Learning Hunting behavior in cubs through Observation, Imitation and Reinforcement Learning

Pragadheeshwaran Thirumurthi *Prarthana G Alevoor*
{pragadh, prar1986}@cs.ucla.edu
Professor: Michael Dyer {dyer@cs.ucla.edu}

INTRODUCTION:

We have proposed an animat learning environment where a cub learns to hunt from a mother by using imitation learning as well as reinforcement learning. The cub (learning agent) imitates and learns the sequence of goals for hunting by observing the actions performed by its mother (teaching agent). Later when the cub grows, it stops imitation and starts hunting on its own along with positive and negative reinforcements from the mother. We try to see how the changes in environment affect the learning behavior of the cub. Our hope is that the cub will perform better than the mother and tries to behave the same way when it becomes a mother.

HYPOTHESES:

We hypothesize an environment where the cub learns the hunting behavior by imitating the actions of the mother. This learning is performed over various phases with various kinds of learning.

1. *Observation and Imitation Learning Phase*
2. *Reinforcement Learning Phase*
3. *Cub hunting independently and finally becoming mother*

While performing various learning, the rules defined for the actions and the inputs to define particular actions play an important role in the neural network. We also hypothesize, that even though cub stops imitating the mother as it grows, there are possibilities that some inhibited characteristics of the cub are shown when in similar situations.

GOALS:

We define the goals based on the phases of learning

Phase 1:

a) The cub observes the hunting behavior of the mother.
b) The cub learns various goals that have to be applied under different environment (type and nature of prey).

Phase 2:

a) The cub imitates the actions learnt from mother
b) The mother gives reinforcement for the actions that the cub performs.
c) The cub uses positive reinforcement learning from the mother to achieve the goal.

Phase 3:

a) The cub performs actions on its own
b) The cub senses new environment(group of preys) and implements its learnt behavior
c) It uses negative reinforcement learning by itself
d) A special neuron gets activated when the Cub becomes mother

ENVIRONMENT AND PHYSICS:

The key design decision made while creating the environment was that the physical representation of the world is simplified so that more time is focused on the various experiments conducted for the learning behaviors.

1. *Environment*: The environment is depicted in a 2-D world in a 500 x 500 board (array of points). Each physical object is randomly placed on the board via an image which occupies certain width and height. The object is represented by the middle (x, y) point as its co-ordinate position. Multiple objects might occupy the same tile. Thus each object is free to move in any direction without being blocked by other animats.
2. *Creatures*:
    a) Prey: The environment consists of 2 kinds of preys with different sizes to represent the diversity of the prey sizes for learning
        i. Goat: Small, Medium and Large sized
        ii. Large sized buffalo

        The preys do not have any learning component. They can sense the predators around the range and moves in a direction away from the predator if possible, else they move in a random direction.

    a) Predator: Lion Mother and Cub

        Predators move in random directions in search of preys to hunt only when it is hungry (energy is in certain range). Predator attacks the prey by overcoming the speed of the prey and attacking the prey till it dies.

The environment controls the number of preys appearing on the board. It starts with random placement of preys on board. It chooses a time period based on the age of the cub to introduce more preys in the environment.

Each creature has a vision sensor and is associated with the range which varies for various creatures and varies over the age of the cub. As the cub grows its range improves and it can sense farther objects by age.

The creatures can move forwards or backwards and may face one of 8 directions (0°, 45°, 90°, 135°, 180°, 225°, 270°, or 315°). At each time cycle, the creatures may either move one tile in any of the 8 directions. Each creature moves by a constant step based on its range, size and speed. Movement of the animat is controlled changing the (x, y) co-ordinates of the animat

Each of the creatures has energy which is slowly lost over time at a rate based on the actions performed by the creatures. Since our project aims to capture the hunting behavior for the cub (predator), we decided not to reduce the energy of the prey while it is wandering but only gets reduced when it’s being attacked. Though for the predators the energy decreases over time but increases whenever it kills and eats the preys. The amount of energy modifies various abilities of the creatures, such as attack strength, deciding fighting actions and the prey to hunt. When the energy of a creature reaches zero it is dead and hence disappears from the world.

SYSTEM ARCHITECTURE:

This architecture consists of three phases of learning represented by 3 networks. The network A affects network B and network B affects network C by using the weights.

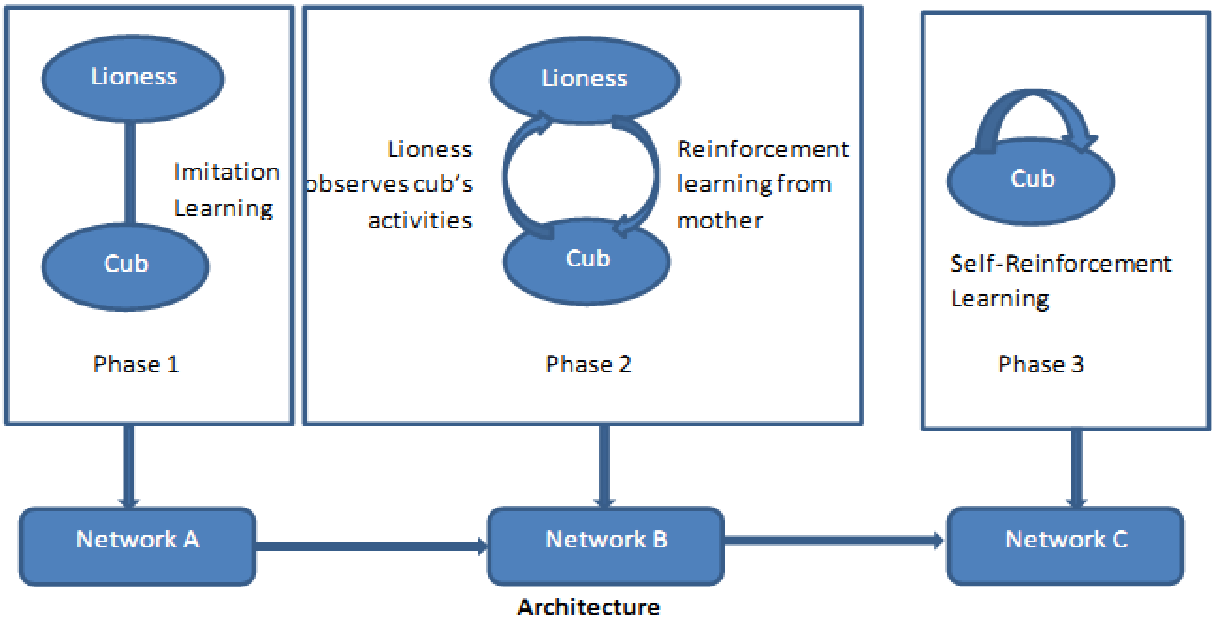


Fig 1: Phases of learning

Phase 1 (Imitation Learning):

The cub's neural network does not know the goals (what to hunt, when to hunt) and how to achieve them. By observing the mother, the cub builds its neural network and learns the goals and the sequences of actions for various goals.

- Goals include the sequence of actions that a cub should perform to achieve the goals.
    - Initial hunting behavior includes both successful and unsuccessful hunt.
- One-to-one connection between sensors that detects the properties of the lioness and corresponding sensor and effector nodes in the cub
    - Scenario: When the lioness senses a small prey, it first snarls and then pounces.
    - This triggers the snarl and pounce motor neuron in the cub.

Phase 2 (Reinforcement learning):

In this phase, the cub stops following the mother. The cub processes environmental information and chooses appropriate action that has to be applied.

- The sensors and effector nodes of the cub gets triggered when an action is initiated by the cub
    - Scenario: The cub sees the prey, its snarl and pounce motor neuron gets triggered.
- The weights in the network are altered by the reinforcements from the mother.

Phase 3(Self learning):
In this self-learning phase, the cub is detached from lioness and does not receive any reinforcement from the mother.

- In case of an unsuccessful hunt, the cub modifies its neural network based on the new environment)
- The cub's actions and negative reinforcement affect the weights in the network.
    - Scenario: While hunting, the cub's energy level deceases while being attacked by a group of preys and it fails.
    - This causes change of weights and the cub learns to go for hunt with an increased energy or avoid such situations.
- As the cub grows it imitates its mother in later stages of life.
    - Scenario: While the mother hunts a group of preys and when it senses danger it 'growls' to show the sign of pain.
    - When the cub becomes mother, it growls when it senses its cub in danger.

Simulation:
The simulation is programmed in Java. Java gives the flexibility of using object model. Classes represent the Animats with their sensors and neurons, the world and the neural network. Each of the objects is kept in a collection depending on its type for easier access. Further code has been explained in the [Appendix F] describing the algorithms and neural network architecture.

PREY (GOATS AND BUFFALOS):
Preys are randomly placed in the world. The world uses the age of the cub to determine the time period to introduce new preys into the world. The age of the cub also determines the number of preys to be grouped together around the same area while being introduced to the world.

Sensors: Their "sensors" track the existence of any predator in its range which is based on size and type of animat. Animats with smaller sizes have smaller range. The sensor also tracks the existence of other animats of the same type of this prey.

