Multimodal Classification: Case Studies

Andrzej Skowron[1], Hui Wang[2], Arkadiusz Wojna[3], and Jan Bazan[4]

[1] Institute of Mathematics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
skowron@mimuw.edu.pl

[2] School of Computing and Mathematics
University of Ulster at Jordanstown
Northern Ireland, BT37 0QB, United Kingdom
h.wang@ulst.ac.uk

[3] Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
wojna@mimuw.edu.pl

[4] Institute of Mathematics
University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
bazan@univ.rzeszow.pl

Abstract. Data models that are induced in classifier construction often consist of multiple parts, each of which explains part of the data. Classification methods for such multi-part models are called multimodal classification methods. The model parts may overlap or have insufficient coverage. How to deal best with the problems of overlapping and insufficient coverage? In this paper we propose a hierarchical or layered approach to this problem. Rather than seeking a single model, we consider a series of models under gradually relaxing conditions, which form a hierarchical structure. To demonstrate the effectiveness of this approach we consider two classifiers that construct multi-part models – one based on the so-called lattice machine and the other one based on rough set rule induction, and we design hierarchical versions of the two classifiers. The two hierarchical classifiers are compared through experiments with their non-hierarchical counterparts, and also with a method that combines k-nearest neighbors classifier with rough set rule induction as a benchmark. The results of the experiments show that this hierarchical approach leads to improved multimodal classifiers.

Keywords: hierarchical classification, multimodal classifier, lattice machine, rough sets, rule induction, k-nearest neighbors.

1 Introduction

Many machine learning methods are based on generation of models with separate model parts, each of which explains part of a given dataset. Examples include decision tree induction [20], rule induction [7] and the lattice machine [33].

A decision tree consists of many branches, and each branch explains certain number of data examples. A rule induction algorithm generates a set of rules as a model of data, and each rule explains some data examples. The lattice machine generates a set of hypertuples as a model of data, and each hypertuple covers a region in the data space. We call this type of learning *multimodal learning* or *multimodal classification*.

In contrast some machine learning paradigms do not construct models with separate parts. Examples include neural networks, support vector machines and Bayesian networks.

In the multimodal learning paradigm the model parts may overlap or may have insufficient coverage of a data space, i.e., the model does not cover the whole data space. In a decision tree the branches do not overlap and cover the whole data space. In the case of rule induction, the rules may overlap and may not cover the whole data space. In the case of lattice machine the hypertuples overlap and the covering of the whole data space is not guaranteed too.

Overlapping makes it possible to label a data example by more than one class whereas insufficient coverage makes it possible that a data example is not labeled at all. How to deal best with the overlapping and insufficient coverage issues?

In this paper we consider a hierarchical strategy to answer this question. Most machine learning algorithms generate different models from data under different conditions or parameters, and they advocate some conditions for optimal models or let a user specify the condition for optimal models. Instead of trying to find the 'optimal' model we can consider a series of models constructed under different conditions. These models form a hierarchy, or a layered structure, where the bottom layer corresponds to a model with the strictest condition and the top layer corresponds to the one with the most relaxed condition. The models in different hierarchy layers correspond to different levels of pattern generalization.

To demonstrate the effectiveness of this strategy we consider two multimodal classifiers: one is the lattice machine (LM), and the other one is a rough set based rule induction algorithm RSES-O. We apply the hierarchical strategy in these two classifiers, leading to two new classification methods: HLM and RSES-H.

HLM is a hierarchical version of the lattice machine [33]. As mentioned earlier, the lattice machine generates hypertuples as model of data, but the hypertuples overlap (some objects are multiply covered) and usually only a part of the whole object space is covered by the hypertuples (some objects are not covered). Hence, for recognition of uncovered objects, we consider some more general hypertuples in the hierarchy that covers these objects. For recognition of multiply covered objects, we also consider more general hypertuples that cover (not exclusively) the objects. These covering hypertuples locate at various levels of the hierarchy. They are taken as neighborhoods of the object. A special voting strategy has been proposed to resolve conflicts between the object neighborhoods covering the classified object.

The second method, called RSES-H, is a hierarchical version of the rule-based classifier (hereafter referred to by RSES-O) in RSES [22]. RSES-O is based on rough set methods with optimization of rule shortening. RSES-H constructs a hierarchy of rule-based classifiers. The levels of the hierarchy are defined by different levels of minimal rule shortening [6,22]. A given object is classified by the classifier from the hierarchy that recognizes the object and corresponds to the minimal generalization (rule shortening) in the hierarchy of classifiers.

We compare HLM and RSES-H through a series of experiments with their non-hierarchical counterparts, LM [30,32] and RSES-O. We also compare the two algorithms with a state of the art classifier, RIONA, which is a combination of rough sets with the k-nearest neighbors (kNN) classifier [15,22]. The evaluation of described methods was done through experiments with benchmark datasets from UCI Machine Learning Repository [9] and also with some artificially generated data. The results of our experiments show that in many cases the hierarchical strategy leads to improved classification accuracy.

This paper extends the paper [24]. In this paper we provide more details on how the layers of HLM and RSES-H are constructed and a brief description of the reference algorithm RIONA. We add experimental results for artificially generated data containing noise and analyze how the hierarchical methods deal with noise. We also analyze the statistical significance of the classification accuracy improvement provided by the hierarchical approach.

