Machine Learning and Agents

Piotr Jędrzejowicz

Chair of Information Systems, Gdynia Maritime University,
Morska 83, 81-225 Gdynia, Poland
pj@am.gdynia.pl

Abstract. The paper reviews current research results integrating machine learning and agent technologies. Although complementary solutions from both fields are discussed the focus is on using agent technology in the field of machine learning with a particular interest on applying agent-based solutions to supervised learning. The paper contains a short review of applications, in which machine learning methods have been used to support agent learning capabilities. This is followed by a corresponding review of machine learning methods and tools in which agent technology plays an important role. Final part gives a more detailed description of some example machine learning models and solutions where the asynchronous team of agents paradigm has been implemented to support the machine learning methods and which have been developed by the author and his research group.

1 Introduction

Contemporary definition sees machine learning as a discipline that is concerned with the design and development of algorithms that allow computers to learn behaviors based on empirical data. Data can be seen as examples that illustrate relations between observed objects. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data. Parallel to recent developments in the field of machine learning, mainly as a result of convergence of many technologies within computer science such as object-oriented programming, distributed computing and artificial life, the agent technology has emerged. An agent is understood here as any piece of software that is designed to use intelligence to automatically carry out an assigned task, mainly retrieving and delivering information. Tweedale and co-authors [62] outline an abridged history of agents as a guide for the reader to understand the trends and directions of future agent design. This description includes how agent technologies have developed using increasingly sophisticated techniques. It also indicates the transition of formal programming languages into object-oriented programming and how this transition facilitated a corresponding shift from scripted agents (bots) to agent-oriented designs which is best exemplified by multiple agent systems (MAS). A MAS tries to solve complex problems with entities called agents, using their collaborative and autonomous properties [38]. Basic MAS properties are listed in [34].

J. O'Shea et al. (Eds.): KES-AMSTA 2011, LNAI 6682, pp. 2–15, 2011.

During the last decade developments in the fields of machine learning and agent technologies have, in some respect, become complementary and researchers from both fields have seen ample opportunities to profit from solutions proposed by each other. Several agent-based frameworks that utilize machine learning for intelligent decision support have been recently reported. Learning is increasingly being seen as a key ability of agents, and research into learning agent technology, such as reinforcement learning and supervised or unsupervised learning has produced many valuable applications. In this paper the focus is however on using agent technology in the field of machine learning with a particular interest on applying agent-based solutions to supervised learning. Supervised learning is the machine learning task of inducing a function from training data which is a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal or class label). A supervised learning algorithm analyzes the training data and produces an induced function, which is called a classifier if the output is discrete, or a regression function if the output is continuous. The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations.

There are several ways the machine learning algorithm can profit from applying agent technology. Among them the following will be addressed in this paper:

- There are numerous machine learning techniques where parallelization can speed-up or even enable learning. Using a set of agents may, in such circumstances, increase efficiency of learning.
- Several machine learning techniques directly rely on the collective computational intelligence paradigm, where a synergetic effect is expected from combining efforts of various program agents.
- There is a class of machine learning problems known as the distributed machine learning. In the distributed learning a set of agents working in the distributed sites can be used to produce some local level solutions independently and in parallel. Later on local level solutions are combined into a global solution.

The paper is organized as follows. Section 2 contains a short review of applications, in which machine learning methods have been used to support agent learning capabilities. Section 3 offers a corresponding review of machine learning methods and tools in which agent technology plays an important role. Section 4 gives more detailed description of some example machine learning models and solutions where the agent paradigm has been implemented and which have been developed by the author and his research group. Finally, conclusions contain suggestions on future research and possible deeper integration of machine learning and agent technology.

2 Learning Agents

Probably the most often used approach to provide agents with learning capabilities is the reinforcement learning. An excellent survey of multiagent reinforcement learning can be found in [10]. As it was pointed out by Sutton and Barto [57] reinforcement learning is learning what to do - how to map sitations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward. In describing properties of the reinforcement learning the authors directly refer to the notion of agent. In their view a learning agent must be able to sense the state of the environment and must be able to take actions that affect the state. The agent also must have goal or goals relating to the state of the environment [57].

Theoretical developments in the field of learning agents focus mostly on methodologies and requirements for constructing multiagent systems with learning capabilities. Connection of the theory of automata with the multiagent reinforcement learning is explored in [45]. Shoham et.al. [53] claim that the area of learning in multi-agent systems is today one of the most fertile grounds for interaction between game theory and artificial intelligence. In [41] challenges motivated by engineering applications and the potential appeal of multi-agent learning to meet these challenges are discussed.

