Data Mining, Decision Support and Meta-Learning: towards an Implicit Culture architecture for KDD

E. Blanzieri[1,3], P. Giorgini[2], P. Massa[1], and S. Recla[1]

[1] ITC-irst Trento - Italy - {blanzier,massa,recla}@itc.it
[2] Dep. of Mathematics - University of Trento - Italy - pgiorgini@science.unitn.it
[3] Present: Dep. of Psychology - University of Turin - Italy - blanzier@psych.unito.it

Abstract. During data mining process users and algorithms could take advantage of the experiences other users or algorithms (agents) had. Implicit Culture is the relation between a set and a group of agents such that the elements of the set behave according to the culture of the group. The goal of a System for Implicit Culture Support (SICS) is to establish an implicit culture phenomenon, namely when the elements of a set behave according to the culture of a generally different group of agents. Supporting Implicit Culture phenomena can be useful for both artificial and human agents that is the kind of agents (miners and learners) that are involved in the data mining process. The architecture of a SICS relies on the solution of two learning problems: (i) induction of a cultural theory on the behavior patterns of the group and (ii) prediction of a scene such that the elements of the set will behave consistently with the cultural theory. An implemented SICS solves the latter problem exploiting a generalization of a Collaborative Filtering algorithm. In this paper, we recall the concept of Implicit Culture, present the architecture of a System for Implicit Culture Support (SICS) and argue that the application of a SICS to data mining process could be an interesting way of dealing with some of the problems that the practice of data mining presents.

1 Introduction

Data mining is a step in the complex process of Knowledge Discovery in Databases [6]. During the whole process a lot of choices have to be made: data cleaning, data selection, formats etc. All this decisions could be usefully helped by a decision support system. In particular, when the choices concern learning algorithms and on their parameter values, a considerable help can come from meta-learning techniques [4].

One possibility for decision support is Case-Based Reasoning. Similar solved problems can be of help. However, usually, a practisioner is focused on the domain of her own application and does not want to be bothered by the details of another, possibly helpful case. Moreover, the user operates ignoring the experience other users had on other applications. Having this knowledge available and easy-to-use would improve the results of the process.

In this paper, we suggest that approaching the problem in the recently proposed framework of Implicit Culture [2, 3] can lead to interesting insights. Moreover, adopting a System for Implicit Culture Support (SICS in the following) architecture can lead to a solution of some of the problems of practical application of data mining.

The paper is organized as follows. Section 2 introduces SICSs and their application. In Section 3, we informally introduced the concept of Implicit Culture and in Section 4 we present the architecture of a SICS and argue about a possible application in data mining. The final section hosts conclusions and future work.

2 Systems for Implicit Culture Support

Systems for Implicit Culture Support [2] have the goal of estabilishing an Implicit Culture phenomenon. Implicit Culture is the relation existing between a set and a group of agents such that the elements of the set behave according to the culture of the group. Supporting Implicit Culture is effective in solving the problem of improving the performances of agents acting in an environment where more-skilled agents are active. The architecture of SICS proposed in [2, 3] relies on the exploitation of learning techniques. In particular, it is possible to identify two learning problems: (i) induction of a cultural theory on the behavior patterns of the group and (ii) prediction of a scene such that the elements of the set will behave consistently with the cultural theory.

The concept of Implicit Culture can help the analysis of existing systems and suggest ideas for the synthesis of new ones. In fact, support of Implicit Culture can be useful for various applications: Computer Supported Collaborative Work, Group profiling, Decision Support, Cognitive modeling of social phenomena, e-books, Computer Mediated Communication Systems and, as we briefly present in the following, generalization of collaborative filtering, Knowledge Management and requirements control of agent-based systems.