It is necessary to note that our hierarchical strategy to multimodal classification is different from the classical hierarchical classification framework (see, e.g., [11,27,19,8,3,2,17]), which aims at developing methods to learn complex, usually hierarchical, concepts. In our study we do not consider the hierarchical structure of the concepts in question; therefore our study is in fact a *hierarchical approach to flat classification.*

The paper is organized as follows. Section 2.1 introduces the lattice machine classifier LM used as the basis for the hierarchical HLM. Section 2.2 describes the rough set method RSES-O used as the basis for the hierarchical RSES-H. Section 2.3 presents the algorithm RIONA used in experiments as the reference classifier. In Section 3 we introduce the hierarchical classifier HLM and in Section 4 the hierarchical RSES-H is presented. Section 5 provides experimental results obtained for the described classifiers and Section 6 concludes the paper with a brief summary.

2 Multimodal Classifiers

In this section we present in some detail three multimodal classifiers. In later sections we will present their hierarchical counterparts.

2.1 The Lattice Machine

The lattice machine [30,32,33] is a machine learning paradigm that constructs a generalized version space from data, which serves as a model (or hypothesis) of data. A model is a hyperrelation, or a set of hypertuples (patterns), such that each hypertuple in the hyperrelation is equilabeled, supported, and maximal. Being equilabeled means the model is consistent with data (i.e., matches objects with the same decision only); being maximal means the model has generalization capability; and being supported means the model does not generalize beyond the information given in the data. When data come from Euclidean space, the model is a set of hyperrectangles consistently, tightly and maximally approximating the data. Observe that, this approach is different from decision tree induction, which aims at partition of the data space. Lattice machines have two basic operations: a construction operation to build a model of data, and a classification operation that applies the model to classify data. The model is in the form of a set of hypertuples [31]. To make this paper self-contained we review the concepts of hypertuple.

Let $R = \{a_1, a_2, \cdots, a_n\}$ be a set of attributes, and y be the class (or decision) attribute; $dom(a)$ be the domain of attribute $a \in R \cup \{y\}$. In particular we let $C = dom(y)$ – the set of class labels. Let $V \overset{\text{def}}{=} \prod_{i=1}^{n} dom(a_i)$ and $L \overset{\text{def}}{=} \prod_{i=1}^{n} 2^{dom(a_i)}$. V is called the *data space* defined by R, and L an *extended data space*. A (given) *dataset* is $D \subseteq V \times C$ – a sample of V with known class labels. If we write an element $t \in V$ by $\langle v_1, v_2, \cdots, v_n \rangle$ then $v_i \in dom(a_i)$. If we write $h \in L$ by $\langle s_1, s_2, \cdots, s_n \rangle$ then $s_i \in 2^{dom(a_i)}$ or $s_i \subseteq dom(a_i)$. An element of L is called a *hypertuple*, and an element of V a *simple tuple*. The difference between the two is that a field in a simple tuple is a value (hence *value-based*) while a field in a hypertuple is a set (hence *set-based*). If we interpret $v_i \in dom(a_i)$ as a singleton set $\{v_i\}$, then a simple tuple is a special hypertuple. L is a lattice under the ordering [30]: for $s, t \in L$,

$$t \leq s \Longleftrightarrow t(x) \subseteq s(x) \tag{1}$$

with the sum and product operations given by

$$t + s = \langle t(x) \cup s(x) \rangle_{x \in R}. \tag{2}$$

$$t \times s = \langle t(x) \cap s(x) \rangle_{x \in R}. \tag{3}$$

Here $t(x)$ is the projection of t onto attribute x.

The Lm algorithm [31] constructs the unique model but it is not scalable to large datasets. The efficient algorithm CaseExtract, presented in [30], constructs such a model with the maximal condition relaxed. Such a model consists of a set of hypertuples which have disjoint coverage of the dataset.

Let D be a dataset, which is split into k classes: $D = \{D_1, D_2, \cdots, D_k\}$ where D_i and D_j are disjoint, $i \neq j$. The CaseExtract algorithm [30] is as follows:

```
– For $i = 1$ to $k$:
    • Initialization: let $X = D_i, H_i = \emptyset$.
    • Repeat until $X$ is empty:
        1. Let $h \in X$ and $X = X \setminus \{h\}$.
        2. For each $g \in X$, if $h+g$ is equilabeled then $h = h+g$ and $X = X \setminus \{g\}$
        3. Let $H_i = H_i \cup \{h\}$.
– $H = \bigcup_{i=1}^{k} H_i$ is a model of the data.
```

Note that $h + g$ is defined in Eq.(2). This algorithm bi-partitions X into a set of elements the sum of which is an equilabeled element, and a new X consisting of the rest of the elements. The new X is similarly bi-partitioned until X becomes empty.

When such a model is obtained, classification can be done by the C2 algorithm [32]. C2 distinguishes between two types of data: those that are covered by one and only one hypertuple (primary data), those that are covered by more than one hypertuple (secondary data) and those that are not covered (tertiary data). Classification is based on two measures. Primary data t is put in the same class as the hypertuple that covers t, and secondary and tertiary data are classified with the use of these two measures.

