A Hierarchical Approach to Multimodal Classification

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 consists of multiple parts, each of which explains part of the data. Classification methods for such models are called the 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 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 implemented it in 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. This leads to hierarchical versions of the classifiers. The classification performance of these two hierarchical classifiers is compared with C4.5, Support Vector Machine (SVM), rule based classifiers (with the optimisation of rule shortening) implemented in Rough Set Exploration System (RSES), and a method combining k-nn with rough set rule induction (RIONA in RSES). 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-NN.

1 Introduction

Many machine learning methods are based on generation of different models with separate model parts, each of which explains part of a given dataset. Examples include decision tree induction [24], rule induction [1] and the lattice machine [21]. 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

D. Ślęzak et al. (Eds.): RSFDGrC 2005, LNAI 3642, pp. 119–127, 2005.
© Springer-Verlag Berlin Heidelberg 2005

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 implemented it in two classifiers that construct multi-part models: one based on the lattice machine, and the other one based on rough set rule induction. This leads to two new classification methods.

The first method, called HCW, is a hierarchical version of the CaseExtract/CE (or CE/CE for short) classifier – the operations in the lattice machine [21]. 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 neighbourhoods of the object. A special voting strategy has been proposed to resolve conflicts between the object neighbourhoods 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 [23]. RSES-O is based on rough set methods with optimisation 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 [2,23]. A given object is classified by the classifier from the hierarchy that recognizes the object and corresponds to the minimal generalisation (rule shortening) in the hierarchy of classifiers.

We compare the performance of HCW and RSES-H with the well known classifiers C5.0 [13], SVM [17], and also with CE/C2 [18,20], RSES-O, and RIONA that is a combination of rough sets with k-nn [7,23]. The evaluation of described methods was done through experiments with benchmark datasets from UCI Machine Learning Repository. The results of experiments show that in many cases the hierarchical approach leads to improved classification accuracy.

It is necessary to note that our hierarchical approach to multimodal classification is different from the classical hierarchical classification framework (see, e.g., [6,16,12,5,3,4,8]), 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.*

2 HCW: Hierarchical Lattice Machine

In this section we present one implementation of our hierarchical approach to multimodal classification. This is a hierarchical version of the CE/C2 algorithm in the lattice machine, referred to by HCW.

A lattice machine [18,21] is a machine learning paradigm that constructs a generalised 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 equilabelled, supported, and maximal. Being equilabelled means the model is consistent with data (i.e., matches objects with the same decision only); being maximal means the model has generalisation capability; and being supported means the model does not generalise 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 in the form of a set of hypertuples, and a classification operation that applies the model to classify data. The Lm algorithm [19] constructs the unique model but it is not scalable to large datasets. The efficient algorithm CaseExtract presented in [18] constructs such a model with the maximal condition relaxed. When such a model is obtained, classification can be done by the C2 algorithm [20]. C2 distinguishes between two types of data: those that are covered by at least one hypertuple (primary data), and those that are not (secondary data). Classification is based on two measures. Primary data t is put in the same class as the hypertuple that covers t, and secondary data are classified with the use of these two measures. Some variants of C2 are discussed in [21]. However, the classification accuracy is still not desirable for secondary data. Table 1 shows some experimental results by CE/C2. We can see clearly that C2 performed extremely well on primary data. Since the overall performance is the weighted average of those of primary and secondary data, it is clear that C2 performed poorly on secondary data.

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 equilabelled, while those at higher layers may not be equilabelled.

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 k-nearest neighbour method, but it uses counting instead of distance to weigh relevant data.

2.1 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 neighbourhood 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 neighbourhood we can take a majority voting with or without weighting. This is the essence of the well-known k-nearest neighbour (kNN) method [11,10].

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 [15,9,22]), 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 neighbourhood of a query tuple t, that cover both t and any data tuple x in the neighbourhood. 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 set to find out, for tuples t and x in D, the number of hypertuples that cover both t and x. We call this number the *h-count* of t and x, denoted by $cnt(t, x)$. The important issue here is how to calculate the h-count for every tuple in D.

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:

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

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 $cnt(t, x) = \prod_i N_i$.

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

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

where D_q is a subset of D consisting of all q class simple tuples. $K(t, q)$ is the total of the h-counts 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.

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

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 neighbourhood of the query;

Step 3. Apply wkNN to classify t.