Collaborative filtering [5, 14, 8] is a popular technique exploited in recommendation systems. The goal is information filtering, namely to extract from a usually long list of items (e.g., links or products) a little set that the user could prefer. Collaborative filtering exploits correlation in the pattern of preferences expressed actively o passively by other users in terms of ratings. Differently from the content-based filtering, collaborative filtering does not rely on the content or shape of objects. The central idea is to automate the process of recommending items to a user on the base of the opinions of people with similar preferences. Billsus and Pazzani have tackled collaborative filtering problems in a Machine Learning perspective [1]. As suggested by Blanzieri and Giorgini [2] collaborative filtering can be seen as a SICS and Blanzieri et al. [10] implemented a SICS that exploits instance-based learning techniques, showing that, in a particular domain and with a simple a priori theory, the system is functionally equivalent to Collaborative Filtering. The three-steps algorithm for proposing the scene can be considered a generalization of a Collaborative Filtering algorithm, where the similarity is performed on executed actions and not on ratings. This generalization is non–trivial for it puts the SICS in a wider framework than simple CF. In fact it is possible to vary the domain, the cultural constraint theory, and also to deal with artificial agents.

In Knowledge Management, generally knowledge is categorized as being either codified (explicit) or tacit (implicit). Knowledge is said being explicit when it is possible to describe and share it among people through documents and/or information bases. Knowledge is said being implicit when it is embodied in the capabilities and abilities of the members of a group of people. In [11], knowledge creation processes have been characterized in terms of tacit and explicit knowledge transformation processes, in which, instead of considering new knowledge as something that is added to the previous, they conceive it as something that transforms it. Implicit Culture can be applied successfully in this context. In particular, the idea is to build systems able to capture implicit knowledge, but instead of sharing it among people, change the environment in order to make new people behave in accordance with this knowledge.

Advantages of Implicit Culture support has been proposed also for artificial agents [3], in particular for controlling the requirements of agent-based systems. Autonomy of agents, unknown properties of the environment and insertion of new agents do not allow to foresee completely the multi-agent system's behavior in the modeling phase. As a consequence, the overall system can fail to fulfill the desired requirements. In particular, requirements should persist after a composition changing of the group of agents, that is the new agents should act consistently with the culture of the group. Using SICSs, it is possible to modify the view that the agents have of the environment and, consequently, change the set of possible actions that the agents can perform in the environment. Working on the possible actions, a SICS is able to lead the new agents to act consistently with the behavior of the group. Blanzieri et al. [9] shown how to use a SICS for supporting multi-agent interaction, presenting the implementation of the eCulture Brokering System. Implicit Culture Support improved the interaction among agents without need to equip the agents with additional capabilities.

Implicit Culture is related to the area of multi-agents learning. A relevant portion of research on multi-agents learning [15] deals with reinforcement learning (e.g., [7]). From the point of view of reinforcement learning the works of Price and Boutillier [13] on *Implicit Imitation* are relevant to our work. The critical difference is that a SICS does not imitate but *induce* an agent to imitate or more generally to act consistently with another group of agents. In the broad area of web personalization a relevant work is the one by Paliouras et. al. [12] who clusters communities of on-line users. In the Implicit Culture perspective the groups are given, consequently an integration of the methods would be interesting. Finally, our approach can be seen is an attempt of supporting organizational learning in the direction proposed in [16].

3 Implicit Culture

When an agent begins to act in an environment without enough knowledge or skills, its behavior will be far from optimal. The problems that the new agent has to face are even more complex if some other agents are active in the same environment. They would probably have more knowledge and would be more skilled. Moreover, they might not be willing to share their knowledge and sometimes not even able to represent or communicate it. In order to improve its behavior, the new agent should act consistently with the culture of the other agents. In fact, in this "new kid in town" scenario the agent is enable to cope with the environment and with the other agents. More depressingly, a group of agents have the knowledge and actively exploit it. In the case of humans the phenomenon is sometimes referred as "cultural shock". Indeed, knowledge about the environment and about the behavior of the agents is part of their culture and that is what the new agent lacks.

