DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGNING ADAPTIVE OPPONENT MODELS FOR RTS GAMES

by

İsmail Aybars MORALI

October, 2010

İZMİR

DESIGNING ADAPTIVE OPPONENT MODELS FOR RTS GAMES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master Of Science
in Computer Engineering

by

İsmail Aybars MORALI

October, 2010
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGNING ADAPTIVE OPPONENT MODELS FOR RTS GAMES"** completed by **İSMAİL AYBARS MORALI** under supervision of **YRD. DOÇ. DR. H. ŞEN ÇAKIR** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

YRD. DOÇ. DR. H. ŞEN ÇAKIR

Supervisor

(Jury Member)

(Jury Member)

Prof.Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I have to thank my previous supervisors, Yrd. Doç. Dr. Emine Ekin and Öğr. Gör. Dr. Özlem Öztürk. Without their inspiration and mentorship, probably I would not able to progress so much and even never heard about the environments I studied for thesis. My current supervisor, Yrd. Doç. Dr. H. Şen Çakır, it was a pleasure to study with an quite experienced advisor, even for a short period. Besides academical personnel, most of support was from my family, my girlfriend and my friends; appreciation for their understanding.

One last word for open source community, they really deserve gratitude. Their unique mentality produced the tools I have used for thesis like RoboCup Soccer Server, Dainamite and Wright Eagle teams, Eclipse, RapidMiner, Dia etc.

İ. Aybars MORALI

DESIGNING ADAPTIVE OPPONENT MODELS FOR RTS GAMES

ABSTRACT

Games, reality of nature, have a very important part in our life. Games are fed by challenges and real challenges can only be created with help of AI. This thesis deals with corner kicking in soccer game. In thesis, it is studied in RoboCup 2D simulation league with Dainamite team against Wright Eagle team and focused on corner kicking from a different aspect than other all RoboCup teams. While corner kick situations are all the time treated by an ordinary short pass or kick in situation by other teams, in this thesis it is treated as corner kick that allows carom positions or running players which requires a long kick inside the penalty area. In corner kick, it is only decided where to kick the ball, positioning of players in the field is excluded. For the classification method, Support Vector Machine's LibSVM implementation is used and experiments are performed on parameters of SVM and MySVM implementation.

Keywords: Games, challenge, AI, corner kick, RoboCup, Soccer Server, SVM, classification

GERÇEK ZAMANLI STRATEJİ OYUNLARI İÇİN UYARLAMALI RAKİP MODELİ TASARIMI

ÖZ

Oyunlar, doğanın gerçeği, hayatımızda çok önemli bir yer tutar. Oyunlar zorluklar ile beslenir ve gerçekçi zorluk hissi sadece YZ yardımı ile yaratılabilir. Bu tez futbol oyununda köşe vuruşu kullanmayı ele alır. Tezde, RoboCup 2B benzetim liginde Wright Eagle takımına karşı Dainamite takımı ile çalışılmıştır ve köşe vuruşuna diğer tüm RoboCup takımlarından daha farklı yaklaşılmıştır. Köşe vuruşu diğer takımlar tarafından her zaman sıradan bir kısa pasmışçasına ya da taç atışı kullanırmışçasına kullanılırken,bu tezde karambol pozisyonlar veya koşan oyuncular gibi ceza sahasına uzun pas gerektirecek olaylara izin verecek şekilde köşe vuruşu olarak ele alınmıştır. Köşe vuruşunda topun sadece nereye doğru vurulacağına karar verilmiştir, oyuncuların saha içindeki yerleşimi konu dışında bırakılmıştır. Sınıflandırma yöntemi olarak Support Vector Machine'in LibSVM gerçekleştirimi kullanılmış ve SVM'nin parametreleri ve MySVM gerçekleştirimi deneyler yapılmıştır.

Anahtar sözcükler: Oyunlar, zorluk, YZ, köşe vuruşu, RoboCup, Soccer Server, SVM, sınıflandırma

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

Games are reality of nature. Human being, especially kids, intuitively play games. According to a research, Riley (2009) claims that 82% of children in USA between 2-17 describe themselves gamers. Playing games is encouraged by aware parents because it is a powerful and entertaining way of improving several skills like language, physical, thinking, creative and social skills. It is also essential for psychological development of a person. Besides human being, intelligent animals also need games in their life and they play various games. They are even observed during cheating in games, as stated in Ross's (2010) research. Games, seemingly such a simple and natural term, evolved in parallel to human evolution and in information age, a new branch is formed called video games. After high processing power in computers and great visualizing combined with video games, new generation games become very different from the games played before computers. Result is fascinating, video game industry, interactive entertainment as the fancy name, now has a market size of several billion dollars.

1. 1 Adaptive Opponent Model in RTS

Video games are being improved and getting more complicated. Due to the improvement, expectations and satisfaction levels are greatly raised. Old games were narrow scoped and user interaction was restricted. Current generation games have gigantic content and users have much more freedom in their choices during game plays. Thus each play of the game is nearly unique and the solution space of the game is enormous. For this reason, opponent models should be specifically handled, differently from old games. Moreover, game objects are increased in count as games got complicated. For coordination of those objects, artificial intelligence is needed. In old games, there were few objects and game play was straight; not free like today. And adjusting the limited objects would adjust most parameters and difficulty of the game. But now games contain much more independent objects and game play is not only based on a few templates. Such a complexity cannot be handled with an if-then

command sequence. Another aspect, providing reality in simulation, sports and strategy games is a very difficult task. Although companies develop their artificial intelligence algorithms for their sport games in each version every year, players are known to complain about the silliness of the intelligence of the game. So, artificial intelligence is not sufficient enough and it is one of the most important subjects in games. Because artificial intelligence provides the most crucial part in a realistic way for gamer, “challenge”.

1.1.1 Game Types

Game types are conventionally described as sports, simulation, real time strategy, action, adventure etc. But with the new generation games, it has become getting impossible to classify games for last 10 years. No longer have games belonged to only one type. A game contains several types in it. Dungeon Keeper, at first sight can be classified as real time strategy, has fighting creatures and any creature can be chosen and possessed, letting gamers play from creatures eye. Game type changes from real time strategy to first person shooter. GTA series, may be called action game, contains game parts of types real time strategy, flight simulator, car racing, first person shooter, puzzle, adventure and many others. The game Spore consists of 5 stages, starting as a cell and after evolution finishing in space age. It is played as platform game in cellular stage, role playing game and action game in creature stage, role playing game and real time strategy game in tribal, civilization and space stages. Pro Evolution Soccer 2010 can be played as soccer, managership or role playing types. As a result it is not possible to describe games type generally. Instead, games related contents type should be used.

1.1.2 Corner Kick

Corner kick resides in a special place in soccer. It is a great chance for scoring a goal. Corner kicks has a unique property in soccer games in which it is possibly repetitive. After having a shot a second one won't be available easily. Especially in free kick, after a free kick a new one is almost impossible. But it is quite possible to

have three successive corner kicks. Another important point, corner kick set play gathers almost all players to opponent penalty area. During positioning and on the way back, a remarkable duration exists and makes corner kick owner team relaxed. Corner kick is a great pressure element for goal chance and gathering players to opponent field. Only disadvantage is, it is vulnerable to counter attacks.

Corner kicks occurred frequently (mean = 10.85) in the English Premier League and provided many goal-scoring opportunities (1 in 3) although these opportunities were only converted 1 in 11 times. These findings were similar to Hill and Hughes (2001) and Olsen and Larsen (1997). Interestingly, the 1:11 ratio is the same as Partridge and Franks (1989a, 1989b) found for crosses, suggesting communality between the two aspects of play. (Taylor, James and Mellalieu, 2005, p. 228).

As mentioned by Taylor and friends, corner kick plays an important role in all soccer matches without any exceptions. So improving this point is crucial for overall performance.

1.2 Main Objective

This thesis focuses on one of the strategy parts of the soccer game, corner kick. For this task a soccer game simulator used and “where to kick the ball in a corner kick” question is answered, i.e. decided, with machine learning techniques.

1.3 Related Work

The lack of multi-agent soccer game environments for developing prevents working on specialized subjects like corner kicking. There is not a very similar work to this thesis. The closest works are RoboCup simulation leagues, but teams there are not completely ready to deal with this kind of detailed tasks and corner kick is always ignored. Corner kick is considered as a standard kick in or a standard passing to teammates shortly. None of the teams care about corner kick as mentioned in Chapter 1.1.2.

Shi (et. al., 2009) stated that they have used Markov Decision Process (MDP) in their kicking ability, for their team Wright Eagle. Another team, HelliBASH, Zanjani (et. al., 2009) explained their progress for decision making when ball is possessed but there is not a special action for corner kicking.

1.4 Guide to Thesis

Second chapter explains RoboCup and related terms with it. Also machine learning method is explained. Third chapter describes the problem, problems solution, experiments and finally experiment's results. Chapter four contains conclusion and the last words.

CHAPTER TWO
SOFTWARE ENVIRONMENT

There is not unfortunately much soccer game simulator can be worked on. From available softwares, RoboCup community seems to be the ideal solution.

2. 1 RoboCup

RoboCup is defined as “an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined” (Kitano et al., 1997, p73). RoboCup is alive since 1995 and every year official tournaments and side-events are organized. Officially and bravely aim is described as “By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of the FIFA, against the winner of the most recent world cup for human players” (Kitano et al., 1997, p73). In RoboCup researches include a lot of topics instead of one, that's how basically distinguished with traditional researches.

Softwares used for RoboCup are open source and mostly participants internalize open source paradigm, so this field has remarkable accumulation of knowledge. This mentality allows focusing on only desired problem and creates a positive feedback loop by providing basis to researches, like this thesis.

2.1.1 Leagues

RoboCup consists of four branches. RoboRescue robots search and rescue in real tough environments, @Home constructs robots that interact with human and help them in daily life. RoboCup Junior encourages kids to science with entertaining robots and dances. The last branch is the main branch, soccer. Soccer branch consists of small sized, medium sized, humanoid, standard platform and simulation leagues. Simulation leagues are created in form of standard platform leagues. They are specially created in order to abstract from the hardware layer and allowing the

developer to work on algorithms more. Simulation league has two varieties, two dimensional and three dimensional. This thesis is implemented on 2D simulation environment but with necessary modification 3D simulation environment is also possible to run the softwares on.

2.1.1.1 2D Simulation League

Figure 2.1 The soccer simulator. Dainamite vs WrightEagle.

In 2D simulation league there are two teams consist of eleven players and an automatic referee exists. Simulation server and the viewer module "server monitor" are independent softwares, there is a special message format for communication with server. Players all are independent, they sense themselves and they decide themselves. They are separate processes and the only communication method is the very restricted in-game talking commands through server. In Figure 2.1, RoboCup Soccer Server Monitor can be seen; two teams are connected and ready to start match.

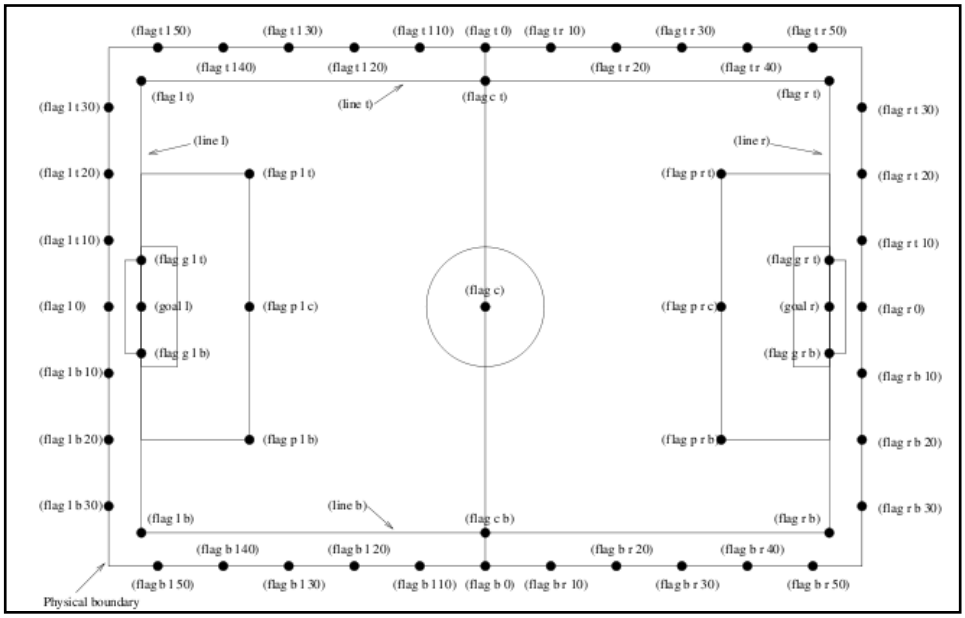

Figure 2.2 Flags on the field. (Chen et al. (2002))

Players understand their positions with the flags seen in Figure 2.2. They just see flags, opponents and ball. They have to process these information with information gathered before and calculate their position. Game rules are realistic, even the offside rule is implemented. Another point for realism, most values is affected by artificial noise.

