A Framework for Agent-Based Distributed Machine Learning and Data Mining

Jan Tozicka
Gerstner Laboratory
Czech Technical University
Technická 2, Prague, 166 27
Czech Republic
tozicka@labe.felk.cvut.cz

Michael Rovatsos
School of Informatics
The University of Edinburgh
Edinburgh EH8 9LE
United Kingdom
mrovatso@inf.ed.ac.uk

Michal Pechoucek
Gerstner Laboratory
Czech Technical University
Technická 2, Prague, 166 27
Czech Republic
pechouc@labe.felk.cvut.cz

ABSTRACT

This paper proposes a framework for agent-based distributed machine learning and data mining based on (i) the exchange of meta-level descriptions of individual learning processes among agents and (ii) online reasoning about learning success and learning progress by learning agents. We present an abstract architecture that enables agents to exchange models of their local learning processes and introduces a number of different methods for integrating these processes. This allows us to apply existing agent interaction mechanisms to distributed machine learning tasks, thus leveraging the powerful coordination methods available in agent-based computing, and enables agents to engage in *meta-reasoning* about their own learning decisions. We apply this architecture to a real-world distributed clustering application to illustrate how the conceptual framework can be used in practical systems in which different learners may be using different datasets, hypotheses and learning algorithms. We report on experimental results obtained using this system, review related work on the subject, and discuss potential future extensions to the framework.

General Terms

Theory

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

Keywords

Multiagent learning, distributed machine learning, frameworks and architectures, unsupervised clustering

1. INTRODUCTION

In the areas of machine learning and data mining (cf. [14, 17] for overviews), it has long been recognised that parallelisation and distribution can be used to improve learning performance. Various techniques have been suggested in this respect, ranging from the low-level integration of independently derived learning hypotheses (e.g. combining different classifiers to make optimal classification decisions [4, 7], model averaging of Bayesian classifiers [8], or consensus-based methods for integrating different clusterings [11]), to the high-level combination of learning results obtained by heterogeneous learning "agents" using meta-learning (e.g. [3, 10, 21]).

All of these approaches assume homogeneity of agent design (all agents apply the same learning algorithm) and/or agent objectives (all agents are trying to cooperatively solve a single, global learning problem). Therefore, the techniques they suggest are not applicable in societies of *autonomous* learners interacting in *open systems*. In such systems, learners (agents) may not be able to integrate their datasets or learning results (because of different data formats and representations, learning algorithms, or legal restrictions that prohibit such integration [11]) and cannot always be guaranteed to interact in a strictly cooperative fashion (discovered knowledge and collected data might be economic assets that should only be shared when this is deemed profitable; malicious agents might attempt to adversely influence others' learning results, etc.).

Examples for applications of this kind abound. Many distributed learning domains involve the use of sensitive data and prohibit the exchange of this data (e.g. exchange of patient data in distributed brain tumour diagnosis [2]) – however, they may permit the exchange of local learning hypotheses among different learners. In other areas, training data might be commercially valuable, so that agents would only make it available to others if those agents could provide something in return (e.g. in remote ship surveillance and tracking, where the different agencies involved are commercial service providers [1]). Furthermore, agents might have a vested interest in negatively affecting other agents' learning performance. An example for this is that of fraudulent agents on eBay which may try to prevent reputation-learning agents from the construction of useful models for detecting fraud.

Viewing learners as autonomous, self-directed agents is the *only* appropriate view one can take in modelling these distributed learning environments: the agent metaphor be-

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

comes a necessity as oppossed to preferences for scalability, dynamic data selection, "interactivity" [13], which can also be achieved through (non-agent) distribution and parallelisation in principle.

Despite the autonomy and self-directedness of learning agents, many of these systems exhibit a sufficient overlap in terms of individual learning goals so that beneficial cooperation might be possible if a model for flexible interaction between autonomous learners was available that allowed agents to

1. exchange information about different aspects of their own learning mechanism at different levels of detail without being forced to reveal private information that should not be disclosed,
2. decide to what extent they want to share information about their own learning processes and utilise information provided by other learners, and
3. reason about how this information can best be used to improve their own learning performance.

Our model is based on the simple idea that autonomous learners should maintain meta-descriptions of their own learning processes (see also [3]) in order to be able to exchange information and reason about them in a rational way (i.e. with the overall objective of improving their own learning results). Our hypothesis is a very simple one:

> *If we can devise a sufficiently general, abstract view of describing learning processes, we will be able to utilise the whole range of methods for (i) rational reasoning and (ii) communication and coordination offered by agent technology so as to build effective autonomous learning agents.*

To test this hypothesis, we introduce such an abstract architecture (section 2) and implement a simple, concrete instance of it in a real-world domain (section 3). We report on empirical results obtained with this implemented system that demonstrate the viability of our approach (section 4). Finally, we review related work (section 5) and conclude with a summary, discussion of our approach and outlook to future work on the subject (section 6).

