \leftarrow 1 - \frac{1}{nn}\sum_{i=1}^{nn} W_{NN_i} \cdot E_j(\mathbf{x}_{NN_i})$
  return *Weighted_Voting*(**W,**C_1(**x**),...,C_m(**x**))
end

Algorithm 1.The algorithm for local feature selection with dynamic integration of classifiers

Thus, the modified version of the dynamic classifier integration algorithm for local feature selection also includes the learning phase and the two variations of the application phase (Algorithm 1). In the learning phase first, all possible feature subsets to be analyzed are generated. Some heuristic procedure can be used at this step to restrict the number of feature subsets to be analyzed. In our experiments, we simply generate all possible feature combinations including different amounts of features. After this, the training set **T** is partitioned into folds. Then the cross-validation technique is used to estimate the errors E_j of the base classifiers C_j built on the previously generated feature subsets $\mathbf{F}_j$. The meta-level training

set $\mathbf{T}^*$ is formed that contains features of the training instances $\mathbf{x}_i$and the estimations of the errors of the base classifiers on those instances E_j. The last part of the learning phase includes training of the base classifiers C_j on the whole training set and finishes with training a guiding C4.5 decision tree *GTree* that will be used for preliminary local feature filtering.

Both DS and DV versions of the application phase begin with finding a path for the current instance in the guiding decision tree *GTree*. For future consideration only those base classifiers are taken into account, which are built with features lying on this path. Thus we considerably restrict the number of the classifiers used. After this, the integration of selected classifiers is done as in usual dynamic integration. In DS, the classification error E_j^* is predicted for each base classifier C_j using the WNN procedure and a classifier with the smallest error (with the smallest number in the case of ties) is selected to make the final classification. In the DV version, each base classifier C_j has a weight W_j and the final classification is derived using weighted voting.

Both the dynamic classifier integration technique and its modification for local feature selection require a measure of distance to find nearest neighbors. The success of the above techniques depends critically on the distance function [23]. In our previous experiments we have found the Heterogeneous Euclidean-Overlap Metric (HEOM) as a proper choice and it will also be used in these experiments.

5. Experimental Setting

In this section, we present our experimental setting that is used in the experiments described in the following sections 6-8. We conduct the experiments on 13 data sets taken from the UCI machine learning repository [6], the characteristics of which are presented in Table 1. Here we use a similar experimental environment, as the one with which we earlier experimented with the dynamic classifier integration in Puuronen *et al.* [22] and with the unguided local feature selection in Skrypnyk *et al.* [29]. The exception is that this time we use the algorithm to build C4.5 decision trees with and without pruning (Quinlan, 1993) at the end of the learning phase. These are used for local feature filtering in the beginning of the application phase. In this filtering approach, for each test instance only those feature combinations that include features present in the path taken by the test instance in the decision tree are considered.

For each data set 30 test runs are made. First in each run 30 percent of the instances of the data set are picked up to the test set by random sampling. The rest 70 percent of the instances form the training set which is then passed to the learning algorithm. Second, this training set is divided into 10 folds using stratified random sampling because we apply 10-fold cross-validation to build the cross-validation history for the dynamic integration of the base classifiers [22]. The base classifiers themselves are learned using the C4.5 decision tree algorithm with pruning [24]. Two different ensembles of base classifiers are formed: 1) an ensemble including base classifiers based on exactly one feature only and 2) an ensemble including base classifiers based on the subsets of exactly two features. Figure 2 presents in graphical form the averages of accuracies of all the base classifiers with one feature as they are presented in the table of Appendix 1.

The figure shows for each data set the averages of minimum, maximum, and average accuracies of all the base classifiers over all the 30 runs. Corresponding to each data set, the lowest end of the vertical bar is the minimum value, the highest end of the vertical bar is the maximum value and the dot is placed in the bar at the average value. As can be seen in Figure 3, the averages for data sets Glass, LED, LED17, MONK-1, Soybean, and Zoo are smaller than 0.5.

Table 1. Data sets used in the experiments

Data set	Instances	Classes	Features	
			Categorial	Continuous
Breast	286	2	9	0
DNA promoter	106	2	57	0
Glass	214	6	0	9
Heart	270	2	8	5
Iris	150	3	0	4
LED	300	10	7	0
LED17	300	10	24	0
Liver	345	2	0	6
Lymphography	148	4	15	3
MONK-1,3	432	2	6	0
Soybean	47	4	0	35
Zoo	101	7	16	0

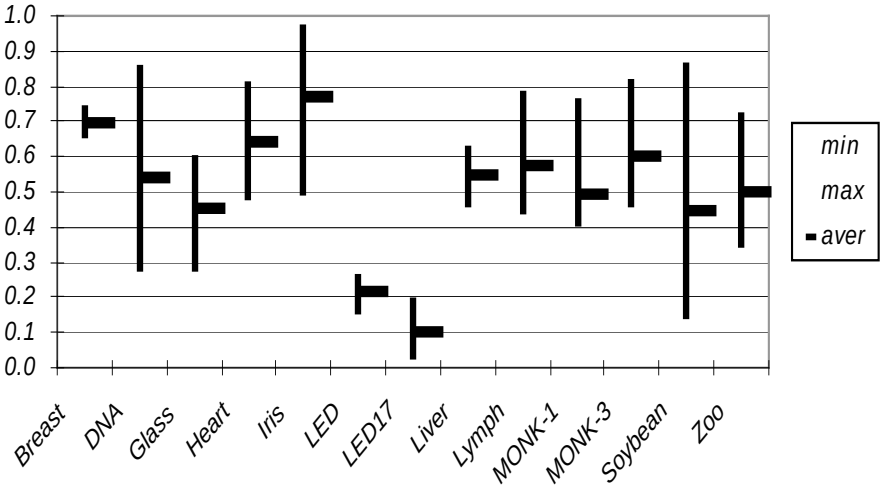

Figure 2. Averages of minimum, average, and maximum values of the accuracies of all the one-feature base classifiers.

The corresponding results of all the base classifiers using two features are shown in Figure 3 and included in the table of Appendix 2. As can be seen in Figure 3 the averages for data sets LED, LED17, and Soybean are smaller than 0.5. The averages of the average accuracies is about 0.06 higher on average over all the data sets but the differences vary a lot as can be seen in Figure 4.

The above figures describe the unguided situation when all the one or two-feature base classifiers are taken into account with every instance in the test set. In the guided versions only those base classifiers that are selected by the guiding decision tree are actually taken into account. This changes the situation

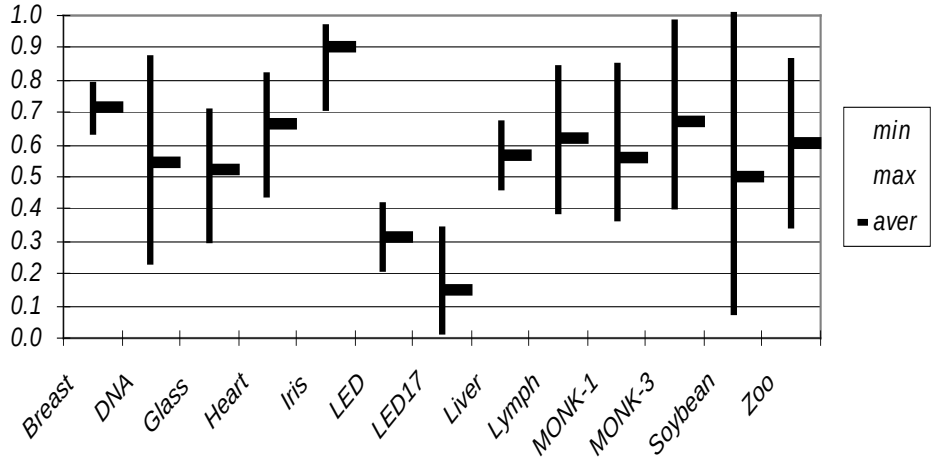

Figure 3. Averages of minimum, average, and maximum values of the accuracies of all the two-feature base classifiers.