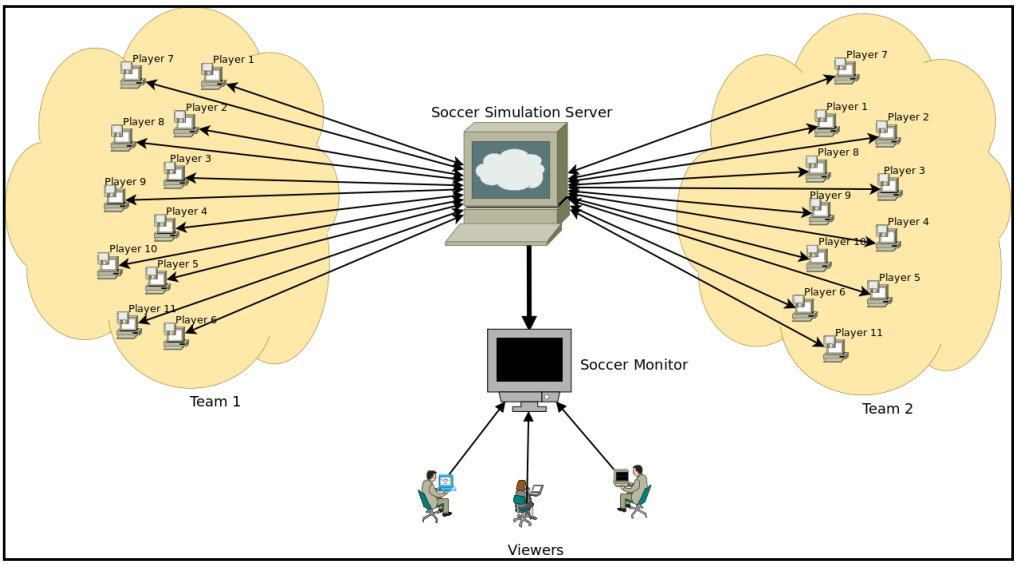

Figure 2.3 Overview of soccer simulation server..

Figure 2.3 states the overview of the simulation server. Two teams connect to the server and each player is a separate process. All processes run the same code which acts differently according to the player numbers or roles. Interprocess communication is forbidden, so all players have to communicate through server with special commands, “hear” and “say”. Players can say a very short string and close players hear the said string. Tactics, positioning and all organizations are communicated in this way. In Figure 2.3 modular design can be seen explicitly. As a result of the modular design, arbitrary number of monitors can connect to the simulation server. Or a monitor can connect to a server on the network, which is the official way in RoboCup tournaments to use the computing power efficiently.

2.1.2 Teams

Reusing a team instead of writing from beginning has great advantages. A lot of basic and time consuming developments like communication with server, centralizing the players’ eye views, passing are abstracted. In this thesis, it is studied with Dainamite and Wright Eagle teams which are both open source and suitable with the latest 2D simulation server. Wright Eagle is one of the best teams and between 2005 and 2010 has two first place and four second place rankings. Dainamite is much weaker then Wright Eagle and doesn't have remarkable success in the last years. For development Dainamite team that has a good coding system is used. For opponent Wright Eagle team is used. Improving weaker team against stronger team is more meaningful and valid for performance testing.

2.2 Limitations and constraints

Teams have some limitations that affect the result. Dainamite team used to start corner kicks with a short pass, not a specially handled situation. They can only pass to a teammate close to the corner. When all teammates are positioned away, it is stuck and player cannot kick ball. Besides this, instead of the perfect passing and movement of Wright Eagle, Dainamite was not so fluent in moves. Wright Eagle runs a lot and consumes players’ stamina in the last part of the match.

2D simulation is somewhat different from the real life corner kicks. In real corner kick ball can pass over players but in 2D simulation naturally third dimension, height, does not exist. This lack of ability prevents integrating the players near the far post to the game. Another difference is 2D simulation does not allow curved kicks; kicks can only be done on a straight line.

2.3 Classification Method

As the classification method, Support Vector Machines (SVM) is used. Proven power and wide usage area makes SVM attractive. SVM is often compared to Artificial Neural Network (ANN) which is definitely the most used classification algorithm. But SVM has a big advantage over ANN about local optima, while ANN suffers from it, SVM is not affected.

2.3.1 Support Vector Machine

Support Vector Machines, proposed by Vapnik et. al. (1995), is based on statistical learning theory. SVM is a supervised learning algorithm and used for classification and regression analysis. It is a very general technique, it can classify any kind of data. Since SVM proposed, it has shown very good performance and now SVM is a very powerful alternative to other classification algorithms.

2.3.1.1 Algorithm

SVM basically has two steps, mapping the data to feature space and separating data into two classes.

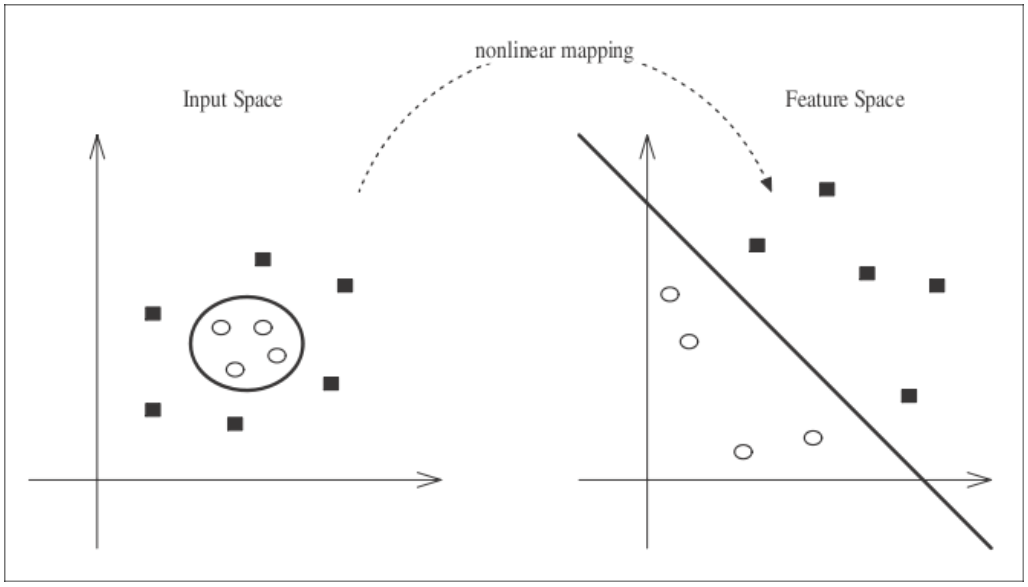

Figure 2.4 Mapping in SVM. (Busuttil (2003))

First SVM maps data to high dimensional feature space, an example can be seen in Figure 2.4. In the feature space, each coordinate represents a feature of data, points represent data itself. In this very complex space, relations between data are found. The mapping process is called “kernel trick”.

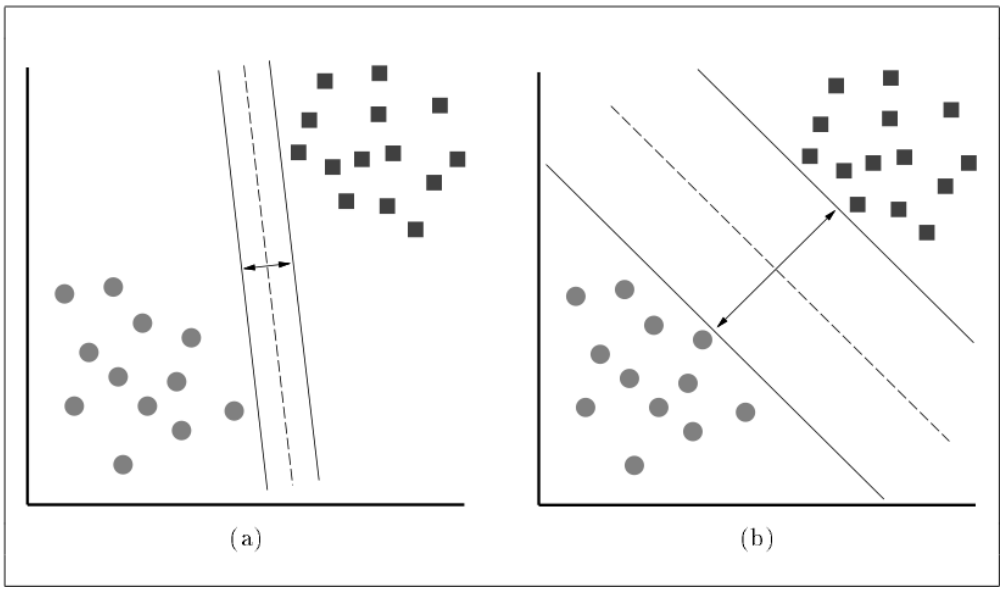

Figure 2.5 Separating planes in SVM. (Osuna et al. (1997))

After mapping data, SVM constructs hyperplanes in high dimensional space to separate into two classes. Infinite number of hyperplanes exists and optimal one

should be chosen. Optimal hyperplane, which is unique, is the best separating one and also called maximum margin hyperplane. Optimal hyperplane has the maximum margin between lines. In Figure 2.5, two different hyperplanes are shown. The hyperplane in (a) also divides the classes perfectly but is not the optimal one. In (b), hyperplane has the maximum margin of all possible hyperplanes, therefore it is the optimal solution. Actually, for this class separating issue, SVM is not a search algorithm, it is considered as an optimization algorithm.

"Support Vector Machines were originally designed for binary classification" (Hsu et al, 2002, p415). As mentioned in algorithm, feature space is divided into two classes. Only two classes are not sufficient for most of real world problems. Hence, various methods are developed to overcome this binary classification problem.

In order to classify more than two classes, binary classifier SVM is used several times internally. Major methods are one-versus-one and one-versus-all algorithms. In one-versus-one algorithm, all classes are run against each other, which is equal to binary combination of all classes. Totally, k(k–1)/2 times SVM is run, where k represents class count in the problem. The class which wins most is chosen as result. In second method, one-versus-all, for each class SVM is run against remaining classes as they are one class. Totally k times SVM is run for one-versus-all. The winner is chosen as the result class. After performing experiments on large problems, Hsu et al. (2002) states that one-versus-one method may be more suitable for practical use. Already, major implementations of SVM use one-versus-one method for multi-class classification.

2.3.1.2 Parameters

SVM has two very important parameters. One of them is kernel type, which is the feature mapping. Kernel types maps the data as feature in a space. Most common examples of kernel types are linear, polynomial, Gaussian (rbf) and spline kernels. Kernel type strongly depends on training data. For choosing the right kernel type, Howly et al. (2005) used a genetic algorithm.

Second parameter is C, the cost of misclassification. C controls the tradeoff between complexity and accuracy. When cost is increased, misclassification is punished and accuracy is forced to increase. But C should be set carefully because it causes over fitting. Forcing the classifier too much produces a very complex model and extremely complex models are not useful in real environment.

2.3.1.3 Implementations

SVM has different implementations. Most common used are MySVM and LibSVM. MySVM is an implementation based on an optimization algorithm described by Joachims (1999). It can work on classification, pattern recognition, regression and distribution problems. LibSVM is implementation of Chang et al. (2001). It is very comprehensive can be used on all asks. MySVM is simpler and faster than LibSVM. Both implementations use one-versus-one method for multi-class classification.

CHAPTER THREE
PROBLEM AND SOLUTION MODEL

3.1 Problem

In order to deal with a formal problem, the subject is defined as kicking corner kicks in RoboCup 2D simulation soccer game. Corner kicking is related with positioning but positioning is not covered in this thesis as it is another subject and needed to be studied separately. So problem is deciding where to kick after players positioned in a corner kick. In this chapter, problems solution under constraints mentioned before is explained.