2. ABSTRACT ARCHITECTURE

Our framework is based on providing formal (meta-level) descriptions of learning processes, i.e. representations of all relevant components of the learning machinery used by a learning agent, together with information about the state of the learning process.

To ensure that this framework is sufficiently general, we consider the following general description of a learning problem:

> Given data $D \subseteq \mathcal{D}$ taken from an instance space $\mathcal{D}$, a hypothesis space $\mathcal{H}$ and an (unknown) target function $c \in \mathcal{H}$[1], derive a function $h \in \mathcal{H}$ that approximates c as well as possible according to some performance measure $g : \mathcal{H} \rightarrow \mathcal{Q}$ where $\mathcal{Q}$ is a set of possible levels of learning performance.

This very broad definition includes a number of components of a learning problem for which more concrete specifications can be provided if we want to be more precise. For the cases of classification and clustering, for example, we can further specify the above as follows: Learning data can be described in both cases as $\mathcal{D} = \times_{i=1}^{n}[A_i]$ where $[A_i]$ is the domain of the ith attribute and the set of attributes is $A = \{1, \ldots, n\}$. For the hypothesis space we obtain

$$\mathcal{H} \subseteq \{h | h : \mathcal{D} \rightarrow \{0, 1\}\}$$

in the case of classification (i.e. a subset of the set of all possible classifiers, the nature of which depends on the expressivity of the learning algorithm used) and

$$\mathcal{H} \subseteq \{h | h : \mathcal{D} \rightarrow \mathbb{N}, h \text{ is total with range } \{1, \ldots, k\}\}$$

in the case of clustering (i.e. a subset of all sets of possible cluster assignments that map data points to a finite number of clusters numbered 1 to k). For classification, g might be defined in terms of the numbers of false negatives and false positives with respect to some validation set $V \subseteq \mathcal{D}$, and clustering might use various measures of cluster validity to evaluate the quality of a current hypothesis, so that $\mathcal{Q} = \mathbb{R}$ in both cases (but other sets of learning quality levels can be imagined).

Next, we introduce a notion of *learning step*, which imposes a uniform basic structure on all learning processes that are supposed to exchange information using our framework. For this, we assume that each learner is presented with a finite set of data $D = \langle d_1, \ldots d_k \rangle$ in each step (this is an ordered set to express that the order in which the samples are used for training matters) and employs a training/update function $f : \mathcal{H} \times \mathcal{D}^* \rightarrow \mathcal{H}$ which updates h given a series of samples $d_1, \ldots, d_k$. In other words, one learning step always consists of applying the update function to all samples in D exactly once. We define a *learning step* as a tuple

$$l = \langle D, H, f, g, h \rangle$$

where we require that $H \subseteq \mathcal{H}$ and $h \in H$.

The intuition behind this definition is that each learning step completely describes one learning iteration as shown in Figure 1: in step t, the learner updates the current hypothesis h_{t-1} with data D_t, and evaluates the resulting new hypothesis h_t according to the current performance measure g_t. Such a learning step is equivalent to the following steps of computation:

1. train the algorithm on all samples in D (once), i.e. calculate $f_t(h_{t-1}, D_t) = h_t$,
2. calculate the quality g_t of the resulting hypothesis $g_t(h_t)$.

We denote the set of all possible learning steps by L. For ease of notation, we denote the components of any $l \in L$ by $D(l)$, $H(l)$, $f(l)$ and $g(l)$, respectively. The reason why such learning step specifications use a subset H of $\mathcal{H}$ instead of $\mathcal{H}$ itself is that learners often have explicit knowledge about which hypotheses are effectively ruled out by f given h in the future (if this is not the case, we can still set $H = \mathcal{H}$).

A *learning process* is a finite, non-empty sequence

$$\mathbf{l} = l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_n$$

of learning steps such that

$$\forall 1 \leq i < n \,.h(l_{i+1}) = f(l_i)(h(l_i), D(l_i))$$

[1] By requiring this we are ensuring that the learning problem can be solved in principle using the given hypothesis space.

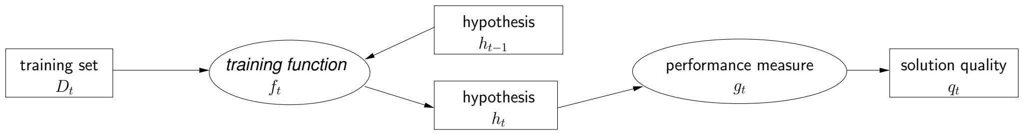


Figure 1: A generic model of a learning step

i.e. the only requirement the transition relation $\rightarrow \subseteq L \times L$ makes is that the new hypothesis is the result of training the old hypothesis on all available sample data that belongs to the current step. We denote the set of all possible learning processes by $\mathcal{L}$ (ignoring, for ease of notation, the fact that this set depends on $\mathcal{H}$, $\mathcal{D}$ and the spaces of possible training and evaluation functions f and g). The *performance trace* associated with a learning process l is the sequence $\langle q_1, \ldots, q_n \rangle \in Q^n$ where $q_i = g(l_i)(h(l_i))$, i.e. the sequence of quality values calculated by the performance measures of the individual learning steps on the respective hypotheses.