Symeonidis et.al. [58] present an approach that takes the relevant limitations and considerations into account and provides a gateway on the way data mining techniques can be employed in order to augment agent intelligence. This work demonstrates how the extracted knowledge can be used for the formulation initially, and the improvement, in the long run, of agent reasoning. Preux et.al. [47] present MAABAC, a generic model for building adaptive agents: they learn new behaviors by interacting with their environment. Agents adapt their behavior by way of reinforcement learning, namely temporal difference methods. The paper [52] presents a systematic approach to introduce machine learning in the design and implementation phases of a software agent. It also presents an incremental implementation process for building asynchronous and distributed agents, which supports the combination of machine learning strategies. Rosaci in [51] proposes a complete MAS architecture, called connectionist learning and inter-ontology similarities (CILIOS), for supporting agent mutual monitoring.

In [42] the concepts of stigmergy and entropy are imported into learning automata based multi-agent systems with the purpose of providing a simple framework for interaction and coordination in multi-agent systems and speeding up the learning process. Another extension was proposed in [8]. The authors suggest a merging, and hence an extension, of two recent learning methods, utility-based learning and strategic or adversarial learning. Utility-based learning brings to the forefront the learner's utility function during induction. Strategic learning anticipates strategic activity in the induction process when the instances are intelligent agents such as in classification problems involving people or organizations. The resulting merged model is called the principal-agent learning. Loizos [39] argues that when sensing its environment, an agent often receives

information that only partially describes the current state of affairs. The agent then attempts to predict what it has not sensed, by using other pieces of information available through its sensors. Machine learning techniques can naturally aid this task, by providing the agent with the rules to be used for making these predictions. For this to happen, however, learning algorithms need to be developed that can deal with missing information in the learning examples in a principled manner, and without the need for external supervision. It is shown that the Probably Approximately Correct semantics can be extended to deal with missing information during both the learning and the evaluation phase.

Numerous reinforcement learning applications have been recently reported in the literature. Some interesting examples include a proposal of reinforcement learning for agent-based production scheduling [63], a case-based reinforcement learning algorithm (CRL) for dynamic inventory control in a multi-agent supply-chain system [35]. Supervised learning techniques have been also applied to support agents learning capabilities. In [65], a support vector machine (SVM) based multiagent ensemble learning approach is proposed for credit risk evaluation. Different SVM learning paradigms with much dissimilarity are constructed as intelligent agents for credit risk evaluation. Multiple individual SVM agents are trained using training subsets. In the final stage, all individual results produced by multiple SVM agents in the previous stage are aggregated into an ensemble result.

3 Agent-Based Machine Learning

Recently, several machine learning solutions and techniques have been reported to rely on applying agent technologies. They belong to the two broad classes - universal one and dedicated to particular applications. Solutions and techniques belonging to the first class involve applications of the multi agent systems, including A-Teams and the population-based methods. This section contains a review of some recent universal and dedicated solutions with the exception of those based on the A-Team paradigm. Machine learning solutions using the A-Team paradigm are discussed in a detailed manner in Section 4.

3.1 Universal Solutions and Techniques

As it has been observed in [40] industry, science, and commerce fields often need to analyze very large datasets maintained over geographically distributed sites by using the computational power of distributed systems. The Grid can play a significant role in providing an effective computational infrastructure support for this kind of data mining. Similarly, the advent of multi-agent systems has brought us a new paradigm for the development of complex distributed applications. Through a combination of these two techniques an Agent Grid Intelligent Platform and an integrated toolkit VAStudio used as a testbed were proposed. Using grid platform as a testbed was also suggested in [50]. The author presents a parallel learning method for agents with an actor-critic architecture based on

artificial neural networks. The agents have multiple modules, where the modules can learn in parallel to further increase learning speed. Each module solves a subproblem and receives its own separate reward signal with all modules trained concurrently. The method is used on a grid world navigation task showing that parallel learning can significantly reduce learning time.

Kitakoshi et al.[36] describe an on-line reinforcement learning system that adapts to environmental changes using a mixture of Bayesian networks. Machine learning approaches, such as those using reinforcement learning methods and stochastic models, have been used to acquire behavior appropriate to environments characterized by uncertainty. The results of several experiments demonstrated that an agent using the proposed system can flexibly adapt to various kinds of environmental changes.

Gifford in his Ph.D. dissertation [24] advocates an approach focused on the effects of sharing knowledge and collaboration of multiple heterogeneous, intelligent agents (hardware or software) which work together to learn a task. As each agent employs a different machine learning technique, the system consists of multiple knowledge sources and their respective heterogeneous knowledge representations. Experiments have been performed that vary the team composition in terms of machine learning algorithms and learning strategies employed by the agents. General findings from these experiments suggest that constructing a team of classifiers using a heterogeneous mixture of homogeneous teams is preferred.

