s c c h
software competence center
hagenberg

Technical Report
SCCH–TR–0016

Interpretation of Self-Organizing Maps with Fuzzy Rules

Mario Drobics, Werner Winiwarter, Ulrich Bodenhofer
mario.drobics@scch.at
werner.winiwarter@scch.at
ulrich.bodenhofer@scch.at

Abstract — Exploration of large and high-dimensional data sets is one of the main problems in data analysis. Self-organizing maps (SOMs) can be used to map large data sets to a simpler, usually two-dimensional, topological structure. This mapping is able to illustrate dependencies in the data in a very intuitive manner and allows fast location of clusters. However, because of the black-box design of neural networks, it is difficult to get qualitative descriptions of the data. In our approach, we identify regions of interest in SOMs by using unsupervised clustering methods. Then we apply inductive learning methods to find fuzzy descriptions of these clusters. Through the combination of these methods, it is possible to use supervised machine learning methods to find simple and accurate linguistic descriptions of previously unknown clusters in the data.

Key words — *Data Mining, Machine Learning, Fuzzy Logic, Neural Networks*

ICTAI 2000 - The Twelfth IEEE International Conference on Tools with Artificial Intelligence, Vancouver, BC

Kplus
Kompetenzzentren-Programm

1 Introduction

With the rapid increase of the amount of stored information, the need to extract important knowledge from this data arises. Data about production processes, customers, and other topics are collected in large high-dimensional databases. These databases store detailed information about many past events. But to be able to predict future events, one has to find global trends and coherences.

Large data sets cannot be analyzed directly. On the one hand, the data may contain missing values and outliers, while, on the other hand, large datasets (with one million records or more) do not allow interactive data exploration. One possibility to reduce the amount of raw information and to find general structures are self-organizing maps (SOMs) [11]. Using SOMs to create a map of the data space reduces the number of sample nodes to a manageable amount, while additionally "smoothing" the input data. This means that outliers have only minor influence and eventually uncovered regions (holes) in the input space are covered by "interpolated" nodes.

Once a SOM is trained with sample data, it is possible to analyse data visually using different plots like contour plots or U-matrix plots. While these plots are very intuitive for analyzing dependencies, they lack a mathematical or linguistic description of the data. Several publications deal with the problem of analyzing SOMs (e.g. [7, 9]), but none of them is able to create qualitative descriptions of the data.

In this paper, we present a combination of self-organizing maps, clustering methods, and classification rule mining to find essential information, where fuzzy sets which are associated with natural language terms (e.g. *low* and *very high*) are used to partition the input space. Then more complex descriptions are built to describe the clusters. As the user should be able to interact with the system, the performance was one of the main design issues.

The remaining paper is organized as follows. First, we give a short introduction to self-organizing maps, the used clustering methods, and fuzzy set theory (Sect. 2). Then we present a brief review of related work in Sect. 3. Next we describe in Sect. 4, how descriptions are generated. Finally, example applications can be found in Sect. 5.

2 Preliminaries

2.1 Self-organizing maps

Kohonen introduced Self-Organizing Maps (SOM) in 1982 [11]. His idea was to create a neural network to represent the input space using the topological structure of a grid to store neighborhood relations. In contrast to most neural network methods that use the desired result to compute the weights of the network, SOMs need no reference output to tune their parameters, but are able to organize themselves (unsupervised learning).

An SOM defines a mapping of an n-dimensional input space $\mathcal{R}$ to an m-dimensional output space $\mathcal{C}$ (we used $m = 2$). The output space consists of $N_{\mathcal{C}}$ neurons. They are embedded in the topological structure of $\mathcal{C}$, which may be a m-dimensional grid or any other graph structure. To each neuron $s \in \mathcal{C}$ of the output space, a parametric *weight vector* in the input space $w_s = [\mu_{s1}, \mu_{s2}, \ldots, \mu_{sn}]^T \in \mathcal{R}$ is associated.

The mapping

$$\begin{aligned}\phi_w : \mathcal{R} &\to \mathcal{C} \\ x &\mapsto \phi_w(x)\end{aligned}$$

defines the *neural map* from the input space $\mathcal{R}$ to the topological structure $\mathcal{C}$, where $\phi_w(x)$ is defined as

$$\phi_w(x) = \text{argmin}_i \{\|x - w_i\|\}.$$

This means that every input sample is mapped to that neuron of the output layer whose weight vector is closest to the input (see Fig. 1).