The problem of having the new agent acting consistently with the knowledge and the behaviors of the group could be solved by improving the capabilities of the agent in terms of communication, knowledge and learning. The first solution is just "to ask someone", but in a multi-agent setting, this is *not* a simple solution. In fact, it is necessary to know what to ask (knowledge about the problem), how to ask (a language for expressing the problem), and who to ask to (some brokering facility). The second possible solution is representing the relevant knowledge and providing it to the agent. If the knowledge required is objective and relatively static, the representation can be done observing the environment and describing it. Building ontologies is a common way of addressing this problem. Unfortunately, the environment can be partially unknown and intrinsically dynamic. As a third option, it is possible to equip the agent with both observational and learning capabilities and acquire skills by imitation of the other agents. As a drawback, these capabilities are rather complex and their application requires resources.

When the environment is partially under control, the problem can be tackled in a very different way. Instead of working on the agent capabilities, it is possible to modify the view that the agent has of the environment and consequently its actions. In fact, changing in a proper way the set of possible actions that the agent can perform in the environment can lead the agent to act consistently with the behavior that a member of the group would have. The group itself can have optimized its behavior on the particular environment. Moreover, neither the new agent nor a member of the group is required to know about it and so they share the same culture in an implicit way.

Implicit Culture is the relation between a set and a group of agents such that the elements of the set behave according to the culture of the group. In the following, we informally introduce this notion, whereas in Appendix A, we recall the formal definition of Implicit Culture presented in [2, 3].

We assume that the agents perceive and act in an environment composed of objects and other agents. In this perspective, agents are objects that are able to perceive, act and, as a consequence of perception, know. Actions have as arguments objects, agents or both objects and agents. `Smooth(data-1)`, `look_for(learner)` and `run(learner-1,data-1)` are examples of the three different cases, respectively. Before executing an action, an agent faces a scene formed by a part of the environment, namely objects and agents, and actions that are possible in it (as a particular case, the agent can be part of the scene when reflexive actions are possible). For example, an agent `miner-1` faces `learner-1`, `learner-2`, `data-1`, `data-2` and can perform `run(learner-1,data-1)`, `run(learner-1,data-2)`, `run(learner-2,data-1)`, and `run(learner-2,data-2)`. Hence, an agent executes an action in a given situation, namely the agent facing a scene at a given time, so the agent executes situated actions. For instance, the agent `miner-1` executed the action `run(learner-2,data-1)` in a situation such that it was facing the scene composed of `learner-1`, `learner-2`, `data-1`, `data-2`.

After a situated action has been executed, the agents face a new scene. At a given time the new scene depends on the environment and on the situated executed action. If `miner-1` performs `run(learner-2,data-1)`, the `miner-1` will have the scene it faces changed because `learner-2` is now busy and not available to perform any service.

The situated executed action that an agent chooses to execute depends on its private states and, in general, it is not deterministically predictable with the information available externally. Rather, we assume it can be characterized in terms of probability and expectations. As an example, given a `miner` facing a scene in which it can perform `run(the-best-learner,data-1)`, `run(the-worst-learner,data-1)` the expected situated action can be `run(the-best-learner,data-1)`.

Given a group of agents let us suppose that there exists a theory about their expected situated actions. If the theory is consistent with the executed actions of the group, it can be considered a cultural constraint for the group. The theory captures the knowledge and skills of the members about the environment. For instance:

$$\begin{aligned}
&\exists \mathtt{y} \in \mathtt{Learners}: \\
&\quad \forall \mathtt{x} \in \mathtt{Miners}, \mathtt{d} \in \mathtt{datasets}, \mathtt{l} \in \mathtt{learners} \\
&\quad \mathtt{run}(\mathtt{x},\mathtt{y},\mathtt{d}) \wedge \mathtt{send-partial-results}(\mathtt{y},\mathtt{x},\mathtt{l}) \rightarrow \mathtt{resume-run}(\mathtt{x},\mathtt{l},\mathtt{d})
\end{aligned} \tag{1}$$

expresses that, there exists an agent `y` of the group of learners such that, if a miner `x` runs `y` on the data `d` and `y` sends partial results to `x`, then `x` will request `l` to resume the run on `d`. This means that the results provided by `y` encounters the favour of `x`.