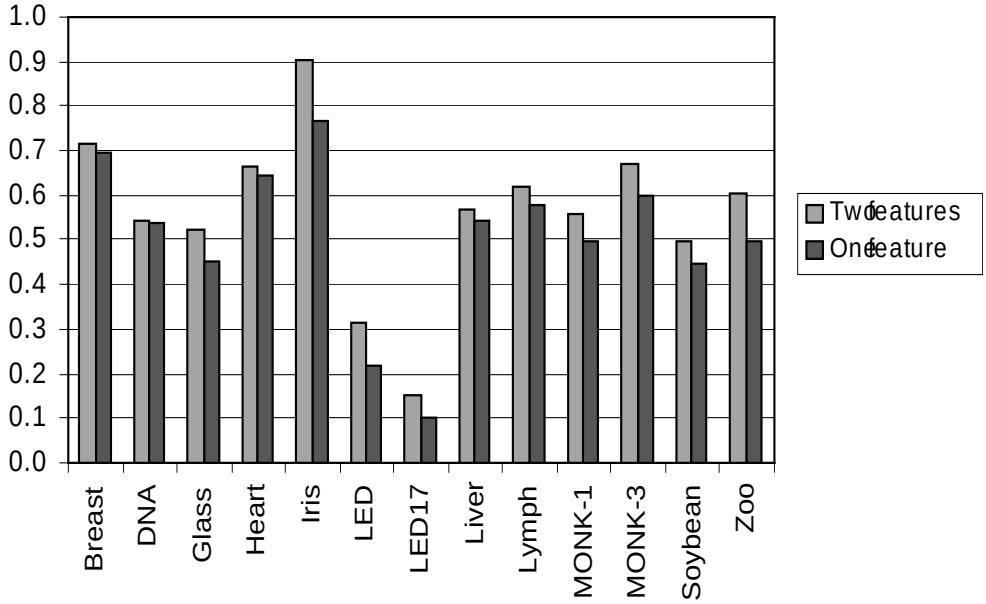

Figure 4. Averages of the average accuracies of one-feature and two-feature base classifiers.

depending on the data set. Figure 5 presents the corresponding accuracy values of selected classifiers for one-feature base classifiers when a C4.5 decision tree without pruning is used to guide the use of the base classifiers and in Figure 6 corresponding results with pruning are shown.

When Figures 2, 5, and 6 are compared as in Figure 7 we can see that with most data sets the guidance raises the average accuracy of the base classifiers. The biggest counterexample is the data set Zoo.

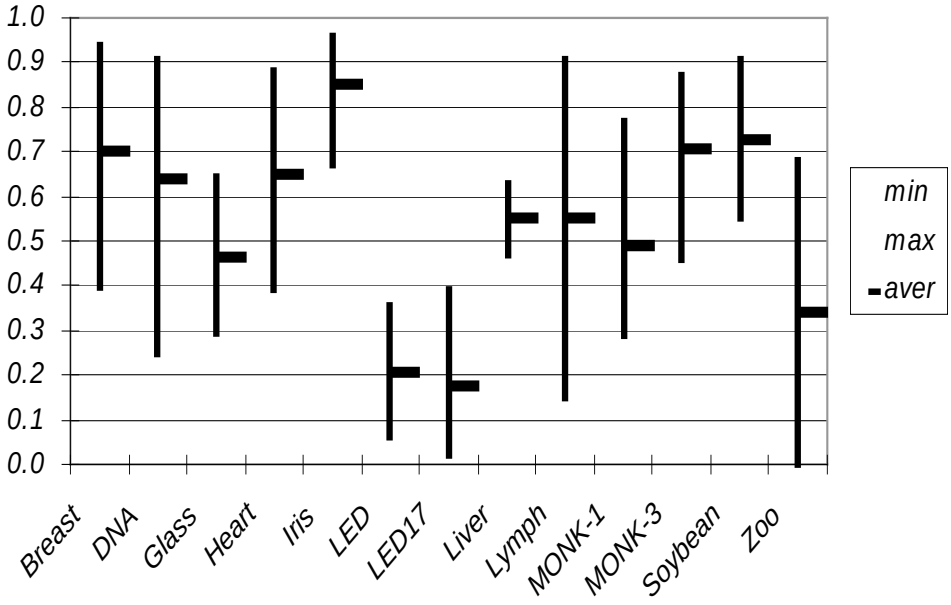

Figure 5. Averages of minimum, average, and maximum values of the accuracies of the one-feature base classifiers with C4.5 guidance without pruning.

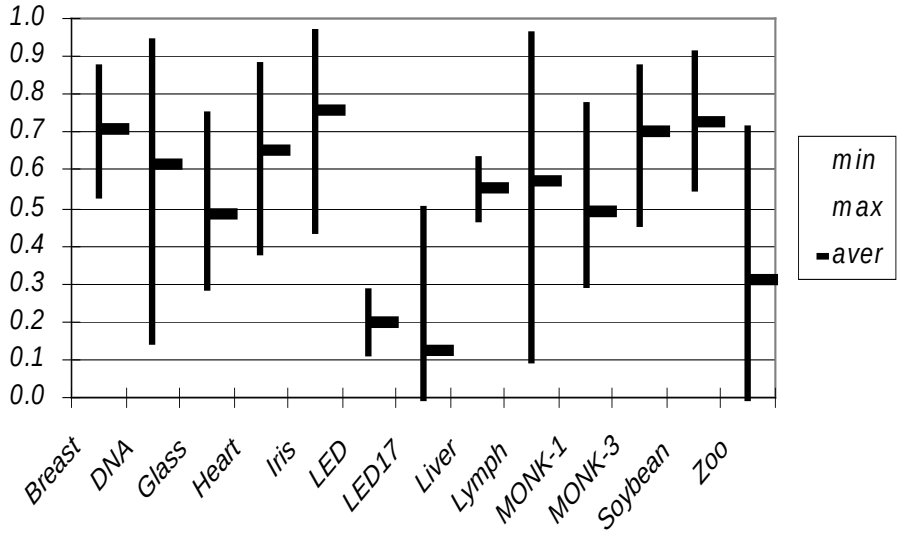

Figure 6. Averages of minimum, average, and maximum values of the accuracies of the one- feature base classifiers with C4.5 guidance with pruning.

The above figures describe the situation with one-feature base classifiers. The corresponding situation with respect to the two-feature base classifiers is described in Figures 8, 9, and 10. Figure 8 presents the accuracy values of selected classifiers for two-feature base classifiers when C4.5 decision tree without pruning is used to guide the use of the base classifiers and in Figure 8 corresponding results with pruning are shown.

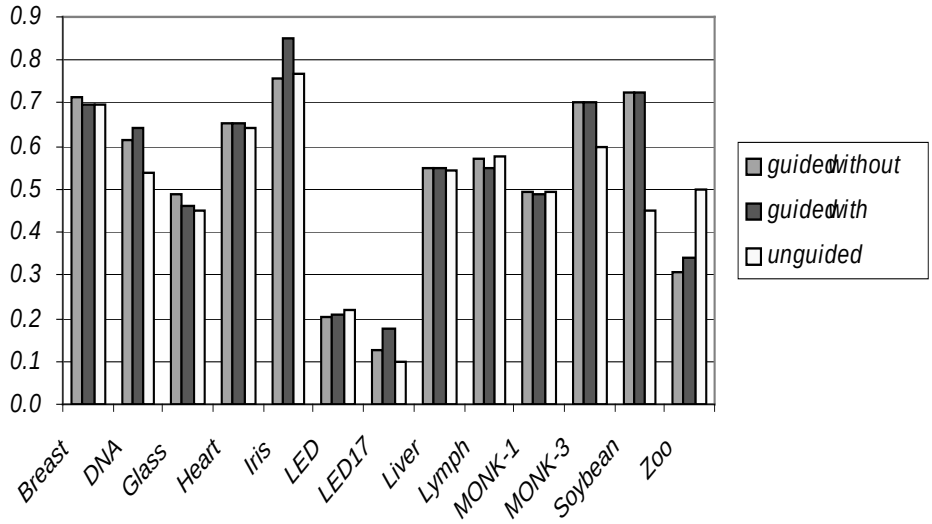

Figure 7. Averages of average of the one-feature base classifiers with C4.5 guidance with and without pruning, and without guidance.