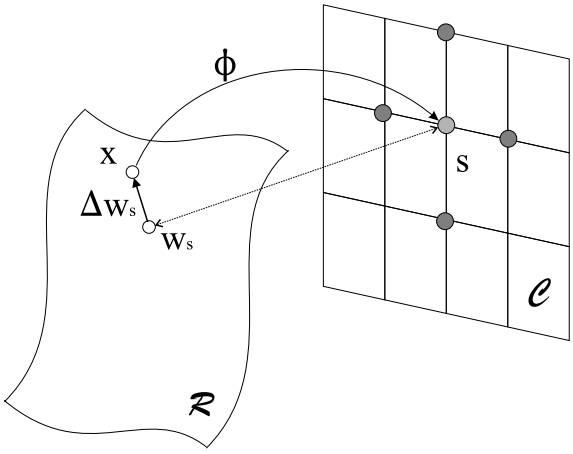

Figure 1: Mapping defined by a SOM

The goal of the learning process is to reduce the overall *distortion error*

$$d(x) = \sum_{x \in I} \sum_{i \in \mathcal{C}} h_{ci} \|x - w_i\| \frac{1}{|I|},$$

where I denotes the set of input samples and h_{ci} denotes the neighborhood relation in $\mathcal{C}$ between neuron i and the best matching one c. For details about the learning algorithm see [12].

2.2 Clustering

Though the number of data vectors is heavily reduced through the mapping defined by an SOM, it is necessary to use a large number of nodes to gain emergence (i.e. appearence of global patterns through the interaction of the neurons) and a certain degree of detail. To reduce the number of interesting regions even further (which is unavoidable when seeking global trends), we used two clustering methods alternatively: *Single-linkage clustering* and *modified Ward clustering* [2].

Single Linkage Clustering: The single linkage clustering corresponds to a kind of nearest neighbor clustering. The clusters are created by connecting all nodes whose distance is lower than a predefined boundary.

Ward Clustering: Ward clustering belongs to the group of agglomerative clustering methods. Starting with each node in a separate cluster, in each step, the two clusters that are closest according to a distance measure are merged and a new center of gravity is computed for this cluster. In our approach, a modified Ward clustering method is used that also takes the neighborhood relation into account, so that only adjoining clusters are merged (SOM Ward clustering).

By varying the number of clusters (the default value is computed using a heuristic *cluster indicator*), the required degree of detail can be defined. In our experience, large datasets with about one million records required SOMs with 4000 nodes and resulted in 5–15 clusters.

Each of these clusters represents a region of the input data that contains somehow similar samples. In the past, the knowledge and experience of a data analyst were necessary to interpret these results. Our goal was to open up the black-box SOM and to create understandable interpretations of the clusters.

2.3 Fuzzy sets and inference

As class borders are often imprecise, dealing with "crisp" descriptions can produce undesired and instable results. To prevent this, we created fuzzy rules to describe the clusters. In this section, we will first describe what fuzzy sets are, and how they are used to create fuzzy predicates.

Fuzzy sets were introduced by L. A. Zadeh in 1965 [22]. In classical set theory, a set S is associated with its characteristic function, which is defined as 0 if $x \notin S$ and 1 if $x \in S$. Fuzzy set theory extends the concept of characteristic functions. The membership function $\mu_S(x)$ of a fuzzy set S defines the degree of membership to which an element x belongs to S. This degree may be any value from the unit interval $[0, 1]$, where 0 indicates that x is definitely not in S, and a value of 1 that x is part of S. For example, if we define the membership function of the fuzzy set of *tall* people to yield 0 for heights smaller than 170cm, 1 for heights greater than 185cm, and linearly increasing in between, a person with 180cm, would be *tall* with a degree of 0.67.

The set of all possible predicates that describe a variable is (especially if numerical data is used) rather big or even infinite. It is, therefore, necessary to reduce the number of descriptions that are analyzed to a manageable amount.

