Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)

Basic Technologies for Knowledge Transfer in Intelligent Systems

Oliver Buchtala and Bernhard Sick
Faculty of Computer Science and Mathematics, University of Passau, Germany
e-mail: {buchtala, sick}@fmi.uni-passau.de

Abstract— **Knowledge transfer is one of the most important mechanisms of human evolution. The ontogeny of humans enables them to act efficiently in a very dynamic environment. Thus, it would be highly desirable to enable "intelligent" artificial systems to behave in a similar way. This article introduces basic technologies that are needed for that purpose. With these technologies – components of a future knowledge transfer toolbox – it is possible to detect novel concepts that arise in the input space of a classifier or existing classification rules that become obsolete. Then, prototypes of new rules can be created automatically using an on-line clustering mechanism. These prototypes are compared to already existing rules, rated, and eventually accepted or discarded. In case of acceptance, a human expert labels the rules which are then both integrated into the "own" classifier and sent to other classifiers. Thus, knowledge transfer between "intelligent" artificial systems becomes possible and the overall system is provided with a new kind of self-optimization ability.**

I. Introduction

Animals or humans interact in various ways:

- Individuals independently obey certain rules such that a whole swarm of individuals exhibits a certain behavior (e.g., movement of bird flocks or fish schools).
- Individuals exchange information in simple or sometimes even complex ways in order to achieve a common goal (e.g., pheromone trails of ants or bee dance languages).
- Individuals learn from each other by observing each other (e.g., juvenile chimpanzees learning from adults to use simple tools).
- Individuals learn from each other by communicating rules (e.g., children that learn from their teachers in school).

All these interactions may lead to an emergent behavior of the overall group of individuals (cf. [1]).

The ambitious long-term objective of our work is to study the – possibly emergent – behavior of intelligent distributed systems – e.g., teams of robots, software agents, animats, or smart nodes of sensor networks – that exchange knowledge in form of rules. The advantage of such a knowledge transfer is obvious. Individuals may behave proactively: Before certain situations come up in their local environment, they will already be enabled to handle them. From a technical viewpoint there are two other important benefits: First, techniques and ontologies needed for knowledge transfer are independent from a particular application domain. Second, the communication effort needed for knowledge transfer may be significantly lower than the effort needed for an exchange of information (e.g., resulting from observations) as knowledge is more abstract and often more valuable than information.

Knowledge transfer in artificial, intelligent systems is a new idea and this article introduces some basic technologies that are needed for its realization. The article makes a first, important step, but most of the components set out here will be further improved in the future. Section II describes the components of a knowledge transfer toolbox we realized up to now, Section III provides a few experimental results, and Section IV summarizes the major findings and gives an outlook to our future work.

II. Components of a Knowledge Transfer Toolbox

In this section we first provide an overview of the basic components of a knowledge transfer toolbox and then describe the components we realized up to now in some more detail. In our work, we are focusing on classification problems. That is, the rules we want to transfer assign a certain area of the input space of a classifier to a certain class. Depending on the result of this classification, appropriate reactions may be initiated.

A. Overview

In our current implementation of a knowledge transfer toolbox, the following components are realized:

- **Active Classifier:** The classifier paradigm is a modified radial basis function (RBF) neural network which also can be seen as a fuzzy system (FS).
- **Novelty Detector:** The novelty detector notices missing knowledge (rules) in the active classifier and causes further actions.
- **Prototype Generator:** This component uses a sliding window of recent data points to create new rule prototypes (premises).
- **Prototype & Rule Assessor:** This component provides measures that are needed to rate rule prototypes and rules.
- **Observed Classifier:** An observed classifier (there may be several instances of this component) consists of rules that are currently not actively used but observed.
- **Rule Integrator & Communicator:** With this component, rules are sent to other individuals and fused with already existing rules to build a new classifier.

These different components work – at least partly – in an asynchronous way. Their cooperation can be described as follows: An active classifier processes the incoming data of an intelligent system (in the following: individual) independently from the other components. This classifier can be trained from data and consists of interpretable rules. The novelty detector

observes the active classifier and tries to detect novel *concepts* (clusters) within the data. To achieve this goal, a specific property of our classifier paradigm is exploited. In the case novelty is detected, the novelty detector emits an order to the prototype generator which utilizes an on-line clustering mechanism to determine new clusters (concepts) within the currently observed data. Prototypes can be seen as *premises* of candidate rules that are potentially useful. New prototypes, active rules and rules received from other individuals are combined in an observed classifier to undergo further analysis. Various measures can be utilized for that purpose. After a successful evaluation, a rule prototype must be labeled by a human expert. Then, the rule (*novel knowledge*) is integrated into the active classifier and sent to other individuals by means of a simple broadcasting mechanism. The active rules are also copied into another observed classifier. This observed classifier is utilized to detect rules that are no longer necessary (*obsolete knowledge*) by means of very similar rating mechanisms.

The process of finding new rules is decoupled from the active classification by introducing observed classifiers. Several measures can be applied to the observed classifiers in order to predict the impact of knowledge modifications on the active classifier. It should also be emphasized that the human expert (who is simulated in our experiments up to now) is involved in a very efficient way: He or she does not label a huge number of data points but only a few new rule prototypes.

It has already been mentioned that a comparable idea of knowledge transfer does not exist. Our work has been inspired by various research areas: An appropriate machine learning paradigm, for example, could be found in the area of *Soft Computing* [2]. There is a few work on knowledge extraction in this field, too (see [3] for some references). Clustering methods could be adopted from the field of *Pattern Recognition* [4]. The area of *Data Mining* [5] aims at measuring the interestingness of knowledge (validity, novelty, interpretability, etc.) [6] and provided some ideas for novelty detection. However, new solutions had to be found for most of the components.