If a set of new agents performs actions that satisfy the cultural constraints of the group (we call them cultural actions w.r.t. the group) the problem of their suboptimal behavior with respect to the group is solved. We call Implicit Culture a relation between a set of agents G' and a group G such that the agents of G' perform actions that satisfy a cultural constraint for G. When a set and a group of agents are in Implicit Culture relation, we have an Implicit Culture phenomenon. As a consequence of an Implicit Culture phenomenon, the actions of a new `miner` will be far more effective if it faces only the `learners` that other agents previously run on similar problems.

4 A SICS architecture

The definition of Implicit Culture (Appendix A briefly discusses the use of "implicit") does not give sufficient conditions for its realization, posing the problem of its support in practise. The general architecture of a SICS proposed in [2] allows to achieve the goal of estabilishing an implicit culture phenomenon following two steps. First, the elaboration of a cultural constraint theory $\mathbf{\Sigma}$ from a given domain and a set of situated executed actions of a group G. Second, the proposal to a group G' of a set of scenes such that the expected situated actions of the set of agents G' satisfies $\mathbf{\Sigma}$. Both the steps present learning problems.

A general SICS (see Figure 1-a) consists of three components: observer, inductive module and composer. The observer stores the situated executed actions of a group of agents G in oder to make them available for the other components. The inductive module uses these actions to produce a cultural constraint theory $\mathbf{\Sigma}$ for G. Finally, the composer, using the theory $\mathbf{\Sigma}$ and the actions, manipulates the scenes faced by a set of agents G' in such a way that their expected situated actions are cultural action w.r.t G. As a result, the agents of G' executes (on average) cultural actions w.r.t G (implicit culture phenomenon).

In Figure 1-a the composer proposes to the agents a, b, and c the scenes σ_{t+1}, σ'_{t+1}, and σ''_{t+1}, respectively. Notice that in this case the agents b and c belong to both G and G'. This means that also their situated actions are stored in DB and thus they are used to elaborate the theory $\mathbf{\Sigma}$ and the new scenes.

In general, our implemented architecture accepts cultural theories expressed by a set of rules of the form:

$$A_1 \wedge \cdots \wedge A_n \rightarrow C_1 \wedge \cdots \wedge C_m$$

in which $A_1 \wedge \cdots \wedge A_n$ is said antecedent and $C_1 \wedge \cdots \wedge C_m$ consequent. The idea is to express that "if in the past the antecedent has happened, then there exists in the future some scenes in which the consequent will happen". Antecedent and consequent are conjunctions of atoms, namely two types of predicates: observations on an agent and conditions on times. For instance, $\texttt{run(x,y,s,t}_1\texttt{)}$ is a predicate of the first type, that says that the miner `x` runs the learner `y` on the data `d` at time $\texttt{t}_1$; whereas $\texttt{less(t}_1\texttt{,t}_2\texttt{)}$ is an example of the second type and it simply says that $\texttt{t}_1\texttt{<t}_2$.

The composer proposes to a set of agents G' a set of scenes such that their expected situated actions satisfy a cultural constraint theory $\mathbf{\Sigma}$ for a group G. The main idea is splitting the problem in two sub-problems: (i) find the cultural actions and (ii) find the scenes where such actions are the expected situated actions. Figure 1-b shows the composer in detail. Basically, the composer consists of two main submodules and an additional component:

- the *Cultural Actions Finder* (CAF), that takes as inputs the theory $\mathbf{\Sigma}$ and the executed situated actions of G', and produces as output the cultural actions w.r.t. G (namely, the actions that satisfy $\mathbf{\Sigma}$). The CAF matches the executed situated actions of G' with the antecedents of the rules of $\mathbf{\Sigma}$. If it finds an action that satisfies the antecedent of a rule, then it takes the consequent of the rule as a cultural action.