We decided to use a constant number of sets for each input parameter. By default, $n = 7$ sets are used. Depending on the underlying context of the variable under consideration, we associate natural language expressions like *very very low*, *medium*, *large*, etc. with each set. To find a segmentation of the input space according to the distribution of the sample data, the centers of the sets are first initialized by dividing the input range into n equally sized intervals. Then these centers are iteratively updated by using a modified *k-means* algorithm [13] with simple neighborhood interaction. Usually a few iterations (≤ 10) are sufficient. The fuzzy sets are then created as trapezoid membership functions (see Fig. 2) around the previously computed centers.

A fuzzy set S induces a fuzzy predicate

$$t(x \text{ is } S) = \mu_S(x),$$

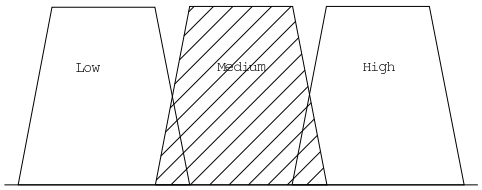

Figure 2: Membership functions for three fuzzy sets

where t is the function which assigns a truth value to the assertion x is S. In a straightforward way, a fuzzy set S can also be used to define the following fuzzy predicates [4, 5]:

$$\begin{aligned} t(x \text{ is not } S) &= 1 - \mu_S(x) \\ t(x \text{ is at least } S) &= \sup\{\mu_S(y) \mid y \geq x\} \\ t(x \text{ is at most } S) &= \sup\{\mu_S(y) \mid y \leq x\} \end{aligned}$$

Logical operations on the truth values are defined using t-norms and t-conorms, i.e. commutative, associative, and non-decreasing binary operations on the unit interval with neutral elements 1 and 0, respectively [20, 10]. These operations can be defined in various ways; we restrict ourselves to the well-known examples *min* and *max*:

$$\begin{aligned} T_{\mathrm{M}}(x, y) &= \min(x, y) \\ S_{\mathrm{M}}(x, y) &= \max(x, y). \end{aligned}$$

Then conjunctions and disjunctions of fuzzy predicates can be defined as follows:

$$\begin{aligned} t(A(x) \wedge B(x)) &= T\big(t(A(x)), t(B(x))\big) \\ t(A(x) \vee B(x)) &= S\big(t(A(x)), t(B(x))\big) \end{aligned}$$

Since we want to predict the class membership of the samples, we are interested in fuzzy rules of the form `IF` $A(x)$ `THEN` $C(x)$ (for convenience, we refer to such a rule as $A(x) \rightarrow C(x)$, although the rule should rather be regarded as a conditional assignment than as a logical implication) where $C(x)$ is a fuzzy predicate that descibes the class membership and $A(x)$ may be a compound fuzzy predicate composed—by means of conjunction—of several simple predicates from above, e.g.

$$\begin{aligned}&\texttt{IF } (\text{age}(x) \text{ is high}) \wedge (\text{size}(x) \text{ is at least high})\\ &\texttt{THEN } (\text{class}(x) \text{ is } 3).\end{aligned}$$

The degree of fulfillment of such a statement for a given sample x is defined as $t(A(x) \wedge C(x))$ (corresponding to so-called Mamdani inference [14, 23]). The fuzzy set of samples fulfilling a predicate A, which we denote with $A(I)$, is defined as (for $x \in I$)

$$\mu_{A(I)}(x) = t(A(x)).$$

Let us define the *cardinality* of a fuzzy set S of input samples as the sum of $\mu_S(x)$, i.e.

$$|S| = \sum_{x \in I} \mu_S(x).$$

Finally, the cardinality of samples fulfilling a rule $A \rightarrow C$ can be defined by

$$\begin{aligned}|A(I) \cap C(I)| &= \sum_{x \in I} t\big(A(x) \wedge C(x)\big)\\ &= \sum_{x \in I} T\big(t(A(x)), t(C(x))\big).\end{aligned}$$

3 Related work

SOMs are very popular for data analysis, but still only few methods for analyzing SOMs have been developed in the past. We will describe the two most promissing ones, the $sig*$ and the *LabelSOM*.