B. Active Classifier

We define the radial basis function classifier RBFFS (cf. [3], [7]) as a hybrid system that can be seen as both, an RBF neural network (NN) and a Mamdani-type fuzzy system (FS). With the following definition we gain the advantages of two worlds: Trainability of NN and interpretability of FS (cf. [8]).

From the viewpoint of a NN, the RBFFS may be defined as follows (cf. Figure 1):

1) The RBFFS has three layers of neurons: input U_I, hidden U_H, and output layer U_O. Feed-forward connections exist between U_I and U_H as well as between U_H and U_O. A scalar weight ($w_{(i,j)}^{(I,H)}$ or $w_{(j,l)}^{(H,O)}$) is associated with each connection.
2) The activation of each hidden neuron $j \in U_H$ is determined using a multivariate Gaussian function:

$$a_j^{(H)}(k) \stackrel{\text{def}}{=} \frac{a'_j(k)}{\sum_{m=1}^{|U_H|} a'_m(k) + \max\{\varepsilon - \sum_{m=1}^{|U_H|} a'_m(k), 0\}}$$

with

$$a'_j(k) \stackrel{\text{def}}{=} e^{\left(-\sum_{i=1}^{|U_I|} \frac{(w_{(i,j)}^{(I,H)} - x_i(k))^2}{r_{(i,j)}^2}\right)}$$

and a user-defined parameter ε (typically with $\varepsilon = \frac{1}{e}$; e being the base of the natural logarithm), where $k = 1, 2, \ldots$ denotes the number of the pattern and $\mathbf{x}(k) \stackrel{\text{def}}{=} (x_1(k), \ldots, x_{|U_I|}(k))$ is the external input. The activation function is parameterized by the weight vector $\mathbf{w}_j^{(I,H)} \stackrel{\text{def}}{=} (w_{(1,j)}^{(I,H)}, \ldots, w_{(|U_I|,j)}^{(I,H)})$ and a parameter vector $\mathbf{r}_j \stackrel{\text{def}}{=} (r_{(1,j)}, \ldots, r_{(|U_I|,j)})$.
3) Each output neuron $l \in U_O$ computes its activation as a weighted sum:

$$a_l^{(O)}(k) \stackrel{\text{def}}{=} \sum_{j=1}^{|U_H|} w_{(j,l)}^{(H,O)} \cdot a_j^{(H)}(k).$$

The external output vector of the network, $\mathbf{y}(k) \stackrel{\text{def}}{=} (y_1(k), \ldots, y_{|U_O|}(k))$, consists of the activations of output neurons, i.e., $y_l(k) \stackrel{\text{def}}{=} a_l^{(O)}(k)$.

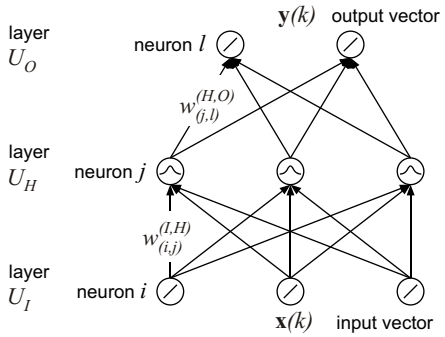

Fig. 1. Structure of a radial basis function neural network classifier.

Note that with an abbreviation for univariate Gaussians $a'_j(k) \stackrel{\text{def}}{=} \prod_{i=1}^{|U_I|} \varphi_{(i,j)}(k)$. In the following, the $\varphi_{(i,j)}$ are called basis functions; $w_{(i,j)}^{(I,H)}$ is the center of such a basis function and $r_{(i,j)}$ is its radius. The vectors $\mathbf{w}_j^{(I,H)}$ and $\mathbf{r}_j$ describe an axes-oriented hyperellipsiod in the input space of the RBFFS. Thus, $\mathbf{w}_j^{(I,H)}$ can be regarded as a center of a hyperellipsoidal cluster – big x in Figure 2 – and $\mathbf{r}_j$ defines the shape of the cluster – ellipses in Figure 2. The activation of a hidden neuron describes the similarity between an input pattern $\mathbf{x}(k)$ and a center based on a matrix norm (Mahalanobis distance measure with diagonal covariance matrix).

The parameters of an RBFFS must be determined by means of training algorithms such as gradient-based techniques or clustering techniques in combination with methods for the solution of linear least-squares (LLS) problems (see, e.g., [9] and our own work on RBF network training in [10]). For an iterative training step we use penalty terms (*regularization*

technique) to enforce small radii (*weight decay*) and to enforce normalized output weights in the interval $[0, 1]$.

For a classification problem, each class is typically assigned its own output neuron using an orthogonal representation of classes for training. A winner-takes-all approach is used for the final decision on class membership.

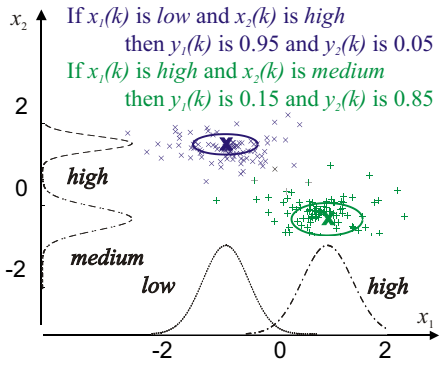

Fig. 2. Example of a classifier consisting of two rules operating in a two-dimensional input space ($|U_I| = 2$ and $|U_H| = 2$).

