Artificial Intelligence 187–188 (2012) 90–114

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Learning from others: Exchange of classification rules in intelligent distributed systems

Dominik Fisch, Martin Jänicke, Edgar Kalkowski, Bernhard Sick *

Intelligent Embedded Systems Lab, Faculty of Electrical Engineering and Computer Science, University of Kassel, Germany

ARTICLE INFO

Article history:
Received 8 November 2010
Received in revised form 31 March 2012
Accepted 6 April 2012
Available online 12 April 2012

Keywords:
Classification
Rule exchange
Collaborative learning
Uncertain knowledge
Probabilistic modeling
Collective intelligence
Interestingness

ABSTRACT

Learning by an exchange of knowledge and experiences enables humans to act efficiently in a very dynamic environment. Thus, it would be highly desirable to enable intelligent distributed systems to behave in a way which follows that biological archetype. We believe that knowledge exchange will become increasingly important in many application areas such as intrusion detection, driver assistance, or robotics. Constituents of a distributed system such as software agents, cars equipped with smart sensors, or intelligent robots may learn from each other by exchanging knowledge in form of classification rules, for instance. This article proposes techniques for the exchange of classification rules that represent uncertain knowledge. For that purpose, we introduce methods for knowledge acquisition in dynamic environments, for gathering and using meta-knowledge about rules (i.e., experience), and for rule exchange in distributed systems. The methods are based on a probabilistic knowledge modeling approach. We describe the results of two case studies where we show that knowledge exchange (exchange of learned rules) may be superior to information exchange (exchange of raw observations, i.e. samples) and demonstrate that the use of experiences (meta-knowledge concerning the rules) may improve that rule exchange process further. Some possible real application scenarios are sketched briefly and an application in the field of intrusion detection in computer networks is elaborated in more detail.

1. Introduction

Animals or humans interact in various ways:

- Individuals independently obey certain rules such that a whole swarm of individuals exhibits a certain behavior (such as in the case of movement of bird flocks or fish schools).
- Individuals exchange information in often simple or sometimes even complex ways in order to achieve a common goal (examples are pheromone trails of ants or bee dance languages).
- Individuals learn from each other by observing each other (babies learn from their parents by mimicking, for instance).
- Individuals learn from each other by communicating their knowledge, e.g., in form of rules (such as children that learn from their teachers in school).
- Individuals learn from each other by exchange of experiences, i.e., meta-knowledge about rules (for example, in a team of more or less experienced engineers who develop a new product together).

* Corresponding author.
E-mail addresses: fisch@uni-kassel.de (D. Fisch), jaenicke@uni-kassel.de (M. Jänicke), edgar.kalkowski@uni-kassel.de (E. Kalkowski), bsick@uni-kassel.de (B. Sick).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.04.002

A self-organized collaboration of individuals that all profit from an exchange of their knowledge or meta-knowledge (experience) can certainly be seen as a higher-level form of *collective intelligence* or *symbiotic intelligence*. The advantages of such a collaboration of individuals may be manifold:

- Individuals may be enabled to react more timely on certain critical situations.
- Individuals may even behave pro-actively: Before they are confronted with certain situations, they already know how to handle them.
- Teams of individuals may be enabled to solve problems that they cannot solve by their own.

In our work, we focus on basic technologies for knowledge exchange (in our case exchange of classification rules) in distributed computer systems such as intelligent teams of cooperating robots, collaborative smart sensor systems, or software agents in the Internet. In dynamic environments, the exchange of locally learned rules will become increasingly important. The overall environment, e.g., the Internet, may be seen as a huge source of information from which knowledge in form of classification rules can be gathered by means of appropriate machine learning techniques. The exchange and utilization of knowledge that is locally acquired but of global importance may lead to a certain kind of self-optimization of the overall distributed system. This is of particular importance in steadily changing environments, where novel knowledge emerges and obsolete knowledge must be discarded (cf. [1–3], for instance).

From a technical viewpoint there are two other important advantages of knowledge exchange: First, the communication effort needed for an exchange of knowledge locally learned from samples (measured data) may be significantly lower than the effort needed for an exchange of that data, as knowledge is more abstract and often more valuable than the raw data. Second, techniques for knowledge exchange are highly independent from a particular application domain (in contrast to an exchange of samples).

In our work, the constituents of the distributed system (simply referred to as *agents* in the following) locally acquire and apply classification rule sets that represent uncertain knowledge such as the rule set

$$\text{if } x_1 \text{ is } \textit{medium} \text{ and } x_2 \text{ is } \textit{high} \text{ and } x_3 \text{ is } \textit{very high} \text{ then } c_1 \text{ is } 0.1 \text{ and } c_2 \text{ is } 0.9,$$
$$\text{if } x_1 \text{ is } \textit{high} \text{ and } x_2 \text{ is } \textit{low} \text{ and } x_3 \text{ is } \textit{low} \text{ then } c_1 \text{ is } 0.8 \text{ and } c_2 \text{ is } 0.2.$$

In this example, we have a three-dimensional input space with attributes x_i ($i \in \{1, 2, 3\}$) that might be numerical or categorical and two output categories c_1 and c_2. The evaluation of a rule premise for a given input sample $\mathbf{x} = (x_1, x_2, x_3)$ leads to a value that describes the gradual assignment of $\mathbf{x}$ to a certain (yet not clearly delimited) region of the input space. The evaluation of a rule then leads to a gradual assignment of $\mathbf{x}$ to certain classes. All rules of the rule set have to be superimposed in an appropriate way to come to a final conclusion about the class membership of $\mathbf{x}$.

At first glance, these rules seem to be fuzzy rules, but we will use a probabilistic framework here to model *uncertainty*. Uncertainty is a term that plays an important role in machine learning, but in the literature it is utilized in various ways [4–7]. Here, the meaning of the term uncertainty is adopted from [5], where *uncertain* is a kind of generic term for other terms such as *likely*, *plausible*, *imprecise*, or *vague*. In our work, we assume that the input samples of a classifier are produced by a set of "data-generating" processes. We are uncertain that a sample is associated with a certain process and is covered by a certain rule. This kind of uncertainty is due to random influences being inherent to the observed process, measurement errors, etc.

Rules that are locally acquired by the agents should be *potentially useful* for other agents. That is, if those agents observe the same processes in their environment, the rules can successfully be applied. As we will see, the same (good) classification results can be obtained with classifiers consisting of very different rule sets. We claim that the set of rule premises must be *generative* to have the property of being potentially useful. We will show how this property can be achieved by means of probabilistic knowledge acquisition techniques (see also [8]). Informally, we state that in our approach the rule premises must model clusters of data in the input space of the classifier for that purpose. A model or a classifier based on a certain model (e.g., a density model of the data) are called "generative" if they could basically be used to generate data that have the same characteristics as the observed data used to find the model parameters. In many cases probabilistic techniques are used to obtain generative models.

Our work addresses classification in *dynamic environments*. We assume that at certain points in time novel processes emerge in the environment, i.e., start to produce data. Also, processes may stop to produce data. In the first case, which we call *novelty*, new rules must be found and added to the rule set. In the second case, which we call *obsoleteness*, rules must be deleted from the rule set. In our case, novelty detection is based on probabilistic models of the observed processes. We also need techniques for obsoleteness detection and for an appropriate reaction on stated novelty or obsoleteness, i.e., for an adaptation of the classifier. We will introduce techniques that have the following two properties:

1. The detection of novelty or obsoleteness is realized without utilizing any feedback from the local environment, i.e., only by assessing the observed input samples. The agent refrains from asking a human application expert or from using information about classification performance which we assume to be unknown to the agent.
2. The reaction on stated novelty or obsoleteness is realized in a partly unsupervised way. The reason is that we want to avoid interaction with a human expert in the case of obsoleteness and reduce it to a minimum in the case of novelty. In the latter case, the premises of new rules are found autonomously by the agent, and humans are only be asked to label them, i.e., to find the conclusions of the new rules.

The reason for focusing on techniques with these properties will become clear later. By labeling rules instead of observed samples, human application experts are involved in a very efficient way. As we will see, the participation of humans can even be avoided in some applications, e.g., if conclusions are implicitly defined by the region of the input space which is covered by a rule premise.

The acquired rules may then be exchanged between the agents in the distributed system. Agents may either accept or discard newly received rules immediately or they may store them in a cache for later use. In this case, it necessary to assess the *usefulness* of rules. If this measure is considered to be high enough, the rule is taken over from the rule cache into the "active" classifier. A rule or a classifier are called "active" if they are currently applied by an agent. We claim that additional meta-knowledge about these rules (i.e., *experience*) might help to improve this process. In this article, we will assess the *informativeness* of rules.

In a nutshell: We focus on basic technologies for an exchange of classification rules representing uncertain knowledge within distributed environments including methods for an acquisition of knowledge (generative rule sets) in dynamic environments, and methods for gathering and using meta-knowledge about rules (experience).

A typical application field of such technologies is the field of network intrusion detection. Intrusion detection agents locally detect novel kinds of attacks, create rule premises for the classification of these attacks by their own, ask system administrators for corresponding rule conclusions, and send the rules to other agents that may profit from that knowledge (either by eventually integrating the rule in their own active classifier or by fusing it with existing knowledge).

The remainder of the article is structured as follows: In Section 2 we make some comments on related work in the field of rule exchange in distributed environments. Work that is related to our methods is cited in Section 3, where we lay all the theoretical and methodological foundations (rule representation, reasoning, novelty and obsoleteness detection, rule acquisition, interestingness assessment). Section 4 presents the results of some case studies and in Section 5 we discuss the lessons we have learned from our research in the field of knowledge exchange. In Section 6 we first sketch some possible applications of the proposed techniques in the fields of driver assistance systems, robotics, and information security. Then, an application in the field of intrusion detection in computer networks is elaborated in some more detail. The major findings are finally summarized in Section 7.

2. Related work

Distributed intelligent systems that work in a collaborative manner recently became an active research issue (for an overview of techniques see, e.g., [9]). They are proposed for various application fields such as smart sensor networks (e.g., [10,11]), the semantic web (e.g., [12]), and distributed intrusion detection systems (e.g., [13,14]). In the majority of these approaches, information is locally acquired and pre-processed by the intelligent systems and thereafter sent to special processing units (which can be either centralized or distributed).

Less common, but closer related to our approach is work dealing with collaborating agents that learn from each other by exchanging locally inferred rules. In [15], agents employ inductive logic programming to adapt to changes in their environment. Different information exchange strategies (low-level, i.e., samples, and high-level, i.e., learned Horn logic rules) are compared and evaluated in a simulated logistic scenario, in which teams of trading agents learn the properties of the environment in order to optimize their operation performance. Ref. [16] proposes collaborating logic-based agents that additionally take the usefulness of exchanged rules into account. Two measures for rule assessment are described. The first one evaluates the benefit of the agent reaching its objective using the received rule. The second one acquires rules from various agents and compares the results. Depending on its usefulness value, a received rule is either integrated into the agent's rule base or discarded. The proposed techniques, however, are not evaluated. Ref. [17] investigates the exchange of different kinds of knowledge (i.e., samples, sample–action–reward vectors, and learned state transitions) between agents that are equipped with reinforcement learning techniques. The different exchange strategies are evaluated within a simulated predator–prey environment. Ref. [18] also considers the exchange of sample–action–reward ratings. There, it is shown that the learning rate of a group of obstacle-avoiding mobile robots can be significantly increased. In [19], an extension of learning classifier systems to multi-agent systems is proposed. At fixed points in time, agents exchange classification rules. Existing rules within an agent's rule base are replaced by received rules if they exhibit a higher strength value. Again, this approach is based on reinforcement learning and, thus, requires permanent feedback from the environment. This is also the case in [20] where agents equipped with support vector machines (SVM) exchange newly learned support vectors. Correctly classified samples yield a reward which is used by the agents to adapt their SVM to changes in the environment. A periodic exchange of knowledge between agents with different learning paradigms (i.e., table based Q-learning and neural networks trained with backpropagation) is presented in [21]. This approach, however, is based on an application- and learner-specific intermediate knowledge representation that must be defined in advance. Additionally, the choice of representation greatly influences the performance of the overall agent system.

Altogether, we can state that our approach that addresses the exchange of classification rules representing *uncertain* knowledge without using feedback from the environment has not been investigated yet. Our novel approach also offers dedicated techniques for rule acquisition and assessment in dynamic environments.

3. Theoretical and methodological foundations

In this section we describe the basic techniques for knowledge representation, acquisition, assessment, and exchange.

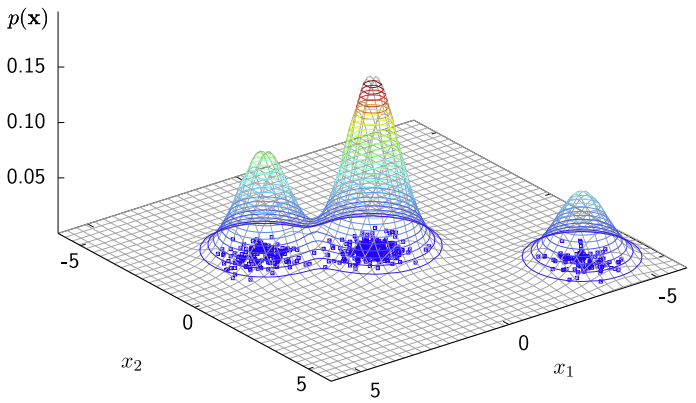


Fig. 1. Example for a GMM with three components.

3.1. Rules representing uncertain knowledge

The classifiers we are using here are probabilistic classifiers. For an input sample $\mathbf{x}$ in an I-dimensional input space we want to compute the posterior distribution of the classes $p(c|\mathbf{x})$, i.e., the probabilities for class membership given an input $\mathbf{x}$. To minimize the risk for classification errors we then select the class with the highest posterior probability (cf. the principle of *winner-takes-all*). According to our previous publications [8,22], $p(c|\mathbf{x})$ can be decomposed as follows:

$$\begin{aligned} p(c|\mathbf{x}) &= \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} = \frac{(\sum_{j=1}^{J} p(\mathbf{x}|j)p(j|c))p(c)}{p(\mathbf{x})} = \sum_{j=1}^{J} \frac{p(j|c)p(c)}{p(j)} \cdot \frac{p(\mathbf{x}|j)p(j)}{p(\mathbf{x})} \\ &= \sum_{j=1}^{J} \underbrace{\frac{\int_{\mathbf{x}\in\mathcal{R}_c} p(j|\mathbf{x})\,\mathrm{d}\mathbf{x} \cdot p(c)}{p(j)}}_{p(c|j)} \cdot \underbrace{\frac{p(\mathbf{x}|j)p(j)}{\sum_{j'=1}^{J} p(\mathbf{x}|j')p(j')}}_{p(j|\mathbf{x})}. \end{aligned} \tag{1}$$

In this classification approach based on a so-called *mixture density model* $p(\mathbf{x})$,

- the conditional densities $p(\mathbf{x}|j)$ ($j \in \{1,\ldots,J\}$) are the components of the model,
- $p(j)$ is a multinomial distribution with parameters π_j (the mixing coefficients or rule weights),
- the $p(c|j)$ are multinomial conditional distributions with parameters $\xi_{j,c}$, and
- $\mathcal{R}_c$ is the (not necessarily connected) region of the input space associated with class c.