3.2 Solution Model

Target in a corner kick may be a player, an area or a direction.

Choosing a player as target means ignoring most specialties of corner kick. It cannot fulfill that players can move, carom positions or directly shooting to goal, as a result overall performance dramatically drops. Choosing an area instead of a player makes more sense. But target area lacks in two points, size of the areas and exactly which point to kick in area. Answering these questions is very difficult. Moreover when corner kick is kicked ball may be intercepted by a teammate or an opponent. Choosing directions is the most native answer, suitable to the nature of the kick and the journey of the ball after kick. It overcomes disadvantages of its alternatives.

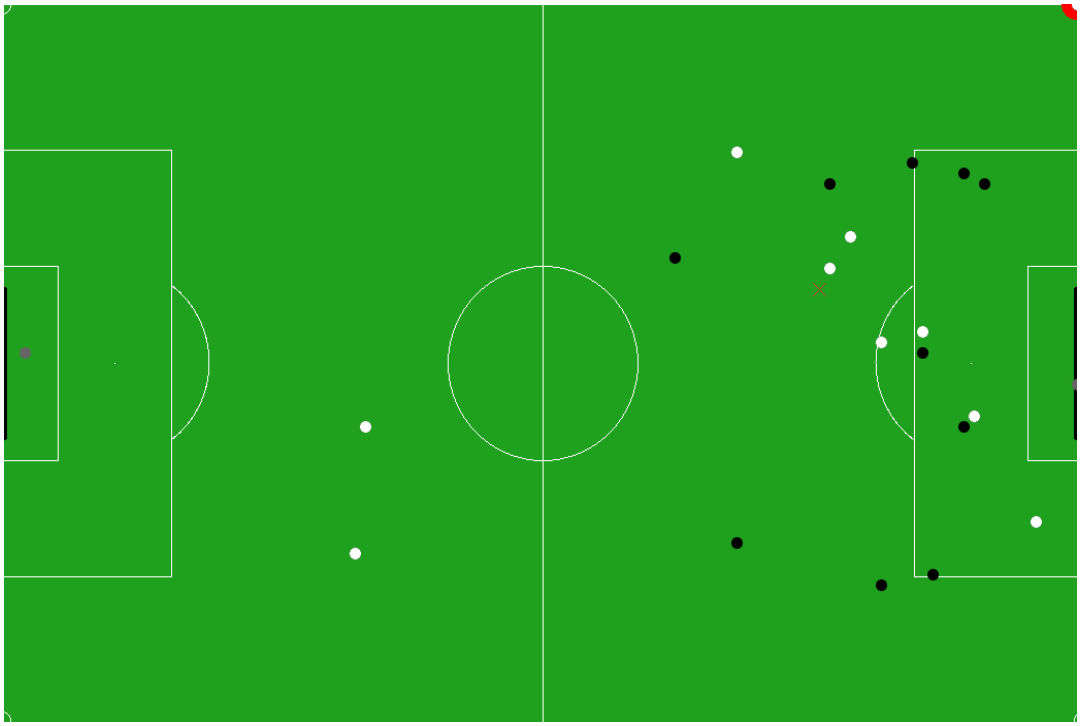

Figure 3.1 Scene 190, result is counter attack, ball is kicked to the red cross. One of the corner kicks during training.

As seen in Figure 3.1, in Scene 190, the two players in the center of the penalty area has a chance to run and possess the ball when ball is kicked close to the goal. Only way to provide is using directions for kicking.

Classification in solution should run with successful samples. In this thesis success of corner kick means one of the results of goal, kick in and timeout. It may be claimed that only goal should be considered as it changes the scoreboard but in a 7 minutes game having the ball 30 seconds in the opponent field or forcing the opponent to kick the ball out of field are remarkable successes. Having the ball in opponent field spends time while team is winning; also it always contains a shooting chance which has power to change the score.

Problems solution is quite simplified. There is a rectangle, there are 22 points in it and one of possible six directions from corner should be chosen. Problem is now ready to be applied any classification algorithms.

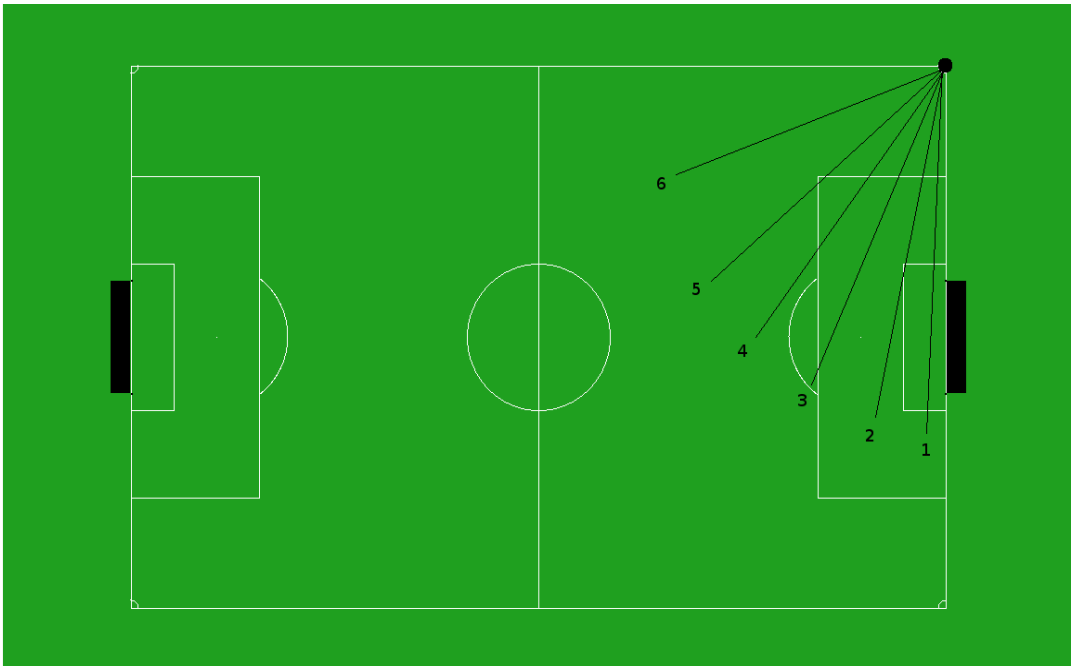

Figure 3.2 Directions to be kicked through.

Figure 3.2 shows the possible six directions. After classification, ball will be kicked through the classified direction.

3.3 Data Gathering

All classification methods need sufficiently big training set to operate and data should be gathered in natural ways. So, Dainamite's source code is modified and simulation server's trainer coach which is designed for such purposes is used. In normal games teams connect to server and they play the match. But with the help of the trainer coach, server rules are modified and game is set to play only consecutive corner kicks in desired conditions. From now on, the team uses corner kicks will be named as “our” team and the defender team will be named as “other” team for the sake of simplicity.

The mechanism started to run when teams connect is called “scenario”. A scenario starts when teams connected and ends when the match time is over. Unfortunately match duration can be adjusted but the time cannot be set, so a separate software is programmed. This software, BatchMatch, starts simulation server and then connects

the teams. When match and inherently scenario are over software repeats the scenario in desired counts. As a result, data set is gathered with running the software BatchMatch during days and having played thousands of corner kicks. Flowchart of BatchMatch is shown in Figure 3.3

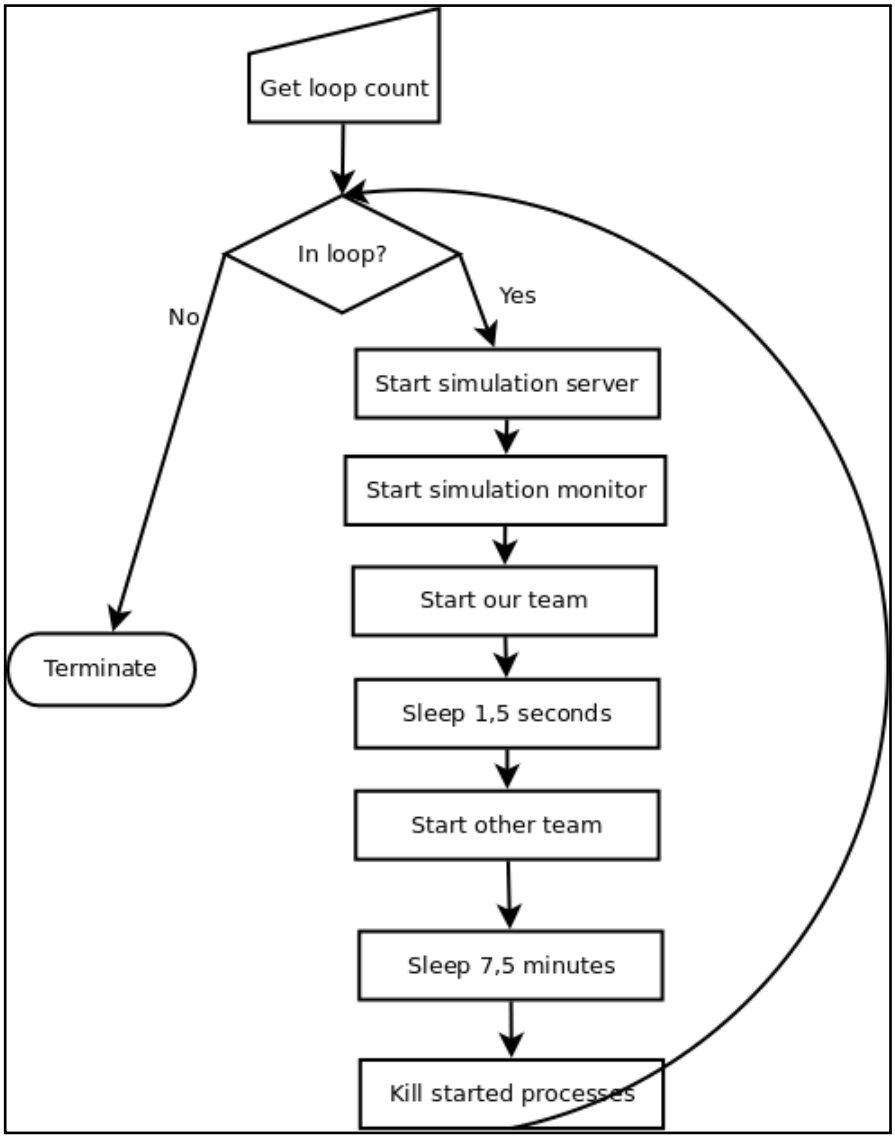

Figure 3.3 Flowchart of BatchMatch.

During a scenario, i. e. match, 17 corner kicks are played in average. Each corner kick and its settings are named as “scene”. In scene first players are positioned according to specific conditions and afterwards corner kick is kicked randomly. After kicking, game is played as a normal match. Scene ends in different ways. Scene ends normally in situations like “out”, “goal”, “kick in own”, “offside”. If ball passes to

our field scene ends with “counter attack”. If scene does not end in 30 seconds a “time out” condition occurs and scene ends. Except these conditions scene ends with “other”.

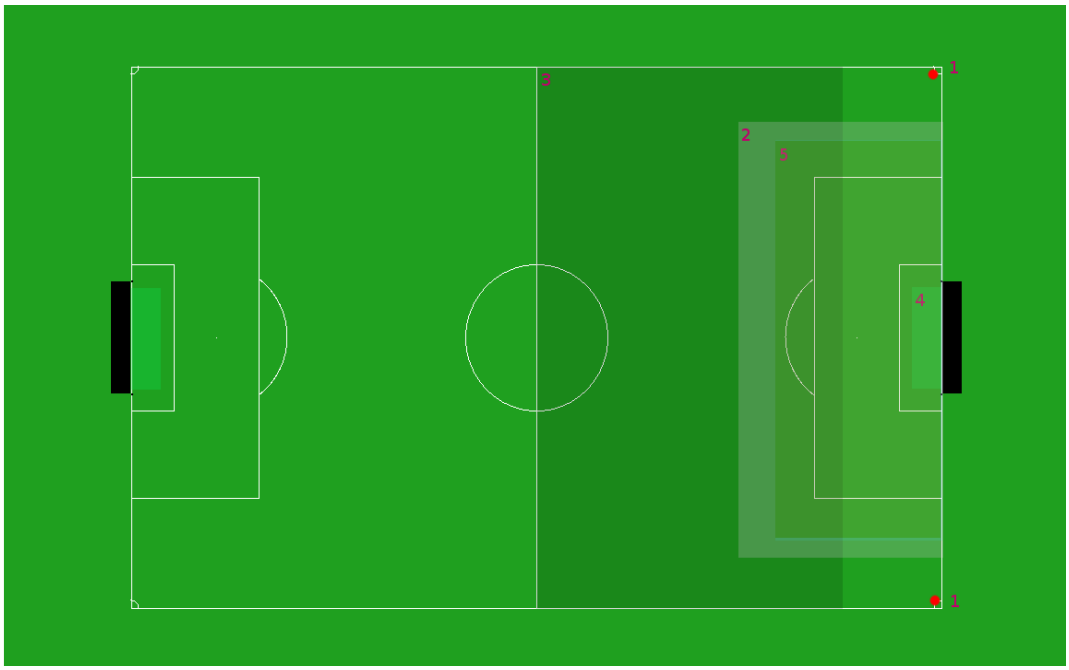

Figure 3.4 Positioning areas in scene.