U-Matrix and *sig*:* Ultsch [21] developed the *U-Matrix* method to analyze the map and the *sig** algorithm to generate descriptions of clusters in the map. Ultsch uses an SOM with many neurons to allow emergence. Then he analyzes the map using his *U-Matrix* method. The *U-Matrix* allows to explore the neighborhood relations of a two-dimensional map through a three-dimensional plot. By protracting the average distance of each node to its neighbors as its height value, regions of similarity can be identified easily.

Finally the $sig*$ algorithm is used to find significant attributes for each class. This algorithm tries to identify the most characteristic rules for each cluster by means of statistical measures. If these descriptions are not accurate enough, additional differentiating rules are computed.

LabelSOM: A completely different approach was chosen by Rauber who developed the *LabelSOM* method to "label" the nodes of the SOM [19]. Unlike Ultsch, he uses smaller maps, where each node describes a larger region of the input space.

The *LabelSOM* algorithm uses the *quantization error* q_i (i.e. the error of the projection) for each node i and dimension k to determine the most important parameters:

$$q_{i_k} = \sum_{x_j \in \mathcal{R}_i} \sqrt{(w_{i_k} - x_{j_k})^2}, \qquad k = 1 \ldots n$$

Though interesting results on real-world problems have been presented [15], it is only possible to find significant parameters, but not descriptions of arbitrary clusters, as needed for more complex data sets.

4 Rule Generation

SOMs are usually analyzed using graphical tools and cluster analysis. The main task of our research is to create intuitive descriptions of the clusters found and to enable the user to identify significant parameters easily. The obtained rules are evaluated using the original data afterwards.

We first experimented with the simple decision tree model ID3 [17] to describe the goal cluster, but the results were not very satisfying, as ID3 tries to describe the whole input space which leads to long and not very expressive descriptions. To create a more suitable set of rules using Inductive Logic Programming (ILP) [6, 16], we developed an algorithm called *FS-FOIL*, that extends the original *FOIL* algorithm [18]. While *FOIL* was developed to find Horn clauses, we modified it to be able to handle first-order fuzzy predicates. This enables us to describe fuzzy class borders and to create more robust rule sets, as small variances in the input data do not lead to a completely different classification.

Though this allows us to describe each part of a cluster through a single rule in a very intuitive way, it is not possible to find more than one description for the same region of the cluster, which can lead to suboptimal results. Therefore, we developed another method called *FF-MINER* based on the *DENSE-MINER* algorithm by Bayardo, Agrawal and Gunopulos [8].

Rules of interest have to be significant and accurate. The *significance* of a rule $A \rightarrow C$ can be described by its *support*

$$\mathrm{supp}(A \rightarrow C) = \frac{|A(I) \cap C(I)|}{|I|},$$

the *accuracy* as its *confidence*

$$\mathrm{conf}(A \rightarrow C) = \frac{\mathrm{supp}(A \rightarrow C)}{\mathrm{supp(A)}},$$

where $\mathrm{supp}(A)$ is defined as

$$\mathrm{supp}(A) = \frac{|A(I)|}{|I|}.$$

In both algorithms, only rules are considered which satisfy reasonable requirements in terms of support and confidence. This is realized by two lower bounds $\mathrm{supp}_{\min}$ and $\mathrm{conf}_{\min}$, respectively [1].

4.1 FS-FOIL

The basic concept of *FS-FOIL* (fuzzy set *FOIL*) is to find a representation of the goal cluster by stepwise coverage. Each coverage is found by means of beam search. Starting with an empty rule, all previously computed rules are expanded. Then the new set of rules is pruned to reduce computational effort. If a rule satisfies minimum support and confidence, it is added to the list of final descriptions and the nodes covered by this rule are removed from the open list, using fuzzy operations. The basic algorithm is presented in Fig. 3.

```
open_list = goal_cluster;
results = ∅;
// start with the most general rule
testRules = {∅};
// till all samples are covered
do {
    resultsTmp = testRules;
    testRules = ∅;
    // expand all previous rules
    for each curRule ∈ resultsTmp
    {
        tmpRules = expand(curRule);
        testRules += tmpRules;
    }
    // accurate & significant?
    for each candidate ∈ testRules
        if (candidate.conf > conf_min) ∧
           (candidate.sup > supp_min)
        {
            // add rule to result set
            results += candidate;
            // remove covered nodes from open nodes
            open_list = open_list - candidate.matchedNodes;
            // go on with the most general rule
            testRules = {∅};
            exit for;
        }
    pruneRules(testRules);
} while |open_list| > MAXERROR;
return results;
```

