

# Adaptive Cooperative Fuzzy Logic Controller

Justin Ammerlaan

David Wright

School of Computing  
University of Tasmania,  
Churchill Avenue, Sandy Bay, Hobart,  
Email: jammer1a@utas.edu.au

## Abstract

Fuzzy logic is a natural basis for modelling and solving problems involving imprecise knowledge and continuous systems. Unfortunately, fuzzy logic systems are invariably static (once created they do not change) and subjective (the creator imparts their beliefs on the system). In this paper we address the question of whether systems based on fuzzy logic can effectively adapt themselves to dynamic situations.

To answer this question, we firstly design and implement an adaptive fuzzy logic agent for playing RoboCup soccer tournaments and then conduct a statistical analysis of the performance of the agent. We also extend the agent to incorporate adaptation in the cooperative situations that arise in this domain, and evaluate its performance under such conditions.

Results from our analysis conclusively prove that our adaptive fuzzy logic agent can adapt rapidly and successfully to the changing dynamic situations with which it is presented.

## 1 Introduction

Fuzzy Logic provides a simple method to empower computer systems with knowledge and reasoning ability that can handle imprecision, complexity and continuity. However, Fuzzy Logic systems, while easy to create, have subjective knowledge inherent in their construction process. The representation of this subjective knowledge is static; once it is created it does not adjust to suit the domain. Adaptation of this knowledge to provide a more optimal representation is one method of improving the output and allowing the system to respond to changes apparent in the domain.

Adaptive systems are typically implemented in the form of learning systems based on neural networks (Kruse & Nrnberger 1998). These systems, while producing good results in the application areas researched, have the burden of poor computational complexity relative to the number of inputs to the system, as well as being slow to adapt to new situations (Langholz 1992). This causes these approaches to be unsuited to an environment that requires *real time* adaptation with *limited* sample data and *communication* of discovered optimisations in the presence of real world communication restraints.

What then can be used to optimise the adaptation process of systems within these domains? This paper presents:

- a novel approach to on-line adaptation and optimisation using linguistic modifiers and a fuzzy logic based adaptive function;
- an implementation of cooperative adaptation that allows two or more agents to effectively communicate and cooperatively adapt via this online fuzzy logic adaptation system;
- a statistical verification of the performance and efficacy of the proposal.

## 2 Background

### 2.1 Fuzzy Logic Adaptive Techniques

To design a fuzzy controller, linguistic rules and membership functions to represent linguistic values must be determined. The specification of good linguistic rules typically depends on the knowledge of the domain expert. The translation of these linguistic rules into fuzzy set theory is not formalized and arbitrary choices have to be made. This includes choice of a membership function representing a linguistic term and the degree of overlap with other linguistic terms in the same fuzzy variable.

As an example, consider a temperature variable and the linguistic terms *hot*, *medium* and *cold*. If the linguistic term *hot* is considered, obviously the corresponding fuzzy set definitions should consist of a function reaching its maximum at the value the operator assigns as depicting *hot*. Neither the shape, the support of the membership function or the degree of overlap with the linguistic term *medium* is uniquely determined by the linguistic term *hot*. In general, the domain expert has some idea about the range and the shape of the membership function, however there is some imprecision in the boundaries and shape of the function (Pearson 2001).

#### 2.1.1 Neural Networks

Neural Networks have been suggested as a method of offline adapting and optimising fuzzy controllers to increase their performance. A straightforward approach is to assume a certain shape for the membership functions which depend on different parameters that can be learned by a Neural Network. This idea was carried out in (Nauck, Flawonn & Kruse 1992) where the membership functions are assumed to be symmetrical triangular functions depending on two parameters one of them determining where the function reaches its maximum the other giving the length of the support. Another approach is summarised in (Kruse & Nrnberger 1998). The literature is extensive regarding this topic, with many well regarded and successful implementations documented, a summary and bibliography of these can be found in (Nauck & Kruse 2002).

### 2.1.2 Neural Networks based Adaptation Issues

Neural networks, while providing a good general approach for adapting fuzzy logic parameters, have some limitations as follows.

Firstly, the knowledge contained within a neural network regarding adapted parameters is stored in the synapses of the interconnecting neurons. This method of storing adaptation information becomes expensive if a requirement of the implemented system is for information to be transmitted across multiple neural networks. It is also difficult to analyse this information and represent it in a comprehensible manner.

Secondly, regardless of the neural network learning system that is in place, whether it utilises reinforcement learning or a feed forward / backward learning system, each neural network is typically retrained on the receipt of new information regarding the domain. This fundamental retraining process causes neural networks to grow increasingly burdensome in terms of performance as the number of parameters to be adapted increases. This factor alone restricts such systems from being used in domains that require on-line adaptation or real-time execution.

Finally, neural networks used in the context of adapting specific parameters of a given system do not make use of any local or operator knowledge of potential optimisation paths for these parameters.

The solution examined in this paper proposes a method for solving these problems in the form of a fuzzy logic rule-base for adapting fuzzy logic parameters, which, as well as being capable of real-time execution, allows for the inclusion of local knowledge relevant to each individual set of fuzzy membership functions.