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

3 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 [1,23] 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 [1,23]). 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 thresholds (for the positive region after 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 by 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.

4 Evaluation

The two hierarchical classifiers described in Section 2.1 (HCW) and in Section 3 (RSES-H) were evaluated by experiments on some popular benchmark datasets from UCI Machine Learning Repository[1]. Each classifier was tested 10 times on each dataset with the use of 5-fold cross-validation. The average results are shown in Table 1.

For comparison, we also include experimental results by the well-known decision tree C5.0 [13] and support vector machine (SVM) [17], the two non-hierarchical methods CE/C2 and RSES-O based on rules, and RIONA.

CE/C2 is a lattice machine based classifier [21]. The CaseExtract algorithm builds, as a classifying function, a set of overlapping hypertuples and the C2 algorithm classifies a data example.

RSES-O implemented in RSES [2,23] is a rule based method with the rule shortening optimization. It is based on the decision rules with the minimal number of descriptors and the operation of rule shortening [1]. The classifier is constructed over the set of rules obtained by shortening these minimal rules using the optimal threshold.

RIONA [7] is another classification algorithm implemented in RSES [2,23] that combines the k nearest neighbor method with rule induction. The method induces a distance measure and distance-based rules. For classification of a given

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Table 1. General information on the datasets and the 5-fold cross validation success rate of C5.0, SVM, CE/C2 on all data, CE/C2 on primary data, HCW, RSES-H, RSES-O and RIONA. The "PP" column is the percentage of primary data. Note that the SVM results for "German", "TTT" and "Vote" are not available because they have categorical attributes and this method does not work with categorical attributes directly. For HCW, RSES-H and RSES-O the standard deviations are also provided. In the columns HCW and RSES-H the superscripts denote the levels of statistical significance of difference between these classifiers and RSES-O: 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 the method is higher than in RSES-O and minus otherwise.

Data	General Info			5CV success rate								%PP
	Att	Exa	Cla	C5.0	SVM	CE/C2	Prim.	HCW	RSES-H	RSES-O	RIONA	
Anneal	38	798	6	96.6	91.3	96.8	97.6	$96.0\pm0.4^{+5}$	$96.2\pm0.5^{+5}$	94.3 ± 0.6	92.5	93.4
Austral	14	690	2	90.6	85.3	83.5	95.1	$92.0\pm0.4^{+5}$	$87.0\pm0.5^{+3}$	86.4 ± 0.5	85.7	87.2
Auto	25	205	6	70.7	68.3	76.1	86.8	$76.5\pm1.4^{+5}$	$73.7\pm1.7^{+5}$	69.0 ± 3.1	76.7	56.1
Diabetes	8	768	2	72.7	77.7	71.0	71.7	72.6 ± 0.8^{-5}	73.8 ± 1.2^{0}	73.8 ± 0.6	75.4	66.8
German	20	1000	2	71.7		72.5	72.6	71.4 ± 0.9^{-4}	$73.2\pm0.9^{+5}$	72.2 ± 0.4	74.4	65.4
Glass	9	214	3	62.4	63.9	64.0	69.6	$71.3\pm1.2^{+5}$	$63.4\pm1.8^{+3}$	61.2 ± 2.5	66.1	79.4
Heart	13	270	2	77.0	83.3	77.0	82.1	79.0 ± 1.0^{-5}	$84.0\pm1.3^{+0}$	83.8 ± 1.1	82.3	61.5
Hepatitis	19	155	2	80.0	80.7	81.2	84.1	78.7 ± 1.2^{-5}	81.9 ± 1.6^{-0}	82.6 ± 1.3	82.0	69.0
Iris	4	150	3	94.7	94.0	92.7	97.6	94.1 ± 0.4^{-1}	$95.5\pm0.8^{+0}$	94.9 ± 1.5	94.4	82.7
Sonar	60	208	2	71.6	72.2	69.7	81.4	73.7 ± 0.8^{-0}	$75.3\pm2.0^{+0}$	74.3 ± 1.8	86.1	59.2
TTT	9	958	2	86.2		83.5	96.2	95.0 ± 0.3^{-5}	$99.1\pm0.2^{+1}$	99.0 ± 0.2	93.6	94.9
Vehicle	18	846	4	71.9	76.8	75.7	76.7	$67.6\pm0.7^{+5}$	$66.1\pm1.4^{+4}$	64.2 ± 1.3	70.2	61.7
Vote	18	232	2	96.5		94.2	98.5	95.4 ± 0.5^{-5}	$96.5\pm0.5^{+0}$	96.4 ± 0.5	95.3	89.7
Average success rate				80.20	79.35	79.84	85.38	81.79	81.98	80.93	82.67	

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.