Motion motors: Nonexistence of any predators causes a random motion of the preys. When it senses a predator, its hardwired brain tries to move away from the predator based on the direction of the predator sensed by its sensor.

Hardwired Brain:
The preys' brain is a simple decision tree and state machine which takes in direction of the predator and decides the movement of the motor and its direction. Prey brain logic:

1. While alive check whether predator is in its range
2. If predator found
    a) If prey not in a group then move away from the predator (decision tree presented in Appendix F-1)
    b) Else attack the predator
3. If no predator found then randomly wander

PREDATORS:
Mother lion and cub aim to hunt the preys.

Energy: They need to eat to maintain their energy to survive and perform actions. Predator hunts only when the energy is between certain range (20%-70%). Energy decreases when the cub moves randomly or performs any of the actions. Various movements and actions consume energy at various rates. The decrease in the prey's energy while hunting also depends on the predator strength. When the predator hunts and eats the prey, its energy increases.

Actions: The actions they can perform are

1. Search
2. Move
3. Run (Move at a faster speed)
4. Snarl
5. Roar
6. Eat
7. Pounce
8. Growl

Selection of Prey: The predator selects the prey to hunt within its range based on a combination of the prey's energy, distance and its energy. First it decides the weakest prey:

Weakest prey = distance_weight*distance_prey + prey_energy_weight * prey_energy

Then based on the predator's energy and the type of prey, it makes a decision whether to approach this prey or not. The type of prey gives a sense of the energy it might gain by hunting it.

How cub differs from mother:

- The age of the cub increases with the clock cycle.
- The range of the cub increases by 50 board points when its age increases. Finally when the cub matures and becomes a mother it will have the same range as was started with a mother.
- Cub learns to perform a new action – drawback when it senses its energy is less and it can't fight anymore

Sensors/States:
Vision Sensor: The predators have a vision sensor which gives all the information required from the surrounding environment (within its range) such as

- The type, size and position of the prey.
- When the predator is very close to the prey it can sense the energy level of the prey.
- Senses the prey

Neural Network:
The brain and actions of the predators are controlled by neural network. Both mother and cub together have a single feed forward neural network which decides the actions to be taken by the predator thus learning the goals.

- The neural network is based on switching mechanism where the input and output source of the neural network varies based on the network's state or some events.

  - We initially train the network based on the inputs received from the mother and output the action to be taken based on the surrounding environment. This forms a one-to-one mapping of the actions performed by the mother to the actions performed by the cub.
  - The output values fire the actions in the mother in the training period. During this part of the training, the cub observes the actions of the mother by following the mother and stores it in the episodic memory which is later used by the cub while it hunts on its own.
- These set of actions are used to learn the goal sequences which can be either internal goal nodes like hunger which trigger 'eat' action or external goals like 'hunt_prey' which triggers serious of actions for hunting(search, run, snarl, pounce, claw) [Appendix A-2: mentioned in the neural model]
- The action selection neural network is updated once per cycle.
- The actions performed by the lion (mother/cub) based on the output is decided based on a set of rules or a classifier system. Each of the rules has a set of conditions to be met for an action to fire. This classifier system is the basis to give the corrected values for back propagation in the neural network.[Appendix B]
- *Action-Selection Neural Net*

  Inputs:

  - Lion energy(mother/cub) energy
  - distance to weakest prey
  - type of prey
  - energy of the prey
  - size of the prey

  Outputs:

  - Search for a prey
  - Pounce on a prey
  - Eat a prey
  - Roar/Snarl/Growl
  - Claw the prey
  - Drawback from hunting

- Further when the cub reaches the age of 2 years we use the neural net to take the inputs from the cub's environment and trigger an action neuron on the cub.
  - At this stage we also develop this neural network by using reinforcement learning, where the mother gives positive and negative reinforcements.
  - For example, if the cub tries pouncing on a prey which is very far then the mother scolds the cub.
  - Similarly if the cub takes a correct action like snarling to frighten the prey then the mother praises.
- When the cub reaches the age of 4 years, it tries to develop its brain further via self-learning.
  - In this stage we try to show that the cub adds another goal 'drawback' to its nodes where it runs away when being attacked by a group of preys.
- The age of the cub also triggers another neuron for the cub only when it matures.
  - When the cub becomes mother at the age of 5 years, the neural network triggers the neuron responsible for nursing the young ones and makes a growling sound when in danger which was triggered in its mother when it was a cub.

METHODOLOGY:

Our major aim was to measure the learning of the cub during various phases.

- Initial task was to decide the various inputs to be given to the neural model, how these inputs would differ between the cub and the mother. For example, the mother need not consider the energy of the prey while hunting where as for the cub; a powerful prey can cause harm and hence it should avoid it.
- Next task was to build the neural model taking inputs from the mother. While building this we tried implementing the reinforcement learning – whether to hunt or not based on the surrounding environment like the energy of the lion, size and distance of the prey. This gave the ideal behavior of the mother while hunting which can be henceforth compared with the cub. [Appendix E-1]
- Another major task was to decide when the cub should move to the next phase of learning. This could be either decided by the mother like after some “n” iterations of hunting; or could be decided by the properties associated with cub. We chose to decide based on the age of the cub as the cub’s brain grows and it learns to take its own decisions. Hence when the cub grows it decides to leave the mother and hunt on its own.
- Another measure of learning was failure analysis where the cub learns on its own whether to approach the prey or even to drawback when its energy becomes low while hunting. Since each action is based on a set of rules we have used the classifier method to select the actions based on the input conditions.
- When the cub becomes mother, it triggers a new neuron. Hence age criterion plays an important role in deciding the activation of networks in Fig 1.

EXPERIMENTS, RESULTS and INTERPRETATION:

In our proposal we listed several stages/experiments. We did not include basic stages like the system “correctness” test to make sure that the inputs from the lion (mother/cub) triggers actions. We performed various experiments based on our phases:

Phase I:

Experiment:

- Lioness hunts a prey of various sizes, single preys and groups and decides its actions based on the classifier system
- Cub observes the mother to build its episodic memory

Results and Analysis:

- The error graph was obtained for the mother [Appendix E-2]. As seen from the graph, the mother reaches the ideal behavior and is successful in hunting all the preys in the world.
- The episodic memory is associated with the inputs from the mother’s surroundings.

Phase II:

Experiment:

- Gets activated when the cub reaches a certain age

- Cub uses the already built episodic memory to decide the goals and actions based on its surrounding inputs.
- Cub also gets inputs from mother – reinforcements like scold and praise
- See how the change in the vision range of the cub as it grows affects its hunting

Results and Analysis:

- The neural network switches the inputs and outputs to the cub when the cub reaches age 2.
- When the cub goes hunting on its own, the error graph [Appendix E-2] obtained is different from the learning growth of the neural model with inputs from the mother. We believe this is due to the fact that a new input – energy of the prey causes a major change in the neural model. Since energy of the prey is not a criterion for the mother while hunting as it is strong to hunt any kind of prey. Whereas since the cub is in growing stages, the energy of the prey will help decide which prey to choose.
- With the reinforcements from the mother, the neural model starts performing better(reduced error rate)
- When the cub grows, its range changes by age. Hence in the beginning it has a low range of 100pts and grows to be 250points when it becomes a mother. As seen from the screenshot [Appendix C-4] , a lower range for the cub compared to the preys (range of 150pts) causes the preys to be more intelligent and hence avoid the cub. Hence in the initial stages a cub would hardly find a prey in its range. But when its range increases it performs successful hunting.

Phase III:

Experiments:

- Cub hunts independently
- When the cub senses the preys in the group it should drawback based on its energy
- When the cub becomes mother, a new neuron should be triggered where it performs growling when its cubs are in danger.

Results and Analysis:

- We observed that the preys sense their surroundings and start attacking when in groups. But the cub does not drawback in various scenarios and eventually dies.
- When the cub becomes mother after 5 years, the growling action is triggered when it senses danger.

Effectiveness of the neural network:

We wanted to compare the neural model with a simple case base scenario.

- Use case bases [Appendix F-8] which consists of a list of cases and each case is a combination of agent inputs and the actions performed based on the inputs.
- Initially build the case bases from the hunting behavior of the mother(cub observing)
- When the cub starts hunting on its own, this model finds the closest case from the case base for the inputs from the cub.

- Calculation of closest case (based on the Euclidian distance) from the list of cases is based on the smallest distance of the similar cases. The action performed from the closest case would be the action output of this model.
- This action output would trigger the action for the cub.

Experiment:

- Compare the error rate of this model with the neural model developed before

Results and Analysis:

- This model does not perform well at any point of time, as per the graph [Appendix E-3] but the average over a shorter period seems to be better than the neural model. This is due to the fact that the neural model initially takes some time for learning and starts performing better only at a later stage.

INSTRUMENTATION

Based on the various experiments conducted, we measured the performance of the neural network developed.