## 3 Adaptive Agent Implementation

### 3.1 Robocup

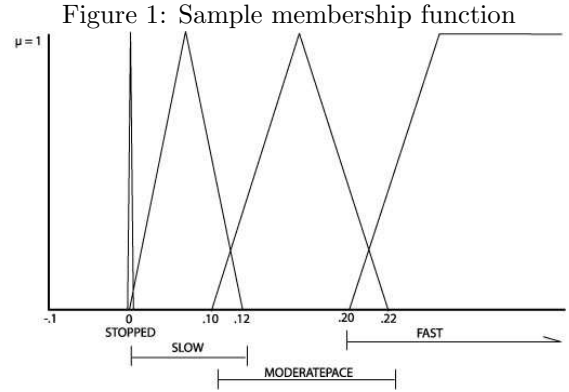
The *Robocup SoccerServer* is a system that enables autonomous agents consisting of programs written in various languages to play a match of soccer against each other (Chen 2002). The soccer server is written to support competition amongst multiple virtual soccer players in an uncertain multi-agent environment, with real-time demands as well as semi-structured conditions. A match is carried out in a client/server style. A server, *soccerserver*, provides a virtual field, communicates with clients, controls a game according to rules and simulates all movements of a ball and players.

### 3.2 Fuzzy Agents

The fuzzy logic system constructed as part of the research for this system, extends a simple interface for use in any domain. Each agent has access to the same set of:

1. operating ranges or fuzzy variables within the domain;
2. membership functions on those ranges; and
3. corresponding linguistic variables.

Examples of the input variables for the system are the agent based (those variables that relate to the agents physical properties) input variables such as *speed* and *stamina*; the object based input variables such as *ball distance* and *goal direction*; the inferred input variables such as *ball velocity* and *own player relative direction to object*. Examples of the output variables



for the System include *kick{distance, direction}* and *turn{direction}*. It can be observed that the Robocup domain is quite complex with many variables pertaining to each agents environment and state. The number of input variables within the system is  $\sim(400 \times 22)$ .

### 3.3 Inference Process

Consider the following multi-input, multi-output system: let  $x = (x_1, x_2, \dots, x_n)^T$  be the input vector and  $y = (y_1, y_2, \dots, y_m)^T$  be the output vector. The linguistic variable  $x_i$  in the universe of discourse  $U$  is characterized by  $T(x) = \{T_x^1, T_x^2, \dots, T_x^k\}$  and  $\mu(x) = \{\mu_x^1, \mu_x^2, \dots, \mu_x^k\}$  where  $T(x)$  is a term set of  $x$ ; that is, it is the set of names of linguistic values of  $x$ , with each  $T_x^i$  being a fuzzy member and the membership function  $\mu_x^i$  defined on  $U$ .

As an illustration of application within the fuzzy agent implementation, consider a simplified fuzzy inference system for controlling an agent's movement toward an object such as the ball. Let there be two inputs ( $n = 2$ ) and one output ( $m = 1$ ) with the two inputs representing the agents current speed (*speed*) and the distance to the ball (*distance*), and let the output of the System be power to be applied (*power*). Let  $x_1$  indicate the speed,  $T(x_1)$  represent its term set  $\{\text{stopped, slow, moderate\_pace, fast}\}$ , and the universe of discourse be  $[0 - .3]$ . Let  $x_2$  indicate the distance, with the universe of discourse,  $[0 - 100]$ , and the corresponding term set be  $\{\text{nextto, close, near, far}\}$ . Similarly, linguistic variable  $y$  in the universe of discourse  $V$  is characterized by  $T(y) = \{T_y^1, T_y^2, \dots, T_y^k\}$ , where  $T(y)$  is a term set of  $y$ ; that is,  $T$  is the set of names of linguistic values of  $y$ , with each  $T_y^i$  being a fuzzy membership function  $\mu_y^i$  defined on  $V$ .

If the variable  $y$  represents power, then  $T(y)$  represents a term set  $\{\text{no\_power, some\_power, moderate\_power, lots\_power, max\_power}\}$ , and the universe of discourse is  $[0 - 200]$ . In order to map input variables  $x_1$  and  $x_2$  and output variable  $y$  to the corresponding linguistic variables it is necessary that the fuzzy sets and linguistic variables be defined. A sample membership function for the input and output variables is shown in Figure 1.

The first step in evaluating the output of a fuzzy inference system is to apply the inputs and determine the degree to which they belong to each of the fuzzy sets. The fuzzifier block performs the mapping from the input feature space to fuzzy sets in a certain universe of discourse. A specific value  $x_1$  is then mapped to the fuzzy set  $T_{x_1}^1$  with degree  $\mu_{x_1}^1$ , to fuzzy set  $T_{x_1}^2$  with degree  $\mu_{x_1}^2$ , and so forth.

A fuzzy rule base contains a set of fuzzy rules  $R$ . A single *if ... then ...* rule assumes the form 'if  $x$

is  $T_x$  then  $y$  is  $T_y$ '. Some examples of rules might be 'if *speed* is *slow* and *distance* is *far* then *power* is *maxpower*', and 'if *speed* is *fast* and *distance* is *nextto* then *power* is *somewpower*'.