Quteishat et.al. [49] proposed a neural network-based multi-agent classifier system using the trust, negotiation, and communication reasoning model. The main contribution of this work is that a novel trust measurement method, based on the recognition and rejection rates, was suggested.

Several important methods can be grouped under the umbrella of the collective or collaborative learning. In [27] it was shown show how Evolutionary Dynamics (ED) can be used as a model for Qlearning in stochastic games. Analysis of the evolutionary stable strategies and attractors of the derived ED from the Reinforcement Learning (RL) application then predict the desired parameters for RL in Multi-Agent Systems (MASs) to achieve Nash equilibriums with high utility. Secondly, it was shown how the derived fine tuning of parameter settings from the ED can support application of the COllective INtelligence (COIN) framework. COIN is a proved engineering approach for learning of cooperative tasks in MASs. In [26] authors propose a collaborative machine learning framework to exploit inter-user similarities. More specifically, they present a kernel-based learning architecture that generalizes the well-known Support Vector Machine learning approach by enriching content descriptors with inter-user correlations.

Another umbrella covers learning classifier systems introduced by Holland [28] which use simple agents representing set of rules as a solution to a machine learning problem. A Pittsburgh-type LCS has a populations of separate rule sets, where the genetic algorithm recombines and reproduces the best of these rule sets. In a Michigan-style LCS there is only a single population and

the algorithm's action focuses on selecting the best classifiers within that rule set. Analysis of the properties of LCSs, comparison of several proposed variants and overview of the state of the art can be found in [4], [5], [64] and [9]. Useful extension of the LCS concept was proposed in [55]. This paper introduces a new variety of learning classifier system (LCS), called MILCS, which utilizes mutual information as fitness feedback. Unlike most LCSs, MILCS is specifically designed for supervised learning. Yet another extension introduces a mechanism for recognizing a current situation by determining a boundary between self and others, and investigates its capability through interaction with an agent [59]. An integration of several cognitively inspired anticipation and anticipatory learning mechanisms in an autonomous agent architecture, the Learning Intelligent Distribution Agent (LIDA) system was proposed in [44].

Ensemble techniques have proved to be very successful in boosting the performance of several types of machine learning methods. In [6] authors illustrate its usefulness in combination with GAssist, a Pittsburgh-style Learning Classifier System. Effective and competitive ensembles constructed from simple agents represented by expression trees induced using Gene Expression Programming have been proposed in [32] and [33]. Their approach has been tested using several ensemble constructing techniques including AdaBoost learning, voting pool of classifiers, incremental learning, cluster based learning, mass functions based learning and meta-learning.

Agent technology seems to be a natural tool for the distributed systems. Combining approaches to distributed learning with agent technology is considered as the promising and at the same time challenging problem in the distributed learning research [37]. In [67] an agent paradigm was proposed as a tool for integration of different techniques into an effective strategy of learning from data. The proposed hybrid learning system integrates basic components of the learning process. Data pre-processing, selection, transformation and induction of the learning and post-learning models are carried out by a set of agents cooperating during the task execution. Several agent-based architectures have already been proposed to solve the distributed learning problems. It is usually assumed that each site can have one or more associated agents, processing the local data and communicating the results to other agents that control and manage the knowledge discovery process. Examples include Papyrus [48], MALE [54], ANIMALS [56] and MALEF [61]. In [2] EMADS, a hybrid peer-to-peer agent based system comprising a collection of the collaborating agents distributed across a network, was described.

3.2 Dedicated Solutions and Techniques

In the machine learning literature numerous applications solving particular machine learning problem type or task where agent technology have played an important, even if supporting, role have been recently reported. In this short review the focus is on some example cases where agent technology has been used in an innovative manner.

Fan et.al. [23] have developed a two-stage model for personalized and intelligent information routing of online news. At the first stage, persistent user queries are extracted from rated documents based on Robertsonś Selection Value (RSV). At the second stage, genetic programming is applied to discover the optimal ranking function for individual user. Pazzani and Billsus [46] developed a learning information agent called Syskill and Webert which could learn a user profile for the identification of interesting web documents. A separate user profile was created for each individual information topic. Web documents were represented as Boolean feature vectors, and each feature had a binary value indicating if a particular keyword appeared in the document or not. Feature selection was conducted based on Expected Information Gain which tends to select words appearing more frequently in positive documents. The classification mechanism of Syskill and Webert was based on a naive Bayesian classifier. The paper [3] focuses on symbolic transducers and recurrent neural preference machines to support the task of mining and classifying textual information. These encoding symbolic transducers and learning neural preference machines can be seen as independent agents, each one tackling the same task in a different manner.