That is, we have a classifier (rule set) consisting of J *rules*, where each rule j is described by a distribution $p(j|\mathbf{x})$ (which we call the *rule premise*) and a distribution $p(c|j)$ (which we call the *rule conclusion*). We can state that the former could be trained in an unsupervised way while class labels for samples are needed for the latter. For a particular sample $\mathbf{x}'$, the values $p(j|\mathbf{x}')$ are called responsibilities (i.e., of the component for the sample).

Which kind of density functions can we use for the components? Basically, we may have categorical as well as numerical input dimensions. In the former case, multinomial distributions could be used whereas Gaussian distributions can often be used in the latter. For many practical applications, the use of Gaussian models can be motivated by the generalized *central limit theorem* which roughly states that the sum of independent samples from any distribution with finite mean and variance converges to a normal distribution as the sample size goes to infinity (cf., e.g., [23]). Although being possible, we do not consider categorical input dimensions (i.e., multinomial distributions) in this article. Thus, we assume that the components are multivariate Gaussian (i.e., normal) distributions with centers $\boldsymbol{\mu}_j$ and covariance matrices $\boldsymbol{\Sigma}_j$, i.e.,

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{(2\pi)^{\frac{I}{2}} |\boldsymbol{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-0.5 \left(\Delta_{\boldsymbol{\Sigma}_j}(\mathbf{x}, \boldsymbol{\mu}_j)\right)^2\right) \tag{2}$$

with the distance measure (matrix norm) $\Delta_{\mathbf{M}}$ defined by

$$\Delta_{\mathbf{M}}(\mathbf{v}_A, \mathbf{v}_B) = \sqrt{(\mathbf{v}_A - \mathbf{v}_B)^{\mathrm{T}} \mathbf{M}^{-1} (\mathbf{v}_A - \mathbf{v}_B)}. \tag{3}$$

$\Delta_{\mathbf{M}}$ defines the *Mahalanobis distance* of vectors $\mathbf{v}_A, \mathbf{v}_B \in \mathbb{R}^I$ based on an $I \times I$ covariance matrix $\mathbf{M}$. Fig. 1 gives an example for a Gaussian mixture model (GMM) for $p(\mathbf{x}) = \sum_{j=1}^{3} p(\mathbf{x}|j)p(j)$ in a two-dimensional input space with three components. The level curves correspond to the surfaces of constant density.

This classifier is a so-called *generative* classifier. In the following the classifier is referred to as CMM (classifier based on a mixture model). Generative classifiers not only aim at achieving high classification rates, they also try to model the processes

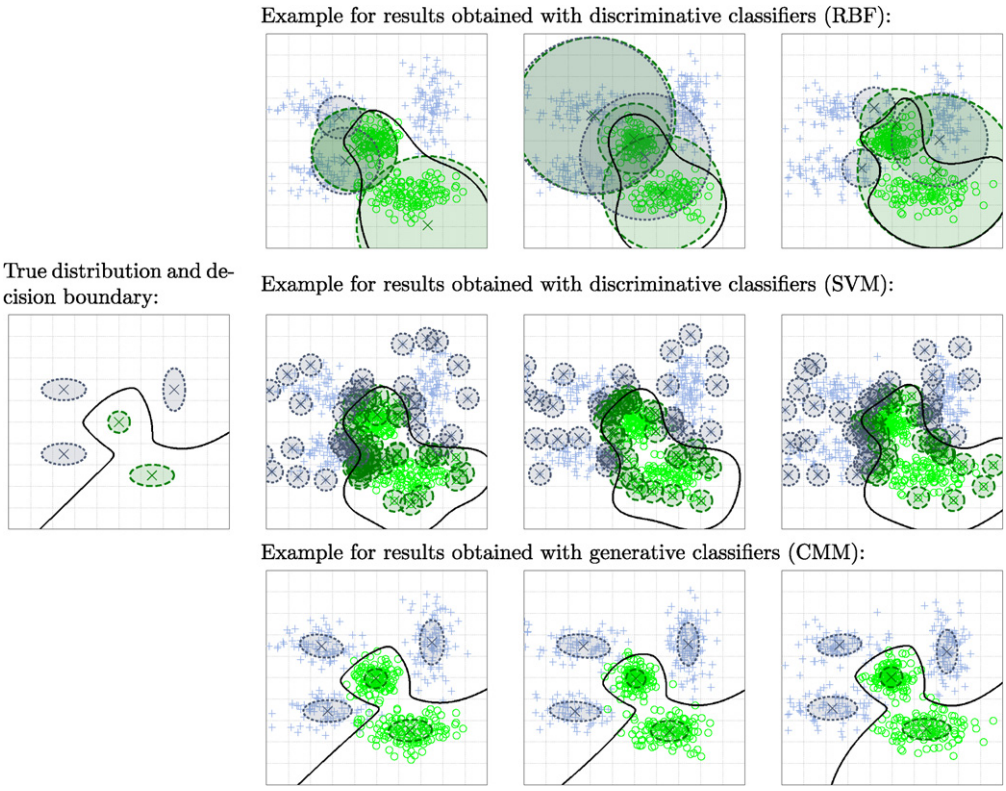


Fig. 2. Comparison of the variability of discriminative and generative classifiers.

from which the observed data originate. For that purpose, the components—which are here assumed to be Gaussian—must model clusters in the data.

As stated in Section 1, knowledge must be potentially useful for other agents. As an important pre-requisite we need training techniques that lead to a kind of "canonical" form of the classifiers. That is, if data originating from the same underlying processes are observed, the rules learned from those data must be similar, i.e., exhibit a low variability (from different sets of samples, very similar rules must be learned). The following illustrative example shows that this is the case for our generative classifier together with the training techniques to be described below but not, for example, for discriminative classifiers whose only objective is to maximize classification performance. Well-known examples for such discriminative classifiers are radial basis function (RBF) neural networks trained with resilient propagation and support vector machines (SVM). If these paradigms use Gaussian basis functions (i.e., as hidden neurons in RBF networks or as kernel in SVM) they are functionally equivalent to our generative approach as we have shown in [22]. Their variability, however, may be substantially higher such that knowledge exchange between RBF networks or between SVM does not make sense.

Fig. 2 shows a probability distribution on the left (three components belong to class "blue", two to class "green"). The input space is two-dimensional. The big Xs describe the centers of the Gaussians, the ellipses are level curves of the Gaussians (surfaces of constant density) with shapes defined by the covariance matrices. The black solid line is an approximation of the decision boundary. Then, we generate three training data sets from the same distribution, each with 500 data points. We train RBF networks, SVM, and our generative classifier CMM on these data sets. The results are depicted on the right of Fig. 2. For RBF and CMM the ellipses represent the placement and the shape of the hidden neurons and the mixture model components, respectively. For SVM, the samples selected as support vectors are indicated by circles. It can be seen that the discriminative solutions (RBF, on the top, SVM in the middle), though having similar decision boundaries (at least in the area between the data clusters), differ greatly in the placement and the form of the hidden neurons and the selected support vectors. The generative approach (CMM, on the bottom) actually models the underlying sample-generating processes and, thus, produces classifiers with low variability.

With regard to the exchange of knowledge in form of rules (i.e., not the whole classifier but only parts of the knowledge base) we can state that classifiers with a low variability are an important pre-requisite as rules are only potentially useful for other agents if they influence the decision boundary of all agents in a similar way. Then, the agents can be enabled to learn from each other. For a discussion of relations between our generative classifier and various other classifier paradigms see [22].

Finally, we state that a rule is given by $I + \frac{I\cdot(I+1)}{2} + 1 + C$ values: the center, the elements on and above the diagonal of the covariance matrix, the mixing coefficient, and the number of classes (C).

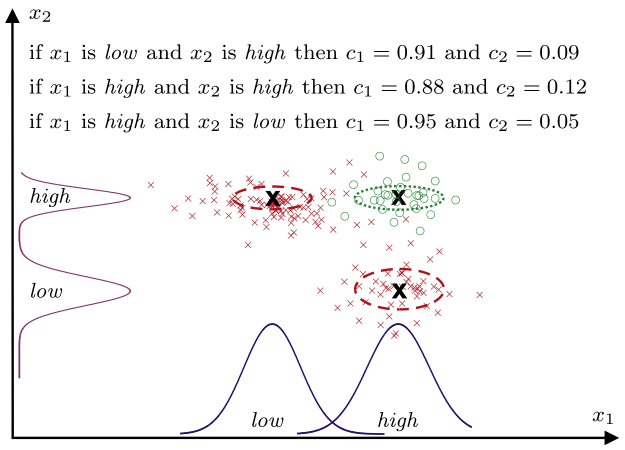


Fig. 3. Example of a classifier consisting of three rules.

3.1.1. Human-readable rules

As a side note we want to mention another interesting property of the described classifier paradigm. In some applications (cf. Section 6) it is desirable to extract human-readable rules from the trained classifier. This is possible with our classifier if it is parametrized accordingly: Basically, there are no restrictions necessary concerning the covariance matrices $\boldsymbol{\Sigma}_j$. However, if they are forced to be diagonal (i.e., assuming no dependencies between input dimensions), each multivariate Gaussian $p(\mathbf{x}|j)$ can be split into a product consisting of I univariate Gaussians $\psi_{i,j}$. Then, the following human-readable rule set can be extracted from a classifier:

$$\text{if } x_1 \text{ is } \psi_{1,1} \ldots \text{ and } x_i \text{ is } \psi_{i,1} \ldots \text{ and } x_I \text{ is } \psi_{I,1} \text{ then } c_1 \text{ is } \xi_{1,1} \text{ and } c_2 \text{ is } \xi_{1,2} \text{ and } \ldots$$

$$\vdots$$

$$\text{if } x_1 \text{ is } \psi_{1,J} \ldots \text{ and } x_i \text{ is } \psi_{i,J} \ldots \text{ and } x_I \text{ is } \psi_{I,J} \text{ then } c_1 \text{ is } \xi_{J,1} \text{ and } c_2 \text{ is } \xi_{J,2} \text{ and } \ldots$$

The variables in this rule set are the input variables x_i (components of the I-dimensional input variable $\mathbf{x}$), and the output variable c which represents the classes. The rule premises are realized by conjunctions of the univariate Gaussians $\psi_{i,j}$ (which are obtained from $p(\mathbf{x}|j)$) whereas the conclusions are modeled by the $\xi_{j,c}$ (which essentially correspond to the parameters of the $p(c|j)$ as mentioned above; see [22] for more details). These rules enable reasoning based on uncertain samples as shown in Eq. (1). They have a form which is very similar to that of fuzzy rules, but they have a different (i.e., probabilistic) interpretation.

Fig. 3 gives an example for such a classifier. Again, the big Xs describe the centers of the rules, the ellipses are level curves of the multivariate Gaussians (surfaces of constant density) with shapes defined by the covariance matrices. In the case of diagonal covariance matrices, these ellipses are axes-oriented; in the case of isotropic covariance matrices they become circles. Of course, this readability is accomplished at the cost of a limited modeling capability of the classifier and should, thus, only be used if the application demands this kind of human-readable rules.

3.2. Knowledge acquisition in dynamic environments

In the following we distinguish two situations where a classifier must be trained: An off-line training of the classifier with a labeled data set which is needed to build a classifier for a static environment and an on-line technique that adapts an existing classifier whenever new rules become necessary (novelty) or existing rules are no longer needed (obsoleteness). This approach, which is needed in a dynamic environment, is based on the technique needed for the static case.

3.2.1. Off-line knowledge acquisition

The basic technique to train the various parameters of the classifier in an off-line manner is a *maximum likelihood* (ML) technique. Given a training data set consisting of an input data set $\mathbf{X}$ with N input samples $\mathbf{x}_n$ and a corresponding target (classes) data set $\mathbf{T}$ it is assumed that the $\mathbf{x}_n$ are independent and identically distributed (i.i.d.). Then, the parameters of the density $p(\mathbf{x})$ are computed in an unsupervised manner. With $\boldsymbol{\theta}$ being the overall set of model parameters consisting of all $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j the log-likelihood function

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta}) \tag{4}$$

must be maximized with respect to the parameters $\boldsymbol{\theta}$. For the mixture density models used here a closed formula for the optimization of the log-likelihood does not exist. Instead, an iterative method called *expectation maximization* (EM) can be applied (cf. [24–26], for instance). EM iteratively maximizes the likelihood in two alternating steps, the E (expectation) step and the M (maximization) step. Once the parameters of the mixture model $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j have been computed this way, the parameters $\xi_{j,c}$ can be obtained in a supervised approach using $\mathbf{T}$.

We use a more sophisticated version of this EM technique called variational Bayesian inference (VI, for details see [8,24]). The main difference is that VI is a Bayesian parameter estimation technique that considers prior knowledge in order to make the training process more robust (i.e., to avoid singularities of the likelihood). Our version of VI is slightly modified to optimize the number of components by its own.

It is assumed that for the "generation" of a sample $\mathbf{x}_n$ one of the J components has actually been "responsible". To describe the "assignment" of samples to components, an additional random variable $\mathbf{z}_n$ (so-called *latent variable*) is introduced for each sample. With $\mathbf{Z}$ denoting the set of all latent variables, the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ now becomes

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{5}$$

In a Bayesian parameter estimation approach such as VI (cf. [27,24]) prior distributions are introduced for the parameters $\boldsymbol{\theta}$. The functional form of the prior distributions must be chosen in a way such that the posterior distributions have the same type (so-called conjugate density functions) [24].

For a more detailed discussion on Bayesian inference and, particularly, VI see [24]. Here, we only give the resulting formulas needed for the estimation of the parameters of the Gaussian mixture model.