For a multi-input, multi-output system, where the  $i$ -th fuzzy rule is  $R_i = \text{if}(x_1 \text{ is } T_{x_1}, \text{ and } \dots, x_p \text{ is } T_{x_p}) \text{ then } (y_1 \text{ is } T_{y_1}, \text{ and } \dots, y_q \text{ is } T_{y_q})$  put is provided by:

The  $p$  preconditions of  $R_i$  form a fuzzy set  $T_{x_1} \times T_{x_2} \dots \times T_{x_p}$ , and the consequent is the union of  $q$  independent outputs. In a multi-input, single-output system, such as the current example, then the consequent reduces to ( $y \text{ is } T_{y_1}$ ).

Within the fuzzy logic agent implementation, interpreting an *if ... then ...* rule is a three part process:

1. Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1 based on the inputs  $x_1$  and  $x_2$
2. If there are multiple parts to the antecedent, such as linguistic modifiers, apply the relevant operators and resolve the antecedent to a single number between 0 and 1, i.e the result being the degree of support of this antecedent for the rule; and
3. Apply the implication method, using the degree of support for the entire rule to shape the output fuzzy set. If the rule has more than one antecedent, the fuzzy min operator is applied to obtain one number that represents the result of applying that rule.

For example, consider an  $i$ -th rule:

$$R_i : \text{if } x_1 \text{ is } T_1^i \text{ and } x_2 \text{ is } T_2^i \text{ then } y_1 \text{ is } T_y^i \quad (1)$$

Then the firing strength or membership of the rule can be defined as:

$$\alpha_i = \min(\mu_{x_1}^i(x_1), \mu_{x_2}^i(x_2)) \quad (2)$$

Equation 2 represents fuzzy intersection with the minimum operator. Each fuzzy rule yields a single number that represents the firing strength of that rule. The firing strength is then used to shape the output fuzzy set that represents the consequent part of the rule. The implication method is defined as the shaping of the consequent (the output fuzzy set), based on the antecedent. The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. The method used within the fuzzy logic agent implementation is the minimum method expressed in Equation 3.

$$\mu_y^i(w)' = \min(\alpha_i, \mu_y^i(w)) \quad (3)$$

where  $w$  is the variable that represents the support value of the membership function.

After the firing strengths of the rules have been obtained, the corresponding output fuzzy sets must be combined into one composite fuzzy set. The process of combining output fuzzy sets into a single set is called aggregation, a process that unifies the outputs of all the rules. Essentially, aggregation takes all fuzzy sets that represent the output for each rule and combines them into a single fuzzy set that is used as the input to the defuzzification process. Aggregation occurs only once for each output variable. The inputs to the aggregation process are truncated or modified output fuzzy sets which are obtained from the output of the implication process. The output of the aggregation process is a single fuzzy set that represents the output variable. Since the aggregation

method is commutative, the order in which the rules are executed is not important.

The method used for aggregation within the fuzzy logic agent implementation is the max method expressed in Equation 4. If there is two rules with output fuzzy sets represented by  $\mu_y^1$  and  $\mu_y^2$  then the output is provided by:

$$\mu_y(w) = \max(\mu_y^1(w), \mu_y^2(w)) \quad (4)$$

In order to get a crisp value for output  $y$ , a defuzzification process must be applied. The input to the defuzzification process is a fuzzy set (the aggregate output fuzzy set), and the output of the defuzzification process is a single crisp number. The method used for defuzzification in the fuzzy logic agent implementation is the centroid method. The centroid method is typically the most commonly used defuzzification method in the literature. It is the most widely used technique because, when it is applied, the defuzzified values tend to move smoothly around the output fuzzy region (Pearson 2001).

### 3.4 Setpiece Model

In order to divide the actions an agent may undertake into manageable segments, to remove the necessity for one large highly complex rule-base, and to introduce the concepts of action competition and adaptation, a setpiece based approach is undertaken. Setpieces are essentially a description of a particular action that an agent may perform as well as describing the world state in which a setpiece is applicable. Setpieces describe the actions to undertake if a given setpiece is chosen in that they are constructed for each type of play that is desired. For instance a setpiece for 'moving to the ball', which is based on certain conditions, may be implemented as described in the previous section in order to provide this ability to the agents.

#### 3.4.1 Setpiece Predictor

The setpiece predictor consists of a fuzzy rule base designed to predict the validity of firing the setpiece given the current world state. The setpiece predictor accepts a vector of input fuzzy variables which describe the current world state and applies them to the fuzzy rule base as described in the previous section. The output of this fuzzy rule-base is a fuzzy value that provides the prediction for success of this setpiece based on the current world state. This prediction value is then normalised in order to be comparable to other setpiece predictions.

#### 3.4.2 Setpiece Actioner

The actioner consists of a second fuzzy rule-base which instructs the system to perform a particular task specific to the requirements of the setpiece. This secondary fuzzy rule-base allows for a fine grained degree of control over the specific actions of a setpiece, with different fuzzy outputs being possible for slightly different world states.

