

UNIVERSITY OF WATERLOO

Faculty of Mathematics

**An Application of Neural Networks in
Learning of RoboCup Soccer Playing Team Strategy**

Date: Oct 1, 2002

Group Members: Kai-min Kevin Chang 99354459
Kevin Lai 98189912

Supervisor: Dr. Mohamed Kamel

Table of Contents

1. Problem Statement.....	1
2. Background	2
2.1 Platform.....	2
2.2 Literature Survey	4
3. Design.....	7
3.1 Simplification	7
3.1.1 Information	7
3.1.2 Control.....	7
3.1.3 Objectives	8
3.2 Neural Network	9
3.2.1 Types of Network.....	9
3.2.2 Design Goals.....	11
3.2.3 Models.....	12
3.3 Data Extraction.....	16
3.4 Choice of Programming Language.....	18
4. Software Architecture	20
4.1 Architecture	20
4.2 Client - Java.....	22
4.3 Global Positioning System – C++	24
4.4 Interoperability	25
5. Implementation	27
5.1 Neural Network	27
5.1.1 An Implementation of Neural Network Using Java.....	27
5.1.2 Exported Code from Neural Ware	29
6. Experiment	30
6.1 Manipulation	30
6.1.1 Autonomous Model – Varying Hidden Layers	30
6.1.2 Central Model – Varying Training Set Representation	31
6.2 Method	32
6.2.1 Neural Ware Setup	32
6.2.2 Training Set	33
6.2.3 Team Setup	34
6.2.4 Measure	34
6.3 Result	35
6.3.1 Original Game Play.....	35
6.3.2 NN Client	37

6.4 Debug.....	39
7. Conclusions.....	40
8. References.....	41
Appendix A – GPS.....	42
GlobalMapServer.java.....	42
GlobalMapServer.h.....	44
GlobalMapServerImp.cc.....	46
GlobalMap.h.....	48
GlobalMap.cc.....	49
Appendix B – NN.....	55
Recurrent.java.....	55
Backpropagation.java.....	61
Appendix C – NN Weights.....	64
C.1 A3P0.....	64
C.2 A3P1.....	65
C.3 A3P2.....	67
C.4 C3P0.....	69
C.5 C3P1.....	71
C.6 C3P2.....	74
C.7 C3P3.....	76
Table 1 Comparison of the Mirostot and RoboCup platforms.....	2
Table 2 Language Comparison Matrix.....	19
Table 3 Structures.....	32
Figure 1 Central processing model.....	13
Figure 2 Autonomous, fully omniscient agent model.....	14
Figure 3 Autonomous, partially omniscient agent.....	15
Figure 4 A snap shot of the visualization of RoboCup Simulation game through SoccerMonitor.....	17
Figure 5 Central NN Model.....	21
Figure 6 Individual NN Model.....	21
Figure 7 Atan overview.....	23
Figure 8 Java Virtual Machine and Soccer Server overview.....	25
Figure 9 Field description.....	35
Figure 10 Example of passing and solo attacking.....	36
Figure 11 Player stickiness.....	36

1. Problem Statement

The goal of this project is to investigate the application of neural networks (NN) in learning a team strategy for the soccer-playing robots in the RoboCup competition.

Different models of NN clients will be investigated, namely the central processing model and the autonomous agent model. Although not an explicit goal, the application should be designed with the possibility of online learning in future enhancement.

2. Background

2.1 Platform

RoboCup and Mirobot are the two most prominent platforms for the soccer-playing robots. Although both platforms attempt to facilitate the advancement of robotics through the simulation of a soccer game, they vary in terms of the modelling complexities and the underlying assumptions. The chart below compares the two platforms in their most generic settings.

	Mirobot	RoboCup Simulation
Accessibility	Complete	Incomplete (Players can only sense their surroundings)
Memory	Shared	Non-shared
Processor	Central	Autonomous agent
Number of agents	~3	~11
Perceived information	Simple (Coordinates, velocity and direction of the ball, and players of each team)	Complex (All of Mirobot, plus miscellaneous information such as wind, stamina, head, and body direction, etc.)
Control of the agent	Simple (Left and right spins of the robots)	Complex (Built-in action command such as move(), dash(), kick(), pass(), etc.)

Table 1 Comparison of the Mirobot and RoboCup platforms

Mirosot platform presents a simplified world to work on. Complete accessibility of the world and the sharing of memory facilitate the possibility of a central processor. The relative simple information perceived and ease of control further simplifies the problem. In the early stage of the development, these simplifications enable fast convergence of result. One disadvantage of the Mirosot platform is that it is less well-known than the RoboCup platform. Relatively little documentation has been published.

In contrast, RoboCup is the more complicated platform. Truly autonomous robot agents are intended because of the incomplete accessibility and non-shared memory assumptions. In addition, the types of information perceived by the agents and the controls of agents are more complicated. Despite the challenging assumptions, the RoboCup platform is the more developed platform among the two. Substantial theories based on RoboCup have been published, in addition to the archived descriptions, analysis and competition logs of all teams in previous competitions. A full-scale simulator suite has also been developed for the official simulation league and can be easily employed for the present research.

Currently, the PAMI Lab in University of Waterloo owns a set of Mirosoft competition equipments, which includes physical robots and a soccer field platform. Nevertheless, because the present research focuses on strategic learning, a physical platform can be omitted. In fact, a software simulator provides an easier interface to the learning program because the complexities of the physical equipments are avoided. Instead of dealing with mechanical matters, the research can focus on the development of NN, which is already complicated in its own ways.

We decide to combine and adopt the advantages of the two platforms. In particular, the simulator program of the RoboCup platform is employed, with the simplified working assumptions of the Mirosoft platform. The simplifications made will be discussed in the following section.

2.2 Literature Survey

Because RoboCup Simulation League platform has been established for over a number of years, many software packages already exist. In our literature survey, we have found two packages on the Internet which can assist our research.

Atan Package

Atan is a Java implementation of the RoboCup client front end. A client front end shields out the mundane details regarding the communication protocol with the server.

Thus, a native Java object such as “player” can be used to directly control a player.

URL: <http://vsoc.sourceforge.net/atan/>

Contact: Wolfgang Wagner wolfgang.wagner@iname.com

VSOC Package

VSOC is a hybrid approach to RoboCup simulation league. A combination of neural network and genetic algorithm is employed. The idea underpinning VSOC is the use of “training camp”; different training camps are created with different objectives. For example, the “attacker camp” may have the objective to score, while the “defender camp” emphasizes on driving the ball away from the home goal. When a match is needed, a team will “recruit” different players from different camps, accordingly to the preferable style of play (e.g. offensive vs. defensive style). After a match, the winning team will then subject to a selection process by the genetic algorithm. Thus, the optimal playing team is bound to objective machine selection, instead of biased human selection. The idea of the training camp and the recruiting process is very similar to the strategy adopted by the soccer teams in the real world.

However, the lack of training as a team is inconsistent with the present research.

Nonetheless, VSOC package presents a good starting point to the NN approach. In particular, the VSOC project suggests, through experimentations, that the optimal structure of the feed forward network is the network without any hidden layers.

URL: <http://vsoc.sourceforge.net/index.html>

Contact: Wolfgang Wagner wolfgang.wagner@iname.com

3. Design

3.1 Simplification

3.1.1 Information

In general, three types of *dynamic* information can be perceived by a player, namely the coordinates, velocities, and directions of any moving objects. Unfortunately, the use of all information overwhelms NN learning. In order to facilitate the convergence speed of NN learning, we decide to employ only the coordinates information and discard the velocities and directions information. Nonetheless, more information can be used if necessary. The amount of information perceived by an agent is a parameter that can be manipulated.

3.1.2 Control

Intermediate control is assumed. In the Mirosoft platform, the most basic operations of left and right spins of the robot's wheels are used. All actions of an agent are composed by varying the spins of the wheels. However, such low level controls are very difficult and inefficient to use. Thus, we take advantage of the RoboCup's action commands, namely `move()` and `kick()`. Abstraction at this level makes the controls easier. The parameters of each function are specified as follows.

move (x, y)

kick (*direction, power*)

3.1.3 Objectives

A close examination of previous RoboCup competitions reveals one important factor of success. Winning teams obtain better result because of the more effective passing strategy. Since passing the ball by definition requires cooperation of at least two players, the pass action presents a good testing bed for cooperative agents. Thus, effective passing strategy of a team is to be investigated.

The emphasis on the passing action further simplifies the problem domain. On one hand, concrete evaluation function of “an effective pass” can be composed. On the other hand, the simplification facilitates the data collection process. Depending on the evaluation function, a “training record” (unit of training data; NN learning may consume thousands of records in order to converge) can be defined. For example, a training record can be composed of the duration from the possession of the ball until the loss of the possession.

Some ideas of the evaluation function of passing are brain-stormed.

f_1 = Amount of time the ball is in possession

f_2 = Total distance the ball has been passed

f_3 = How close the ball is dribbled toward the enemy goal.

The final evaluation is then an aggregate function of weighted f_1 , f_2 and f_3 .

$$F = w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3$$

Unfortunately, we are unable to apply the evaluation function in the current study.

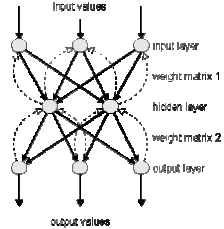
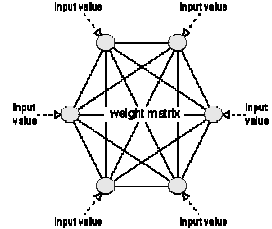
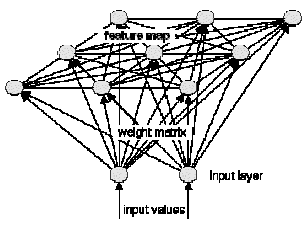
As later experimentation reveals, temporal information is difficult to be encoded in a back propagation NN. In our final experiment, one training record is simply a “snapshot” of the state of game play, which includes coordinates of all moving objects (e.g. players and the ball). More details will be provided in the following section.

3.2 Neural Network

3.2.1 Types of Network

There are many types of NNs designed with different objectives. In particular, three most commonly used networks are reviewed in the present study: the back propagation network by Hinton, Rumelhart & Williams (1986), the Hopfield net by Hopfield (1982) and the Kohonen Feature Map by Kohonen (1982). A table

describing major features of each network is included as follows.

	Back Propagation	Hopfield	Kohonen Feature Map
Diagram			
Type	feed forward	recurrent	recurrent
Layers	1 input layer 1 or more hidden layers 1 output layer	1 matrix	1 input layer 1 map layer
Learning Rule	delta rule	Hebb Rule	self organization
Learning Method	Supervised	unsupervised	unsupervised
Application	complex logical operations pattern classification behavior learning	pattern association optimization problems	pattern classification optimization problems simulation

* The table and diagrams are adopted from Fröhlich, J. (1999)

After consulting Dr. Kamel and Nayer Wanas, we have decided to employ the back propagation network because of the following reasons. First, the back propagation network is the most generic network among the three. While Hopfield net and Kohonen map are designed with specific applications in sight, the back propagation provides a black box approach to any problem. As long as the input and output are specified, the specific details of the network's internal structure can be omitted. [Note, later experimentation refutes the claim.] Second, the back propagation network is the most widely studied network and it is also the simplest to implement.

Extensive literature on back propagation network exists to aid the implementation.

3.2.2 Design Goals

There are two major goals in the design of the NN:

1. *Weighted game*

The idea of weighted games should be employed. When the NN is being trained, each training trial will be weighted by a weighting function. For example, older, inferior games are weighted less, while newer, superior games are weighted more.

$$weight(game) = \frac{1}{w_1 \cdot date_{game} + w_2 \cdot rank_{game}}$$

In particular, if online learning is made possible in the future, the current game can be assigned to a huge weight, which helps the network to quickly adapt to the present opponent. Although online learning in a single game may be extremely difficult, improvement over a match (a series of games against the same opponent) can certainly be foreseen.

Unfortunately, we never have the chance to experiment with the idea of weighted game in the current research.

2. *Efficient training*

In order for fast convergence of learning, the NN must be designed with simplicity.

Fast convergence of learning is the only way to achieve online-learning.

Although online-learning is not an explicit goal of the project, this future extensibility must be apparent during the design phase.

3.2.3 Models

In this section, three models of the network structure are proposed. We determine the number of input and output nodes for the black-box NN module.

Let

$$p = \text{number of players on a team} = 11$$

$$f = \text{number of players in focus} = 4 \quad (\text{for autonomous, partially omniscient model})$$

$$I = \text{coordinate} - x + \text{coordinate} - y \quad [+ \text{velocity} + \text{direction}] = 2$$

$$O = \text{coordinate} - x + \text{coordinate} - y \quad [+ \text{velocity} + \text{direction}] = 2$$

1. Central processing model

With the assumption of complete accessibility and shared memory, one enormous NN can be trained. Coordinates information of all players and the ball will be fed into the network as the input. With complete information, the central processor will compute the optimal team strategy and, as the output, determine the move commands of each player in the team. In addition, the player with the ball possession will be assigned the kick command. Theoretically, the full accessibility and shared memory will help the network to find an optimal strategy. However, in reality, the huge number of nodes may prevent the network from converging to any solution.