For the parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Lambda}_j$ (note that $\boldsymbol{\Lambda}_j = \boldsymbol{\Sigma}_j^{-1}$), a Gaussian–Wishart distribution must be chosen as prior [24], i.e.,

$$q(\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j) = \mathcal{N}\big(\boldsymbol{\mu}_j|\mathbf{m}_j, (\beta_j\boldsymbol{\Lambda}_j)^{-1}\big)\mathcal{W}(\boldsymbol{\Lambda}_j|\mathbf{W}_j, \nu_j), \tag{6}$$

where $\mathbf{m}_j$, β_j, $\mathbf{W}_j$ and ν_j are temporary variables that are determined during the iterative optimization process (see below). A Dirichlet distribution is chosen as prior of the mixing coefficients π_j, i.e.,

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}). \tag{7}$$

Here, $\boldsymbol{\alpha}$ consists of J temporary variables α_j.

Similar to EM, VI is composed of alternating E and M steps.

In the E step, the responsibilities $\gamma_{n,j}$ (i.e., of component j for sample $\mathbf{x}_n$) are estimated:

$$\gamma_{n,j} = \frac{\rho_{n,j}}{\sum_{j'=1}^{J} \rho_{n,j'}}, \tag{8}$$

with

$$\ln \rho_{n,j} = \mathbb{E}[\ln \pi_j] + \frac{1}{2}\mathbb{E}\big[\ln|\boldsymbol{\Lambda}_j|\big] - \frac{I}{2}\ln(2\pi) - \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_j,\boldsymbol{\Lambda}_j}\big[(\mathbf{x}_n - \boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Lambda}_k(\mathbf{x}_n - \boldsymbol{\mu}_j)\big] \tag{9}$$

where ($\psi(\cdot)$ is the Digamma function)

$$\mathbb{E}[\ln \pi_j] = \psi(\alpha_j) - \psi\left(\sum_{j'=1}^{J} \alpha_{j'}\right), \tag{10}$$

$$\mathbb{E}\big[\ln|\boldsymbol{\Lambda}_j|\big] = \sum_{i=1}^{I} \psi\left(\frac{\nu_j + 1 - i}{2}\right) + I\ln 2 + \ln|\mathbf{W}_j|, \tag{11}$$

$$\mathbb{E}_{\boldsymbol{\mu}_j,\boldsymbol{\Lambda}_j}\big[(\mathbf{x}_n - \boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Lambda}_j(\mathbf{x}_n - \boldsymbol{\mu}_j)\big] = I\beta_j^{-1} + \nu_j(\mathbf{x}_n - \mathbf{m}_j)^{\mathrm{T}}\mathbf{W}_j(\mathbf{x}_n - \mathbf{m}_j). \tag{12}$$

The model parameters are estimated in the M step: With

$$N_j = \sum_{n=1}^{N} \gamma_{n,j}, \tag{13}$$

being the "effective" number of samples "generated" by component j the estimates are:

$$\boldsymbol{\mu}_j = \frac{1}{N_j}\sum_{n=1}^{N} \gamma_{n,j}\mathbf{x}_n, \tag{14}$$

$$\boldsymbol{\Lambda}_j^{-1} = \frac{1}{N_j}\sum_{n=1}^{N} \gamma_{n,j}(\mathbf{x}_n - \boldsymbol{\mu}_j)(\mathbf{x}_n - \boldsymbol{\mu}_j)^{\mathrm{T}}, \tag{15}$$

$$\pi_j = \frac{\alpha_0 + N_j}{J\alpha_0 + N}. \tag{16}$$

It is also necessary to update the temporary variables that are needed again for the subsequent E step:

$$\alpha_j = \alpha_0 + N_j, \tag{17}$$

$$\beta_j = \beta_0 + N_j, \tag{18}$$

$$\mathbf{m}_j = \frac{1}{\beta_j}(\beta_0 \mathbf{m}_0 + N_j \boldsymbol{\mu}_j), \tag{19}$$

$$\mathbf{W}_j^{-1} = \mathbf{W}_0^{-1} + N_j \boldsymbol{\Lambda}_j^{-1} + \frac{\beta_0 N_j}{\beta_0 + N_j}(\boldsymbol{\mu}_j - \mathbf{m}_0)(\boldsymbol{\mu}_j - \mathbf{m}_0)^{\mathrm{T}}, \tag{20}$$

$$\nu_j = \nu_0 + N_j. \tag{21}$$

The parameters indexed with 0, namely α_0, β_0, $\mathbf{m}_0$, ν_0, and $\mathbf{W}_0$ are the hyper-parameters of the VI which can be used to influence the behavior of the algorithm in a desired way, e.g., to avoid singularities or to cope with sparse data.

The values of these hyper-parameters must be set depending on the data set (e.g., the overall mean of the data set, the number of input dimensions I, or the overall variance of the data set) and the model properties (e.g., the number of components J).

E and M steps are iterated until a given stopping criterion is met (e.g., no or only slight improvements of the likelihood for a fixed number of steps). Then, it is possible to determine the output weights, i.e., the ML estimators of the parameters $\xi_{j,c}$ of multinomial distributions $p(c|j)$. For this supervised step we need the set of targets $\mathbf{T}$. With I_c we denote the index set of all samples from the overall training set $\mathbf{X}$ for which c is the assigned target class. Then, with $p(j|c) = \int_{\mathbf{x}\in\mathcal{R}_c} p(j|\mathbf{x})\,\mathrm{d}\mathbf{x}$ where $\mathcal{R}_c$ is the region of the input space associated with class c, we get the ML estimators

$$\xi_{j,c} = \frac{\frac{1}{|I_c|}\sum_{n\in I_c} \gamma_{n,j} \cdot \frac{|I_c|}{N}}{\frac{N_j}{N}} \tag{22}$$

$$= \frac{1}{N_j}\sum_{n\in I_c} \gamma_{n,j}. \tag{23}$$

The optimization of the component number is done implicitly by the VI algorithm. The "effective" number of samples N_j for which a component is responsible (cf. Eq. (13)) can be used as a decision criterion. The higher this number, the more "relevant" is the respective component. If a component is not relevant enough (a test criterion is realized with a simple threshold), it is simply deleted from the model. That is, the VI training approach must be started with a number of components that must be higher than the number that is expected to be required.

Algorithm 1: Off-line classifier training.

1. The parameters of the Gaussian mixture model are found in an unsupervised approach:
   (a) Initialize the parameters $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j and the hyper-parameters α_0, β_0, $\mathbf{m}_0$, ν_0 and $\mathbf{W}_0$ with appropriate, valid values.
   (b) Compute the responsibilities $\gamma_{n,j}$ of a component j for a sample $\mathbf{x}_n$ in the expectation (E) step according to Eq. (8).
   (c) Compute estimates of the model parameters $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j in the maximization (M) step according to Eqs. (14), (15), and (16).
   (d) If an appropriate stopping criterion (e.g., number of iterations) is not met, go to step 1(b).
2. The parameters of the output distributions are then found in a supervised approach with Eq. (23).

Algorithm 1 shows that the rule premises can be found in an unsupervised step whereas the rule conclusions are determined in a second, supervised step. The two-step training procedure also allows to construct the classifier in a slightly different way if labeled data are not available: After the first, unsupervised step, the components may also be labeled by a human domain expert.

By defining distributions over the parameters of the classifier, VI explicitly models uncertainty regarding the parameterization of the classifier. This could be interpreted as a kind of *confidence* in rules (cf. also [28]), a fact which is not fully exploited up to now (cf. Section 7).

3.2.2. On-line knowledge acquisition

In a dynamic environment, an agent must be able to detect the need for either generating new rules or taking over rules obtained from other agents. Also, the need for discarding obsolete rules must be detected. Then, both situations must be handled appropriately.

In the literature exist a number of different approaches to realize novelty detection (for an overview see, e.g., [29–32]; techniques based on GMM are introduced in [33–37]). The key problem of on-line novelty or obsoleteness detection is that in the case of uncertain information a single sample is not sufficient to make a definite decision. An example for this

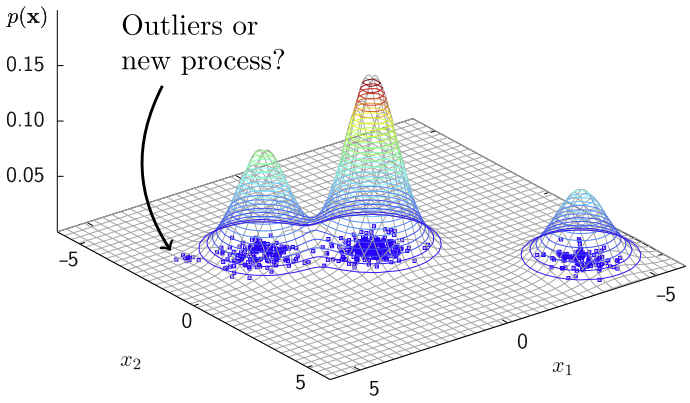


Fig. 4. Example of the novelty detection challenge.

challenge is illustrated in Fig. 4. The question is here: Are the observations on the left-hand side outliers from the existing processes or are they the first observations generated by a newly emerging process?

In order to deal with this problem, our novel approaches for on-line detection of novelty and obsoleteness basically realize the following idea:

- Each observed sample is assessed with respect to a certain property (e.g., a possible assignment to an existing rule).
- An appropriate penalty or reward value is determined.
- These values are accumulated in order to determine the "status" of single rules or the overall rule set.
- It is checked whether a "dissatisfaction" threshold is reached.

We sketch the algorithm for novelty detection here. This algorithm checks whether a new sample $\mathbf{x}$ can be associated with one of the existing rules (i.e., components) in the active classifier or not. As we use a GMM for $p(\mathbf{x})$, we use a user-defined parameter α to define a hyper-ellipsoid around each center $\boldsymbol{\mu}_j$ such that we can expect that a $1-\alpha$ percentage of the labels produced by the process which is modeled by component j lies within that hyper-ellipsoid. Let $\mathcal{E}_j \subset \mathbb{R}^I$ be the hyper-ellipsoid of component j with size and shape defined by $\boldsymbol{\Sigma}_j$.

Algorithm 2: Novelty detection.

1. For each rule j compute the squared Mahalanobis distance between the sample $\mathbf{x}$ and the center $\boldsymbol{\mu}_j$ (cf. Eq. (3)) and compare this distance to a threshold γ to decide on the assignment of the sample to rule j:

$$z_j(\mathbf{x}) := \begin{cases} 1, & \text{if } (\Delta_{\boldsymbol{\Sigma}_j}(\mathbf{x}, \boldsymbol{\mu}_j))^2 \leqslant \gamma, \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

   The threshold γ is determined by means of a $1-\alpha$ quantile of the χ^2 distribution as the squared distances can be assumed to be distributed this way (typically, $\alpha \in \{0.05, 0.1\}$).
2. Check whether the sample $\mathbf{x}$ can be assigned to at least one rule (note that $z(\mathbf{x}) \in \{0, 1\}$):

$$z(\mathbf{x}) := 1 - \prod_j \big(1 - z_j(\mathbf{x})\big). \tag{25}$$

3. Compute an update value for the overall novelty status of the classifier (rule set):

$$\Delta(\mathbf{x}) := \eta \cdot \Big(\underbrace{-\nu \cdot \big(1 - z(\mathbf{x})\big)}_{\text{penalty}} + \underbrace{(1-\nu) \cdot z(\mathbf{x})}_{\text{reward}} \Big), \tag{26}$$

   with ν being the sensitivity which is needed to correct influences of overlapping components (see below) and η being the step size which controls the reaction time.
4. The new novelty status is then

$$s_{\text{nov}} := s_{\text{nov}} + \Delta(\mathbf{x}) \tag{27}$$

   where s_{nov}, which must be initialized appropriately (e.g., with $s_{\text{nov}} := 1$), can be regarded as the degree of "satisfaction" with respect to the currently used rules.
5. If s_{nov} sinks below a given threshold σ (e.g., $\sigma := 0.2$), there is a need to integrate one or several new rules into the classifier.

How can the factor ν be determined? Basically, ν must be chosen in a way such that there is an equilibrium of penalties and rewards as long as no changes of the underlying distribution $p(\mathbf{x})$ occur. That is, the expectation of the updates must be

$\mathbb{E}_{p(z(\mathbf{x}))}[\Delta(\mathbf{x})] = 0$. Therefore, ν must depend on the probability that a sample lies within the union of the hyper-ellipsoids $\mathcal{E} = \bigcup_{j=1}^{J} \mathcal{E}_j$. That is,

$$\nu = p\big(z(\mathbf{x}) = 1\big) = \int_{\mathcal{E}} p(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{28}$$

The integral can be approximatively evaluated by means of an appropriate Monte Carlo method. The factor ν must be re-estimated after every adaption of the rule set, e.g., in case of novel or obsolete rules. The time complexity for processing one sample with the novelty detection algorithm is $\mathcal{O}(J \cdot I^2)$ which is very efficient and well suited for on-line applications.

Whenever novelty is stated, the rule set must be adapted accordingly by adding new rules. Basically, we may use the VI technique on a sliding window of recent samples to find new rule premises. To avoid changes of the already existing premises, the centers and covariance matrices of existing components are fixed and only those of new components and all the mixture coefficients are adapted. A rule conclusion, i.e., an estimate of the parameters of the distribution $p(c|j)$ for a new component j can then be obtained in various ways:

1. Application experts can be asked to label a set of recently observed samples (e.g., measured within a sliding window). These labels are then used to determine values for the parameters $\xi_{j,c}$.
2. Application experts can be asked to label a new rule j, i.e., to assign it uniquely to one of the classes.
3. In the case of rule exchange, certain rules, and in particular their conclusions, may be taken over from other agents. This alternative is described in much more detail in Section 3.3.
4. Rule conclusions may also be given implicitly if, for instance, certain regions of the input space are known to be assigned to certain classes. This includes the case that an agent stores discarded rules in a rule cache for later use.

In any case, the output distributions of all other rules must be adapted as well (see below).

In our case studies in Section 4 we combine the cases 2 and 3 (i.e., experts are asked if no rules received from other agents are available in a cache) and show that this approach is in some aspects superior to a solution where samples are exchanged between agents instead of rules.

As an example, we sketch the algorithm for novelty handling in case 2 here, which again leads to a probabilistic, generative classifier.

Algorithm 3: Novelty handling with human experts.