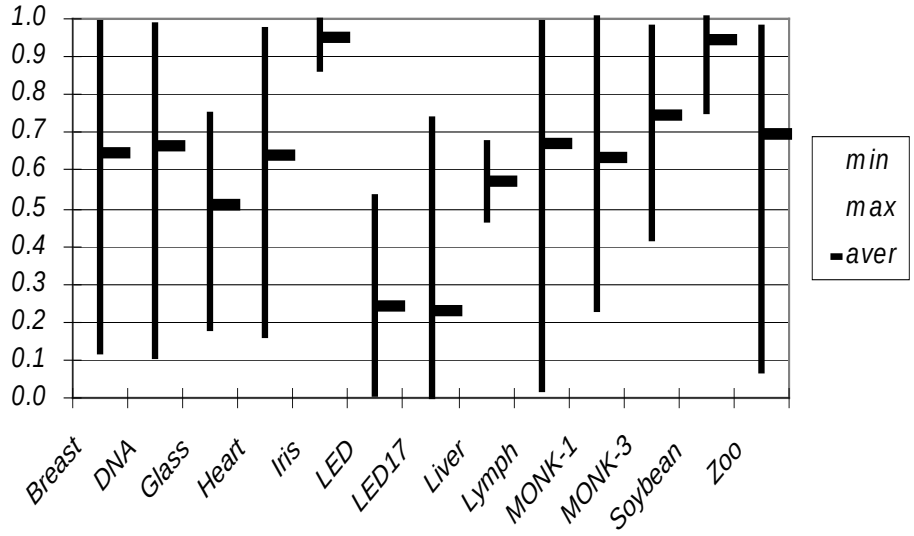

Figure 8. Averages of minimum, average, and maximum values of the accuracies of the two-feature base classifiers with C4.5 guidance without pruning.

When Figures 3, 8, and 9 are compared as in Figure 10 we can see that with most data sets the guidance raises the average accuracy of the base classifiers.

As a conclusion about the experimental setting we are able to notice that LED and LED17 data sets remain in both one- and two-feature situation far from being even near the average accuracy of 0.5 of the base classifiers. Thus we can expect that they are not the data sets with which the accuracy of an ensemble is able to raise to a reasonable level [35].

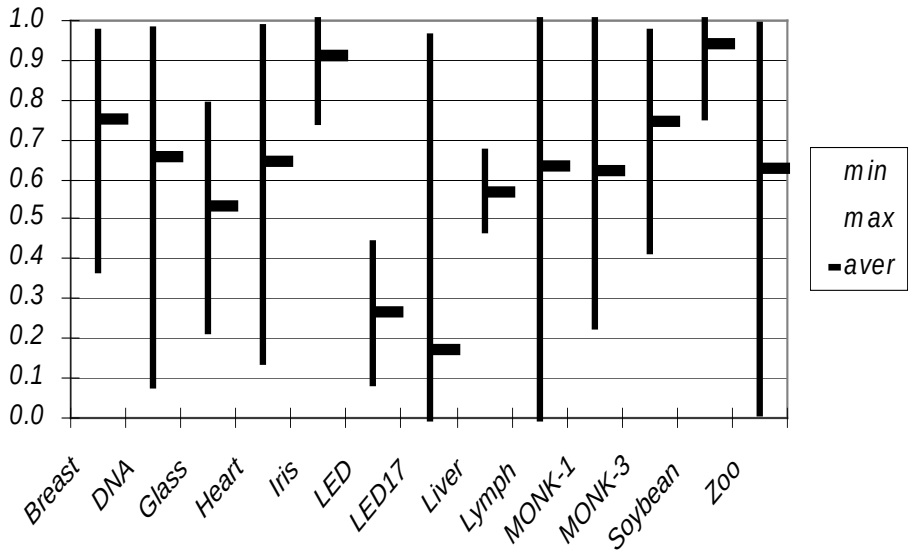

Figure 9. Averages of minimum, average, and maximum values of the accuracies of the two-feature base classifiers with C4.5 guidance with pruning.

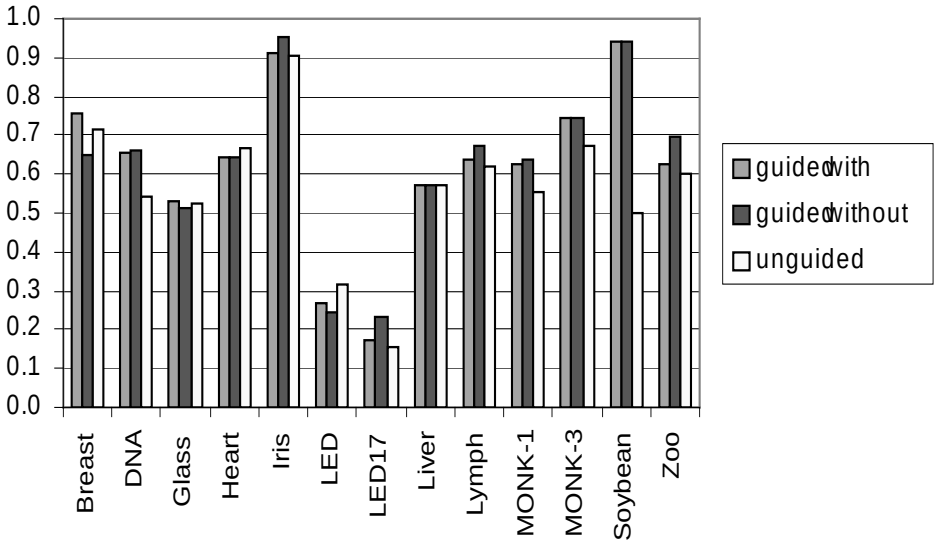

Figure 10. Averages of average of the two-feature base classifiers with C4.5 guidance with pruning, without pruning and unguided situation.

6. Static and Dynamic Approaches

In this section, we present our experiments with the experimental setting described in section 5 with the static approaches CVM (Cross-Validated Majority [26] and WV (Weighted Voting [1] and our dynamic approaches DS (Dynamic Selection) and DV (Dynamic Voting).

Figure 11 presents the accuracies of the approaches achieved with different data sets without guidance with one-feature base classifiers. As can be seen with the data sets DNA, LED17, Lymphography, MONK-1, MONK-3, Soybean, and Zoo the static selection approach results in higher accuracies than the static weighted voting technique. For all these data sets the average accuracy of the base classifiers is smaller than the middle value of accuracy interval ((max+min)/2). For the rest of the data sets average accuracy is at least as big as the middle value. Thus static selection seems to work better than voting on average when there are a few good classifiers in the ensemble. The relation is similar also with dynamic approaches except with the Soybean data set. CVM resulted in a higher average accuracy with seven data sets than DS. WV resulted in a higher average accuracy with eight data sets than DV.