Jansen [30] discusses various ways in which mobile agents could be applied to problem of detecting and responding to intrusions. Abraham et.al. [1] proposed a distributed Intrusion Detection System (IDS) consisting of several IDS over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring. In a distributed environment, system is implemented using co-operative intelligent agents distributed across the network. To detect intrusions in a network three fuzzy rule based classifiers are constructed. Moskovitch et.al. [43] conducted a comprehensive experiment for testing the feasibility of detecting unknown computer worms, employing several computer configurations, background applications, and user activities.

In [25], the authors utilize multi-agent machine learning and classifier combination to learn rock facies sequences from wireline well log data. The paper focuses on how to construct a successful set of classifiers, which periodically collaborate, to increase the classification accuracy. Utilizing multiple, heterogeneous collaborative learning agents is shown to be successful for this classification problem. Solving the pursuit problem with heterogeneous multiagent system using reinforcement learning was investigated in [29].

Zhang and Zhang [66] present , a multiagent data warehousing (MADWH) and multiagent data mining (MADM) approach for brain modeling. An algorithm named Neighbor–Miner is proposed for MADWH and MADM. The algorithm is defined in an evolving dynamic environment with semiautonomous neurofuzzy agents.

4 Machine Learning with A-Teams

Paradigms of the population-based methods and multiple agent systems have been during early nineties integrated within the concept of the asynchronous team of agents (A-Team). Acording to Talukdar et.al. [60] an asynchronous

team is a collection of software agents that cooperate to solve a problem by dynamically evolving a population of solutions. Current implementations of the A-Team concept are characterized by a high level of accessibility, scalability and portability. A review of the A-Team solutions and implementations can be found in [31]. All the machine learning solutions reviewed in this section are the A-Team implementations built using the JADE-Based A-Team middleware environment (JABAT) proposed in [7].

Two early applications of the A-Team paradigm to machine learning include training the cascade correlation learning architecture [15] and training the feed-forward artificial neural networks proposed in [16]. Recent solutions are focused on data reduction and distributed learning.

Data reduction in the supervised machine learning aims at deciding which instances and which features from the original training set should be retained for further use during the learning process. Data reduction is considered as an important step towards increasing effectiveness of the learning process when the available training sets are large or distributed and when the access to data is limited and costly. Data reduction performed without losing extractable information can result in increased capabilities and generalization properties of the learning model. It is obvious that removing some instances from the training set reduces time and memory complexity of the learning process. The data reduction algorithms can be divided into two categories: prototype selection and prototype extraction. Prototype selection is a technique of choosing a subset of reference vectors from the original set, also by reduction of attributes, whereas prototype extraction means the construction of an entirely new set of instances, smaller, in respect to its dimensionality, than the original dataset. Prototype extraction can also include the process of feature construction, where decreasing the number of attributes is carried-out by creating new features on the basis of some transformation of the original attributes. The performance criteria used in data reduction may include the accuracy of classification, the complexity of the hypothesis the classification costs and many other criteria.

An idea of applying agent technology to data reduction has been proposed in several papers of Czarnowski and Jedrzejowicz [14], [17], [18], [12], [20]. In the above papers several architectures, models and strategies for the A-Team based data reduction have been proposed. Using them usually improves quality of the respective supervised machine learning. Most competitive results have been obtained by A-Teams producing clusters of instances from the training set and then selecting instances from these clusters. Although a variety of methods could be used to produce clusters, using the similarity coefficient proposed in [14] and [17] as the clustering criterion have produced better than satisfactory results.

To solve the data reduction problem, several types of optimizing agents carrying out improvement procedures including tabu search, simulated annealing and variety of simple local search algorithms have been used. Basic assumptions behind the proposed approach are as follows:

- A solution is represented by a string consisting of two parts. The first contains numbers of instances selected as prototypes and the second – numbers of attributes chosen to represent the dataset.
- Prototype instances and attributes are selected from clusters through the population-based search carried out by the optimizing agents.
- Initially, potential solutions are generated by random selection of a single instance from each cluster and by random selection of the attribute numbers. Attributes are later adjusted by the attribute manager agent with a view to find the best combination and, at the same time, to unify the set of selected attributes at a global level (only in case of the distributed data reduction).