Let R be a set of attributes, $X \subseteq R$, V_X be the projection of V onto X, and $S \overset{\text{def}}{=} V_X$. V_X is the domain of X. When $X = R$, V_R is the whole data space, i.e., $V_R = V$. Consider a mass function $m : 2^S \to [0,1]$ such that $m(\emptyset) = 0$ and $\sum_{x \in 2^S} m(x) = 1$. Given $a, b \in 2^S$, where $m(b) \neq 0$, the first measure is derived by answering this question: what is the probability that b appears whenever a appears? In other words, if a appears, what is the probability that b will be regarded as appearing as well? Denoting this probability by $C^0_X(b|a)$, one solution is:

$$C^0_X(b|a) = \frac{\sum_{a \cup b \subseteq c} m(c)}{\sum_{b \subseteq c} m(c)}.$$

In the same spirit, another measure is defined as

$$C^1_X(b|a) = \frac{\sum_{c \subseteq b} m(c)}{\sum_{c \subseteq a \cup b} m(c)}.$$

$C^1_X(b|a)$ measures the degree in which merging a and b preserves the existing structure embodied by the mass function.

With the above two measures, the C2 algorithm for classification is as follows [32]. Let $t \in V$, and H be the set of hypertuples generated by the CaseExtract algorithm.

- For each $s \in H$, calculate $C^0_R(s|t)$ and $C^1_R(s|t)$.
- Let A be the set of $s \in H$ which have maximal C^0_X values. If A has only one element, namely $A = \{s\}$, then label t by the label of s. Otherwise, let B be the set of $s \in A$ which have maximal C^1_X values. If B has only one element, namely $B = \{s\}$, then label t by the label of s. Otherwise, label t by the label of the element in B which has the highest coverage.

Some variants of C2 are discussed in [33]. C2 performed extremely well on primary data, but not desirable on secondary and tertiary data.

2.2 RSES-O

The Rough Set Exploration System (RSES) (see [6,5,22]) is a freely available software system toolset for data exploration, classification support and knowledge discovery. Many of the RSES methods have originated from rough set theory introduced by Zdzisław Pawlak during the early 1980s (see [18]). At the moment of writing this paper RSES version 2.2 is the most recent (see [5] for more details).

One of the most popular methods for classifiers construction is based on learning rules from examples. Therefore there are several methods for calculation of the decision rule sets implemented in the RSES system (see [5,22]). One of these methods generates consistent decision rules with the minimal number of descriptors. This kind of decision rules can be used for classifying new objects as a standard rough set method of classifiers construction (see e.g. [23]).

Unfortunately, the decision rules consistent with the training examples can often be inappropriate to classify unseen cases. This happens, e.g. when the number of examples supporting a decision rule is relatively small. Therefore in practice we often use approximate rules instead of consistent decision rules. In RSES we have implemented a method for computing approximate rules (see e.g. [4]). In our method we begin with algorithms for synthesis of consistent decision rules with the minimal number of descriptors from a given decision table. Next, we compute approximate rules from already calculated consistent decision rules using the consistency coefficient. For a given training table D the consistency coefficient *cons* of a given decision rule $\alpha \to q$ (q is the decision class label) is defined by:

$$cons(\alpha \to q) = \frac{\|\{x \in D_q : x \text{ satisfies } \alpha\}\|}{\|\{x \in D : x \text{ satisfies } \alpha\}\|}$$

where D_q denotes the decision class corresponding to q. The original consistent decision rules with the minimal number of descriptors are reduced to approximate rules with consistency coefficient exceeding a fixed (optimal) threshold.

The resulting rules are shorter, more general (can be applied to more training objects) but they may lose some of their precision, i.e., may provide wrong answers (decisions) for some of the matching training objects. In exchange for this we expect to receive more general rules with higher quality of classification for new cases.

The method of classifier construction based on approximate rules is called the RSES-O method.

2.3 Rule Induction with Optimal Neighborhood Algorithm (RIONA)

RIONA [15] is a classification algorithm implemented in RSES [6,22] that combines the kNN classifier with rule induction. The method induces a distance measure and distance-based rules. For classification of a given test object the examples most similar to this object vote for decisions but first they are compared against the rules and the examples that do not match any rule are excluded from voting.

First the algorithm induces a distance measure ρ from a data sample D. The distance measure is defined by the weighted sum of the distance measures ρ_i for particular attributes a_i:

$$\rho(x,y) = \sum_{i=1}^{n} w_i \cdot \rho_i(a_i(x), a_i(y)).$$

RIONA uses the combination of the normalized city-block Manhattan metric for numerical attributes and the Simple Value Difference (SVD) metric for nominal attributes [15]. The distance between values of a numerical attribute a_i is defined by the absolute value difference between these values normalized by the range of attribute values in the data sample D:

$$\rho_i(a_i(x), a_i(y)) = \frac{|a(x) - a(y)|}{a_{\max} - a_{\min}}.$$

where $a_{\min} = \min_{x \in D} a_i(x)$ and $a_{\max} = \max_{x \in D} a_i(x)$. The SVD distance between values of a nominal attribute a_i is defined by the difference between the decision distributions for these values in the data sample D:

$$\rho_i(a_i(x), a_i(y)) = \sum_{q \in C} \left| P(z \in D_q | z \in D^{a_i(x)}) - P(z \in D_q | z \in D^{a_i(y)}) \right|.$$

where $D^{a_i(x_0)} = \{x \in D : a_i(x) = a_i(x_0)\}$. The weights w_i are optimized with the iterative attribute weighting procedure from [35].