From the viewpoint of FS we can say that we have defined an FS with $|U_I|$ input variables, $|U_H|$ rules, and $|U_O|$ output variables (here: classes). The membership functions of the input variables correspond to the Gaussian basis functions of the hidden neurons, singletons are used for the output variables. That is, a fuzzy rule j ($j = 1, \ldots, |U_H|$) has the form

$$\text{if } x_1 \text{ is } \varphi_{(1,j)} \ \ldots \text{ and } x_{|U_I|} \text{ is } \varphi_{(|U_I|,j)}$$
$$\text{then } y_1 \text{ is } w^{(H,O)}_{(j,1)} \ \ldots \text{ and } y_{|U_O|} \text{ is } w^{(H,O)}_{(j,|U_O|)}.$$

The conjunction of variables in the premise of a rule as well as the implications are realized by the product operator. The sum operator is taken to combine the rules (i.e., we use sum-prod-inference). For defuzzification, the height method is applied.

The usage of rules with Gaussian premises is motivated by the *generalized central limit theorem*: Processes with multi-causal origination tend to be normally distributed.

C. Novelty Detector

The task of this component is to detect novel concepts within the incoming data and to decide whether new rule prototypes must be created. We defined the RBFFS paradigm in a way such that we can use it for novelty detection, too. The additional normalization term in the definition of a hidden neuron's activation indicates missing activation in the active classifier. The following measure defines the *recognition factor* $m_{\text{recognition}}$ for a time step k on a sliding data window of length l:

$$m_{\text{recognition}}(k) \stackrel{\text{def}}{=} \frac{1}{l} \sum_{k-l+1}^{k} 1 - \max\{\varepsilon - \sum_{m=1}^{|U_H|} a'_m(k), 0\}.$$

A novel concept is assumed to exist if the value of this measure sinks under a pre-defined threshold $\vartheta_{\text{novelty}}$. To avoid creating new prototypes while there exist prototypes that are already being evaluated, the decision is based on the novelty measure for an observed classifier (see below).

D. Prototype Generator

To determine rules in a dynamically changing environment, on-line mechanisms are necessary. Under the assumption that the measured data originates from Gaussian processes, this task can be solved by simple clustering algorithms. Conventional algorithms such as c-means have to be parametrized with the number of clusters [4]. Here we utilize a simple strategy that tends to produce too many prototypes. More appropriate methods will be applied in the future, e.g., techniques from information theory that determine an appropriate number of clusters automatically (cf. [11]).

The following algorithm is executed on demand using a window of the most recent l data points $X_l(k) \stackrel{\text{def}}{=} \{\mathbf{x}(j) | j \in k-l+1, \ldots, k\}$. Thus, it can be seen as an on-line clustering algorithm.

1) Randomly choose an initial set of centers $\mathcal{C}^{(0)}$ with uniform distribution and probability p from $X_l(k)$. Set $j := 0$. Set the set of barycenters $\mathcal{C} := \emptyset$.
2) Find the next barycenter for each initial center $\mathbf{c}_i^{(0)} \in \mathcal{C}^{(0)}$:
   a) For each $\mathbf{c}_i^{(j)} \in \mathcal{C}^{(j)}$ determine the set of k nearest neighbors kNN_i with an Euclidean distance measure. Then extend kNN_i: $\text{kNN}_i := \text{kNN}_i \cup \mathbf{c}_i^{(j)}$.
   b) Compute the mean $\tilde{\mathbf{c}}_i^{(j+1)} := \frac{1}{k+1} \sum_{\mathbf{x} \in \text{kNN}_i} \mathbf{x}$.
   c) For each $\tilde{\mathbf{c}}_i^{(j+1)}$ find the pattern $\tilde{\mathbf{x}}^{(j+1)} \in \text{kNN}_i$ with minimal Euclidean distance to $\tilde{\mathbf{c}}_i^{(j+1)}$.
   d) Set $\mathbf{c}_i^{(j+1)} := \tilde{\mathbf{x}}^{(j+1)}$.
   e) Add the centers $\mathbf{c}_i^{(j+1)}$ that did not change to the final set $\mathcal{C}$ and the remaining centers (that did change) to $\mathcal{C}^{(j+1)}$.
   f) If $\mathcal{C}^{(j+1)} \neq \emptyset$, set $j := j + 1$ and continue with Step 2a.
3) Remove redundant barycenters (barycenters that are included several times) from $\mathcal{C}$ and also the barycenters with a Mahalanobis distance greater than 1 to one of the existing prototypes.
4) Run a modified c-means clustering algorithm (cf. the algorithm we introduced in [10]) starting with the barycenters in $\mathcal{C}$, leaving the existing prototypes fixed (i.e., they are not adapted by the c-means algorithm).
5) Create rule prototypes with centers $\mathbf{w}_j^{(I,H)}$ (cluster means) and radii $\mathbf{r}_i$ (empirical standard deviations) resulting from that clustering.

The chance to find the actual cluster centers with Step 2 is very high. Of course, this technique is prone to produce suboptimal results, particularly in sparse data areas which are not close to actual cluster centers. However, corresponding prototypes can easily be detected due to the sparseness of the assigned data.

E. Prototype & Rule Assessor

In a dynamically changing environment the assessment of rules should have some dynamic behavior as well. That is, it only makes sense to integrate an offered rule when currently data is observed that could be classified by this rule. To achieve such a functionality, it is necessary to have a certain memory ability. We introduce the following mechanism which is inspired by Markov chain theory (see, e.g., [12]): An evaluated rule j is assigned a fitness value $f_j \in \mathbb{R}$. Measured data points cause a movement within this interval. Good evaluations increase the fitness, bad evaluations decrease it. Once the fitness value reaches one of the interval boundaries, the corresponding rule is either accepted ($f_j > 1$) or discarded ($f_j < 0$). By default, a rule is assumed to be discarded and, therefore, the fitness value generally must tend to sink. This is controlled by a parameter $\lambda_{\text{penalty}} > 0$. Good evaluations of rule measures can compensate the penalizing effect and even increase the fitness. This is controlled by a parameter $1 > \lambda_{\text{reward}} > 0$ (typically with $\lambda_{\text{reward}} > \lambda_{\text{penalty}}$). The evaluation of a rule j starting at time step k can now be done as follows:

1) Initially, set $f_j(k) := 0.5$.
2) For each observed pattern $\mathbf{x}(k+1)$:
   a) $f_j(k+1) := \lambda_{\text{reward}} \cdot f_j(k) - \lambda_{\text{penalty}}$.
   b) If $f_j(k+1) \geq 1$ accept the rule.
   c) If $f_j(k+1) \leq 0$ discard the rule.