Such specifications allow agents to provide a self-description of their learning process. However, in communication among learning agents, it is often useful to provide only partial information about one's internal learning process rather than its full details, e.g. when advertising this information in order to enter information exchange negotiations with others. For this purpose, we will assume that learners describe their internal state in terms of *sets of learning processes* (in the sense of disjunctive choice) which we call *learning process descriptions* (LPDs) rather than by giving precise descriptions about a single, concrete learning process.

This allows us to describe *properties* of a learning process without specifying its details exhaustively. As an example, the set $\{\mathbf{l} \in \mathcal{L} | \forall l = \mathbf{l}[i].D(l) \leq 100\}$ describes all processes that have a training set of at most 100 samples (where all the other elements are arbitrary). Likewise, $\{\mathbf{l} \in \mathcal{L} | \forall l = \mathbf{l}[i].D(l) = \{d\}\}$ is equivalent to just providing information about a single sample $\{d\}$ and no other details about the process (this can be useful to model, for example, data received from the environment). Therefore, we use $\wp(\mathcal{L})$, that is the set of all LPDs, as the basis for designing content languages for communication in the protocols we specify below.

In practice, the actual content language chosen will of course be more restricted and allow only for a special type of subsets of $\mathcal{L}$ to be specified in a compact way, and its choice will be crucial for the interactions that can occur between learning agents. For our examples below, we simply assume explicit enumeration of all possible elements of the respective sets and function spaces (D, H, etc.) extended by the use of wildcard symbols $*$ (so that our second example above would become $(\{d\}, *, *, *, *)$).

2.1 Learning agents

In our framework, a learning agent is essentially a meta-reasoning function that operates on information about learning processes and is situated in an environment co-inhabited by other learning agents. This means that it is not only capable of meta-level control on "how to learn", but in doing so it can take information into account that is provided by other agents or the environment. Although purely cooperative or hybrid cases are possible, for the purposes of this paper we will assume that agents are purely self-interested, and that while there may be a potential for cooperation considering how agents can mutually improve each others' learning performance, there is no global mechanism that can enforce such cooperative behaviour.[2]

Formally speaking, an agent's *learning* function is a function which, given a set of histories of previous learning processes (of oneself and potentially of learning processes about which other agents have provided information) and outputs a learning step which is its next "learning action". In the most general sense, our learning agent's internal learning process update can hence be viewed as a function

$$\lambda : \wp(\mathcal{L}) \rightarrow L \times \wp(\mathcal{L})$$

which takes a set of learning "histories" of oneself and others as inputs and computes a new learning step to be executed while updating the set of known learning process histories (e.g. by appending the new learning action to one's own learning process and leaving all information about others' learning processes untouched). Note that in $\lambda(\{\mathbf{l}_1, \ldots \mathbf{l}_n\}) = (l, \{\mathbf{l}'_1, \ldots \mathbf{l}'_{n'}\})$ some elements $\mathbf{l}_i$ of the input learning process set may be descriptions of new learning data received from the environment.

The λ-function can essentially be freely chosen by the agent as long as one requirement is met, namely that the learning data that is being used always stems from what has been previously observed. More formally,

$$\forall \{\mathbf{l}_1, \ldots \mathbf{l}_n\} \in \wp(\mathcal{L}).\lambda(\{\mathbf{l}_1, \ldots \mathbf{l}_n\}) = (l, \{\mathbf{l}'_1, \ldots \mathbf{l}'_{n'}\})$$
$$\Rightarrow \left(D(l) \cup \Big(\bigcup_{l' = \mathbf{l}'_i[j]} D(l')\Big)\right) \subseteq \bigcup_{l'' = \mathbf{l}_i[j]} D(l'')$$

i.e. whatever λ outputs as a new learning step and updated set of learning histories, it cannot "invent" new data; it has to work with the samples that have been made available to it earlier in the process through the environment or from other agents (and it can of course re-train on previously used data).

The goal of the agent is to output an optimal learning step in each iteration given the information that it has. One possibility of specifying this is to require that

$$\forall \{\mathbf{l}_1, \ldots \mathbf{l}_n\} \in \wp(\mathcal{L}).\lambda(\{\mathbf{l}_1, \ldots \mathbf{l}_n\}) = (l, \{\mathbf{l}'_1, \ldots \mathbf{l}'_{n'}\})$$
$$\Rightarrow l = \arg\max_{l' \in L} g(l')(h(l'))$$

but since it will usually be unrealistic to compute the optimal next learning step in every situation, it is more useful

[2]Note that our outlook is not only different from common, cooperative models of distributed machine learning and data mining, but also delineates our approach from multiagent learning systems in which agents learn *about* other agents [25], i.e. the learning goal *itself* is not affected by agents' behaviour in the environment.