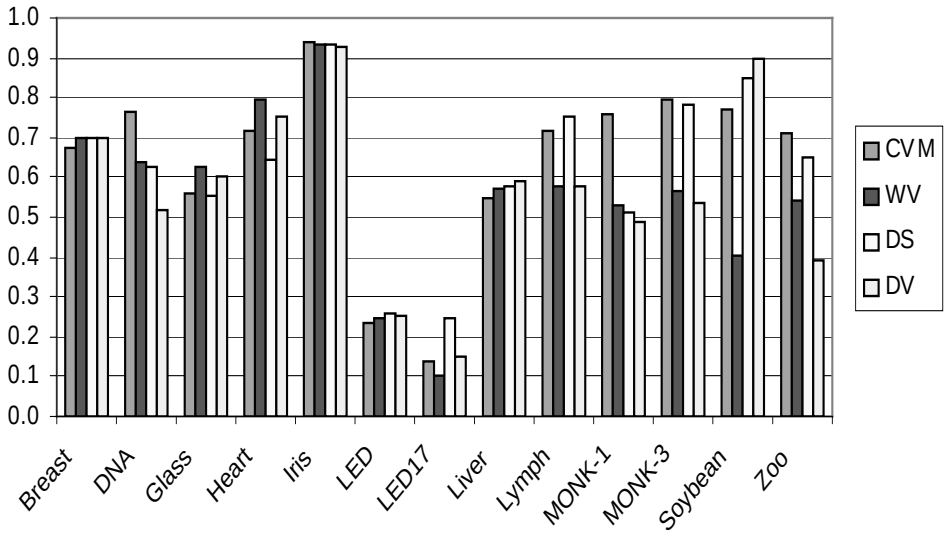

Figure 11. Accuracies of different integrating approaches with different data sets in the unguided situation with one-feature base classifiers.

Figure 12 presents the corresponding accuracies of the integrating approaches in the situation when integration of the one-feature base classifiers is guided by C4.5 decision trees without pruning. As can be seen then the differences in accuracies between different approaches are clearly smaller. Only the data sets MONK-3 and Soybean seem to work clearly better with static approaches than with the dynamic ones. This relationship between the average accuracy and the middle value of accuracies mentioned above was not found in the guided situation.

Figure 13 presents the corresponding accuracies of the integrating approaches in the situation when integration of the one-feature base classifiers is guided by C4.5 decision trees with pruning. As can be seen the differences of accuracies between different approaches are almost the same as without pruning and the data sets MONK-3 and Soybean seem to work clearly better with static approaches than with the dynamic ones as in Figure 12.

Above we analyzed the cases with one-feature base classifiers. Now we look more carefully at the case with two-feature base classifiers. Figure 14 presents the accuracies of the approaches achieved with different data sets in the unguided situation with two-feature base classifiers. As can be seen with the data sets DNA, MONK-1, MONK-3, Soybean, and Zoo static selection results in higher accuracies

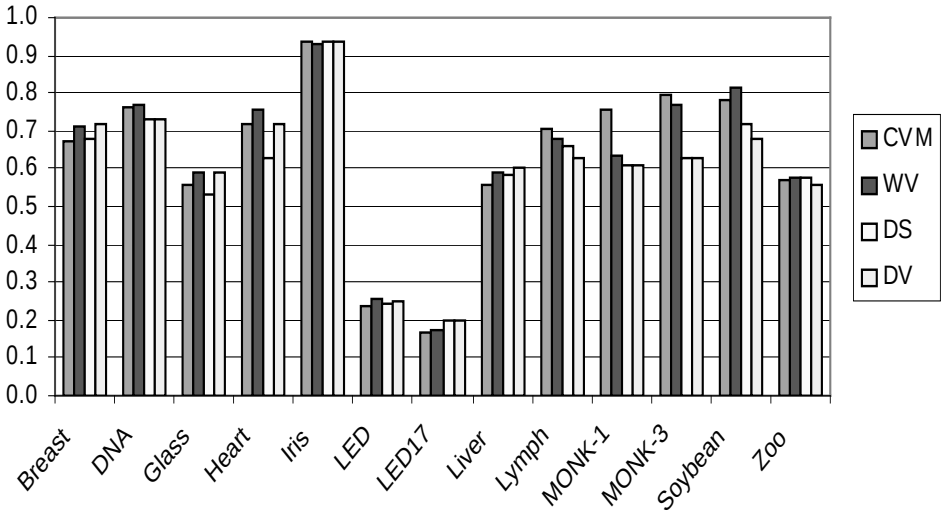

Figure 12. Accuracies of different integrating approaches with different data sets in the guided situation with one-feature base classifiers and without pruning of the decision tree which is guiding the classification process.

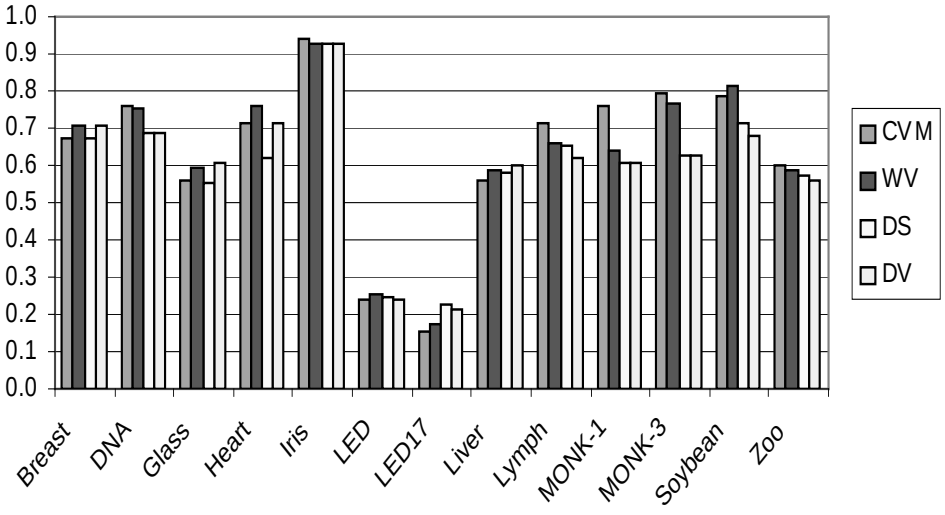

Figure 13. Accuracies of different integrating approaches with different data sets in the guided situation with one-feature base classifiers and with pruning the decision tree which is guiding the classification process.

than the static voting technique. Of these data sets all but Zoo have average accuracy of base classifiers smaller than the middle accuracy of the base classifiers accuracy interval. The relationship between dynamic selection and dynamic voting followed the similar behaviour except that that dynamic selection had clearly higher accuracy than dynamic voting with LED and LED17 data sets. DS was better than CVM with seven data sets when WV was better than DV with eight data sets, but the differences were big only with LED17 and MONK-1 data sets.

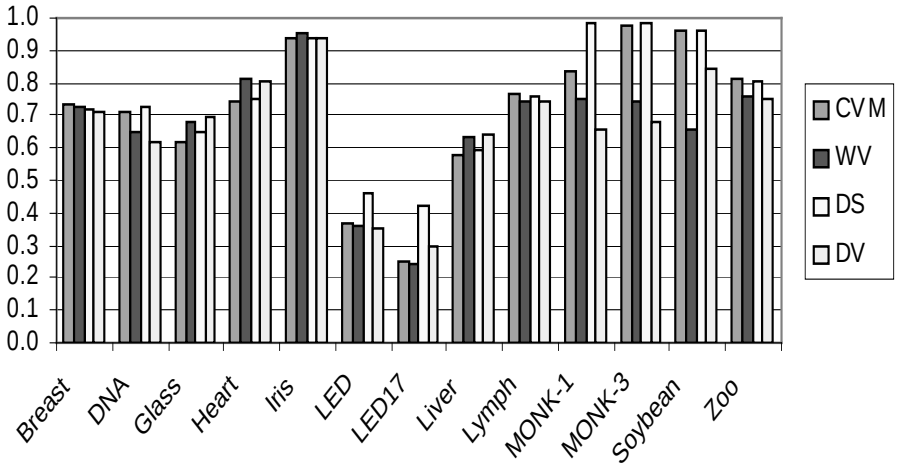

Figure 14. Accuracies of different integrating approaches with different data sets in the unguided situation with two-feature base classifiers.