Currently, we apply two kinds of measures to evaluate rules which are based on rule activation and premise dissimilarity. Rule activation is simply defined as the activation $a_j^{(H)}$ of the hidden neuron in the corresponding classifier. Premise dissimilarity is based on the following measure that determines the degree of overlapping of two univariate Gaussians:

$$overlap(\varphi_{(i,j)}, \varphi_{(i,k)}) \stackrel{\text{def}}{=} e^{-\left(\frac{w_{i,j}^{(I,H)} - w_{i,k}^{(I,H)}}{r_{(i,j)} + r_{(i,k)}}\right)^2}.$$

For axes-oriented multivariate Gaussians, this measure can simply be extended by computing the product over all dimensions. The degree of overlapping of two rule premises j and k can then be determined as follows:

$$overlap(j,k) \stackrel{\text{def}}{=} \prod_{i=1}^{|U_I|} overlap(\varphi_{(i,j)}, \varphi_{(i,k)}).$$

The dissimilarity is then simply computed by

$$dissim(j,k) \stackrel{\text{def}}{=} 1 - overlap(j,k).$$

Currently we provide two fitness measures: One uses only the dissimilarity of premises and the other uses the the minimum of the measures for rule activation and premise dissimilarity.

Let $\mathcal{A}$ be the set of all active rules l. The first fitness measure f_j^1 for a prototype j is independent from a time step k and defined by

$$f_j^1 \stackrel{\text{def}}{=} \min_{l \in \mathcal{A}} dissim(j,l).$$

The second fitness measure for a prototype j at a time step k is

$$f_j^2(k) \stackrel{\text{def}}{=} \min\left(\min_{l \in \mathcal{A}} dissim(j,l), a_j^{(H)}(k)\right).$$

F. Observed Classifier

In principle, observed classifiers consist of rules that are permanently rated using the mechanisms described, but not applied. One observed classifier is needed to detect novel knowledge, i.e., to assess rule prototypes and rules received from other individuals. Both fitness measures can be used for that purpose. To detect obsolete knowledge (needless rules), a second observed classifier is used into which the set of active rules is copied. As a fitness measure, the rule activation measure is applied. It should be emphasized that a more cautious behavior is desirable in this case. That means, the discarding of rules should be done carefully which can simply be achieved by choosing appropriate parameters λ_{reward} and λ_{penalty}. As there is no commit functionality needed, the fitness values of rules are bounded by the maximum value 1.

G. Rule Integrator & Communicator

The rule integration process is currently done by inserting rules without further actions. In the future, further improvements will be provided, e.g., techniques that keep the number of linguistic terms as low as possible. In [3] we describe how this can be done in principle. Also, there are no special methods for communication up to now; we simply broadcast the rules. In the future, we will develop an environment-awareness component that selects individuals that are known to be interested in certain functional knowledge or that are trusted, for instance.

III. Experimental Results

The experiment set out in this section demonstrates the feasibility and the advantages of the methods described in Section II. Three individuals measure and classify data. Individual 1 will be confronted with a new situation (i.e., novel data), it will acquire new functional knowledge (rules) and broadcast this knowledge. Individuals 2 and 3 will evaluate this knowledge applying two different strategies. Individual 2 will accept all rules that are sufficiently dissimilar to its own rule base. Individual 3 will discard rules that have no evidence in its recent data. After some time, when some of the temporarily observed concepts vanish in the environment of individual 1, it will declare respective rules as obsolete and remove them from the rule base. In a nutshell, the experiment shows how the individuals react autonomously on changes in their dynamic environment, how they optimize and evaluate their own rule bases, and how they can profit from exchanging rules with each other.

The data are generated utilizing Gaussian mixture distributions consisting of five different Gaussians $g_1, ..., g_5$ assigned to one of three classes. They have fixed parameters (centers, radii, class assignment), i.e., parameters do not change dynamically throughout the experiment. Dynamic effects are simulated by varying the probability p_i of each Gaussian