In Figure 3.4 positioning areas used in the beginning of the scene are shown. In one of the points labeled with 1 ball is placed and behind the ball our kicker is placed. Our attackers are placed in the rectangle labeled with 2. Our goalkeeper and two defender players’ stays in our own field, the remaining are positioned inside the huge rectangle 3. Other goalkeeper is placed somewhere in rectangle 4. Other attackers are positioned in their own field and remaining players are placed in rectangle 5. Attackers and defenders are especially positioned separately and rectangles are intentionally intersected. So, all kind of combinations are allowed to be set.

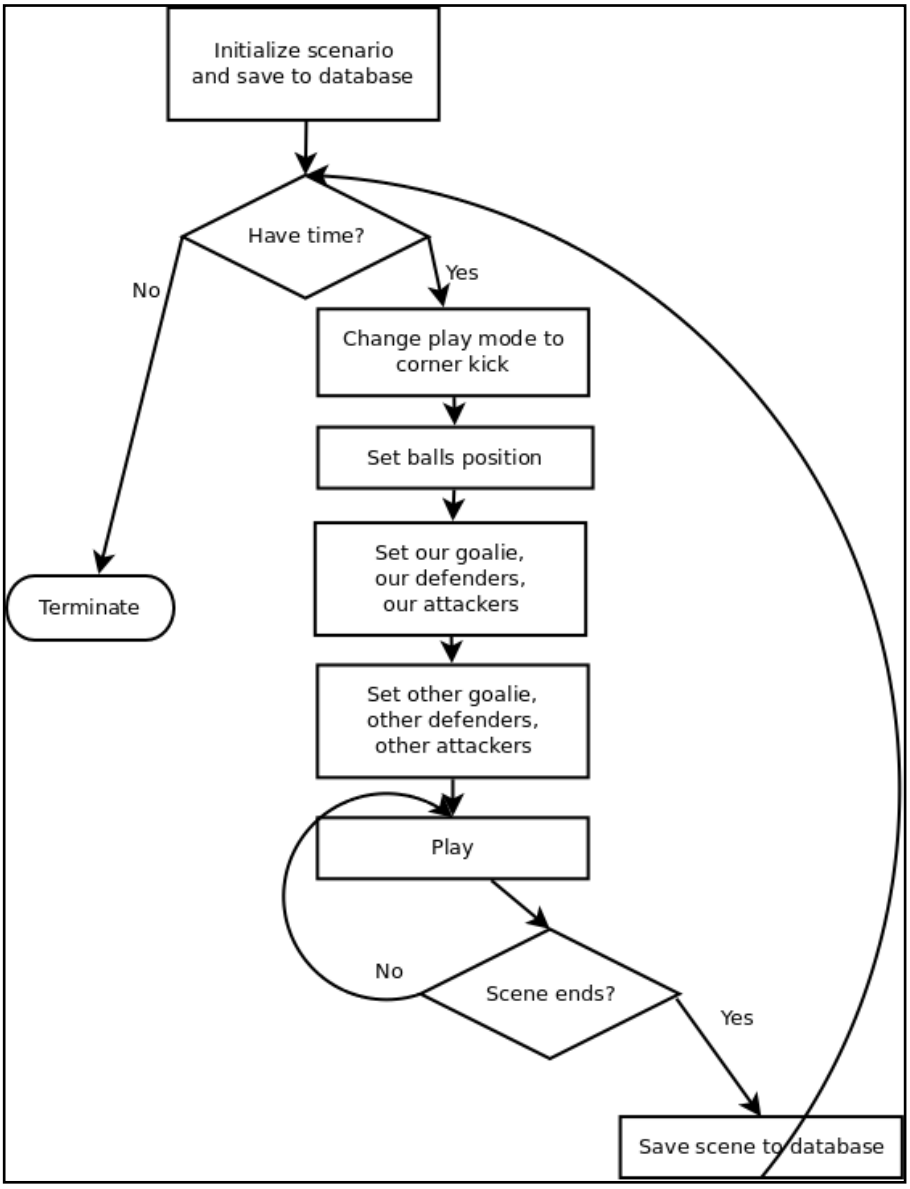

Figure 3.5 Flowchart of scenario.

Figure 3.5 shows the flowchart of scenario preparation. Scenario settings are saved first, than in a loop scenes are set. When a scene ends, its data is saved.

As classification needs huge amount of data, data is stored in a database. For database software SQLite is used. Serverless and simple architecture and its speed make it ideal solution.

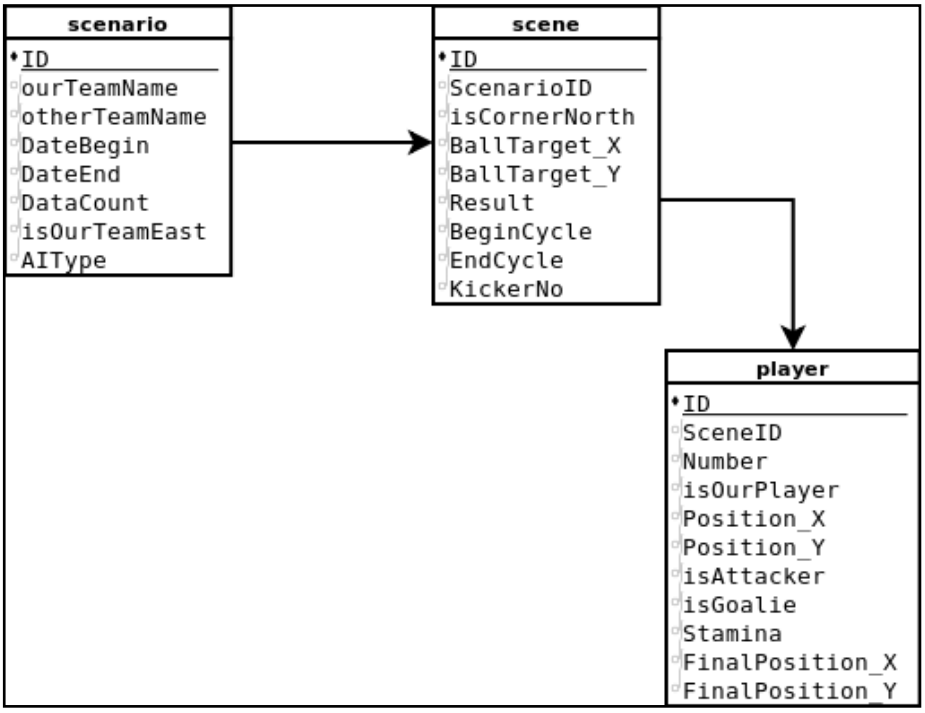

Figure 3.6 Database diagram

As stated in diagram, Figure 3.6, except explanatory data Scenario table stores our team's field which is necessary to interpret the coordinate system. In Scene table corner kick's area, target position, kicker player's number, result, whether players are tired and helper data are stored. Players are stored in a separate table with number, team, role, starting and ending positions.

In total, 2095 scenarios are run and 98168 times corner kick is played. Gathering training set took 280 hours and 47 scenes are played per scenario. Among played corner kicks 8.8% is resulted as goal, 3.6% is kick in, 1.8% is timeout. Totally, 14,2% of corner kicks are successful. More detailed information is given at Chapter 3.5.1.

3.4 Data Preprocessing

Classification system needs data in a proper form to operate and data is not ready to use as its raw form in database. Features are special abstract data representing the data. In this thesis, three feature sets are experimented.

All feature sets has one common feature which is “isTired”. IsTired represent whether the game is close to end. Game lasts for 3000 cycles and isTired feature uses the value 2200 for threshold. If the scene's ending cycle is later then 2200, then that scene's isTired value is true.

All feature sets are calculated with a separate software, FeatureProcess, and stored in database. At the same time players and ball target are drawn on the soccer field and scenes are saved as picture on disk. Scene figures shown in this thesis are automatically created. Whole feature processing process took up approximately 60 hours. Flowchart of FeatureProcess is shown in Figure 3.7.

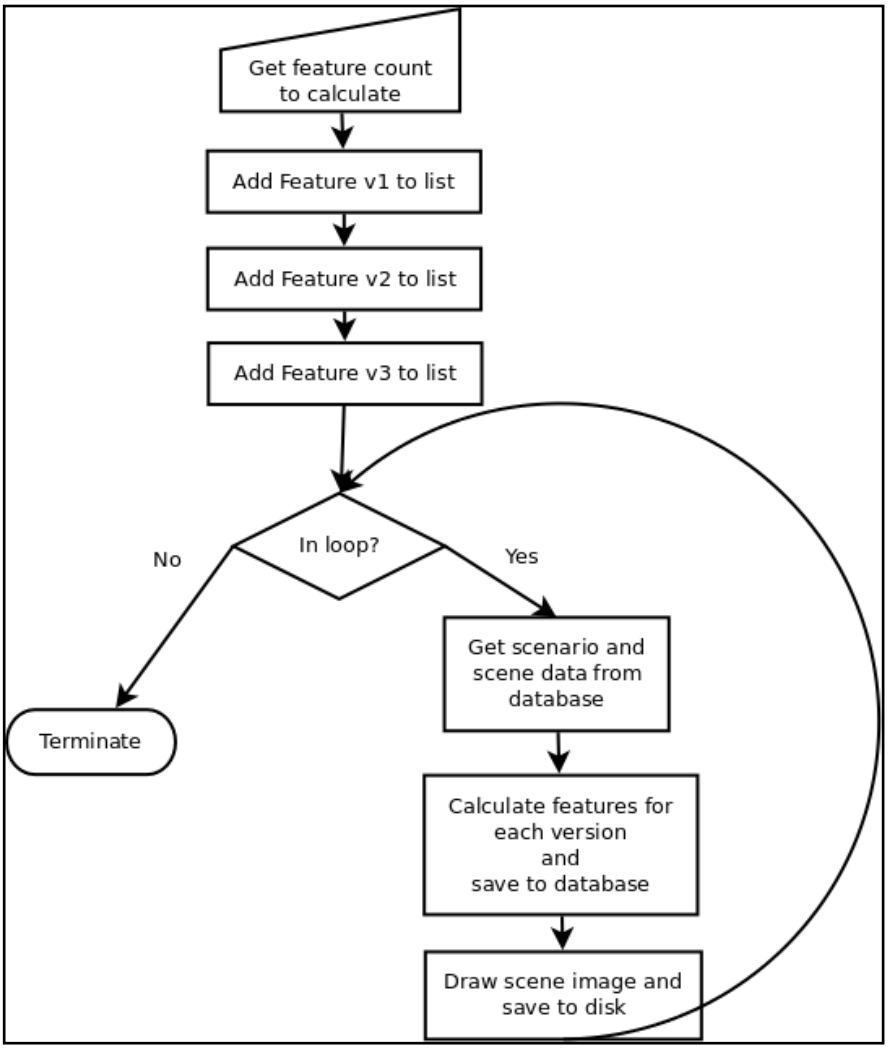

Figure 3.7 Flowchart of FeatureProcess

3.4.1 Feature Set 1

Feature set 1 (Fv1 from now on) is simply made up of coordinates of players. This feature set minimizes the abstraction of the raw data and aims to prevent possible information loss. Machine learning algorithm is expected to find patterns, relations with coordinates. All members of Fv1 are shown at Table 3.1.

