Classifying Agent Behaviour through Relational Sequential Patterns

Grazia Bombini, Nicola Di Mauro, Stefano Ferilli, and Floriana Esposito

University of Bari "Aldo Moro", Department of Computer Science, 70125 Bari, Italy
{gbombini,ndm,ferilli,esposito}@di.uniba.it

Abstract. In Multi-Agent System, observing other agents and modelling their behaviour represents an essential task: agents must be able to quickly adapt to the environment and infer knowledge from other agents' deportment. The observed data from this kind of environments are inherently sequential. We present a relational model to characterise adversary teams based on its behaviour using a set of relational sequences in order to classify them. We propose to use a relational learning algorithm to mine meaningful features as frequent patterns among the relational sequences and use these features to construct a feature vector for each sequence and then to compute a similarity value between sequences. The sequence extraction and classification are implemented in the domain of simulated robotic soccer, and experimental results are presented.

Key words: Sequence Data Mining, Sequence Classification, Relational Sequence Similarity, Adversary Classification, Group Behaviour

1 Introduction

Sequence classification is still considered a problem rather general with application in computer-user modelling, text categorisation, intrusion detection and agent modelling. In Multi-Agent System (MAS), observing other agents and modelling their behaviour (Agent Modelling) represents an essential task. The main idea is to infer knowledge from other agents' deportment to identify the future behaviours of opponents in order to coordinate and cooperate with them or counteract their actions. In multi-agent adversarial environment, the adversary consists of a *team* of opponents that may interfere with the achievement of goals. In this domains agents must be able to adapt to the environment, especially to current opponents' sequences of actions. Indeed, Humans usually try to predict the behaviour of others in order to make a good decision.

The observed data from this kind of environments are inherently sequential, and hence it is necessary to have a mechanism able to handle sequential data. Given a set of labelled training sequences, after the learning phase, the aim of a classifier consists in predicting the class label of an unlabelled sequence. Finding useful features able to characterise a sequence represents the main difficulty. We propose to use a relational learning algorithm to mine meaningful features, as

frequent patterns, among the relational sequences useful to construct a feature vector for each sequence and to compute a similarity value between sequences.

In this paper we consider the problem of identifying agent behaviour in complex domains where it is necessary to consider enormous and possibly continuous state and action spaces. In dynamic environments, the agents have limited time to reason before to choose an action. In order to respond to a new situation, an agent must be able to quickly adapt its deportment to the context. For an effective adaptation, corresponding to adopt the most effective strategy, it is necessary to capture the similarity between observed behaviours. Hence, a mechanism for distinguishing different behaviours and classifying them to recognise different adversary classes is essential. From this point of view, a key challenge is to determine what are the sequential behaviours characterising a team.

Our proposal is to extract from row multi-agent observations of a dynamic and complex environment, a set of relational sequences describing an agent behaviour. Then, the aim is to propose a method to classify different agents using those relational sequences. In particular, this paper focuses on the tasks of sequence classification using a logic representation for the sequences and the extracted feature. The goal of this paper has several aspect:

- abstracting useful atomic actions (*events*) from the multi-agent system log files;
- recognising relational sequences of events that characterise the behaviour of a team;
- extracting useful feature from relational sequences;
- defining a similarity value between two feature-based sequence descriptions;
- comparing different teams' behaviour by means classification.

2 Related work

In competitive domains, the knowledge about the opponent can be very advantageous. To adapt the own team strategy, in domain such as RoboCup Simulation League [1], it is necessary to model and classify the adversary team.

In this research area, Riley and Veloso [2] used a Bayes model to predict opponent movement during some team matches. The learnt model is then used to generate adaptive plans to counter the opponents strategy. In addition, Riley and Veloso [3] modelled high-level adversarial behaviour by classifying the current opponent team into predefined adversary classes. Their system could classify fixed duration windows of behaviour using a set of sequence-invariant action features. An observation occurs over a fixed length of time (i.e. a window) and it affects the accuracy of the classifier and its performance.

Based on these works, then in [4] and [5] has been proposed a similarity-based and a feature-based declarative opponent modelling framework in MAS. Some features of the opponent that describe its behaviour could be extracted and formalised. Such features could then be used to recognise an opponent model and to identify tactical moves of the opponent. In [5] observations of the opponents are stored as state-actions pairs into a case base, distinguishing each team. Then

a similarity measures is used to determine whether a new scenario is similar to one in the case base.