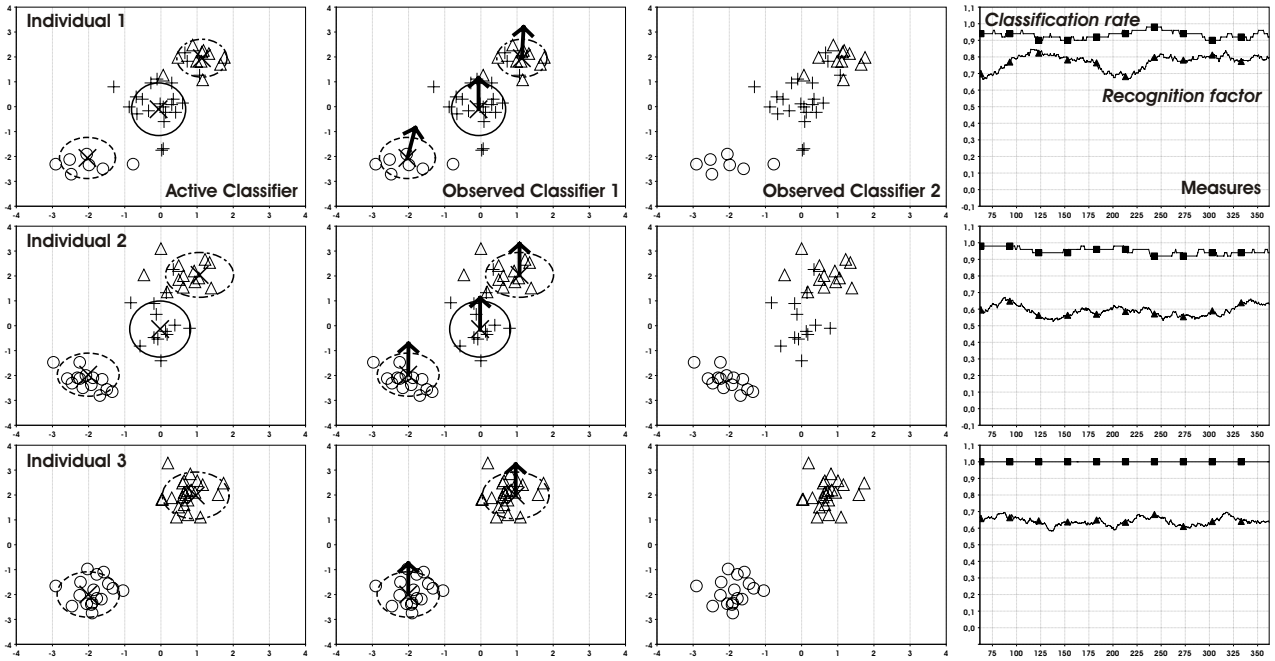

Fig. 3. Phase 1: The three individuals have been initialized with appropriate rules. They measure data that can be classified with a high classification rate (almost 100%). Correspondingly, the recognition factors are at an acceptable level, too. The active rules are rated quite high (observed classifier 1).

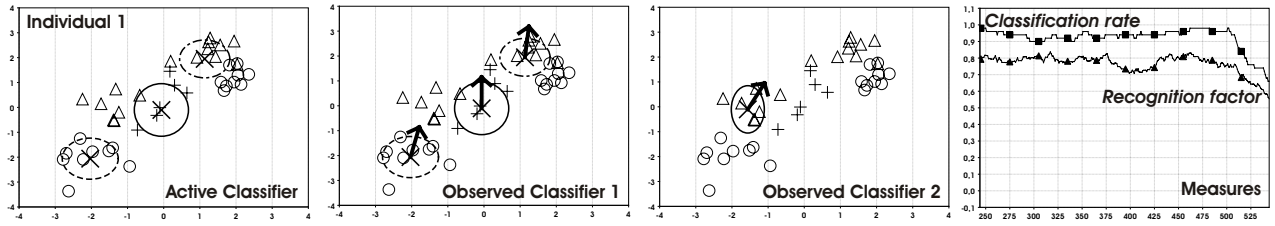

Fig. 4. Phase 2: Individual 1 measures data for which no appropriate rule exists and, therefore, the recognition factor decreases (about time step 500). It decides to create a new rule prototype by means of an on-line clustering mechanism (about time step 525). The new rule can be seen in observed classifier 2.

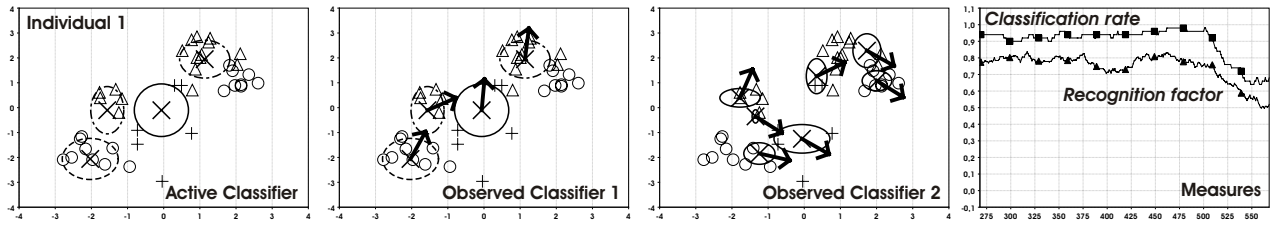

Fig. 5. Phase 3: The new prototype is rated high due to a high activation on the recently measured data. It is accepted and a human expert (here: simulated) is asked to determine a label (class assignment) for the new rule. This rule is then committed to the active classifier. However, the recognition factor of the classifier is still not acceptable. Therefore, the individual decides to create some more prototypes for evaluation (about time step 550).

g_i in the mixture distribution. That is, at each time step k each individual observes a pattern produced by a Gaussian distributed processes which is selected with probability $p_i(k)$. Tables I and II show the parameters for the data generation process.

Individual 1 is equipped with a novelty detection component using $\vartheta_{\text{novelty}} = 0.6$ as decision threshold. The recognition factor is computed using a sliding window of 50 data points. In the prototype generator, the probability is set to $p = 0.25$. The prototype evaluation is parameterized with $\lambda_{\text{reward}} = 0.2$ and $\lambda_{\text{penalty}} = 0.02$ using the linear combination of rule activation and premise dissimilarity as fitness measure. The evaluator for the detection of obsolete rules is parameterized with $\lambda_{\text{reward}} = 0.2$ and $\lambda_{\text{penalty}} = 0.02$. Individual 2 and 3 both

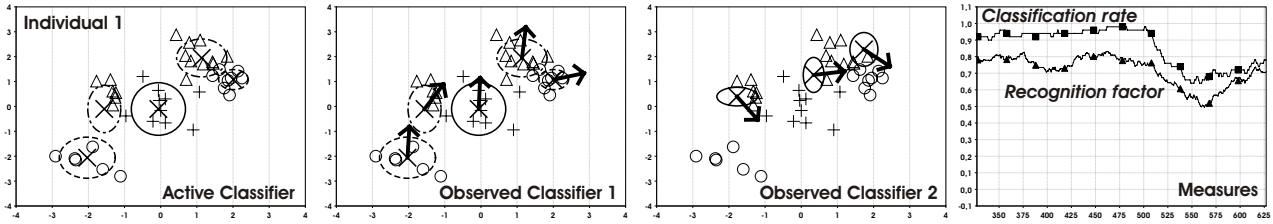

Fig. 6. Phase 4: Only one of these prototypes is rated high enough to be committed to the active classifier (about time step 625).