The solution manager is responsible for organizing the data reduction process through managing the population of solutions called individuals and updating them when appropriate. During the data reduction process the solution manager continues reading individuals (solutions) from the common memory and storing them back after attempted improvement until a stopping criterion is met. During this process the solution manager keeps sending randomly drawn individuals (solutions) from the common memory to optimizing agents. Each optimizing agent tries to improve the quality of the received solution and afterwards sends back the improved solution to the solution manager, which, in turn, updates common memory, replacing a randomly selected individual with the improved one. In each of the above cases the modified solution replaces the current one if it is evaluated as a better one. Evaluation of the solution is carried out by estimating classification accuracy of the classifier, which is constructed taking into account the instances and the attributes as indicated by the solution. Since the computational complexity of the above search procedures is linear, the computational complexity of the fitness evaluation is not greater than the complexity of the classifier induction. In case of the distributed data reduction an additional agent called attribute manager is used. Its role is to coordinate the attribute selection. The attribute manager agent is also responsible for the final integration of attributes selected locally by optimizing agents. The attribute manager actions include receiving candidate attributes form solution mangers, and deciding on the common set of attributes to be used at both the local and the global levels.

An idea of applying A-Team paradigm to solving the distributed learning problem has been evolving since a couple of years. Different solutions were presented in several papers of Czarnowski and Jedrzejowicz [11], [19], [21], [22] and [13]. The proposed approach, denoted as LCDD (Learning Classifiers from Distributed Data), involves two stages, both supported by the collaboration between agents:

- Local, in which the selection of prototypes from the distributed data takes place (A-Teams are used to select prototypes by instance selection andor removing irrelevant attributes).
- Global, consisting of pooling of the selected prototypes and producing the global learning model.

At the local level, that is, at the distributed data sources, agent-based population learning data reduction algorithms are executed in parallel. Instance and

attribute reduction are integrated with the classifier learning process. An important feature of the LCDD approach is A-Teams ability to select instances and attributes in cooperation between agents, thus assuring a homogenous set of prototypes at the global level. In this case, the instance selection is carried out independently at each site through applying the agent-based population search but the attribute selection is managed and coordinated through the process of interaction and collaboration between agents. All the required steps of the proposed approach are carried out by program agents of the four following types:

- Global level manager - agent responsible for managing the process of the distributed learning.
- Optimizing agent - agent executing a solution improvement algorithms.
- Solution manager - agent responsible for managing the population of solutions.
- Attribute manager - agent responsible for the attribute selection coordination.

The parallel process of data reduction at sites is managed by the global level manager. Its role is to manage all stages of the learning process. As the first step the global manager identifies the distributed learning task that is to be coordinated and allocates optimizing agents to the local sites using the available agent migration procedure. Then the global manager initializes parallel execution of all subtasks, that is data reduction processes at local sites. When all the subtasks have been completed, solutions from the local levels are used to produce the global solution. Producing it requires that the global manager is equipped with skills needed to induce the global classifier. When the prototypes obtained from local sites are homogenous then the local prototypes are integrated and the global manager creates the global classifier (meta-classifier), using some machine learning algorithm. When the local level solutions are represented by heterogeneous set of prototypes then the global classifier can be induced by applying one of the meta-classifier strategies like, for e xample, bagging, AdaBoost, majority voting or some hybrid strategy.

5 Conclusions

Main focus of this review is using agent technology in the field of machine learning with a particular interest on applying agent-based solutions to supervised learning. Some references are also made with respect to applying machine learning solutions to support agent technology. The review allows to formulate the following observations:

- Machine learning and agent technology are becoming more and more integrated bringing an important advantages to both fields.
- Machine learning can be seen as a prime supplier of learning capabilities for agent and multiagent systems.
- Agent technology have brought to machine learning several capabilities including parallel computation, scalability and interoperability.

– Agent-based solutions and techniques when applied to machine learning have proven to produce a synergetic effect coming from the collective intelligence of agents and a power of cooperative solutions generated through agent interactions.

Future research should help to further integrate both fields – agent technology and machine learning. Agent based solutions could be used to develop more flexible and adaptive machine learning tools. Collective computational intelligence techniques can be used to effectively solve computationally hard optimization and decision problems inherent to many supervised learning techniques and data reduction problems. Most promising direction for future research seems integration of machine learning and agent technology with a view to obtain effective solutions to the distributed learning problems. On the other hand more compact and reliable machine learning techniques are needed to equip agents with better learning capabilities.