<p><i>Input</i></p> <p>$= p \text{ players} \times 2 \text{ teams} \times I + 1 \text{ ball} \times I + \text{possession}$</p> <p>$= 11 \times 2 \times 2 + 1 \times 2 + 1$</p> <p>$= 44$</p>	<p><i>Output</i></p> <p>$= p \text{ players} \times 1 \text{ team} \times I + \text{kick (velocity, direction)}$</p> <p>$= 11 \times 1 \times 2 + 2$</p> <p>$= 24$</p>
--	---

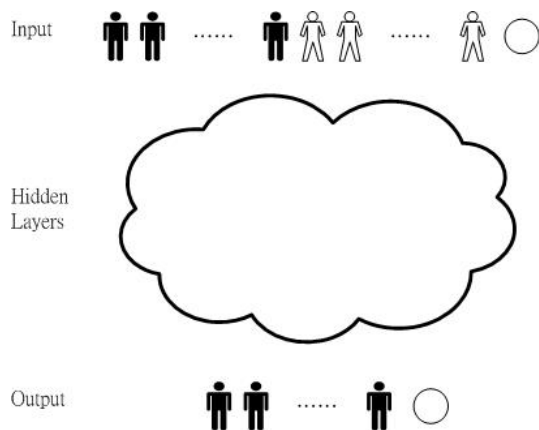


Figure 1 Central processing model

2. *Autonomous, fully omniscient agent model*

Although the assumption of complete accessibility and shared memory is appealing, such assumption may be too strong for any autonomous agent. Hence, the present model is characterized by incomplete accessibility and non-shared memory. Because an agent is autonomous, each agent requires a NN of its own. Despite the large number of networks, the number of output nodes per network is reduced because a player only needs to decide one's own action. Consequently, the size of each network is reduced, which facilitates faster convergence speed. Unfortunately, because all other players can theoretically cluster within a player's view, a huge number of input nodes are still associated with each player.

<p><i>Input</i></p> $= p \text{ players} \times 2 \text{ teams} \times I + 1 \text{ ball} \times I + \text{possession}$ $= 11 \times 2 \times 2 + 1 \times 2 + 1$ $= 44$	<p><i>Output</i></p> $= 1 \text{ players} \times I + \text{kick} (\text{possession}, \text{velocity}, \text{direction})$ $= 1 \times 2 + 3$ $= 5$
--	---

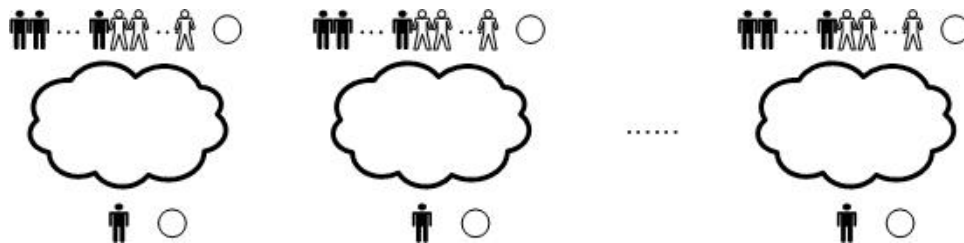


Figure 2 Autonomous, fully omniscient agent model

3. *Autonomous, partially omniscient agent*

To further reduce the size of the NN, this model is designed to enhance the previous model by reducing the number of input nodes. The idea is apparent; players often lock their views on certain objects. For example, strikers usually do not worry about their goalie or defenders in the home court. Rather, knowledge about enemy defenders and goalie may be sufficient to determine the strikers' actions. As a consequence, the number of input nodes can be greatly reduced. The model is hypothesized to have the fastest convergence speed. However, the effectiveness of the computed strategy awaits proof.

<p><i>Input</i></p> $= f \text{ players} \times I + 1 \text{ ball} \times I + \text{possession}$ $= 4 \times 2 + 1 \times 2 + 1$ $= 11$	<p><i>Output</i></p> $= 1 \text{ players} \times I + \text{kick} (\text{possession}, \text{velocity}, \text{direction})$ $= 1 \times 2 + 3$ $= 5$
---	---



Figure 3 Autonomous, partially omniscient agent

Notice, in later experiments, we further simplify the problem from eleven players per team to three players per team. The simplification is justified in order to facilitate

the convergence of the NN learning. Because the number of players is greatly reduced, there is no noticeable difference between the autonomous fully and partially omniscient models. Thus, the two implemented models are referred as the central model and the autonomous model. The number of input and output nodes will be recalculated.

3.3 Data Extraction

As described earlier, the coordinates information of each player and the ball are the only concerns for the data extraction process. There are several ways to obtain the data needed. First of all, the RoboCup simulator program's built-in log recording feature can be used. The RoboCup log is "complete" in a sense that it includes every action taken by every player in every time frame. The advantage of using the well-established RoboCup simulation league platform is that many log files of previous years' competitions are archived and readily available on the Web. However, a parser program may be difficult to write because how the coordinate information can be extracted is unknown.

Another approach is to obtaining the data from the ServerMonitor of the RoboCup simulator program. ServerMonitor is a program that enables the visualization of an

ongoing game play in the soccer server. In order to display a playing game, the ServerMonitor protocol must encapsulate the coordinates for each player and the ball. A program that understands the ServerMonitor protocol can be built to extract the coordinates information. Thus, information of a live game can be extracted. In order to obtain live games, two client teams, which are downloadable from the Web, can be employed routinely to play against each other, while a data absorbing program is attached to the server.

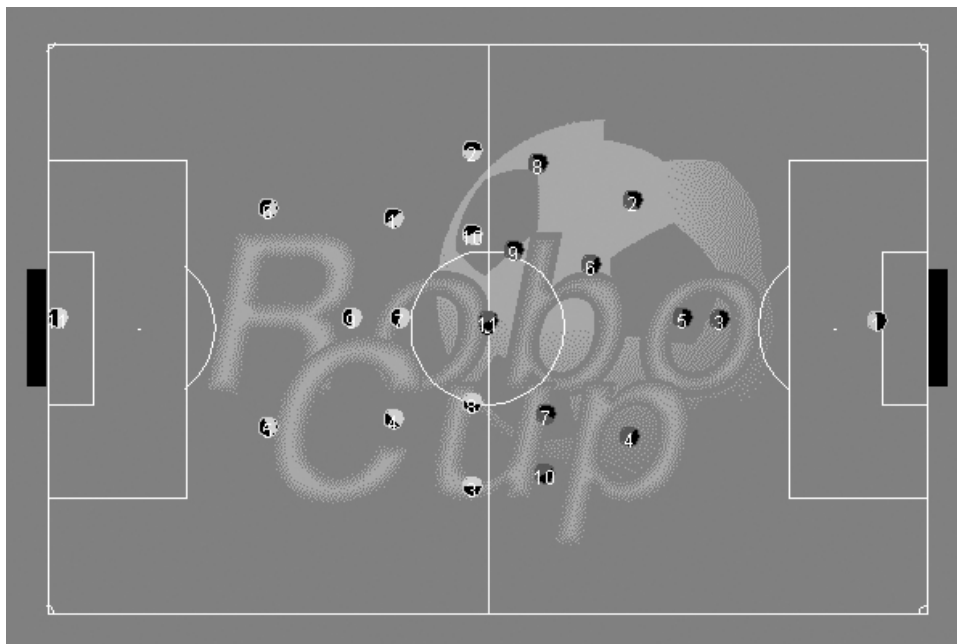


Figure 4 A snap shot of the visualization of RoboCup Simulation game through SoccerMonitor

The third way of extracting data involves the commercial video games of soccer. Unfortunately, this approach is difficult because of the proprietary source codes.

One possibility is to setup a camera shooting onto the screen and use image recognition and processing techniques to extract the information. However, the complexity of image processing restricts its use in present research. Furthermore, this approach is valuable only if the AI engines in the video games are advanced enough to be served as a superior source of input.

3.4 Choice of Programming Language

Learning (training) and recalling are the two components/phases regarding the implementation of an offline learning NN. The structure of the network (such as the connections of the nodes and the weight of each connection) is determined during the offline learning phase. Once the structure of the network is determined, the network structure remains fixed throughout the actual application, that is, the recalling phase.

The Pele Project by Stern et. al. implemented the learning component in Matlab and the recalling component in C++. Although the NN module of Matlab is easy to use, Matlab does not possess enough power to implement online learning. More powerful programming languages must be chosen.

There are many choices of languages for implementation. A table describing

advantages and disadvantages of each language is included as follows.

	Java	C++	C	Matlab
Design Expressiveness	OO	OO	Procedural	Procedural
Integration with other Components	High	Middle	Low	Low
Efficiency	Middle	High	High	High
Portability	High	Middle	High	Low

Table 2 Language Comparison Matrix

The language Java is chosen for two reasons. Firstly, Java is our primary choice because of the programmers' familiarity with the language. Secondly, various open-source programs are already implemented in Java. For example, Ali's Mirosoft simulator program and Atan's RoboCup package are all implemented in Java. The employment of existing packages prevents "reinventing the wheel" and reduces the project overhead.

4. Software Architecture

4.1 Architecture

The overall architecture is comprised of five major components, namely the Soccer Server, Soccer Monitor, Global Positioning System (GPS), player clients and the NN module. The architecture depends on the adopted network model.

The responsibility of the Soccer Server is to simulate the soccer environment and execute the game play. In each time step t , the Soccer Server collects commands from each client and calculates the state of game play in the next time step $t+1$. The state of game play consists of the coordinates of each object, as well as other attributes (i.e. velocity and direction of a ball, etc.) The Soccer Monitor is then used to visualize the state of the game play.

In the central model, one big NN is used to connect the GPS and the clients. At each time step, the central NN obtains the players' coordinates from GPS, calculates and commands the action of each player. In the autonomous model, each player has its own NN component, which is connected to the GPS and Soccer Server directly. The following diagrams illustrate the software architecture for the two models.

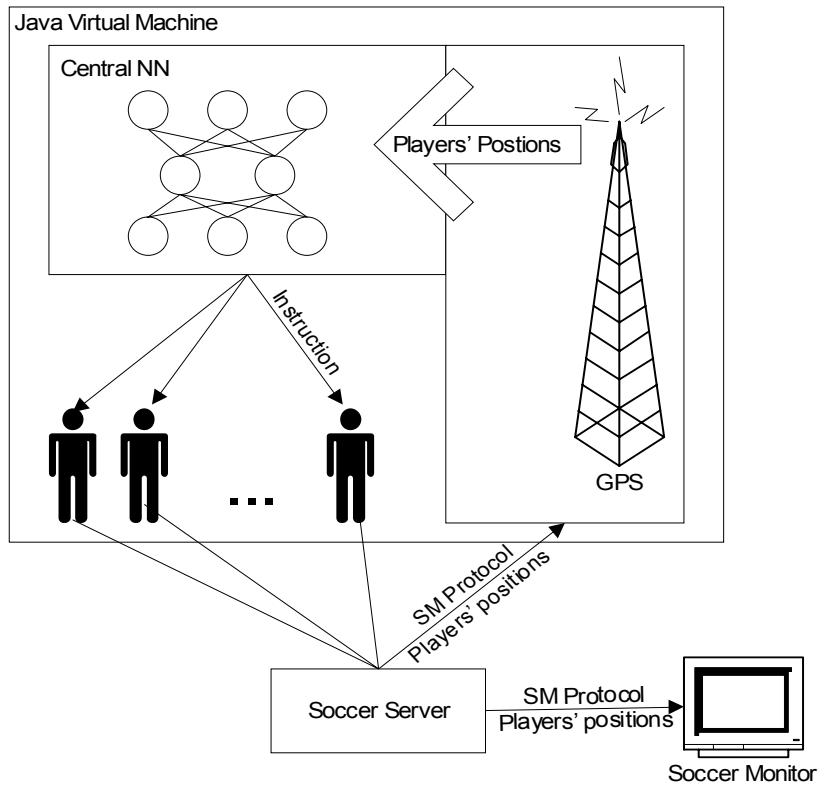


Figure 5 Central NN Model

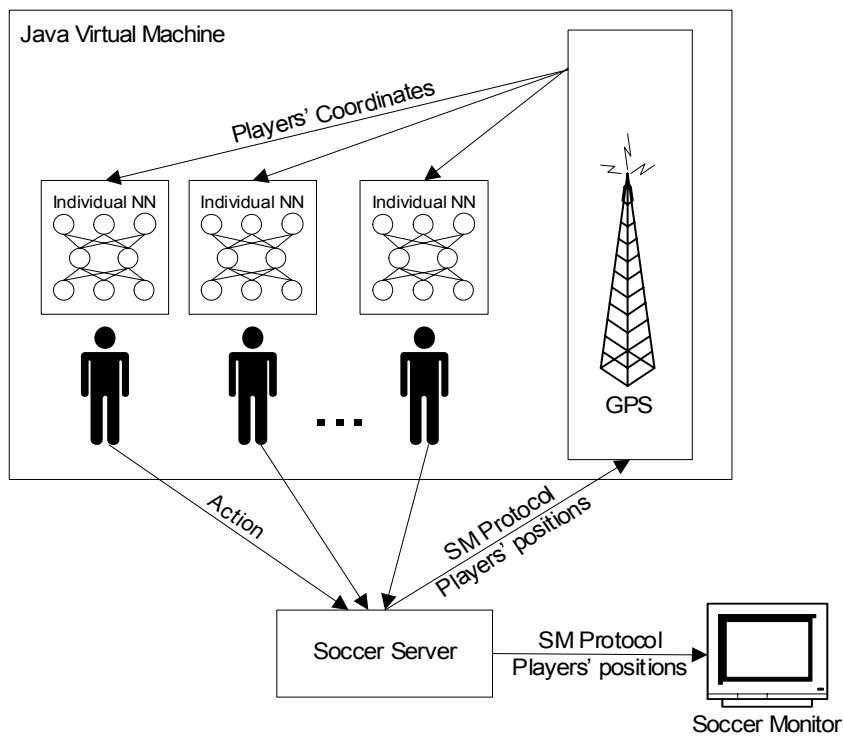


Figure 6 Individual NN Model

4.2 Client - Java

As described in the literature survey, Atan is a convenient Java package for developing soccer clients. The Atan package is comprised of three types of object, namely the player, controller and team. A team is a collection of players and a controller determines the action of each player. Developers only have to extend the controller object to perform the desirable actions.

In the central model, each client controller is implemented with a command queue, which accepts "Command" from the central NN module. In every time step, the client controller dequeues and executes one Command object. The design of Command object is flexible because it can enclose a complex sequence of actions. Complicated command objects, such as dribble and fetch ball, can be made available for future extensions.