In [6], the focus is on the unsupervised autonomous learning of the sequential behaviours of agents from observations of their behaviour. The authors use a hybrid approach to produces time-series of recognised atomic behaviours from observations of a complex and continuous multi-variate world state. This system is able to identify different events corresponding to the basic behaviours of the agents using a set of models specialised on recognising simple and basic behaviours of the agents (e.g., intercept, pass).

In [7] has been proposed a method for modelling the behaviour of individual opponent agents based on their observed input and outputs behaviour, described by low-level actions. Decision trees are learnt to predict the action type of a player.

Similarly to the previous approach, in [8], a symbolic approach based on association rule mining has been presented. The proposed method creates patterns from dynamic scenes using the qualitative information of the environment and the produces a set of prediction rule.

Finally, in [9] has been proposed an approach for recognising and classifying an observed team behaviour. A set of subsequences of events represented by flat symbols, defines a team behaviour. Trie data structures are used to store and classify the the behaviour patterns.

The difference of our method when compared to those previous works regards the logical representation language used to model the sequences and then the proposal of a distance measure between agent behaviours described as logical sequences.

3 Sequence Events Recognition

This section provides a description of the approach we use to extract relational sequence able to describe and characterize the behaviour of a team of agents.

The Robotic Soccer Domain is one of the most interesting environment where the agent modelling techniques has been used. The Robot World Cup Initiative[1] (RoboCup) [1] is a international joint project aiming to bring together researcher from Artificial Intelligence, Robotics and related fields. In an adversarial environment, the predicted behaviour of other agents is referred to as an opponent model. Soccer game is used as a central topic of research. One setting in which opponent modelling research has been conducted is the *RoboCup Simulation league*. In particular, RoboCup Simulation League 2D use Soccer Server System, a client-server system for simulating soccer. The server keeps track of the current state of the world, and stores all the data for a given match in a log file. This represents a stream of consecutive raw observations representing, for each moment of the match, soccer player's positions, position of the ball, ball possessor, and so on. In the soccer domain, it is possible to recognise basic actions

[1] http://www.robocup.org/

such as passing, shooting to the goal, dribbling an opponent player, intercepting the ball, etc. Each team may be recognised with some basic actions used to form coordinated activities attempting to achieve the team goals. Our system is able to recognise this basic action (*high-level events*) from row observations of a stream of multi-agent behaviour. The log stream is processed to infer the high-level events occurred during a match. The aim is to be able to classify a team observing his actions, and hence it is necessary to consider sequences of actions performed by players of the same team.

An event takes place when a ball possession changes or the ball is out of bounds. The event sequence of one team is separated from the events of its opponent team. A sequence is composed by an uninterrupted consecutive high-level events executed by the team in question. In our work, we identify the following high-level events:

- $catch(Player_n, T)$: if $Player_n$ is a goalkeeper and catches the ball close to the penalty box at time T;
- $pass(Player_n, Player_m, T)$:if $Player_n$ kicks the ball and $Player_m$ gains possession, and both the players are of the same team at time T;
- $dribble(Player_n, T)$: if $Player_n$ moves a significant distance since an opponent gains possession of the ball at time T;
- $intercept(Player_n, T)$: if $Player_n$ gains possession at time T, and the previous ball owner belongs of the opponent team;
- $shoot(Player_n, T)$: if $Player_n$ kicks the ball close to the penalty box of the opposite team at time T;
- $outside(Player_n, T)$: if $Player_n$ kicks the ball at time T, and the ball go out of the bounds;
- $goal(Player_n, T)$: if the $Player_n$ kicks the ball at time T, and the ball go in the goal.

Moreover, for each event the system takes into account some description related to the actions and its players:

- $dash(Player_n, T)$: if $Player_n$ at time T dashes;
- $neck(Player_n, T)$: if $Player_n$ at time T looks around;
- $turn(Player_n, T)$: if $Player_n$ at time T changes direction;
- $kick(Player_n, T)$: if $Player_n$ at time T is a ball owner and kicks the ball;
- $chngview(Player_n, T)$: if $Player_n$ at time T changes view.