#### 3.4.3 Setpiece Evaluator

The evaluator evaluates the current world state every cycle against a fuzzy rule base of the required state. The evaluator requires a description of the current world state as well as a description of the world state at the time of firing the setpiece. Additionally each setpiece evaluator requires a maximum time, *maxpayoff* time, after which, if the setpiece is

not successful, the evaluator can return a negative result. The evaluator consists of a fuzzy rule-base defining a fuzzy successful (or unsuccessful) result. This rule-base uses the current world state inputs for its evaluation. The evaluator also obtains the setpiece predictor confidence on the world state at the time of firing, in order to provide it as one of the inputs to the modification function detailed below.

**Modifier Function** The modification function is executed after the maximum amount of time allowed for a setpiece to evaluate to a positive result has expired. At this point the setpiece evaluator calculates the original setpiece prediction confidence and the fuzzy evaluation result, applying the average of the two to the selection table in Table 1 in order to obtain the modifier to be applied.

Table 1: Default modifier mappings

Func-tion	Modi-fier	Situation
0 - 30	extremely	Low confidence, extremely poor result
31 - 55	very	Moderately low confidence, very poor result
56 - 80	plus	Medium confidence, poor result
81 - 90	somewhat	Moderate confidence, somewhat successful result
91 - 100	more_or_less	High confidence, very successful

Upon calculation of the modifier to be applied, the evaluator iterates through the setpiece predictor’s rule-base, discovering those antecedents that evaluated to less than 1 based on the world state at the time (less than 1 indicates a fuzzy decision based on this antecedent has been applied). This constraint is implemented in order that the modification takes place only on those antecedents that influenced the original decision. The selected modifier can then be applied to the specific antecedents that had an impact on the setpiece confidence in that world state.

### 3.5 Cooperative Adaptation

Cooperation is implemented within the system on a setpiece by setpiece approach. This allows for greater flexibility in construction of coordination abilities and in cooperative actions. Each agent has a model of the current team world state as part of the fuzzy logic world variables that can be drawn on for decision making. For instance, within each agent exists the ‘pass to player ahead’ setpiece which cooperates with the ‘receive pass’ and ‘intercept ball’ setpieces.

In general within the Robocup soccer system, as in the majority of cooperative interactions, an interaction between agents consists of two agents performing mutually beneficial operations. Adaptation of this interaction then, requires each agent to communicate their fuzzified degree of success or failure of the interaction to the corresponding agent in order for the appropriate modification to take place. When this is considered within the system described in the previous section, it can be seen that the results of each agent’s setpiece evaluator, after the interaction has taken place, can be applied to the corresponding other agent’s setpiece. This allows for the impression of each agents interpretation of success upon the corresponding agent. Which provides an effective cooperative adaptation model, that allows multiple agents to

interact and undergo modification based on the success or failure of their mutual interaction.

The limitation imposed by the soccer server that communication must take place between agents using the say and hear protocols does not impede this cooperative adaptation model. As each modifier is merely a linguistic expression (such as *extremely* or *more\_or\_less*) it can be transmitted directly, including an interpretation for the means of locating the relevant rule and antecedent. The fuzzy logic adaptation system is ideally suited to this environment due to a minimal amount of information being required for transmission.

## 4 Results

In order to effectively analyse the fuzzy logic adaptation systems ability to improve its performance by adapting its underlying fuzzy membership functions, a set of agents have been implemented, which places a high priority on a particular type of play, with the intention of allowing specific analysis of that play. A general system consisting of multiple types of possible plays would require a great amount of time for analysis as the system would need to be iterated for extensive periods in order to obtain sufficient quantities of data for analysis regarding the individual plays. Thus by restricting the possible actions undertaken by the agents in this implementation, the concept of fuzzy logic adaptation can be easily analysed, without extensive data collection.

In order to facilitate this requirement, the implementation of the System can be described as follows. Two offensive agents are created each with specialised skill sets relating to the abilities of a forward left wing / centre midfielder attacker (henceforth referred to as left wing attacker) and a centre forward player. The left wing attacker consists of the following special setpiece descriptions (in addition to the extensive general skills setpieces):

- Dribbles the ball toward the left side of the goal (if possessed)
- Upon reaching the goal turn to look for a player to pass the ball to
- Kick the ball to the other player.

The centre forward player consists of the following setpiece descriptions:

- Dribble the ball toward the centre of the goal (if possessed)
- If the other player has the ball and the centre forward agent is not close to the goal, or the centre forward player cannot see the ball, move toward the right side of the goal.
- Intercept the ball if it the ball is moving toward the agent and is within a reachable angle
- If the ball is kick-able, the centre player looks for the goal
- If the goal is visible and near and the ball is kick-able the centre player kicks the ball toward the goal.

An opposing agent is created with simple defensive capabilities, designed to imitate the ‘last line of defence’ style of defence. The opposing agent consists of the following setpiece descriptions:

- If the opposing agent can not see the ball or the goal or the ball is far away move to near the right hand side of the goal and near to the right hand player.
- If the ball is next to the agent, any attempt possible is made to kick the ball as far as possible away from the goal.

This opposing agent provides the simple penalty condition for the offensive system.

#### 4.1 Statistical Test 1 - Predictive Ability

As indicated in paragraph 3.4.3 on Setpiece Adaptation, a setpiece is essentially a form of adapting predictor. In this test the ability of the system to improve as a predictor is analysed. As discussed in the section on Setpiece Adaptation (Section 3.4), the system operates by fuzzifying a certain world state, producing a list of plays that are relevant for this state and then selecting the one with the most confidence for success. Thus in order to be executed, a play must have a strong confidence of success.