$i \setminus j$	D_j	H_j	f_j	g_j	h_j
D_i	$p_1^{D\to D}(D_i, D_j)$ $\vdots$ $p_{k_{D\to D}}^{D\to D}(D_i, D_j)$	$\dots$	$\dots$	n/a	$\dots$
H_i	$\vdots$	$\ddots$		n/a	
f_i	$\vdots$		$\ddots$	n/a	
g_i	$\vdots$			n/a	$p_1^{g\to h}(g_i, h_j)$ $\vdots$ $p_{k_{g\to h}}^{g\to h}(g_i, h_j)$
h_i	$\vdots$			n/a	$\ddots$

Table 1: Matrix of integration functions for messages sent from learner i to j

to simply use $g(l')(h(l'))$ as a running performance measure to evaluate how well the agent is performing.

This is too abstract and unspecific for our purposes: While it describes *what* agents should do (transform the settings for the next learning step in an optimal way), it doesn't specify how this can be achieved in practice.

2.2 Integrating learning process information

To specify how an agent's learning process can be affected by integrating information received from others, we need to flesh out the details of how the learning steps it will perform can be modified using incoming information about learning processes described by other agents (this includes the acquisition of new learning data from the environment as a special case). In the most general case, we can specify this in terms of the potential modifications to the existing information about learning histories that can be performed using new information. For ease of presentation, we will assume that agents are *stationary* learning processes that can only record the previously executed learning step and only exchange information about this one individual learning step (our model can be easily extended to cater for more complex settings).

Let $l_j = \langle D_j, H_j, f_j, g_j, h_j \rangle$ be the current "state" of agent j when receiving a learning process description $l_i = \langle D_i, H_i, f_i, g_i, h_i \rangle$ from agent i (for the time being, we assume that this is a *specific* learning step and not a more vague, disjunctive description of properties of the learning step of i). Considering all possible interactions at an abstract level, we basically obtain a matrix of possibilities for modifications of j's learning step specification as shown in Table 1. In this matrix, each entry specifies a family of integration functions $p_1^{c\to c'}, \dots, p_{k_{c\to c'}}^{c\to c'}$ where $c, c' \in \{D, H, f, g, h\}$ and which define how agent j's component c'_j will be modified using the information c_i provided about (the same or a different component of) i's learning step by applying $p_r^{c\to c'}(c_i, c'_j)$ for some $r \in \{1, \dots, k_{c\to c'}\}$. To put it more simply, the collections of p-functions an agent j uses specifies how it will modify its own learning behaviour using information obtained from i.

For the diagonal of this matrix, which contains the most common ways of integrating new information in one's own learning model, obvious ways of modifying one's own learning process include replacing c'_j by c_i or ignoring c_i altogether. More complex/subtle forms of learning process integration include:

- Modification of D_j: append D_i to D_j; filter out all elements from D_j which also appear in D_i; append D_i to D_j discarding all elements with attributes outside ranges which affect g_j, or those elements already correctly classified by h_j;
- Modification of H_i: use the union/intersection of H_i and H_j; alternatively, discard elements of H_j that are inconsistent with D_j in the process of intersection or union, or filter out elements that cannot be obtained using f_j (unless f_j is modified at the same time)
- Modification of f_j: modify parameters or background knowledge of f_j using information about f_i; assess their relevance by simulating previous learning steps on D_j using g_j and discard those that do not help improve own performance
- Modification of h_j: combine h_j with h_i using (say) logical or mathematical operators; make the use of h_i contingent on a "pre-integration" assessment of its quality using own data D_j and g_j

While this list does not include fully fledged, concrete integration operations for learning processes, it is indicative of the broad range of interactions between individual agents' learning processes that our framework enables.

Note that the list does not include any modifications to g_j. This is because we do not allow modifications to the agent's own quality measure as this would render the model of rational (learning) action useless (if the quality measure is relative and volatile, we cannot objectively judge learning performance). Also note that some of the above examples require consulting other elements of l_j than those appearing as arguments of the p-operations; we omit these for ease of notation, but emphasise that information-rich operations will involve consulting many different aspects of l_j.

Apart from operations along the diagonal of the matrix, more "exotic" integration operations are conceivable that combine information about *different* components. In theory we could fill most of the matrix with entries for them, but for lack of space we list only a few examples:

- Modification of D_j using f_i: pre-process samples in f_i, e.g. to achieve intermediate representations that f_j can be applied to
- Modification of D_j using h_i: filter out samples from D_j that are covered by h_i and build h_j using f_j only on remaining samples
- Modification of H_j using f_i: filter out hypotheses from H_j that are not realisable using f_i
- Modification of h_j using g_i: if h_j is composed of several sub-components, filter out those sub-components that do not perform well according to g_i
- ...

Finally, many messages received from others describing properties of their learning processes will contain information about several elements of a learning step, giving rise to yet more complex operations that depend on which kinds of information are available.