Table 3.1 Feature set 1 (Fv1)

Our1_X	Other1_X
Our1_Y	Other1_Y
Our2_X	Other2_X
Our2_Y	Other2_Y
Our3_X	Other3_X
Our3_Y	Other3_Y
Our4_X	Other4_X
Our4_Y	Other4_Y
Our5_X	Other5_X
Our5_X	Other5_X
Our6_Y	Other6_Y
Our6_X	Other6_X
Our7_Y	Other7_Y
Our7_X	Other7_X
Our8_Y	Other8_Y
Our8_X	Other8_X
Our9_Y	Other9_Y
Our9_X	Other9_X
Our10_X	Other10_X
Our10_Y	Other10_Y
Our11_X	Other11_X
Our11_Y	Other11_Y

3.4.2 Feature Set 2

Feature set 2 (Fv2 from now on) is one level more abstract. Fv2 divides the penalty area and it's around into 18 areas and considers the player counts of two teams in these regions. With Fv2 classification about player density is planned. All members of Fv2 are shown at Table 3.2 and the areas on the field are shown in Figure 3.8.

Table 3.2 Feature set 2 (Fv2)

Our_1	Other_1
Our_2	Other_2
Our_3	Other_3
Our_4	Other_4
Our_5	Other_5
Our_6	Other_6
Our_7	Other_7
Our_8	Other_8
Our_9	Other_9
Our_10	Other_10
Our_11	Other_11
Our_12	Other_12
Our_13	Other_13
Our_14	Other_14
Our_15	Other_15
Our_16	Other_16
Our_17	Other_17
Our_18	Other_18

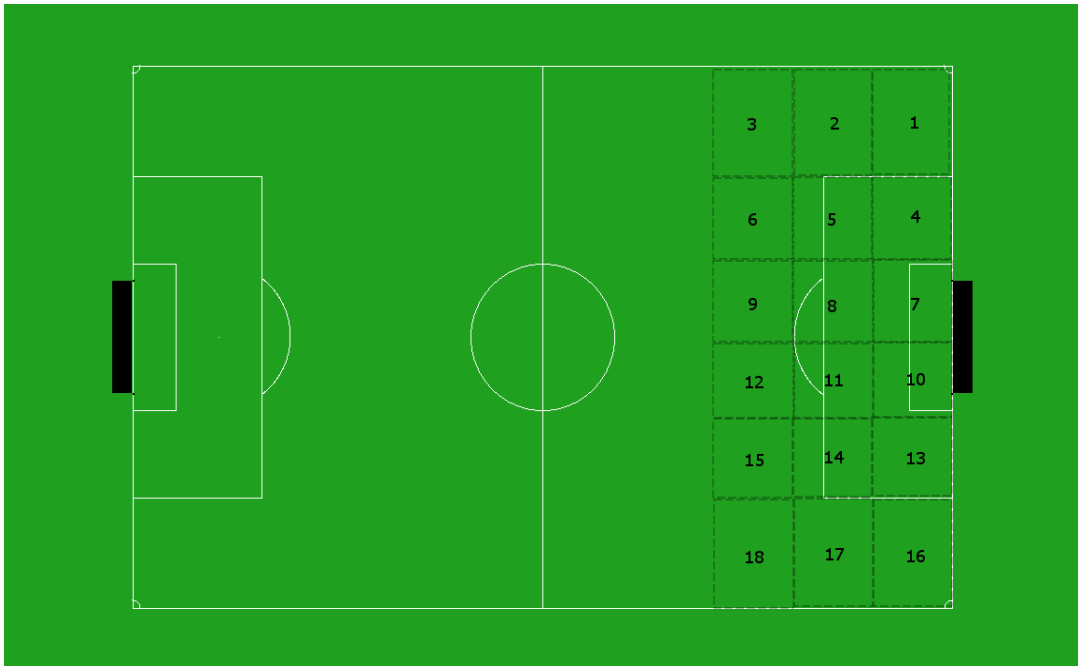

Figure 3.8 18 areas in Fv2

3.4.3 Feature Set 3

Feature set 3 (Fv3 from now on) is completely abstracted feature set. Fv3 aims to run the classification algorithm effectively by extracting key values from player's coordinates. Some features in Fv3 uses secondary directions. These directions are in the middle of the other direction pairs and used for reducing the overall feature count. Secondary directions are between the directions 1 and 2, 3 and 4, 5 and 6. All members of Fv3 are shown at Table 3.3.

Table 3.3 Feature set 3 (Fv3)

SD1_AnyMateToPass	SD2_AnyMateToPass
SD3_AnyMateToPass	SD1_FirstMatesPassableFriends
SD2_FirstMatesPassableFriends	SD3_FirstMatesPassableFriends
SD1_FirstMatesPassableFriendsAvg	SD2_FirstMatesPassableFriendsAvg
SD3_FirstMatesPassableFriendsAvg	Our_PC_Inside
Our_PC_Outside_Near	Our_PC_Outside_Far
Other_Goalie_Position	SD1_CanOpponentIntercept
SD2_CanOpponentIntercept	SD3_CanOpponentIntercept
SD1_IsShootingPossible	SD2_IsShootingPossible

SD3_IsShootingPossible	SD1_Our_PC_CloseToDirection
SD2_Our_PC_CloseToDirection	SD3_Our_PC_CloseToDirection
SD1_Other_PC_CloseToDirection	SD2_Other_PC_CloseToDirection
SD3_Other_PC_CloseToDirection	-

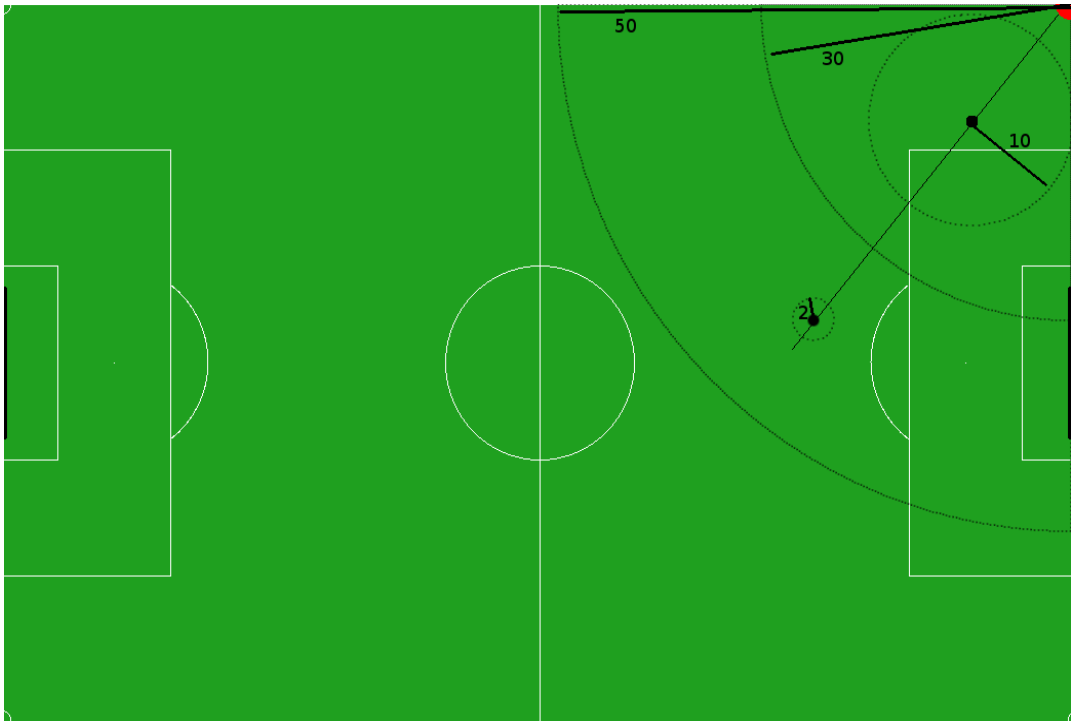

Figure 3.9 Some distances on the soccer field.

Figure 3.9 shows various distances on the field. Shown distances are used in calculation of Fv3.

3.4.3.1 Feature explanations

AnyMateToPass states whether a teammate exists that is close to the corner at most 30 units and close to the direction 10 units.

FirstMatesPassableFriends is the count of teammates that is close to teammates in 15 units which are in range of corner in 50 units and has a distance to the direction at most 10 units. When the value is divided by first mates count, then FirstMatesPassableFriendsAverage is obtained.

Our_PC_Inside states our player count in opponent penalty area. Our_PC_Outside_Near states our player count outside of the opponent penalty area but at most 10 units away in X direction and in the near vertical half of outside area. Similarly Our_PC_Outside_Far gives value of the far vertical half.

Other_Goalie_Position states whether the other goalkeeper is in center, in far part or in closer part.

CanOpponentIntercept states whether our teammate which receives the pass has any opponent not behind him and close to the direction less than 2 units.

IsShootingPossible states whether our teammate which receives the pass has any shoot opportunity. Two different shoot routes is drawn to the goal and if there is not any opponent close to the any route less then 2 units then shooting is considered as possible.

PC_Close_To_Direction is the players count that close to the direction less than 10 units of the teams.

3.5 Constructing Classifier

Before constructing the classifier, to be able to assess the prepared classifier, performance evaluation should be defined.

Figure 3.10 Division of data set.

As seen in Figure 3.10, 15% of data is separated for test set. Performance is evaluated according to both training and test sets. When comparing the performances of parameters, both performance values are considered.

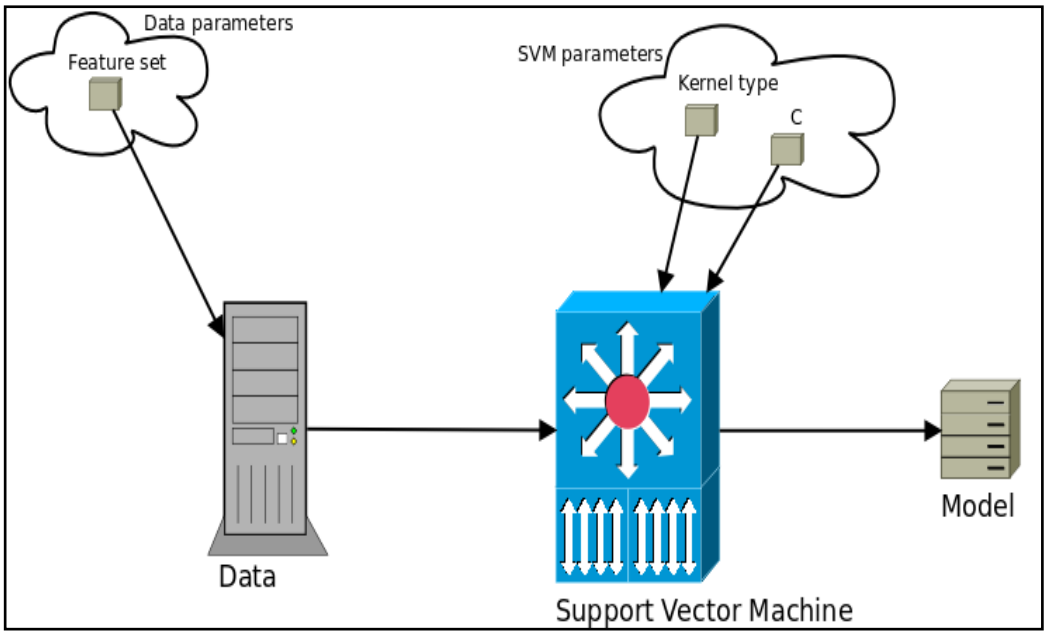

Figure 3.11 Model construction.

Figure 3.11 shows the organization of model construction. Support Vector Machine gets the data as the sole input. Data is the successful scenes mentioned before. During the data preprocessing, all scenes are labeled with the direction of the corner kicks kicked. SVM produces the model as the only output. Model is the artificial intelligence of team. Model will be used during the application of model. Experiments are conducted about both data and SVM. Detailed explanation about parameters and experiments are found in Chapter 3.5.2.

3.5.1 Data Characteristics, Statistics

Table 3.4 and Table 3.5 shows training set and test set respectively. Sum of training and test sets are shown at Table 3.6. At Table 3.7 whole data gathered is shown.