- This network cannot be effectively used without training for a long period. This time frame is indecisive and varies based on the inputs for the network.
- We also observed that a drastic change in the input of the network (prey energy) effects the networks output. This might be due to the fact that this input may not be given any priority while developing the network from the mother's inputs.
- We also observed from the graphs that reinforcement learning was slower as compared to the imitation learning for the cub.
- Another noteworthy observation is that(as per the hypothesis), even though the cub stops imitating the mother at later stages, some of the actions taken by mother like growling when in danger were performed by the cub when it is nursing its young ones.
- As hypothesized we observed that rules play an important role in the learning of the network
- We also observe that the cub does not perform as well as the mother. This might be due to the fact that there are additional inputs to the cubs environment and the fact that neural network is trained on mother and hence is assumed to be the ideal behavior which is difficult to achieve in mirrored neurons.

ISSUES AND PROBLEMS ENCOUNTERED:

Random Direction:

Animats choose a random direction to move if there are no enemy animats in their range. We generate a random number from 1-9 to move the animat in appropriate direction. The problem with random numbers in Java is that the numbers generated are pseudo random numbers which are not really random numbers. So during each run of the program, if the animat's range is free from enemies, the animat might moves towards the same set of points. However if there are enemy animats in its range, the animat would either move towards the prey or away from the prey (based on the type of the animat).

Minimum number of preys for learning:

Initial runs of the experiment had problems with mother dying out too fast due to starvation, hence there was a requirement for minimum number of preys on the world for the predators to hunt and learn.

Action issues:

When the cub tries hunting based on the learnt neural network, the preys move while it is being clawed or pounced upon. So this results in some cases where the cub starts hunting a prey which escapes and then the cub changes its target prey.

Increased vision and smell:

In the proposal we had proposed that the vision and smell sensors increase their power when the lion has reduced energy. Over the period of the project development, we realized that speed of the predator increases when it senses the prey. And also the vision range improves by age. Hence the task has been carried over by age instead of energy.

TOOLS/PACKAGES:

We have used Recursive Porous Agent Simulation Toolkit (Repast) library for the neural framework.

- It is an agent-based modeling and simulation toolkit
- Easier to dynamically access and modify agents and model at run time
- It internally uses JOONE neural network framework

Since we did not want to redo the engineering on the neural framework we decided to use REPAST. Another advantage was the provision of controlling the model while running it as well as tracking the error of the neural model built.

IDE/Language: Java in Eclipse IDE

CURRENT STATUS

Overview of various components and their status:

Environment and Physics [DONE]

- Define the world and the creature
- Set the physics of the environment

Neural Model [DONE]

- Learn to use the REPAST neural model
- Build the model with the input and output switching
- Reload the world every clock tick

Build Episodic Memory of Cub for Imitation [DONE]

- Build the case bases for the mother with agent inputs and the sensory items(features from the environment) while the cub observes the mother
- Map the case bases with the goals(sequence of actions) to be taken

Classifier System [DONE]

- Define the set of rules and the corresponding actions to be taken
- Classifier system to be fed to the neural model for back propagation

Basic GUI [DONE]

- Load the images for each animat

- Include the randomness in the sizes of the preys used

Build Error Graphs [DONE]

- Build error graph for the mother while its hunting
- On the same graph build the cub's error distribution during various stages of learning

Trigger new networks based on cub's age [DONE]

- Track the cub's age and correspondingly trigger various neurons in the various networks
- Activate the mother neuron for the cub when it becomes mother

Reinforcement Learning [DONE]

- The cub hunts on its own
- Mother gives positive and negative reinforcements(scold and praise)
- Feedback the reinforcement to the neural model

Self-Learning [In Progress]

- We introduce the groups in the preys which start attacking the predator when it senses the predator instead of moving away from the predator.
- The logic has been implemented [Appendix F-7] but the neural network is still not learning properly.
- The cub escapes in some cases but dies off in others

CONCLUSION:

Based on the various experiments performed, we conclude that the cub learns the hunting behavior from the mother based on the sequence of goals by imitating the actions performed by the mother. Further reinforcements improved the neural network and the cub applies self-learning in the final stage. It is not completely successful in escaping the groups of preys. Even though the cub could not perform as good as the mother, the cub learns imitation at younger age and applies in its later stages of life in similar situations.

CONTRIBUTIONS:

We have worked together to propose the various kinds of learning – imitation and reinforcement, and submitted our proposal. Later while implementing we divided the work equally. Pragadheeshwaran initially developed and implemented the base structure model for the animats including the base methods which were used in the neural model implemented by Prarthana. Pragadheeshwaran further continued with GUI, Reinforcement learning with positive and negative reinforcements and learning of cub while hunting preys in groups; while Prarthana worked on building the episodic memory and imitation learning, Classifier Systems, Case base model and neuron triggering when the cub becomes mother. Even the report was written based on the work divided during implementation.

References:

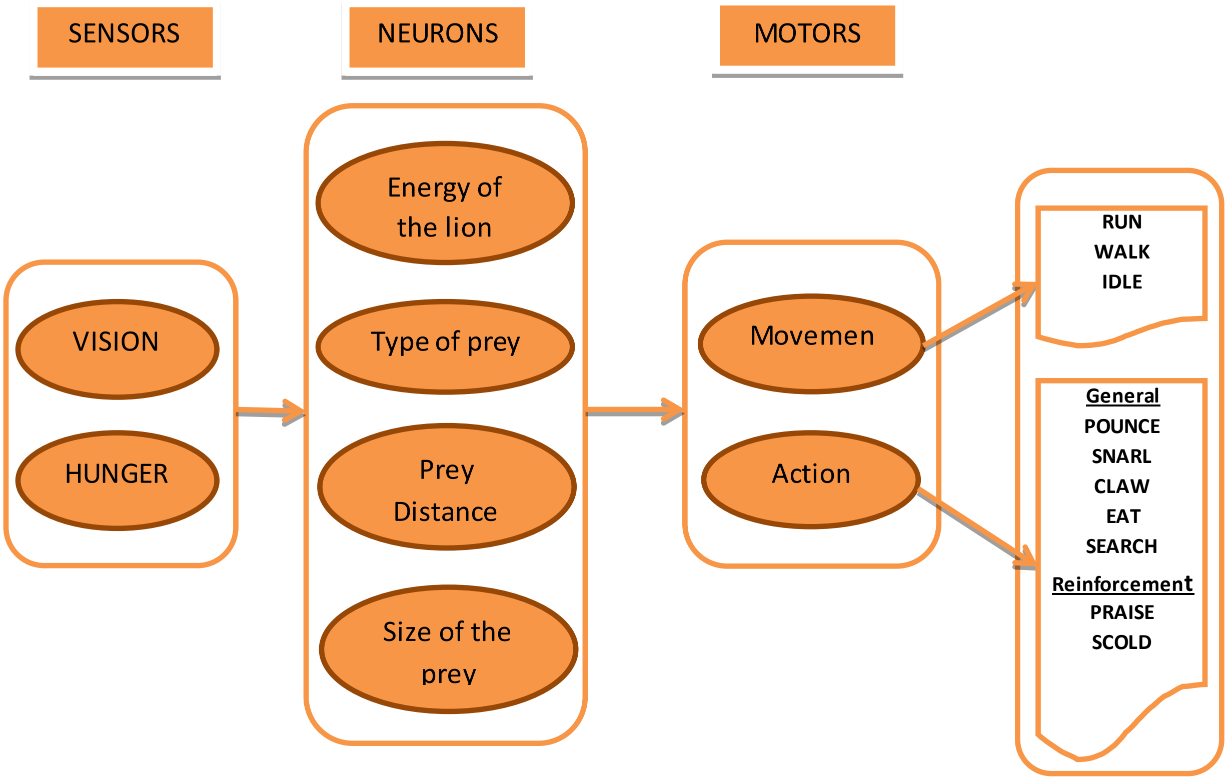
1. http://repast.sourceforge.net/repast_3/index.html
2. http://www.nmai.ca/research-projects/agent-imitation
3. Wood, M. A., *An agent-independent task learning framework* Thesis (Doctor of Philosophy (PhD)). University of Bath

4. Michael Dyer & Frederick Crabbe, *Observation and Imitation: Goal Sequence Learning in Neurally Controlled Construction Animats: VI-MAXSON (2000)*
5. Michael W. Floyd*, Improving the performance of a RoboCup case-based imitation agent through preprocessing of the case base*, Carleton University (Canada).
6. Alfredo Weitzenfeld, "*A Prey Catching and Predator Avoidance Neural-Schema Architecture for Single and Multiple Robots*", ACM 2008