References

1. Abraham, A., Jain, R., Thomas, J., Han, S.Y.: D-SCIDS: Distributed soft computing intrusion detection system. J. Net.Comp. Appl. 30, 81–98 (2007)
2. Albashiri, K.A., Coenen, F., Leng, P.: EMADS: An Extendible Multi-agent Data Miner. Knowl. Bas. Syst. 22, 523–528 (2009)
3. Arevian, G., Wermter, S., Panchev, C.: Symbolic state transducers and recurrent neural preference machines for text mining. Int. J. Approx. Reason. 32, 237–258 (2003)
4. Bacardit, J., Butz, M.V.: Data mining in learning classifier systems: comparing xcs with gassist. In: Kovacs, T., Llorà, X., Takadama, K., Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2003. LNCS (LNAI), vol. 4399, pp. 282–290. Springer, Heidelberg (2007)
5. Bacardit, J., Garrell, J.M.: Bloat Control and Generalization Pressure Using the Minimum Description Length Principle for a Pittsburgh Approach Learning Classifier System. In: Kovacs, T., Llorà, X., Takadama, K., Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2003. LNCS (LNAI), vol. 4399, pp. 59–79. Springer, Heidelberg (2007)
6. Bacardit, J., Krasnogor, N.: Empirical Evaluation of Ensemble Techniques for a Pittsburgh Learning Classifier System. In: Bacardit, J., Bernadó-Mansilla, E., Butz, M.V., Kovacs, T., Llorà, X., Takadama, K. (eds.) IWLCS 2006 and IWLCS 2007. LNCS (LNAI), vol. 4998, pp. 255–268. Springer, Heidelberg (2008)
7. Barbucha, D., Czarnowski, I., Jedrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: An Implementation of the JADE-base A-Team Environment. Int. Trans.Syst. Sc.Appl. 3(4), 319–328 (2008)
8. Boylu, F., Aytug, H., Koehler, G.J.: Principal–Agent Learning. Dec. Supp. Syst. 47, 75–81 (2009)
9. Bull, L., Kovacs, T.: Foundations of Learning Classifier Systems: An Introduction. Stud. Fuzz. Soft Comp. 183, 1–17 (2005)
10. Busniu, L., Babuska, R., Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cyb. 38, 156–171 (2008)
11. Czarnowski, I.: Distributed data reduction through agent collaboration. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS, vol. 5559, pp. 724–733. Springer, Heidelberg (2009)

12. Czarnowski, I.: Prototype Selection Algorithms for Distributed Learning. Pat. Recogn. 43, 2292–2300 (2010)
13. Czarnowski, I.: Distributed learning with data reduction. LNCS Transactions on Collective Computational Intelligence IV. Springer, Heidelberg (to appear, 2011)
14. Czarnowski, I., Jedrzejowicz, P.: An Approach to Instance Reduction in Supervised Learning. In: Coenen, F., Preece, A., Macintosh, A. (eds.) Research and Development in Intelligent Systems XX, pp. 267–282. Springer, London (2004)
15. Czarnowski, I., J ędrzejowicz, P.: An Agent-Based PLA for the Cascade Correlation Learning Architecture. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 197–202. Springer, Heidelberg (2005)
16. Czarnowski, I., Jedrzejowicz, P.: An Agent-based Approach to ANN Training. Knowl.-Based Syst. 19, 304–308 (2006)
17. Czarnowski, I., Jędrzejowicz, P.: An Agent-Based Approach to the Multiple-Objective Selection of Reference Vectors. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 117–130. Springer, Heidelberg (2007)
18. Czarnowski, I., Jedrzejowicz, P.: An agent-based algorithm for data Reduction. In: Bramer, M., Coenen, F., Petridis, M. (eds.) Research and Development of Intelligent Systems XXIV, pp. 351–356. Springer, London (2007)
19. Czarnowski, I., Jędrzejowicz, P.: A Comparison Study of Strategies for Combining Classifiers from Distributed Data Sources. In: Kolehmainen, M., Toivanen, P., Beliczynski, B. (eds.) ICANNGA 2009. LNCS, vol. 5495, pp. 609–618. Springer, Heidelberg (2009)
20. Czarnowski, I., Jedrzejowicz, P.: An Approach to Data Reduction and Integrated Machine Classification. New Gen. Comp. 28, 21–40 (2010)
21. Czarnowski, I., Jedrzejowicz, P.: An agent-based framework for distributed Learning. Eng. Appl. Art. Intel. 24, 93–102 (2011)
22. Czarnowski, I., Jedrzejowicz, P., Wierzbowska, I.: An A-Team Approach to Learning Classifiers from Distributed Data Sources. Int. J. Intel. Inf. Db. Syst. 4(3), 245–263 (2010)
23. Fan, W., Gordon, M., Pathak, P.: An integrated two-stage model for intelligent information routing. Dec. Sup. Syst. 42(1), 362–374 (2006)
24. Gifford, C.: M.: Collective Machine Learning: Team Learning and Classification in Multi-Agent Systems. Ph.D. dissertation, University of Kansas (2009)
25. Gifford, C.M., Agah, A.: Collaborative multi-agent rock facies classification from wireline well log data. Eng. Appl. Art. Intel. 23, 1158–1172 (2010)
26. Hofmann, T., Basilico, J.: Collaborative Machine Learning. In: Hemmje, M., Niederée, C., Risse, T. (eds.) From Integrated Publication and Information Systems to Information and Knowledge Environments. LNCS, vol. 3379, pp. 173–182. Springer, Heidelberg (2005)
27. Hoenl, P.J., Tuyls, K.: Analyzing Multi-agent Reinforcement Learning Using Evolutionary Dynamics. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 168–179. Springer, Heidelberg (2004)
28. Holland, J.H.: Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning, an Artificial Intelligence Approach, vol. II, pp. 593–623. Morgan Kaufmann, Palo Alto (1986)
29. Ishiwaka, Y., Sato, T., Kakazu, Y.: An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. Rob. Autonom. Syst. 43, 245–256 (2003)