Table 3.4 Training set

Direction	Goal	Kick In Own	Time Out	Total
1	2098	337	269	2704
2	2366	614	376	3356

3	1290	503	271	2064
4	569	302	155	1026
5	584	516	171	1271
6	497	694	224	1415
Total	7404	2966	1466	11836

Table 3.5 Test set

Direction	Goal	Kick In Own	Time Out	Total
1	386	67	55	508
2	393	114	86	593
3	233	78	53	364
4	83	62	25	170
5	84	103	27	214
6	85	113	41	239
Total	1264	537	287	2088

Table 3.6 Total data set

Direction	Goal	Kick In Own	Time Out	Total
1	2484	404	324	3212
2	2759	728	462	3949
3	1523	581	324	2428
4	652	364	180	1196
5	668	619	198	1485
6	582	807	265	1654
Total	8668	3503	1753	13924

Table 3.7 Overall data set

Direction	Counter Attack	Goal	Kick In Own	Off Side	Other	Out	Time Out	Total
1	10645	2484	404	1325	753	1300	324	17235
2	16585	2759	728	2149	1242	1512	462	25437
3	12611	1523	581	1513	1023	914	324	18489
4	6031	652	364	737	641	356	180	8961

5	7907	668	619	1201	1114	466	198	12173
6	9856	582	807	1798	2007	558	265	15873
Total	63635	8668	3503	8723	6780	5106	1753	98168

Classifier is implemented with Support Vector Machine (SVM) algorithm in RapidMiner tool. During experiments, observed parameters are SVM Type, Kernel Type, C and Epsilon. The design screen of project is shown in Figure 3.12.

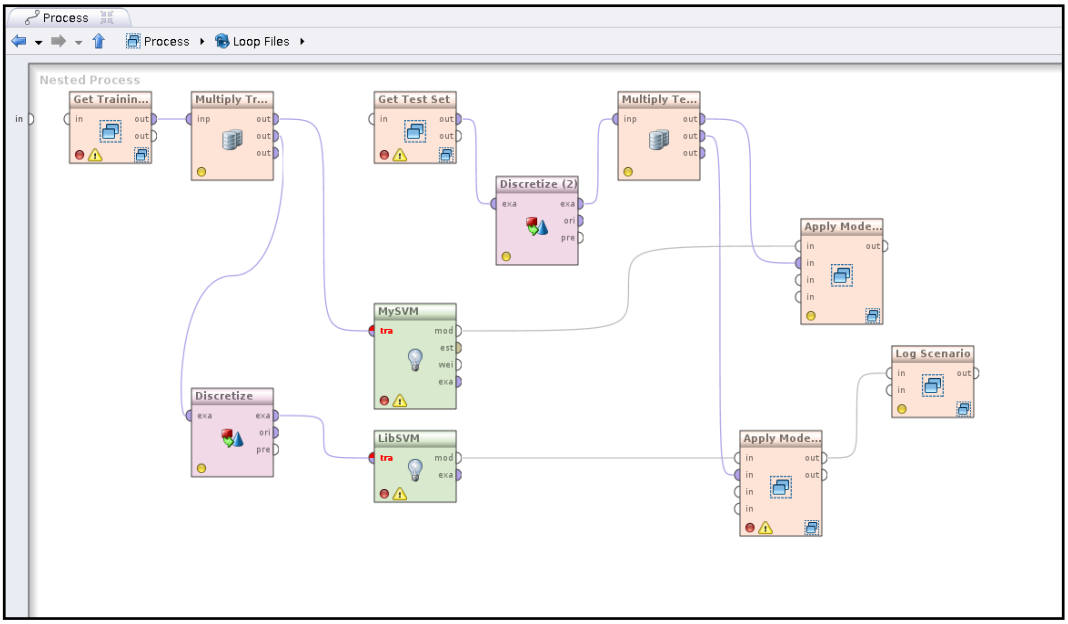

Figure 3.12 Implemented SVM classifier in RapidMiner

3.5.2 Experiments with Variables

3.5.2.1 SVM Type

SVM type is determined by kernel type, C and feature set experiments.

Table 3.8 SVM type versus kernel type and feature set

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
MySVM	Fv1	Dot	0,00	100.000	0,000	0,32	0,31
MySVM	Fv3	Dot	0,00	100.000	0,000	0,32	0,33

MySVM	Fv2	Dot	0,00	100.000	0,000	0,31	0,34
LibSVM	Fv1	Rbf	0,00		0,001	0,45	0,31
LibSVM	Fv3	Rbf	0,00		0,001	0,51	0,50
LibSVM	Fv2	Rbf	0,00		0,001	0,31	0,34
MySVM	Fv1	radial	0,00	100.000	0,000	0,54	0,17
MySVM	Fv3	radial	0,00	100.000	0,000	0,65	0,33
MySVM	Fv2	radial	0,00	100.000	0,000	0,54	0,16
LibSVM	Fv1	linear	0,00		0,001	0,28	0,29
LibSVM	Fv3	linear	0,00		0,001	0,56	0,57
LibSVM	Fv2	linear	0,00		0,001	0,28	0,28
MySVM	Fv1	polynomial	0,00	100.000	0,000	0,45	0,22
MySVM	Fv3	polynomial	0,00	100.000	0,000	0,38	0,35
MySVM	Fv2	polynomial	0,00	100.000	0,000	0,37	0,21
LibSVM	Fv1	Poly	0,00		0,001	0,28	0,29
LibSVM	Fv3	Poly	0,00		0,001	0,57	0,57
LibSVM	Fv2	Poly	0,00		0,001	0,39	0,27
MySVM	Fv1	epachnenikov	0,00	100.000	0,000	0,54	0,17
MySVM	Fv3	epachnenikov	0,00	100.000	0,000	0,56	0,19
MySVM	Fv2	epachnenikov	0,00	100.000	0,000	0,54	0,16
LibSVM	Fv1	sigmoid	0,00		0,001	0,22	0,22
LibSVM	Fv3	sigmoid	0,00		0,001	0,56	0,57
LibSVM	Fv2	sigmoid	0,00		0,001	0,28	0,28

Table 3.9 SVM type versus C and feature set

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
MySVM	Fv1	Dot	0,00	100.000	0,000	0,26	0,26
MySVM	Fv3	Dot	0,00	100.000	0,000	0,30	0,29
MySVM	Fv2	Dot	0,00	100.000	0,000	0,26	0,26
LibSVM	Fv1	Rbf	0,00		0,001	0,31	0,29
LibSVM	Fv3	Rbf	0,00		0,001	0,51	0,52

LibSVM	Fv2	Rbf	0,00		0,001	0,28	0,28
MySVM	Fv1	Dot	10,00	100.000	0,000	0,22	0,23
MySVM	Fv3	Dot	10,00	100.000	0,000	0,27	0,26
MySVM	Fv2	Dot	10,00	100.000	0,000	0,21	0,2
LibSVM	Fv1	Rbf	10,00		0,001	0,66	0,26
LibSVM	Fv3	Rbf	10,00		0,001	0,56	0,56
LibSVM	Fv2	Rbf	10,00		0,001	0,28	0,28
MySVM	Fv1	Dot	20,00	100.000	0,000	0,21	0,2
MySVM	Fv3	Dot	20,00	100.000	0,000	0,26	0,27
MySVM	Fv2	Dot	20,00	100.000	0,000	0,18	0,18
LibSVM	Fv1	Rbf	20,00		0,001	0,83	0,25
LibSVM	Fv3	Rbf	20,00		0,001	0,56	0,57
LibSVM	Fv2	Rbf	20,00		0,001	0,28	0,28
MySVM	Fv1	Dot	100,00	100.000	0,000	0,20	0,2
MySVM	Fv3	Dot	100,00	100.000	0,000	0,21	0,2
MySVM	Fv2	Dot	100,00	100.000	0,000	0,16	0,16
LibSVM	Fv1	Rbf	100,00		0,001	1,00	0,22
LibSVM	Fv3	Rbf	100,00		0,001	0,57	0,57
LibSVM	Fv2	Rbf	100,00		0,001	0,28	0,28
MySVM	Fv1	Dot	300,00	100.000	0,000	0,20	0,2
MySVM	Fv3	Dot	300,00	100.000	0,000	0,18	0,17
MySVM	Fv2	Dot	300,00	100.000	0,000	0,15	0,14
LibSVM	Fv1	Rbf	300,00		0,001	1,00	0,22
LibSVM	Fv3	Rbf	300,00		0,001	0,58	0,58
LibSVM	Fv2	Rbf	300,00		0,001	0,28	0,28

As obviously seen in Table 3.8, MySVM very easily overfits except “dot” kernel type. Also MySVM's performance is generally lower then LibSVM. From Table 3.9 it is understood that performance of MySVM is not good. Though LibSVM overfits

sometimes, it exceeds 50% but MySVM even cannot increase its performance to 50%. As a result, LibSVM is better for this thesis.

3.5.2.2 Feature Set

Feature set is decided by kernel type and C experiments.

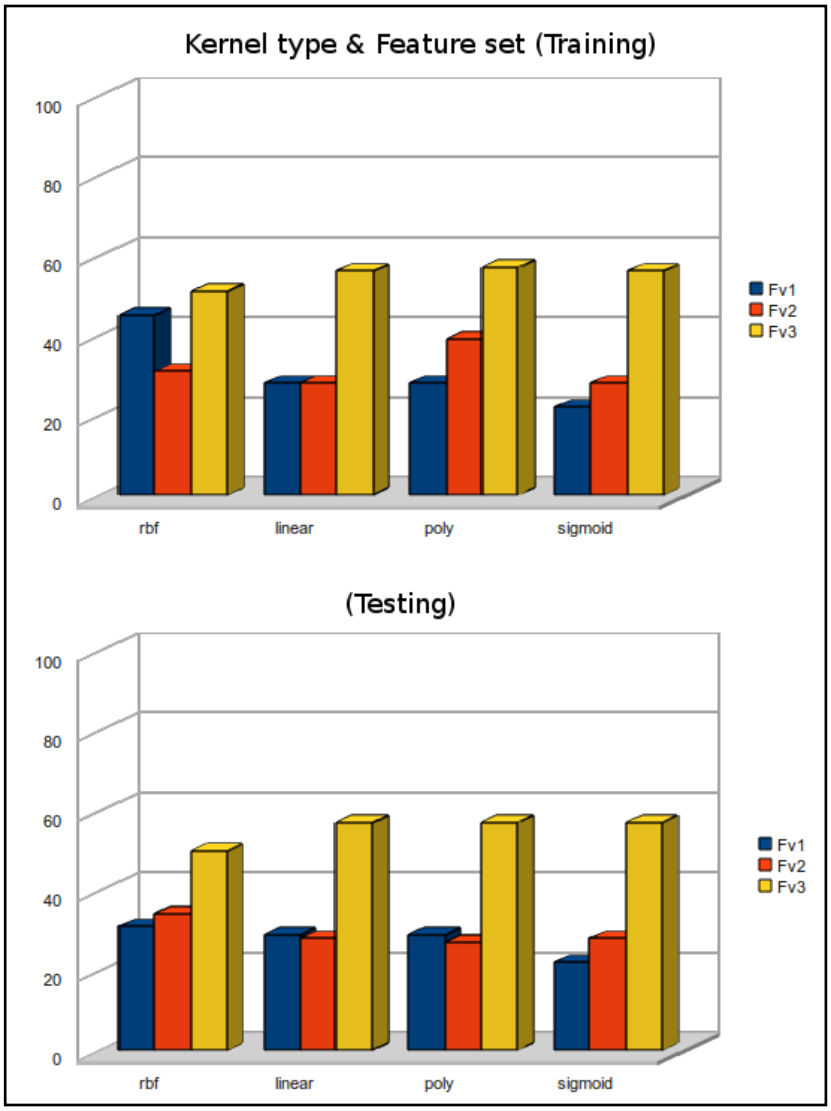

Figure 3.13 Performance chart according to kernel type and feature set.