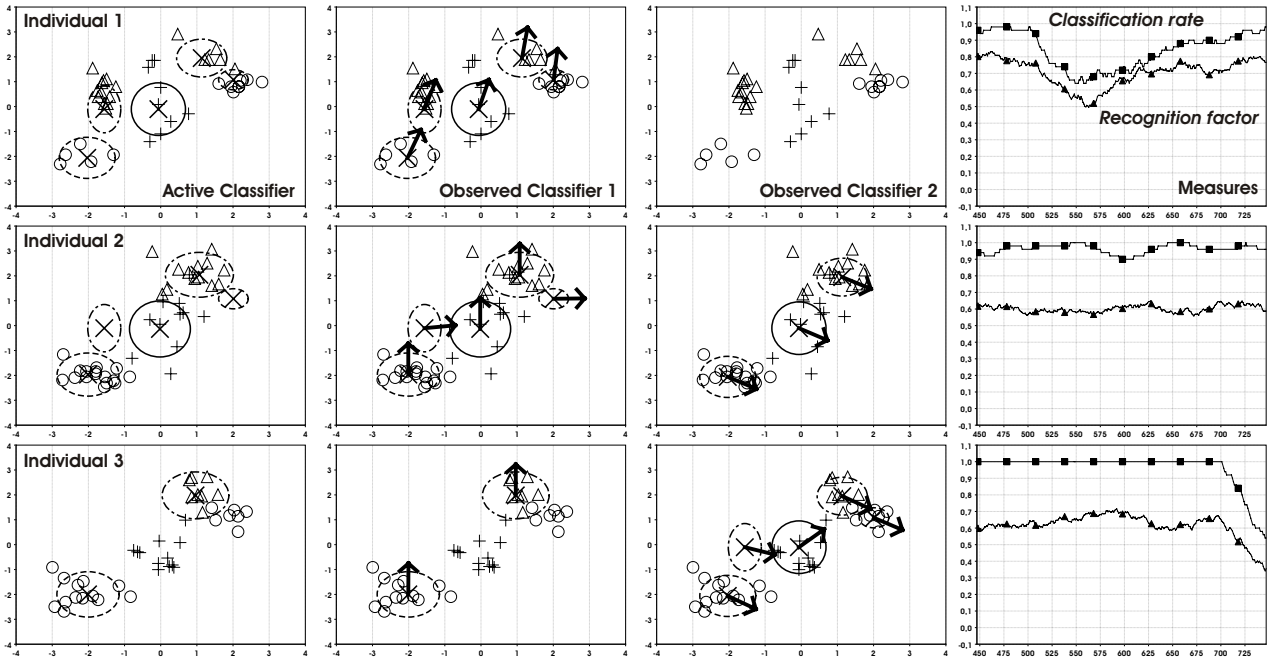

Fig. 7. Phase 5: Approximately at time step 725, individual 1 reaches a state which it assumes to be stable. Then, it sends all its rules (five) to the other individuals. Individual 3 starts to evaluate the activation frequency and the dissimilarity of the received rules. Simultaneously, it measures novel data that can be handled by two of the new rules (near the coordinate points $(2, 1)$ and $(0, 0)$). Individual 2 immediately accepts the two rules that are dissimilar to the rules in the active classifier without evaluating the rules' activation frequency.

TABLE I

PARAMETERS OF THE FIVE GAUSSIANS.

Gaussian	$c_{(i,1)}$	$c_{(i,2)}$	$r_{(i,1)}$	$r_{(i,2)}$	class
g_1	0	0	0.5	1	1
g_2	-2	-2	0.5	0.5	2
g_3	1	2	0.5	0.5	3
g_4	-1.5	0.5	0.3	0.5	3
g_5	2	1	0.2	0.2	2

TABLE II

PARAMETERS OF THE MIXTURE DISTRIBUTIONS (TIME DEPENDENT PROBABILITIES OF GAUSSIANS) FOR EACH INDIVIDUAL.

	Time Steps	p_1	p_2	p_3	p_4	p_5
Individual 1	1 - 500	0.33	0.33	0.33	0	0
	501 - 1000	0.2	0.2	0.2	0.2	0.2
	1001 - 1500	0	0.33	0	0.33	0.33
Individual 2	1 - 1000	0.33	0.33	0.33	0	0
	1001 - 1500	0.2	0.2	0.2	0.2	0.2
Individual 3	1 - 700	0	0.5	0.5	0	0
	701 - 1000	0.25	0.25	0.25	0	0.25
	1001 - 1500	0.2	0.2	0.2	0.2	0.2

do not have a novelty detection component and a prototype generation component. The prototype evaluation components are parameterized with the same parameters as in individual 1, $\lambda_{\text{reward}} = 0.2$ and $\lambda_{\text{penalty}} = 0.02$. Individual 2 utilizes the premise dissimilarity measure, whereas individual 3 uses the minimum of premise dissimilarity and rule activation as a prototype fitness measure.

