CHAPTER 1

Introduction to AI Techniques

Considerable research activities have been reported in the literature on designing knowledge-based systems. These efforts reflect the intuitive truth that *Intelligence* should be associated with *Learning*. An example is a baby that arrives in the world with a built in system that allows him to learn to acquire knowledge through experience. Some machines are now being built with the same principle of knowledge acquisition, using inspiration drawn from Nature.

Machine learning is concerned with computer programs that automatically improve their performance through experience. Research in machine learning has focused on developing approaches for machines to automatically develop strategies and algorithms for solving many types of problems. Some of these problems are data classification, data mining, automatic programming, control theory, and planning. Planning for autonomous systems that act in a particular environment must be able to integrate three basic behaviors: planning, execution, and learning. Planning involves describing a set of actions that will allow the autonomous system to achieve its goal. Execution deals with the interaction with the environment by application of planned actions and observing the resulting effects. Learning is needed to predict the responses of the environment to the system to achieve its goal.

Autonomous intelligent behavior is an area with an emerging interest within artificial intelligent researchers. It integrates many areas, such as robotics, planning, and machine learning. With respect to learning, autonomous systems must generate theories of how their environment reacts to their actions, and how the actions affect the environment. Muti-Agent Systems (MAS) deal with behavior management in collections of several independent entities, or agents.

There are many applications that embody autonomous intelligent behavior. Focusing on autonomous robots, we find several research applications that are trying to model the behavior of multiple autonomous robots trying to interact in a domain. The dog and sheep problem, the pall bearer problem, and multi-joint coordination are considered practical examples of robots that interact in a domain to achieve a common goal [2].

Among these applications we select the dog and sheep problem as an application to be studied deeply. In this problem it requires a simulated mobile dog to herd a simulated mobile sheep into a pen. When extended to multiple dogs and sheep, entirely different tactics must be learned by dogs acting in cooperation. If dogs perform as they would in isolation, their collective performance would be ineffective in herding several sheep. We will spot light some techniques that are useful tools for the proposed work. These techniques are artificial neural networks (ANNs), artificial immune systems (AIS), and multi-agent systems (MAS).

1. Artificial Neural Networks

The interest in neural networks has grown rapidly over the past few years. Diverse fields as engineering, philosophy, physiology, and psychology are integrated by the potential offered by this technology. Professionals in these areas are seeking applications within their discipline. This resurgence of interest has shown both theoretical and application successes. Suddenly, it appears possible to apply computation to realms that were previously restricted to human intelligence; making machines learn and remember in ways that bear a striking resemblance to human processes to attains these goals, and to give a new and significant meaning to Artificial Intelligence [1], and [2].

1.1 Characteristics of Artificial Neural Networks

Artificial neural networks are biologically inspired; that is they are composed of elements that perform in a manner analogous to the elementary function of the biological neuron. These elements are then organized in a way that may be related to the anatomy of the brain. Artificial neural networks exhibit a surprising number of the brain's characteristics. For example, they learn from experience, generalize from previous examples to new, and abstract essential characteristics from inputs containing irrelevant data [2].

1.1.1 Learning

Artificial neural networks can modify their behavior in response to their environment. This factor, more than any other, is responsible for the interest they have received. Given a set of inputs, perhaps with desired outputs, they self-adjust to produce consistent response. A wide variety of training algorithms have been developed to train different types of networks in order to optimize their outputs.

1.1.2 Generalization

Once trained, a network's response can be insensitive to minor variations in its input. This ability to recognize input patterns through noise and distortion is vital for pattern recognition in a real world environment. The structure of a neural network generalizes so that it can deal with imperfect problems resulting in near optimum solutions.

1.1.3 Abstraction

Some networks are capable of abstracting the essence of a set of inputs. For example, a network can be trained on a sequence of distorted versions of the letter "A". After adequate training, application of such a distorted example will cause the network to produce a perfectly formed letter. So, it has learned to produce something that it has never seen before.

1.2 The Artificial Neuron

The artificial neuron was designed to mimic the first order characteristics of the biological neuron. In essence, a set of inputs is applied, each representing the output of another neuron. Each input is multiplied by its corresponding weight, analogous to a synaptic strength, and all of the weighted inputs are then summed to determine the activation level of the neuron. A set of inputs labeled $\mathbf{x_1}$, $\mathbf{x_2}$, ..., and $\mathbf{x_n}$ are applied to the artificial neuron. These inputs, collectively referred to as the vector **X**, correspond to the signal into the synapses of a biological neuron. Each signal is multiplied by associated weights $\mathbf{w_1}$, $\mathbf{w_2}$, …, and $\mathbf{w_n}$ before it is applied to the summation block. These weights, collectively, referred to as vector W. Each weight corresponds to the strength of a single biological synaptic connection, see Figure1.1. The output summation, NET, is further processed by an activation function F to produce the neuron output OUT [3].

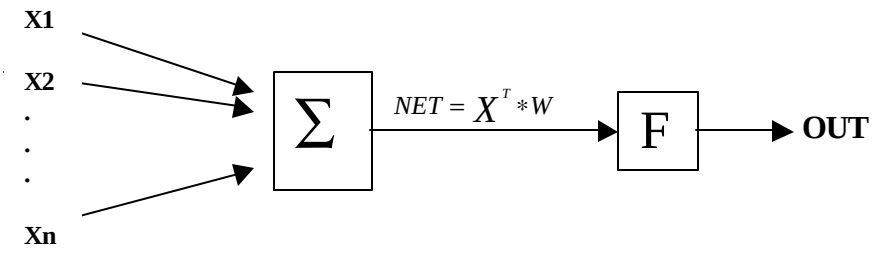

Figure1.1 The Artificial Neuron

The activation function F might be a simple linear function:

$$OUT = K * F(NET) \qquad (1.1)$$

Where K is a constant, and the output OUT is a threshold function:

OUT=1 if NET>T,

OUT=0 otherwise, where T is a constant threshold value.