For instance, given a stream of row observations about soccer player positions and the position of the ball, the system may produce a set of sequences related to the consecutive actions of the players of the same team until the opponent team gains possession of the ball. For instance:

$\{pass(Pl_1, Pl_2, T_1), kick(Pl_1, T_1), neck(Pl_1, T_1), chngview(Pl_2, T_1),$
$dribble(Pl_2, T_2), dash(Pl_2, T_2), shoot(Pl_2, T_3), kick(Pl_2, T_3)\}$

The consecutive actions of players belonging to same team composes a relational sequence. The sequence starts from a recognised action (*event*) for a team. For example a typical event may be $intercept(Pl_n, T)$ or $pass(Pl_n, Pl_m, T)$. Each successive recognised event performed by the same team belongs top the sequence until the opposing team gains the ball possession or the ball is out of bounds. Then the generation of a new sequence starts. Sequences generated from one

action are not relevant for the classification task, since this kind of sequences are the most frequent for all the teams. On the contrary, an interesting sequence is composed at least from two successive actions performed by players of the same team. Sequences represent a symbolic abstraction of the row observation. A set of sequences is created for each team, and this set characterises the observed teams. Separated models are learned for each team. The result of this phase is a set of the most meaningful relational sequences of recognized events that describes each team.

4 Relational Sequential Patterns for Agent Classification

In this section we present a method based on relational pattern mining, to extract meaningful features able to represent relational sequences and a distance function to measure the dissimilarity between two corresponding feature vectors. Finally, those distances will be used in the k-nearest neighbour (k-NN) algorithm to classify the adversary behaviour.

Logical Background: A relational sequence is represented by a set of Datalog [10] atoms, based on a first-order *alphabet* consisting of a set of *constants*, a set of *variables*, a set of *function symbols*, and a non-empty set of *predicate symbols*. Each function symbol and each predicate symbol has an *arity*, representing the number of arguments the function/predicate has. Constants may be viewed as function symbols of arity 0. An atom $p(t_1, \ldots, t_n)$ (or atomic formula) is a predicate symbol p of arity n applied to n terms t_i (i.e., a constant symbol, a variable symbols, or an n-ary function symbol f applied to n terms $t_1, t_2, \ldots, t_n$). A *ground term* or *atom* is one that does not contain any variables. A *clause* is a formula of the form $\forall X_1 \forall X_2 \ldots \forall X_n (L_1 \vee L_2 \vee \ldots \vee \overline{L}_i \vee \overline{L}_{i+1} \vee \ldots \vee \overline{L}_m)$ where each L_i is a literal and $X_1, X_2, \ldots X_n$ are all the variables occurring in $L_1 \vee L_2 \vee \ldots \overline{L}_i \vee \ldots \overline{L}_m$. Most commonly the same clause is written as an implication $L_1, L_2, \ldots L_{i-1} \leftarrow L_i, L_{i+1}, \ldots L_m$, where $L_1, L_2, \ldots L_{i-1}$ is the *head* of the clause and $L_i, L_{i+1}, \ldots L_m$ is the *body* of the clause. Clauses, literals and terms are said to be *ground* whenever they do not contain variables.

A *substitution* θ is defined as a set of bindings $\{X_1 \leftarrow a_1, \ldots, X_n \leftarrow a_n\}$ where $X_i, 1 \leq i \leq n$ is a variable and $a_i, 1 \leq i \leq n$ is a term. A substitution θ is applicable to an expression e, obtaining the expression $e\theta$, by replacing all variables X_i with their corresponding terms a_i. A conjunction A is *θ-subsumed* by a conjunction B, denoted by $A \preceq_\theta B$, if there exists a substitution θ such that $A\theta \subseteq B$. A clause c_1 *θ-subsumes* a clause c_2 if and only if there exists a substitution σ such that $c_1\sigma \subseteq c_2$. c_1 is a *generalization* of c_2 (and c_2 a *specialization* of c_1) under θ-subsumption. If c_1 θ-subsumes c_2 then $c_1 \models c_2$.