Figure 15 presents the corresponding accuracies of the integrating approaches in the situation when integration of the two-feature base classifiers is guided by C4.5 decision trees without pruning. As can be seen then the differences of accuracies between different approaches are clearly smaller. Only the data set MONK-1 seems to work clearly better with static approaches than with the dynamic ones.

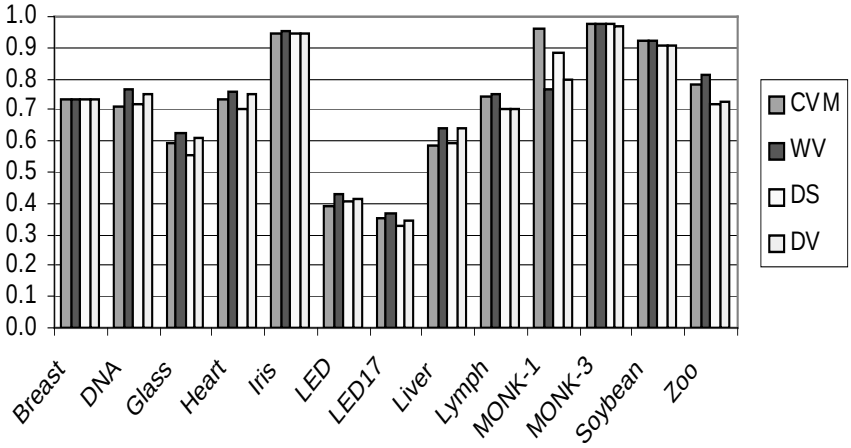

Figure 15. Accuracies of different integrating approaches with different data sets in the guided situation with two-feature base classifiers and without pruning the decision tree which is guiding the classification process.

Figure 16 presents the corresponding accuracies of the integrating approaches in the situation when integration of the two-feature base classifiers is guided by C4.5 decision trees with pruning. As can be seen the differences of accuracies between different approaches are almost the same as without pruning and the data set MONK-1 seems to work clearly better with static approaches than with the dynamic ones as in Figure 15.

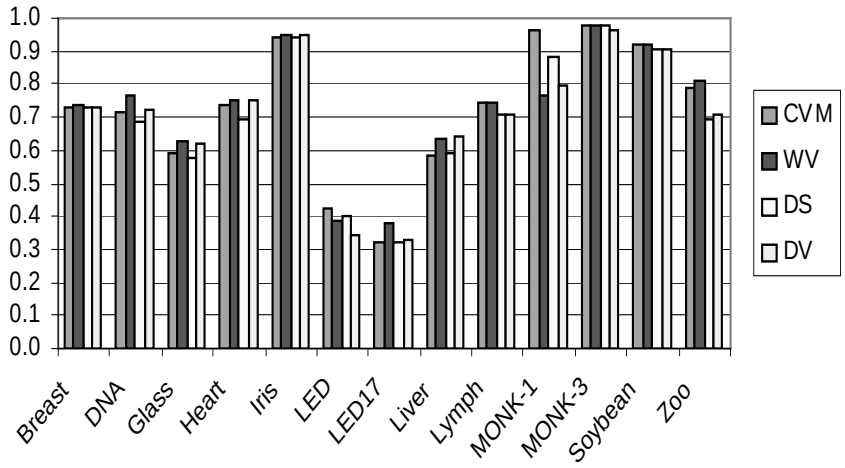

Figure 16. Accuracies of different integrating approaches with different data sets in the guided situation with two-feature base classifiers and with pruning the decision tree which is guiding the classification process.

7. The Benefit of Guidance

In this section, we consider the effect of guidance in two-feature base classifier situations with different types of data sets. The data sets are divided into those which include only categorial features: Breast, DNA, LED, LED17, MONK-1, MONK-3, and Zoo; those which include only continuous features: Glass, Iris, Liver, and Soybean, and those which include both continuous and categorial features: Heart and Lymphography. For each group the accuracies are received using different approaches: the static approaches CVM (Cross-Validated Majority [26] and WV (Weighted Voting [1] and our dynamic approaches DS (Dynamic Selection) and DV (Dynamic Voting).

Figure 17 presents the average accuracies of the approaches achieved over the categorial data sets. It seems that the voting type approaches are able to take more benefit of the guidance than the selection type approaches with data sets including only categorial features.

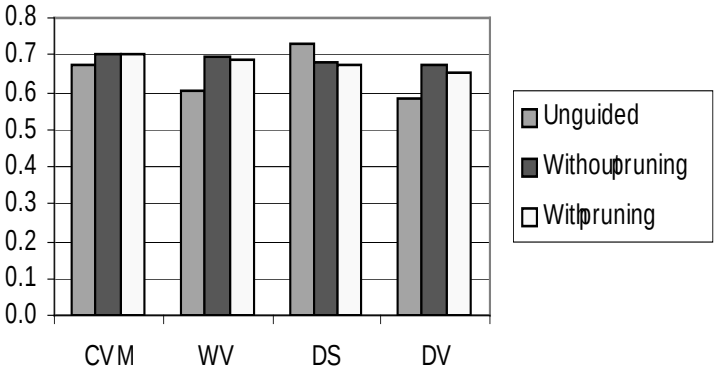

Figure 17. Average accuracies achieved with different integrating approaches with data sets including only categorial features with two-feature base classifiers.

Figure 18 presents the corresponding average accuracies of the integrating approaches in the situation when integration of the two-feature base classifiers with data sets including only continuous features. It seems that the selection-type approaches do not work well with guidance. Only static weighed voting seems to take real benefit of guidance with these data sets including continuous features only.

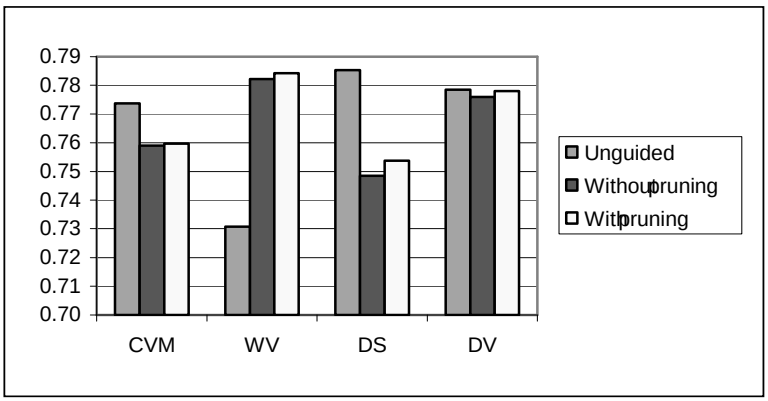

Figure 18. Average accuracies achieved with different integrating approaches with data sets including only continuous features with two-feature base classifiers.

Figure 19 presents the corresponding average accuracies of the integrating approaches in the situation when integration of the two-feature base classifiers with the data sets including both continuous and categorial features. It seems that in this case all the approaches work worse with the guidance than without it.

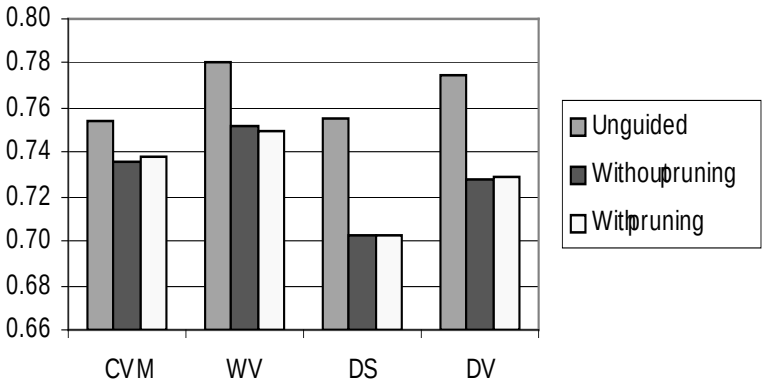

Figure 19. Average accuracies achieved with different integrating approaches with data sets including both continuous and categorial features with two-feature base classifiers.