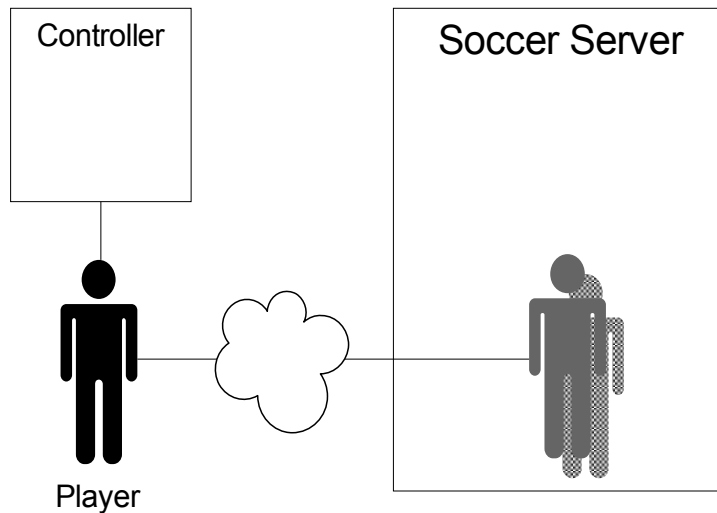


Figure 7 Atan overview

In the autonomous model, the NN module is directly incorporated in the controller.

In every time step, the controller queries the GPS to obtain the current coordinates and submits the information to the NN model to compute its own action.

One problem is sighted in the NN approach to computing actions. Sometimes, the instructed actions will be impossible to complete in one time step. For example, the instruction of moving to opponent's goal when a player is at its own goal is impossible to complete. In this situation, the player will try its best to fulfill the instructed action at the next time step.

4.3 Global Positioning System – C++

As described earlier, the input of the NN module requires the coordinates information of each moving object. The Global Positioning System (GPS) is the module providing this information. The official Soccer Server limits the players' knowledge of the environment to only their immediate surroundings. Thus, a player can only determine its relative position via indirect cues, such as the approximate distance between itself and the goal post. Such information can not be utilized since the input of the NN requires the absolute coordinates, rather than the relative positions.

Fortunately, the Soccer Server and Soccer Monitor are connected via UDP/IP. Because the information enclosed in the protocol is in the form of C struct, we are able to write a C++ program to easily extract the coordinate information.

Upon initiation, the GPS establishes a network connection to the Soccer Server and registers itself as a Soccer Monitor. During the process of the game play, the GPS repeatedly parses the information sent by the Soccer Server and updates its internal attributes at every time step. The detail of the Soccer Monitor protocol is described in the RoboCup manual.

4.4 Interoperability

To make our Java code interoperable with the C++ code, we have adopted the tool provided by the Java package, JNI. JNI allows Java code that runs within a Java Virtual Machine to operate with other applications and libraries written in other languages such as C, C++, and assembly. Thus, an intermediate interface will be built between the Java code and C++ code, as can be seen in the following diagram.

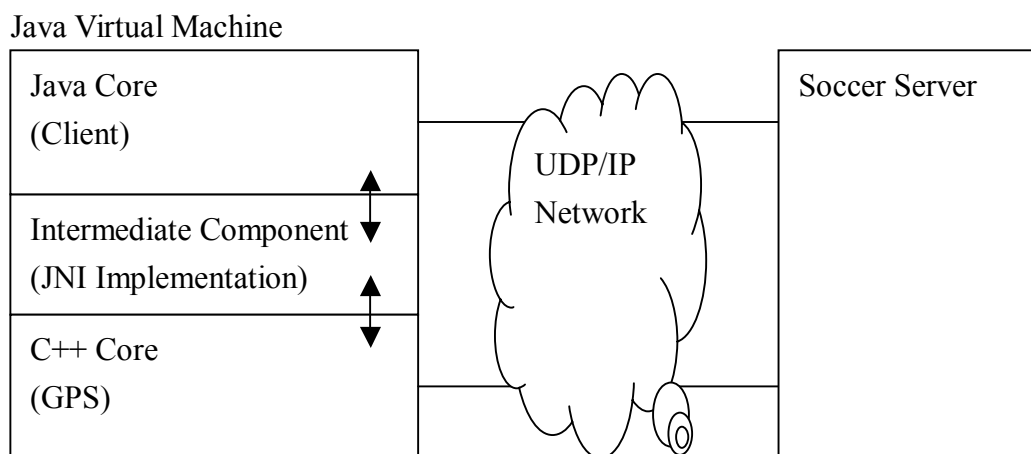


Figure 8 Java Virtual Machine and Soccer Server overview

JNI requires the heterogeneous component to be initially written in Java, declared the methods to be implemented in C++ to be “native methods” and compiled the Java file using the standard Java compiler. Developer then uses the compiled Class file to generate C++ header file for implementation. In the implementation phase, we will

connect the method calls from the intermediate component to GPS and perform type transformation between Java and C++.

5. Implementation

5.1 Neural Network

5.1.1 An Implementation of Neural Network Using Java

For learning purposes, we decide to implement a back propagation network. The structure of the network is implemented with matrices. JAMA package by National Institute of Standards and Technology provides the necessary matrices implementation. The package is extended to provide various matrix manipulations, such as `getRow()`, `sum()`, etc.

In the matrix implementation, each layer corresponds to a $1 \times n$ matrix, where n is the number of nodes in the particular layer. In addition, a matrix defining the connection weights connects every two layers. Suppose the i -th layer contains n_i nodes and the j -th layer contains n_j nodes, then the weight matrix has the dimension of $j \times i$, assuming the input layer is on the top. The network can be constructed with any size. Any two layers can be connected. The weight matrices are initialized with uniformly distributed random numbers between 0 and 1.

The standard delta rule of back propagation network is implemented, according to

Russell & Norvig, 1995.

$$W_{j,i} \leftarrow W_{j,i} + \alpha \cdot a_j \cdot \Delta_i$$

where $\Delta_i = g'(I_i) \times (T_i - O_i)$ on the output layer and

$$\Delta_i = g'(I_i) \times \sum_j W_{j,i} \cdot \Delta_{i-1}$$

on each hidden layer

Because continuous outputs of real number are required for the coordinate information, the sigmoid function is used for the transfer function g .

$$g = \frac{1}{1 + e^{-x}}$$
$$g' = g(1 - g)$$

The correctness of the implementation has been (roughly) verified against a GreaterThan function. A GreaterThan function takes in two inputs x_1, x_2 and outputs $y=1$ if $x_1 > x_2$, otherwise $y=0$. Random training samples are generated. The network consists of 2 inputs, 1 hidden layer of 4 hidden nodes, and 1 output. Moreover, α is set to 0.5 on all layers and across all training samples. With a training set of size 100, 300, and 500, the converged network achieves 93%, 95%, and 100% correctness, respectively.

5.1.2 Exported Code from Neural Ware

While a hands-on implementation of the back propagation network helps us to learn NN, our implementation suffers from the following shortcomings. First of all, the correctness of the implementation is difficult to verify. The correctness against a simple function, such as GreaterThan does not warrant the correctness against complex functions, such as the ball passing team strategy. Secondly, the optimization of codes is limited by our undergraduate level of mathematical knowledge. In particular, efficient matrix manipulations, advanced gradient descent functions calculating the error surface are out of our capabilities.

Commercial NN package provides an easy solution to all of the above problems. As a result of countless working hours, commercial software packages provide access to all of the known networks, which are also thoroughly tested and optimized. In particular, the PAMI lab has recently purchased a Neural Ware software package, which is readily available for our use. Consequently, we decide to experiment with the Neural Ware software package, in addition to our own implementation.

6. Experiment

6.1 Manipulation

Four networks need to be constructed: three autonomous models controlling each player and one central model controlling all players. We decide to explore different network structures for each of the three autonomous clients, as well as different representations of the training data for the central model.

6.1.1 Autonomous Model – Varying Hidden Layers

To determine the NN's optimal structure is an open question. While a network that is too small is incapable of representing the desired function, a network that is too big is subject to the over fitting problem. The only way to find the optimal structure is through experiments. Thus, we have designed three different structures for each of the three autonomous players and compared the performance of the networks against the behaviour of the original clients (TsinghuAeolus).

First of all, we establish the base case structure for the first player. The base case structure has one hidden layer of 11 nodes, where the number 11 is derived from the “log (# records)” rule of thumb. The second player is designed to double the number

of hidden nodes of the base case. Therefore, it has one hidden layer of 22 nodes. The third player is designed to have one more hidden layer than the base case. The additional hidden layer has 8 hidden nodes, where the number 8 is derived from doubling the size of output layer. A summarizing table will be included in later sections.

6.1.2 Central Model – Varying Training Set Representation

For the central model, we experiment with different representations of the training set. By filtering and manipulating the training set, we help the NN to extract and focus learning only on useful information.

First of all, the base case training set is simply the raw representation as specified in the project proposal. Secondly, we randomize the records in the base case. This manipulation is suggested in readings from Neural Networks (Haykin, 1994). Thirdly, we skip every 10 records in the base case. This manipulation is suggested by Ali; his idea is to reduce the resolution of the movement. Instead of training for every little movement, only the big moving trend is used to train the network. Notice, through this manipulation, the size of the training set is also reduced by a factor of 10. That is, we are testing the hypothesis that the original training set is too

big at the same time. Lastly, Ali suggests the conversion of coordinate information in the output to direction and power information. The idea behind such manipulation is that the coordinate information may be too difficult for the NN to compute. Employing the same principle of reducing the resolution, direction and power information are hypothesized to be more sensitive to actions.

The following table summarizes these structures.

Name	ID	Structure
Autonomous Player 1	A3P0	14 11 4
Autonomous Player 2	A3P1	14 22 4
Autonomous Player 3	A3P2	14 11 8 4
Central Raw	C3P0	14 11 8 9
Central Random	C3P1	14 11 8 9
Central Step	C3P2	14 11 8 9
Central Direction_Power	C3P3	14 11 8 9

Table 3 Structures

6.2 Method

6.2.1 Neural Ware Setup

Because our implementation of the NN cannot correctly interpret the network structures of the Neural Ware, all experiments are conducted on the Neural Ware codes. In all of the following experiments, the default setup of the back propagation network is applied, with the exceptions of setting the transfer function to sigmoid

function and deselecting the bipolar input check box.

The “save best” function of the neural ware is used to avoid the problem of over training, which describes the symptom when network is performing well on the training data, but poorly on independent test data. The “save best” function runs in batches of Train/Test cycles and saves the network with the best test result during the run.

6.2.2 Training Set

The training set is obtained from a 3 on 3 log play of TsinghuAeolus vs. Trilearn. There are 3032 records. Undesired duplicate records are removed by piping the log through the “uniq” shell script. Duplicate records represent time out or dead lock during the play, which is not part of any strategy. The training set can further be divided into a “learning set” and “testing set”. On one hand, the learning set is used to train the network and is derived from the top 2/3 of all records. On the other hand, the testing set is used for immediate testing of convergence and is derived from the remaining records.

6.2.3 Team Setup

2 TsinghuAelus clients + NN client vs. 3 Trilearn

Because the NN is learning from past competitions of TsinghuAeolus against Trilearn, we first examine the behaviour of the NN in the exact context. In particular, we replace one client in the TsinghuAeolus with our NN client and have the team play against the original Trilearn client. Unfortunately, the NN client is not too responsive.

2 Fetchball clients + NN client vs. 3 Portugal

One reason to our NN's lack of responsiveness is maybe due to the fact that the two TsinghuAeolus players do not move all the time. In order to maximize the movement of the other clients, fetchball clients are designed to simply chase the ball at all time. Two fetchball clients are teamed up with 1 NN client. We use this setup for all of the experiments.

6.2.4 Measure

The determination of how good our NN is performing is a difficult question. Measures that we proposed earlier, such as game scores and evaluation function cannot be carried out because the client is currently not capable of a competitive game play. Therefore, we can only rely on the visualization of the competition. By

manually examine the behaviour of the NN client, we wish to gain insight into what the network has learned. Furthermore, the standard Root Mean Squared error (RMS) is used to measure how well the NN learning has converged.

6.3 Result

In this section, we will first discuss the behaviour of the original game play and then compare the NN clients to the original clients. The trained weights are included in Appendix C.

6.3.1 Original Game Play

We divide the court into four fields as follows:

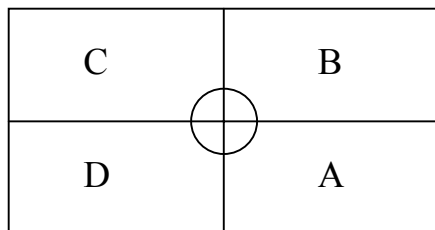


Figure 9 Field description

From the original game play, players tend to stay in certain areas in the court and occasionally perform passing or solo dribbling. From the observation, player 2 and 3 usually cooperate to perform passing and attacking and player 1 usually does the solo dribbling and attacking.

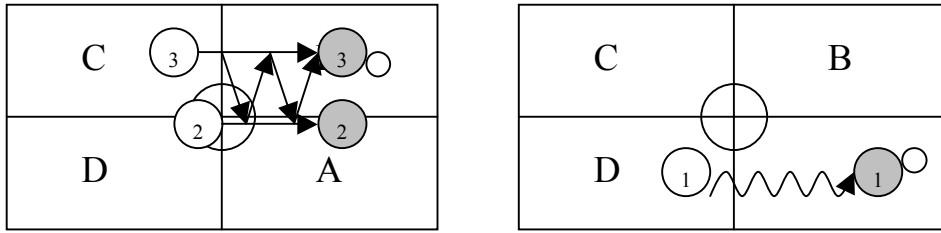


Figure 10 Example of passing and solo attacking

To summarize the original game play, we come up with the following diagram of “player stickiness” to give a visual aid of player distributions during the game play. A player spends most of its time in the darker area and occasionally in the lighter area throughout the game play.