To classify a tuple t RIONA uses the k nearest neighbors $n_1(t), \ldots, n_k(t)$ of t in the data sample D according to the previously defined distance measure ρ. Before voting the nearest neighbors are examined with consistent maximal rules derived from the data sample D [15]. If there is no consistent maximal rule that covers both a given neighbor $n_j(t)$ and the tuple t, the neighbor $n_j(t)$ is excluded from voting. The neighbors that share at least one maximal consistent rule with the tuple t are assigned with the vote weights v_j inversely proportional to square of the distance to t:

$$v_j(t) = \begin{cases} \frac{1}{\rho(n_j(t),t)^2} & \text{if there is consistent maximal rule covering } t \text{ and } n_j(t) \\ 0 & \text{otherwise} \end{cases}.$$

The tuple t is classified by q with the largest sum of nearest neighbor votes $S(t, q) = \sum_{n_j(t) \in D_q} v_j(t)$, where $1 \leq j \leq k$.

The value of k is optimized automatically in the range $1 \leq k \leq 100$ by the efficient leave-one-out procedure [15] applied to the data sample D.

3 HLM: Hierarchical Lattice Machine

In this section we present an implementation of our hierarchical approach to multimodal classification. This is a hierarchical version of the lattice machine, referred to by HLM.

We implement the hierarchical strategy in the lattice machine with the expectation that the classification accuracy of the lattice machine can be improved. Here is an outline of the solution.

We apply the CaseExtract algorithm repeatedly to construct a hierarchy of hypertuples. The bottom layer is constructed by CaseExtract directly from data. Then those data that are covered by the hypertuples with small coverage are marked out in the dataset, and the algorithm is applied again to construct a second layer. This process is repeated until a layer only with one hypertuple is reached. At the bottom layer all hypertuples are equilabeled, while those at higher layers may not be equilabeled.

To classify a data tuple (query) we search through the hierarchy to find a hypertuple at the lowest possible layer that covers the query. Then all data (including both marked and unmarked) covered by the hypertuple are weighted by an efficient counting-based weighting method. The weights are aggregated and used to classify the query. This is similar to the weighted kNN classifier, but it uses counting instead of distance to weigh relevant data.

3.1 Counting-Based Weighting Measure

In this section we present a counting-based weighting measure, which is suitable for use with hypertuples.

Suppose we have a neighborhood D for a query tuple (object) t and elements in D may come from any class. In order to classify the query based on the neighborhood we can take a majority voting with or without weighting. This is the essence of the well-known kNN classifier [14,12].

Weighting is usually done by the reverse of distance. Distance measures usually work for numerical data. For categorical data we need to transform the data into numerical form first. There are many ways for the transformation (see for example [26,10,34]), but most of them are task (e.g., classification) specific.

We present a general weighting method that allows us to count the number of all hypertuples, generated by the data tuples in a neighborhood of a query tuple t, that cover both t and any data tuple x in the neighborhood. Intuitively the higher the count the more relevant this x is to t, hence x should play a bigger role (higher weight). The inverse of this count can be used as a measure of distance between x and t. Therefore, by this count we can order and weight the data tuples. This counting method works for both numerical and categorical data in a conceptually uniform way. We consider next an efficient method to calculate this count.

As a measure of weighting we determine, for tuples t and x in D, the number of hypertuples that cover both t and x. We call this number the *cover* of t and x, denoted by $cov(t, x)$. The important issue here is how to calculate $cov(t, x)$ for every pair (t, x).

Consider two simple tuples $t =< t_1, t_2, \cdots, t_n >$ and $x =< x_1, x_2, \cdots, x_n >$. t is a simple tuple to be classified (query) and x is any simple tuple in D. What we want is to find all hypertuples that cover both t and x. We look at every attribute and explore the number of subsets that can be used to generate a hypertuple covering both t and x. Multiplying these numbers across all attributes gives rise to the number we require.

Consider an attribute a_i. If a_i is numerical, N_i denotes the number of intervals that can be used to generate a hypertuple covering both t_i and x_i. If a_i is categorical, N_i denotes the number of subsets for the same purpose. Assuming that all attributes have finite domains, we have [29]:

$$N_i = \begin{cases} (\max(a_i) - \max(\{x_i, t_i\}) + 1) \times (\min(\{x_i, t_i\}) - \min(a_i) + 1) \\ \qquad\qquad \text{if } a_i \text{ is numerical} \\ 2^{m_i - 1} \quad \text{if } a_i \text{ is categorical and } x_i = t_i \\ 2^{m_i - 2} \quad \text{if } a_i \text{ is categorical and } x_i \neq t_i. \end{cases} \tag{4}$$

where $\max(a_i)$, $\min(a_i)$ are the maximal and the minimal value of a_i, respectively, if a_i is numerical, and $m_i = |dom(a_i)|$, if a_i is categorical.

The number of covering hypertuples of t and x is $cov(t, x) = \prod_i N_i$.

A simple tuple $x \in D$ is then weighted by $cov(t, x)$ in a kNN classifier. More specifically, we define

$$K(t, q) = \sum_{x \in D_q} cov(t, x).$$

where D_q is a subset of D consisting of all q class simple tuples. $K(t, q)$ is the total of the cover of all q class simple tuples. Then the weighted kNN classifier is the following rule (wkNN rule):

t is classified by q_0 that has the largest $K(t, q)$ for all q.

3.2 The Classification Procedure

We now present a classification procedure, called, *hierarchical classification based on weighting* (HLM).

Let D be a given dataset, let HH be a hierarchy of hypertuples constructed from D, and let t be a query – a simple tuple to be classified.

Step 1. Search HH in the bottom up order and stop as soon as a covering hypertuple is found at layer l. Continue searching layer l until all covering hypertuples are found. Let S be a set of all covering hypertuples from this layer;

Step 2. Let $N \leftarrow \{\underline{h} : h \in S\}$, a neighborhood of the query;

Step 3. Apply wkNN to classify t.