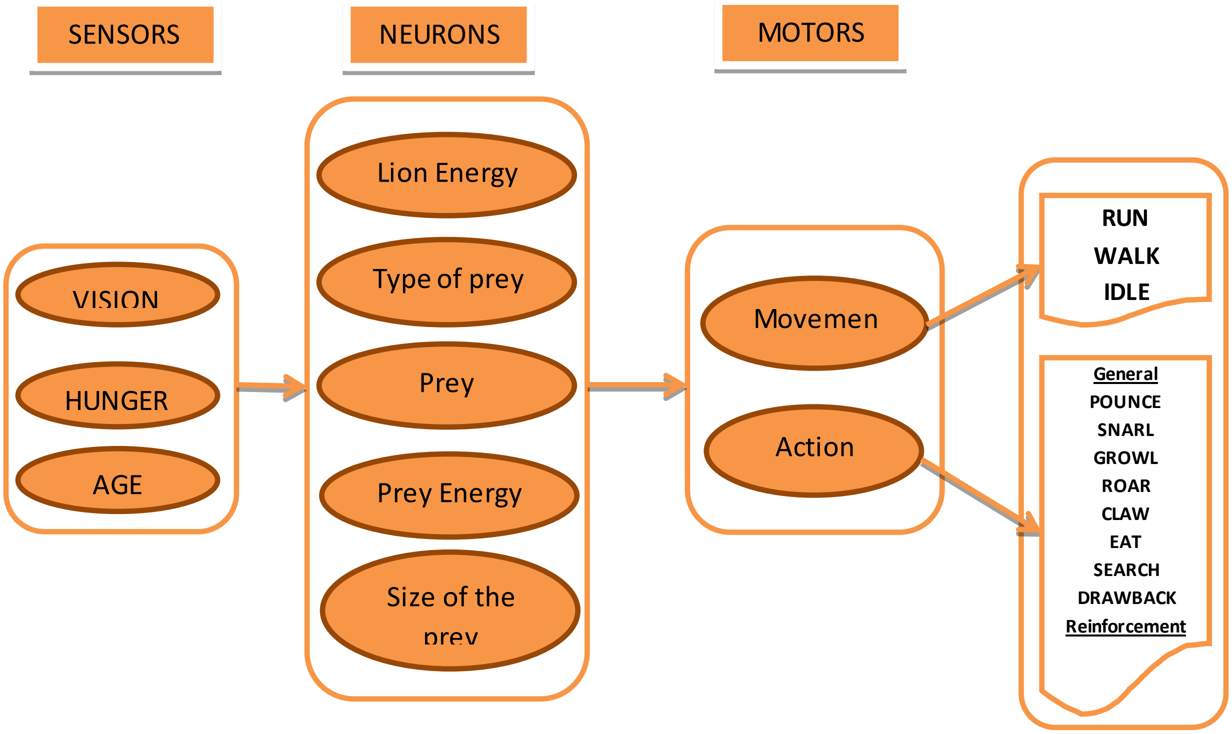
Appendix

A. Neural models

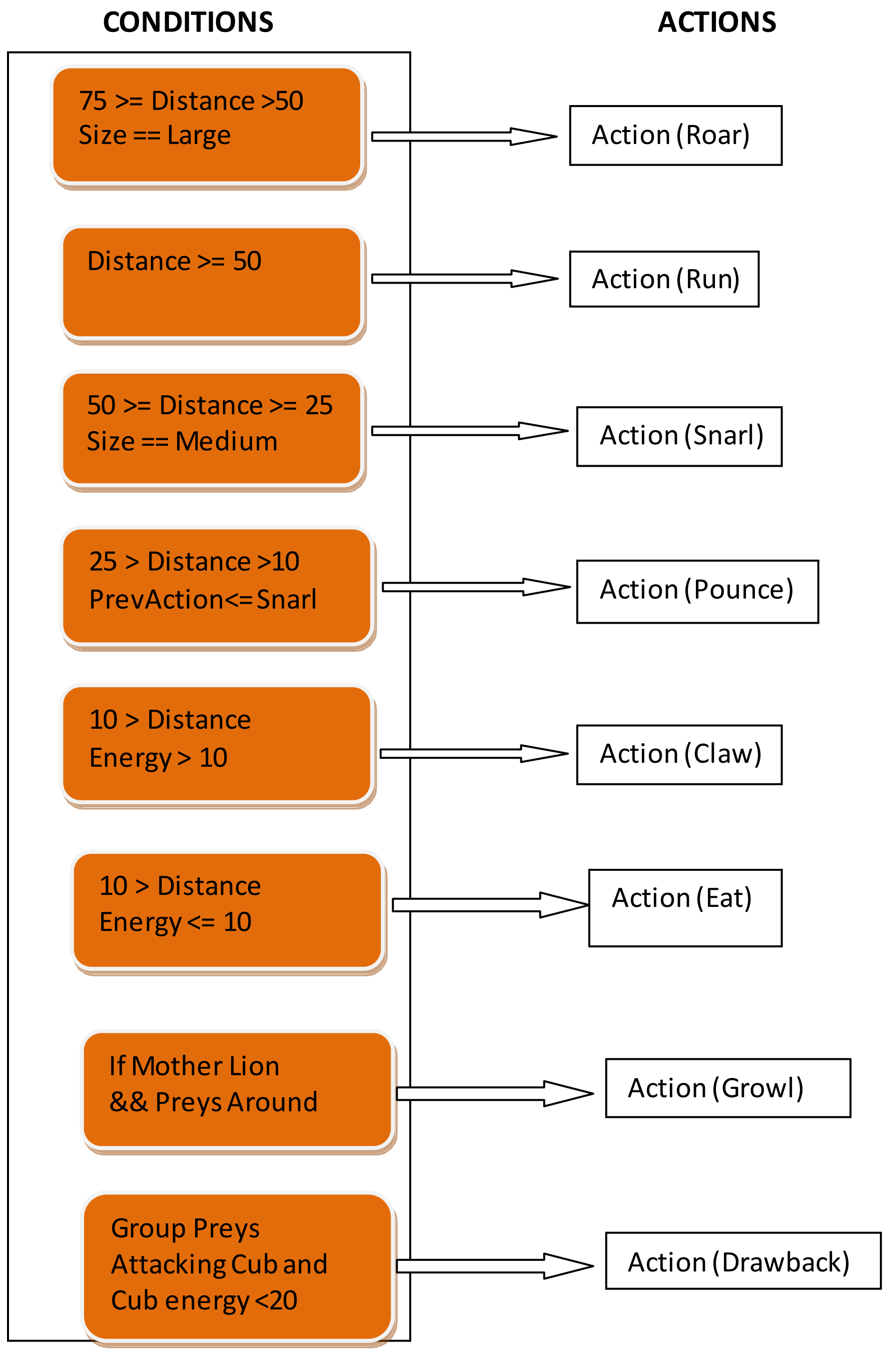
1. Neural model for the mother



2. Neural model for the cub



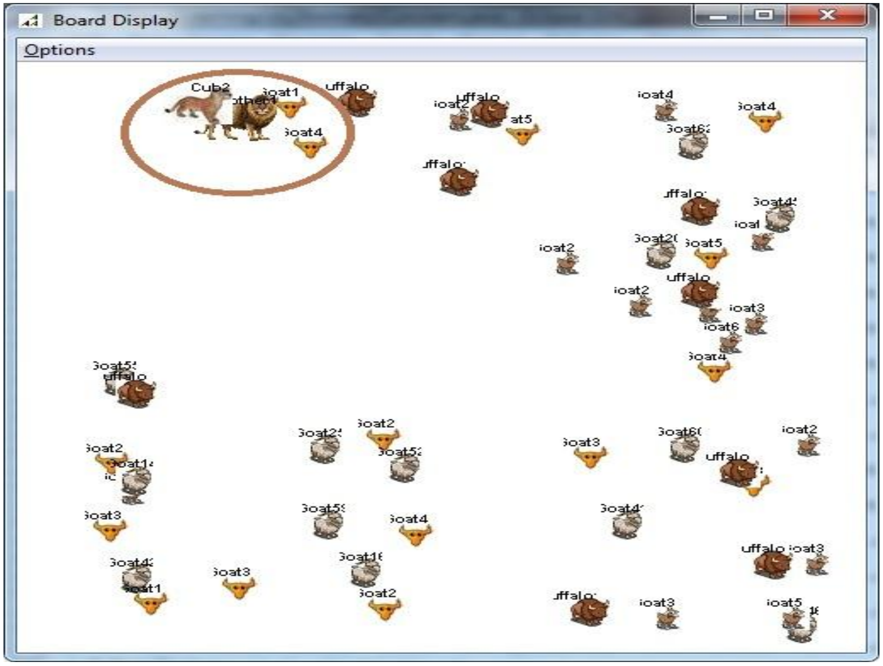
B. Classifier System - Rules and Actions:



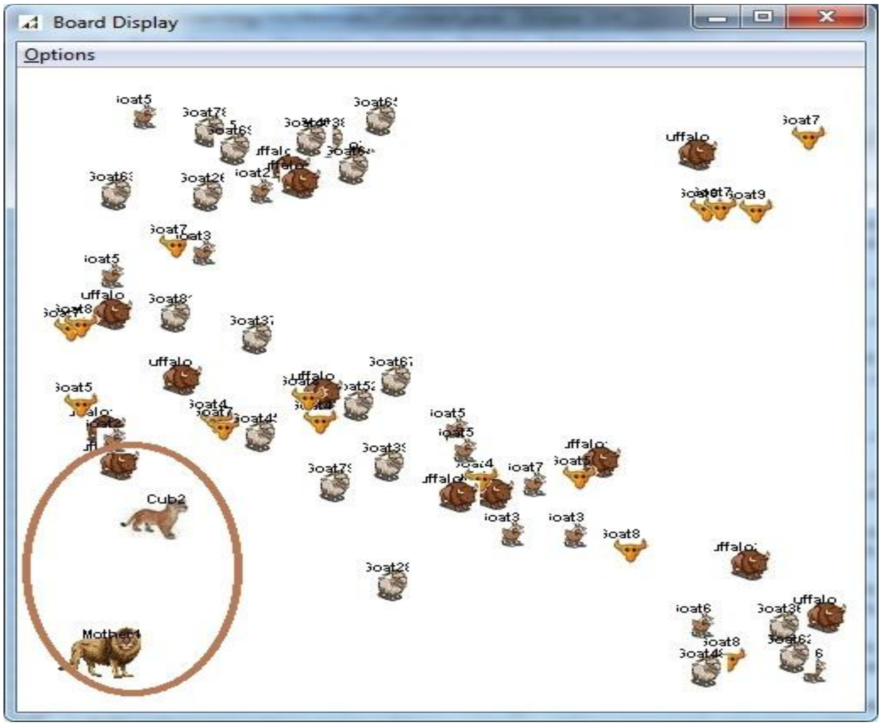
C. Results / Screenshots:

The following section contains screen shots of various scenarios that are encountered during our predator hunting.