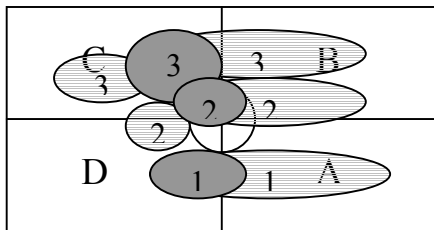


Figure 11 Player stickiness

6.3.2 NN Client

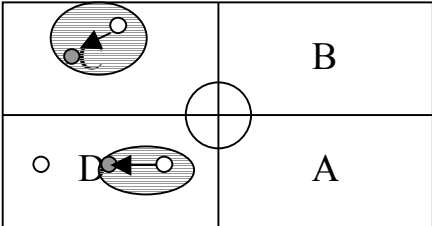
Name	ID	Structure	RMS
Autonomous Player 1	A3P0	14 11 4	0.006931
<p>Comment:</p> <p>Only player 1 is using the NN, the other two players are dummies. When the ball is placed at different corners of the court, player 1 changes its intended position in certain ways.</p>			
<p>The diagrams show a tennis court divided into four quadrants: C (top-left), B (top-right), D (bottom-left), and A (bottom-right). A central circle represents the net. In each diagram, a ball is placed at a corner, and an arrow indicates player 1's intended movement. <ul style="list-style-type: none"> Top-left: Ball at top-right corner, arrow points towards the center. Top-right: Ball at top-left corner, arrow points towards the center. Bottom-left: Ball at bottom-right corner, arrow points towards the center. Bottom-right: Ball at bottom-left corner, arrow points towards the center. </p>			

Name	ID	Structure	RMS
Autonomous Player 2	A3P1	14 22 4	0.102269
<p>Comment:</p> <p>Only player 2 is using the NN, the other two players are dummies. Player 2 only moves within the field B. Player 2 changes its intended position rapidly when the ball is placed at different corners.</p>			
<p>The diagram shows a tennis court with quadrants C, B, D, and A. A shaded oval labeled '2' is positioned in the center of the top half of the court, representing player 2's intended position within field B.</p>			

Autonomous Player 3	A3P2	14 11 8 4	0.004209
<p>Comment:</p> <p>Only player 3 is using the NN, the other two players are dummies. Player 3 only moves within the field C, and reacts to the ball by moving away from it.</p>			

Central Raw	C3P0	14 11 8 9	0.068973
<p>Comment:</p> <p>Player 1 presents a passive character for some reason; it spends most of the time on the line between field C&D. In this model, the players move only within certain invisible boundaries. Player 2 and 3 react slightly to the ball's position by moving toward it. In some circumstances, player 3 moves away from ball's position.</p>			

Central Random	C3P1	14 11 8 9	0.068874
<p>Comment:</p> <p>In this model, the players have similar performance as in the Raw case. Player 2 responds to the position of the ball by moving toward the ball for a few time steps, retracting and moving around randomly afterwards.</p>			

Central Step	C3P2	14 11 8 9	0.068135
<p>Comment:</p> <p>Player 2 and 3 react more responsively to the ball positions. In this model, the players have similar performance as in the Raw case. An interesting snapshot of the test run is that when the ball is placed at their own goal, player 2 and 3 will react rapidly and seem to fetch the ball or save the goal but only up to their invisible boundaries.</p> 			

Central DirectionPower	C3P3	14 11 8 9	0.154538
<p>Comment:</p> <p>In this model, we modify the output from proposed absolute x, y positions for the next time step into appropriate “turn angle” and “dash power” to get to those positions. The result is disastrous because players tend to spin around in circles and do not react according to the positions of the ball.</p>			

6.4 Debug

In order to verify that the NN is using the correct input and output vectors, these vectors are outputted to the Atan logs and are subject to manual verification. In order to verify the correctness of the move command, a radar screen is implemented. In particular, the red dots on the radar display where the NN “wants” to go and the black dots display where the player currently is. Animation of the radar helps verifying the correct implementation of the move command.

7. Conclusions

In summary, although the NN clients are not competitive enough for a real game play; the clients show responses to the placement of the ball and suggest some learning is at work. In particular, small network and small training set appear to have the best performance. The results suggest us to examine the original game log and parse out useless records.

Moreover, another problem of the NN's unresponsiveness is maybe due to the fact that the back propagation network is only reacting to a "snap shot" of the moment. Nayer Wanas suggests two additional algorithms that employ the idea of time duration, namely the times series and hidden Markov model. Unfortunately, those two algorithms were not included in the Neural Ware package and there was not enough time to implement the two algorithms. More research is needed.

8. References

Fröhlich, J. (1999). *Neural Network with Java*.

<http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html>

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing Company.

Russell, S. J., & Norvic, P. (1995). *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice-Hall.

Appendix A – GPS

GlobalMapServer.java

```
package Pami.client;
import java.util.*;
import java.lang.*;

public class GlobalMapServer extends Thread{

    private String host;
    private int port;

    private double[][] player_info;
    private double[] ball_info;

    public static GlobalMapServer instance;

    public GlobalMapServer(String h, int p){

        host = h;
        port = p;

        player_info = new double[22][5];
        ball_info = new double[5];
    }

    public native void init();
    public native void shutdown();
    public native void start_server();

    public native double[] getPlayerInfo(int num);
    public native double[] getBallInfo();

    public native double[] getInput();
```

```

public double[] getOutput() {
    double[] output = new double[9];

    for (int i=0; i<3; i++) {
        double[] p = getPlayerInfo(i);
        output[2*i] = p[0];
        output[2*i+1] = p[1];
    }

    double[] b = getBallInfo();
    output[6] = 0;
    output[7] = 0;
    output[8] = 0;

    return output;
}

public void run(){
    init();
    start_server();
    shutdown();
}

static{
    System.loadLibrary("globalmap");
}

public static void main(String[] argv){

    GlobalMapServer gmap = new GlobalMapServer("localhost", 6000);
    gmap.run();
}
}

```

GlobalMapServer.h

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class Pami_client_GlobalMapServer */

#ifndef _Included_Pami_client_GlobalMapServer
#define _Included_Pami_client_GlobalMapServer
#ifdef __cplusplus
extern "C" {
#endif
/* Inaccessible static: threadInitNumber */
/* Inaccessible static: stopThreadPermission */
#undef Pami_client_GlobalMapServer_MIN_PRIORITY
#define Pami_client_GlobalMapServer_MIN_PRIORITY 1L
#undef Pami_client_GlobalMapServer_NORM_PRIORITY
#define Pami_client_GlobalMapServer_NORM_PRIORITY 5L
#undef Pami_client_GlobalMapServer_MAX_PRIORITY
#define Pami_client_GlobalMapServer_MAX_PRIORITY 10L
/* Inaccessible static: instance */
/*
 * Class:      Pami_client_GlobalMapServer
 * Method:     init
 * Signature:  ()V
 */
JNIEXPORT void JNICALL Java_Pami_client_GlobalMapServer_init
    (JNIEnv *, jobject);

/*
 * Class:      Pami_client_GlobalMapServer
 * Method:     shutdown
 * Signature:  ()V
 */
JNIEXPORT void JNICALL Java_Pami_client_GlobalMapServer_shutdown
    (JNIEnv *, jobject);

/*
```

```

* Class:    Pami_client_GlobalMapServer
* Method:   start_server
* Signature: ()V
*/
JNIEXPORT void JNICALL Java_Pami_client_GlobalMapServer_start_1server
    (JNIEnv *, jobject);

/*
* Class:    Pami_client_GlobalMapServer
* Method:   getPlayerInfo
* Signature: (I)[D
*/
JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getPlayerInfo
    (JNIEnv *, jobject, jint);

/*
* Class:    Pami_client_GlobalMapServer
* Method:   getBallInfo
* Signature: ()[D
*/
JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getBallInfo
    (JNIEnv *, jobject);

/*
* Class:    Pami_client_GlobalMapServer
* Method:   getInput
* Signature: ()[D
*/
JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getInput
    (JNIEnv *, jobject);

#ifdef __cplusplus
}
#endif
#endif

```


GlobalMapServerImp.cc

```
#include <jni.h>
#include <iostream>
#include "GlobalMapServer.h"
#include "GlobalMap.h"

static GlobalMap *gmap_core;

JNIEXPORT void JNICALL Java_Pami_client_GlobalMapServer_init(JNIEnv
*env, jobject obj){

    /* get parameters from java object */

    jclass cls = env->GetObjectClass(obj);

    jfieldID fid = env->GetFieldID(cls, "host", "Ljava/lang/String;");
    jstring jhost = (jstring)env->GetObjectField(obj, fid);

    fid = env->GetFieldID(cls, "port", "I");
    jint port = (jint)env->GetIntField(obj, fid);

    /* init native core */
    const char* host = env->GetStringUTFChars(jhost, 0);
    gmap_core = new GlobalMap(host, port);

}

JNIEXPORT void JNICALL
Java_Pami_client_GlobalMapServer_shutdown(JNIEnv *env, jobject obj){

    gmap_core->close();
    delete gmap_core;

}
```

```

JNIEXPORT void JNICALL
Java_Pami_client_GlobalMapServer_start_1server(JNIEnv *env, jobject
obj){

    gmap_core->start(env, obj);
}

```

```

JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getPlayerInfo(JNIEnv *env, jobject
obj, jint num){

    double pos[5];

    gmap_core->getPlayerInfo(num, pos);

    jdoubleArray arr = env->NewDoubleArray(5);
    env->SetDoubleArrayRegion(arr, 0, 5, pos);

    return arr;
}

```

```

JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getBallInfo(JNIEnv *env, jobject
obj){

    double pos[5];
    gmap_core->getBallInfo(pos);

    jdoubleArray arr = env->NewDoubleArray(5);
    env->SetDoubleArrayRegion(arr, 0, 5, pos);
    return arr;

}

```

```

JNIEXPORT jdoubleArray JNICALL
Java_Pami_client_GlobalMapServer_getInput(JNIEnv *env, jobject obj){

```

```

double input[14];
gmap_core->getInput(input);

jdoubleArray arr = env->NewDoubleArray(14);
env->SetDoubleArrayRegion(arr, 0, 14, input);
return arr;
}

```

GlobalMap.h

```

#ifndef GLOBALMAP_H_
#define GLOBALMAP_H_

#include <string>
#include <stdio.h>
#include <jni.h>
#include <fstream>
#include "udpsocket.h"
#include "types.h"

#define FRAMEVIEW_MODE 10
#define SHOWINFO_SCALE2 65536.0
#define INFO_NUM 5

class GlobalMap{

public:
    GlobalMap(string host, int port);
    ~GlobalMap();

    void start(JNIEnv *, jobject);
    void close();

    void getPlayerInfo(int num, double* pos);

```

```

void getBallInfo(double* pos);

void getInput(double* pos);

private:
int server_msg_type(void * ptr);
void process_positions(void *ptr, JNIEnv *, jobject);
void openPosFiles();
void closePosFiles();
void output_positions();

UDPsocket server;

char buf[10000];
int msg_len;
int msg_id;
ofstream *ofiles[MAX_PLAYER + 1];

double player_info[MAX_PLAYER*2][INFO_NUM];
double ball_info[INFO_NUM];

showinfo_t2 prevState;
};

#endif

```

GlobalMap.cc

```

#include "GlobalMap.h"

GlobalMap::GlobalMap(string host, int port):msg_id(0){

server.init_socket_fd();
server.init_serv_addr(host.c_str(), port);

```

```

char msg[] = "(dispinit version 2)";
server.send_msg(msg, strlen(msg) + 1);

server.recv_msg(buf, msg_len);
}

void GlobalMap::openPosFiles(){

for(int i = 0; i < MAX_PLAYER; i++){

char num[10];
sprintf(num, "%d", i);

string file_name = "pos";
file_name += num;

ofiles[i] = new ofstream(file_name.c_str());
}

ofiles[MAX_PLAYER] = new ofstream("ball");
}

void GlobalMap::closePosFiles(){

for(int i = 0; i < MAX_PLAYER; i++){
ofiles[i]->close();
delete ofiles[i];
}

ofiles[MAX_PLAYER]->close();
delete ofiles[MAX_PLAYER];
}

void GlobalMap::close(){
server.close_socket_fd();
}

GlobalMap::~GlobalMap(){

```

```

server.close_socket_fd();

for(int i = 0; i < MAX_PLAYER; i++){
    char num[10];
    sprintf(num, "%d", i);

    string file_name = "pos";
    file_name += num;

    remove(file_name.c_str());
    cout << "remove " << file_name << endl;
}
}

void GlobalMap::getPlayerInfo(int num, double* pos){

    for(int i = 0; i < INFO_NUM; i++){
        pos[i] = player_info[num][i];
    }
}

void GlobalMap::getBallInfo(double* pos){

    for(int i = 0; i < INFO_NUM; i++)
        pos[i] = ball_info[i];

}

void GlobalMap::getInput(double* pos){

    for(int i = 0; i < 3; i++){

        pos[2*i] = player_info[i][0];
        pos[2*i+1] = player_info[i][1];
    }
}

```

```

}

int j = 0;
for(int i = 3; i < 6; i++, j++){

    pos[2*i] = player_info[MAX_PLAYER + j][0];
    pos[2*i+1] = player_info[MAX_PLAYER + j][1];
}

pos[12] = ball_info[0];
pos[13] = ball_info[1];

}

void GlobalMap::start(JNIEnv *env, jobject obj){
    cout << "GlobalMapServer starting..." << endl;
    int msg_type;

    while(server.recv_msg(buf, msg_len, true )){

        msg_type = server_msg_type(buf);

        if(msg_type == SHOW_MODE){
            process_positions(buf, env, obj);
        }
    }
}

void GlobalMap::output_positions(){

    cout << ">> ";
    for(int i = 0; i < MAX_PLAYER; i++){
        cout << "(" << player_info[i][0] << "," << player_info[i][1] << " )
";
    }
    cout << endl;
}

```