– the *Scenes Producer* (SP), that takes one of the cultural action produced by the CAF and, using the executed situated actions of G, produces scenes such the expected situated action is the cultural action.
– the *Pool*, an additional component, which manages the cultural actions given as input from the satisfaction submodule. It stores, updates, and retrieves the cultural actions, and solves possible conflicts among them.

The SICS architecture requires the solution of two learning problems. A problem of induction of the cultural constraint theory (Inductive Module) and a problem of prediction of scenes (Scenes Producer).

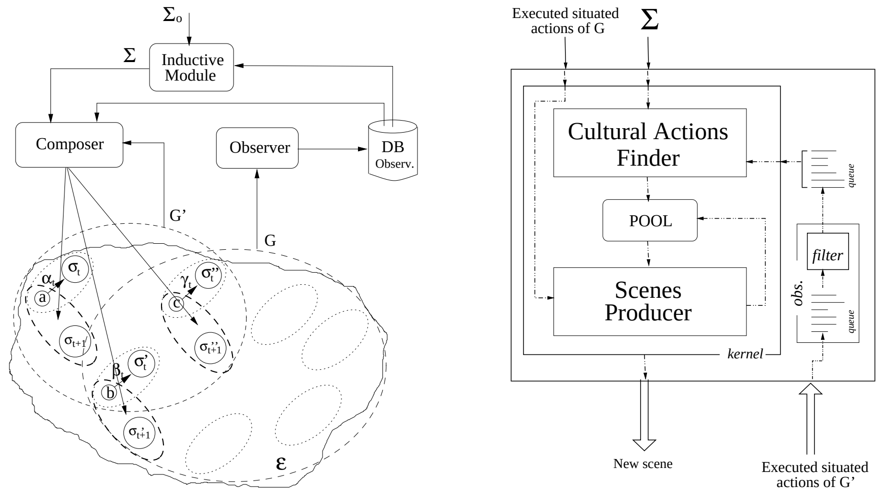
Definition 1 (Inductive Module Problem). *Given a set of situated executed actions performed by the agents of G, find a cultural constraint theory.*

Definition 2 (Scene Producer Problem). *Given a set of situated executed actions of the agents of G and G', and given a cultural action α for the agent x, find a scene s such that the expected situated action of x in the scene s is α.*

The Inductive Module Problem is a rather standard learning problem: inducing the behavior patterns of a group. As we previously noted, we do not require any specific validation of the theory. Obviously, very different effects will be reached depending on the fact that the theory is validated or not. In the Scene Producer Problem the request is on the effectivenes of the scene w.r.t the goal of producing the execution of a given action, namely its *persuasiveness*. In [10] we give of this problem a solution based on a generalization of a memory-based collaborative algorithm.

Posing data mining in the Implicit Culture framework and exploiting the SICS architecture can solve some of the problems encountered in the practice of KDD. Exploring this scenario is a first step towards An Implicit Culture Architecture for data mining. In fact, in the case of KDD the environment is composed by miners (i.e., users), learners (i.e., learning algorithms) , data and parameters and a big set of actions like run, test, select, etc. that the users can do on it. Miners, learners and data could be possibly be distributed. Supporting Implicit Culture means to have the miners act like the others, more expert miners and to have the learners taking in account their and other learner experiences acquired on other runs.

A Data mining application of Implicit Culture presents several advantages. The first, is that no miner is required to know about the others and about other applications and this is an important effect from the usability point of view and with respect to concerns about privacy and data security. This is in a general advantage given by Implicit Culture. Moreover, in a data mining application would be possible to exploit the tremendous amount of knowledge acquired by the meta-learning community. In fact, the inductive module will learn about both



(a) Architecture (b) The composer in detail

Fig. 1.

miners behavior and learners behavior or an interesting mix of the two. Learning using a mix of observations could lead, for example, to empirical definitions of "acceptable performance" or "interesting rule". Another advantage would be that the data mining user is familiar with the learning techniques required by the SICS and hopefully she can consider them acceptable. Implicit knowledge transfer within the agents is what an effective SICS achieves and KDD process can really take an advantage from it.