The experiment is shown in Figures 3 – 10. Each figure corresponds to a certain phase of the experiment (time step). The rows of the figures correspond to the different individuals. The first column shows the two-dimensional input space of an individual's active classifier together with the 40 most recent data points and the active rules. Rules are symbolized by a level curve of the corresponding Gaussian (ellipse) and

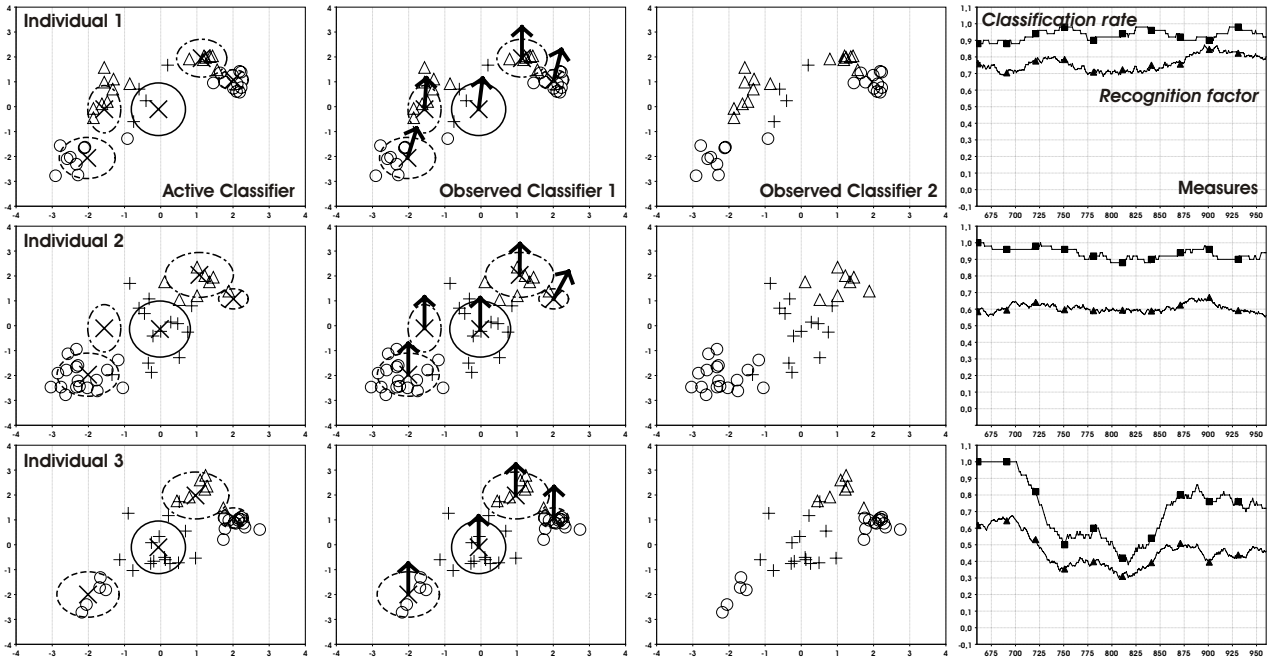

Fig. 8. Phase 6: Individual 3 finally accepted two of the offered rules (roughly at time step 840) and discarded the other rules. That is, it discarded one rule that had been accepted by individual 2.

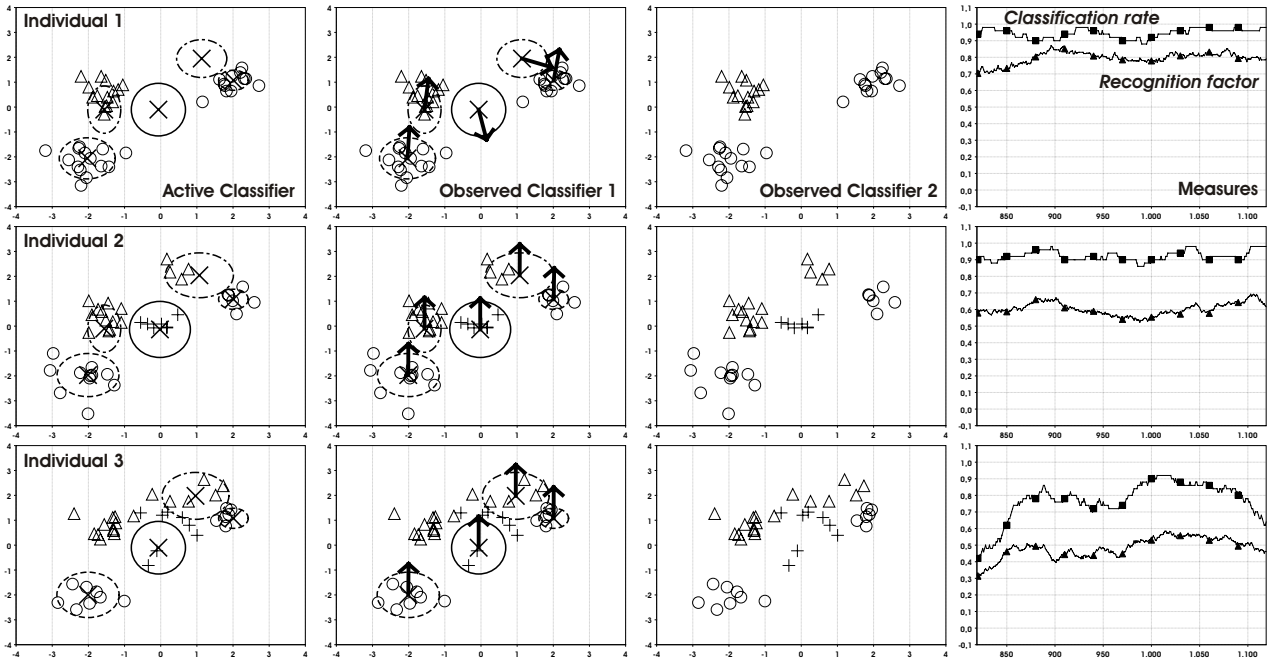

Fig. 9. Phase 7: After time step 1000, the individuals 2 and 3 are confronted with all kind of data (five clusters). Correspondingly, the classification rate of individual 3 is lower than the classification rate of individual 2 which has one more important rule in its active classifier. Individual 1 only receives data corresponding to three clusters and begins to recognize that two rules become obsolete.