```

}

void GlobalMap::process_positions(void *ptr, JNIEnv *env, jobject obj){

    dispinfo_t2 *disp = (dispinfo_t2*)ptr;
    showinfo_t2 &showinfo = disp->body.show;

    long l_x, l_y, ldelta_x, ldelta_y, l_angle;
    double d_x, d_y, ddelta_x, ddelta_y;
    double d_angle;

    for(int i = 0; i < MAX_PLAYER*2; i++){

        const player_t &info_pos = showinfo.pos[i];

        l_x = ntohl(info_pos.x);
        l_y = ntohl(info_pos.y);

        d_x = (double)l_x / SHOWINFO_SCALE2;
        d_y = (double)l_y / SHOWINFO_SCALE2;

        ldelta_x = ntohl(info_pos.deltax);
        ldelta_y = ntohl(info_pos.deltay);

        ddelta_x = (double)ldelta_x / SHOWINFO_SCALE2;
        ddelta_y = (double)ldelta_y / SHOWINFO_SCALE2;

        l_angle = ntohl(info_pos.body_angle);
        d_angle = - (double)l_angle / SHOWINFO_SCALE2;

        player_info[i][0] = d_x;
        player_info[i][1] = d_y;
        player_info[i][2] = ddelta_x;
        player_info[i][3] = ddelta_y;
        player_info[i][4] = d_angle;

    }
}

```



```

l_x = ntohl(showinfo.ball.x);
l_y = ntohl(showinfo.ball.y);

d_x = l_x / SHOWINFO_SCALE2;
d_y = l_y / SHOWINFO_SCALE2;

ldelta_x = ntohl(showinfo.ball.deltax);
ldelta_y = ntohl(showinfo.ball.deltay);

ddelta_x = ldelta_x / SHOWINFO_SCALE2;
ddelta_y = ldelta_y / SHOWINFO_SCALE2;

ball_info[0] = d_x;
ball_info[1] = d_y;
ball_info[2] = ddelta_x;
ball_info[3] = ddelta_y;

prevState = showinfo;
}

int GlobalMap::server_msg_type(void *ptr){
    if ( *((char*)ptr) == '_' )
        return FRAMEVIEW_MODE;

    dispinfo_t *dispinfo_dum= (dispinfo_t*)ptr;
    dispinfo_t & dispinfo= *dispinfo_dum;

    return (ntohs (dispinfo.mode));
}

```

Appendix B – NN

Recurrent.java

```
package Pami.nn;

import Jama.Matrix;
import java.io.*;
import java.util.*;

public class Recurrent implements NN {

    public static void main(String[] args) {
        Recurrent rc = new Recurrent(args[0]);
        rc.print(3,3);

        /*
        Matrix l0 = new Matrix(1,2);
        Matrix l1 = new Matrix(1,3);
        Matrix l2 = new Matrix(1,2);
        Vector l = new Vector();
        l.add(l0);
        l.add(l1);
        l.add(l2);

        Matrix w0 = new Matrix(new double[][] { {1,2}, {3,4}, {5,6} });
        Matrix w1 = new Matrix(new double[][] { {1,3,5}, {2,4,6} });
        Vector w = new Vector();
        w.add(w0);
        w.add(w1);

        FeedForward ff = new FeedForward(l,w);
        ff.update(new Matrix(new double[][] { {3,2} }));
        ff.print(3,3);

        Recurrent rc = new Recurrent(new int[] {2,3,2});
        rc.connections[0][1] = new Matrix(new double[][] { {1,2}, {3,4},
```

```

{5,6} });
    rc.connections[1][2] = new Matrix(new double[][] { {1,3,5},
{2,4,6} });
    rc.update(new Matrix(new double[][] { {3,2} }));
    rc.update(new Matrix(new double[][] { {3,2} }));
    rc.print(3,3);
    */

    /*
    Recurrent nn = new Recurrent(new int[] { 14,1,17,4 });
    nn.init(args[0]);
    nn.update(new Matrix(new double[][]
{ {1,2,3,4,5,6,7,8,9,10,11,12,13,14} })).print(3,3);
    nn.update(new Matrix(new double[][]
{ {1,3,5,7,9,7,5,3,1,3,5,7,9,7} })).print(3,3);
    */
}

/*****/

protected int height = 0;
protected Matrix[] layers = null;
public Matrix[][] connections = null;

protected Matrix[] next = null;

public Recurrent(int[] structure) {
    init(structure);
}

public Recurrent(String path) {
    try {
        BufferedReader br = new BufferedReader(new
FileReader(path+"struct"));

        int num_layers = Integer.parseInt(br.readLine());
        String struct_layers = br.readLine();

```

```

StringTokenizer st = new StringTokenizer(struct_layers);
int[] structure = new int[num_layers];
for (int i=0; i<structure.length; i++)
    structure[i] = Integer.parseInt(st.nextToken());

    init(structure);
    load(path);
}
catch (IOException e) { System.err.println(e); }
}

public void init(int[] structure) {
    this.height = structure.length;
    this.layers = new Matrix[height];
    this.connections = new Matrix[height][height];

    this.next = new Matrix[height];

    for (int i=0; i<height; i++) {
        this.layers[i] = new Matrix(1, structure[i]);
        this.next[i] = new Matrix(1, structure[i]);
    }

    for (int i=0; i<height; i++)
        for (int j=0; j<height; j++)
            this.connections[i][j] = null;
}

public void load(String path) {
    try {
        File dir = new File(path);
        String[] ls = dir.list();

        for (int i=0; i<ls.length; i++) {
            if (ls[i].indexOf("_") != -1) {

                StringTokenizer st = new StringTokenizer(ls[i], "_");
                int from = Integer.parseInt( st.nextToken() );

```

```

        int to = Integer.parseInt( st.nextToken() );

        connections[from][to] = Matrix.read(new BufferedReader(new
FileReader(path+ls[i])));
    }
}
}
catch (IOException e) { System.out.println(e); }
}

/*****/

public Matrix update(Matrix input) {
    if (input!=null) setInput(input);

    for (int i=0; i<height; i++)
        next[i] = g( in(i) );

    next[0] = next[0].plus(input);
    //next[height-1] = in(height-1);

    System.arraycopy(next, 0, layers, 0, layers.length);

    return getOutput();
}

/*
 * l = 2
 *
 * L0 = 1 x 14
 * W02 = 17 x 14
 * L2 = 1 x 17
 *
 * N = L0 x W02' = 1 x 14 x 14 x 17 = 1 x 17
 */
public Matrix in(int l) {

    Matrix N = new Matrix( getL(l).getRowDimension(),

```

```

getL(l).getColumnDimension() );

    for (int i=0; i<height; i++) {
        if (getW(i,l)!=null) {

            Matrix in = getL(i);
            Matrix wt = getW(i,l);

            N = N.plus( in.times( wt.transpose() ) );
        }
    }

    return N;
}

/*****/

public double g(double x) {
    return 1 / ( 1 + Math.pow(Math.E, -1 * x) );
}

public Matrix g(Matrix A) {

    int m = A.getRowDimension();
    int n = A.getColumnDimension();
    double[][] r = A.getArrayCopy();

    for (int i=0; i<m; i++)
        for (int j=0; j<n; j++)
            r[i][j] = g( r[i][j] );

    return new Matrix(r);
}

/*****/

public void print(int m, int n) {

```

```

System.out.println("Layers: ");

for (int i=0; i<height; i++)
    getL(i).print(m,n);

System.out.println("Weights: ");

for (int i=0; i<height; i++) {
    for (int j=0; j<height; j++) {
        Matrix w = getW(i,j);
        if (w!=null) {
            System.out.println("W("+i+", "+j+") =
"+w.getRowDimension()+"x"+w.getColumnDimension());
            getW(i,j).print(m,n);
        }
    }
}

/*****/

public void setInput(Matrix m) {
    setL(0, m);
}

public Matrix getInput() {
    return getL(0);
}

public Matrix getOutput() {
    return getL(height-1);
}

/*****/

public Matrix getL(int i) {
    return layers[i];
}

```

```

public void setL(int i, Matrix m) {
    layers[i] = m;
}

public Matrix getW(int i, int j) {
    return connections[i][j];
}

public void setW(int i, int j, Matrix m) {
    connections[i][j] = m;
}
}

```

Backpropagation.java

```

package Pami.nn;

import Jama.Matrix;
import java.util.Vector;

public class Backpropagation extends FeedForward {

    public Backpropagation(Vector layers, Vector connections) {
        super(layers, connections);
    }

    public Backpropagation(int[] structure) {
        super(structure);
    }

    /**
     *
     */

    public void backprop_update(Matrix examples, double alpha) {

        //while (NETWORK_NOT_CONVERGED) {
        int correct = 0;
        while (correct < examples.getRowDimension()) {

```



```

int m = getInput().getColumnDimension();
int n = getOutput().getColumnDimension();
int t = examples.getRowDimension();

for (int x=0; x<t; x++) {

    Matrix e = examples.getMatrix(x,x, 0,m+n-1);
    Matrix I = e.getMatrix(0,0, 0,m-1);
    Matrix T = e.getMatrix(0,0, m,m+n-1);
    Matrix O = update(I);

    //System.out.println("Training Example " + x);
    //e.print(3,3);
    //print(3,3);

    /*****/

    int i = height - 1;
    int j = i - 1;

    Matrix A = getL(j);
    Matrix W = getW(j);
    Matrix D = T.minus(O);
    D = gP( in(i) ).arrayTimes( D );

    setW(j, D.transpose().times(A).times(alpha).plus(W));

    /*****/

    for (i=i-1,j=j-1; j>=0; i--,j--) {
        A = getL(j);
        W = getW(j);
        D = gP( in(i) ).arrayTimes( D.times(getW(i)) );

        setW(j, D.transpose().times(A).times(alpha).plus(W));
    }
}

```

```

/*****/

O = update(null);
for (int c=0; c<n; c++) O.set(0,c, Math.round(O.get(0,c)));
//O.print(3,3);
//T.print(3,3);

if (T.equals(O)) correct++;
else { System.out.println(correct); correct = 0; }
}

}

System.out.println( "Training Complete" );
}

/*****/

public double gP(double x) {
    return g(1 - g(x));
}

public Matrix gP(Matrix A) {

    int m = A.getRowDimension();
    int n = A.getColumnDimension();
    double[][] r = A.getArrayCopy();

    for (int i=0; i<m; i++)
        for (int j=0; j<n; j++)
            r[i][j] = gP( r[i][j] );