Note that $\underline{h}$ is the set of simple tuples covered by h.

4 RSES-H: Hierarchical Rule-Based Classifier

In this section we present another implementation of our hierarchical approach to multimodal classification. This is a hierarchical version of RSES-O, referred to by RSES-H.

In RSES-H a set of minimal decision rules [7,22] is generated. Then, different layers for classification are created by rule shortening. The algorithm works as follows:

1. At the beginning, we divide original data sets into two disjoint parts: train table and test table.
2. Next, we calculate (consistent) rules with a minimal number of descriptors for the train table (using covering method from RSES [7,22]). This set of rules is used to construct the first (the bottom) level of our classifier.

3. In the successive steps defined by the following consistency thresholds (after rule shortening): $0.95, 0.90, 0.85, 0.80, 0.75, 0.70, 0.65, 0.60$, we generate a set of rules obtained by shortening all rules generated in the previous step. The rules generated in the i-th step are used to construct the classifier with the label $i + 1$ in the classifier hierarchy.
4. Now, we can use our hierarchical classifier in the following way:
   (a) For any object from the test table, we try to classify this object using decision rules from the first level of our classifier.
   (b) If the tested object is classified by rules from the first level of classifier, we return the decision value for this object and the remaining levels of our classifier are not used.
   (c) If the tested object can not be classified by rules from the first level, we try to classify it using the second level of our hierarchical classifier, etc.
   (d) Finally, if the tested object can not be classified by rules from the level with the label 9, then our classifier can not classify the tested object. The last case happens seldom, because higher levels are usually sufficient for classifying any tested object.

The range of thresholds for the rule consistency (see Section 2.2) in the third step of the algorithm presented above have been determined on the basis of experience obtained from the previous experiments (see e.g. [1]). The step between thresholds has been determined to 0.05, because this allows us to make rule search quite precise and the number of thresholds (that have to be checked) is not too large from the computational complexity point of view.

5 Evaluation

Experiments were performed with the two hierarchical classifiers (HLM and RSES-H) described in Section 3.1 (HLM) and Section 4 (RSES-H), their non-hierarchical counterparts (LM and RSES-O based on rules) and RIONA as a benchmark classifier. The purpose of the experiment was two fold: first, we wanted to know whether the hierarchical algorithms improve their non-hierarchical counterparts. Second, we wanted to know the correspondence between the degree of improvement and distribution of data.

For this purpose we considered two types of data: real world data and artificial (or synthetic) data. The former were some popular benchmark datasets from UCI Machine Learning Repository [9], and some simple statistics are shown in Table 1. The latter were generated by Handl and Knowles [16]. The generator is based on a standard cluster model using multivariate normal distributions. We generated six datasets: three of them have two clusters labeled as two separate classes (unimodal data), and the remaining three have four clusters grouped again into two classes (multimodal data). In all cases 20% random noise were added.

In the experiment each classifier was tested 10 times on each dataset with the use of 5-fold cross-validation.

Table 1. General information on the datasets and the 5-fold cross validation success rate with standard deviation of LM, HLM, RSES-H, RSES-O and RIONA

Data	General Info			5CV success rate				
	Att	Exa	Cla	LM	HLM	RSES-O	RSES-H	RIONA
Anneal	38	798	6	95.7±0.3	96.0±0.4	94.3±0.6	96.2±0.5	92.5
Austral	14	690	2	91.9±0.3	92.0±0.4	86.4±0.5	87.0±0.5	85.7
Auto	25	205	6	73.0±1.5	76.5±1.4	69.0±3.1	73.7±1.7	76.7
Diabetes	8	768	2	70.6±0.6	72.6±0.8	73.8±0.6	73.8±1.2	75.4
Ecoli	7	336	8	79.8±1.0	85.6±0.7	72.4±2.3	76.0±1.7	84.1
German	20	1000	2	69.8±0.6	71.4±0.9	72.2±0.4	73.2±0.9	74.4
Glass	9	214	3	63.5±1.2	71.3±1.2	61.2±2.5	63.4±1.8	66.1
Heart	13	270	2	75.2±1.8	79.0±1.0	83.8±1.1	84.0±1.3	82.3
Hepatitis	19	155	2	77.2±0.7	78.7±1.2	82.6±1.3	81.9±1.6	82.0
Horse-Colic	22	368	2	78.2±0.8	76.3±0.9	85.5±0.5	86.5±0.6	84.6
Iris	4	150	3	95.0±0.4	94.1±0.4	94.9±1.5	95.5±0.8	94.4
Sonar	60	208	2	74.2±1.2	73.7±0.8	74.3±1.8	75.3±2.0	86.1
TTT	9	958	2	94.0±0.7	95.0±0.3	99.0±0.2	99.1±0.2	93.6
Vehicle	18	846	4	69.4±0.5	67.6±0.7	64.2±1.3	66.1±1.4	70.2
Vote	18	232	2	96.4±0.5	95.4±0.5	96.4±0.5	96.5±0.5	95.3
Wine	12	178	3	96.4±0.4	92.6±0.8	90.7±2.2	91.2±1.2	95.4
Yeast	8	1484	10	49.9±0.6	51.3±0.7	50.7±1.2	51.9±0.9	58.9
D20c22n0	20	522	2	85.0±0.8	89.4±0.6	88.9±0.9	88.8±1.2	91.4
D20c22n1	20	922	2	87.6±0.5	89.1±0.5	90.1±0.6	90.1±1.0	86.9
D20c22n2	20	838	2	89.2±0.5	91.2±0.3	90.3±0.4	89.9±0.9	89.4
D20c42n0	20	1370	2	81.4±0.5	85.5±0.3	83.6±1.0	84.4±1.4	90.9
D20c42n1	20	1558	2	80.1±0.3	83.8±0.3	88.5±0.4	88.7±0.6	87.1
D20c42n2	20	1524	2	77.9±0.8	79.0±0.5	79.6±0.7	79.8±1.0	83.2
Average success rate				80.53	82.05	81.41	82.3	83.77