1. Use a sliding window of N recently observed samples $\mathbf{x}_n$ to find the parameters of new components (rule premises) and to adapt the mixing coefficients of the already existing (i.e., "old") components:
   (a) Chose a number J^{temp} that is significantly higher than the number of new components that is expected to be needed (which is often only one or two).
   (b) Copy the parameters $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j of the components in the old classifier into a new GMM and initialize the parameters for a number J^{temp} of new components. To find initial centers start with the sample $\mathbf{x}_n$ from the sliding window with

$$\mathbf{x}_n := \arg\min_{\mathbf{x}'_n} \sum_{j=1}^{J} \pi_j \mathcal{N}\big(\mathbf{x}'_n \big| \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big) \tag{29}$$

   computed for the old GMM. The other $J^{\text{temp}} - 1$ new centers are then chosen with a large spread in the input space (cf. [38,39]).
   (c) Run the unsupervised step of the VI algorithm using the samples in the sliding window while keeping the parameters $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ of the already existing rules fixed. VI also determines the number of new rules J^{new} that is actually needed (with $J^{\text{new}} \leqslant J^{\text{temp}}$).
2. Then, the parameters $\xi_{j,c}$ of the new classifier (rule conclusions) are determined by means of an appropriate random sampling technique. For that purpose, we assume a unique assignment of components to classes in the "true" world:
   (a) For each component that already existed in the old GMM determine the class with the highest posterior probability (using the old classifier):

$$\kappa_j := \arg\max_{c} (\xi_{j,c}). \tag{30}$$

   (b) Ask a human application expert to label each new component uniquely and set the corresponding variables κ_j. For that purpose, the rules can be presented in a human-readable form as set out in Section 3.1.
   (c) Take the new GMM as random generator model for a number of artificial input samples and label these samples using the κ_j of the respective rules.
   (d) Run the supervised step of the VI algorithm (Eq. (23)) on these artificial data to determine values for all $\xi_{j,c}$ of the new classifier.

If human experts are assumed to make errors, several experts may be asked and their statements may be superimposed as we have shown in [40]. If the decision in Eq. (30) is not very clear, experts may be asked as well.

Obsoleteness detection is basically very similar to novelty detection. The main difference is that individual measures are needed for every rule in the classifier. Also, penalty and reward factors must be set individually for each rule. Whenever obsoleteness is stated, the corresponding component is deleted and the parameters π_j and $\xi_{j,c}$ of the remaining components are determined anew in a way which is very similar to the "novelty handling" case.

Altogether, we can state that the adaption of rule premises can be done in an unsupervised way, i.e., autonomously by the agents themselves. For the rule conclusions we may need application experts in some cases, but their effort is kept as low as possible as rules are taken over from other agents whenever possible and experts are asked to label rules instead of a (often large number of) samples which can be done much more efficiently.

Finally, we emphasize that both training techniques (on-line and off-line) lead to generative classifiers.

3.3. Interestingness-based rule exchange in a distributed environment

In this article, we only use quite elementary communication techniques—a simple broadcasting of novel rules within the distributed system—and focus on the use of meta-knowledge (knowledge about knowledge, i.e., experience) to improve the rule exchange process. Here, this meta-knowledge is gathered by the recipient of a rule; but basically it could also be transmitted together with a novel rule by the sender.

As mentioned in Section 1, we will assess and exploit the *interestingness* of rules. In the data mining field, interestingness measures are used to decide upon the importance of automatically discovered knowledge (e.g., in form of decision trees, fuzzy rule sets, or association rules) in terms of coverage, support, accuracy, unexpectedness, usefulness, or novelty, for instance. Interestingness measures can be divided into objective measures based, e.g., on statistical properties and subjective measures that are derived from a user's beliefs, expectation, or knowledge [41,42]. Here, we also consider several aspects of interestingness. We may wish to answer the following questions, for instance:

- How *useful* is a rule at this moment, i.e., does it make sense to apply it to the current situation in the environment?
- How *informative* is a rule, i.e., how well-distinguishible is the underlying process from other processes?
- How *unique* is the conclusion of a rule, i.e., the assignment to a class?
- How *important* is the rule compared to other rules in a given rule set?
- How *generative* is the premise of a rule, i.e., how precise does it model an observed process?
- How *comprehensible* is a rule or a rule set for human application experts?

These properties must be measured numerically. In this article, we focus on usefulness and informativeness. We will briefly comment on uniqueness and importance and refer to measures we have described in [22,43] concerning generativity and comprehensibility.

3.3.1. Assessment and exploitation of a rule's usefulness

Whenever a novel rule is received by an agent, it can either be accepted immediately or its usefulness can be assessed over time in order to accept the rule when it is regarded as useful to classify the currently observed samples. In the former case it is taken over into the "active" classifier of the agent as an additional rule or it is fused with an existing rule (cf. [3]). In the latter case, which will be considered here, it is preliminarily stored in a cache and its usefulness is rated over time.

A rule is regarded as being useful if a sufficient amount of samples can be assigned to that rule. Therefore, the method for usefulness assessment is very similar to the penalty/reward methods for the detection of novelty or obsoleteness (cf. Section 3.2). We define a hyper-ellipsoid $\mathcal{E}_j$ around the center of the new rule j, check whether a sample can be assigned to that rule, and determine an update value for the usefulness status s_{use_j} of the rule. Compared to Algorithm 2, we have a slightly different update formula (cf. Eq. (26)), namely

$$\Delta(\mathbf{x}) = \eta_j \cdot \Big(\underbrace{-\nu_j \cdot \big(1 - z_j(\mathbf{x})\big)}_{\text{penalty}} + \underbrace{(1 - \nu_j) \cdot z_j(\mathbf{x})}_{\text{reward}} \Big), \tag{31}$$

with a user-defined step size η_j and

$$\nu_j = \int_{\mathcal{E}_j} p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{32}$$

with integration over $\mathcal{E}_j$ instead of $\mathcal{E}$ (cf. Eq. (28)). The $z_j(\mathbf{x})$ are determined as shown in Eq. (24). If the usefulness status s_{use_j} of the rule exceeds a given threshold, the rule is accepted and added to the rule set of the active classifier. Then, the conclusions of all rules must be adapted such as in the case of novelty handling (cf. Algorithm 3).

Rules may be discarded (deleted from the cache) if they are not accepted within a certain time interval. To avoid a situation where a rule is deleted due to penalization, a lower bound is set to the usefulness status. More concretely, s_{use_j} is not allowed to sink below its initial value—which may be set individually for each rule as described below.

3.3.2. Assessment and exploitation of a rule's informativeness

A novel rule is considered as being very informative by the recipient if it describes a really novel kind of process that might produce data in the environment. To assess informativeness numerically we need a kind of distance measure for rule premises. For this purpose, we use the *Hellinger* distance $\mathrm{Hel}(p, q)$ of two probability densities:

$$\mathrm{Hel}(p,q) = \sqrt{1 - BC(p,q)}, \tag{33}$$

where $BC(p,q)$ denotes the Bhattacharyya coefficient of two distributions defined by

$$BC(p,q) = \int \sqrt{p(\mathbf{x})q(\mathbf{x})}\,\mathrm{d}\mathbf{x}. \tag{34}$$

If, however, p and q are multivariate Gaussians (as in our case the rule premises) then

$$BC(p,q) = \exp\left(-\frac{1}{8}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \left(\frac{\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q}{2}\right)^{-1} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right) \cdot \frac{\sqrt[4]{|\boldsymbol{\Sigma}_p||\boldsymbol{\Sigma}_q|}}{\sqrt{|\frac{\boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q}{2}|}}, \tag{35}$$

where $|\boldsymbol{\Sigma}|$ denotes the determinant of matrix $\boldsymbol{\Sigma}$. The value of $\mathrm{Hel}(p,q)$ is always in the range between 0 and 1. It is 0 if p and q describe the same distribution. The maximum value of 1 is achieved when p places all its probability mass in regions where q assigns a probability of zero and vice versa. The informativeness status s_{inf_j} of a new rule j is determined by its Hellinger distance calculated with respect to the "closest" rule within the currently used active classifier. Rules with $s_j < \psi$ are fused immediately if the conclusions do not contradict. The threshold ψ can be set easily; typically by choosing a rather low value.

Here, we want to accelerate the process of rule acceptance depending on the degree of informativeness: Informative rules should more quickly be transferred from the cache into the active classifier. To achieve this, the step size η_j of the rule's usefulness status s_{use_j} (see above) can be set to a certain fraction of s_{inf_j}. It is also possible to initialize s_{use_j} with s_{inf_j}.

3.3.3. Uniqueness, importance, and combination of interestingness measures

Certainly, various interestingness measures could be defined, and they could be combined and employed in various ways depending on the needs of a particular application.

The *uniqueness* of a rule's conclusion, for instance, can be rated by the agent that receives a new rule. It measures how distinct the conclusion of the new rule is. For that purpose, it is possible to evaluate the difference between the largest value $\xi_{j,c}$—the estimate of the class posterior probability given a certain component—and the second largest. Rules with unique conclusions may also be accepted more easily.

The *importance* of a rule measures the relative weight of a rule in a rule set. A rule is regarded as very important, for example, if its mixing coefficient π_j—the estimate of the component's prior probability—is far above the average mixing coefficient of the sender. The sending agent must also provide information about the number of rules in the rule set in this case. Important rules may be accepted more easily, too.

A linear combination of interestingness measures can be used to control a rule's usefulness assessment, e.g., by initializing the usefulness status or setting the step size as mentioned above.

4. Case studies

In this section we will demonstrate the behavior of the proposed techniques and investigate the performance of different kinds of rule exchange strategies. First, we compare the exchange of information (i.e., samples) to the exchange of knowledge (i.e., learned rules) with respect to classification rates, communication costs, and human expert query costs. The second case study demonstrates the effects of employing interestingness measures in the rule exchange process.

4.1. Information and knowledge exchange

In a first case study, we investigate the behavior of agents that use three different learning strategies. In a simple dynamic scenario four agents classify observed samples. The observed scenario consists of three (initially two) processes that generate normally distributed samples, two (initially one) generating "blue" samples (crosses) and one generating "green" ones (circles). The agents are able to adapt to changes in their observed environment as described in the previous section. All agents basically monitor the same scenario, however, agent 1 observes samples of newly emerging processes earlier than the other agents. The agents' behavior can be summarized as follows:

- Agent 1 is able to "share" its superior sensing and learning ability with other agents by sending all the observed samples as well as the learned rules.
- Agent 2 does not make use of any messages sent by agent 1. The agent learns rules by its own using only the locally observed samples. Like agent 1, it is able to detect novelty, create new rule premises, and query a human expert for rule conclusions. This strategy has some characteristics of a "learning by doing" approach. "Learning by doing" turned out to be an important concept in professional human education. Basically, it says that humans are able to improve their skills and productivity through practice and self-perfection.
- Agent 3 learns from agent 1 by using all samples observed and sent by agent 1. All received samples are merged with "own", i.e., locally observed, samples and are stored in a sample cache. With this sample cache, agent 3 can also perform any necessary actions such as detecting novelty, creating new rule premises, and querying a human expert for rule conclusions. If a new rule is created, it stays in a rule cache until it proved itself being useful. Then, it is integrated into the active classifier.

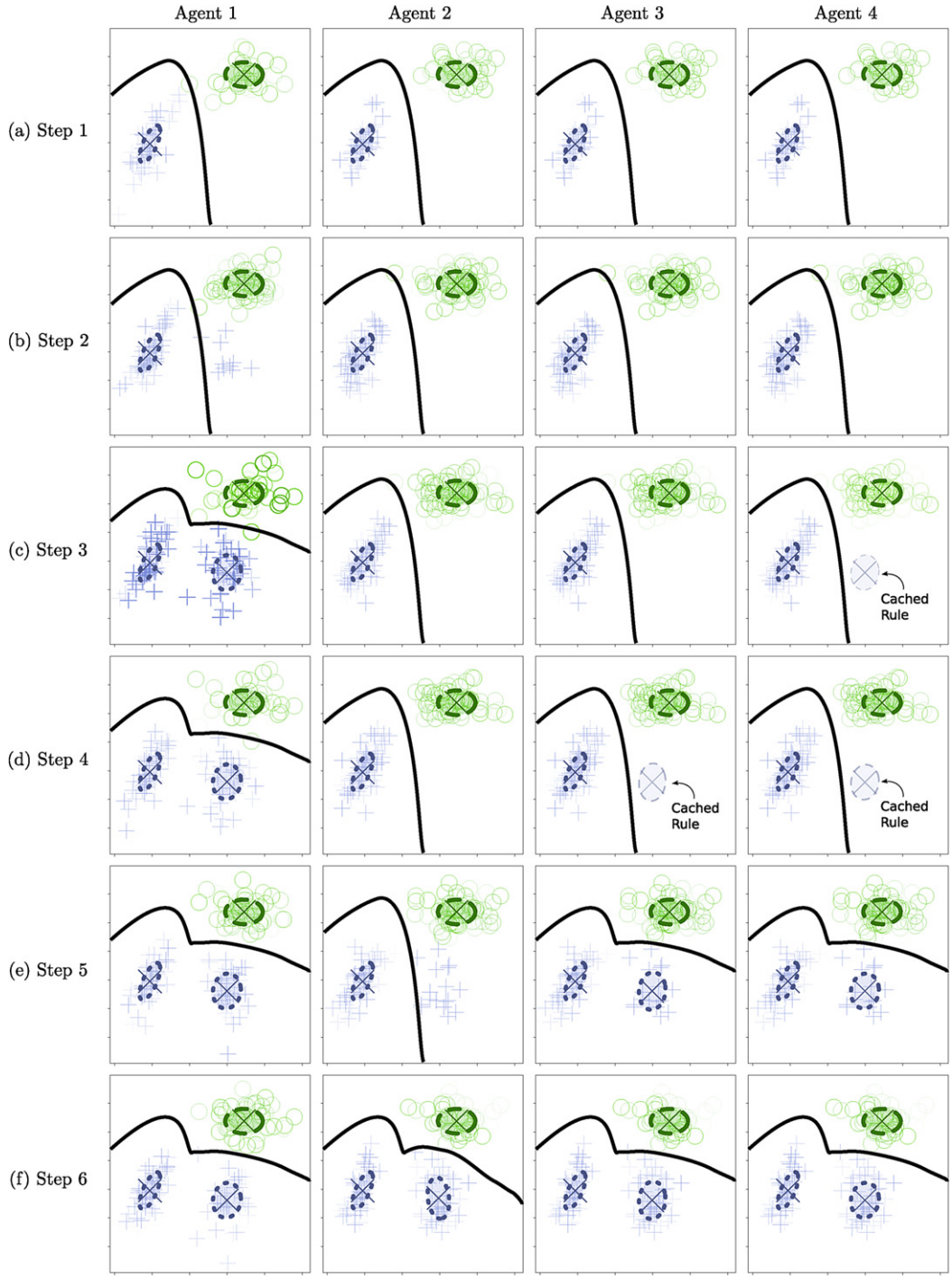


Fig. 5. Case Study 1: Comparison of different learning strategies: no exchange (agent 2), information exchange (agent 3), and knowledge exchange (agent 4).