As the setpiece is modified or learns this confidence level of success is altered for different world states, thus allowing other setpieces which predict higher success to execute instead. For this analysis, however, the other setpieces merely cause a distraction and lengthen the required time to iterate the System. In light of this, the System is instructed to fire the particular play under analysis whenever it has a chance of success. This allows for a much more rapid cycling process and easier evaluation of results without affecting the system in any way. However this causes a problem when the confidence of a setpiece is analysed in order to obtain its prediction strength, as in the ordinary system (one without this simplification) the setpiece would not have fired at all - a setpiece with higher prediction would have fired in its place.

To simulate this setpiece competition and provide a means of choosing between successful and unsuccessful plays, a threshold value has been chosen. This threshold test suggests that a setpiece is said to have successfully predicted a play with a defuzzified confidence level when it provides a confidence level of greater than a threshold of 0.2 (defuzzifications of setpiece predictions range between 0 and 1, with 1 being a very strong prediction of success). This simulation threshold is based on observation of general setpiece interaction, in that, on average, a setpiece has to predict higher than 0.2 to 0.3 fuzzified chance of success before successfully being executed.

The statistic of the ratio of successfully predicted plays vs. unsuccessfully predicted plays provides a means of identifying any changes in the ability of the predictor. This first test compares the ratios over time obtained from an implementation of System with no adaptation implemented against that of an adaptation enabled implementation.

**H0:** The mean of the ratio of successfully predicted plays vs unsuccessfully predicted plays over time for a particular adaptation enabled implementation of System is no different to that of an adaptation disabled implementation of System.

Table 2 displays the ratio of successful vs unsuccessful predictions over time, obtained from multiple iterations of the System.

Figure 2 provides a scatter plot of the ratios of the data in Table 2.

Table 2: Total, successful and unsuccessful predictions over time.

Cycle	Total Predictions	Successful Predictions	Unsuccessful Predictions
500	75	35	40
1000	41	23	18
1500	32	22	10
2000	19	11	8
2500	30	19	11
3000	8	7	1
3500	21	15	6
4000	7	6	1

Figure 2: Scatter-plot of the ratio between successful and unsuccessful predictions over time.

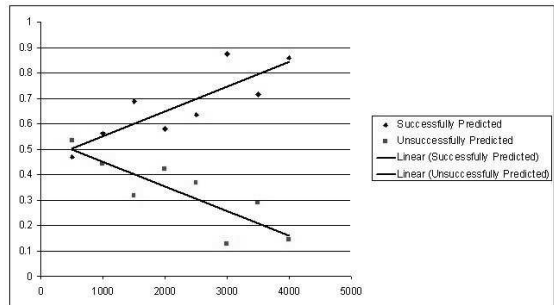


Table 3 displays the ratio of successful vs. unsuccessful predictions over time, obtained from 20 iterations of the System with the adaptation disabled implementation.

Table 3: Total, successful and unsuccessful predictions over time, adaptation disabled.

Cycle	Total Predictions	Successful Predictions	Unsuccessful Predictions
500	34	14	20
1000	12	8	4
1500	14	7	7
2000	13	8	5
2500	14	5	9
3000	19	12	7
3500	18	9	9
4000	11	5	6

Figure 3 provides a scatter plot of the ratios of the data in table 2

Figure 3: Scatter-plot of the ratio between successful and unsuccessful predictions over time.

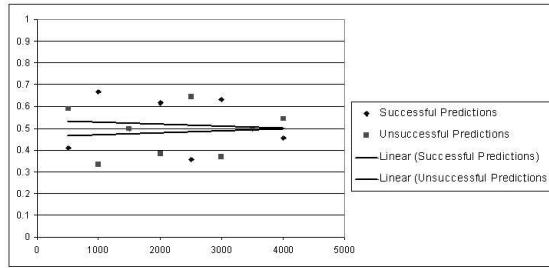


Figure 4 provides the results of a one sample t-test on the data provided in Table 2. The one sample t-test is designed to identify the difference between the mean of a sample and a value (Rudolf, Freund & Wilson 1997). In this case the value is set to 0.526 which is the mean of the adaptation disabled data. This t-test then will produce the t-value and the significance of the difference in mean of the adaptation enabled data and the mean of the adaptation disabled data.

Figure 4: One sample t-test on the adaptation enabled data.

One-Sample Test						
Test Value = .526						
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
learning	2.892	7	.023	.1457	.0266	.2649

**Conclusion:** In order to refute the null hypotheses two conditions must be observed:

1. There must be an indication of positive difference in mean; and
2. This indication must be significant.

From the t-test presented in Figure 4 it can readily be seen that the t value is positive, indicating that the difference between the mean of the adaptation enabled data and the test value is a positive one. The significance of this positive difference is below 0.05, achieving the general statistical standard for significance<sup>1</sup>. This analysis provides sufficient evidence to reject the null hypotheses and suggest that there is a difference in the mean of the ratio of successfully predicted plays vs. unsuccessfully predicted plays over time for a particular adaptation enabled implementation of System against an adaptation disabled implementation of System.