8. The Number of Features Used to Learn the Base Classifiers

In this section, we consider the amount of features taken into account by different approaches in different situations. We present our experiments with the experimental setting described in section 5.

Figure 20 presents the number of features used during the learning phase when the base classifiers are based on only one feature. It can be seen how dramatically the guidance cut the number of features taken into account.

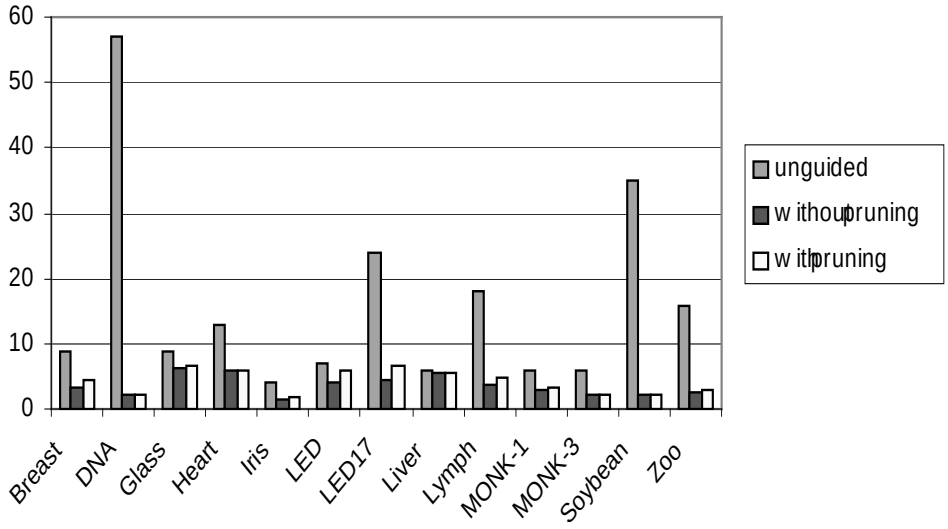

Figure 20. The number of features used during the learning phase in unguided and guided approaches with one-feature base classifiers.

The number of features taken into account over the all data sets with guidance is about 40% of the total number of features. Still the average accuracy over all the data sets and integration methods is a little bit more than 2% higher. Of the different integration methods weighted voting results in more than 8% higher accuracy with this smaller number of features than with all the features; correspondingly dynamic voting gives 5-6% and CVM about 1.5% higher accuracies. The dynamic selection is the only integration method which results in a smaller accuracy (more than 6%) with this smaller feature set than with the whole feature set.

Let's look more closely at data sets having different kinds of features. For those data sets including only *categorial* features, the amount of features taken into account was on average 31% with pruning and 38% without pruning. If the multiple feature data sets DNA and LED17 are not taken into account, the corresponding percentages are 40% and 48% . The average accuracies of the different integration methods compared to the situation with all features were: CVM almost 5% higher, WV about 14% higher, DS about 7% smaller, and DV 13-16% higher over the data sets. If DNA and LED17 are not taken into account, the percentages were: CVM about 3-4% higher, WV about 10-11% higher, DS about 6.5% smaller, and DV 13-15% higher.

For those data sets including only *continuous* features taken into account was on average 53% with pruning and 55% without pruning. The average accuracies with different integration methods were correspondingly: CVM about 2% smaller, WV about 7% smaller, DS more than 4% smaller, and DV almost the same as in the situation with all features.

For the two data sets including both categorial and continuous features, the amount of features taken into account was on average 32% with pruning and 36% without pruning. The average accuracies with

different integration methods were: CVM over 2% smaller, WV about 4% smaller, DS about 7% smaller, and DV about 6% smaller.

We can conclude that the guiding seems to work quite well with data sets including only categorial features, when with less than half of the features it is possible to reach even 10% higher accuracy with the voting methods. When a data set includes also continuous features, then even only 30-50% of the features is enough to reach about 5% smaller accuracy than using all the features.

9. Conclusion

In this paper we described a technique that searches for a division of the feature space identifying the best subsets of features for each instance. The technique is based on the local wrapper approach, and uses the method for dynamic integration of classifiers to determine which classifier and which subset is applied for each new instance. At the application phase, in order to restrict the number of feature combinations being analyzed, we used the C4.5 decision tree built on the whole feature set as a feature filter. For each test instance we considered only those feature combinations that included features present in the path taken by the test instance in the decision tree. Our technique can be applied in the case of implicit heterogeneity when the regions of heterogeneity cannot be easily defined by a simple dependency.

We conducted experiments on data sets of the UCI machine learning repository using ensembles of simple base classifiers each generated on either one or two features. The results achieved are promising and show that the local feature selection in comparison with selecting only one feature set for the whole space can be advantageous in some cases.

We can conclude that the use of decision trees for local feature filtering seems to work quite well with data sets including only categorial features, when with less than half of the features it is possible to reach even 10% higher accuracy than with all the features. When a data set includes also continuous features, then even only 30-50% of the features is enough to reach about 5% smaller accuracy than using all the features.

Further experiments can be conducted to make deeper analysis of applying recursive partitioning and the dynamic integration of classifiers for local feature selection (and particularly to strictly define the dependency between the parameters of local feature selection, characteristics of a domain, and the data mining accuracy). Decision trees built on the whole instance set were used in our experiments for the purpose of preliminary local feature selection. Use of other feature filters can be tested in future experiments. Another potentially interesting topic for further research is the analysis of feature subsets without the restriction on their size. Also it would be interesting to consider an application of this technique to some hard real-world problem.

References

[1] Aha, D. W., Kibler, D., Albert, M. K.: Instance-based learning algorithms, *Machine Learning*, **6**, 1991, 37–66.

[2] Aivazyan, S. A.: *Applied Statistics: Classification and Dimension Reduction*, Finance and Statistics, Moscow, 1989.

[3] Apte, C., Hong, S. J., Hosking, J. R. M., Lepre, J., Pednault, E. P. D., Rosen, B. K.: Decomposition of heterogeneous classification problems, *Advances in Intelligent Data Analysis, Proc. IDA-97* (X. Hiu, P. Cohen, M. Berthold, Eds.), Springer-Verlag, 1997, 17–28.

[4] Atkeson, C. G., Moore, A. W., Schaal, S.: Locally weighted learning, *Artificial Intelligence Review*, **11**(1–5), 1997, 11–73.

[5] Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning*, **36**, 1999, 105–139.

[6] Blake, C. L., Keogh, E., Merz, C. J.: *UCI Repository of Machine Learning Databases*, [http:// www.ics.uci.edu/ ∼mlearn/ MLRepository.html], Department of Information and Computer Science, University of California, Irvine, 1999.

[7] Cardie, C., Howe, N.: Improving minority class prediction using case-specific feature weights, *Proc. 14^{th} International Conference on Machine Learning*, Morgan Kaufmann, 1997, 57–65.

[8] Chan, P., Stolfo, S.: On the accuracy of meta-learning for scalable data mining, *Intelligent Information Systems*, **8**, 1997, 5–28.

[9] Chan, P.: *An extensible meta-learning approach for scalable and accurate inductive learning*, Ph.D. Thesis, Columbia University, 1996.

[10] Dietterich, T. G.: Machine learning research: four current directions, *AI Magazine*, **18**(4), 1997, 97–136.

[11] Domingos, P.: Context-sensitive feature selection for lazy learners, *Journal of Artificial Intelligence Review*, **11**(1–5), 1997, 227–253.

[12] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview, in: *Advances in Knowledge Discovery and Data Mining* (U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Eds.), AAAI/MIT Press, 1997, 1–29.