- Agent 4 learns from agent 1 by using the rules learned and sent by agent 1. Like agent 3, agent 4 integrates a rule into its active classifier once it is regarded as being useful. This strategy has some characteristics of a "learning by teaching" approach. Essentially, it claims that peers, e.g., pupils or students, should improve their knowledge by teaching each other.

In this case study, we outline communications costs, classification errors, and expert invocation costs of the different learning strategies applied in the described dynamic scenario. The whole scenario (simulation over 1000 units) is illustrated in Fig. 5. There, the 2-dimensional input space of the agents is depicted including observed samples (circles and crosses),

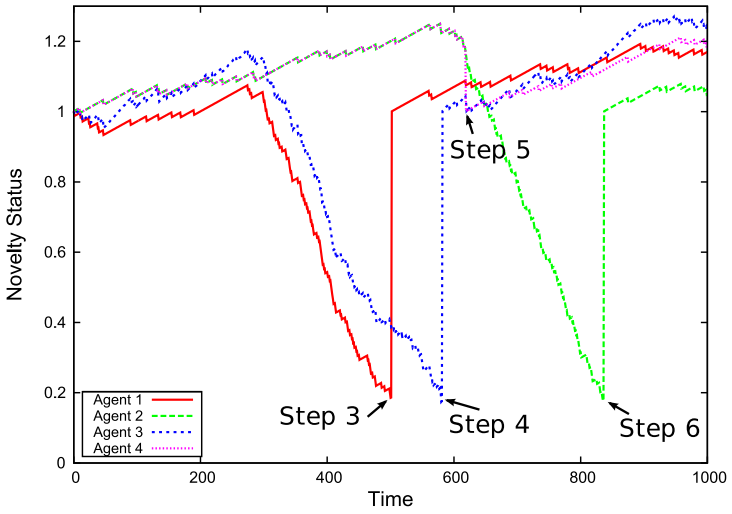


Fig. 6. Case Study 1: Novelty status of the four agents over time. The status value begins to decrease when novelty occurs in the environment of an agent. When novelty is actually detected, it is re-set to an initial value.

Table 1
Case Study 1: Cost analysis.

		Comm. costs	Expert queries
No exchange	Agent 1	0	1
	Agent 2	0	1
	Average	**0**	**1**
Information Exchange	Agent 1	2000	1
	Agent 3	0	1
	Average	**1000**	**1**
Knowledge Exchange	Agent 1	8	1
	Agent 4	0	0
	Average	**4**	**0.5**

premises of active rules (bold ellipses), and premises of cached rules (faint ellipses). The solid black line is an approximation of decision boundary which is implicitly defined by the rules. Only the locally observed samples are shown.

Initially (step 1), all four agents are equipped with the same classifier configuration, Fig. 5a. At step 2 (time 300), agent 1 observes novel samples in the lower right quadrant, Fig. 5b. As its classifier does not yet cover this region in the input space, each new sample therein decreases its novelty status s_{nov} (see Fig. 6, time range 300–500) and, thus, increases its need for self-adaptation. Between steps 2 and 3, s_{nov} sinks below the novelty threshold $\sigma = 0.2$ (cf. Section 3.2) and novelty handling is performed, Fig. 5c. As a consequence, s_{nov} of agent 1 returns to its initial value again (Fig. 6, time 500). Agents 2, 3, and 4 did not locally observe any novel samples yet. However, agent 1 broadcasts the learned rule and agent 4 puts it into its rule cache to assess its usefulness, Fig. 5c. The decision boundary of agent 4 is not affected so far. As agent 3 not only assesses its locally observed samples but additionally the samples sent by agent 1, its novelty status also sinks (Fig. 6, time range 300–550). However, due to the merging of local and remote samples, the fraction of novel samples is lower than that of agent 1 leading to a slower decay of s_{nov} for agent 3. At step 4, the novel samples sent by agent 1 cause the novelty status of agent 3 to fall below the threshold $\sigma = 0.2$ and, thus, lead to a new rule, Fig. 5d. The slightly different rule premise (ellipse) compared to the rule created by agent 1 is also due to the merging of samples. Now, agents 3 and 4 both assess the usefulness of the new rules. Between steps 4 and 5 (time 600), the second "blue" process also becomes visible to the remaining three agents and causes the usefulness of the cached rules to increase until they are integrated into the active classifier at step 5, Fig. 5e. For agent 2, these samples lead to a decrease of its novelty status s_{nov} (Fig. 6, time range 600–850) until it performs novelty handling at step 6, Fig. 5f.

The three exchange strategies yield different performance values with respect to communication costs, number of expert queries (cf. Table 1), and classification error (cf. Fig. 7). Communication costs are calculated based on the amount of data which is exchanged (cf. Section 3.1), i.e., one exchanged rule increases the communication costs of the sending agent by 8 units as 8 parameter values have to be sent. Transmitting a single, 2-dimensional sample costs 2 units. An expert query could be seen as being very expensive, because a human application expert is certainly a scarce resource in many applications.

- Exchanging no knowledge naturally exhibits no communication costs. However, both agents 1 and 2 have to query a human expert for a rule conclusion and both agents misclassify about the same amount of samples. The mean classification error of agent 1 is 0.07% and that of agent 2 is 0.08%.

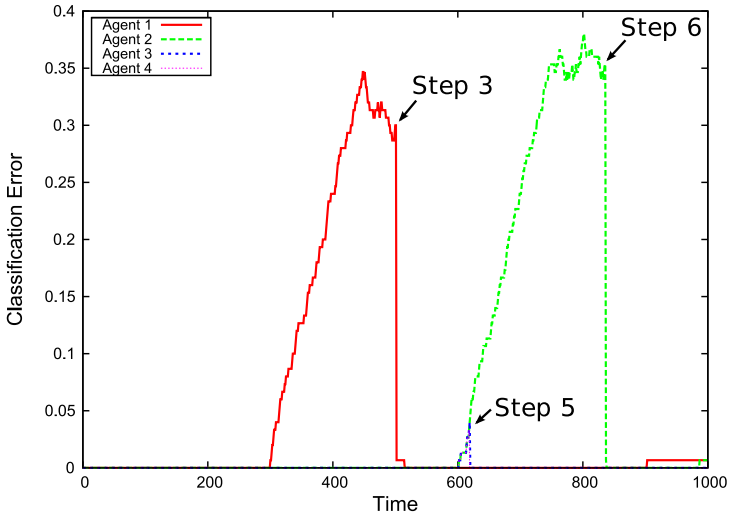


Fig. 7. Case Study 1: Classification error of the four agents over time. When novelty occurs, the classification error increases and with novelty reaction, it decreases to (nearly) zero again.

- With the information exchange strategy agent 1 sends 1000 observed samples to agent 3. Additionally, both agents 1 and 3 ask a human expert for a rule conclusion. As agent 3 receives the new rule prior to the emergence of the process in its input space, its mean classification error is significantly lower (i.e., 0.03%).
- In the case of knowledge exchange agent 1 sends two rules to agent 4. As the rule already contains a conclusion agent 4 refrains from querying the human expert. This strategy also yields a low classification error of 0.03% for agent 4.

Based on the results of this case study we can state that the collaboration strategies (i.e., the exchange of information and knowledge) are beneficial for the overall system performance if the agents have to cope with similar situations. The high communication costs of exchanging raw samples can be avoided by exchanging knowledge in form of learned rules. In this case, even the number of human expert invocations is reduced to a minimum.

4.2. Interestingness-based rule exchange

Humans not only gain and apply rules that they have learned by their own or acquired from other humans. They also gain experience with that rules, which could be seen as a kind of meta-knowledge, and they may also profit from experience of other humans. This biological archetype motivates the second case study. Here, several agents utilize the knowledge (rules) sent by another agent in various ways:

- Similar to the first case study, agent 1 is able to “share” its superior sensing ability with other agents by sending learned rules.
- Agent 2 has a “greedy” behavior by immediately integrating the received rules into the active classifier without assessing them.
- Agent 3 assesses the usefulness of a received rule, i.e., it measures how often the rule is covered by locally observed samples in order to decide when to integrate it.
- Agent 4 acts even more carefully and considers the informativeness of a rule in the decision process. That is, the more distant a new rule is with respect to existing rules, the faster it will be integrated.

The aim of the second case study is to investigate the effect of incorporating a certain kind of experience into the rule integration process.

The scenario consists of five (initially three) processes that generate normally distributed samples. However, the fifth process is *only* seen by agent 1 and not by agents 2, 3, and 4. Initially, all four agents are equipped with the same classifier configuration: One rule in the lower left quadrant of the input space representing a process that generates “green” samples (i.e., circles), and two rules on the right half for two “blue” processes (i.e., crosses), cf. Fig. 8. Again, the solid black line indicates the decision boundary of the rule set.

In the input space of agent 1 emerges a new process, generating “green” samples (i.e., circles) in the upper left quadrant. As a consequence, its novelty status decreases (cf. Fig. 11) until it sinks below the threshold $\sigma = 0.0$ leading to self-adaptation and, thus, to the creation of a new rule. Agent 1 immediately sends this new rule to agents 2, 3 and 4, Fig. 9a. Following different rule integration strategies, however, the three agents perform different actions. The greedy agent 2 immediately integrates the received rule into its active classifier. Agents 3 and 4 put the received rule under observation

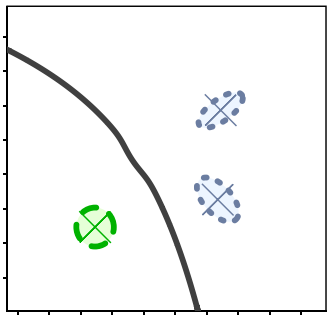


Fig. 8. Case Study 2: Initial classifier configuration of the four agents.

to assess their usefulness. Agent 3, relying only on the usefulness measure for rule integration, sets the initial usefulness measure to a fixed value of 0.7. Agent 4 additionally considers the informativeness of the rule and, thus, sets the initial value of the rule to a higher value (because it is "quite distant" to existing rules), see Fig. 10a. It also chooses a higher step size η which makes sure the usefulness of the rule will rise quickly even if only few samples emerge at the corresponding region of the input space leading to a fast integration of the rule. Between steps 1 and 2, the second green process also emerges for agents 2, 3 and 4 leading to an increase of the usefulness of the cached rule for agents 3 and 4 (Fig. 10a, from time step 700 on). At step 2, Fig. 9b, both agents integrated the cached rule into their active classifier. Note, however, that agent 4 integrated the rule prior to agent 3 (Fig. 10a, time step 725) due to the higher initial usefulness value and larger step size.

At step 3, a third green process emerges in the input space of agent 1 between the existing blue processes, Fig. 9c, that decreases its novelty status until it performs self-adaptation and creates a new rule at step 4, Fig. 9d. Remember that this fifth process is only seen by agent 1 and will never be seen by agents 2, 3 and 4. This fact is, of course, unknown to agent 1 and, thus, it sends the new rule to the remaining agents. Again, the greedy agent 2 immediately integrates the rule into its active classifier whereas agents 3 and 4 put the rule under observation to assess its usefulness, Fig. 10b. Agent 3 initializes the usefulness with its fixed value of 0.7. Agent 4 considers the informativeness of the rule and as it is close to an existing rule in the active classifier it decreases the initial value. It also decreases the step size η for this rule requiring more samples in a shorter time interval for it to be integrated.

As the new rule is close to rules for existing processes, some of their samples lead to an erroneous increase of the rule's usefulness for both agents, Fig. 10b. As a consequence, at step 5, agent 3 wrongly regards the rule as being useful enough and integrates it into its active classifier, Fig. 9e. The samples of the existing processes also lead to an increase of the rule's usefulness for agent 4, however, its adapted step size and initial usefulness value prevented the integration of the received rule which is deleted from the cache after 1000 time units without being useful (pre-defined time-out) at step 6, Fig. 9f.

The error agents 2 and 3 make by wrongly integrating the received rule can be assessed by looking at the classification error those agents make (cf. Fig. 12). Whereas agent 4 has a classification error of 0.0 in the time interval 1500 to 2500 the agents 2 and 3 have a classification error of about 0.05 from the point in time at which they integrate the received rule. If evaluating the mean classification error over the whole experiment agent 1 turns out to be the worst with an average classification error of 0.0878. It is followed by agent 2 with an average of 0.0143 and agent 3 with 0.0083. The best agent with regard to the mean classification error is agent 4 with 0.0006.

This case study shows that there is a trade-off concerning the rule integration strategy. An immediate integration of exchanged rules (i.e., a "greedy" strategy) may allow for pro-active behavior. That is, before certain situations emerge in the environment, agents are already enabled to deal with that situations. However, being too greedy may result in the integration of improper rules. It became clear that a combination of our proposed interestingness measures enables sophisticated rule acceptance strategies with two important properties. First, the risk of integrating improper rules that are not helpful can be reduced. Second, the integration of rules that are likely to be valuable can be accelerated.

5. Lessons learned

In our research in the field of knowledge exchange we gained some experience concerning the characteristics and the limitations of our proposed techniques that we want to summarize in this section.

In our earlier work focusing on knowledge exchange (cf. [1,2]) we did not address the importance of the generative approach. There, we used a discriminative classifier trained with penalty terms (i.e., regularization technique) to constrain its form and, thus, facilitate the exchange of knowledge between agents that are equally trained. The generative classifier presented in this work, however, inherently offers this possibility while having additional advantages such as optimization of the number of components and a faster training algorithm.

With regard to the generative properties we can state that if the distribution assumptions are—at least partially—met, the VI training algorithm is able to model the data very well. In this case, the classification performance is usually close to that of discriminatively trained classifiers such as RBF networks (for more details see [8,22]) or support vector machines.