A *relational sequence* is an ordered list of atoms. Given a sequence $\sigma = (s_1 s_2 ... s_m)$, a sequence $\sigma' = (s'_1 s'_2 ... s'_k)$ is a *subsequence* (or *pattern*) of the sequence σ, indicated by $\sigma' \sqsubseteq \sigma$, if $1 \leq k \leq m$, $\exists j, 1 \leq j \leq m - k$ and a substitution θ s.t. $\forall i, 1 \leq i \leq k$: $s'_i\theta = s_{j+i}$. A subsequence occur in a sequence if exists at least a mapping from elements of σ' into the element of σ such that the

previous condition are hold. In our case, that subsequence is a *relational pattern*. The *support* of a sequence σ in a set of sequences $\mathcal{S}$ corresponds to the number of sequences in $\mathcal{S}$ containing the sequence σ: support(σ)= $|\{\sigma' | \sigma' \in \mathcal{S} \wedge \sigma \sqsubseteq \sigma'\}|$.

Relational Sequence Features: Before to design a classification method based on k-Nearest Neighbour (k-NN), it is necessary to define an appropriate *similarity measure* between sequences. The measure will be applied to data objects represented as a set of features expressing the special properties (features) of a sequence in a specific domain. A way to represent a sequence as a feature vector is to use the patterns occurring in the sequence as true features. In this section we introduced and translate the concept of k-grams and g-gapped to the relational case.

Given an alphabet of symbols $\mathcal{A}$, and let $k \geq 1$ be a positive integer, then a ***k*-gram** (k-mers) is a sequence σ of symbols over $\mathcal{A}$ of length k ($\sigma \in \mathcal{A}^k$, $|\sigma| = k$). For a given sequence $\sigma = (s_1 s_2 \dots s_t)$, the k-grams of interest are all subsequences $\sigma' = (s_i s_{i+1} \dots s_{i+k-1})$ of length k occurring in σ. Given a sequence $\sigma = (s_1 s_2 \dots s_t)$ with $|\sigma| = t$, we define K_σ as the set of all k-grams Ω_k of σ, $1 \leq k \leq t$:

$$K_\sigma = \bigcup_{k=1}^{t} \Omega_k = \bigcup_{k=1}^{t} \{\omega_{k1}, \omega_{k2}, \dots \omega_{kn_k}\}$$

where $\omega_{ki} = (s_i s_{i+1} \dots s_{i+k-1})$, and $n_k = t - k + 1$. Given a set of sequences $\mathcal{S} = \{\sigma_i\}_{i=1}^n$, $\mathcal{K}$ is the set of all k-grams on all the sequences belonging to $\mathcal{S}$ and represents a set of features over $\mathcal{S}$, where $\mathcal{K} = \sum_{i=1}^n |\sigma_i|(|\sigma_i| - 1)/2$.

Given an alphabet of atoms $\mathcal{A}$, a **relational *k*-gram** is a relational sequence σ of length k defined over $\mathcal{A}$. Given a set of relational sequences $\mathcal{S} = \{\sigma_i\}_{i=1}^n$, $\mathcal{K}$ is the set of all relational k-grams on all the sequences belonging to $\mathcal{S}$: $\mathcal{K} = \bigcup_{i=1}^n K_{\sigma_i}$ where K_{σ_i} is the set of all relational k-grams over the sequence σ_i.

Given a sequence $\sigma = (s_1 s_2 \dots s_n)$ defined over an alphabet $\mathcal{A}$, a ***g*-gapped** occurring in σ is defined as a sequence

$$\sigma' = (s_k s_{k+1} \dots s_{k+a} s_{k+a+g} s_{k+a+g+1} \dots s_{k+a+g+b}),$$

where $k \geq 1$ and $k + a + g + b \leq n$. In particular, a g-gapped is made up of two consecutive sub-sequences of σ separated by a gap. A gap of length 0 make the g-gapped a k-gram where $k = a + b$. Given a sequence σ with $|\sigma| = t$, we define G_σ the set of all g-gapped Ψ_g of σ, $1 \leq g \leq t - 2$: $G_\sigma = \bigcup_{g=1}^{t-2} \Psi_g$. Given a set of sequences $\mathcal{S} = \{\sigma_i\}_{i=1}^n$, $\mathcal{G}$ is the set of all g-gapped on all the sequences belonging to $\mathcal{S}$ and represents a set of features over $\mathcal{S}$. To use the concept of g-gapped in a relational domain, we have introduced in the background knowledge a relational operator *followat_**n*** that represents the n-th direct successor in the relational sequence. Fixing the value of n, ranging from 1 to a maximum given value, it is possible to define the gap between the subsequences.