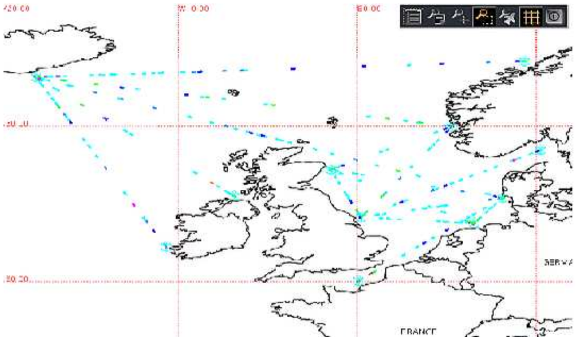


Figure 2: Screenshot of our simulation system, displaying online vessel tracking data for the North Sea region

3. APPLICATION EXAMPLE

3.1 Domain description

As an illustration of our framework, we present an agent-based data mining system for clustering-based surveillance using AIS (Automatic Identification System [1]) data. In our application domain, different commercial and governmental agencies track the journeys of ships over time using AIS data which contains structured information automatically provided by ships equipped with shipborne mobile AIS stations to shore stations, other ships and aircrafts. This data contains the ship's identity, type, position, course, speed, navigational status and other safety-related information. Figure 2 shows a screenshot of our simulation system.

It is the task of AIS agencies to detect anomalous behaviour so as to alarm police/coastguard units to further investigate unusual, potentially suspicious behaviour. Such behaviour might include things such as deviation from the standard routes between the declared origin and destination of the journey, unexpected "close encounters" between different vessels on sea, or unusual patterns in the choice of destination over multiple journeys, taking the type of vessel and reported freight into account. While the reasons for such unusual behaviour may range from pure coincidence or technical problems to criminal activity (such as smuggling, piracy, terrorist/military attacks) it is obviously useful to pre-process the huge amount of vessel (tracking) data that is available before engaging in further analysis by human experts.

To support this automated pre-processing task, software used by these agencies applies clustering methods in order to identify outliers and flag those as potentially suspicious entities to the human user. However, many agencies active in this domain are competing enterprises and use their (partially overlapping, but distinct) datasets and learning hypotheses (models) as assets and hence cannot be expected to collaborate in a fully cooperative way to improve overall learning results. Considering that this is the reality of the domain in the real world, it is easy to see that a framework like the one we have suggested above might be useful to exploit the cooperation potential that is not exploited by current systems.

3.2 Agent-based distributed learning system design

To describe a concrete design for the AIS domain, we need to specify the following elements of the overall system:

1. The datasets and clustering algorithms available to individual agents,
2. the interaction mechanism used for exchanging descriptions of learning processes, and
3. the decision mechanism agents apply to make learning decisions.

Regarding 1., our agents are equipped with their own private datasets in the form of vessel descriptions. Learning samples are represented by tuples containing data about individual vessels in terms of attributes $A = \{1, \dots, n\}$ including things such as width, length, etc. with real-valued domains ($[A_i] = \mathbb{R}$ for all i).

In terms of learning algorithm, we consider clustering with a fixed number of k clusters using the k-means and k-medoids clustering algorithms [5] ("fixed" meaning that the learning algorithm will always output k clusters; however, we allow agents to change the value of k over different learning cycles). This means that the hypothesis space can be defined as $\mathcal{H} = \{\langle c_1, \dots, c_k \rangle | c_i \in \mathbb{R}^{|A|}\}$ i.e. the set of all possible sets of k cluster centroids in $|A|$-dimensional Euclidean space. For each hypothesis $h = \langle c_1, \dots, c_k \rangle$ and any data point $d \in \times_{i=1}^{n}[A_i]$ given domain $[A_i]$ for the ith attribute of each sample, the assignment to clusters is given by

$$C(\langle c_1, \dots, c_k \rangle, d) = \arg \min_{1 \leq j \leq k} |d - c_j|$$

i.e. d is assigned to that cluster whose centroid is closest to the data point in terms of Euclidean distance.

For evaluation purposes, each dataset pertaining to a particular agent i is initially split into a training set D_i and a validation V_i. Then, we generate a set of "fake" vessels F_i such that $|F_i| = |V_i|$. These two sets assess the agent's ability to detect "suspicious" vessels. For this, we assign a confidence value $r(h, d)$ to every ship d:

$$r(h, d) = \frac{1}{|d - c_{C(h,d)}|}$$

where $C(h, d)$ is the index of the nearest centroid. Based on this measure, we classify any vessel in $F_i \cup V_i$ as fake if its r-value is below the median of all the confidences $r(h, d)$ for $d \in F_i \cup V_i$. With this, we can compute the quality $g_i(h) \in \mathbb{R}$ as the ratio between all correctly classified vessels and all vessels in $F_i \cup V_i$.

As concerns 2., we use a simple Contract-Net Protocol (CNP) [20] based "hypothesis trading" mechanism: Before each learning iteration, agents issue (publicly broadcasted) Calls-For-Proposals (CfPs), advertising their own numerical model quality. In other words, the "initiator" of a CNP describes its own current learning state as $(*, *, *, g_i(h), *)$ where h is their current hypothesis/model. We assume that agents are sincere when advertising their model quality, but note that this quality might be of limited relevance to other agents as they may specialise on specific regions of the data space not related to the test set of the sender of the CfP.