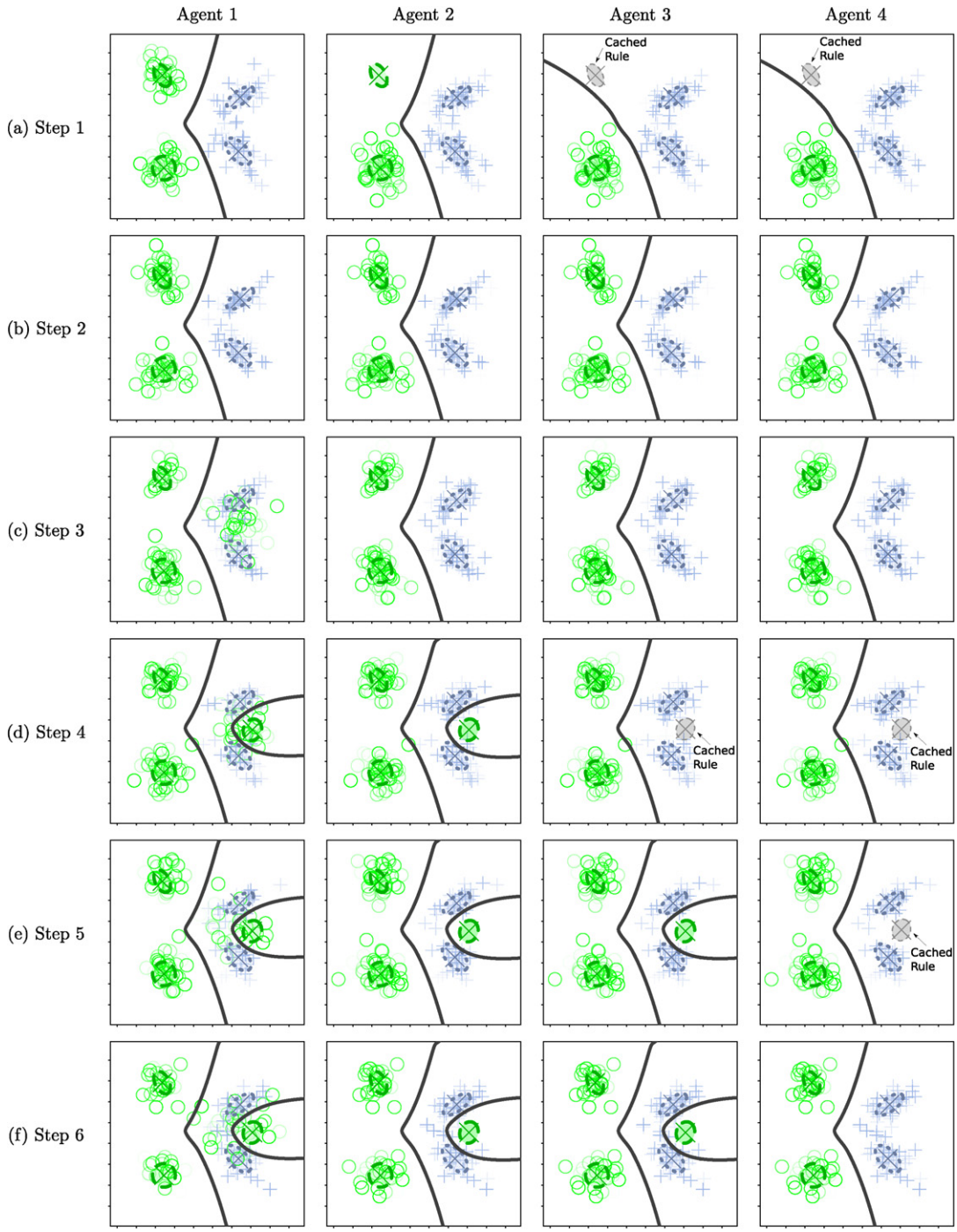
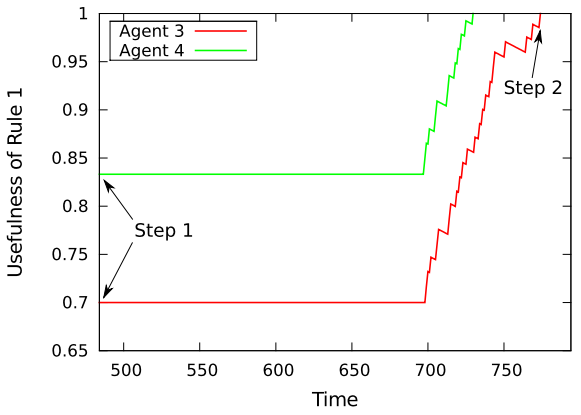


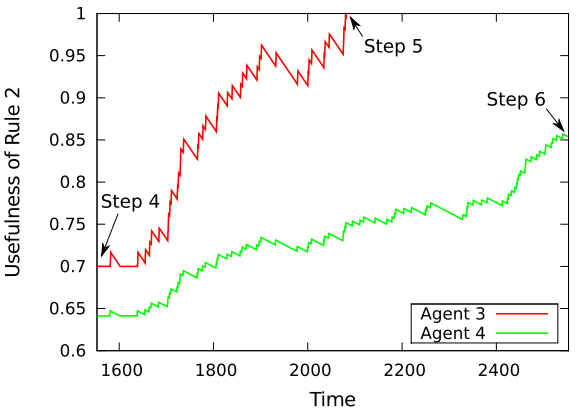
Fig. 9. Case Study 2: Interestingness-based rule exchange. Agent 1 detects novelty in it's environment, generates new rules and broadcasts them to the other agents. Agent 2 directly integrates received rules into it's classifier. Agent 3 first evaluates their usefulness and agent 4 additionally considers the informativeness to initialize the usefulness evaluation with appropriate values.

The VI algorithm is not parameter-free but it reacts not very sensitive to their setting [8]. Actually, the impact of hyper-parameters increases if the used training data set is small.

Our proposed techniques for novelty and obsoleteness detection are only suited for scenarios in which the underlying processes can be regarded as being time-invariant for time intervals of a minimal length. That is, after changes of the underlying processes (novelty or obsoleteness) we need a minimum of observed samples to detect that changes before any further changes occur. Although this critical minimum amount of samples can be controlled by the step size parameter η



(a) Usefulness of the first rule that was broadcast by agent 1 at timestep 484.



(b) Usefulness of the second rule that was broadcast by agent 1 at timestep 1553.

Fig. 10. Case Study 2: Usefulness of cached rules over time. Agent 4 adapts the initial value and the step size according to the interestingness of the received rule while agent 3 always uses the same values.

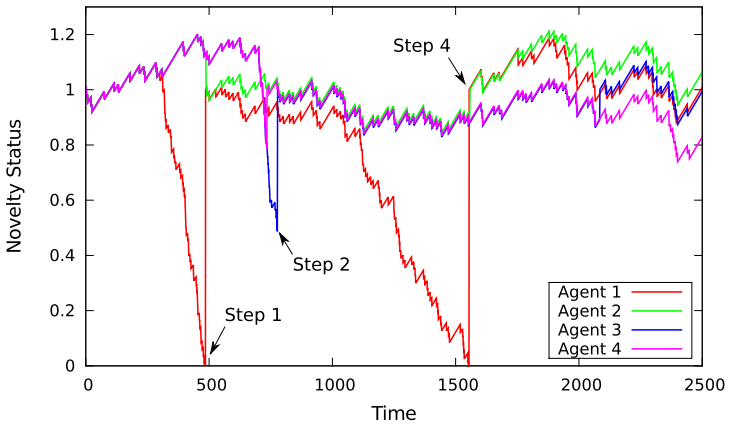


Fig. 11. Case Study 2: Novelty status of the four agents over time. The status value begins to decrease when novelty occurs in the environment of an agent. When a new rule is integrated into the classifier the novelty status is reset to an initial value.

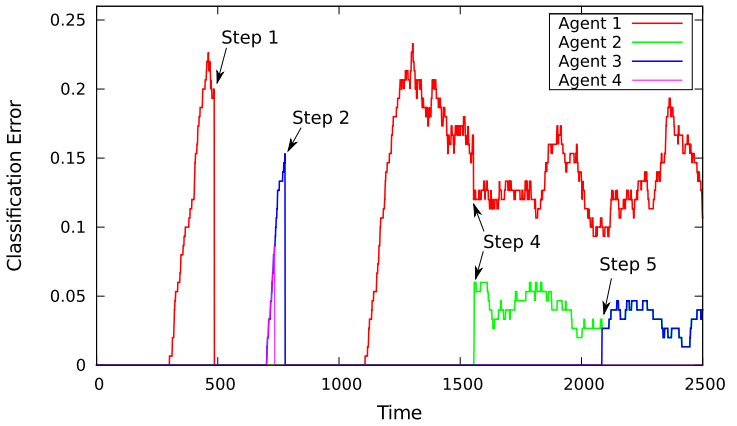


Fig. 12. Case Study 2: Classification error of the four agents over time. When novelty occurs in the environment of an agent the classification error rises until a new rule is integrated into the classifier to deal with the novel samples (e.g. steps 1 and 2). At steps 4 and 5 the agents 2 and 3 mistakenly integrate new rules that are not actually needed which causes an unnecessary rise in those agent's classification error.

of the novelty detection algorithm, there always is the trade-off between modeling accuracy (the more samples the better) and reaction time. Novelty detection and reaction with different techniques and the mentioned trade-off are investigated in some more detail in [44,45] (referred to as emergence detection and measurement there).

The knowledge exchange techniques presented here extend our work in [3] where we also discuss the properties of different rule integration strategies.

We also want to mention that the techniques presented in this work are rather generic. Using them in a real scenario (cf. Section 6) certainly requires application-specific modifications.

Finally, we want to make some comments on a very important aspect of knowledge exchange, the possible need of human domain experts for an assessment of rules. It should be emphasized that human integration is not needed to determine the number of classes in the GMM in advance, to parameterize the VI algorithm (good heuristics exist for that purpose), find component parameters in the case of novelty detection, for the random sampling technique used to adapt existing rule conclusions, etc. The only task for which in some cases—depending on the application—humans are integrated in the rule exchange process is the labeling of rules, i.e., the assignment of rule premises (that are learned by the system in an unsupervised way) to classes. There are, however, applications where even this task can be accomplished by the system itself. This is the case, for example, if the assignment of rule premises to classes is implicitly given by the positioning of the rule premise in the input space of the CMM. Examples for both cases will be given in the following section.

6. Application fields

Having introduced a novel methodology for the exchange of rules representing uncertain knowledge, we will now briefly sketch some possible real-world application scenarios in the fields of driver assistance systems, robotics, and information security. The latter will then be elaborated in some more detail.

6.1. Overview

Driver assistance systems are a key technology to improve traffic safety and lower the number of deadly accidents. Direct communication between cars will further enhance this field of driver safety. In the context of foresighted driving, the knowledge exchange techniques presented in this article could be used to build warning systems where cars do not only rely on their locally observed samples but also on the knowledge of other cars (cf., e.g., our work on collaborative intelligence of cars described in [46,47]).

Another potential application field of rule exchange techniques is robotics. Our vision is that intelligent robots will exchange knowledge concerning strategy detection, assessment, or selection, for instance. Basically, they may decide autonomously when, with whom, and what kind of knowledge they exchange. The goal is to improve the collaboration of robots by knowledge exchange.

Finally, distributed intrusion detection systems (DIDS, see [50]) may also benefit from our proposed techniques. DIDS consist of collaborating intrusion detection agents (IDA) that protect a computer network against attacks. Intrusion detection is a multi-level process and, thus, IDA are typically based on an architecture similar to the one we have described in [48,49] (cf., Fig. 13): The *sensor layer* provides the interface to the network and the host on which the agent resides. Sensors acquire raw data from both the network and the host, filter incoming data, and extract interesting and potentially valuable (e.g., statistical) information which is needed to construct an appropriate event. An even may, for example, summarize relevant information contained in header data of TCP packets. At the *detection layer*, different detectors, e.g., classifiers trained with machine learning techniques such as support vector machines or conventional rule-based systems assess these events and search for known attack signatures (misuse detection) and suspicious behavior (anomaly detection). In case of attack suspicion, they create alerts which are then forwarded to the *alert processing layer*. Alerts may also be produced by firewalls (FW) or the like. At the alert processing layer, the alert aggregation module has to combine alerts that are assumed to belong to a specific attack instance. Thus, so-called meta-alerts are generated. Meta-alerts are used or enhanced in various ways, e.g., scenario detection or decentralized alert correlation. An important task of the *reaction layer* is reporting.

Our techniques for rule exchange may be employed at the detection layer, for instance. There, IDA could exchange rules representing knowledge that is necessary to classify observed events as attacks or legitimate traffic. If they learn to cope with a new attack type by autonomously learning the rule premise and querying a human expert for the rule conclusion (i.e., what type of attack was detected), this knowledge may be distributed within the DIDS such that other IDA are enabled to behave pro-actively. In this case, rule premises as well as rule conclusions must be exchanged.

IDA may also exchange rules on the alert processing layer. There, a number of alerts originating from a single attack instance must be aggregated to a meta-alert. The goal is to get an estimate of the current attack situation. A meta-alert corresponds to an autonomously learned rule premise that describes a current attack instance. This could be sent to other IDA. Detecting novelty then corresponds to the start of a new attack instance and obsoleteness to the termination of an existing instance. Thus, the overall DIDS will be enabled to detect distributed attacks (decentralized alert correlation in Fig. 13). In this application, conclusions are implicitly given by a local security strategy. We have shown in [49] that the aggregation of intrusion alerts by means of probabilistic modeling with novelty and obsoleteness techniques is a very promising approach. The field of intrusion detection typically requires support for continuous as well as categorical dimensions. Thus, our approach must be extended accordingly.

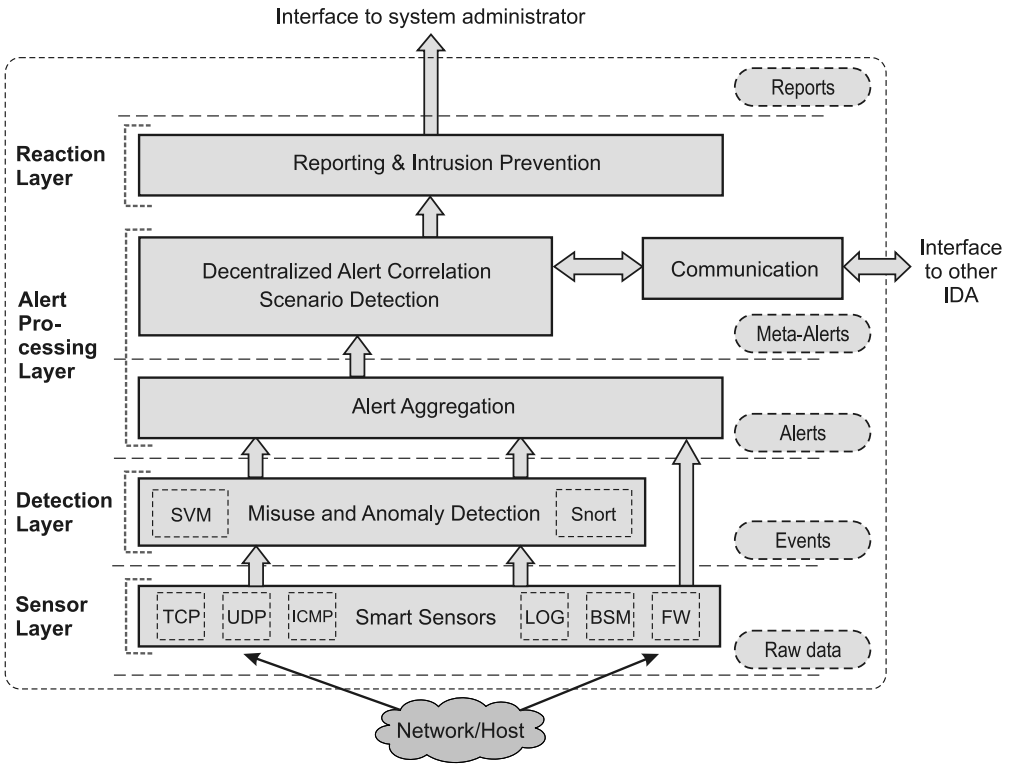


Fig. 13. Architecture of an Intrusion Detection Agent (IDA) [48,49].