#### 4.2 Statistical Test 2 - Response of Modification:

The adaptation ability of the system should be such that a local maximum is reached with very few test instances. This is due to the nature of the application domain: there may be very few actual executions of a setpiece in a given game. In this test local maxima are defined as the point at which there exists the most positive mean difference compared to those

<sup>1</sup>The general statistical standard for significance is 95% confidence for most applications.

Table 4: Differences in means

Cycles	Successful Predictions	Mean After	Mean Before	Difference
0	.526			
500	.467	.6717	.526	.1457
1000	.561	.7010	.4963	.2047
1500	.688	.7244	.5179	.2045
2000	.579	.7317	.5603	.1715
2500	.633	.7699	.564	.2059
3000	.875	.8154	.5756	.2399
3500	.714	.7856	.6184	.1673
4000	.857	.8571	.6303	.2268

points around it. While this test restricts itself to identifying the earliest local maxima, there may exist higher maxima at later stages of the game itself. This analysis, then, attempts to discover the number of cycles before a local maxima is reached. This figure can then be used to count the average number of executions during the number of cycles. A local maximum is a good indication of achieving performance better than that of the control system (adaptation disabled implementation) within the shortest possible time.

The number of cycles required before a local maxima is reached can be ascertained by calculating for every cycle, the difference between the mean of the successful predictions before the cycle, and the mean of the predictions after the cycle. Those cycle points with a difference greater than the cycle points surrounding them indicate that a local maxima has been achieved. This is summarised in table 4 for the data obtained from the optimal implementation of the System discussed in Tests 1 and 2. The average mean from the adaptation disabled implementation is included at cycle 0 to provide a starting point for the analysis.

It can readily be seen from Table 4 that the first local maxima lies between the cycle points 1000 and 1500, it seems safe to assume that this can be approximated to 1250 cycles. A second local maxima can be identified at the cycle point 3000, this appears to be the true maxima for the given set of data.

From the raw data, provided by the multiple iterations of this implementation of the System, the average number of plays in a 1250 cycle block is 4. From the same data the average number of plays over a 3000 cycle block is 11.

**Conclusion:** These results indicate that the System is achieving a local maxima within 4 executions of that setpiece and the corresponding positive or negative modifications.

#### 4.3 Statistical Test 3 - Comparison with other Predictors:

As discussed in the previous test the fuzzy logic adaptation system described in section 3.4 is essentially functioning as a predictor, with a specific implementation for each type of action to be predicted. In this test the predictive ability of the fuzzy logic evaluator is compared to the predictive ability of other types of predictors on the Robocup problem domain. This comparison aims to provide evidence that the fuzzy logic predictor is able to predict, based on the raw data of the system, with a high degree of success. In order to provide the comparison, the raw data was taken from the iterations of the adaptation disabled implementation of the System and con-

Table 5: Results of other predictors

Naive Bayes	IBk	C4.5
59.96	62.5	65

verted to the weka arff format. Weka is a collection of machine adaptation algorithms for solving real-world data mining problems and provides an efficient means of evaluating a dataset (Witten 2000). The following machine adaptation algorithms were selected for comparison:

**Naive Bayes:** The Naive-Bayes predictor computes conditional probabilities of the classes given the instance and picks the class with the highest posterior. Attributes are assumed to be independent, an assumption that is unlikely to be true, but the algorithm is nonetheless very robust to violations of this assumption.

**IBk:** IBk is an implementation of the k-nearest neighbours classifier that employs a distance metric.

**C4.5 decisions trees:** C4.5 uses the concept of information gain to make a tree of classificatory decisions with respect to a previously chosen target classification. The information gain can be described as the effective decrease in entropy (usually measured in terms of ‘bits’) resulting from making a choice as to which attribute to use and at what level.

Table 5 provides the average of the successful predictions after 300 iterations of each algorithm on the data used by the multiple iterations of the adaptation disabled implementation of the System (this includes several calculated data variables). These algorithms were provided with a 66 / 33 split with 66% allocated for training and 33% allocated for evaluation.

The results obtained from Test 3 (4.2) suggest that the system reaches an initial local maxima after 1250 cycles. The mean of the successive positive predictions from this point forward is approximately 71%. If the second local maxima at 3000 cycles is considered, the mean of the successive positive predictions is approximately 81%.

**Conclusion:** A machine adaptation algorithm that has knowledge of a domain should perform better than an algorithm which has no knowledge beyond the raw data. This is true for the case presented in Test 5 and due to this fundamental difference in approach there can be no direct comparisons drawn between the performance of other algorithms and the fuzzy logic classifier however, the difference between the prediction results of the machine adaptation algorithms and that of the fuzzy classifier indicates a positive result.

This difference in performance can be attributed to two factors:

1. The inherent fuzziness of the system causes better prediction and handling of vague and imprecise data; and
2. The operators knowledge of the domain causes better prediction.

#### 4.4 Statistical Test 4 - Analysis of Prediction Confidence

In the fuzzy logic system described in section 3.4, the return value of the setpiece indicator is a confidence level of to what degree a particular setpiece believes it will be successful if fired in the current world state. This test aims to analyse the prediction ability of the predictor as it learns. False positives are those predictions in which the predictor suggested that it would be successful and was in fact incorrect.