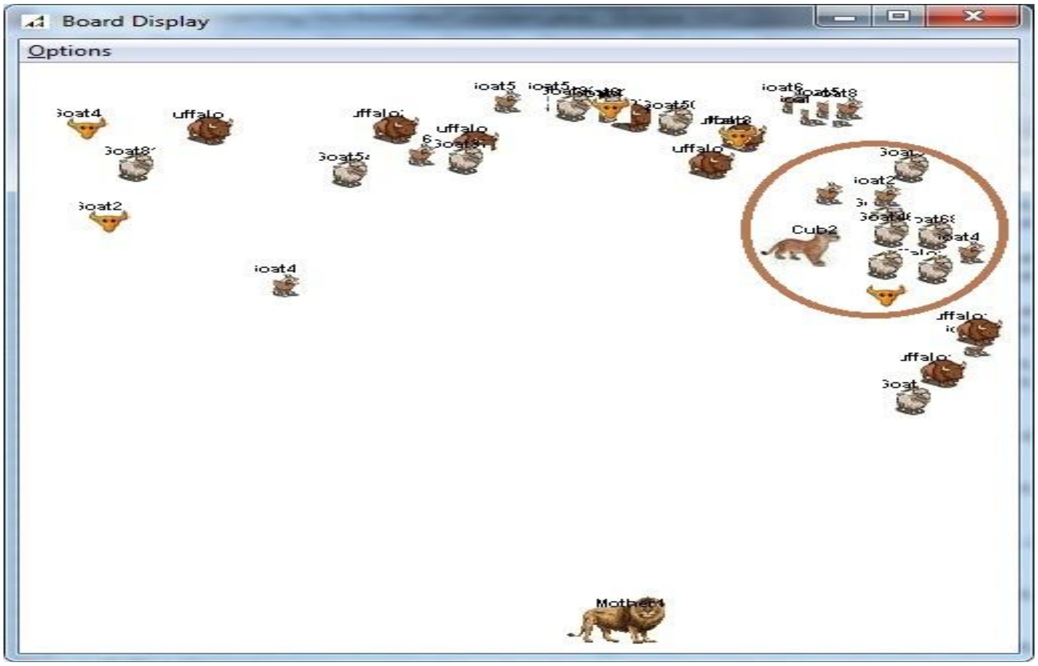
1. **Phase I: Cub observer the mother's hunting behavior and builds its neural network**



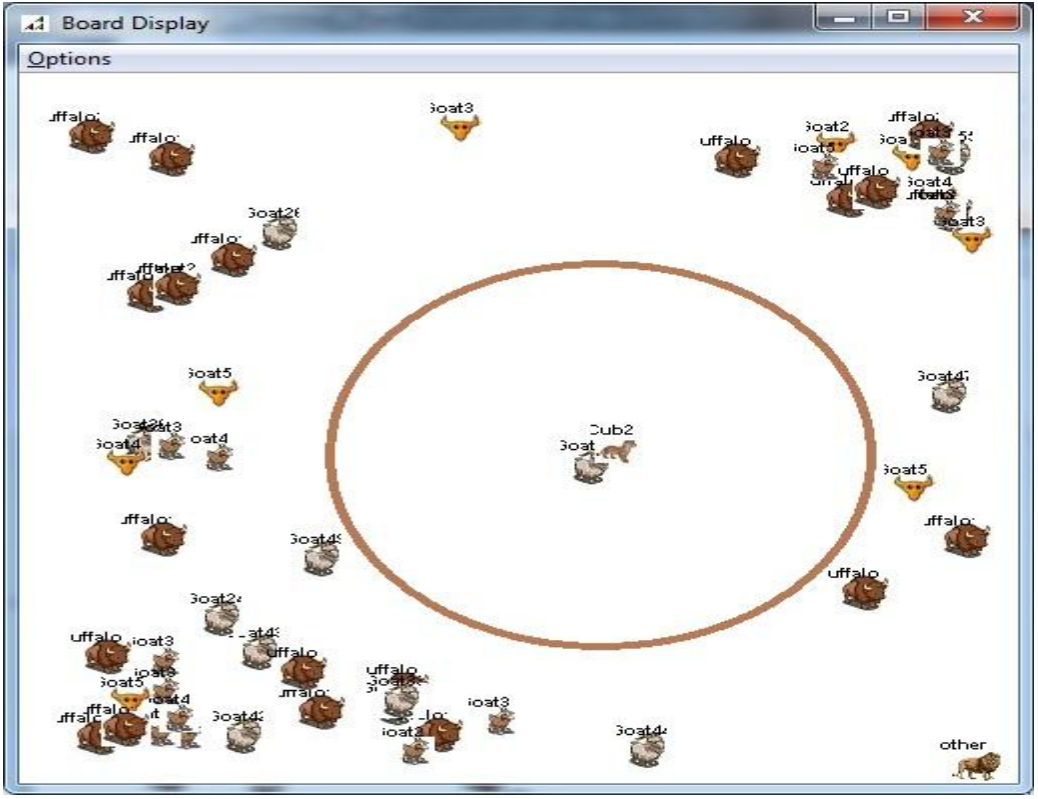
2. **Phase II: Mother observes the cub performing the hunt and gives reinforcement to the cub**



3. Phase III: Hunting Group of Prey – Self Learning



4. **Sense Range:** As each prey has a sense range, they tend to move away from the cub's sense range. This is observed in the following screenshot.



D. Sample Output Run:

Case 1:
In the following output snippet, the Cub performs the following actions for successful hunt of a Goat

- Search – 0.0
- Pounce – 0.4
- ClawChase – 0.5
- Claw – 0.6
- Eat – 0.7

```
Animat to Be Hunt: Goat; Distance: 0.12666352012619664; Energy: 100; Size: 15; Previous
Action: 0.0; Action To Be Taken0.0
Animat to Be Hunt: Goat; Distance: 0.12007815063537368; Energy: 100; Size: 15; Previous
Action: 0.4; Action To Be Taken0.4
Animat to Be Hunt: Goat; Distance: 0.12007815063537368; Energy: 100; Size: 15; Previous
Action: 0.5; Action To Be Taken0.5
Animat to Be Hunt: Goat; Distance: 0.07763814586933462; Energy: 55; Size: 15; Previous
Action: 0.6; Action To Be Taken0.6
Animat to Be Hunt: Goat; Distance: 0.09063297913161555; Energy: 10; Size: 15; Previous
Action: 0.7; Action To Be Taken0.7
```

Case 2:
In the following output snippet, the Cub performs the following actions for successful hunt of a Buffalo

- Search – 0.0
- Claw – 0.6
- Eat – 0.7

```
Animat to Be Hunt: Buffalo; Distance: 0.12135639555005293; Energy: 100; Size: 15; Previous
Action: 0.0; Action To Be Taken0.0
Animat to Be Hunt: Buffalo; Distance: 0.08340815083658593; Energy: 100; Size: 15; Previous
Action: 0.6; Action To Be Taken0.6
Animat to Be Hunt: Buffalo; Distance: 0.07447633568272734; Energy: 55; Size: 15; Previous
Action: 0.6; Action To Be Taken0.6
Animat to Be Hunt: Buffalo; Distance: 0.08340815083658593; Energy: 10; Size: 15; Previous
Action: 0.7; Action To Be Taken0.7
```

Case 3:

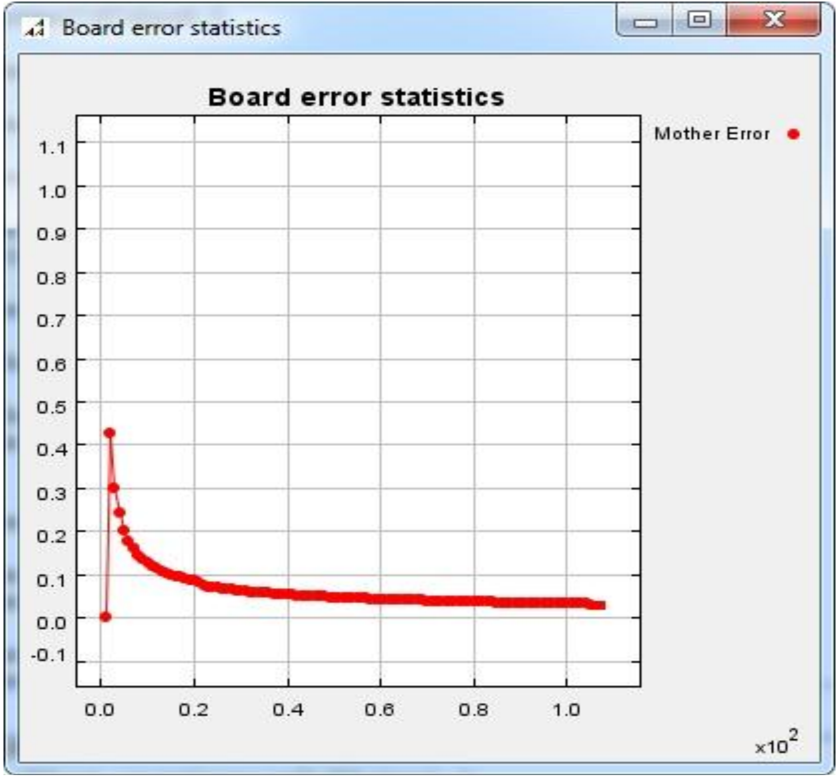
In the following output snippet, the Cub performs the following actions for successful hunt of a Buffalo

- Search – 0.0
- Pounce – 0.4
- ClawChase – 0.5
- Claw – 0.6
- Eat – 0.7

```
Animat to Be Hunt: Buffalo; Distance: 0.12450454844770316; Energy: 100; Size: 15; Previous
Action: 0.0; Action To Be Taken0.0
Animat to Be Hunt: Buffalo; Distance: 0.10590615112585099; Energy: 100; Size: 15; Previous
Action: 0.4; Action To Be Taken0.4
Animat to Be Hunt: Buffalo; Distance: 0.10590615112585099; Energy: 100; Size: 15; Previous
Action: 0.5; Action To Be Taken0.5
Animat to Be Hunt: Buffalo; Distance: 0.064632855578925; Energy: 55; Size: 15; Previous
Action: 0.6; Action To Be Taken0.6
Animat to Be Hunt: Buffalo; Distance: 0.07659657174517254; Energy: 10; Size: 15; Previous
Action: 0.7; Action To Be Taken0.7
```

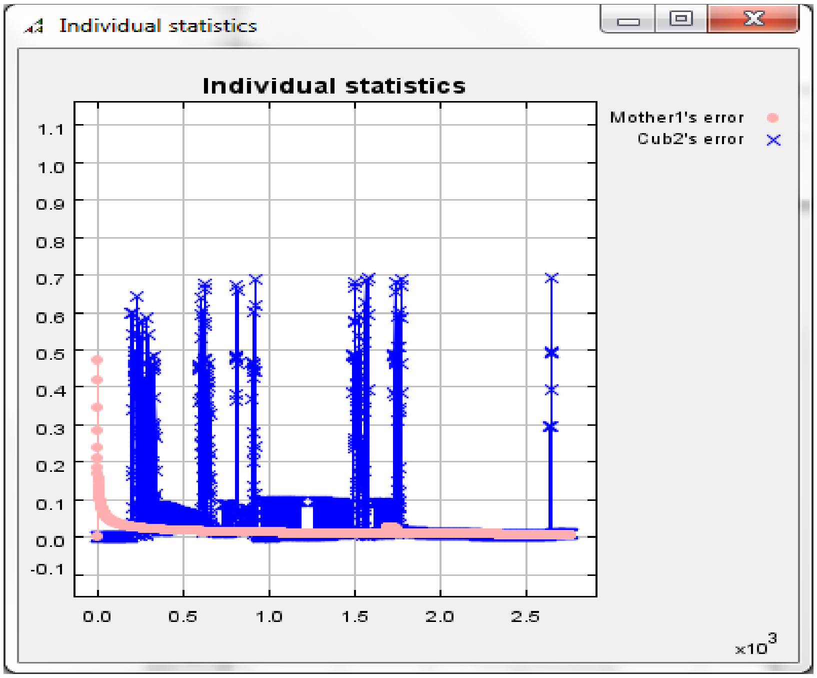
E. Experiments:

1. **Error rate analysis for Neural network training with mother inputs**

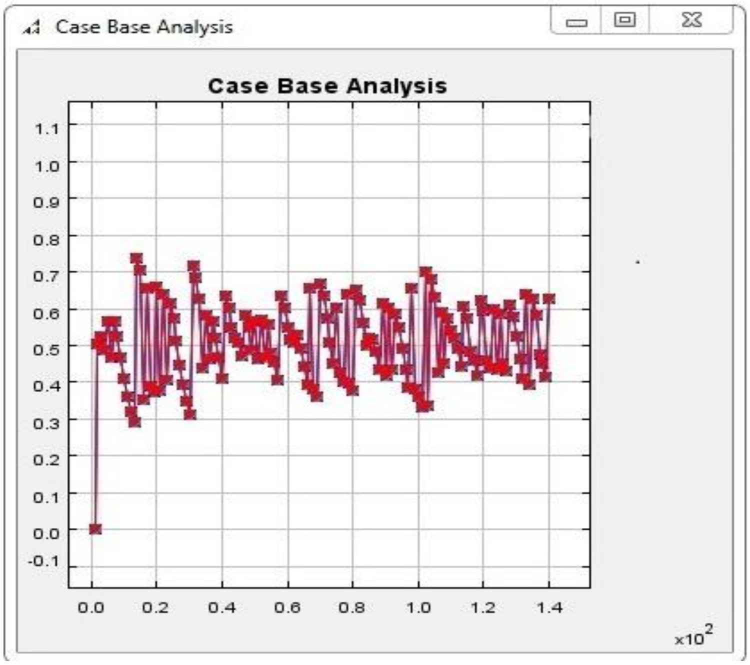


2. Error statistics analysis for mother and cub

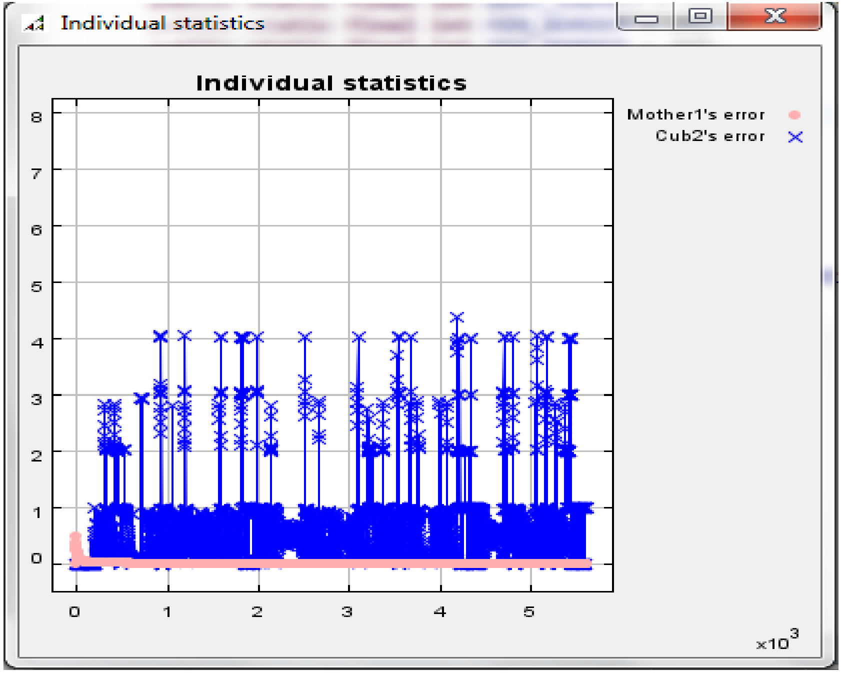
In the initial stage the mother trains the model which is later used by the cub, the error rate decreases after the cub has reinforcements from the mother and finally the groups of preys are introduced which again increases the error rate. But eventually the cub learns as seen in by the decrease in error rate.