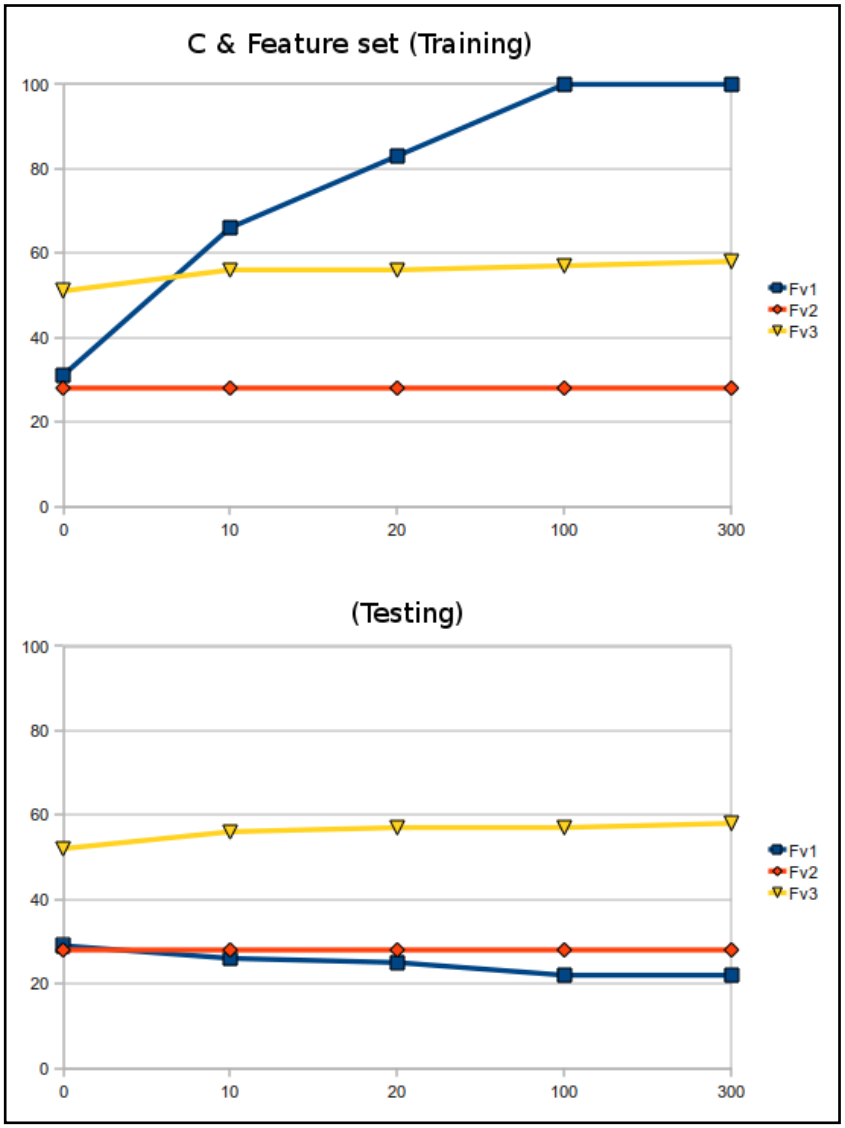

Figure 3.14 Performance chart according to C and feature set.

In Figure 3.13 Fv1 and Fv2 are equal, Fv3 is by far the best. In Figure 3.14 even Fv1 seems better then Fv2, it loses because of over fitting. Fv3 is chosen obviously. Whole detailed tables of charts are under Appendix.

3.5.2.3 Kernel Type

Kernel type is determined by C experiment.

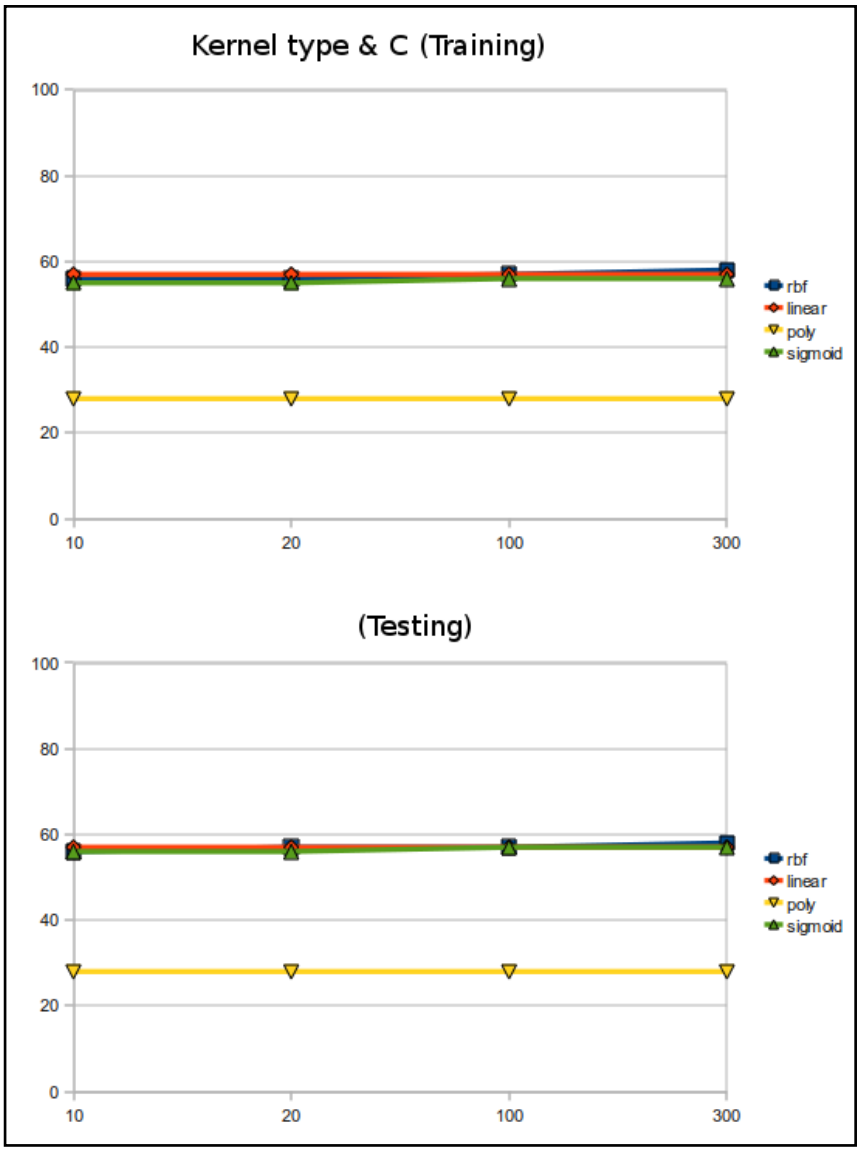

Figure 3.15 Performance chart according to C and kernel type.

In Figure 3.15 “poly“ is definitely bad. “linear” needs 7 times more time than others but no extra performance in return. When “sigmoid” compared to “rbf” it has slightly low performance but “rbf” chosen because it is a more general solution. Whole detailed tables of charts are under Appendix.

3.5.2.4 C

After determining SVM type, kernel type and feature set, C is experimented.

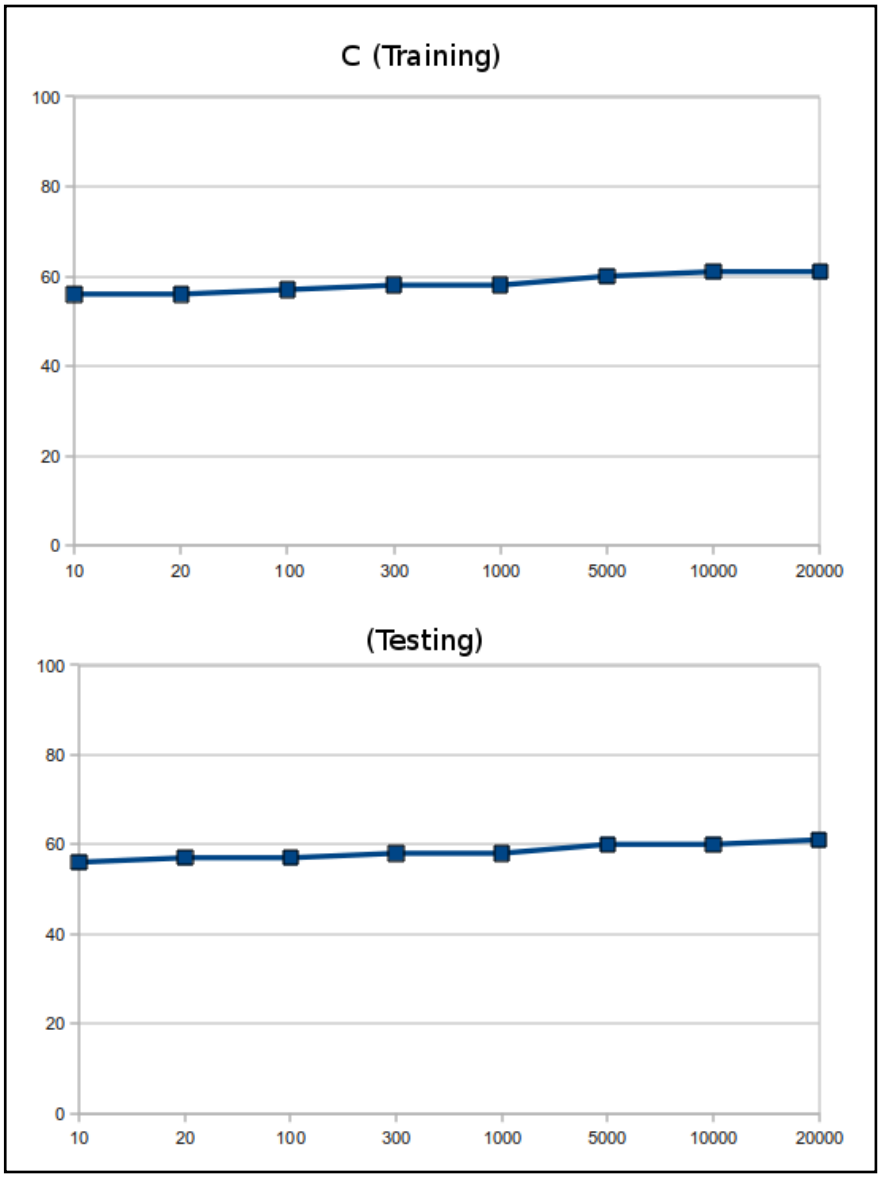

Figure 3.16 Performance chart according to C.

From Figure 3.16, performance and C seems to have direct proportion. But after a point, performance increment is not sufficient. So it is a good choice to set C equal 5000, not to let it increase more, as it is the most important reason of over fitting. Whole detailed tables of charts are under Appendix.

3.5.2.4 Epsilon

Epsilon is experimented with C.

Table 3.10 Epsilon versus C

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
LibSVM	Fv3	Rbf	0,00		0,010	0,51	0,52
LibSVM	Fv3	Rbf	0,00		0,100	0,51	0,52
LibSVM	Fv3	Rbf	0,00		0,500	0,52	0,52
LibSVM	Fv3	Rbf	300,00		0,010	0,58	0,58
LibSVM	Fv3	Rbf	1000,00		0,010	0,58	0,58
LibSVM	Fv3	Rbf	5000,00		0,010	0,60	0,6
LibSVM	Fv3	Rbf	300,00		0,100	0,57	0,58
LibSVM	Fv3	Rbf	1000,00		0,100	0,58	0,58
LibSVM	Fv3	Rbf	5000,00		0,100	0,60	0,6
LibSVM	Fv3	Rbf	300,00		0,500	0,57	0,58
LibSVM	Fv3	Rbf	1000,00		0,500	0,58	0,58
LibSVM	Fv3	Rbf	5000,00		0,500	0,60	0,6
LibSVM	Fv3	Rbf	1000,00		1,000	0,58	0,58
LibSVM	Fv3	Rbf	1000,00		2,000	0,58	0,57
LibSVM	Fv3	Rbf	1000,00		5,000	0,12	0,12

There is almost no difference for several values of epsilon, seen from Table 3.10. So default value 0.001 is leaved as it is.

3.5.3 Summary

Classifier is constructed with the parameters LibSVM, Fv3, rbf, C = 5000 and epsilon = 0.001 and achieved training and test performances of 60%. Performance is enough for such kind of classifications. Increasing the performance carries a huge

risk of over fitting and probably when it is applied to real environment, performance drops dramatically, just like in the experiments.