The activation function F might be a non-linear function too; if the function compresses the range of NET, so that OUT never exceeds some low limits regardless the value of

NET, the this function is called a squashing function (1.2). The squashing function if often chosen to be a logistic function, or "sigmoid", meaning S-shaped, as seen in Figure1.2. This function is expresses mathematically by:

$$OUT = \frac{1}{1 + e^{-NET}} \qquad (1.2)$$

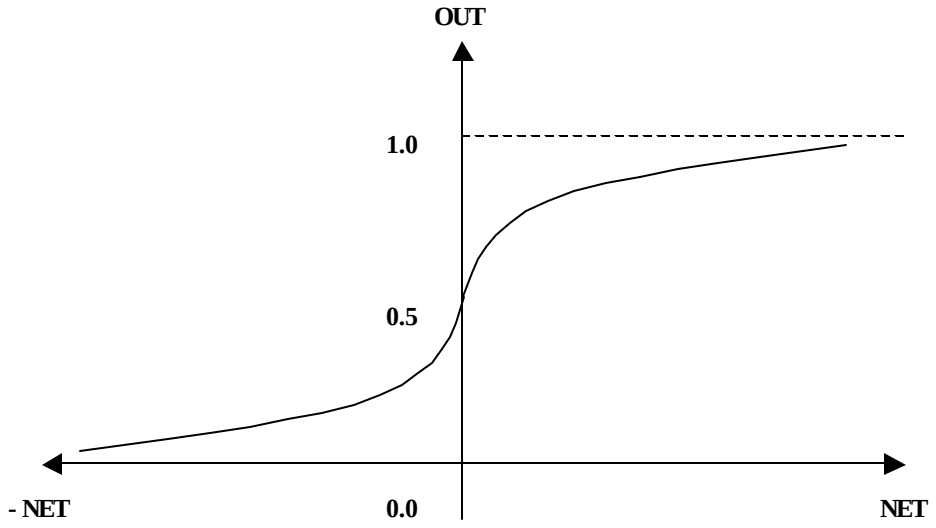

Figure1.2 The Sigmoidal Logistic Function

This non-linear gain characteristic solves the noise saturation dilemma. In addition to that the sigmoid function has other advantages such as:

- The squashing function has a simple derivative

$$F = OUT\,(1-OUT) \qquad (1.3)$$

- Mulit-layer networks have greater representational power than single layer nets if non-linearity is introduced.

- Automatic gain control, so large signals will be accommodated without saturation, while small signals are allowed to pass through without excessive attenuation.

1.3 Single-Layer And Multi-Layer Artificial Neural Networks

Although a single neuron can perform simple pattern detection function, the power of neural computation comes from connecting neurons into networks. The simplest network is a group of neurons arranged in a layer as shown in Figure1.3. The circular nodes on the left side serve only to distribute the inputs; they perform no computation and hence will not be considered to constitute a layer. The set of inputs X has each of its elements connected to each artificial neuron through a separate weight. It is convenient to consider the weights, of any layer, to be elements of a matrix W. The dimension of the matrix are *m* rows by *n* columns, where *m* is the number of inputs and *n* the number of neurons. In this way it may be seen that calculating the set of neuron outputs Y for a layer is a simple matrix multiplication, Thus Y=XW.