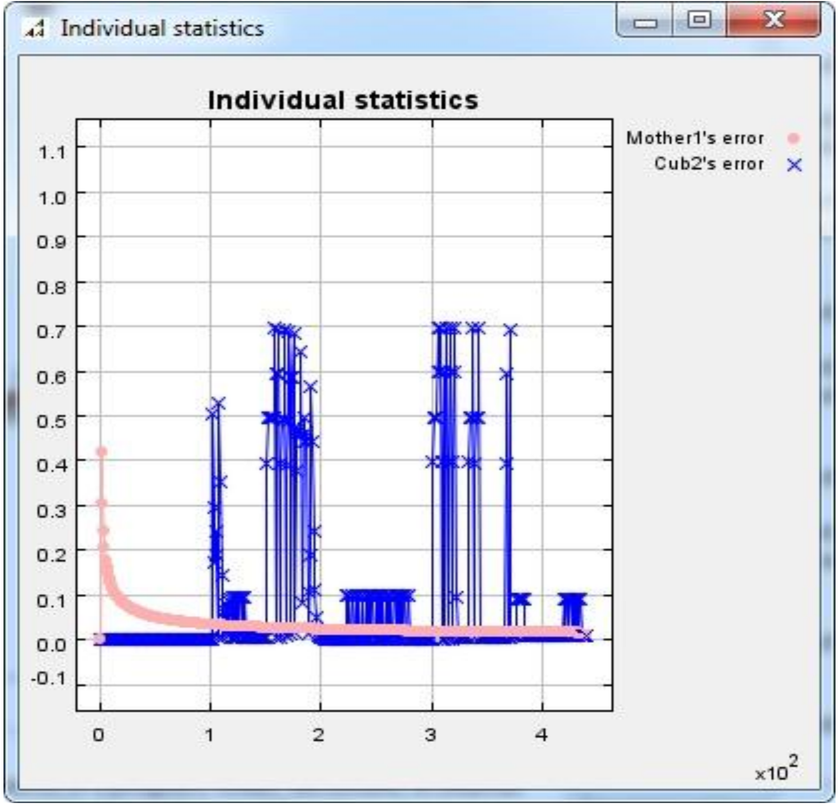
3. Error rate analysis of the Cub based on Case-Base Model



4. **Initial stages of learning while imitating the mother(one of the cases)**



5. **Overall Learning of the Cub**



As we could see the error probability of the cub finally converges to zero.

F. Source Code:
The source for various key functionalities are shown below along with comments to explain the code.

1. **Algorithm:** The algorithm to determine to direction of movement in a prey/predator is show below.

```
If Animat is a Prey,
        If Predator is present in vicinity.
                Find the location of the predator
                Find directions away from the predator
                If directions away from predator are available
                        Move in a direction away from the predator
                If no direction is available (in case of surrounding walls/border)
                        Move in a random available direction
        Move in a random available direction
        Scan for nearby predators and continue the process
If Animat is a Predator,
        If Prey is in vicinity
                Find the direction in which the prey is located
                Move in that direction approaching the prey
        If no Prey is present, move in a random direction
        Scan for nearby preys and continue the process
```

2. **Prey Movement:**

```
//If Animat is a Prey
if(objAnimatVisionSensorVisionSensor.animatType ==GOAT ||
objAnimatVisionSensorVisionSensor.animatType==BUFFALO)
{
//Scan for enemies in surrounding area
        ArrayList<Animat> enemyList = isEnemyPresent(objAnimatVisionSensor);
// If no enemies are present, find a random direction to move and move in that direction
        if(enemyList.isEmpty())
        {
                int direction=findRandomDirection(objAnimatVisionSensor);
                moveAnimat(objAnimatVisionSensor, direction, Constant.SPEED);
        }

        //If Predator is present in vicinity
        else
        {
                // Find available directions away from the predator and move in that direction
        ArrayList<Integer> direction = findMoveDirection(objAnimatVisionSensor,
enemyList);
        int x=0,dir=0;
        x = Random.uniform.nextIntFromTo(0,direction.size()-1);
        direction = direction.get(x);
        moveAnimat(objAnimatVisionSensor,  dir, Constant.SPEED+1);
    }
    enemyList.clear();
}
```

3. Predator Movement:

```
// If animat is a predator
if(objAnimatVisionSensor.animatType==LION  || objAnimatVisionSensor.animatType==CUB)
{
    // Check if enemy animat is present
    ArrayList<Animat> enemyList = isEnemyPresent(objAnimatVisionSensor);
    // If no enemy is present move in a random direction
    if(enemyList.isEmpty())
    {
        int direction=findRandomDirection(objAnimatVisionSensor);
        moveAnimat(objAnimatVisionSensor,  direction, Constant.SPEED);
    }
    // If prey present, choose direction of prey and move in that direction
    else
    {
        ArrayList<Integer> direction = findMoveDirection(objAnimatVisionSensor,al);
        int dir=0,x=0;
        do
        {
            x = Random.uniform.nextIntFromTo(0,direction.size()-1);
            direction = direction.get(x);
        }while(dir==5);
        moveAnimat(objAnimatVisionSensor,  direction, Constant.SPEED+3);

    }
    enemyList.clear();
}
```

4.Action Selection:

```
// This function is used to determine  the action a predator should perform based on various
conditions
public double determineAction(Animat  animatToBHunt, Animat animatHunting)
{
    // The default action that would be performed  is Search for Preys
    double actionToBTaken=ActionConstants.SEARCH;
    double distance =0;
    // If any prey is present nearby
    if(animatToBHunt!=null)
    {
        // Obtain distance of the prey from the predator
        distance = getNormalizeddistanceance(animatHunting, animatToBHunt);
        // The cub will drawback if there are more preys located closely
        if(animatHunting.animatType == 3 && Lion.DRAWBACK)
                actionToBTaken = ActionConstants.DRAWBACK;
        // If the prey is large sized and not closer to the predator, the cub performs Roar
    action
        else if(distance>.50 && distance<=.75 && animatToBHunt.size ==
Constant.largeSize) actionToBTaken = ActionConstants.ROAR;
        // If the prey is not closer to the predator, the predator Runs towards the prey
        else if(distance>=.50) actionToBTaken = ActionConstants.RUN;
        // If the prey is medium sized and closer to the predator, the cub performs Snarl
action
        else if(distance<=.50 && distance>=.25 && animatToBHunt.size ==
Constant.mediumSize) actionToBTaken = ActionConstants.SNARL;
        // If the prey is closer to the predator and if the previous action performed is
Snarl, then the predator Pounces on the prey
        else if(distance<.25 && distance>.10 && prevAction <= ActionConstants.SNARL)
actionToBTaken = ActionConstants.POUNCE;
        // If the energy of the energy of the prey is very less, the predator eats the prey
        else if(distance<.10 && animatToBHunt.energy<=10) actionToBTaken =
ActionConstants.EAT;
        // If the prey is not weak enough, the predator claws the predator to decrease its
energy level
        else if(distance<.10 && animatToBHunt.energy>10) actionToBTaken =
ActionConstants.CLAW;
        if(animatHunting.animatType==4 && Lion.DRAWBACK)
            actionToBTaken = ActionConstants.GROWL;
        prevAction = actionToBTaken;
    }
    else
        actionToBTaken = ActionConstants.SEARCH;
    return actionToBTaken;
}

// The predator runs towards prey at a greater speed
// This decreases the energy of the prey considerably
public void run(Animat predator, Animat prey)
```

```
{
        predator.energy -=4;
        board.moveTowards(predator,prey);
}

// The predator moves in random to find available preys
public void search(Animat predator)
{
        board.moveFarther(predator);
}

// Snarl action reduces the energy of the prey and also the predator closes on the prey
public void snarl(Animat predator, Animat prey)
{
        predator.energy -=1;
        prey.energy -= predator.energy/5;
        board.moveTowards(predator, prey);
}

// Roar action reduces the energy of the prey and the predator moves towards the prey
public void roar(Animat predator, Animat prey)
{
        predator.energy -=2;
        prey.energy -=predator.energy/4;
        board.moveTowards(predator, prey);
}

// Claw action considerably decreases the energy of the prey.
public void claw(Animat predator, Animat prey)
{
        predator.energy -=2;
        prey.energy -= predator.energy/3;
}

// Pounce actions moves the predator closer to the prey
public void pounce(Animat predator, Animat prey)
{
        predator.energy -=3;
        prey.energy -= predator.energy/2;
        board.moveTowards(predator,prey);
}

// Eat action kills the prey and the prey is removed from the board. Energy of the prey is gained
by they predator
```

```
public void eat(Animat predator, Animat prey)
{
    prey.energy = 0;
    if(prey.animatType == 1)
    {
        ((Buffalo)prey).killBuffalo();
        predator.energy += Constant.BUFFALO_ENERGY;
    }
    else if(prey.animatType == 2)
    {
        ((Goat)prey).killGoats();
        predator.energy += Constant.GOAT_ENERGY;
    }
    board.emptyAnimat();
}

// The cub will drawback and move far from the prey if there are more preys located closeby
public void drawback(Animat predator, Animat prey)
{
    predator.energy -=5;
    board.moveFarther(predator);
}
```