    return new Matrix(r);
}
}

```

Appendix C – NN Weights

C.1 A3P0

Input to Hidden1

0.222	-0.642	-0.120	-0.154	-0.314	-0.152	0.245	-0.093	0.109	-0.364	0.183	-0.126	-0.064	0.021
0.143	0.022	-0.193	0.081	0.057	0.127	0.154	-0.235	0.083	-0.055	0.006	0.237	-0.118	-0.093
-0.683	0.065	0.068	0.014	0.105	0.183	0.043	-0.004	-0.119	0.017	-0.078	0.122	0.255	0.004
0.028	-0.378	-0.260	-0.219	0.114	0.053	0.199	-0.223	-0.101	-0.265	-0.222	0.039	0.166	-0.023
0.030	-0.383	-0.244	-0.156	-0.040	-0.159	0.069	0.138	-0.187	-0.262	-0.336	-0.059	-0.228	0.073
0.169	0.360	0.097	-0.039	-0.239	0.172	-0.224	0.068	0.186	-0.142	0.192	0.233	-0.162	0.182
-0.612	0.275	-0.129	-0.090	-0.123	-0.199	0.137	-0.157	0.204	-0.005	0.097	-0.217	0.106	-0.329
-0.308	0.539	-0.141	-0.188	-0.071	-0.068	-0.230	0.294	-0.046	0.282	-0.283	-0.131	0.247	0.249
0.019	-0.240	0.130	0.084	-0.149	-0.160	0.093	0.226	0.090	-0.201	0.256	-0.027	-0.005	-0.165
-0.130	-0.224	-0.096	-0.222	-0.141	0.088	-0.204	-0.024	-0.025	0.044	-0.312	-0.191	0.088	0.226
-0.155	0.180	-0.073	0.127	-0.018	-0.110	-0.096	0.021	-0.154	-0.030	-0.038	0.098	-0.194	0.205

Input to Output

2.162	-0.075	0.001	-0.109	0.087	0.001	-0.286	-0.030	0.001	0.041	0.095	0.011	0.024	0.071
-0.092	2.038	-0.018	-0.037	0.007	-0.044	-0.267	-0.045	-0.006	-0.137	0.013	-0.048	0.029	0.037
0.000	-0.014	-0.028	-0.101	-0.068	0.008	-0.266	-0.006	0.010	-0.091	-0.091	-0.062	0.004	0.008
-0.100	-0.113	0.007	-0.062	-0.024	0.005	-0.112	-0.071	0.047	-0.080	0.034	0.003	-0.003	-0.009

Bias to Hidden1

0.144
-0.033
-0.125
0.106
-0.032
0.115
-0.032
-0.388
-0.061
0.201
-0.282

Bias to Output

-0.288
-0.199

-0.299

-0.403

Hidden1 to Output

0.182	0.019	-0.545	-0.017	-0.162	0.024	-0.458	-0.264	-0.039	-0.131	-0.077
-0.561	0.022	0.117	-0.510	-0.466	0.168	0.036	0.286	-0.203	-0.174	0.076
-0.381	0.196	-0.297	-0.311	-0.361	-0.205	-0.223	0.080	0.135	-0.249	-0.022
-0.199	0.065	-0.236	-0.434	-0.063	-0.289	-0.333	0.219	-0.188	-0.162	0.051

C.2 A3P1

Input to Hidden1

-0.448	-0.078	-0.898	-0.035	-0.619	0.007	0.019	0.026	0.389	-0.420	0.582	-0.342	-0.114	-0.017
-0.312	0.052	-1.077	0.093	-0.430	0.095	-0.028	-0.242	0.415	-0.304	0.474	-0.120	-0.034	-0.425
-0.133	-0.277	-0.325	-0.394	0.022	0.103	-0.111	-0.104	-0.187	-0.038	-0.160	-0.034	0.148	0.083
-0.422	0.184	-0.967	-0.089	-0.256	0.174	-0.121	-0.187	0.224	-0.337	0.284	-0.249	0.208	-0.043
0.336	-0.117	-0.162	-0.520	0.142	-0.129	-0.121	0.025	-0.414	-0.157	-0.518	-0.058	-0.354	0.231
-0.146	0.119	-0.236	0.190	-0.705	0.135	-0.491	-0.052	0.344	-0.328	0.606	-0.037	-0.134	-0.003
-0.703	0.440	-1.305	0.002	-0.734	-0.065	-0.061	-0.064	0.797	-0.297	0.918	-0.624	0.197	-0.327
0.141	0.195	-0.174	-0.694	-0.039	-0.190	-0.218	0.233	-0.168	0.261	-0.474	-0.180	0.173	0.300
-0.052	-0.153	0.238	-0.209	-0.280	-0.305	-0.140	0.155	0.102	-0.220	0.210	-0.123	-0.053	-0.242
0.386	-0.397	0.378	-0.536	0.090	-0.038	-0.340	-0.148	-0.410	0.196	-0.838	-0.044	-0.134	0.381
0.148	-0.111	0.023	-0.155	0.124	-0.205	-0.133	-0.045	-0.443	0.034	-0.513	0.119	-0.402	0.249
-0.176	-0.051	-0.425	-0.233	0.168	0.104	-0.184	0.234	-0.040	0.252	-0.728	0.177	-0.003	-0.156
-0.113	-0.224	-0.504	-0.159	-0.153	-0.158	-0.243	0.080	0.029	-0.061	0.170	-0.111	-0.064	-0.242
-0.393	-0.229	-0.215	0.162	-0.231	-0.284	-0.343	-0.100	0.265	-0.203	0.096	0.027	0.208	-0.346
0.004	0.089	-0.189	-0.129	-0.113	0.098	-0.351	0.089	-0.016	-0.175	-0.148	-0.293	-0.264	-0.393
0.264	-0.443	-0.251	0.051	0.227	-0.181	-0.391	0.187	-0.558	-0.156	-0.679	-0.079	-0.343	0.254
-0.287	-0.102	-0.677	-0.159	-0.131	-0.040	-0.045	-0.053	-0.049	0.038	-0.267	-0.324	-0.328	0.137
-0.661	-0.027	-0.654	0.142	-0.535	-0.331	-0.102	-0.215	0.147	0.066	0.493	-0.364	0.296	-0.482
-0.309	0.057	-0.312	0.257	-0.233	-0.288	-0.300	-0.133	0.133	0.048	-0.038	-0.017	-0.280	-0.202
0.358	-0.122	0.777	-0.553	0.310	-0.148	-0.260	-0.313	-0.444	0.281	-0.730	-0.013	-0.279	0.334
0.051	-0.237	0.440	-0.255	0.034	-0.202	0.100	0.123	-0.579	0.301	-0.635	0.098	-0.239	-0.112
0.233	-0.271	0.183	-0.297	0.144	-0.247	0.038	-0.268	-0.359	0.296	-0.521	-0.154	-0.282	0.138

Input to Output

-0.090	0.056	1.972	0.032	-0.023	-0.024	-0.299	0.014	0.044	-0.043	0.001	-0.111	-0.045	-0.033
-0.084	-0.075	0.006	1.809	-0.023	-0.068	-0.053	-0.052	-0.136	0.057	0.015	0.005	-0.081	0.188
-0.151	0.001	-0.152	-0.068	-0.126	-0.079	-0.276	0.131	0.032	-0.063	-0.110	-0.117	0.147	0.007

1.399 -0.981 2.177 -0.597 1.388 -0.443 -0.263 0.214 -1.118 0.758 -2.525 0.883 -0.766 0.839

Bias to Hidden1

-0.171
-0.318
-0.300
-0.333
-0.207
-0.260
-0.400
-0.349
-0.338
0.122
-0.272
-0.155
0.024
-0.271
-0.385
0.040
0.044
-0.334
-0.331
-0.009
-0.068
-0.034

Bias to Output

-0.036
-0.106
-0.145
0.051

Hidden1 to Output

-0.289 -0.222 -0.228 -0.230 -0.138 0.188 -0.351 -0.124 0.242 0.068 -0.159 -0.382 -0.279 0.043 -0.122 -0.292 -0.376 0.033 -0.096 0.266 0.118 -0.137
-0.097 -0.083 -0.269 -0.057 -0.280 0.132 -0.191 -0.457 -0.142 -0.186 -0.129 -0.186 -0.113 0.128 -0.140 0.189 -0.009 -0.111 0.148 -0.106 -0.160 -0.101
-0.262 -0.050 -0.145 -0.031 0.086 -0.335 -0.206 -0.179 -0.187 -0.105 -0.274 -0.135 -0.173 -0.273 0.094 0.074 -0.088 -0.018 -0.049 0.115 -0.233 -0.090
-0.442 -0.463 0.058 -0.385 0.302 -0.311 -0.805 0.150 0.023 0.510 0.331 0.154 -0.095 -0.131 -0.030 0.321 -0.013 -0.442 -0.064 0.551 0.448 0.370

C.3 A3P2

Input to Hidden1

-0.062	-0.151	-0.015	-0.104	-0.726	0.079	0.183	-0.087	0.125	-0.310	0.232	-0.093	-0.116	0.043
0.143	-0.148	-0.083	0.152	-0.700	0.425	0.191	-0.156	0.114	-0.091	-0.045	0.297	-0.107	-0.132
-0.004	-0.186	0.020	0.014	-0.248	-0.459	0.064	-0.036	-0.197	-0.033	-0.180	0.047	0.181	0.152
0.026	0.036	-0.263	-0.215	0.307	-0.024	0.068	-0.122	-0.088	-0.098	-0.138	0.065	0.200	0.026
0.296	0.050	-0.154	-0.048	-0.437	-0.513	0.083	0.107	-0.250	-0.256	-0.287	-0.085	-0.276	0.283
0.241	0.026	0.170	-0.035	-0.648	0.128	-0.223	0.143	0.229	-0.098	0.180	0.239	-0.075	0.139
-0.086	0.135	-0.063	-0.164	-0.328	-0.432	0.076	-0.095	0.172	0.051	0.142	-0.275	0.100	-0.285
0.144	0.329	-0.216	-0.208	0.242	-0.630	-0.093	0.258	-0.129	0.223	-0.303	-0.197	0.231	0.250
0.113	-0.120	0.213	0.097	-0.735	-0.486	0.120	0.300	0.089	-0.199	0.151	-0.031	0.010	-0.118
0.086	-0.087	-0.157	-0.303	0.323	0.042	-0.244	-0.147	-0.018	0.040	-0.209	-0.239	0.073	0.224
0.050	0.198	-0.137	0.138	0.207	-0.589	0.021	-0.064	-0.198	-0.110	-0.085	0.057	-0.213	0.230

Input to Hidden2

-0.238	0.201	0.054	0.129	-0.086	0.432	0.002	0.227	0.196	0.241	-0.295	0.278	0.243	-0.184
0.110	0.080	-0.157	-0.048	-0.140	-0.138	0.028	-0.040	0.081	-0.290	-0.214	-0.015	0.272	-0.026
0.234	-0.022	-0.061	-0.244	-0.006	-0.343	-0.208	0.181	0.125	-0.233	-0.290	-0.020	0.150	-0.189
-0.251	0.003	-0.013	0.163	-0.375	0.051	0.116	-0.229	-0.246	-0.270	0.079	-0.036	-0.299	-0.084
-0.190	-0.202	0.158	-0.087	0.145	-0.057	-0.293	-0.220	-0.144	0.139	0.122	-0.044	0.060	-0.196
-0.098	-0.236	0.032	0.116	0.090	0.099	-0.110	0.058	0.205	-0.141	0.031	0.257	0.281	-0.310
0.282	-0.283	0.001	0.294	0.172	-0.554	0.071	0.102	0.200	-0.188	-0.208	-0.160	0.243	-0.039
0.079	0.106	0.253	-0.165	0.312	0.409	0.198	-0.125	-0.247	-0.241	0.174	-0.082	-0.078	-0.155

Input to Output

0.075	0.006	0.008	0.086	1.908	-0.016	0.189	0.053	0.054	-0.094	0.033	0.060	-0.081	-0.048
0.079	0.001	-0.066	-0.066	0.088	2.010	-0.267	0.036	-0.091	-0.011	-0.073	-0.121	0.030	0.158
-0.019	-0.058	-0.006	0.047	-0.024	0.031	-0.210	0.009	0.027	-0.012	0.031	0.035	0.020	-0.047
0.100	0.083	-0.058	-0.002	0.077	-0.218	0.153	-0.018	-0.112	-0.062	-0.122	-0.092	-0.029	0.103

Bias to Hidden1

0.056
-0.010
-0.102
-0.042
-0.021
0.106
-0.100
-0.213

-0.038
0.159
-0.135
Bias to Hidden2
0.010
0.130
-0.218
-0.142
-0.280
0.094
-0.106
-0.146
Bias to Hidden3
-0.322
-0.231
-0.063
0.264
Hidden1 to Hidden2
0.309 0.156 -0.209 0.138 -0.101 0.111 -0.143 -0.129 0.127 0.000 0.030
-0.166 0.217 0.270 -0.247 -0.222 0.233 0.110 0.262 0.022 0.087 0.166
-0.314 0.224 -0.290 -0.242 -0.282 -0.178 -0.178 0.147 0.220 -0.205 0.008
0.003 0.252 -0.097 -0.311 0.101 -0.128 -0.167 0.282 -0.035 -0.052 0.198
-0.216 0.131 0.008 0.156 0.108 -0.289 0.114 0.052 -0.078 -0.206 0.012
0.198 -0.143 0.016 -0.059 -0.250 0.117 0.040 -0.056 0.079 -0.040 0.004
0.213 -0.194 0.313 -0.047 -0.210 -0.286 -0.251 -0.115 -0.005 -0.031 0.144
0.235 0.007 0.243 0.225 -0.184 -0.112 -0.330 0.128 -0.320 -0.108 0.209
Hidden1 to Output
-0.414 -0.521 -0.327 0.087 -0.366 -0.401 -0.226 0.199 -0.499 0.350 0.151
0.090 0.362 -0.451 -0.128 -0.414 -0.033 -0.230 -0.608 -0.276 -0.146 -0.518
0.174 -0.205 0.082 -0.194 0.157 -0.100 -0.059 0.102 0.105 0.285 0.282
0.053 0.262 -0.223 -0.005 -0.354 0.205 0.218 -0.212 -0.031 -0.125 -0.389
Hidden2 to Output
-0.276 -0.005 0.104 -0.427 0.045 0.132 -0.051 0.387
0.372 0.093 -0.151 0.150 0.005 0.284 -0.479 0.183
0.169 -0.041 -0.094 -0.239 0.161 0.003 -0.255 0.151
0.248 -0.305 -0.066 -0.196 0.004 0.343 -0.151 -0.178

C.4 C3P0

Input to Hidden1

-0.970	0.209	-1.232	0.407	-1.287	-0.349	-0.021	0.087	0.813	-0.755	1.181	-0.519	-0.081	-0.205
-0.794	0.248	-1.357	0.787	-0.722	0.374	0.137	-0.083	0.580	-0.290	0.624	-0.113	0.070	-0.372
-0.445	-0.375	-0.718	-0.200	-0.262	-0.226	-0.054	-0.140	0.023	-0.283	0.266	-0.203	0.264	-0.282
0.205	-0.478	-0.063	-0.648	0.209	0.280	-0.091	-0.151	-0.661	0.140	-1.118	0.214	-0.234	0.183
0.243	-0.237	0.044	-0.706	0.690	-0.144	0.072	0.032	-0.610	-0.133	-0.755	0.056	-0.422	0.257