Let be $\mathcal{A}$ an alphabet of atoms, a **relational *g*-gapped** is a relational sequence β defined over $\mathcal{A}$ in which there is almost an atom like *followat_**n***. G_{σ_i}

is the set of all relational g-gapped over the sequence belonging to $\mathcal{S}$:

$$\mathcal{G} = \bigcup_{i=1}^{n} G_{\sigma_i}$$

where G_{σ_i} is the set of all relational g-gapped over the sequence σ_i. In particular, $\mathcal{K}$ and $\mathcal{G}$ represent the set of all relational features over $\mathcal{S}$. We define $\mathcal{K}(\alpha) \subseteq \mathcal{K}$ and $\mathcal{G}(\alpha) \subseteq \mathcal{G}$, the set of relational k-grams and g-gapped having a support greater than α: $\mathcal{K}(\alpha) = \{\sigma | \sigma \in \mathcal{K} \wedge \mathit{support}(\sigma) \geq \alpha\}$ and $\mathcal{G}(\alpha) = \{\sigma | \sigma \in \mathcal{G} \wedge \mathit{support}(\sigma) \geq \alpha\}$.

Mining Relational Sequential Patterns: In order to select the best set of features, we use an Inductive Logic Programming (ILP) [11] algorithm , based on [12], for discovering relational patterns from sequences. It is based on a level-wise search method, known in data mining from the Apriori algorithm [13]. It takes into account the sequences, tagged with the belonging class, and the α parameter denoting the minimum support of the patterns. It is essentially composed by two steps, one for generating pattern candidates and the other for evaluating their support. The level-wise algorithm makes a breadth-first search in the lattice of patterns ordered by a specialization relation. Starting from the most general patterns, at each level of the lattice the algorithm generates candidates by using the lattice structure and then evaluates the frequencies of the candidates. Since the monotonicity of pattern frequency (if a pattern is not frequent then none of its specializations is frequent), in this phase some patterns may be discarded.

The generation of the patterns actually present in the sequences of the dataset, is based on a top-down approach. The algorithm starts with the most general patterns. These initial patterns are all of length 1 and are generated by adding an atom to the empty pattern. Then, at each step it tries to specialize all the potential patterns, discarding those that do not occur in any sequence and storing the ones whose length is equal to the user specified input parameter *maxsize*. Furthermore, for each new refined pattern, semantically equivalent patterns are detected, by using the θ-subsumption relation, and discarded. In the specialisation phase, the specialisation operator under θ-subsumption is used. Basically, the operator adds atoms to the pattern. Finally, the algorithm may use a background knowledge $\mathcal{B}$ (a set of Datalog clauses) containing constraints on how to explore the lattice.

Distance Function over Relational Sequences: Given a set of sequences $\mathcal{S}$, we apply the algorithm previously described [12], to find all the relational k-grams $\mathcal{K}(\alpha)$ and relational g-gapped $\mathcal{G}(\alpha)$ over the set $\mathcal{S}$ with a support at least equal to α. The ordered set of *features* $\mathcal{F}$ that will be used to compute the boolean vector representation of each sequence is defined in the following way. Given a sequence $\sigma \in S$, and $\mathcal{F} = \{\mu_i\}_{i=1}^{n}$ where $\mu_i \in \mathcal{K}(\alpha) \cup \mathcal{G}(\alpha)$ represents the set of relational k-grams and g-gapped over $\mathcal{S}$, the *feature vector* of σ is

$$V_\sigma = (f_1(\sigma), f_2(\sigma), ..., f_n(\sigma)) \qquad \text{where} \qquad f_i(\sigma) = \begin{cases} 1 \ \mathit{if} \ \mu_i \sqsubseteq \sigma \\ 0 \ \mathit{otherwise} \end{cases}$$

Now, the distance function $d_r(\cdot,\cdot)$ between two relational sequences σ_1 and σ_2 is computed using the classical Tanimoto measure [14]:

$$d_{r_1}(\sigma_1,\sigma_2) = \frac{n_{1\sigma_1} + n_{1\sigma_2} - 2n_{1\sigma_{12}}}{n_{1\sigma_1} + n_{1\sigma_2} - n_{1\sigma_{12}}} = \frac{2(n - n_{1\sigma_{12}})}{2n - n_{1\sigma_{12}}} \tag{1}$$

where $n_{1\sigma_i} = n = |\mathcal{F}|$ is the number of the features, and $n_{1\sigma_{12}} = |\{f_i | f_i(\sigma_1) = f_i(\sigma_2)\}|$ is the number of features with the same value in both σ_1 and σ_2. However, this basic formulation takes into account features not appearing (with value 0) in the sequences, and in case of a lot of feature this can lead to underfitting.