Subsequently, (some) agents may issue bids in which they advertise, in turn, the quality of their own model. If the bids (if any) are accepted by the initiator of the protocol who issued the CfP, the agents exchange their hypotheses and the next learning iteration ensues.

To describe what is necessary for 3., we have to specify (i) under which conditions agents submit bids in response to a CfP, (ii) when they accept bids in the CNP negotiation process, and (iii) how they integrate the received information in their own learning process. Concerning (i) and (ii), we employ a very simple rule that is identical in both cases: let g be one's own model quality and g' that advertised by the CfP (or highest bid, respectively). If $g' > g$ we respond to the CfP (accept the bid), else respond to the CfP (accept the bid) with probability $P(g'/g)$ and ignore (reject) it else. If two agents make a deal, they exchange their learning hypotheses (models). In our experiments, g and g' are calculated by an additional agent that acts as a global validation mechanism for all agents (in a more realistic setting a comparison mechanism for different g functions would have to be provided).

As for (iii), each agent uses a single model merging operator taken from the following two classes of operators (h_j is the receiver's own model and h_i is the provider's model):

- $p^{h \to h}(h_i, h_j)$:
  - *m-join:* The m best clusters (in terms of coverage of D_j) from hypothesis h_i are appended to h_j.
  - *m-select:* The set of the m best clusters (in terms of coverage of D_j) from the union $h_i \cup h_j$ is chosen as a new model. (Unlike m-join this method does not prefer own clusters over others'.)
- $p^{h \to D}(h_i, D_j)$:
  - *m-filter:* The m best clusters (as above) from h_i are identified and appended to a new model formed by using those samples not covered by these clusters applying the own learning algorithm f_j.

Whenever m is large enough to encompass *all* clusters, we simply write *join* or *filter* for them. In section 4 we analyse the performance of each of these two classes for different choices of m.

It is noteworthy that this agent-based distributed data mining system is one of the simplest conceivable instances of our abstract architecture. While we have previously applied it also to a more complex market-based architecture using Inductive Logic Programming learners in a transport logistics domain [22], we believe that the system described here is complex enough to illustrate the key design decisions involved in using our framework and provides simple example solutions for these design issues.

4. EXPERIMENTAL RESULTS

Figure 3 shows results obtained from simulations with three learning agents in the above system using the k-means and k-medoids clustering methods respectively. We partition the total dataset of 300 ships into three disjoint sets of 100 samples each and assign each of these to one learning agent. The *Single Agent* is learning from the whole dataset. The parameter k is set to 10 as this is the optimal value for the total dataset according to the Davies-Bouldin index [9]. For *m-select* we assume $m = k$ which achieves a constant

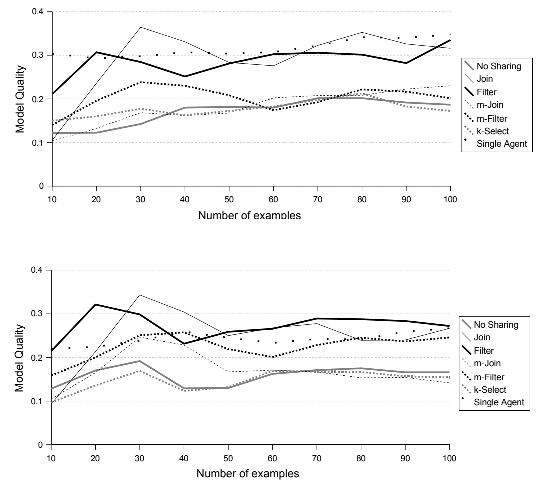


Figure 3: Performance results obtained for different integration operations in homogeneous learner societies using the k-means (top) and k-medoids (bottom) methods

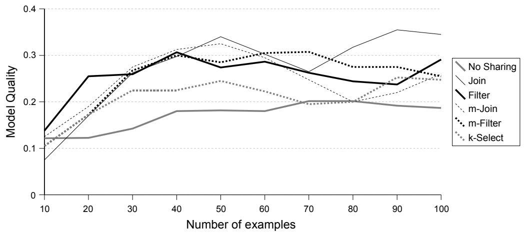
model size. For *m-join* and *m-filter* we assume $m = 3$ to limit the extent to which models increase over time.

During each experiment the learning agents receive ship descriptions in batches of 10 samples. Between these batches, there is enough time to exchange the models among the agents and recompute the models if necessary. Each ship is described using width, length, draught and speed attributes with the goal of learning to detect which vessels have provided fake descriptions of their own properties. The validation set contains 100 real and 100 randomly generated fake ships. To generate sufficiently realistic properties for fake ships, their individual attribute values are taken from randomly selected ships in the validation set (so that each fake sample is a combination of attribute values of several existing ships).

In these experiments, we are mainly interested in investigating whether a simple form of knowledge sharing between self-interested learning agents could improve agent performance compared to a setting of isolated learners. Thereby, we distinguish between *homogeneous* learner societies where all agents use the same clustering algorithm and *heterogeneous* ones where different agents use different algorithms.