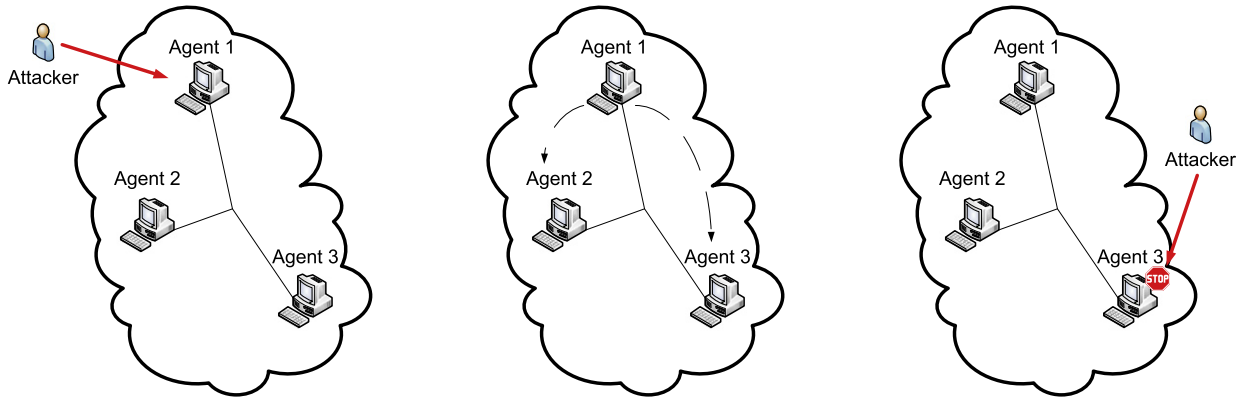
The intrusion detection scenario showed two conceptually different examples for knowledge exchange: In the first one (detection layer) humans are involved in this process. In the second one (alert processing layer), knowledge exchange is effected in a completely autonomous way without the support of humans. The reason is that we focus on the exchange of rule premises and conclusions are "added" locally by the different IDA themselves. Similar to this kind of knowledge exchange are the applications in the field of robotics and driver assistance systems that do not require the efforts of humans, too.

6.2. Knowledge exchange in a distributed intrusion detection system

The concept of IDA that collaborate by exchanging locally acquired detection knowledge is outlined in Fig. 14. There, three agents protect a computer network which is illustrated by a cloud symbol. In the first step, an attacker launches an attack against agent 1. Initially, this attack is unknown to all agents. Since agent 1 is situation-aware (with novelty detection and reaction capabilities as described in the previous sections) it is able to learn new detection rules for this attack. Then, it broadcasts this new knowledge to the two other agents. At some later point in time, agent 3, for instance, is targeted by the same kind of attack. As the required knowledge is already available, it is able to detect this attack immediately.

To demonstrate this concept with real data from the field of intrusion detection, we construct a scenario that is based on parts of the KDD-Cup'99 network intrusion data set. This data set contains about 5 million connection records (each with 34 continuous and 7 categorical attributes) of captured network traffic (i.e., both legitimate background traffic and attacks) [51]. From 41 available attributes we select 6 basic ones: *src_bytes* and *dst_bytes* (number of transmitted bytes from source to destination and vice versa), *count* and *srv_count* (number of connections in the last 2 seconds to the same host and the same service, respectively), *srv_serror_rate* and *dst_host_srv_diff_host_rate* (percentage of erroneous connections in the last 2 seconds to the same host and the same service, respectively). All of these attributes are numerical with large value ranges such that they can be handled as continuous variables.

As depicted in Fig. 14, in this scenario collaborate three agents. For every agent we create a data set that consists of 75 000 samples (here referred to as connection records) that contain both, legitimate background traffic (i.e., *normal* traffic) and attack traffic. We select four attack types: *Back*, *Ipsweep*, *Neptune*, and *Smurf*. Back is a Denial-of-Service (DoS) attack that is launched against Apache web servers. An attacker sends requests to the web server that contain many leading slashes. When the server processes these requests, it slows down significantly and is, therefore, unable to process any further requests. Ipsweep is a relatively simple tool for probing networks. It conducts network scans and is able to search hosts listening on a specific network. Neptune and Smurf are also DoS attacks that exploit vulnerabilities of the network protocol implementation of certain operating systems.



(a) Unknown attack launched against agent 1.

(b) Agent 1 distributes its new knowledge.

(c) Agent 3 immediately detects the attack.

Fig. 14. Collaborative intrusion detection agents.

Table 2
Attack schedule.

Agent 1		Agent 2		Agent 3	
Time	Traffic	Time	Traffic	Time	Traffic
0–10 000	Normal	0–15 000	Normal	0–20 000	Normal
10 001–22 000	Neptune	15 001–27 000	Ipsweep	20 001–32 000	Ipsweep
22 001–30 000	Normal	27 001–35 000	Normal	32 001–40 000	Normal
30 001–34 400	Back	35 001–47 000	Neptune	40 001–44 421	Back
34 401–55 000	Normal	47 001–60 000	Normal	44 422–50 000	Normal
55 001–67 000	Smurf	60 001–72 000	Smurf	50 001–62 000	Smurf
67 001–75 000	Normal	72 001–75 000	Normal	62 001–75 000	Normal

The agents have to classify the connection records and at every time step, each agent gets a new connection record from its data set as input. The structure of the data sets (i.e., the attack schedule) for this experiment is set out in Table 2. The entries with the attack names represent a mixture of records of the corresponding attack and background traffic (ratio 1:3).

It can be seen that the Neptune attack is first launched against agent 1 at time step 10 001 and later against agent 2 at time step 35 001. Thus, agent 2 could potentially benefit from the knowledge gained by agent 1. The same potential advantage exists for the other agents and different attack types. The attack schedule and the potential benefit of knowledge exchange is illustrated in Fig. 15. The first instance of a certain attack type is printed in bold font. The dashed arrows indicate beneficial propagation of newly acquired detection knowledge.

All agents are equipped with novelty detection and novelty reaction capabilities as described in Algorithms 2 and 3. Their active classifiers are initially trained with 5000 background traffic records. That is, when the experiment starts they can only classify normal traffic and not attack traffic. If novelty is detected, the agent adapts its active classifier based on a sliding window of the last 500 samples. Class assignments of new rules are provided by a simulated human domain expert, e.g., a system administrator. Here, the true sample labels are known from the data set. If a new rule is assigned to an attack, it is immediately sent to the other agents where it is placed in a rule cache for further usefulness evaluation. In this example, samples covered by the cached rule increase its usefulness (cf. Eq. (31)). Time has a decreasing influence on the usefulness status. If the usefulness status of a rule exceeds a threshold it is transfered to the active classifier (i.e., actually used) and removed from the cache. The mixture coefficient of the rule is set according to on the information of the sending agent.

The trajectories of the novelty status of all agents are depicted in Fig. 16. Agent 1 detects the start of the Neptune attack roughly at time step 10 300 and learns new rules which are sent immediately to the other two agents. This procedure is repeated for the Back attack around time step 30 400. The Smurf attack, in contrast, yields no significant decrease of the novelty status since agent 1 already possesses the required knowledge sent by agent 3 around time step 50 250. Both agents 2 and 3 only detect one time the need for new knowledge and, thus, benefit twice from the collaboration with the other agents.

The usefulness evaluation of received rules is demonstrated in Fig. 17 which shows the trajectories of the usefulness status of two rules. The Neptune rule (represented by the blue trajectory) from agent 1 is received by agent 2 at time step 10 298 and placed in its cache. During the course of the experiment some outlier samples are covered by that rule. These samples result in small spikes of its usefulness status which, however, immediately returns to the initial value. With the start of the Neptune attack at time step 35 001, more and more samples are covered by the rule yielding a rapid increase

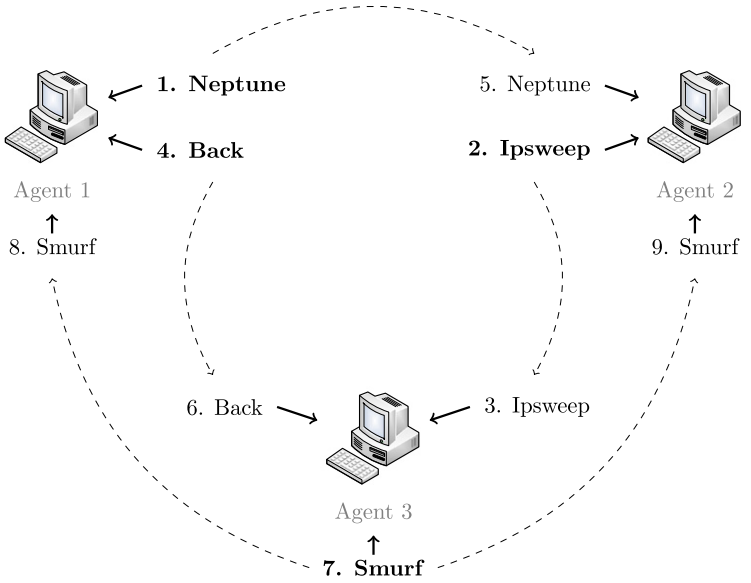


Fig. 15. Illustration of the attack schedule and beneficial knowledge exchange.

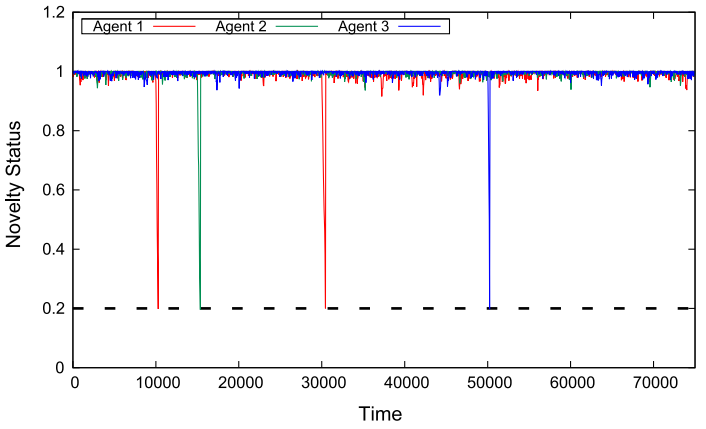


Fig. 16. Novelty status of the three agents.

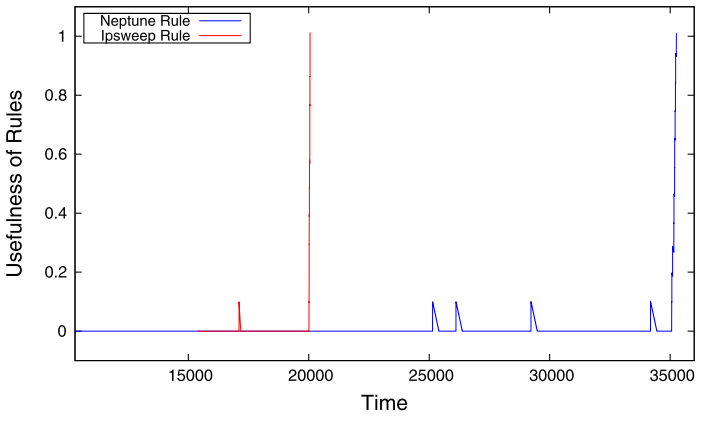


Fig. 17. Usefulness of cached rules.

Table 3
Agent 1: Classification results.

Attack	Without knowledge exchange			With knowledge exchange		
	Cor	FA	MA	Cor	FA	MA
Back	977 (88.8%)	10 (0.0%)	123 (11.2%)	977 (88.8%)	11 (0.0%)	123 (11.2%)
Neptune	2914 (97,1%)	5 (0.0%)	86 (2.9%)	2914 (97.1%)	6 (0.0%)	86 (2.9%)
Smurf	**2923** (97,4%)	3 (0.0%)	77 (2.6%)	**2986** (99.5%)	4 (0.0%)	14 (0.5%)

Table 4
Agent 2: Classification results.

Attack	Without knowledge exchange			With knowledge exchange		
	Cor	FA	MA	Cor	FA	MA
Ipsweep	2835 (94.5%)	56 (0.1%)	165 (5.5%)	2835 (94.5%)	77 (0.1%)	165 (5.5%)
Neptune	**2916** (97.2%)	8 (0.0%)	84 (2.8%)	**2979** (99.3%)	8 (0.0%)	21 (0.7%)
Smurf	**2915** (97.2%)	2 (0.0%)	85 (2.8%)	**2988** (99.6%)	4 (0.0%)	12 (0.4%)

Table 5
Agent 3: Classification results.

Attack	Without knowledge exchange			With knowledge exchange		
	Cor	FA	MA	Cor	FA	MA
Back	**984** (89.2%)	3 (0.0%)	119 (10.8%)	**1082** (98.1%)	9 (0.0%)	21 (1.9%)
Ipsweep	**2633** (87.8%)	8 (0.0%)	367 (12.2%)	**2894** (96.5%)	12 (0.0%)	106 (3.5%)
Smurf	2912 (97.1%)	0 (0.0%)	88 (2.9%)	2912 (97.1%)	0 (0.0%)	88 (2.9%)

of its usefulness status. Eventually, agent 1 integrates it in its active classifier. The same behavior can be observed for the Ipsweep rule that is received by agent 3 at time step 15 391 and integrated in the active classifier at time step 20 052.