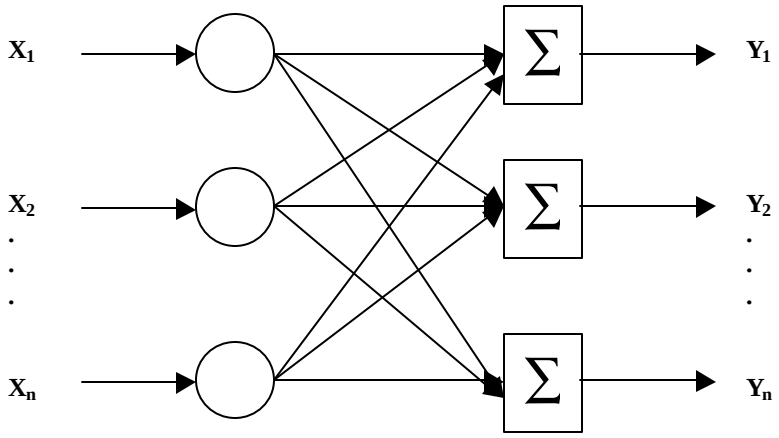

Figure1.3 Single Layer Neural Network

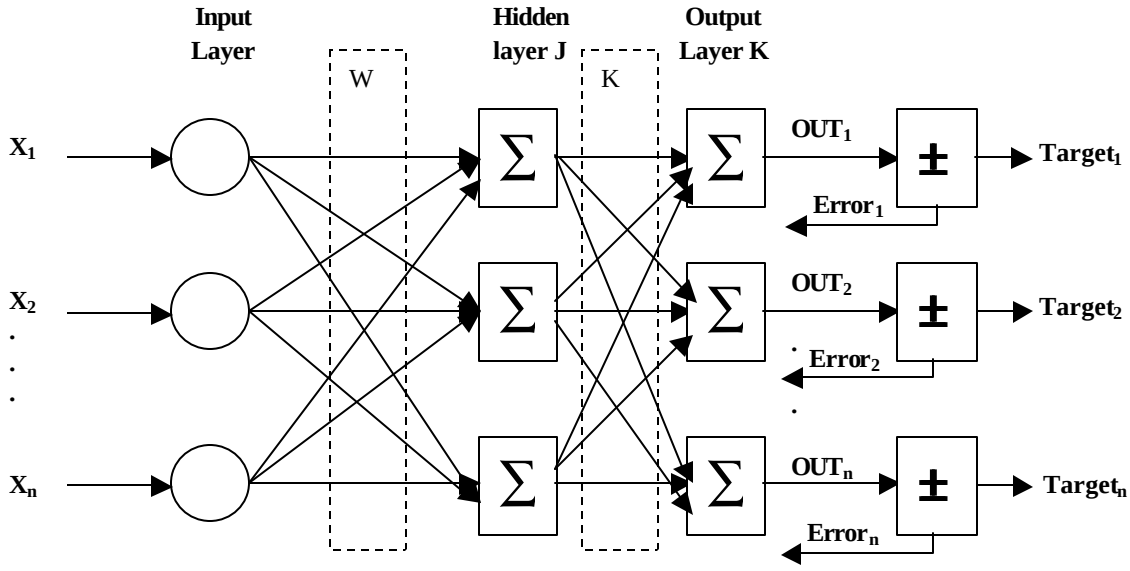

Figure1.4 Multi-layer Neural Networks

Figure1.4 shows that a multilayer network consists of alternating sets of neurons and weight matrices W, and K. As previously discussed, the input layer does no summation or computation, it is only a fan out stage. Therefore, a layer consists of a set of weights and the subsequent neurons that sum and squash the signals.

1.4 Training of Artificial Neural Networks

A network is trained so that application of a set of inputs produces the desired set of outputs. Each input, or output, set is referred to as a vector. Training is accomplished by sequentially applying a predetermined procedure. During training, the network weights gradually converge to a value such that each input vector produces the desired output vector. We have two main categories of training algorithms, and they are categorized as supervised and unsupervised training algorithms [4].

1.4.1 Supervised Training

Supervised training requires pairing of input with the desired output; together these are called a training pair. Usually, a network is trained over a number of such training pairs. When an input vector is applied, the output of the network is calculated and

compared with the corresponding target vector. The error *(Target - OUT)* is fed back through the network and weights are changed according to an algorithm that minimizes this error. The vectors of the training set are applied sequentially, and errors are calculated and weights adjusted for each vector, until the error for the entire training set is at an acceptably low level. Supervised training is the algorithm used in this research.

1.4.2 Unsupervised Training

Unsupervised training is a far more plausible model of learning in the biological system. Hence, no comparisons to predetermined ideal responses of the training set exists. The training algorithm modifies network weights to produce output vectors. Application of one of the training vectors, or a vector that is sufficiently similar to it, will produce the same pattern of outputs. The training process, therefore, extracts the statistical properties of the training set and groups similar vectors into classes.

1.5 Network Learning

The artificial neural network's learning ability represents the most intriguing property. Like the biological systems, these networks modify themselves as a result of experience to produce a more desirable behavior pattern. Learning can be either supervised or unsupervised. Supervised learning requires an external "teacher" that evaluates the behavior of the system and directs the subsequent modifications. Unsupervised learning requires no teacher; the network organizes itself to produce the desired changes. Thus, the act of adjusting the weights is commonly called "training", and the network is said to "learn".

As an example of supervised training we have the PERCEPTION structure, as seen in Figure1.5. Rosenblatt proves the PERCEPTRON learning theorem, and this proof demonstrated that a PERCEPTRON could learn any thing it could represent. It is important to distinguish between representation and learning. Representation refers to the ability of a PERCEPTRON to simulate a specified function.

Learning requires the existence of a systematic procedure for adjusting the network weights to produce that function. The training method of PERCEPTRON can be summarized as follows:

Step] Apply an input pattern X and calculate OUT.

Step2 (a) If the calculated actual output (OUT) is equal to the desired output, then go to step 1 and apply another pattern.

(b) If the calculated actual output is different from the desired output, the weight adjustment must be made in the direction that minimizes that difference.

Step3 Go to step 1, and apply another pattern.

1.6 The Delta Rule

An important generalization of the PERCEPTRON training algorithm is called the Delta rule. To see how this was developed, note that step 2 of PERCEPTRON training algorithm may be tested and generalized by introducing the term δ, which is the difference between the desired, or target output T, and the actual output A.

$$\delta = (T-A) \qquad (1.4)$$

The PERCEPTRON training algorithm is satisfied if δ is multiplied by the value of each input x_i and this product is added to the corresponding weight. To generalize this, a "learning rate" coefficient η is multiplied by the δx_i product to allow control of the average size of weight changes.

$$\Delta_I=\eta\delta x_I \qquad (1.5)$$

$$W_I(n+1)=W_I(n)+ \Delta_I \qquad (1.6)$$

Where,

Δ_I = the correction associated with the ith input x_i.

W_i (n+1) = the value of weight i after adjustment (at step n+1).

W_i (n) = the value of weight i before adjustment (at step n).

n= the number of steps required to coverge.

The delta rule modifies weighs appropriately for target and actual output of either polarity and for both continuous and binary inputs and outputs. Keep in mind that there is no predetermined information about how many steps will be required to train the network.

1.7 Backpropagation Networks

The invention of the backpropagation algorithm has played an important role in the resurgence of interest in artificial neural networks. Backpropagation is a systematic method for training multilayer artificial neural networks. It is also a dramatically expanded range of problems to which artificial neural networks can be applied, and it has generated many successful demonstrations of its power. It is considered being a valuable tool in classification problems.