0.385	-0.329	0.627	-0.270	-0.495	0.003	-0.450	0.114	-0.394	0.156	-0.808	0.435	-0.565	0.279
-0.915	0.517	-1.628	0.584	-0.441	-0.013	0.006	-0.062	0.992	-0.491	1.235	-0.714	0.223	-0.496
0.244	0.136	-0.541	-0.630	-0.346	-0.660	-0.171	0.139	-0.080	0.115	-0.126	-0.373	0.229	0.009
0.093	-0.398	0.779	-0.206	0.066	-0.552	0.065	0.273	-0.299	0.014	-0.546	0.215	-0.270	-0.092
0.173	-0.265	-0.447	-0.432	-0.402	-0.029	-0.390	-0.222	0.097	-0.081	0.125	-0.411	0.151	-0.004
-0.145	-0.054	-0.502	-0.179	-0.408	-0.082	-0.185	-0.246	-0.016	-0.272	0.367	-0.256	-0.124	-0.159

Input to Hidden2

-0.680	0.385	-0.153	0.298	-0.095	0.194	-0.131	0.202	0.252	0.130	-0.170	0.119	0.241	-0.284
0.243	0.060	-0.023	-0.404	-0.295	-0.592	0.037	0.019	-0.100	-0.153	-0.510	0.050	0.140	0.008
0.122	-0.052	-0.393	-0.260	-0.413	-0.168	-0.277	0.113	0.191	-0.305	-0.114	-0.155	0.184	-0.262
-0.603	0.005	-0.240	-0.043	-0.063	-0.109	0.064	-0.205	-0.362	-0.229	-0.197	-0.028	-0.422	-0.090
-0.089	-0.311	0.529	-0.254	0.336	-0.098	-0.374	-0.204	-0.372	0.235	-0.277	0.117	-0.106	-0.100
-0.279	-0.144	-0.273	0.400	-0.458	-0.089	-0.199	0.047	0.298	-0.224	0.223	0.070	0.290	-0.378
0.518	-0.607	0.237	0.028	0.176	-0.393	0.033	0.075	-0.038	-0.048	-0.688	-0.007	0.030	0.000
0.184	0.077	0.218	-0.357	0.155	0.136	0.162	-0.177	-0.299	-0.260	0.148	-0.138	-0.088	-0.235

Input to Output

1.758	-0.429	-0.114	0.059	0.079	-0.026	0.144	-0.050	0.091	-0.044	0.255	0.043	-0.034	0.112
-0.205	1.774	0.188	-0.172	0.223	-0.051	-0.331	0.068	-0.192	-0.028	-0.079	-0.085	0.088	0.140
-0.143	0.047	2.061	0.150	-0.043	-0.077	-0.359	0.013	0.025	-0.018	0.097	0.022	-0.005	-0.078
0.212	-0.104	0.092	1.866	0.044	-0.231	-0.023	-0.118	-0.227	-0.071	-0.210	-0.060	-0.145	0.156
0.093	0.036	-0.066	0.018	1.889	0.078	-0.431	0.110	0.126	-0.117	-0.048	0.031	0.034	-0.181
0.141	0.109	-0.032	-0.182	-0.046	1.975	-0.322	0.002	-0.050	-0.029	-0.048	-0.064	-0.049	0.122
0.105	-0.195	0.245	0.464	0.108	-0.583	0.561	-0.060	0.024	0.186	-0.294	-0.036	0.165	0.163
-0.059	0.011	-0.031	-0.070	-0.194	0.016	-0.425	0.120	0.023	-0.084	-0.260	-0.067	0.230	0.067
0.988	-0.957	1.875	-0.418	1.152	-0.346	-0.306	0.274	-1.022	0.801	-2.139	0.684	-0.689	0.658

Bias to Hidden1

-0.356
-0.210
-0.312

-0.133
0.029
-0.034
-0.348
-0.334
-0.022
-0.049
-0.446

Bias to Hidden2

-0.172
0.174
-0.331
-0.188
-0.311
-0.051
-0.098
-0.184

Bias to Hidden3

-0.390
-0.274
-0.229
0.073
-0.038
-0.209
-0.015
-0.451
0.296

Hidden1 to Hidden2

0.306	0.087	-0.249	0.036	-0.226	-0.016	-0.129	-0.171	0.029	-0.087	0.017
-0.218	0.123	0.269	-0.192	-0.123	0.316	0.031	0.360	0.113	0.125	0.179
-0.257	0.257	-0.296	-0.290	-0.353	-0.213	-0.155	0.115	0.164	-0.217	-0.014
-0.052	0.199	-0.099	-0.265	0.115	-0.170	-0.200	0.311	-0.082	-0.056	0.201
-0.369	0.016	-0.039	0.198	0.161	-0.265	-0.047	0.019	-0.076	-0.231	-0.012
0.267	-0.126	-0.009	-0.174	-0.355	0.053	0.084	-0.073	0.028	-0.095	0.000
0.076	-0.289	0.277	0.032	-0.148	-0.215	-0.452	-0.142	0.023	-0.010	0.089
0.199	-0.055	0.236	0.231	-0.133	-0.084	-0.368	0.151	-0.278	-0.125	0.196

Hidden1 to Output

-0.311	-0.392	-0.329	-0.023	-0.235	-0.168	-0.217	0.192	-0.323	0.363	0.105
--------	--------	--------	--------	--------	--------	--------	-------	--------	-------	-------

0.104	0.179	-0.438	-0.239	-0.302	-0.133	0.014	-0.304	-0.210	-0.227	-0.305
-0.040	-0.507	-0.087	-0.361	-0.025	-0.088	-0.330	-0.072	0.148	0.108	0.172
-0.049	0.285	-0.307	-0.222	-0.563	0.088	0.104	-0.430	-0.111	-0.304	-0.405
-0.339	-0.298	-0.009	-0.220	0.377	-0.411	0.295	-0.127	0.026	-0.104	-0.123
-0.476	0.294	-0.171	0.226	0.019	-0.033	0.030	-0.480	-0.279	-0.094	0.069
-0.142	0.286	0.104	-0.066	0.287	-0.086	0.109	0.141	0.388	0.134	0.077
-0.123	-0.323	0.241	-0.218	0.153	-0.365	0.062	-0.154	0.129	0.277	-0.287
-0.952	-0.750	-0.374	0.678	0.508	0.677	-1.057	-0.047	0.515	-0.080	-0.254

Hidden2 to Output

-0.431	0.087	0.158	-0.516	-0.073	0.040	-0.031	0.287
0.318	0.206	-0.051	0.137	-0.005	0.187	-0.459	0.031
0.092	-0.209	-0.173	-0.387	0.194	-0.077	-0.280	0.149
0.212	-0.526	-0.174	-0.242	-0.091	0.373	-0.116	-0.348
-0.114	-0.380	-0.349	-0.018	0.123	-0.512	-0.231	0.181
0.094	-0.574	0.170	-0.030	0.102	-0.020	-0.265	-0.083
0.159	0.405	-0.064	0.150	0.030	0.035	0.105	0.414
-0.083	-0.262	-0.070	0.110	0.240	-0.397	-0.113	0.082
-0.252	0.336	-0.227	0.144	0.601	-0.363	0.567	0.136

C.5 C3P1

Input to Hidden1

-0.881	0.232	-1.267	0.336	-1.190	-0.489	-0.003	0.007	0.692	-0.666	1.155	-0.531	-0.020	-0.177
-0.907	0.301	-1.434	0.782	-0.717	0.360	0.089	-0.143	0.622	-0.277	0.656	-0.178	0.082	-0.263
-0.479	-0.393	-0.728	-0.204	-0.274	-0.207	-0.110	-0.096	0.027	-0.275	0.203	-0.150	0.255	-0.299
0.271	-0.504	-0.069	-0.674	0.247	0.341	-0.067	-0.154	-0.641	0.146	-1.094	0.187	-0.240	0.170
0.251	-0.236	0.036	-0.752	0.688	-0.110	0.060	0.065	-0.512	-0.190	-0.712	0.117	-0.437	0.269
0.369	-0.361	0.619	-0.199	-0.503	-0.045	-0.410	0.060	-0.357	0.174	-0.740	0.394	-0.542	0.305
-0.958	0.552	-1.754	0.581	-0.390	0.010	-0.192	-0.076	1.011	-0.404	1.239	-0.827	0.306	-0.317
0.196	0.149	-0.589	-0.597	-0.359	-0.607	-0.168	0.142	-0.025	0.111	-0.109	-0.366	0.201	0.083
0.099	-0.411	0.763	-0.273	0.074	-0.524	0.002	0.283	-0.261	-0.037	-0.509	0.201	-0.347	-0.067
0.087	-0.266	-0.441	-0.437	-0.398	-0.055	-0.406	-0.232	0.091	-0.010	0.000	-0.432	0.104	0.034
-0.129	-0.074	-0.542	-0.176	-0.413	-0.066	-0.207	-0.231	-0.010	-0.216	0.268	-0.257	-0.128	-0.159

Input to Hidden2

-0.678	0.374	-0.166	0.325	-0.125	0.179	-0.151	0.188	0.250	0.133	-0.153	0.093	0.246	-0.257
0.214	0.077	-0.087	-0.453	-0.348	-0.600	0.019	0.029	-0.072	-0.159	-0.463	0.041	0.136	0.046
0.122	-0.059	-0.386	-0.248	-0.407	-0.191	-0.298	0.100	0.186	-0.300	-0.131	-0.158	0.184	-0.276

-0.580	-0.032	-0.219	-0.079	-0.059	-0.117	0.049	-0.200	-0.371	-0.231	-0.227	-0.008	-0.439	-0.071
-0.074	-0.302	0.534	-0.305	0.343	-0.111	-0.356	-0.199	-0.367	0.232	-0.315	0.105	-0.131	-0.089
-0.282	-0.147	-0.225	0.414	-0.478	-0.111	-0.228	0.038	0.268	-0.202	0.224	0.050	0.296	-0.372
0.534	-0.662	0.210	-0.027	0.197	-0.389	0.033	0.101	-0.017	-0.033	-0.677	0.005	0.014	0.006
0.194	0.054	0.230	-0.411	0.184	0.127	0.135	-0.181	-0.314	-0.261	0.094	-0.129	-0.127	-0.215

Input to Output

1.762	-0.414	-0.130	0.046	0.068	-0.031	0.141	-0.068	0.084	-0.040	0.258	0.040	-0.037	0.129
-0.198	1.760	0.175	-0.176	0.210	-0.031	-0.322	0.078	-0.167	-0.021	-0.084	-0.073	0.069	0.125
-0.159	0.058	2.030	0.138	-0.075	-0.075	-0.346	-0.007	0.052	-0.020	0.102	0.011	0.012	-0.056
0.204	-0.108	0.075	1.848	0.049	-0.229	-0.026	-0.103	-0.211	-0.069	-0.218	-0.047	-0.158	0.166
0.089	0.049	-0.075	0.028	1.891	0.074	-0.438	0.092	0.124	-0.104	-0.045	0.019	0.030	-0.186
0.140	0.100	-0.048	-0.190	-0.044	1.976	-0.324	0.002	-0.052	-0.015	-0.042	-0.054	-0.048	0.122
0.104	-0.169	0.246	0.454	0.096	-0.587	0.568	-0.053	-0.006	0.199	-0.283	-0.006	0.155	0.134
-0.045	-0.007	-0.023	-0.055	-0.186	0.009	-0.420	0.089	0.026	-0.073	-0.244	-0.085	0.181	0.084
1.167	-0.898	1.891	-0.575	1.265	-0.265	-0.267	0.216	-1.030	0.758	-2.139	0.627	-0.756	0.813

Bias to Hidden1

-0.319
-0.268
-0.370
-0.096
0.031
-0.009
-0.596
-0.331
-0.085
-0.075
-0.464

Bias to Hidden2

-0.199
0.155
-0.351
-0.203
-0.292
-0.077
-0.075
-0.211

Bias to Hidden3

-0.384

-0.252
-0.219
0.062
-0.052
-0.220
0.013
-0.445
0.229

Hidden1 to Hidden2

0.279	0.087	-0.268	0.012	-0.251	-0.022	-0.151	-0.188	0.023	-0.097	0.001
-0.186	0.138	0.273	-0.198	-0.140	0.295	0.056	0.353	0.091	0.126	0.186
-0.268	0.245	-0.311	-0.300	-0.360	-0.223	-0.144	0.108	0.154	-0.231	-0.025
-0.051	0.196	-0.093	-0.270	0.111	-0.174	-0.206	0.311	-0.081	-0.043	0.207
-0.355	0.022	-0.021	0.191	0.170	-0.271	-0.023	0.036	-0.066	-0.221	-0.017
0.256	-0.148	-0.016	-0.193	-0.372	0.064	0.076	-0.092	0.020	-0.111	-0.005
0.108	-0.292	0.301	0.032	-0.146	-0.225	-0.429	-0.130	0.011	0.008	0.108
0.175	-0.073	0.218	0.234	-0.128	-0.096	-0.390	0.137	-0.283	-0.136	0.180

Hidden1 to Output

-0.305	-0.421	-0.331	-0.006	-0.237	-0.191	-0.214	0.195	-0.322	0.326	0.109
0.076	0.213	-0.451	-0.250	-0.293	-0.133	-0.019	-0.305	-0.203	-0.218	-0.337
-0.073	-0.552	-0.092	-0.374	-0.031	-0.077	-0.352	-0.085	0.169	0.113	0.148
-0.039	0.288	-0.303	-0.207	-0.570	0.132	0.103	-0.429	-0.112	-0.296	-0.402
-0.330	-0.279	-0.006	-0.223	0.382	-0.430	0.300	-0.128	0.025	-0.107	-0.132
-0.484	0.303	-0.172	0.231	0.020	-0.054	0.043	-0.471	-0.266	-0.103	0.075
-0.131	0.266	0.099	-0.051	0.274	-0.075	0.090	0.142	0.381	0.140	0.077
-0.116	-0.329	0.252	-0.230	0.149	-0.375	0.072	-0.151	0.130	0.294	-0.277
-0.898	-0.839	-0.388	0.656	0.459	0.616	-1.089	-0.116	0.545	-0.096	-0.230

Hidden2 to Output

-0.439	0.079	0.160	-0.526	-0.077	0.036	-0.034	0.279
0.308	0.226	-0.049	0.107	-0.007	0.206	-0.465	0.022
0.078	-0.231	-0.162	-0.393	0.203	-0.069	-0.291	0.160
0.231	-0.538	-0.164	-0.241	-0.107	0.396	-0.114	-0.352
-0.128	-0.378	-0.338	-0.013	0.117	-0.508	-0.227	0.183
0.088	-0.566	0.176	-0.028	0.099	-0.022	-0.263	-0.074
0.137	0.397	-0.070	0.154	0.039	0.033	0.125	0.427
-0.099	-0.254	-0.065	0.111	0.237	-0.390	-0.113	0.092
-0.316	0.241	-0.241	0.179	0.652	-0.390	0.571	0.213

C.6 C3P2

Input to Hidden2

-0.875	0.215	-1.083	0.388	-1.225	-0.409	-0.005	0.021	0.778	-0.692	1.115	-0.380	-0.155	-0.229
-0.844	0.336	-1.301	0.704	-0.569	0.368	0.172	-0.166	0.539	-0.234	0.633	0.013	0.050	-0.441
-0.433	-0.331	-0.640	-0.193	-0.244	-0.186	-0.062	-0.113	0.062	-0.277	0.205	-0.165	0.199	-0.307