To assess the effect of the collaboration we evaluated the classification performance of this experiment with and without knowledge exchange. Table 3 shows the number of correctly classified attack samples (Cor), the number of false alarms (FA) and the number of missing alarms (MA) of agent 1. In case of the attacks Back and Neptune, agent 1 is the first target and, thus, has to learn the corresponding detection knowledge in both scenarios on is own. Therefore, there is no significant difference in the classification performance. For the Smurf attack, however, agent 1 can benefit from the knowledge sent by agent 3 which improves its classification performance.

Similar results can be observed for the classification performance of agent 2 (cf. Table 4) and agent 3 (cf. Table 5). For both agents the exchange of knowledge improves the classification rates of two attacks (agent 2: Neptune and Smurf, agent 3: Back and Ipsweep).

Certainly, this particular example is not innovative from the viewpoint of intrusion detection as well-known rules for the detection of Back and Portsweep attacks already exist. Moreover, the classification performance could be further improved by selecting more sophisticated attributes of the connection records. However, this experiment shows the potential of collaborative learning as such and it demonstrates how agents can benefit from knowledge exchange.

This example also underlines the claim that collaborating agents are able to solve problems that they cannot solve alone. If intrusion detection agents are confronted with distributed attacks, they may exchange their locally gained knowledge in order to fuse that knowledge and to detect attacks they cannot recognize alone.

7. Conclusion and outlook

In this article we presented basic technologies for an exchange of classification rules that represent uncertain knowledge within distributed, intelligent systems. We introduced novel methods for knowledge acquisition in dynamic environments based on a probabilistic modeling approach. Our proposed techniques are able to detect the need for new rules and to adapt accordingly in a self-optimizing manner. New rules are either learned in a Bayesian approach or taken over from other intelligent systems. For the latter case we developed methods for gathering and using meta-knowledge about rules (i.e., interestingness) to improve the rule exchange process. In two case studies we demonstrated the properties of the proposed techniques and we also investigated a real application scenario. The proposed techniques lead to a collective intelligence of the overall distributed system.

It became clear that our proposed approach for modeling knowledge allows for a kind of self-awareness, i.e., a system is able to assess its own abilities and “knows what it knows”. These techniques can be used in the field of organic computing [52] to develop systems that interact and learn from each other.

A number of very interesting research issues, however, are still open:

- More sophisticated communication mechanisms than broadcasting should be investigated. Agents could actively query other agents for new rules by means of peer-2-peer methods such as distributed hash tables.

- Experiments with large-scale agent systems should be conducted to investigate positive and negative effects of knowledge exchange. In this context, it would also be interesting to introduce trust and reputation mechanisms.
- Scenarios must be investigated where the input spaces of classifiers differ, e.g., because different subsets of sensor sets are used (overlapping input spaces) or because physically different types of sensors are used to make basically the same observations (e.g., distance measurement with laser or radar sensors).
- The advantages of our Bayesian parameter estimation approach must be fully exploited. A Bayesian approach for knowledge fusion must be developed (cf. multimodal sensor interpretation and related approaches [53,54]). The fusion of rules should be straightforward (similar to the integration of prior knowledge), if hyper-distributions are exchanged between agents.
- Uncertainty regarding the parameterization of rules should be considered in the knowledge exchange process, too. In our case, uncertainty is basically expressed by the variance of the hyper-distributions. This idea is similar to the concept of confidence as mentioned in [28].

Other important issues have been addressed meanwhile: The classifier with its probabilistic model is extended by multinomial distributions that allow for categorical input dimensions. Also, additional interestingness measures have been introduced to consider even more aspects of meta-knowledge about exchanged rules. Our work in the field of active learning [55] is also influenced by the techniques presented here.

Acknowledgements

This work was supported by the German Research Foundation (DFG) under grants SI 674/3-2 and SI 674/3-3 within the priority program Organic Computing.

References

[1] O. Buchtala, B. Sick, Functional knowledge exchange within an intelligent distributed system, in: P. Lukowicz, L. Thiele, G. Tröster (Eds.), Architecture of Computing Systems—ARCS 2007, Proceedings of the 20th International Conference on Architecture of Computing Systems—System Aspects in Pervasive and Organic Computing, Zurich, Switzerland, in: LNCS, vol. 4415, Springer, Berlin, Heidelberg, New York, 2007, pp. 126–141.
[2] O. Buchtala, B. Sick, Basic technologies for knowledge transfer in intelligent systems, in: Proceedings of the First IEEE Symposium on Artificial Life (IEEE—ALife'07), Honolulu, HI, 2007, pp. 251–258.
[3] D. Fisch, M. Jänicke, E. Kalkowski, B. Sick, Learning by teaching versus learning by doing: Knowledge exchange in organic agent systems, in: Proceedings of the IEEE Symposium on Intelligent Agents (IA 2009), Nashville, TN, 2009, pp. 31–38.
[4] J. Halpern, Reasoning About Uncertainty, MIT Press, Cambridge, MA, 2003.
[5] A. Motro, P. Smets (Eds.), Uncertainty Management in Information Systems—From Needs to Solutions, Springer-Verlag, London, UK, 1997.
[6] S. Parsons, A. Hunter, A review of uncertainty handling formalisms, in: A. Hunter, S. Parsons (Eds.), Applications of Uncertainty Formalisms, in: Lecture Notes in Computer Science, vol. 1455, Springer, Berlin/Heidelberg, 1998, pp. 8–37.
[7] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU): An outline, Information Sciences 172 (1–2) (2005) 1–40.
[8] D. Fisch, B. Sick, Training of radial basis function classifiers with resilient propagation and variational Bayesian inference, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2009), Atlanta, GA, 2009, pp. 838–847.
[9] L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art, Autonomous Agents and Multi-Agent Systems 11 (2005) 387–434.
[10] V.M. Bove Jr., J. Mallett, Collaborative knowledge building by smart sensors, BT Technology Journal 22 (4) (2004) 45–51.
[11] V. Cantoni, L. Lombardi, P. Lombardi, Future scenarios of parallel computing: Distributed sensor networks, Journal of Visual Languages and Computing 18 (2007) 484–491.
[12] A. Paschke, H. Boley, A. Kozlenkov, B. Craig, Rule responder: RuleML-based agents for distributed collaboration on the pragmatic web, in: ICPW '07: Proceedings of the 2nd International Conference on Pragmatic Web, Tilburg, Netherlands, 2007, pp. 17–28.
[13] J. Carbo, A. Orfila, A. Ribagorda, Adaptive agents applied to intrusion detection, in: 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic, Lecture Notes in Computer Science, vol. 2691, 2003, pp. 445–453.
[14] A. Abraham, R. Jain, J. Thomas, S.Y. Han, D-scids: Distributed soft computing intrusion detection system, Journal of Network and Computer Applications 30 (2007) 81–98.
[15] M. Jakob, J. Tozicka, M. Pechoucek, Collaborative learning with logic-based models, in: K. Tuyls, A. Nowe, Z. Guessoum, D. Kudenko (Eds.), Adaptive Agents and Multi-Agents Systems, Lecture Notes in Computer Science, vol. 4865, Springer, 2008, pp. 102–116.
[16] S. Costantini, A. Tocchio, Learning by knowledge exchange in logical agents, in: 6th AI*IA/TABOO Joint Workshop "From Objects to Agents", Camerino, Italy, 2005, pp. 1–8.
[17] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: Readings in Agents, Morgan Kaufmann, 1997, pp. 487–494.
[18] I.D. Kelly, D.A. Keating, Faster learning of control parameters through sharing experiences of autonomous mobile robots, International Journal of Systems Science 27 (7) (1998) 783–793.
[19] K. Takadama, T. Terano, K. Shimohara, Learning classifier systems meet multiagent environments, in: IWLCS '00: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems, Springer, Berlin/Heidelberg, Germany, 2001, pp. 192–210.
[20] M. Jändel, Cooperating classifiers, in: N. Krasnogor, M. Melián-Batista, J. Pérez, J. Moreno-Vega, D. Pelta (Eds.), Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), in: Studies in Computational Intelligence, vol. 236, Springer, Berlin/Heidelberg, Germany, 2009, pp. 213–225.
[21] C. Gifford, A. Agah, Sharing in teams of heterogeneous, Collaborative Learning Agents 24 (2) (2009) 173–200.
[22] D. Fisch, B. Kühbeck, B. Sick, S. Ovaska, So near and yet so far: New insight into properties of some well-known classifier paradigms, Information Sciences 180 (2010) 3381–3401.
[23] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, Chichester, New York, NY, 2001.
[24] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, NY, 2006.
[25] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society B 39 (1) (1977) 1–38.
[26] G.J. McLachlan, T. Krishnan, The EM Algorithm and Extensions, Wiley, New York, NY, 1997.

[27] T.S. Jaakkola, Tutorial on variational approximation methods, in: M. Opper, D. Saad (Eds.), Advanced Mean Field Methods: Theory and Practice, MIT Press, Cambridge, MA, 2001, pp. 129–160.
[28] B. Bahrami, K. Olsen, P.E. Latham, A. Roepstorff, G. Rees, C.D. Frith, Optimally interacting minds, Science 329 (August 2010) 1081–1085.
[29] M. Markou, S. Singh, Novelty detection: A review—Part 1: Statistical approaches, Signal Processing 83 (12) (2003) 2481–2497.
[30] M. Markou, S. Singh, Novelty detection: A review—Part 2: Neural network based approaches, Signal Processing 83 (12) (2003) 2499–2521.
[31] M. Markou, S. Singh, An approach to novelty detection applied to the classification of image regions, IEEE Transactions on Knowledge and Data Engineering 16 (4) (2004) 396–407.
[32] S. Cateni, V. Colla, M. Vannucci, Outlier detection methods for industrial applications, in: J. Aramburo, A.R. Trevino (Eds.), Advances in Robotics, Automation and Control, IN-TECH, Vienna, 2008, pp. 265–282.
[33] S.J. Hickinbotham, J. Austin, Neural networks for novelty detection in airframe strain data, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN), Como, Italy, vol. 6, 2000, pp. 375–380.
[34] M. Lauer, A mixture approach to novelty detection using training data with outliers, in: L.D. Raedt, P. Flach (Eds.), Proceedings of the Twelfth European Conference on Machine Learning (EMCL), Freiburg, Germany, 2001, pp. 300–311.
[35] A. Flexer, E. Pampalk, G. Widmer, Novelty detection based on spectral similarity of songs, in: Proceedings of 6th International Conference on Music Information Retrieval (ISMIR), London, UK, 2005, pp. 260–263.
[36] D. Clifton, P. Bannister, L. Tarassenko, A framework for novelty detection in jet engine vibration data, Key Engineering Materials 347 (2007) 305–312.
[37] C. Weiss, A. Zell, Novelty detection and online learning for vibration-based terrain classification, in: Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS), Baden Baden, Germany, 2008, pp. 16–25.
[38] O. Buchtala, P. Neumann, 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 (IJCNN 2003), Portland, OR, vol. 2, 2003, pp. 1025–1030.
[39] O. Buchtala, A. Hofmann, B. Sick, Fast and efficient training of RBF networks, in: O. Kaynak, E. Alpaydin, E. Oja, L. Xu (Eds.), Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003 (Joint International Conference), Istanbul, Turkey, in: LNCS, vol. 2714, Springer-Verlag, Berlin, Heidelberg, New York, 2003, pp. 43–51.
[40] D. Andrade, T. Horeis, B. Sick, Knowledge fusion using Dempster–Shafer theory and the imprecise Dirichlet model, in: Proceedings of the IEEE Conference on Soft Computing in Industrial Applications (SMCia), Muroran, Japan, 2008, pp. 142–148.
[41] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Knowledge discovery and data mining: Towards a unifying framework, in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, 1996, pp. 82–88.
[42] K. McGarry, A survey of interestingness measures for knowledge discovery, Knowledge Engineering Review 20 (1) (2005) 39–61.
[43] D. Fisch, E. Kalkowski, B. Sick, In your interest: Objective interestingness measures for a generative classifier, in: Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011), Rome, Italy, 2011, pp. 414–423.
[44] D. Fisch, M. Jänicke, B. Sick, C. Müller-Schloer, Quantitative emergence—a refined approach based on divergence measures, in: Proceedings of the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010), Budapest, Hungary, 2010, pp. 94–103.
[45] D. Fisch, M. Jänicke, C. Müller-Schloer, B. Sick, Divergence measures as a generalised approach to quantitative emergence, in: C. Müller-Schloer, H. Schmeck, T. Ungerer (Eds.), Organic Computing—A Paradigm Shift for Complex Systems, Autonomous Computing, Birkhäuser-Verlag, Basel, Switzerland, 2011, pp. 53–66 (Ch. 1.3).
[46] T. Schön, B. Sick, M. Strassberger, Hazard situation prediction using spatially and temporally distributed vehicle sensor information, in: Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007), Honolulu, HI, 2007, pp. 261–268.
[47] M. Reiss, B. Sick, M. Strassberger, Collaborative situation-awareness in vehicles by means of spatio-temporal information fusion with probabilistic networks, in: Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals/06), Logan, UT, 2006, pp. 189–194.
[48] O. Buchtala, W. Grass, A. Hofmann, B. Sick, A fusion-based intrusion detection architecture with organic behavior, in: The First CRIS International Workshop on Critical Information Infrastructures (CIIW), Linköping, Sweden, 2005, pp. 47–56.
[49] A. Hofmann, B. Sick, On-line intrusion alert aggregation with generative data stream modeling, IEEE Transactions on Dependable & Secure Computing 8 (2009) 282–294.
[50] M. Treaster, A survey of distributed intrusion detection approaches, Tech. Rep. CR/0501001, National Center for Supercomputing Applications (NCSA), Champaign, IL, 2005.
[51] S. Hettich, S.D. Bay, The UCI KDD archive, http://kdd.ics.uci.edu, 1999.
[52] R.P. Würtz (Ed.), Organic Computing, Understanding Complex Systems, Springer, Berlin/Heidelberg, Germany, 2008.
[53] K.P. Körding, D.M. Wolpert, Bayesian integration in sensorimotor learning, Nature 427 (January 2004) 244–247.
[54] M.O. Ernst, A Bayesian view on multimodal cue integration, in: G. Knoblich, I.N. Thornton, M. Grosjean, M. Shiffrar (Eds.), Human Body Perception from the Inside Out, Oxford University Press, NY, USA, 2006, pp. 105–131.
[55] T. Reitmaier, B. Sick, Active classifier training with the 3DS strategy, in: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, 2011, pp. 88–95.