1.7.1 Network Configuration

Figure 1.7 shows the neuron used as the fundamental building block for backpropagation networks with F as a sigmoid function. A set of inputs is applied either from the outside or from a previous layer. Each of these is multiplied by a weight, and the products are summed. This summation of product is termed NET and must be calculated for each neuron in the network. After NET is calculated, an activation function F is applied to modify it, thereby producing the signal OUT. The activation function usually used for backpropagation is called the sigmoid function. As mentioned before, the sigmoid function is desirable in that it has a simple derivative, and can expressed as follows:

$$\partial OUT/\partial NET=OUT(l-OUT) \qquad (1.7)$$

Figure1.4 shows a multilayer network suitable for training with backpropagation. The first set of neurons, connected to the input, which serves only as distribution points; they perform no input summation. The input signal is simply passed through to the weights on their outputs. Each neuron in subsequent layer produces NET and OUT signals as described before.

1.7.2 Network Training

The objective of training the network is to adjust the weights so that the application of inputs produces the desired set of outputs. These input-output sets can be referred to as vectors. Training assumes that each input vector is paired with a target vector representing the desired output; together they are called a training pair. Usually, a network is trained over a number of training pairs. This group of training pairs is called the training set. Before starting the training process, all of the weights must be initialized to small random numbers. This ensures that the network is not saturated by large values of the weights. Training the backpropagation network requires the following steps;

Step] Select the next training pair from the training set; apply the input vector to the network input.

Step2 Calculate the output of the network (actual output).

Step3 Calculate the error between the network output and the desired output (the target vector from the training pair).

Step4 Adjust the weights of the network in a way that minimizes the error.

Step5 Repeat steps 1 through 4 for each vector in the training set until the error for the entire set is acceptably low.

Calculations are performed on a layer-by-layer basis. For example, referring to Figure *1.4*, the outputs of neurons in the first hidden layer are calculated. Then, these outputs are applied as inputs to the second hidden layer. The outputs of the second hidden layer are calculated and these outputs constitute the network output vector. In step 3, each of the network outputs, labeled OUT in Figure1.4, is subtracted from its corresponding component of the target vector to produce an error. This error is used in step 4 to adjust the weights of the network, where the polarity and magnitude of the weight changes are determined by the training algorithm.

After enough repetitions of these four steps, the error between actual outputs and target outputs should be reduced to an acceptable value, and the network is said to be trained.In the previous training algorithm, step 1 and 2 constitute a “forward pass” such

that the signal propagates from the network input to its output. Step 3 and 4 are a "reverse pass"; here, the calculated error signal propagates backward through the network where it is used to adjust weights. These two passes are now expanded and expressed in more mathematical form.

1.7.2.1 Forward Pass

Step 1 and 2 can be expressed in vector form as follows: an input vector X is applied and an output vector Y is produced. The input target vector pair (X and T) comes from the training set. The calculation is performed on X to produce the output vector Y. As we have seen, calculations in multilayer networks are done layer by layer, starting with the layer nearest to the input. The NET value of each neuron in the first layer is calculated as the weighted sum of its neuron's inputs. The activation function then squashes *NET* to produce the output value *OUT* value for each neuron in that layer. Once, the set of outputs for a layer is found, it serves as input to the next layer.

The process is repeated layer by layer, until the final set of network outputs is produced. This process can be stated in vector notation. The weights among neurons can be considered to be a matrix W. For example, the weight from neuron 8 in layer 2 to neuron *5* in layer 3 is designated $W_{8,5}$ Rather than using the summation of products, the NET vector for a layer N may be expressed as the product of X and W; In vector notation $N=X^TW$. The application of function F to the net of vector N, component by component, produces the output vector O. Thus, for a given layer, the following expression describes calculation process.

$$O=F(X^TW) \qquad (1.8)$$

The output vector for one layer is the input vector for the next, so calculating outputs of the final layer requires the application of the equation (1.8) in each layer, from the network's input to its output.

1.7.2.2 Reverse Pass

1.7.2.2.1 Output Layer Weights

This adjustment is easily accomplished because target values are available for each neuron in the output layer. The use of the Delta rule, as described before, will solve this modification problem. Interior layers are referred to as "hidden layers", as their outputs have no target values for comparison; hence training is more complicated.

To give a mathematical interpretation for the modification process, consider the process of training a single weight from neuron P in the hidden layer j to neuron q in the output layer k. The output of a neuron q in layer K is subtracted from its target value to produce an error signal. This signal is multiplied by the derivative of the function calculated for neuron q, thereby producing the backprobagation δ values.

$$\delta = OUT\ (1- OUT)\ (Target - OUT) \qquad (1.9)$$

The δ is multiplied by OUT from a neuron p. which is the source neuron for the weight in question. This product is in turn multiplied by a training rate coefficient and the result is added to the weight.

An identical proceeding from a neuron in the hidden layer to a neuron m the output layer. The following equations illustrate this calculation:

$$\Delta W_{pq,k} = \eta \delta_{q,k}\ OUT_{p,j} \qquad (1.10)$$

$$W_{pq,k}\ (n+l) = W_{pq,k}\ (n) + AW_{pq,k} \qquad (1.11)$$

where,

$W_{pq,k}\ (n)$ = the value of a weight from neuron p in the hidden layer to neuron q in the OUT layer at step n (before adjustment).

$W_{pq,k}\ (n+ 1)$ = value of the weight at step (n+1) after adjustment.

$\delta_{q,k}$ = the value of δ for neuron q in the output layer k.

$OUT_{p,j}$ = the value of OUT for neuron p in the hidden layer j.

n = number of steps required to converge.

1.7.2.2.2 Hidden Layers Weights

Hidden layers have no target vectors, so the training process described above cannot be used to adjust its weights. This lack of training target stymied efforts to train multilayer networks. Backpropagation trains the hidden layers by propagating the output error back through the network layer by layer, and adjusting the weights of each layer. Equations (1.10), (1.11) are used for all layers, both output and hidden; however, for hidden layers, δ must be generated without benefit of a target vector. Firstly, δ is calculated for each neuron in the output layer as in equation (1..9). It is used to adjust the weights feeding into the output layer, then it is propagated back through the same weights to generate a new value of δ for each neuron in the last hidden layer.