the position of the center (big x). The class assignments of rules and data points are indicated by different line and symbol types. The second column (observed classifier 1) shows again the two-dimensional input space with the data points and the active rules together with the rating of non-obsoleteness indicated by the direction (between up and down)

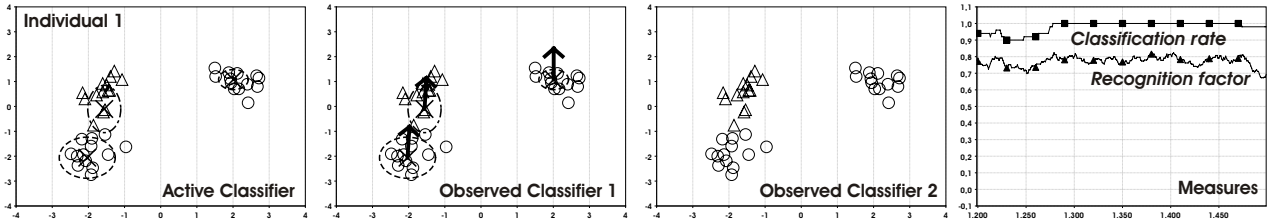

Fig. 10. Phase 8: Finally, individual 1 discarded the two obsolete rules without any deterioration of classification performance.

of thick arrows originating at the rule centers (e.g., ↑: very good, ↓: very bad, →: undecided). Similarly, if applicable, currently observed prototypes and rules (e.g., generated by the prototype generator or delivered by another individual) are drawn in the third column (observed classifier 2). The last column depicts the classification rate of the active classifier and the recognition factor for novelty detection. It must be kept in mind that the individuals do not see the class labels of measured data or their own classification rate. This information is only provided in the figures for a better understanding of the individuals' behavior. Our experiments show that there is a very high correlation (about 0.8) of the classification rate and the recognition factor.

IV. Conclusion and Outlook

In this article we provided basic technologies for knowledge transfer in intelligent systems and demonstrated the feasibility and advantages of these methods. However, this work should be seen as a first step towards the development of a comprehensive toolbox for knowledge transfer. There are many possible application areas for knowledge transfer, for example in the fields of intrusion detection in computer networks (cf. [13]) or monitoring and optimization of production processes. Other application fields include, for example, mobile robots or the Semantic Web. With our approach we intend to extend the adaptation capabilities of agent systems by means of a social learning capability. This enables us to study properties of such systems such as adaptation and self-organization on a more complex, social level, which is a quite interesting aspect of artificial life.

In the future, we will focus on techniques that can be seen as part of a *self-awareness* and an *environment-awareness* component of an intelligent system. The fitness detection mechanisms described here will be part of the former. Additional measures for the assessment of classifiers, rules, and prototypes will be defined to detect concept drift, for instance. The goal is, to provide knowledge transfer techniques that do not require a human expert in many applications. New training techniques are needed to enforce the interpretability and representativity of rules and classifiers. We must improve the techniques for rule integration: Similar rules (or linguistic terms) must be fused – we discuss a first approach in [3] – to keep the classifiers interpretable. Also, we will develop mechanisms for an active measurement of other individuals' competence. Finally, as stated in Section I, we want to develop methods for an assessment of the overall system's emergent behavior (cf. our remarks on emergence in technical systems in [1]). Therefore, we must be able to quantify the impact of knowledge transfer.

Acknowledgements

This work is supported by the German Research Foundation (DFG), grant SI 674/3-1.

References

[1] C. Müller-Schloer and B. Sick, "Emergence in organic computing systems: Discussion of a controversial concept," in *Autonomic and Trusted Computing, Proceedings of the 3rd International Conference ATC-06,* Wuhan, ser. LNCS, L. T. Yang, H. Jin, J. Ma, and T. Ungerer, Eds., no. 4158. Springer Verlag, Berlin, Heidelberg, New York, 2006, pp. 1 – 16.

[2] L. A. Zadeh, "What is soft computing?" *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, vol. 1, no. 1, p. 1, 1997.

[3] O. Buchtala and B. Sick, "Techniques for the fusion of symbolic rules in distributed organic systems," in *Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals/06),* Logan, 2006, pp. 85 – 90.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, Chichester, New York, 2001.

[5] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*. Pearson Education, Upper Saddle River, 2003.

[6] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Knowledge discovery and data mining: Towards a unifying framework," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD 1996),* Portland, 1996, pp. 82 – 88.

[7] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," Massachusetts Institute of Technology – Artificial Intelligence Laboratory & Center for Biological Information Procesing – Whitaker College," A.I. Memo No. 1140, C.B.I.P. Paper No. 31, 1989.

[8] H. C. Andersen, A. Lotfi, and L. C. Westphal, "Comments on "Functional equivalence between radial basis function networks and fuzzy inference systems"," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1529 – 1531, 1998.

[9] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Networks*, vol. 14, no. 4–5, pp. 439–458, 2001.

[10] O. Buchtala, P. Neumann, and B. Sick, "A strategy for an efficient training of radial basis function networks for classification applications," in *Proceedings of the IEEE-INNS International Joint Conference on Neural Networks,* Portland, vol. 2, 2003, pp. 1025 – 1030.

[11] O. Buchtala, "Transformation von Radialen-Basisfunktionen-Netzen in Fuzzy Systeme," Master's thesis, University of Passau, Department of Mathematics and Computer Science, 2005.

[12] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. American Mathematical Society, Providence, 1997.

[13] O. Buchtala and B. Sick, "Functional knowledge exchange within an intelligent distributed system," in *Proceedings of the 20th International Conference on Architecture of Computing Systems – System Aspects in Pervasive and Organic Computing (ARCS 2007),* Zurich, 2007, (accepted – to appear).