As can be seen from the performance plots in Figure 3 (homogeneous case) and 4 (heterogeneous case, two agents use the same method and one agent uses the other) this is clearly the case for the (unrestricted) join and filter integration operations ($m = k$) in both cases. This is quite natural, as these operations amount to sharing all available model knowledge among agents (under appropriate constraints depending on how beneficial the exchange seems to the agents). We can see that the quality of these operations is very close to the *Single Agent* that has access to all training data.

For the restricted ($m < k$) m-join, m-filter and m-select methods we can also observe an interesting distinction,



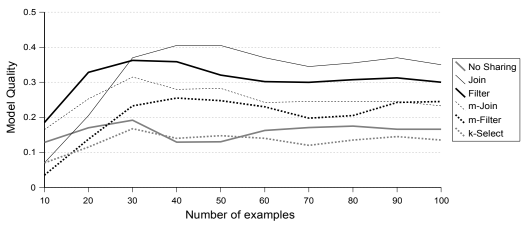


Figure 4: Performance results obtained for different integration operations in heterogeneous societies with the majority of learners using the k-means (top) and k-medoids (bottom) methods

namely that these perform similarly to the isolated learner case in homogeneous agent groups but better than isolated learners in more heterogeneous societies. This suggests that heterogeneous learners are able to benefit even from rather limited knowledge sharing (and this is what using a rather small $m = 3$ amounts to given that $k = 10$) while this is not always true for homogeneous agents. This nicely illustrates how different learning or data mining algorithms can specialise on different parts of the problem space and then integrate their local results to achieve better individual performance.

Apart from these obvious performance benefits, integrating partial learning results can also have other advantages: The *m-filter* operation, for example, decreases the number of learning samples and thus can speed up the learning process. The relative number of filtered examples measured in our experiments is shown in the following table.

	k-means	**k-medoids**
filtering	30-40 %	10-20 %
m-filtering	20-30 %	5-15 %

The overall conclusion we can draw from these initial experiments with our architecture is that since a very simplistic application of its principles has proven capable of improving the performance of individual learning agents, it is worthwhile investigating more complex forms of information exchange about learning processes among autonomous learners.

5. RELATED WORK

We have already mentioned work on distributed (non-agent) machine learning and data mining in the introductory chapter, so in this section we shall restrict ourselves to approaches that are more closely related to our outlook on distributed learning systems.

Very often, approaches that are allegedly agent-based completely disregard agent autonomy and prescribe local decision-making procedures a priori. A typical example for this type of system is the one suggested by Caragea et al. [6] which is based on a distributed support-vector machine approach where agents incrementally join their datasets together according to a fixed distributed algorithm. A similar example is the work of Weiss [24], where groups of classifier agents learn to organise their activity so as to optimise global system behaviour.

The difference between this kind of *collaborative agent-based learning systems* [16] and our own framework is that these approaches assume a joint learning goal that is pursued collaboratively by all agents.

Many approaches rely heavily on a homogeneity assumption: Plaza and Ontanon [15] suggest methods for agent-based intelligent reuse of cases in case-based reasoning but is only applicable to societies of homogeneous learners (and coined towards a specific learning method). An agent-based method for integrating distributed cluster analysis processes using density estimation is presented by Klusch et al. [13] which is also specifically designed for a particular learning algorithm. The same is true of [22, 23] which both present market-based mechanisms for aggregating the output of multiple learning agents, even though these approaches consider more interesting interaction mechanisms among learners.

A number of approaches for sharing learning data [18] have also been proposed: Grecu and Becker [12] suggest an exchange of learning samples among agents, and Ghosh et al. [11] is a step in the right direction in terms of revealing only partial information about one's learning process as it deals with limited information sharing in distributed clustering.

Papyrus [3] is a system that provides a markup language for meta-description of data, hypotheses and intermediate results and allows for an exchange of all this information among different nodes, however with a strictly cooperative goal of distributing the load for massively distributed data mining tasks.

The MALE system [19] was a very early multiagent learning system in which agents used a blackboard approach to communicate their hypotheses. Agents were able to critique each others' hypotheses until agreement was reached. However, all agents in this system were identical and the system was strictly cooperative.

The ANIMALS system [10] was used to simulate multi-strategy learning by combining two or more learning techniques (represented by heterogeneous agents) in order to overcome weaknesses in the individual algorithms, yet it was also a strictly cooperative system.

As these examples show and to the best of our knowledge, there have been no previous attempts to provide a framework that can accommodate both *independent* and *heterogeneous* learning agents and this can be regarded as the main contribution of our work.

6. CONCLUSION

In this paper, we outlined a generic, abstract framework for distributed machine learning and data mining. This framework constitutes, to our knowledge, the first attempt to capture complex forms of interaction between *heterogeneous* and/or *self-interested* learners in an architecture that can be used as the foundation for implementing systems that use complex interaction and reasoning mechanisms to enable agents to inform and improve their learning abilities with information provided by other learners in the system, provided that all agents engage in a sufficiently similar learning activity.