3.5.4 Applying Model

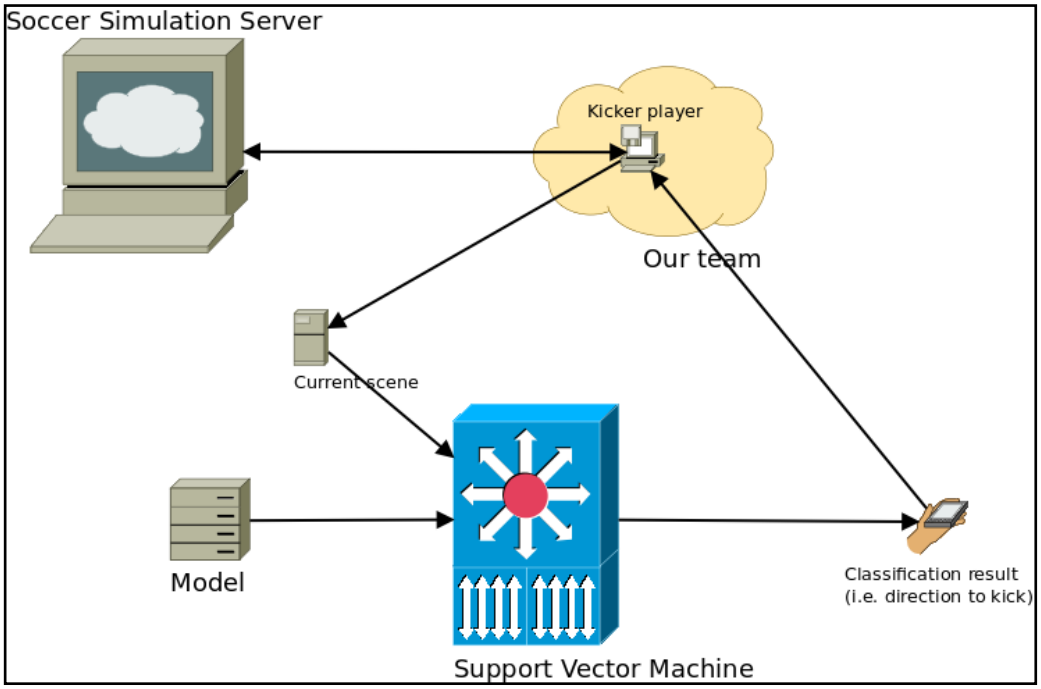

Figure 3.17 Model application.

Figure 3.17 shows the organization of model application. When a player, kicker, needs to decide where to kick the corner kick, player uses the decision process described in Figure 3.17. SVM needs two parameters to apply the model, the model and data. Model is produced before, in model construction phase. Kicker player prepares scene data and same preprocessing step is applied to the scene data. SVM is run with model and processed scene data and produces the direction as output which is the classification result. Using the classification result, kicker players kicks the ball into the classified direction.

Constructed model is integrated into Dainamite's source code, as it will use RapidMiner. After 9000 corner kicks, the result is being shown in Table 3.11.

Table 3.11 Applying the model

Direction	Counter Attack	Goal	Kick In Own	Off Side	Other	Out	Time Out	Total
1	792	305	59	140	65	149	62	1572
2	1199	323	91	229	104	191	82	2219
3	932	215	83	205	125	108	63	1731
4	483	92	40	64	51	57	6	793
5	576	74	49	157	117	54	29	1056
6	743	81	94	188	214	79	38	1437
Total	4725	1090	416	983	676	638	280	8808

Successful results to be calculated, Dainamite's performance in corner kicks increased from 14.2% to 20.2%, i.e. there is a performance increment like 40%.

When constraints of Dainamite team are considered with Wright Eagle's perfection, it can be said that Dainmite's passing is really bad and Wright Eagle mostly counter attacks. With this improvement, simple passing errors are decreased and inherently there is chance to use the ball more wisely.

CHAPTER FOUR
CONCLUSION

In this thesis, it is aimed to decide where to kick the ball in a corner kick. RoboCup soccer simulation server and RoboCup teams are used. Decision making is performed with Support Vector Machine classification which implemented in Rapid Miner software.

Raw data is converted into meaningful values called features. Three different feature sets are proposed and experimented. A very big data set about a size of 98.168 corner kicks is gathered. During data gathering, a secondary software is used to play matches consecutively. 15% of the data set is separated as test set and remaining is used as training set. Performance is evaluated on both training and test sets.

Support Vector Machine is constructed and the model is produced. Classification performance on training and test sets are 60%. Experiments about SVM type, feature set and SVM parameters are performed. LibSVM as SVM type, feature version 3 as feature set are preferred. For SVM parameters, rbf for kernel type, 5000 for C, 0.001 for epsilon is preferred. After model's production, new corner kicks are generated and model is applied in decision process. Eventually, performance is increased.

Dainamite team cannot play well, cannot react to ball and opponents, testing could not be done in real environment. Testing in real environment instead of randomly positioned laboratory conditions would give a more solid result, and probably a better success rate. When players positioned randomly, such positions are set that Dainamite team cannot achieve that positions against Wright Eagle in normal game and these positions exists in data sets. This kind of scenes may not occur in normal games but due to poor passing and reacting skills of Dainamite testing could be only done in random positioning. There is no more teams which has open source available and compatible to latest simulation server version so another team could not be used instead of Dainamite.

4. 1 Future Work

As new teams emerged in RoboCup or existing teams opened their code to public research, new teams may be included in data set in the future. Each summer RoboCup tournaments are held. After the tournament, all teams are supposed to be compatible to the latest soccer server version. New teams may be used to extend the dataset.

Corner kick in soccer can be improved by set plays. As mentioned before, positioning of players excluded but in future it may also be implemented although it is definitely a separate subject to study on. To implement positioning, more teams and obviously more corner kicks will be needed. In that case a more compressive modeling shall be designed.

APPENDIX A

EXPERIMENT RESULTS

Table A.1 Kernel type and feature set

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
LibSVM	Fv1	Rbf	0,00		0,001	0,45	0,31
LibSVM	Fv3	Rbf	0,00		0,001	0,51	0,50
LibSVM	Fv2	Rbf	0,00		0,001	0,31	0,34
LibSVM	Fv1	linear	0,00		0,001	0,28	0,29
LibSVM	Fv3	linear	0,00		0,001	0,56	0,57
LibSVM	Fv2	linear	0,00		0,001	0,28	0,28
LibSVM	Fv1	poly	0,00		0,001	0,28	0,29
LibSVM	Fv3	poly	0,00		0,001	0,57	0,57
LibSVM	Fv2	poly	0,00		0,001	0,39	0,27
LibSVM	Fv1	sigmoid	0,00		0,001	0,22	0,22
LibSVM	Fv3	sigmoid	0,00		0,001	0,56	0,57
LibSVM	Fv2	sigmoid	0,00		0,001	0,28	0,28

Table A.2 Feature set and C

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
LibSVM	Fv1	Rbf	0,00		0,001	0,31	0,29
LibSVM	Fv3	Rbf	0,00		0,001	0,51	0,52
LibSVM	Fv2	Rbf	0,00		0,001	0,28	0,28
LibSVM	Fv1	Rbf	10,00		0,001	0,66	0,26
LibSVM	Fv3	Rbf	10,00		0,001	0,56	0,56
LibSVM	Fv2	Rbf	10,00		0,001	0,28	0,28

LibSVM	Fv1	Rbf	20,00		0,001	0,83	0,25
LibSVM	Fv3	Rbf	20,00		0,001	0,56	0,57
LibSVM	Fv2	Rbf	20,00		0,001	0,28	0,28
LibSVM	Fv1	Rbf	100,00		0,001	1,00	0,22
LibSVM	Fv3	Rbf	100,00		0,001	0,57	0,57
LibSVM	Fv2	Rbf	100,00		0,001	0,28	0,28
LibSVM	Fv1	Rbf	300,00		0,001	1,00	0,22
LibSVM	Fv3	Rbf	300,00		0,001	0,58	0,58
LibSVM	Fv2	Rbf	300,00		0,001	0,28	0,28

Table A.3 Kernel type and C

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
LibSVM	Fv3	Rbf	10,00		0,001	0,56	0,56
LibSVM	Fv3	Linear	10,00		0,001	0,57	0,57
LibSVM	Fv3	Poly	10,00		0,001	0,28	0,28
LibSVM	Fv3	sigmoid	10,00		0,001	0,55	0,56
LibSVM	Fv3	Rbf	20,00		0,001	0,56	0,57
LibSVM	Fv3	Linear	20,00		0,001	0,57	0,57
LibSVM	Fv3	Poly	20,00		0,001	0,28	0,28
LibSVM	Fv3	sigmoid	20,00		0,001	0,55	0,56
LibSVM	Fv3	Rbf	100,00		0,001	0,57	0,57
LibSVM	Fv3	Linear	100,00		0,001	0,57	0,57
LibSVM	Fv3	Poly	100,00		0,001	0,28	0,28
LibSVM	Fv3	sigmoid	100,00		0,001	0,56	0,57
LibSVM	Fv3	Rbf	300,00		0,001	0,58	0,58
LibSVM	Fv3	Linear	300,00		0,001	0,57	0,57
LibSVM	Fv3	Poly	300,00		0,001	0,28	0,28
LibSVM	Fv3	sigmoid	300,00		0,001	0,56	0,57

Table A.4 C

svm type	file	kernel type	C	max iterations	epsilon	training perf.	test perf.
LibSVM	Fv3	Rbf	10,00		0,001	0,56	0,56
LibSVM	Fv3	Rbf	20,00		0,001	0,56	0,57
LibSVM	Fv3	Rbf	100,00		0,001	0,57	0,57
LibSVM	Fv3	Rbf	300,00		0,001	0,58	0,58
LibSVM	Fv3	Rbf	1000,00		0,001	0,58	0,58
LibSVM	Fv3	Rbf	5000,00		0,001	0,60	0,6
LibSVM	Fv3	Rbf	10000,00		0,001	0,61	0,6
LibSVM	Fv3	Rbf	20000,00		0,001	0,61	0,61

REFERENCES

Busuttil S. (2003). *Support vector machines*. Retrieved September 10, 2010, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.7930.

Chang, C. C., Lin, C. J. (2001). *LIBSVM: a Library for Support Vector Machines*. Retrieved Jun 18, 2010, from http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

Chen M., Foroughi E., Heintz F., Huang Z. X., Kapetanakis S., Kostiadis K., Kummeneje J., Noda I., Obst O., Riley P., Steffens T., Wang Y., & Yin X.. (July 26, 2002). *RoboCup Soccer Server User Manual: for Soccer Server version 7.08.* Retrieved May 2, 2010, from http://sourceforge.net/projects/sserver.

Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning, 20,* 273-297.

Joachims, T. (1999). Advances in kernel methods: Support vector learning. Making large-Scale SVM Learning Practical. *In Advances in Kernel Methods - Support Vector Learning*. 169-183

Howley, T., & Madden, M. G. (2005). The genetic kernel support vector machine: Description and evaluation. *Artificial Intelligence Review, 24 (3),* 379-395.

Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 20,* 415-425.

Kitano H., & Asada M. (1998). RoboCup Humanoid Challenge: That's One Small Step for A Robot, One Giant Leap for Mankind. *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-98)*

Osuna, E. E., Freund R., & Girosi F. (March 1997). *Support vector machines: Training and applications*. Retrieved September 10, 2010, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.418.

Riley, D. (2009). *Among American Kids Ages 2-17, 82 Percent Report They Are Gamers*, retrieved Sep. 5, 2010, http://www.npd.com/press/releases/press_091202.html.

Ross, D. (2010). *Great apes 'play' tag to keep competitive advantage*, retrieved Sep. 7, 2010, http://www.port.ac.uk/aboutus/newsandevents/frontpagenews/title,115359,en.html.

Shi, K., Bai, A., Tai, Y., & Chen, X. (Mar., 2009). *WrightEagle2009 2D Soccer Simulation Team Description Paper*. Retrieved Feb., 10, 2010, from http://www.wrighteagle.org/en/robocup/2D/tdps/WrightEagle2009_2D_Soccer_Simulation_Team_Description_Paper.pdf.

Taylor, J. B., James, N., & Mellalieu, S. D. (2003). 33 - Notational Analysis of Corner Kicks in English Premier League Soccer. *Science and football,2003.* (5), 225-228

Vapnik, V. (1998). *Statistical Learning Theory*. USA:Wiley.

Zanjani, M. A., Saharkhiz, S., Nosrati, M., Ghanizadeh, E., Bakhtiarnia, A., Maleki, M. R., Montazeri, M. R., Bakhtiari, M., Kaviani, P. (Mar., 2009). HelliBASH 2D Soccer Simulation Team Description Paper. Retrieved Feb., 10, 2010, from http://romeo.ist.tugraz.at/robocup2009/tdps/hellibash-rc09-tdp.pdf.