[13] Heath, D., Kasif, S., Salzberg, S.: Committees of decision trees, in: *Cognitive Technology: In Search of a Humane Interface* (B. Gorayska, J. Mey, Eds.), Elsevier Science, Amsterdam, 1996, 305–317.

[14] Howe, N., Cardie, C.: Examining locally varying weights for nearest neighbor algorithms, *Case-Based Reasoning Research and Development, Proc. 2^{nd} International Conference on Case-Based Reasoning* (D. Leake, E. Plaza, Eds.), LNAI, Springer-Verlag, 1997, 455–466.

[15] Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proc. 14^{th} International Joint Conference on Artificial Intelligence* (C. Mellish, Ed.), Morgan Kaufmann, 1995.

[16] Koppel, M., Engelson, S.: Integrating multiple classifiers by finding their areas of expertise, *Proc. AAAI-96 Workshop On Integrating Multiple Learning Models for Improving and Scaling Machine Learning Algorithms*, 1996, 53–58.

[17] Liu, H., Motoda, H.: *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, 1998.

[18] Merz, C. J.: Dynamical selection of learning algorithms, in: *Learning from Data, Artificial Intelligence and Statistics* (D. Fisher, H.-J. Lenz, Eds.), Springer-Verlag, New York, 1996.

[19] Merz, C. J.: *Classification and regression by combining models*, Ph.D. Thesis, Department of Information and Computer Science, University of California, Irvine, 1998.

[20] Merz, C. J.: Using correspondence analysis to combine classifiers, *Machine Learning*, **36**(1–2), 1999, 33–58.

[21] Oza, N., Tumer, K.: *Dimensionality reduction through classifier ensembles*, Technical Report NASA-ARC-IC-1999-126, Computational Sciences Division, NASA Ames Research Center, 1999.

[22] Puuronen, S., Terziyan, V., Tsymbal, A.: A dynamic integration algorithm for an ensemble of classifiers, *Foundations of Intelligent Systems: Proc. 11^{th} International Symposium ISMIS'99* (Z. W. Ras, A. Skowron, Eds.), LNAI 1609, Springer-Verlag, 1999, 592–600.

[23] Puuronen, S., Tsymbal, A., Terziyan, V.: Distance functions in dynamic integration of data mining techniques, *Data Mining and Knowledge Discovery: Theory, Tools and Technology II, Proc. SPIE* (B. V. Dasarathy, Ed.), Vol. 4057, SPIE, 2000, 22–32.

[24] Quinlan, J. R.: *C4.5 Programs for Machine Learning*, Morgan Kaufmann, San Mateo CA, 1993.

[25] Quinlan, J. R.: Bagging, boosting, and C4.5, *Proc. 13^{th} National Conference on Artificial Intelligence*, Portland OR, 1996, 725–730.

[26] Schaffer, C.: Selecting a classification method by cross-validation, *Machine Learning*, **13**, 1993, 135–143.

[27] Schapire, R. E: A brief introduction to boosting, *Proc. 16^{th} International Joint Conference on Artificial Intelligence*, 1999.

[28] Skalak, D. B.: *Combining nearest neighbor classifiers*, Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst, 1997.

[29] Skrypnyk, I., Terziyan, V., Puuronen, S., Tsymbal, A.: Learning feature selection for medical databases, *Proc. 12^{th} IEEE Symposium on Computer-Based Medical Systems*, Stamford CT, USA, IEEE CS Press, 1999, 53–58.

[30] Terziyan, V., Tsymbal, A., Puuronen, S.: The decision support system for telemedicine based on multiple expertise, *International Journal of Medical Informatics*, **49**(2), 1998, 217–229.

[31] Todorovski, L., Dzeroski, S.: Combining multiple models with meta decision trees, *Principles of Data Mining and Knowledge Discovery, Proc. PKDD 2000* (D. A. Zighed, J. Komorowski, J. Zytkow, Eds.), Lyon, France, LNAI 1910, Springer-Verlag, 2000, 54–64.

[32] Tsymbal, A., Puuronen, S.: Bagging and boosting with dynamic integration of classifiers, *Principles of Data Mining and Knowledge Discovery, Proc. PKDD 2000* (D. A. Zighed, J. Komorowski, J. Zytkow, Eds.), Lyon, France, LNAI 1910, Springer-Verlag, 2000, 116–125.

[33] Tsymbal, A.: Decision committee learning with dynamic integration of classifiers, *Current Issues in Databases and Information Systems, Proc. ADBIS-DASFAA 2000* (J. Štuller, J. Pokorný, B. Thalheim, Y. Masunaga, Eds.), Prague, Czech Republic, LNCS 1884, Springer-Verlag, 2000, 265–278.

[34] Tsymbal, A., Puuronen, S., Terziyan, V.: Advanced dynamic selection of diagnostic methods, *Proc. 11^{th} IEEE Symposium on Computer-Based Medical Systems* Lubbock TX, USA, IEEE CS Press, 1998, 50–54.

[35] Tumer, K., Ghosh, J.: Error correlation and error reduction in ensemble classifiers, *Connection Science*, **8**(3–4), 1996, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches, 385–404.

[36] Webb, G. I.: MultiBoosting: a technique for combining boosting and wagging, *Machine Learning*, **40**(2), 2000, 159–196.

[37] Wolpert, D.: Stacked generalization, *Neural Networks*, **5**(2), 1992, 241–259.

Appendix 1. Accuracy values and numbers of analyzed features for our algorithm of local feature selection with feature subsets each including one feature