These values of δ are used, in turn, to adjust the, weights of this hidden layer, using equations (1.10, 1.11), and in a similar way are propagated back to all preceding layers. Consider a single neuron in the hidden layer just before the output layer. In the forward pass, this neuron propagates its output value to neuron in the output layer through the interconnecting weights. During training, these weights operate in reverse, passing the value δ from the output layer back to the hidden layer. Each of these weights is multiplied by the δ value of the neuron to which it connects in the output layer. The value of δ needed for the hidden-layer neuron is produced by summing all such products and multiplying by the derivative of the sigmoid function:

$$\delta_{p,j} = OUT_{p,j} (1-OUT_{p,j}) (\Sigma\delta_{q,k} W_{pq,k}) \qquad (1.12)$$

Now, with δ in hand, the weight feeding the first hidden layer can be adjusted using equation 3.12 modifying indices to indicate the correct layers. For each neuron in a given hidden layer δ's must be calculated, and all weights associated with that layer must be adjusted. This is repeated, moving back toward the input layer by layer, until all weights are adjusted.

2. Artificial Immune Systems

The main function of the immune system is to protect the body from pathogens and cancer. Vertebrate immune systems are more complex than the invertebrates. They are characterized by two important properties, which are *memory* and *specificity*. In the case of invertebrate, the immune system consists mainly of Phagocytes which are non-specific. This means that it will not remember any previous antigen, and will use the same attacking strategey each time. Phagocytes has no receptors for specific pathogens, which means that these cells will engulf and try to kill any pathogen. On the other hand, the vertebrate host has evolved more specialized cells called *Lymphocytes*. These Lymphocytes are pathogen specific, which means that they have distinct receptors to interact with different pathogens [5].

To combat antigens, nature has provided us with the immune system. The blood, lymph nodes, and bone marrow act with the liver, spleen, thymus, and tonsils to produce and deliver specialized cells, including B-lymphocytes, T-lymphocytes, and phagocytes. These cells limit the severity and duration of colds, Fight infections in the nose and throat, help wounds to heal, destroy some cancers, and much more.

2.1 Innate Versus Acquired Immunity

There are two types of immunity, innate immunity and adaptive or acquired immunity. Also, the immune system response can be divided into humoral immunity, and cell mediated response [6].

2.1.1 Innate Immunity

The innate immunity can be regarded as natural resistance of the host to foreign pathogens. There are a number of external and internal lines of defenses in the innate immunity. As an examples we find Lysozymes in tears, and skin inflamation as a resistance to a peneterating pathogen. The innate immunity is the first line of defense against the foreign pathogens, and it uses the non-specific strategy while attacking it. Phagocytes engulf the foreign pathogen, and try to kill it.

Some examples on the same line of defense are *Monocytes*, *Macrophages*, and *Neutrophils*. There are other types of cells that is called Natural killer cells *NK-cells* that also use non-specific response to protect the host against the foreign pathogen.

2.1.2 Acquired Immunity

In contrast to the innate immune system, the acquired immune system uses a specific response to pathogens. The important advantage of the acquired immunity is the use of memory through lymphocytes. After getting rid of the foreign pathogen the lymphocytes change into memory cells. These memory cells will recognize rapidly the same pathogen when it evades the host again, and eliminate it before causing any damage. The two major types of lymphocytes are T-cells, and B-cells. B-cells have direct contact with the antigen when interacting with it. On the other hand, T-cell can bind to the antigen only after it is processed and presented by other cells. B-cells are the basic building block of the humoral immunity through the production of antibodies. Cell mediated immunity is contributed by T-cells mediated response. T-cells have many forms like the helper T-cell which helps either B-cells, or phagocytic macrophages. Another form that the T-cell can be is the cytotoxic T-cells, which recognize cells infected by virus or cancer, and eliminate them.

2.2 Antigens

An antigen (Ag) can be defined as a substance that triggers specific immune response. In vertebrates, the host system does not respond to its own proteins, and that is called *tolerance*. T-cells and B-cells that are capable of recognizing self-cells are eliminated during maturation phase,. An antigen may carry several epitops, and consequently this will trigger the production of several antibodies, see Figure1.5.

Generally, T or B cells do not recognize all of these epitopes, instead they recognize part of it. So, a single Ag may attract the attention of several T or B cells. Also, two different antigens may carry the same *crossreactive* epitopes, which means that an antibody produced for that antigen can interact with another one .

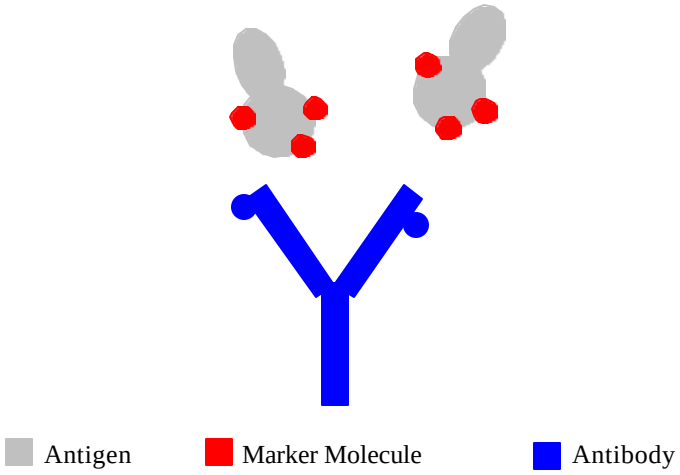

Figure1.5 Antigen Antibody Interaction

2.3 Immune Cells

Cells destined to become immune cells are produced in the bone marrow. The descendants of some stem cells become lymphocytes, while others develop into a second group of immune cells known as phagocytes. The two major classes of lymphocytes are B cells and T cells. B cells complete their maturation in the bone marrow. On the other hand, T cells migrate to the thymus; an organ that lies high behind the breastbone. Each lymph node contains specialized compartments that house a great number of B lymphocytes, T lymphocytes, capable of presenting antigen to T cells. Thus, the lymph node brings together the several components needed to start an immune response [7].