Figure 3: FS-FOIL

The expansion of a rule (`expand()`) is performed by appending each new simple fuzzy predicate to the rule. This means that, starting with the most general rule (the empty one), more and more specific rules are created. Pruning is necessary to reduce computational effort. This is done in two steps. First, all rules with support lower than $\mathrm{supp}_{\min}$ are removed. In a second step, the rules are sorted according to the *FOIL* measure (an entropy measure that prefers larger sets). The

FOIL measure of a rule $A \rightarrow C$ over the set of samples I is defined as:

$$G \;=\; |A \cap C| \;\left(\log_2 \frac{|A \cap C|}{|A|} - \log_2 \frac{|C|}{|I|} \right).$$

As the most promising rules for further expansion have usually higher values G, only a certain number of rules are kept. This allows to reduce the search in the solution space, while keeping a minimum breadth.

4.2 FF-MINER

To gain more insight information about the goal cluster, we developed *FF-MINER* (Fast Fuzzy Miner) as a fuzzification of the *DENSE-MINER* algorithm, which aims at finding more than one accurate and significant description of the goal cluster. As such a set of all rules is still very large and redundant, a partial order with respect to the two criteria support and confidence is defined, where only the maximal rules with respect to this partial order are considered [3]. Finding all maximal elements is quite difficult, as the search space has to be pruned without losing significant results. We decided to skip more rules to increase performance, because the goal is not to mine all maximal rules, but to get a good overview of potential descriptions in an acceptable time. How exact pruning can be done is described in detail in [8]. By removing all correctly classified samples from the open list and applying the algorithm to the remaining samples like in *FS-FOIL*, sometimes significantly better results can be obtained than with the *FS-FOIL* algorithm. However, computational effort is much higher.

The algorithm to compute one level of descriptions is described in Figure 4. Finding all maximal (`findMax()`) rules in the set of all available rules is done by removing all rules which are not maximal with respect to the partial ordering.

5 Samples

5.1 Two rings

This data set contains the three-dimensional coordinates of two rings plus the class information. This sample was also used by Ultsch in several of his publications (e.g. [21]) and is mentioned here to allow comparisons. One ring is parallel to the xy plane, the other one parallel to the xz plane (see Fig. 5).

The data consisting of 20001 randomly generated points were presented to a network with 2041 nodes. The network was trained with the class information to show how predefined patterns can be learned. Identifying the goal classes in an unsupervised approach is also possible, but the artifical structure of the data necessitated 14 clusters or manual cluster selection using graphical plots (as done by Ultsch).

Afterwards, we applied the *FS-FOIL* algorithm to find descriptions of the clusters. In this example, we decided to use only three fuzzy sets for each variable (NEGATIVE, ZERO, POSITIVE). The fuzzy set borders were computed by the method described in Sect. 2.3.

The result of the *FS-FOIL* algorithm classifying the ring parallel to the xy plane is as follows:

```
results = ∅;
// start with the most general rule
testRules = {∅};
// till no more rules are open
do {
    resultsTmp = testRules;
    testRules = ∅;
    // expand all previously computed rules
    for each curRule ∈ resultsTmp
    {
        tmpRules = expand(curRule);
        // store promising candidates
        for each candidate ∈ tmpRules
            if (candidate.conf > conf_min) ∧
              (candidate.sup > curRule.sup * 1.01)
            {
                results += candidate;
                testRules += candidate;
            }
    }
    // pruning reduces computational affort
    pruneMax(testRules);
} while testRules ≠ ∅;
// remove non maximal rules
findMax(rules);
return results;
```