Equation (1) may be extended in the following way:

$$d_{r_2}(\sigma_1,\sigma_2) = \frac{n_{2\sigma_1} + n_{2\sigma_2} - 2n_{2\sigma_{12}}}{n_{2\sigma_1} + n_{2\sigma_2} - n_{2\sigma_{12}}} = \sum_{i=1}^{n} \frac{f_i(\sigma_1) + f_i(\sigma_2) - 2f_i(\sigma_1)f_i(\sigma_2)}{f_i(\sigma_1) + f_i(\sigma_2) - f_i(\sigma_1)f_i(\sigma_2)}$$

where $n_{2\sigma_i} = \sum_{j=1}^{n} f_j(\sigma_i)$ is the number of the features holding in the sequence σ_i, and $n_{\sigma_{12}} = |\{f_i | f_i(\sigma_1) = f_i(\sigma_2) = 1\}|$ is the number of features that hold both in σ_1 and σ_2.

5 Experimental results

As discussed in Section 3, in order to evaluate our approach we analyse log files of soccer games of the RoboCup 2008 Exercise Competitions[2]. This is a preceding event for RoboCup initiative, and includes a 2D simulation league. We have implemented a system that is able to identify and extract the interesting sequences of coordinated team behaviours using the recorded observations (logs) of this simulation games.

There is an underlying assumption, that the strategy of a team does not change during the competition. We have analysed the log files for 4 teams, concerning to 4 matches of the competition, 2 matches for each team. One adversary class was created for each team by analysing the log files of two matches of the same team, producing a set of relational sequences. Each sequence is made up of interesting uninterrupted consecutive actions performed by players of the same team representing its characteristic behaviour. From the row observations of the log files we have obtained the dataset. It consists of 443 sequences, defined on 7 atomic behaviours (catch, pass, dribble, etc.) and 5 action descriptions (neck, turn, kick, etc.). In particular, we have 112 sequences for the first team $C0$, 106 sequence for the second team $C1$, 93 sequence for the third team $C2$ and 132 sequence for the fourth team $C3$.

After having created these adversary classes, the goal was to identify the team using a sequence regarding its actions. A weighted 10-NN classifier was constructed and tested using the 10-fold cross-validation to find the classification accuracy. In the first step, the set $\mathcal{K}(\alpha)$ of frequent k-grams has been mined. Here, α denotes the support of each k-gram $\sigma \in \mathcal{K}(\alpha)$ corresponding to the ratio $support(\sigma)/|\mathcal{S}|$, where $\mathcal{S}$ is the set of sequences in the training set for each fold. In

[2] http://robocup-cn.org/en/exercise/08/

Table 1. Classification accuracy using only k-grams as features and 10-fold cross-validation.

	Class	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean
$\alpha = 0.25$	C0	54.55	40	40	10	60	60	36.36	30	75	30	39.42
	C1	70	66.67	72.73	60	22.22	10	30	60	10	30	43.16
	C2	100	100	75	88.89	100	88.89	100	88.89	88.89	90	92.06
	C3	66.67	76.92	53.85	84.62	84.62	100	92.31	38.46	100	85.71	78.32
$\alpha = 0.20$	C0	36.36	50	40	20	50	70	72.73	40	33.33	40	42.24
	C1	90	75	72.73	40	22.22	20	20	50	10	40	43.99
	C2	100	100	83.33	88.89	100	88.89	100	88.89	100	90	94
	C3	80	92.31	100	76.92	92.31	100	92.31	46.15	100	85.71	88.75

Table 2. Classification accuracy using k-grams and g-gapped as features and 10-fold cross-validation.