To illustrate that the abstract principles of our architecture can be turned into concrete, computational systems, we described a market-based distributed clustering system which was evaluated in the domain of vessel tracking for purposes of identifying deviant or suspicious behaviour. Although our experimental results only hint at the potential of using our architecture, they underline that what we are proposing is feasible in principle and can have beneficial effects even in its most simple instantiation.

Yet there is a number of issues that we have not addressed in the presentation of the architecture and its empirical evaluation: Firstly, we have not considered the cost of communication and made the implicit assumption that the required communication "comes for free". This is of course inadequate if we want to evaluate our method in terms of the total effort required for producing a certain quality of learning results. Secondly, we have not experimented with agents using completely different learning algorithms (e.g. symbolic and numerical). In systems composed of completely different agents the circumstances under which successful information exchange can be achieved might be very different from those described here, and much more complex communication and reasoning methods may be necessary to achieve a useful integration of different agents' learning processes. Finally, more sophisticated evaluation criteria for such distributed learning architectures have to be developed to shed some light on what the right measures of optimality for autonomously reasoning and communicating agents should be.

These issues, together with a more systematic and thorough investigation of advanced interaction and communication mechanisms for distributed, collaborating and competing agents will be the subject of our future work on the subject.

Acknowledgement: We gratefully acknowledge the support of the presented research by Army Research Laboratory project N62558-03-0819 and Office for Naval Research project N00014-06-1-0232.

7. REFERENCES

[1] `http://www.aislive.com`.

[2] `http://www.healthagents.com`.

[3] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: A System for Data Mining over Local and Wide Area Clusters and Super-Clusters. In *Proc. of the Conference on Supercomputing*. 1999.

[4] E. Bauer and R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, 1999.

[5] P. Berkhin. *Survey of Clustering Data Mining Techniques*, Technical Report, Accrue Software, 2002.

[6] D. Caragea, A. Silvescu, and V. Honavar. Agents that Learn from Distributed Dynamic Data sources. In *Proc. of the Workshop on Learning Agents*, 2000.

[7] N. Chawla and S. E. abd L. O. Hall. Creating ensembles of classifiers. In *Proceedings of ICDM 2001*, pages 580–581, San Jose, CA, USA, 2001.

[8] D. Dash and G. F. Cooper. Model Averaging for Prediction with Discrete Bayesian Networks. *Journal of Machine Learning Research*, 5:1177–1203, 2004.

[9] D. L. Davies and D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:224–227, 1979.

[10] P. Edwards and W. Davies. A Heterogeneous Multi-Agent Learning System. In *Proceedings of the Special Interest Group on Cooperating Knowledge Based Systems*, pages 163–184, 1993.

[11] J. Ghosh, A. Strehl, and S. Merugu. A Consensus Framework for Integrating Distributed Clusterings Under Limited Knowledge Sharing. In *NSF Workshop on Next Generation Data Mining*, 99–108, 2002.

[12] D. L. Grecu and L. A. Becker. Coactive Learning for Distributed Data Mining. In *Proceedings of KDD-98*, pages 209–213, New York, NY, August 1998.

[13] M. Klusch, S. Lodi, and G. Moro. Agent-based distributed data mining: The KDEC scheme. In *AgentLink*, number 2586 in LNCS. Springer, 2003.

[14] T. M. Mitchell. *Machine Learning*, pages 29–36. McGraw-Hill, New York, 1997.

[15] S. Ontanon and E. Plaza. Recycling Data for Multi-Agent Learning. In *Proc. of ICML-05*, 2005.

[16] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[17] B. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. In N. Ye, editor, *Data Mining Handbook*, pages 341–358, 2002.

[18] F. J. Provost and D. N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proc. of AAAI-96*, pages 74–79. AAAI Press, 1996.

[19] S. Sian. Extending learning to multiple agents: Issues and a model for multi-agent machine learning (ma-ml). In Y. Kodratoff, editor, *Machine Learning – EWSL-91*, pages 440–456. Springer-Verlag, 1991.

[20] R. Smith. The contract-net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[21] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan. Jam: Java Agents for Meta-Learning over Distributed Databases. In *Proc. of the KDD-97*, pages 74–81, USA, 1997.

[22] J. Tožička, M. Jakob, and M. Pěchouček. Market-Inspired Approach to Collaborative Learning. In *Cooperative Information Agents X (CIA 2006)*, volume 4149 of LNCS, pages 213–227. Springer, 2006.

[23] Y. Z. Wei, L. Moreau, and N. R. Jennings. Recommender systems: a market-based design. In *Proceedings of AAMAS-03)*, pages 600–607, 2003.

[24] G. Weiß. A Multiagent Perspective of Parallel and Distributed Machine Learning. In *Proceedings of Agents'98*, pages 226–230, 1998.

[25] G. Weiss and P. Dillenbourg. What is 'multi' in multi-agent learning? *Collaborative-learning: Cognitive and Computational Approaches*, 64–80, 1999.