0.339	-0.572	-0.107	-0.597	0.280	0.315	-0.079	-0.118	-0.631	0.180	-1.070	0.258	-0.162	0.241
0.237	-0.254	0.060	-0.650	0.684	-0.114	0.024	0.098	-0.522	-0.144	-0.725	0.039	-0.348	0.312
0.414	-0.393	0.590	-0.219	-0.444	0.030	-0.354	0.108	-0.419	0.213	-0.817	0.424	-0.566	0.284
-0.844	0.451	-1.463	0.552	-0.471	-0.008	-0.147	-0.090	1.011	-0.436	1.380	-0.620	0.257	-0.306
0.238	0.097	-0.420	-0.662	-0.277	-0.626	-0.128	0.175	-0.069	0.138	-0.184	-0.333	0.194	0.069
0.043	-0.363	0.650	-0.193	0.020	-0.491	0.053	0.301	-0.206	-0.055	-0.391	0.113	-0.187	-0.075
0.152	-0.291	-0.389	-0.465	-0.342	-0.051	-0.429	-0.173	0.079	-0.021	0.018	-0.418	0.152	0.027
-0.087	-0.061	-0.432	-0.185	-0.350	-0.091	-0.173	-0.255	-0.004	-0.231	0.312	-0.234	-0.150	-0.158

Input to Hidden2

-0.639	0.405	-0.078	0.293	-0.050	0.175	-0.147	0.177	0.258	0.153	-0.188	0.136	0.234	-0.275
0.222	0.091	-0.064	-0.398	-0.299	-0.596	0.093	0.003	-0.068	-0.166	-0.416	0.030	0.177	0.000
0.137	-0.046	-0.382	-0.241	-0.397	-0.185	-0.272	0.117	0.211	-0.301	-0.079	-0.135	0.197	-0.270
-0.558	0.014	-0.169	-0.055	-0.045	-0.098	0.110	-0.219	-0.381	-0.220	-0.231	-0.005	-0.406	-0.083
-0.102	-0.294	0.479	-0.277	0.331	-0.086	-0.340	-0.193	-0.356	0.239	-0.296	0.077	-0.095	-0.060
-0.263	-0.126	-0.218	0.415	-0.429	-0.104	-0.204	0.020	0.292	-0.216	0.267	0.074	0.259	-0.386
0.519	-0.683	0.171	0.033	0.186	-0.357	0.058	0.121	0.000	-0.045	-0.638	0.028	0.083	0.007
0.197	0.067	0.268	-0.390	0.195	0.135	0.162	-0.182	-0.320	-0.251	0.118	-0.155	-0.077	-0.204

Input to Output

1.765	-0.413	-0.111	0.067	0.075	-0.025	0.164	-0.042	0.088	-0.055	0.238	0.039	-0.010	0.119
-0.219	1.691	0.192	-0.176	0.213	-0.049	-0.315	0.083	-0.176	-0.019	-0.072	-0.076	0.081	0.157
-0.129	0.053	2.016	0.157	-0.025	-0.077	-0.322	-0.005	0.031	-0.020	0.118	0.027	-0.015	-0.068
0.216	-0.102	0.076	1.841	0.029	-0.227	-0.014	-0.115	-0.219	-0.064	-0.196	-0.060	-0.137	0.171
0.088	0.036	-0.072	0.012	1.915	0.079	-0.397	0.080	0.130	-0.123	-0.056	0.030	0.035	-0.191
0.126	0.099	-0.044	-0.181	-0.043	1.994	-0.320	0.000	-0.040	-0.028	-0.034	-0.064	-0.037	0.113
0.102	-0.193	0.264	0.435	0.068	-0.548	0.572	-0.050	-0.006	0.198	-0.354	-0.003	0.132	0.178
-0.125	0.008	-0.069	-0.102	-0.206	-0.022	-0.404	0.112	0.019	-0.099	-0.283	-0.066	0.344	0.132
1.011	-0.845	1.658	-0.372	1.164	-0.248	-0.221	0.203	-1.040	0.894	-2.105	0.688	-0.603	0.680

Bias to Hidden1

-0.280
-0.143
-0.320

-0.102
-0.013
0.077
-0.560
-0.283
-0.042
-0.096
-0.431

Bias to Hidden2

-0.188
0.243
-0.326
-0.132
-0.278
-0.054
-0.084
-0.181

Bias to Hidden3

-0.369
-0.261
-0.203
0.077
-0.001
-0.213
-0.006
-0.423
0.391

Hidden1 to Hidden2

0.273	0.072	-0.272	0.023	-0.221	-0.019	-0.151	-0.180	0.026	-0.097	0.000
-0.171	0.164	0.293	-0.180	-0.118	0.310	0.082	0.378	0.114	0.147	0.204
-0.255	0.263	-0.307	-0.294	-0.356	-0.215	-0.135	0.109	0.163	-0.223	-0.016
-0.040	0.218	-0.086	-0.248	0.138	-0.151	-0.195	0.329	-0.067	-0.026	0.216
-0.361	0.019	-0.025	0.200	0.155	-0.267	-0.042	0.046	-0.079	-0.215	-0.015
0.269	-0.135	-0.007	-0.176	-0.340	0.054	0.094	-0.087	0.036	-0.104	0.009
0.104	-0.271	0.289	0.036	-0.149	-0.208	-0.421	-0.127	0.002	0.015	0.104
0.185	-0.070	0.228	0.232	-0.123	-0.090	-0.381	0.151	-0.274	-0.123	0.195

Hidden1 to Output

-0.319	-0.413	-0.315	0.014	-0.243	-0.175	-0.218	0.190	-0.324	0.356	0.113
--------	--------	--------	-------	--------	--------	--------	-------	--------	-------	-------

0.095	0.230	-0.452	-0.257	-0.318	-0.138	0.003	-0.321	-0.204	-0.244	-0.322
-0.050	-0.544	-0.080	-0.376	-0.034	-0.075	-0.335	-0.056	0.165	0.116	0.167
-0.041	0.277	-0.297	-0.220	-0.546	0.109	0.082	-0.429	-0.104	-0.298	-0.405
-0.341	-0.268	-0.004	-0.206	0.385	-0.415	0.296	-0.132	0.035	-0.100	-0.126
-0.472	0.324	-0.166	0.243	0.024	-0.030	0.037	-0.473	-0.270	-0.101	0.064
-0.159	0.277	0.098	-0.065	0.280	-0.072	0.081	0.151	0.345	0.138	0.075
-0.105	-0.305	0.251	-0.209	0.159	-0.379	0.068	-0.133	0.125	0.293	-0.283
-0.956	-0.745	-0.350	0.699	0.488	0.720	-1.115	0.008	0.446	-0.030	-0.222

Hidden2 to Output

-0.440	0.088	0.162	-0.528	-0.074	0.049	-0.023	0.305
0.342	0.218	-0.059	0.141	-0.003	0.194	-0.491	0.035
0.104	-0.205	-0.163	-0.385	0.197	-0.059	-0.288	0.173
0.230	-0.528	-0.168	-0.243	-0.106	0.400	-0.097	-0.360
-0.100	-0.374	-0.347	-0.044	0.127	-0.499	-0.220	0.197
0.096	-0.561	0.167	-0.032	0.092	-0.026	-0.256	-0.080
0.152	0.393	-0.077	0.171	0.027	0.014	0.110	0.421
-0.074	-0.225	-0.055	0.113	0.236	-0.382	-0.110	0.084
-0.267	0.251	-0.233	0.194	0.634	-0.380	0.569	0.206

C.7 C3P3

Input to Hidden1

-0.285	-0.088	-0.936	0.125	-1.368	-0.359	-0.057	-0.149	0.450	-0.439	0.601	-0.281	-0.198	0.260
-0.233	-0.566	-0.191	-0.409	0.738	0.208	-0.088	-0.236	-0.304	0.074	-0.735	-0.072	-0.983	-0.259
-0.269	-0.004	-1.288	0.383	-1.423	-0.383	-0.285	-0.426	0.083	-0.657	0.711	-0.264	0.668	-0.624
-0.206	-0.254	-0.352	-0.382	-0.188	-0.011	-0.215	-0.643	-0.394	-0.315	-0.436	-0.278	-0.262	-0.213
0.011	-0.242	-0.746	-0.167	-0.405	-0.295	-0.409	-0.191	-0.362	-0.588	-0.186	-0.243	-0.417	-0.351
0.137	-0.228	0.199	-0.146	-0.819	-0.130	-0.585	-0.135	-0.108	-0.157	-0.182	-0.021	-0.483	0.313
-0.728	0.089	-0.264	-0.210	-0.142	-0.141	-0.300	-0.140	0.096	-0.049	-0.081	-0.570	-0.190	-0.465
-0.164	0.520	-1.210	0.182	-1.401	-0.501	-0.523	0.005	0.218	-0.220	0.441	-0.409	0.834	-0.310
-0.093	-0.415	-0.212	-0.026	-0.729	-0.510	-0.400	0.104	-0.091	-0.424	-0.057	-0.128	-0.239	-0.301
-0.237	-0.280	-0.699	-0.360	-0.464	0.014	-0.619	-0.312	-0.044	-0.096	-0.131	-0.429	-0.085	0.154
0.100	0.449	-1.527	0.874	-1.488	-0.626	-0.172	-1.199	0.225	-1.068	1.567	-0.586	0.841	-1.292

Input to Hidden2

-0.490	-0.214	-0.072	-0.170	0.333	0.112	-0.375	0.087	-0.104	0.258	-0.724	-0.092	-0.306	-0.199
0.208	-0.215	-0.746	-0.220	-0.767	-0.557	-0.414	-0.165	0.015	-0.401	-0.277	-0.104	0.206	0.040
0.079	-0.167	-0.364	-0.311	-0.315	-0.339	-0.381	-0.048	0.073	-0.331	-0.293	-0.246	-0.036	-0.328

-0.243	-0.274	-0.355	-0.024	-0.351	-0.160	-0.196	-0.356	-0.378	-0.344	-0.025	-0.112	-0.488	-0.136
-0.303	-0.505	0.040	-0.273	0.062	-0.178	-0.514	-0.333	-0.365	0.093	-0.149	-0.136	-0.250	-0.157
-0.265	-0.511	-0.202	-0.038	-0.109	-0.181	-0.415	-0.064	0.041	-0.079	-0.303	-0.080	-0.160	-0.161
0.203	-0.400	-0.377	0.245	-0.557	-0.586	-0.275	-0.277	0.104	-0.487	-0.185	-0.288	0.238	-0.286
-0.192	-0.200	-0.180	-0.408	-0.103	0.193	-0.245	-0.193	-0.428	-0.313	-0.058	-0.272	-0.306	-0.262

Input to Output

-0.228	0.172	0.068	-0.112	0.235	-0.028	0.350	-0.072	0.052	-0.004	-0.172	-0.029	-0.075	0.058
1.188	-1.081	-0.317	-1.177	1.106	-0.058	-0.009	0.394	-0.355	0.383	-0.852	0.007	-0.884	1.583
-0.176	-0.219	-0.160	0.418	0.233	-0.032	-0.147	0.172	0.003	0.008	0.228	-0.011	-0.077	0.074
-0.588	-0.068	2.092	-0.059	1.153	-0.340	0.126	0.153	-0.615	0.476	-1.894	-0.340	-1.082	0.563
-0.140	-0.123	0.017	0.025	-0.028	0.253	-0.331	0.318	-0.005	0.089	0.032	0.269	0.281	-0.480
-0.230	-0.207	0.122	-0.496	1.114	1.133	0.114	-0.415	-0.336	-0.212	-0.239	-0.470	-1.343	-0.285
-0.422	0.096	0.561	-0.088	1.451	0.095	0.023	0.253	-0.177	-0.120	-1.235	0.224	0.022	-0.573
0.172	-0.177	0.496	-0.297	0.302	0.005	-0.395	0.060	-0.066	-0.034	-0.515	-0.064	-0.141	-0.027
0.302	-0.433	0.821	-0.138	0.475	-0.288	-0.391	-0.004	-0.368	0.103	-0.791	0.273	-0.165	0.303

Bias to Hidden1

-0.261
-0.272
-0.578
-0.379
-0.610
-0.271
-0.552
-0.797
-0.651
-0.291
-0.416

Bias to Hidden2

-0.422
-0.391
-0.423
-0.498
-0.524
-0.240
-0.499
-0.660

Bias to Hidden3

-0.169

0.199
-0.019
0.336
0.004
0.360
-0.567
-0.390
-0.002

Hidden1 to Hidden2

0.095	0.022	-0.376	0.051	-0.175	-0.066	-0.278	-0.249	-0.003	-0.083	-0.218
-0.238	0.038	0.197	-0.377	-0.298	0.110	-0.024	0.193	-0.056	-0.014	0.147
-0.315	0.147	-0.296	-0.306	-0.310	-0.248	-0.217	0.123	0.190	-0.233	-0.017
-0.098	0.118	-0.173	-0.399	0.023	-0.255	-0.291	0.220	-0.145	-0.114	0.128
-0.271	0.089	-0.062	0.096	0.061	-0.358	0.034	-0.030	-0.133	-0.242	-0.118
0.105	-0.249	-0.066	-0.145	-0.267	0.026	-0.039	-0.116	0.033	-0.081	-0.122
0.181	-0.314	0.268	-0.162	-0.302	-0.372	-0.360	-0.199	-0.086	-0.111	0.175
0.121	-0.115	0.111	0.096	-0.269	-0.244	-0.447	-0.008	-0.405	-0.234	0.024

Hidden1 to Output

-0.155	-0.239	-0.276	0.068	-0.228	-0.180	0.005	0.154	-0.250	0.356	0.025
0.127	0.475	-0.473	-0.020	-0.125	0.076	-0.196	-0.374	-0.106	0.102	-0.692
0.142	-0.215	0.050	-0.160	0.142	-0.131	-0.080	0.060	0.094	0.287	0.128
-0.289	0.489	-0.562	0.025	-0.330	0.143	0.219	-0.432	-0.140	-0.093	-1.107
-0.054	-0.147	0.106	-0.127	0.288	-0.188	0.291	0.003	0.159	0.033	0.101
-0.212	0.248	-0.114	0.174	0.130	0.006	0.153	-0.269	-0.080	0.063	-0.016
-0.616	0.500	-0.414	-0.321	-0.079	-0.507	0.131	-0.364	-0.166	-0.152	-0.740
-0.196	-0.198	0.108	-0.195	0.169	-0.324	0.041	-0.278	0.134	0.296	-0.516
-0.447	-0.153	-0.599	0.297	-0.005	0.368	-0.425	-0.399	0.157	-0.258	-0.718

Hidden2 to Output

-0.131	0.125	0.146	-0.262	0.044	0.174	-0.138	0.363
0.357	0.277	0.061	0.117	0.133	0.347	-0.217	0.089
0.142	-0.092	-0.102	-0.220	0.155	0.024	-0.245	0.124
0.489	-0.456	-0.029	-0.142	0.136	0.266	-0.263	-0.074
-0.072	-0.118	-0.186	0.112	0.118	-0.270	-0.168	0.243
0.189	-0.258	0.144	0.115	0.243	0.104	-0.089	0.129
0.218	-0.221	-0.175	-0.152	-0.188	-0.128	-0.470	0.203
0.004	-0.296	-0.046	0.092	0.293	-0.336	-0.139	0.153
-0.277	0.055	-0.269	0.100	0.335	-0.215	0.187	-0.135