Figure 4: FF-MINER

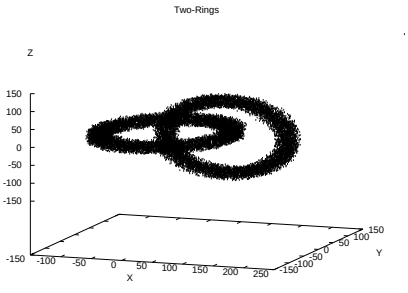

Figure 5: The two-ring example

```
(conf% supp%    pos    neg   pos%   neg%)
rule

( 1.00, 0.38,  7535,     0, 75.35,  0.00)
[y != zer]
```

```
( 0.89, 0.06,  1131,   142, 11.31,  1.42)
[x == neg] && [y == zer]

( 0.24, 0.06,  1189,  3719, 11.89, 37.18)
[x == pos] && [z == zer]
```

We see that the first rule matches all samples with a y value different to 0. Its confidence is 100% and it matches 38% of all samples and 75% of the samples of this class respectively. It was found by comparing all rules of length 1. As its confidence and support exceed the required lower bounds, it is accepted and those nodes which match are removed from the list of open nodes.

To find the second rule, all rules of length 1 are compared again. As no rule matches the remaining samples well enough, the best five ones (according to the *FOIL* measure) are expanded to find rules of length 2. Finally, the second rule, which has 89% confidence, was found by this procedure. Together with the first one, 86% of the samples of the goal class are covered.

Though the last rule has only 24% confidence, the result is quite good, as no better prediction is possible with 3 sets per dimension.

The main benefits of our approach is that the obtained rules are very close to a description a human would give and that the results are obtained very quickly, if the SOM has already been computed.

5.2 Economic data

This data set consists of macro-economic data about the USA recorded monthly between 1963 and 1985. The goal of the analysis was to find (unknown) regions of interest in the data and to label them.

First, a SOM with 344 nodes was trained with the data, then 8 clusters were identified using the modified SOM Ward clustering (see Fig. 6).

These clusters were analyzed using the *FS-FOIL* algorithm. Describing each of the clusters required about 3 rules with a confidence above 80 percent. Applying *FF-MINER* to the clusters showed that for two clusters better rules exist, which were not found by *FS-FOIL*. Using these more accurate rules, the total minimum confidence was lifted above 90 percent for 90 percent of the data.

As an example, we will take a look at the results for the 2th cluster as returned from *FS-FOIL*:

```
Goal Test:  Cluster = is_2

(conf%  sup%    pos    neg   pos%   neg%)

( 0.97, 0.11,    36,     1, 49.51,  0.37)
[Treas.Bond_(30Y) == M] && [CPI_(rel.chg) <= M]

( 0.90, 0.05,    17,     2, 22.71,  0.65)
[Prime_Rate == VL] && [Unemployment == M]

( 0.83, 0.02,     8,     2, 10.46,  0.57)
[Treas.Bond_(30Y) == L] && [S&P_P/E_ratio == L]
```