The average results with standard deviations are shown in Table 1. HLM obtained the higher accuracy than its non-hierarchical counterpart LM on 17 data sets and it lost on 6 data sets. The best improvements were for the data sets *Glass* (7.8%) and *Ecoli* (5.8%). The difference between RSES-H and its non-hierarchical counterpart RSES-O is even more distinct: RSES-H outperformed RSES-O on 19 data sets and lost on 3 data sets only. The best improvements were for the data sets *Auto* (4.7%) and *Ecoli* (3.6%).

The supremacy of the hierarchical methods over the non-hierarchical ones can be also noticed in the total average accuracy: HLM accuracy is 1.5% higher than LM accuracy and similarly RSES-H accuracy is almost 1% higher than RSES-O accuracy. On average both hierachical methods ouperformed both non-hierarchical methods.

The benchmark classifier RIONA has the highest total average accuracy but the difference to hierarchical methods is much smaller than to non-hierarchical classifiers. The advantage of RIONA over HLM and RSES-H comes from a few specific data sets (*Sonar*, *Ecoli* and *Yeast*) where the nearest neighbor component helps a lot in overcoming the problem of a large number of attributes or classes.

Table 2. The levels of statistical significance of difference when comparing the hierarchical methods against the non-hierarchical methods: 5 is 99.5%, 4 is 99%, 3 is 97.5%, 2 is 95%, 1 is 90% and 0 is below 90%. Plus indicates that the average accuracy of a hierarchical method is higher than that of a non-hierarchical method and minus otherwise.

Data	General Info			Statistical significance			
	Attrib	Exampl	Classes	HLM vs		RSES-H vs	
				LM	RSES-O	LM	RSES-O
Anneal	38	798	6	+3	+5	+5	+5
Austral	14	690	2	+0	+5	−5	+3
Auto	25	205	6	+5	+5	+0	+5
Diabetes	8	768	2	+5	−5	+5	0
Ecoli	7	336	8	+5	+5	−5	+5
German	20	1000	2	+5	−4	+5	+5
Glass	9	214	3	+5	+5	−0	+3
Heart	13	270	2	+5	−5	+5	+0
Hepatitis	19	155	2	+5	−5	+5	−0
Horse-Colic	22	368	2	−5	−5	+5	+5
Iris	4	150	3	−5	−1	+2	+0
Sonar	60	208	2	−0	−0	+1	+0
TTT	9	958	2	+5	−5	+5	+1
Vehicle	18	846	4	−5	+5	−5	+4
Vote	18	232	2	−5	−5	+0	+0
Wine	12	178	3	−5	+4	−5	+0
Yeast	8	1484	10	+5	+1	+5	+3
D20c22n0	20	522	2	+5	+1	+5	−0
D20c22n1	20	922	2	+5	−5	+5	+0
D20c22n2	20	838	2	+5	+5	+3	−0
D20c42n0	20	1370	2	+5	+5	+5	+1
D20c42n1	20	1558	2	+5	−5	+5	+0
D20c42n2	20	1524	2	+5	−3	+5	+0
Wins/Losses quite probable ($>$ 90%)				16/5	11/11	16/4	11/0
Wins/Losses certain ($>$ 99.5%)				15/5	8/8	13/4	5/0

One could ask whether the differences in accuracy between the hierarchical and non-hierarchical methods are really significant. To answer this question in Table 2 we compared the hierarchical HLM and RSES-H against the non-hierarchical LM and RSES-O and provided the statistical significance of differences between the accuracy of classifiers on particular data sets using the one-tail unpaired Student's t-test [25].

Comparing HLM against LM (see Figure 1) one can see that for almost all the data sets the differences are significant. In other words, in 16 cases HLM provided the statistically significant improvement in accuracy over LM (for 15 data sets this improvement is practically certain) and the accuracy has significantly fallen in 5 cases. This confirms the conclusion that, in general, it is worth to apply hierarchical HLM instead of non-hierarchical LM.

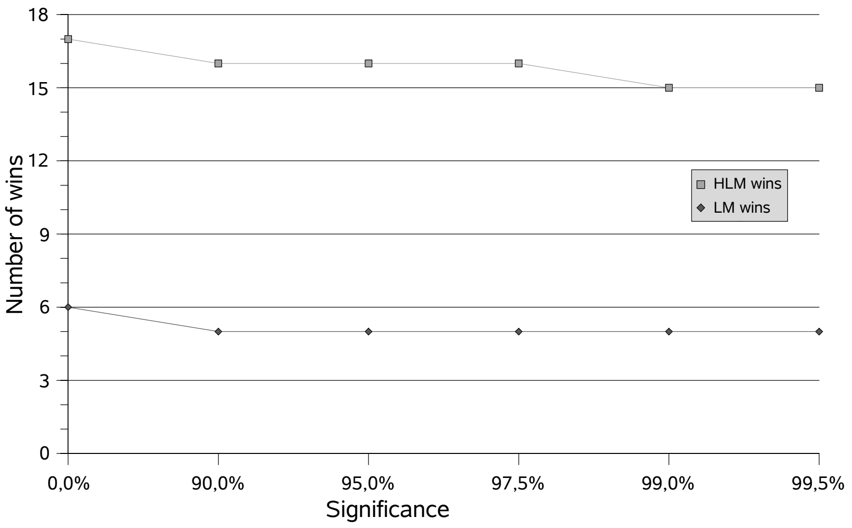

Fig. 1. The number of data sets where HLM outperforms LM and the number of datasets where HLM loses in dependence of significance level of accuracy difference

The comparison between HLM and RSES-O does not show supremacy of any method. However, for some datasets (*Australian*, *Auto*, *Ecoli* and *Glass*) HLM provided significantly better results than both RSES-H and RSES-O.