30. Jansen, W.A.: Intrusion detection with mobile agents. Comp. Comm. 25, 1392–1401 (2002)
31. Jędrzejowicz, P.: A-Teams and Their Applications. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS, vol. 5796, pp. 36–50. Springer, Heidelberg (2009)
32. Jędrzejowicz, J., Jędrzejowicz, P.: A Family of GEP-Induced Ensemble Classifiers. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS, vol. 5796, pp. 641–652. Springer, Heidelberg (2009)
33. Jędrzejowicz, J., Jędrzejowicz, P.: Two Ensemble Classifiers Constructed from GEP-Induced Expression Trees. In: Jędrzejowicz, P., Nguyen, N.T., Howlet, R.J., Jain, L.C. (eds.) KES-AMSTA 2010. LNCS, vol. 6071, pp. 200–209. Springer, Heidelberg (2010)
34. Jennings, N., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Aut. Ag. Multi-Ag. Syst. 1, 7–38 (1998)
35. Jiang, C., Sheng, Z.: Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. Exp. Syst. Appl. 36, 6520–6526 (2009)
36. Kitakoshi, D., Shioya, H., Nakano, R.: Empirical analysis of an on-line adaptive system using a mixture of Bayesian networks. Inf. Sc. 180, 2856–2874 (2010)
37. Klusch, M., Lodi, S., Moro, G.L.: Agent-Based Distributed Data Mining: The KDEC Scheme. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) Intelligent Information Agents. LNCS (LNAI), vol. 2586, pp. 104–122. Springer, Heidelberg (2003)
38. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems - A modal logic formulation. Artif. Int. 149(1), 31–60 (2003)
39. Loizos, M.: Partial observability and learnability. Artif. Int. 174, 639–669 (2010)
40. Luo, J., Wang, M., Hu, J., Shi, Z.: Distributed data mining on Agent Grid: Issues, platform and development toolkit. Fut. Gen. Comp. Syst. 23, 61–68 (2007)
41. Mannor, S., Shamma, J.S.: Multi-agent learning for engineers. Artif. Int. 171, 417–422 (2007)
42. Masoumi, B., Meybodi, M.R.: Speeding up learning automata based multi agent systems using the concepts of stigmergy and entropy. Exp. Syst. Appl. (to appear, 2011)
43. Moskovitch, R., Elovici, Y., Rokach, L.: Detection of unknown computer worms based on behavioral classification of the host. Comp. Stat. Data Anal. 52, 4544–4566 (2008)
44. Negatu, A., D'Mello, S.K., Franklin, S.: Cognitively Inspired Anticipatory Adaptation and Associated Learning Mechanisms for Autonomous Agents. In: Butz, M.V., Sigaud, O., Pezzulo, G., Baldassarre, G. (eds.) ABiALS 2006. LNCS (LNAI), vol. 4520, pp. 108–127. Springer, Heidelberg (2007)
45. Nowé, A., Verbeeck, K., Peeters, M.: Learning automata as a basis for multi agent reinforcement learning. In: Tuyls, K., Hoen, P.J., Verbeeck, K., Sen, S. (eds.) LAMAS 2005. LNCS (LNAI), vol. 3898, pp. 71–85. Springer, Heidelberg (2006)
46. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. 27(3), 313–331 (1997)
47. Preux, P., Delepoulle, S., Darcheville, J.-C.: A generic architecture for adaptive agents based on reinforcement learning. Inf. Sc. 161, 37–55 (2004)
48. Prodromidis, A., Chan, P.K., Stolfos, S.J.: Meta-learning in Distributed Data Mining Systems: Issues and Approaches. In: Kargupta, H., Chan, P. (eds.) Advances in Distributed and Parallel Knowledge Discovery, vol. 3, AAAI/MIT Press, Menlo Park (2000)