**H0:** There is no decrease in prediction confidence for false positives in the adaptation enabled implementation compared to the adaptation disabled implementation

Figures 6 and 5 display the data obtained from averaging the prediction confidence for each time segment for the false positives, in the optimal iteration of the adaptation enabled implementation and the adaptation disabled implementation.

Figure 5: Adaptation disabled prediction confidence

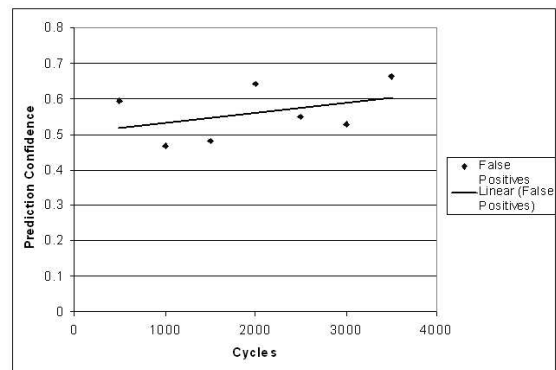
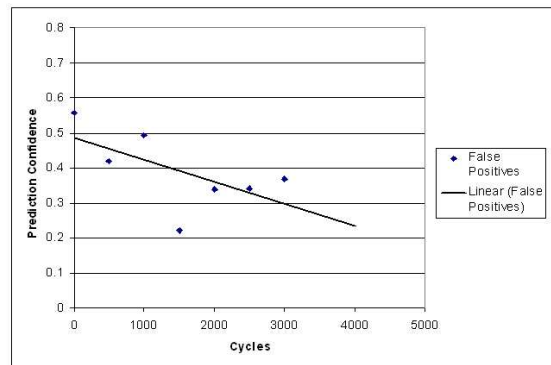


Figure 6: Adaptation enabled prediction confidence



The following Figure (7) presents the results of a one sample t-test performed on the adaptation enabled data presented in a scatter-plot in Figure 6. This one sample t-test tests for a difference between the mean of the adaptation disabled data and that of the adaptation enabled data.

Figure 7: One way t-test on the adaptation enabled prediction confidence

One-Sample Test						
Test Value = .53						
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Time	-3.312	6	.016	-.1387	-.2412	-.0362

**Conclusion:** From the t-test presented in Figure 7 it is readily apparent that there is a significant difference between the mean of the adaptation enabled data series and that of the adaptation disabled data series. Moreover it is obvious from visual analysis of the data that this difference is negative, indicating that the prediction confidence on false positives is decreasing for the adaptation enabled data series. This provides sufficient evidence to reject the null hypotheses.

The nature of the system described in section 3.4 indicates that if a setpiece’s false positives decrease, another setpiece is more likely to be chosen in its place.

#### 4.5 Statistical Test 5 - Adaptation around opposition’s actions with cooperation

In Section 3.4 a description of the cooperative adaptation ability of the agents is discussed. While the previous results included the impact of this cooperative adaptation, this test proposes to attempt to isolate the effect that the cooperative adaptation is having on the individual agent’s interaction with the environment.

The setpiece ‘move to cross receiving position’ requires an agent to locate the other player’s goal, and move into an optimal cross receiving position. This optimal position involves, among other things, being an optimal distance and direction from the opposing players in order to prevent them from interrupting the play and an optimal distance and direction from the goal. As the receiving agent and the providing agent interact and adapt, the averages of the distances and directions of the receiving agent to the objects in the field should change to represent this adaptation.

In order to effectively analyse and isolate the effect of the cooperative adaptation, the variable of ‘distance from the centre attacker and the defensive player’ is selected for analysis. If this value changes over time compared to the adaptation disabled implementation of the System then it can be suggested that the System has successfully discovered the optimal distance to maintain from the opposing players.

**H0:** The distance between the centre attacker and the defensive player does not change over time.

The following figures display the plot of the raw data for the variable ‘distance between the centre attacker and the defensive player’ over time for both the adaptation disabled implementation of the System and the adaptation enabled implementation of the System.

Figure 8: Adaptation disabled distance between the centre attacker and the defensive player over time.

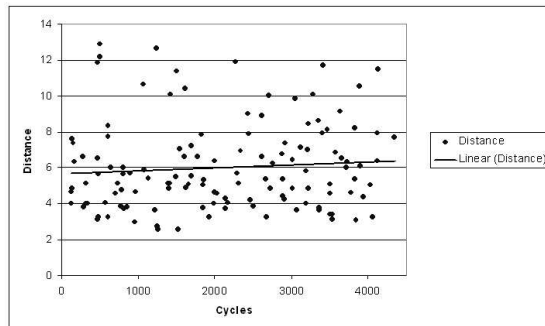
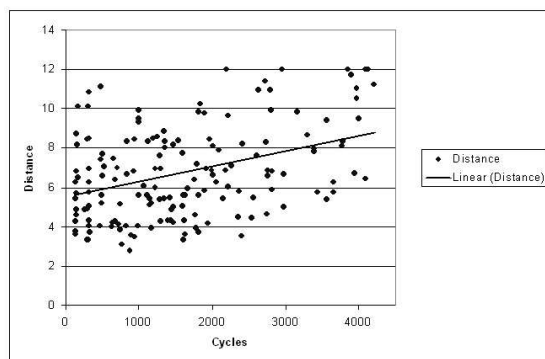


Figure 9: Adaptation enabled distance between the centre attacker and the defensive player over time.