DB	Base classifiers: C4.5 with pruning			Integration method						Features		
	min	aver	max	agree	cover	CVM	WV	DS	DV	min	aver	max
Breast	0.662	0.696	0.735	0.473	0.882	0.674	0.697	0.697	0.697	9	9	9
	0.396	0.699	0.939	0.552	0.810	0.676	0.714	0.678	0.717	2.100	3.431	6.433
	0.535	0.711	0.872	0.533	0.819	0.673	0.710	0.674	0.709	2.100	4.431	7.533
DNA promoter	0.282	0.540	0.854	0.000	1.000	0.765	0.640	0.626	0.518	57	57	57
	0.245	0.641	0.909	0.581	0.922	0.762	0.767	0.728	0.729	1.100	2.054	3.100
	0.151	0.612	0.940	0.503	0.938	0.758	0.756	0.685	0.690	1.433	2.319	3.367
Glass	0.282	0.450	0.595	0.002	0.882	0.560	0.629	0.554	0.604	9	9	9
	0.292	0.463	0.646	0.035	0.847	0.559	0.587	0.535	0.590	2.500	6.248	8.000
	0.295	0.486	0.746	0.035	0.847	0.558	0.595	0.552	0.609	2.500	6.495	8.567
Heart	0.489	0.643	0.801	0.006	1.000	0.719	0.795	0.646	0.753	13	13	13
	0.394	0.651	0.881	0.133	0.985	0.718	0.755	0.628	0.717	2.367	5.781	7.967
	0.382	0.653	0.876	0.125	0.986	0.716	0.757	0.622	0.715	2.367	5.929	8.033
Iris	0.498	0.769	0.966	0.341	0.984	0.942	0.932	0.933	0.926	4	4	4
	0.671	0.848	0.961	0.865	0.966	0.935	0.930	0.934	0.937	1.000	1.633	2.433
	0.441	0.756	0.965	0.848	0.972	0.938	0.930	0.929	0.928	1.000	1.765	2.933
LED	0.162	0.217	0.259	0.046	0.462	0.233	0.245	0.262	0.255	7	7	7
	0.062	0.207	0.354	0.146	0.426	0.236	0.257	0.245	0.251	2.033	4.129	6.533
	0.119	0.201	0.281	0.085	0.455	0.237	0.255	0.246	0.243	3.733	5.711	7.000
LED17	0.037	0.100	0.192	0.000	0.688	0.138	0.103	0.244	0.148	24	24	24
	0.022	0.175	0.390	0.007	0.446	0.164	0.173	0.196	0.198	2.867	4.265	6.233
	0.000	0.126	0.500	0.001	0.533	0.155	0.175	0.229	0.213	3.800	6.697	10.267
Liver	0.469	0.545	0.620	0.058	0.969	0.546	0.572	0.581	0.593	6	6	6
	0.470	0.551	0.629	0.090	0.964	0.559	0.589	0.581	0.603	1.333	5.650	6.000
	0.470	0.551	0.629	0.090	0.964	0.559	0.589	0.581	0.603	1.333	5.650	6.000
Lympho-graphy	0.446	0.577	0.778	0.028	0.988	0.717	0.581	0.753	0.576	18	18	18
	0.152	0.550	0.908	0.368	0.886	0.703	0.680	0.661	0.627	1.733	3.579	6.267
	0.100	0.573	0.956	0.271	0.947	0.715	0.657	0.656	0.617	2.000	4.898	7.767
MONK-1	0.414	0.496	0.757	0.070	0.960	0.757	0.531	0.515	0.487	6	6	6
	0.291	0.490	0.768	0.349	0.920	0.757	0.637	0.609	0.607	1.000	3.060	5.233
	0.298	0.492	0.768	0.345	0.930	0.757	0.638	0.608	0.605	1.000	3.129	5.400
MONK-3	0.467	0.599	0.811	0.282	1.000	0.793	0.568	0.786	0.537	6	6	6
	0.460	0.704	0.872	0.588	1.000	0.793	0.770	0.627	0.629	2.000	2.115	3.000
	0.460	0.704	0.872	0.588	1.000	0.793	0.770	0.627	0.629	2.000	2.115	3.000
Soybean	0.152	0.449	0.861	0.000	1.000	0.770	0.406	0.852	0.900	35	35	35
	0.550	0.727	0.909	0.527	0.891	0.785	0.812	0.715	0.679	1.433	2.294	2.633
	0.550	0.727	0.909	0.527	0.891	0.785	0.812	0.715	0.679	1.433	2.294	2.633
Zoo	0.351	0.499	0.714	0.203	0.862	0.713	0.541	0.651	0.393	16	16	16
	0.000	0.338	0.681	0.504	0.771	0.572	0.577	0.575	0.558	1.233	2.713	4.867
	0.000	0.310	0.705	0.504	0.790	0.600	0.586	0.575	0.558	1.233	2.886	5.500
Average	0.362	0.506	0.688	0.116	0.898	0.641	0.557	0.623	0.568			
	0.308	0.542	0.757	0.365	0.833	0.632	0.634	0.593	0.603			
	0.292	0.531	0.771	0.343	0.852	0.634	0.633	0.592	0.600			

Appendix 2. Accuracy values and numbers of analyzed features for our algorithm of local feature selection with feature subsets each including two features

DB	Base classifiers: C4.5 with pruning				Integration method					Features		
	min	aver	max	agree	cover	CVM	WV	DS	DV	min	aver	max
Breast	0.644	0.716	0.787	0.389	0.929	0.733	0.728	0.722	0.709	9	9	9
	0.125	0.648	0.989	0.603	0.810	0.733	0.734	0.733	0.732	2.167	3.251	6.000
	0.375	0.754	0.972	0.557	0.832	0.733	0.735	0.729	0.728	2.267	4.777	7.600
DNA	0.240	0.544	0.869	0.000	1.000	0.711	0.650	0.728	0.621	57	57	57
promoter	0.111	0.663	0.979	0.490	0.911	0.714	0.767	0.715	0.751	1.033	2.014	3.200
	0.084	0.654	0.979	0.383	0.939	0.714	0.765	0.685	0.722	1.367	2.314	3.467
Glass	0.309	0.524	0.701	0.000	0.972	0.617	0.679	0.646	0.694	9	9	9
	0.188	0.510	0.747	0.078	0.919	0.590	0.623	0.551	0.607	2.500	6.258	8.067
	0.218	0.530	0.788	0.077	0.922	0.593	0.631	0.574	0.617	2.500	6.571	8.700
Heart	0.448	0.666	0.815	0.004	1.000	0.742	0.815	0.750	0.803	13	13	13
	0.165	0.641	0.967	0.113	0.976	0.732	0.755	0.702	0.751	2.033	5.850	7.900
	0.141	0.645	0.983	0.103	0.977	0.734	0.754	0.696	0.752	2.033	5.939	7.967
Iris	0.716	0.906	0.965	0.682	0.990	0.936	0.955	0.934	0.934	4	4	4
	0.871	0.951	0.992	0.934	0.966	0.944	0.951	0.946	0.949	1.000	1.717	2.433
	0.745	0.909	0.999	0.924	0.975	0.944	0.951	0.944	0.947	1.000	1.849	2.900
LED	0.217	0.316	0.414	0.058	0.835	0.371	0.362	0.462	0.348	7	7	7
	0.013	0.245	0.525	0.266	0.706	0.394	0.432	0.405	0.413	2.200	4.119	6.533
	0.087	0.265	0.440	0.152	0.814	0.424	0.389	0.404	0.342	3.800	5.729	7.000
LED17	0.026	0.152	0.334	0.000	0.985	0.253	0.245	0.423	0.300	24	24	24
	0.006	0.231	0.735	0.079	0.731	0.354	0.371	0.329	0.344	2.767	4.274	6.600
	0.000	0.171	0.959	0.022	0.851	0.321	0.376	0.323	0.325	3.567	6.693	10.233
Liver	0.468	0.570	0.666	0.010	1.000	0.581	0.634	0.597	0.644	6	6	6
	0.474	0.571	0.669	0.049	0.990	0.584	0.637	0.591	0.642	1.000	5.691	6.000
	0.474	0.571	0.669	0.049	0.990	0.584	0.637	0.591	0.642	1.000	5.691	6.000
Lympho-	0.395	0.620	0.834	0.004	1.000	0.766	0.745	0.761	0.746	18	18	18
graphy	0.022	0.671	0.989	0.470	0.887	0.739	0.749	0.703	0.705	1.533	3.473	5.633
	0.000	0.636	1.000	0.355	0.938	0.743	0.745	0.710	0.706	1.667	4.728	7.300
MONK-1	0.370	0.556	0.842	0.012	1.000	0.839	0.753	0.988	0.660	6	6	6
	0.234	0.635	1.000	0.512	0.976	0.963	0.764	0.879	0.796	1.000	3.073	5.167
	0.231	0.624	1.000	0.505	0.991	0.967	0.764	0.885	0.797	1.000	3.125	5.267
MONK-3	0.408	0.672	0.975	0.125	1.000	0.975	0.741	0.983	0.678	6	6	6
	0.422	0.747	0.973	0.909	1.000	0.975	0.975	0.978	0.965	1.867	2.078	3.000
	0.422	0.747	0.973	0.909	1.000	0.975	0.975	0.978	0.965	1.867	2.078	3.000
Soybean	0.085	0.498	1.000	0.000	1.000	0.961	0.655	0.964	0.842	35	35	35
	0.756	0.942	1.000	0.797	0.958	0.918	0.918	0.906	0.906	1.433	2.221	2.633
	0.756	0.942	1.000	0.797	0.958	0.918	0.918	0.906	0.906	1.433	2.221	2.633
Zoo	0.352	0.603	0.858	0.216	0.941	0.810	0.754	0.804	0.749	16	16	16
	0.073	0.695	0.973	0.628	0.867	0.781	0.812	0.717	0.729	1.200	2.748	4.933
	0.011	0.627	0.987	0.609	0.888	0.790	0.807	0.697	0.710	1.200	2.893	5.433
Average	0.360	0.565	0.774	0.115	0.973	0.715	0.670	0.751	0.671			
	0.266	0.627	0.888	0.456	0.900	0.725	0.730	0.704	0.715			
	0.273	0.621	0.904	0.419	0.929	0.726	0.727	0.702	0.705			