The relation between the results of RSES-H and LM is similar to the relation HLM vs LM. The differences in accuracy are significant for almost all the data sets and in most cases RSES-H outperformed LM.

The interesting relation is between the results of RSES-H and RSES-O (see Figure 2). There is no data set on which RSES-H was significantly worse than RSES-O. On other hand, in half of cases RSES-H improved significantly RSES-O. This indicates that the extension of RSES-O to RSES-H is rather stable: RSES-H keeps the level of the RSES-O accuracy. There are no risk while replacing RSES-O with RSES-H and a significant chance of improving the results.

Another interesting observation can be made when one focuses on artificial data. In comparison with LM both hierarchical methods provide significant improvement on all generated data sets. This indicates that lattice machine does not deal well with noisy data and both hierarchical methods do it better. Therefore in cases of noisy data the lattice machine is particularly recommended to be extended to hierarchical approach.

The different situation is in case of RSES-O. The comparison of both hierarchical methods with RSES-O on artificial data does not show the significant supremacy of any method. This suggests that to overcome the problem of noise the rule optimization used in RSES-O is equally effective as the hierarchical approach.

The experiments confirmed our original expectation that the performance of LM and RSES-O can be improved by our hierarchical approach. The cost is

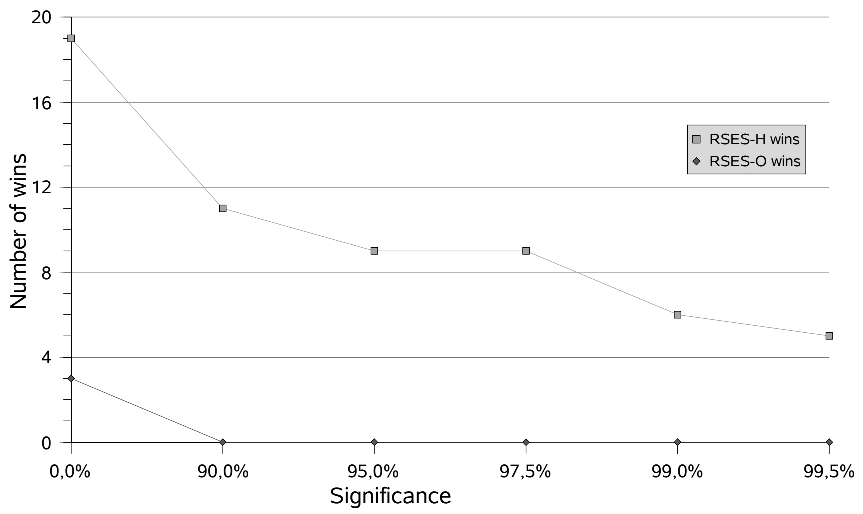

Fig. 2. The number of data sets where RSES-H outperforms RSES-O and the number of datasets where RSES-H loses in dependence of significance level of accuracy difference

some extra time to construct the hierarchy and to test some new objects using this hierarchy.

The experimental results provide also the hint which hierarchical method to use in dependence of the expected profit. RSES-H is safer: it usually provides smaller improvements than HLM but it never worsens the accuracy. Application of HLM can result in a greater improvement but there is also a higher risk of worsening the results.

6 Conclusions

In the paper we addressed the problem of balancing between accuracy and universality in multimodal classification models. The accurate model parts can be specific and cover part of data space only. On the other hand more general model parts can lose accuracy. To solve this problem we proposed the hierarchical approach to multimodal classification. The hierarchical approach introduces a number of multimodal layers instead of a single one, each with different accuracy and universality. Such a hierarchical model tries to classify data with the most accurate layer and if it fails then it moves through more and more general layers until an answer is found. Such a hierarchical approach provides a kind of adaptation to test case difficulty. The proper graduation between the successive levels allows to classify data with the optimal balance between accuracy and universality.

The experimental results confirmed that such a hierarchical model is more effective than one-layer multimodal methods: it gives higher classification accuracy. Hierarchical approach guarantees also a certain level of adaptability to data: it deals well with noise in data.

In future research we plan to develop hierarchical multimodal methods for incremental learning. Incremental learning requires reconstruction of particular layers when new examples arrive. In hierarchical model the reconstruction time of a layer can be reduced by the use of knowledge from the adjoining layers.

The other interesting research direction is the application of a hierarchical rule model in RIONA. In that classifier rules are used to validate and filter nearest neighbors (see Section 2.3). The interesting question is whether one can improve RIONA accuracy still more by replacing the single set of rules with a hierarchical rule model in the validation step.