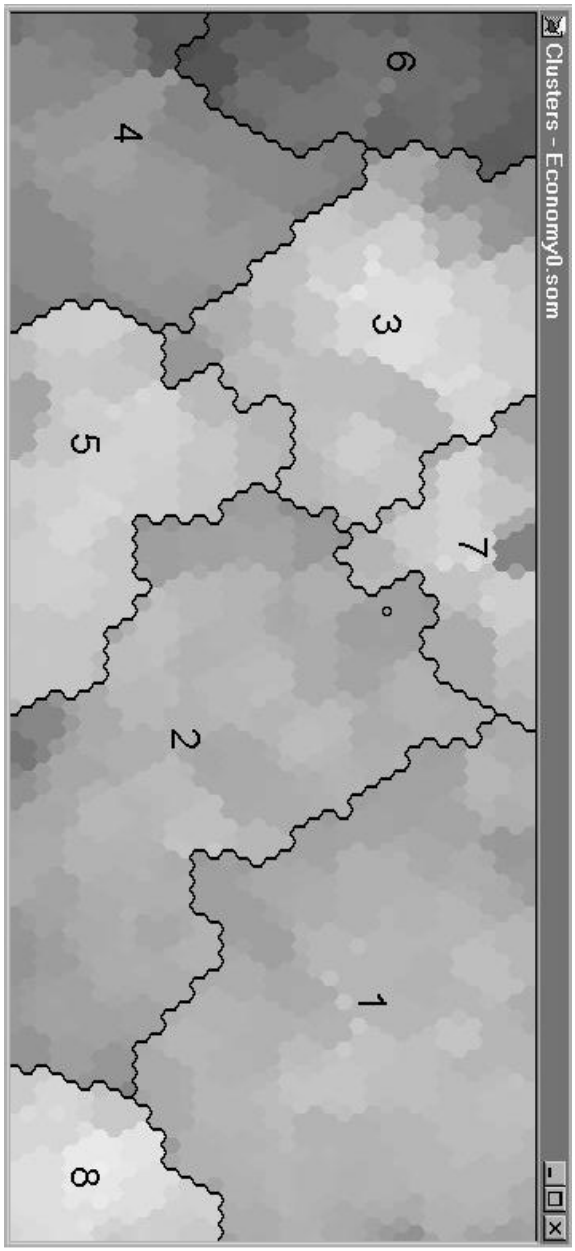

Figure 6: Clusters in the economic data set

```
( 0.75, 0.02,     7,     2,  9.79,  0.90)
[S&P_500_/_CPI >= L] && [Prime_Rate == VL] &&
[Treas.Bond_(30Y) != M] && [S&P_P/E_ratio != M] &&
[Gold_Price_/_CPI != VL] && [CPI_(rel.chg) <= M] &&
[Unemployment >= VL]
```

As we can see, 92.47% of the class is described by the four rules returned. The confidence of each rule is above 75%. It is interesting to find the parameter `Treas.Bond (30Y)` in three

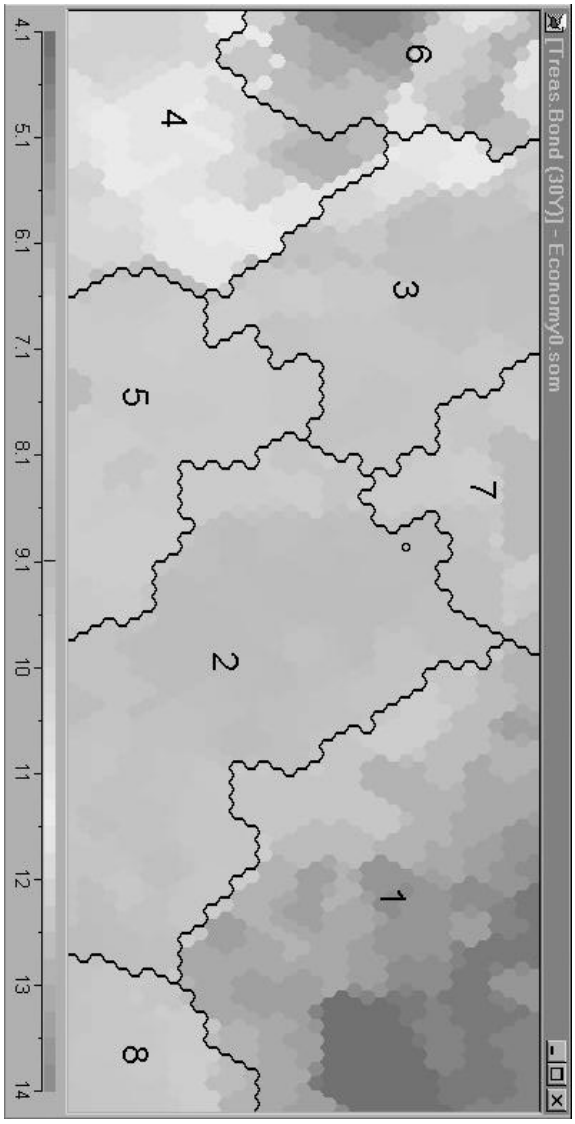

Figure 7: The parameter Treas.bond (30Y)

rules but with different values. In the first rule, all samples with *medium* `Treas.Bond (30Y)` are matched. These nodes build the main part of the cluster (49.51%). The third one, covers all nodes with *low* `Treas.Bond (30Y)`, while the last one (with the lowest support), covers only those with a `Treas.Bond (30Y)` that is not *medium*. This indicates that the `Treas.Bond` parameter is very characteristic for this cluster, but some other parameters have to be considered, too. If we take a look at Fig. 7 we see a contour plot for the `Treas.Bond (30Y)` parameter that illustrates these results. We see, that the clusters 3, 5, 7, and 8 also share a medium `Treas.Bond (30Y)`.