**Conclusion:** From visual observation of the linear trends on the plots in Figures 8 and 9 it can readily be seen that the distance between the centre attacker and the defensive player is increasing in the adaptation plot, and remaining approximately linear in the no adaptation plot. Observation of a trend is sufficient evidence for this test. Due to the randomness apparent in this variable, a t-test (which relies on variance) would not be able to provide a sufficient degree of confidence. This observable upward trend, however, provides sufficient evidence to reject the null hypothesis.

The results obtained from this test indicate that the system cooperatively adapts to the opposing agent’s actions. While the difference between the no adaptation case and the adaptation case is not overly large, this indicates that the original setpiece rule-base definition for this variable was reasonably good, only requiring moderate amounts of modification before attaining an optimal definition.

## 5 Conclusion and Further Work

This paper has presented an investigation into the construction and performance of a cooperative adaptive fuzzy logic system for Robocup, in order to determine the viability of such an approach in a complex, strict, real-time and imprecise domain. Firstly, the development of a fuzzy logic system provided the basis for the construction of the cooperative adaptive agents. Secondly the development of the agents was undertaken. Thirdly, the resulting system was statistically analysed. This system produced positive results which are summarised as follows.

The tests on the System concluded the following:

- The system increases its ability to predict a successful play;
- The modifier output combination ‘adaptation’ provides an optimal medium between the two extremes of stability and adaptability;
- The system is achieving a local maxima in prediction ability within 4 executions;
- The system provides better prediction ability with 4 training points and even better prediction ability after 11 training points than 3 classifier machine learning algorithms each with 88 training points;
- The system reduces error as the prediction confidence for false positives decreases; and
- The system is effectively cooperatively adapting around the oppositions playing style.

In summary, Fuzzy Logic can be used for cooperative adaptive real time agent control. The adaptation and cooperative adaptive methods implemented operate successfully. These conclusive results provide evidence that the system is operating successfully and justifies further work to develop and refine the proposed approach to cooperative adaptive real-time fuzzy logic adaptation systems.

## 5.1 Further Work

Further development of the fuzzy adaptive controller would take the following form:

### 5.1.1 Hierarchical Setpiece Implementation

The current approach for controlling state of the fuzzy adaptive agent is to provide setpieces capable of competing across all states. This is considered somewhat restrictive due to the complexity of the construction of the predictors for each setpiece. In order to rectify this it is suggested that a System created where  $n$ -levels of setpiece prediction exist. This would allow for a much finer granularity of control over the current actions of an agent even in changing world states.

### 5.1.2 Individual Modifier Function for each Setpiece

The fuzzy logic adaptive agent uses a single generic fuzzy logic modifier function (see 3.4.3). A possible improvement on this approach could involve specifying for each setpiece an optimal modifier function mapping, this could take the form of a further fuzzy rule base, which itself could be optimised.

### 5.1.3 Use of better modifiers

The fuzzy logic adaptive agent uses a simple set of linguistic modifiers to achieve adaptation on the underlying fuzzy sets. Recent advances on linguistic modifiers (Marin-Blazquez & Shen 2001) are defining new types of modifiers that better represent linguistic terms and better modify the underlying fuzzy sets, such applications include contrast modifiers that increase and decrease the fuzziness of a system as well as ‘second generation’ fuzzy modifiers.

### 5.1.4 Acquiring New Rules

The implemented system relies on the underlying rule-base to be both partially correct and descriptive, it is proposed that a System created that construct additional rules both within setpiece prediction and within setpiece actions. This could take the form of a fuzzy rule creator rule base that suggests appropriate new rules based on the evaluations of the actions undertaken on the data.

## 6 References

### References

- Chen, M. (2002), *Robocup Soccer Server Users Manual*.
- Kruse, R. & Nrnberger, A. (1998), ‘Learning methods for fuzzy systems’, *Non-Linear Electromagnetic Systems: Advanced Techniques and Mathematical Methods* .
- Langholz, G. (1992), *Hybrid Architectures for Intelligent Systems*, CRC Press.
- Marin-Blazquez, J. G. & Shen, Q. (2001), Linguistic hedges on trapezoidal fuzzy sets: A revisit, IEEE International Conference on Fuzzy Systems.
- Nauck, D., Flawonn, F. & Kruse, R. (1992), ‘Fuzzy sets, fuzzy controllers, and neural networks’.
- Nauck, D. & Kruse, R. (2002), ‘What are neuro-fuzzy classifiers?’.
- Pearson (2001), *Fuzzy Logic Fundamentals*.
- Rudolf, J., Freund, W. & Wilson, J. (1997), *Statistical methods*, San Diego : Academic Press.
- Witten, F. E. (2000), *Data Mining, Practical machine learning tools and techniques with java Implementations*, Morgan Kaufmann.