5. Scan function:

```
// Scan area for Lion
// Scan for nearby prey within its Range
public ArrayList<Animat> scanArea(int RANGE)
{
    // HashMap is used to sort the preys based on distances and energy
    // Closest prey with lowest energy would be choosen
    HashMap<Double, Animat> hmRange = new HashMap<Double, Animat>();
    ArrayList<Animat> alBuffalo = new ArrayList<Animat>();
    ArrayList<Animat> alGoats = new ArrayList<Animat>();
    if(!alRange.isEmpty())
        alRange.clear();
    // Go through the available goats to find the nearby goats
    for (Iterator iter = board.getGoats().iterator(); iter.hasNext();)
    {
        Animat animat = (Animat) iter.next();
        if(animat.getX()>=this.getX()-RANGE && animat.getX()<=this.getX()+ RANGE)
        {
            if(animat.getY()>=this.getY()-RANGE && animat.getY()<=this.getY()+
RANGE)
```

```
            {
                hmRange.put(getMetric(animat.getEnergy(),
board.getDistance(this, animat)), animat);
                alGoats.add(animat);
            }
        }
    }

    // Go through the available buffalo to find the nearby goats
    for (Iterator iter = board.getBuffalos().iterator(); iter.hasNext();)
    {
        Animat animat = (Animat) iter.next();
        if(animat.getX()>=this.getX()-RANGE && animat.getX()<=this.getX()+ RANGE)
        {
            if(animat.getY()>=this.getY()-RANGE && animat.getY()<=this.getY()+
RANGE)
            {
                hmRange.put(getMetric(animat.getEnergy(),
board.getDistance(this, animat)), animat);
                alBuffalo.add(animat);
            }
        }
    }

    // The values added to the hashmap are sorted based on the key (Distance & Energy)
    // This gives the nearest prey with least energy
    TreeSet<Double> keys = new TreeSet<Double>(hmRange.keySet());
    for(double key: keys)
    {
        alRange.add(hmRange.get(key));
    }

    //To Check if the Prey are in a Group
        if(this.animatType==3 && alGoats.size() >3)
        {
            DRAWBACK = true;
            alRange.clear();
        }

        //To Check if the Prey are in a Group
        if(this.animatType==3 && alBuffalo.size() >3)
        {
            DRAWBACK = true;
            alRange.clear();
        }

    return alRange;
}
```

6. Scan for animals near Buffalo

```
public ArrayList<Animat> scanArea(int RANGE)
{
    if(!alRange.isEmpty())
        alRange.clear();
    // Get nearby buffalos in range
    for (Iterator iter = board.getBuffalos().iterator(); iter.hasNext();)
    {
        Buffalo buffalo= (Buffalo) iter.next();
        if(buffalo.getX()>=this.getX()-RANGE && buffalo.getX()<=this.getX()+ RANGE)
        {
            if(buffalo.getY()>=this.getY()-RANGE && buffalo.getY()<=this.getY()+
RANGE)
                alRange.add(buffalo);
        }
    }

    //If more number of buffalos are present nearby, the buffolos would not be bothered
about the presence of lion
    if(alRange.size() >=5)
    {
        alRange.clear();
        return alRange;
    }

    // Get nearby goats in range
    for (Iterator iter = board.getLions().iterator(); iter.hasNext();)
    {
        Lion lion= (Lion) iter.next();
        if(lion.getX()>=this.getX()-RANGE && lion.getX()<=this.getX()+ RANGE)
        {
            if(lion.getY()>=this.getY()-RANGE && lion.getY()<=this.getY()+ RANGE)
                alRange.add(lion);
        }
    }
    return alRange;
}
```

7. Neural Network:

```
// This method is trains the Neural network
public synchronized void train() throws NeuralException
{
    // Get the animat that has to be Hunt
    animatToBHunt = board.getAnimatToHunt(getCurrentLion());
    this.net.getNet().getMonitor().setLearningRate(.8);
    this.net.getNet().getMonitor().setMomentum(0.8);
    // Determine the action that the cub should take based on various input  parameters
    if(animatToBHunt != null)
        actionShouldveBeen = board.determineAction(animatToBHunt,
getCurrentLion());
    else
        actionShouldveBeen = ActionConstants.SEARCH;
    // Compute the network's error
    System.out.println("Action  Should Have Been: "+actionShouldveBeen+";  Retrieved
Value: "+retrievedValue);
    Lion animal = getCurrentLion();
    double err = Math.abs(actionShouldveBeen  - retrievedValue);
    animal.setError(err);
    // Get the object that watches over the training
    Monitor monitor = net.getNet().getMonitor();

    // Set the monitor parameters
    monitor.setTrainingPatterns(1);
    monitor.setTotCicles(1);

    // Setup the inputs for the next round of training
    this.desiredNetworkOutput
    .setInputArray(new double[][] { { actionShouldveBeen } });

    this.inputForTraining.setInputArray(properInput());

    // Now actually train the network
    try {
        this.net.train(inputForTraining);
    } catch (NeuralException ex) {
        SimUtilities.showError("Error  training neural network  for agent \""
                + "\"", ex);
        throw ex;
    }
}

// The age of the Cub is considered for the cub to start training its neural network
```

```
public void preStep() throws NeuralException
{
        // If the cub has attained certain age, the cub starts self learning
        if(this.age > Constant.CUBAGE)
                self = true;
        if(self)
                train();
        // reset for this step
        this.wasScolded = false;
}

@Override
// The step methods gets called repeatedly
public void step() throws NeuralException{
        if(this.age > Constant.CUBAGE)
        {
                self = true;
                board.getCub().setRange(Constant.LIONRANGE);
        }
                        try {
                                this.retrievedValue = retrieve();
                                setActionPerformed();
                        } catch (NeuralException ex) {
                                SimUtilities.showError(
                                                "Error computing the next action to perform. \n"
                                                                + "Agent \"" + "\".", ex);
                                throw ex;
                        }
}

@Override
public void postStep() throws Exception{
        // The age of the cub is determined by the tickCount of the Repast model
        this.age = (int)NeuralModel.model.getTickCount();
        if(self){
                //move cub based on action taken
                 actionPerformed = (int) getActionPerformed();
                // Positive reinforcement is given to the cub if it performs the action that should
be performed
                if (Math.abs(actionShouldveBeen - actionPerformed)<=epsilon)
                {
                        praise();
                        doAction();
                }
                // Negative reinforcement is given to the cub if it does not perform the action
that should be performed
                else
                {
                        scold();
                        animatToBHunt = board.getAnimatToHunt(getCurrentLion());
                        if(animatToBHunt != null)
                                actionShouldveBeen = board.determineAction(animatToBHunt,
getCurrentLion());
                        else
                                actionShouldveBeen = ActionConstants.SEARCH;
                        doAction();
                }

        }

        //Observation learning (Following the mother) is carried out for the cub if it has not
reached the age
        else{
                Mother mother = (Mother) board.getMother();
                double x = mother.getX();
                double y = mother.getY();
                this.setX(x - 10);
                this.setY(y - 10);
                this.animatX = x-10;
                this.animatY = y-10;
                 actionPerformed = (int) getActionPerformed();
                animatToBHunt = board.getAnimatToHunt(getCurrentLion());
                if(animatToBHunt != null)
                        actionShouldveBeen = board.determineAction(animatToBHunt,
getCurrentLion());
                else
                        actionShouldveBeen = ActionConstants.SEARCH;
                doAction();
        }
}

// Cub starts performing the hunt only when it has reached a certain age.
private Lion getCurrentLion(){
        if(this.age > Constant.CUBAGE){
                return board.getCub();
        }else return board.getMother();
}
```

```
// The function returns the inputs that has to be fed into the neural network
protected double[][] properInput(){
        double nSize[][] = new double[1][6];
        Animat lion = getCurrentLion();
        if(animatToBHunt != null)
        {
                nSize[0][0] = animatToBHunt.getPosX()+animatToBHunt.getPosY();
                nSize[0][1] = animatToBHunt.getSize();
                nSize[0][2] = animatToBHunt.getAnimatType();
                nSize[0][3] = animatToBHunt.getEnergy();
        }
        else
        {
                nSize[0][0] = 0.0;
                nSize[0][1] = 0.0;
                nSize[0][2] = 0.0;
                nSize[0][3] = 0.0;
        }
        nSize[0][4] = lion.getPosX()+ lion.getPosY();
        nSize[0][5] = lion.getEnergy();
        return nSize;
}

// This function is used for the cub's step function in determining the action that should be
performed
protected double[] getInputs(){
        double nSize[] = new double[6];
        if(animatToBHunt!=null)
        {
                nSize[0] = animatToBHunt.getPosX()+animatToBHunt.getPosY();
                nSize[1] = animatToBHunt.getSize();
                nSize[2] = animatToBHunt.getAnimatType();
                nSize[3] = animatToBHunt.getEnergy();
        }
        else
        {
                nSize[0] = 0.0;
                nSize[1] = 0.0;
                nSize[2] = 0.0;
                nSize[3] = 0.0;
        }
        Animat cub = getCurrentLion();
        nSize[4] = cub.getPosX()+ cub.getPosY();
        nSize[5] = cub.getEnergy();
        return nSize;
}

//Called only by cub during its imitationNN build
public Animat getAnimatToHunt(Lion cub){
        if(cub.animatToHunt == null)
                cub.doHunt(cub);
        return cub.animatToHunt;
}
```

8. Case Base:

```
// Function to build case bases for learning the
public void createCaseBases(Animat animatToHunt, Animat animatHunting)
{
        AgentInputs av1 = new AgentInputs();
        SensoryItem f1, f2;
        f1 = new SensoryItem("AnimatHunting",animatHunting);
        f2 = new SensoryItem("AnimatToHunt",animatToHunt);
        av1.addSensoryItem(f1);
        av1.addSensoryItem(f2);
        List<AgentAction> act1 = new ArrayList<AgentAction>();
        double actionToBTaken=board.determineAction(animatToHunt,  animatHunting);
        AgentAction a1 = new AgentAction(actionToBTaken);
        act1.add(a1);
        Case c1 = new Case(av1,act1);
        this.caseBase.addCase(c1);
}

public CaseBase getCaseBase()
{
        return this.caseBase;
}
```