5 Conclusions and future work

We suggested to approach the Knowlege Discovery in Databases process in an Implicit Culture framework. We suggested that, building a SICS for Data mining application could be a viable way of solving some of the problems met in the practice of data mining. Of course, deeper analysis of the data mining domain in terms of objects and actions performed is required and the practical realization of these ideas requires a big and interdisciplinary effort.

APPENDIX A: Formal Definition of Implicit Culture

We consider *agents* and *objects* as primitive concepts to which we refer with strings of type *agent_name* and *object_name*, respectively. We define the *set of agents* $\mathcal{P}$ as a set of *agent_name* strings, the *set of objects* $\mathcal{O}$ as a set of *object_name* strings and the *environment* $\mathcal{E}$ as a subset of the union of the set of agents and the set of objects, i.e., $\mathcal{E} \subseteq \mathcal{P} \cup \mathcal{O}$.

Let *action_name* be a type of strings, E be a subset of the environment ($E \subseteq \mathcal{E}$) and s an *action_name*.

Definition 3 (action). *An action α is the pair $\langle s, E\rangle$, where E is the argument of α ($E = arg(\alpha)$).*

Let $\mathcal{A}$ be a set of actions, $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{E}$.

Definition 4 (scene). *A scene σ is the pair $\langle B, A\rangle$ where, for any $\alpha \in A$, $arg(\alpha) \subseteq B$; α is said to be* possible in σ. *The* scene space $\mathcal{S}_{\mathcal{E},\mathcal{A}}$ *is the set of all scenes.*

Let T be a numerable and totally ordered set with the minimum t_0; $t \in T$ is said to be a *discrete time*. Let $a \in \mathcal{P}$, α an action and σ a scene.

Definition 5 (situation). *A situation at the discrete time t is the triple $\langle a, \sigma, t\rangle$. We say that a* faces *the scene σ at time t.*

Definition 6 (execution). *An* execution *at time t is a triple $\langle a, \alpha, t\rangle$. We say that a* performs *α at time t.*

Definition 7 (situated executed action). *An action α is a* situated executed action *if there exists a situation $\langle a, \sigma, t\rangle$, where a performs α at the time t and α is possible in σ. We say that a performs α in the scene σ at the time t.*

When an agent performs an action in a scene, the environment reacts proposing a new scene to the agent. The relationship between the situated executed action and new scene depends on the charateristics of the environment, and in particular on the laws that describe its dynamics. We suppose that it is possible to describe such relationship by an environment-dependent function:

$$F_{\mathcal{E}} : A \times \mathcal{S}_{\mathcal{E},\mathcal{A}} \times T \to \mathcal{S}_{\mathcal{E},\mathcal{A}} \tag{2}$$

Given a situated executed action α_t performed by an agent a in the scene σ_t at the time t, $F_{\mathcal{E}}$ determines the new scene σ_{t+1} $(= F_{\mathcal{E}}(\alpha_t, \sigma_t, t))$ that will be faced at the time $t+1$ by the agent a.

While $F_{\mathcal{E}}$ is supposed to be a deterministic function, the action that an agent a performs at time t is a random variable $h_{a,t}$ that assumes values in $\mathcal{A}$.

Let $a \in \mathcal{P}$ and $\langle a, \sigma, t\rangle$ be a situation.