Acknowledgments. The research has been supported by the grant 3 T11C 002 26 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. Bazan, J.: A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Table, Polkowski L., Skowron A. (eds.): *Rough Sets in Knowledge Discovery.* Heidelberg: Physica-Verlag 321-365, 1998.
2. J. Bazan. Classifiers based on two-layered learning. Lecture Notes in Artificial Intelligence 3066, Springer, Heidelberg, 356–361, 2004.
3. J. Bazan, S. Hoa Nguyen, H. Son Nguyen, A. Skowron. Rough set methods in approximation of hierarchical concepts. *Proc. of RSCTC'2004*, Lecture Notes in Artificial Intelligence 3066, Springer, Heidelberg, 346–355, 2004.
4. J. G. Bazan, H. Son Nguyen, S. Hoa Nguyen, P. Synak, J. Wróblewski. Rough Set Algorithms in Classification Problem, L. Polkowski, S. Tsumoto, T. Y. Lin, (eds.): Rough Set Methods and Applications. Heidelberg: Physica-Verlag 49-88, 2000.
5. J. G. Bazan, M. Szczuka. The Rough Set Exploration System. In *Transactions on Rough Sets* III, Lecture Notes in Computer Science 3400, 2005, 37-56.
6. J. Bazan, M. Szczuka, A. Wojna, M. Wojnarski. On the evolution of Rough Set Exploration System, Proc. of RSCTC'2004, Lecture Notes in Artificial Intelligence 3066, Springer, Heidelberg, 592–601, 2004.
7. J. G. Bazan, M. Szczuka, J. Wróblewski. A New Version of Rough Set Exploration System, Proc. of RSCTC'2002, Lecture Notes in Artificial Intelligence 2475, Springer-Verlag, Heidelberg, 397-404, 2002.
8. S. Behnke. Hierarchical Neural Networks for Image Interpretation. Lecture Notes in Artificial Intelligence 2766, Springer, Heidelberg, 2003.
9. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
10. S. Cost, S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
11. T. G. Dietterich. Ensemble Learning. In M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, Second edition, Cambridge, MA: The MIT Press, 405-408, 2002.
12. S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Trans. Syst. Man Cyber.*, 6:325–327, 1976.

13. Ivo Düntsch and Günther Gediga. Simple data filtering in rough set systems. *International Journal of Approximate Reasoning*, 18(1–2):93–106, 1998.
14. E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report TR4, USAF School of Aviation Medicine, Randolph Field, TX, 1951.
15. G. Góra and A. G. Wojna. RIONA: a new classification system combining rule induction and instance-based learning. *Fundamenta Informaticae*, 51(4):369–390, 2002.
16. J. Handl and J. Knowles. Cluster generators: synthetic data for the evaluation of clustering algorithms. `"http://dbkweb.ch.umist.ac.uk/handl/generators/"`.
17. S. Hoa Nguyen, J. Bazan, A. Skowron, H. Son Nguyen. Layered learning for concept synthesis. Lecture Notes in Artificial Intelligence 3100, *Transactions on Rough Sets I*:187–208, Springer, Heidelberg, 2004.
18. Z. Pawlak. Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, 1991.
19. T. Poggio, S. Smale. The Mathematics of Learning: Dealing with Data. *Notices of the AMS* 50(5):537-544, 2003.
20. Ross Quinlan. Improved Use of Continuous Attributes in C4.5. Journal of Artificial Intelligence Research, 4:77-90, 1996.
21. R. Quinlan. Rulequest research data mining tools. `http://www.rulequest.com/`.
22. RSES: Rough set exploration system. http://logic.mimuw.edu.pl/˜rses, Institute of Mathematics, Warsaw University, Poland.
23. A. Skowron. Boolean reasoning for decision rules generation, Proc. of ISMIS'93, Lecture Notes in Artificial Intelligence 689, Springer-Verlag, Heidelberg, 295–305, 1993.
24. A. Skowron, H. Wang, A. G. Wojna, J. G. Bazan. A Hierarchical Approach to Multimodal Classification, Proc. of RSFDGrC'2005, Lecture Notes in Artificial Intelligence 3642, Springer-Verlag, Heidelberg, 2005, 119–127.
25. G. W. Snedecor, W. G. Cochran. Statisitical Methods, Iowa State University Press, Ames, IA, 2002, eighth edition.
26. C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communication of ACM*, 29:1213–1229, 1986.
27. P. Stone. *Layered Learning in Multi-agent Systems: A Winning Approach to Robotic Soccer.* MIT Press, Cambridge, MA, 2000.
28. V. N. Vapnik. *Statistical learning theory.* Wiley New York, 1998.
29. H. Wang. Nearest neighbors by neighborhood counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6), June 2006.
30. H. Wang, W. Dubitzky, I. Düntsch, and D. Bell. A lattice machine approach to automated casebase design: Marrying lazy and eager learning. In *Proc. IJCAI99*, Stockholm, Sweden, 254–259, 1999.
31. H. Wang, I. Düntsch, D. Bell. Data reduction based on hyper relations. *Proceedings of KDD98, New York*, 349–353, 1998.
32. H. Wang, I. Düntsch, G. Gediga. Classificatory filtering in decision systems. *International Journal of Approximate Reasoning*, 23:111–136, 2000.
33. H. Wang, I. Düntsch, G. Gediga, A. Skowron. Hyperrelations in version space. *International Journal of Approximate Reasoning*, 36(3):223–241, 2004.
34. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.
35. A. G. Wojna. Analogy-Based Reasoning in Classifier Construction. *Transactions on Rough Sets* IV, Lecture Notes in Computer Science 3700, 2005, 277-374.