Applying *FF-Miner* for the second cluster shows, that more accurate descriptions can be obtained without the `Treas.Bond (30Y)` parameter.

```
Goal Test:  Cluster = is_2
```

```
(conf%  sup%    pos    neg   pos%   neg%)

( 1.00, 0.13,    44,     0, 60.52,  0.00)
[S&P_500_/_CPI >= L] && [S&P_P/E_ratio != VVL] &&
[Gold_Price_/_CPI >= L] && [CPI_(rel.chg) <= M]
&& [Unemployment != M]

( 0.97, 0.18,    60,     2, 82.70,  0.70)
[S&P_500_/_CPI >= L] && [S&P_P/E_ratio != VVL] &&
[Gold_Price_/_CPI >= L] && [CPI_(rel.chg) <= M]
```

These rules are more accurate and significant than those generated with *FS-FOIL*, but also more complex than the previous ones. Depending on the problem as well as on the obtained results which quality criteria to favor, we can choose the more accurate (but less significant) first rule, or the more general second one. This freedom of choice is one of the main benefits of *FF-Miner*.

6 Conclusion

In this paper, we have presented a new approach to analyze large datasets by identifying and characterizing interesting clusters. We use a combination of self-organizing maps, clustering methods, and rule induction. It was described how fuzzy rules and the underlying fuzzy sets are created to find linguistic predicates, and how knowledge can be extracted from neural networks. The main benefit of our approach is that it is able to combine the flexibility of unsupervised learning with the accuracy and efficiency of supervised learning methods. These methods can be applied to data sets from business and finance as well as to industrial process data.

Future research will focus on using fuzzy clustering methods and on the extraction of knowledge about specific parameters of the data.

7 Acknowledgements

This work has been done in the framework of the *Kplus Competence Center Program* which is funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

In particular, the authors would like to thank *eudaptics software gmbh* for providing their commercial SOM tool *Viscovery SOMine* which has been used extensively to create the SOMs and the clusters for the examples presented in this paper.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 1993.

[2] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[3] R. Bayardo and R. Agrawal. Mining the most interesting rules. In S. Chaudhuri and D. Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, N.Y., 1999. ACM Press.

[4] U. Bodenhofer. The construction of ordering-based modifiers. In G. Brewka, R. Der, S. Gottwald, and A. Schierwagen, editors, *Fuzzy-Neuro Systems '99*, pages 55–62. Leipziger Universitätsverlag, 1999.

[5] U. Bodenhofer. *A Similarity-Based Generalization of Fuzzy Orderings*, volume C 26 of *Schriftenreihe der Johannes-Kepler-Universität Linz*. Universitätsverlag Rudolf Trauner, 1999.

[6] I. Bratko and S. Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, 1995.

[7] G. Deboeck and T. K. (Eds). *Visual Explorations in Finance with Self-Organizing Maps*. Springer-Verlag, London, 1998.

[8] R. J. B. Jr., R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*, pages 188–197. IEEE Computer Society, 1999.

[9] S. Kaski and T. Kohonen. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In A.-P. N. Refenes, Y. Abu-Mostafa, J. Moody, and A. Weigend, editors, *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets, London, England, 11–13 October, 1995*, pages 498–507. World Scientific, Singapore, 1996.

[10] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publishers, Dordrecht, 2000. (to appear).

[11] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[12] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.

[13] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, 1980.

[14] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis of fuzzy controllers. *Int. J. Man-Mach. Stud.*, 7:1–13, 1975.

[15] D. Merkl and A. Rauber. Automatic labeling of self-organizing maps for information retrieval. In *Proceedings of the 6th Intl. Conf. on Neural Information Processing (ICONIP'99)*, 1999.

[16] S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19 & 20:629–680, 1994.

[17] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[18] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[19] A. Rauber. LabelSOM: On the labeling of self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, 1999.

[20] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. North-Holland, Amsterdam, 1983.

[21] A. Ultsch. Knowledge acquisition with self-organizing neural networks. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 208–210, Vienna, Austria, 1992. John Wiley & Sons.

[22] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[23] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst. Man Cybern.*, 3(1):28–44, 1973.