Definition 8 (expected action). *The* expected action *of the agent a is the expected value of the variable $h_{a,t}$, that is $E(h_{a,t})$.*

Definition 9 (expected situated action). *The* expected situated action *of the agent a is the expected value of the variable $h_{a,t}$ conditioned by the situation $\langle a, \sigma, t\rangle$, that is $E(h_{a,t} | \langle a, \sigma, t\rangle)$.*

Definition 10 (party). *A set of agents $G \subseteq \mathcal{P}$ is said to be a* party.

Let $\mathcal{L}$ be a language used to describe the environment (agents and objects), actions, scenes, situations, situated executed actions and expected situated actions, and G be a party.

Definition 11 (cultural constraint theory). *The* Cultural Constraint Theory for G *is a theory expressed in the language $\mathcal{L}$ that predicates on the expected situated actions of the members of G.*

Definition 12 (group). *A party G is a* group *if exists a cultural constraint theory $\mathbf{\Sigma}$ for G.*

Definition 13 (cultural action). *Given a group G, an action α is a* Cultural Action w.r.t. *G if there exists an agent $b \in G$ and a situation $\langle b, \sigma, t \rangle$ such that*

$$\{E(h_{b,t}|\langle b, \sigma, t \rangle) = \alpha\}, \mathbf{\Sigma} \not\vdash \perp$$

where $\mathbf{\Sigma}$ is a cultural constraint theory for G.

Definition 14 (implicit culture). Implicit Culture *is a relation $\rtimes\!\!\!\lhd$ between two parties G and G' such that G and G' are in relation ($G \rtimes\!\!\!\lhd G'$) iff G is a group and the expected situated actions of G' are cultural actions w.r.t G.*

Definition 15 (implicit culture phenomenon). Implicit Culture Phenomenon *is a pair of parties G' and G related by the Implicit Culture.*

We justify the "implicit" term of implicit culture because its definition makes no reference to internal states of the agents. In particular, there is no reference to believes, desires or intentions and in general on epistemic states or to any knowledge about the cultural constraint theory itself or even the two composition of the two groups. In the general case, the agents do not anything explicitly in order to produce the phenomenon.

References

1. D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proceedings of the International Conference on Machine Learning*, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers.
2. E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture. In *Proceedings of the Workshop on Agents and Reccomender Systems*, Barcellona (http://www.science.unitn.it/ pgiorgio/ic), 2000.
3. E. Blanzieri P. Giorgini and F. Giunchiglia. mplicit culture and multi-agent systemss. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, (SSGRR 2000)*, L'Aquila - Italy (http://www.science.unitn.it/ pgiorgio/ic), 2000.
4. C. Giraud-Carrier and B. Pfahringer, editors. *Proceedings of the ICML99 workshop on Recent Advances in Meta-Learning and Future Work*. 1999.
5. D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
6. J. Han and M. Kamber. *Data mining concepts and techniques.* Morgan Kaufmann, 2001.
7. J. Hu and M. Wellman. Multiagent reinforcement learning theoretical framework and an algorithm. In *Proceedings of the ICML*, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers.

8. J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
9. E. Blanzieri P. Giorgini P. Massa and S. Recla. Implicit culture for multi-agent interaction support. In *Proc. of the Int. Conf. on Cooperative Information Systems COOPIS*, 2001.
10. E. Blanzieri P. Giorgini P. Massa and S. Recla. Information access in implicit culture framerwork. Technical report, ITC-irst, 2001.
11. I. Nonaka and H. Takeuchi. *The knowledge Creating Company.* Oxford University Press, New York, 1995.
12. G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. Spyropoulos. Clustering the users of large web sites into communities. In *Proceedings of the ICML*, Stanford, 2000. Morgan Kaufmann Publishers.
13. B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proceedings of the ICML*, Bled, 1999. Morgan Kaufmann Publishers.
14. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, 1994.
15. P. Stone and M. Veloso. Multiagent systems: A survey from amachine learning perspective. *Autonomous Robots*, 8(3):345–383, June 2000.
16. K. Takadama, T. Terano, and K. Shimohara. Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents*, 1999.