49. Quteishat, A., Lim, C.P., Tweedale, J., Jain, L.C.: A neural network-based multi-agent classifier system. Neurocomp. 72, 1639–1647 (2009)
50. Raicevic, P.: Parallel reinforcement learning using multiple reward signals. Neurocomp. 69, 2171–2179 (2006)
51. Rosaci, D.: CILIOS: Connectionist inductive learning and inter-ontology similarities for recommending information agents. Inf. Sys. 32, 793–825 (2007)
52. Sardinha, J.A.R.P., Garcia, A., de Lucena, C.J.P., Milidiú, R.L.: A Systematic Approach for Including Machine Learning in Multi-agent Systems. In: Bresciani, P., Giorgini, P., Henderson-Sellers, B., Low, G., Winikoff, M. (eds.) AOIS 2004. LNCS (LNAI), vol. 3508, pp. 198–211. Springer, Heidelberg (2005)
53. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artif. Int. 171(7), 365–377 (2007)
54. Sian, S.: Extending Learning to Multiple Agents: Issues and a Model for Multi-Agent Machine Learning (Ma-Ml). In: Kodratoff, Y. (ed.) EWSL 1991. LNCS, vol. 482, pp. 440–456. Springer, Heidelberg (1991)
55. Smith, R.E., Jiang, M.K., Bacardit, J., Stout, M., Krasnogor, N., Hirst, J.D.: A learning classifier system with mutual-information-based fitness. Evol. Int. 1(3), 31–50 (2010)
56. Stolfo, S., Prodromidis, A.L., Tselepis, S., Lee, W., Fan, D.W.: JAM: Java Agents for Meta-learning over Distributed Databases. In: 3rd International Conference on Knowledge Discovery and Data Mining, pp. 74–81. AAAI Press, Newport Beach (1997)
57. Sutton, R.S., Barto, A.G.: Reinforcement Learning. An Introduction. MIT Press, Cambridge (1998)
58. Symeonidis, A.L., Chatzidimitriou, K.C., Athanasiadis, I.N., Mitkas, P.A.: Data mining for agent reasoning: A synergy for training intelligent agents. Eng. Appl. Artif. Int. 20, 1097–1111 (2007)
59. Takadama, K., Inoue, H., Shimohara, K., Okada, M., Katai, O.: Agent architecture based on an interactive self-reflection classifier system. Artif. Life Rob. 5, 103–108 (2001)
60. Talukdar, S., Baerentzen, L., Gove, A., De Souza, P.: Asynchronous Teams: Cooperation Schemes for Autonomous Agents. J. Heur. 4(4), 295–321 (1998)
61. Tozicka, J., Rovatsos, M., Pechoucek, M., Urban, U.: MALEF: Framework for Distributed Machine Learning and Data Mining. Int. J. Int. Inf. Db. Sys. 2(1), 6–24 (2008)
62. Tweedale, J., Ichalkaranje, N., Sioutis, C., Jarvis, B., Consoli, A., Phillips-Wren, G.E.: Innovations in multi-agent systems. J. Net. Comp. Appl. 30(3), 1089–1115 (2007)
63. Wang, Y.-C., Usher, J.M.: Application of reinforcement learning for agent-based production scheduling. Eng. Appl. Artif. Int 18, 73–82 (2005)
64. Wilson, S.W.: State of XCS Classifier System Research. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 1999. LNCS (LNAI), vol. 1813, pp. 63–81. Springer, Heidelberg (2000)
65. Yu, L., Yue, W., Wang, S., Lai, K.K.: Support vector machine based multiagent ensemble learning for credit risk evaluation. Exp. Sys. Appl. 37, 1351–1360 (2010)
66. Zhang, W.-R., Zhang, L.: A multiagent data warehousing (MADWH) and multi-agent data mining (MADM) approach to brain modeling and neurofuzzy control. Inf. Sc 167, 109–127 (2004)
67. Zhang, S., Wu, X., Zhang, C.: Multi-Database Mining. IEEE Computational Intelligence Bulletin 2(1), 5–13 (2003)