2.4 B-Cells and Antibodies

B-Cell is one of the major arms of the immune system mechanisms, and it is responsible for the humoral response. The name humoral comes from these fluids that circulate arround the body known as humors. Each B cell is programmed to make one specific antibody. When a B cell encounters its triggering antigen, it produces many large plasma cells. Every plasma cell is a factory for producing antibody. Each of the plasma cells descended from a given B cell produces millions of identical antibody molecules and pours them into the bloodstream, see Figure1.6. A given antibody matches an antigen as a key matches a lock, and marks it for destruction.

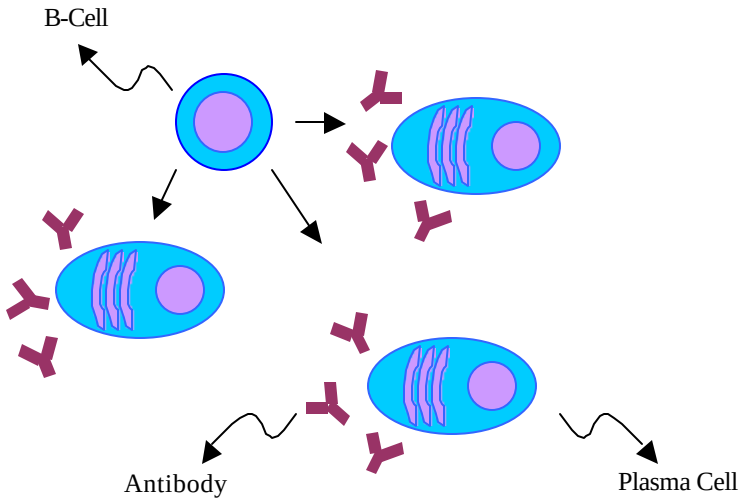

Figure1.6 Production of Antibodies

2.5 T-Cells and Lymphokines

T-Cells play two rolls in the immune system defense. B cells cannot make antibody against most substances without regulatory T-cell help. On the other hand, Cytotoxic T-cells, directly attack body cells that are infected. Another important regulatory T cells are "helper" cells. Typically identifiable by the T4 cell marker, helper T cells activate B cells and other T cells as well as natural killer cells and macrophages. Another subset of T cells contributes by turning off or "suppress" these cells. T cells work by secreting cytokines or, Lymphokines which are considered to be chemical messagers.

2.6 Macrophages

Macrophages are responsible for carrying the initial attack against an invasion launched by antigens. Macrophages are distributed throughout body tissues , and they rid the body of worn-out cells and other debris. Foremost among the cells that "present" antigen to T cells, having first digested and processed it, macrophages play a crucial role in initiating the immune response. As secretory cells, Monocytes and Macrophages are essintial to the regulation of immune responses and the development of inflammation; they produce an array of powerful chemical called Monokines including enzymes, complement proteins, and regulatory factors such as interleukin-1. Sometimes antigens change themselves, and that is why we continue to get sick.

2.7 An Overview of the Immune System

When foreign antigen enters the body, it triggers B-cells to produce antibodies, which bind to the antigen and clear it from the body; this is called Humoral immune response. The cell-mediated response involves helper T-cells and T cytotoxic (CTL) cells. Helper T-cells (T_h) can be divided into two sub fields: T_{h1} and T_{h2}. T_{h1} cells help B-cells, where T_{h1} cells activate macrophages [8]. CTL cells kill virtually infected or Cancer cells, see Figure1.7.