On average HCW and RSES-H outperform C5.0, SVM, CE/C2 and RSES-O and are almost the same as RIONA (the advantage of RIONA is only due to the dataset *Sonar*). To provide more details on the benefit from hierarchical approach we compared HCW and RSES-H against the non-hierarchical RSES-O (on average RSES-O gives better accuracy than than CE/C2) and computed the statistical significance of their difference using the one-tail unpaired Student's t-test [14].

Comparing RSES-H against RSES-O one can see that for 6 datasets RSES-H dominates with at least 97.5% confidence level and there is no dataset on which RSES-H was significantly worse.

The comparison between HCW and RSES-O does not show supremacy of any method. However, for some datasets (*Australian*, *Auto* and *Glass*) HCW provided significantly better results than both RSES-H and RSES-O. Moreover, it outperformed CE/C2 in 9 out of 13 datasets. The best improvements were for the datasets *TTT* (11.5%), *Australian* (8.5%) and *Glass* (7.3%). It is interesting to note that the performance of CE/C2 on the *Vehicle* dataset was significantly better (8.1%) than the performance of HCW. Observe also (see column Prim in Table 1) that CE/C2 performed extremely well on primary data, i.e., data covered by hypertuples induced by lattice machine.

The experiments confirmed our original expectation that the performance of CE/C2 and RSES-O can be improved by our hierarchical approach. The cost is some extra time to construct the hierarchy and to test some new objects using this hierarchy.

Observe also that for most of the tested datasets the best method selected from the discussed rough set methods (RSES-O, RSES-H), combination of rough sets with k-nn (RIONA), and lattice machine (CE/C2,HCW) outperforms the methods C5.0 and SVM.

5 Conclusions

The experimental study has shown that the discussed hierarchical approach leads to improvement of classification accuracy. We plan to develop more advanced methods for hierarchical classification in incremental learning.

Acknowledgements. The research has been supported by the grants 4 T11C 040 24 and 3 T11C 002 26 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. 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.
2. 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.
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. Bazan. Classifiers based on two-layered learning. Lecture Notes in Artificial Intelligence 3066, Springer, Heidelberg, 356–361, 2004.
5. S. Behnke. Hierarchical Neural Networks for Image Interpretation. Lecture Notes in Artificial Intelligence 2766, Springer, Heidelberg, 2003.
6. 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.
7. 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.
8. 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.
9. S. Cost, S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
10. S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Trans. Syst. Man Cyber.*, 6:325–327, 1976.

11. E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report TR4, USAF School of Aviation Medicine, Randolph Field, TX, 1951.
12. T. Poggio, S. Smale. The Mathematics of Learning: Dealing with Data. *Notices of the AMS* 50(5):537-544, 2003.
13. R. Quinlan. Rulequest research data mining tools. `http://www.rulequest.com/`.
14. G. W. Snedecor, W. G. Cochran. Statisitical Methods, Iowa State University Press, Ames, IA, 2002, eighth edition.
15. C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communication of ACM*, 29:1213–1229, 1986.
16. P. Stone. *Layered Learning in Multi-agent Systems: A Winning Approach to Robotic Soccer.* MIT Press, Cambridge, MA, 2000.
17. V. N. Vapnik. *Statistical learning theory.* Wiley New York, 1998.
18. 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.
19. H. Wang, I. Düntsch, D. Bell. Data reduction based on hyper relations. *Proceedings of KDD98, New York*, 349–353, 1998.
20. H. Wang, I. Düntsch, G. Gediga. Classificatory filtering in decision systems. *International Journal of Approximate Reasoning*, 23:111–136, 2000.
21. H. Wang, I. Düntsch, G. Gediga, A. Skowron. Hyperrelations in version space. *International Journal of Approximate Reasoning*, 36(3):223–241, 2004.
22. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.
23. RSES: Rough set exploration system. http://logic.mimuw.edu.pl/˜rses, Institute of Mathematics, Warsaw University, Poland.
24. R. Quinlan. Improved Use of Continuous Attributes in C4.5. Journal of Artificial Intelligence Research, 4:77-90, 1996.