	Class	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean
$\alpha = 0.25$	C0	54.55	40	50	20	90	70	18.18	40	22.22	40	44.49
	C1	80	58.33	81.83	60	77.778	10	40	50	20	30	50.79
	C2	100	100	100	88.89	100	100	100	88.89	77.78	90	94.56
	C3	73.33	76.92	76.92	69.23	69.23	100	62.62	61.54	100	85.71	80.52
$\alpha = 0.20$	C0	36.36	50	30	30	50	70	72.73	70	33.33	50	49.24
	C1	90	83.33	72.73	40	100	20	60	30	20	40	55.61
	C2	100	90	90	88.89	100	100	88.89	77.78	88.89	80	90.44
	C3	86.67	76.92	92.31	69.23	92.31	100	53.85	61.54	100	85.71	81.85

this experiment, α has been set to 0.20 and 0.25, and the algorithm extracted, respectively, 501.4 and 315.2 k-grams on average on the 10 fold. The average accuracy results was, respectively, 68.63 and 63.9. The results for different value of α are shown in Table1.

In the second experiment, maintaining the same values of α, both k-grams and g-gapped has been mined. In Table2 are reported the results, with an average accuracy results of, respectively, 69.29 and 67.76. The algorithm extracted, respectively, $1031,7$ and $781,8$ k-grams and g-gapped on average on the 10 fold. In all the experiments, there are 4 classes to be distinguish, and hence the accuracy on guessing is 25%. Considering this guessing accuracy as a baseline we can say that the proposed method is able to classify with a high accuracy the teams C2 and C3 and with a sufficient accuracy the teams C0 and C1.

6 Conclusions

In multi-agent adversarial environment, the adversary consists of a team of opponents that may interfere with the achievement of goals. In this domains agents must be able to adapt to the environment, especially to the current opponents' sequences of actions.

In this paper we propose a relational model to represent an adversary team based on its observed behaviour. A similarity measure and a classification ap-

proach for relational sequences has been applied to adversary classification. Experimental results obtained on RoboCup competition are encouraging. As a future work, we will investigate methods for extracting patterns with a high discriminative power, and we will compare different similarity functions.

References

1. Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., Asada, M.: The robocup synthetic agent challenge 97. In: IJCAI'97: Proceedings of the 15th international joint conference on Artifical intelligence, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1997) 24–29
2. Riley, P., Veloso, M.: Planning for distributed execution through use of probabilistic opponent models. (2002) 72–81
3. Riley, P., Veloso, M.M.: On behavior classification in adversarial environments. In Parker, L.E., Bekey, G.A., Barhen, J., eds.: DARS, Springer (2000) 371–380
4. Steffens, T.: Feature-based declarative opponent-modelling. In: RoboCup. (2003) 125–136
5. Steffens, T.: Similarity-based opponent modelling using imperfect domain theories. In: CIG. (2005)
6. Kaminka, G.A., Fidanboylu, M., Chang, A., Veloso, M.M.: Learning the sequential coordinated behavior of teams from observations. In Kaminka, G.A., Lima, P.U., Rojas, R., eds.: RoboCup. Volume 2752 of Lecture Notes in Computer Science., Springer (2002) 111–125
7. Ledezma, A., Aler, R., Sanchis, A., Borrajo, D.: Predicting opponent actions by observation. (2005) 286–296
8. Lattner, A.D., Miene, A., Visser, U., Herzog, O.: Sequential pattern mining for situation and behavior prediction in simulated robotic soccer. In: RoboCup 2005: Robot Soccer World Cup IX. (2006) 118–129
9. Iglesias, J.A., Ledezma, A., Sanchis, A., Kaminka, G.A.: Classifying efficiently the behavior of a soccer team. In et al., W.B., ed.: Intelligent Autonomous Systems 10. IAS-10. (2008) 316–323
10. Ullman, J.: Principles of Database and Knowledge-Base Systems. Volume I. Computer Science Press (1988)
11. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679
12. Esposito, F., Di Mauro, N., Basile, T.M.A., Ferilli, S.: Multi-dimensional relational sequence mining. Fundam. Inf. **89**(1) (2009) 23–43
13. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules, Menlo Park, CA, USA, American Association for Artificial Intelligence (1996) 307–328
14. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience (November 2000)