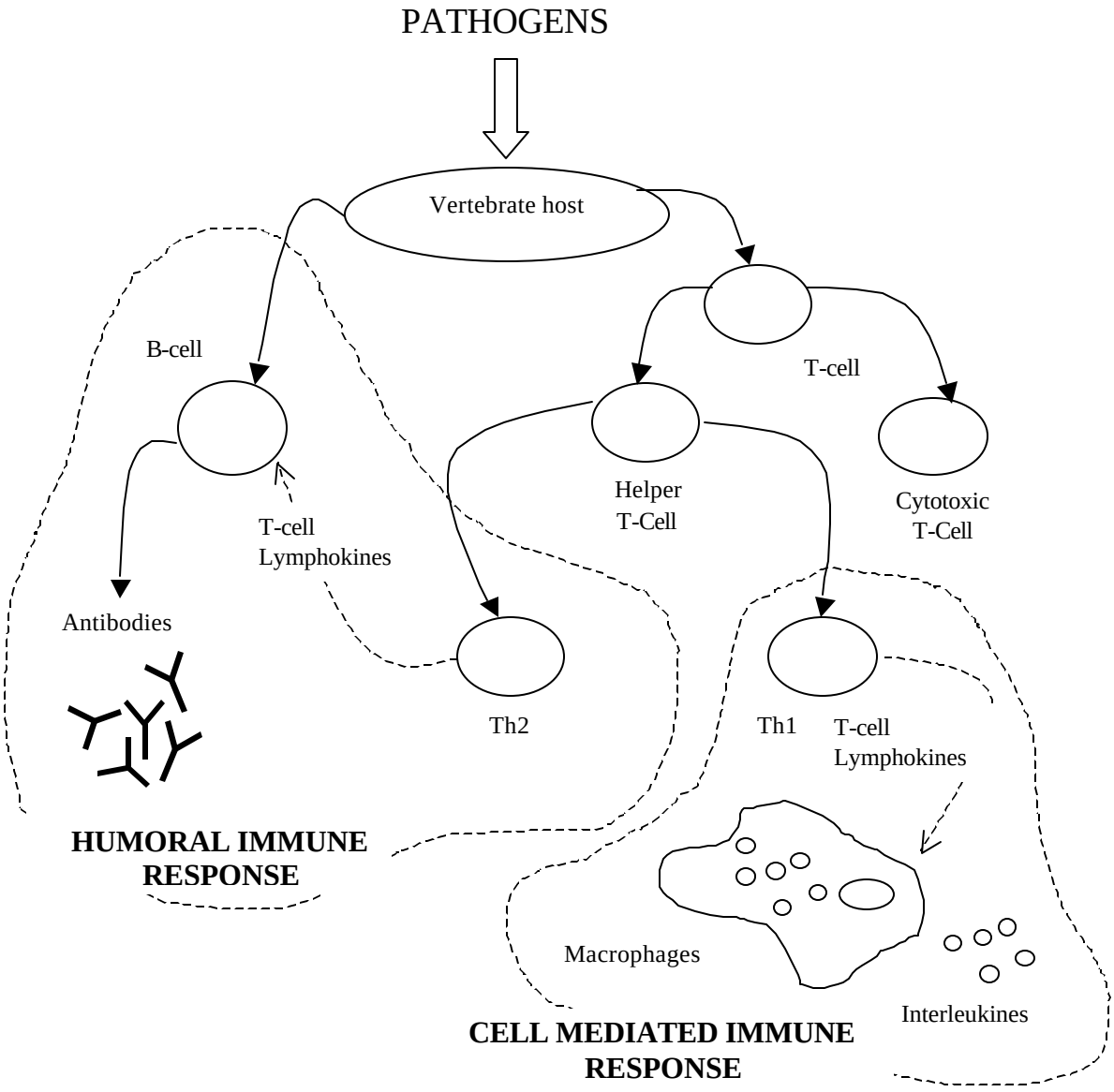

Figure1.7 An Over view of the Immune System

2.7.1 Humoral Response

When the B-cell proliferate, all of its descendants will make this uniquely rearranged set of antibodies. B-cells continue to multiply, various mutants arise; these allow for the natural selection of antibodies that provide better and better "fits" for antigen elimination. The result of this entire process is that a limited number of B-cells can respond to an unlimited number of antigens. Antibodies are triggered when a B-cell encounters its matching antigen, and digest it. Antigen fragments are displayed on B-cell distinctive markers. The combination of antigen fragments, and marker molecules attract the mature matching helping cells. T-Cells secrete Lymphokines allow B-cells to multiply and mature into antibody producing Plasma cell. Antibodies are released into the blood stream, and they lock into matching antigens. These antigen-body complexes are soon overcome either by the complement cascade, or by the liver and spleen [9], see Figure1.8.

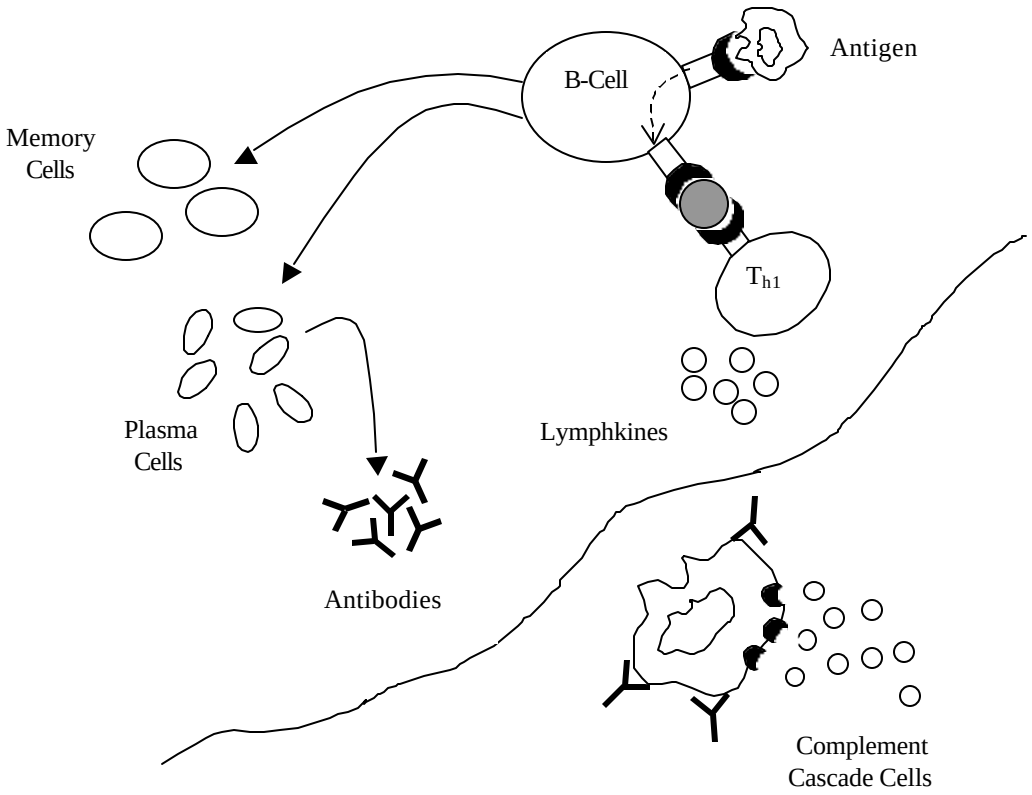

Figure1.8 Humoral Response

2.7.2 Cell Mediated Response

Machrophages initiate the cell mediated response, or by other antigen-presenting cell. The antigen-presenting cell digest the antigen, and then displays antigen fragments on its own surface. Bound to the antigen fragment is an MHC molecule. These fragments capture the T cell's attention. A T cell whose receptor fits this antigen binds to it. This bond stimulates the antigen-presenting cell to secrete Interleukins required for T cell activation and performance, see Figure1.9.

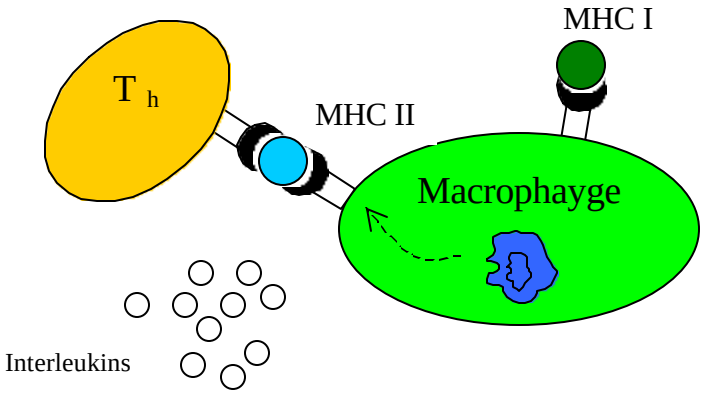

Figure1.9 The Cell Mediated Response

2.8 Analysis of Lines of Defense

The human immune system attempts to quickly control the spread of antigens once they have been identified. There are several other lines of defense against antigens besides the immune system. The first line of defense is the skin, which prevents the invasion of most micro organisms. The proteins and acidity of the saliva in the mouth and stomach digest harmless microorganisms. However, if there is a cut in the skin, or fluid transmission occurs, pathogens can invade the body.

The second line of defense is the cell-mediated response of the immune system. Macrophages are circulating throughout the body that destroy the invading microorganisms by phagocytosis. The last line of defense is known as the humoral immune response. Many types of immune cells are triggered to move into the affected area, and a great deal of antibodies and phagocytes destroy the invading antigen.

2.9 Memory Cells

Some of the lymphocytes activated during the primary immune response remain dormant and keep circulating in the immune system for a long time. These lymphocytes carry the memory of the encountered antigen, and therefore these long-lived cells are called memory cells. Memory can also be maintained by long-lived antigen (not necessarily by a population of long-lived distinct memory cells). Whenever T cells and B cells are activated, some of the cells become "memory" cells. Then, the next time that an individual encounters that same antigen, the immune system is primed to destroy it quickly. The degree and duration of immunity depend on the kind of antigen, its amount, and how it enters the body. An immune response is also dictated by heredity; some individuals respond strongly to a given antigen, others weakly, and some not at all [10].

Memory T Cells: Memory T cells are formed during an immune response. As the term implies, memory T cells remember past attacks by antigens, and can respond with increased strength during subsequent invasions by a particular pathogen. Memory T cells are long lasting immune cells, and react to particular antigens. Unlike T cells that recirculate in the blood and lymph, memory T cells often circulate throughout the entire body, especially in the site they were originally activated. Memory T cells rely on memory helper T cells for launching a global immune response. Look below for links

Memory Helper T Cells: Memory helper T cells are also known as memory effector T cells. Memory helper T cells are used by memory T cells to launch an immune response against an attack by pathogens. Memory helper T cells react in much the same way as helper T cells, except that they are stimulated by memory T cells. Memory helper T cells can differentiate into a cytotoxic T cell that attacks abnormal cells, or into a helper T cell that stimulates an immune response from B cells. Look below for links to other immune cells.

Memory B Cells: Little is currently known about memory B cells. However, memory B cells are probably similar to memory T cells in that they retain a strong affinity to low concentrations of antigen, and are able to launch a strong immune response following

stimulation by a particular antigen that they are sensitive to. Like normal B cells, memory B cells circulate throughout the entire body. However, they are significantly longer lived, on scales of a few months to years. Look below for links to other immune cells.

3. Multi-Agent Systems

The field of Artificial intelligence and its techniques has become an interesting field of research in the last few decades. *Distributed Artificial Intelligence* (DAI), as a sub field of AI, has existed for less than two decades. DAI is concerned with systems that consist of multiple independent entities that interact in a domain. It has been divided into two sub disciplines: Distributed Problem Solving (DPS), and Muti-Agent Systems (MAS). Distributed Problem Solving (DPS): Focuses on the information management aspects of systems with several branches working together towards a common goal. Muti-Agent Systems (MAS): Deals with behavior management in collections of several independent entities, or agents.

There are many definitions for an *Agent*. One of them is that an agent is an entity with goals, actions, and domain knowledge all situated in an environment. The way the agent acts is its behavior, and there should be an interaction between this behavior and the environment that surrounds him. Another definition is that an *Agent* is a bundle of skills: Problem solving skills (what the agent can do), Social skills (what the agent choose to do), and learning skills (how the agent transforms experience into new skills). In this case, the agent environment is the set of all things that he can affect or by which he is affected [11], and [12].

3.1 Single-Agent Systems

The agent in a single-agent system models itself, the environment, and its interactions. Single agents independent entities with their own goals, actions, and knowledge (experience). Even if there are other agents in the world they are just considered part of the environment. This means that although agents are *also* a part of the environment, they are explicitly modeled as having their own goals, actions, and domain knowledge.

3.2 Multiagent Systems

In case of multiagent systems several agents exist which model each other's goals and actions. These agents might have direct interaction among each others, i.e. they may have communication, and this interaction could be viewed as environmental stimuli.
Multiagent systems differ from single-agent systems in that the environment's dynamics can be determined by other agents. So, all multiagent systems can be viewed as having dynamic environments, which they affect, and be affected by it [13].

Important factors in designing MAS

_ Number of agents.

_ Real time managment?

_ Agents' goals?

_ Information about environment.

_ Agent-action dependencies.

3.3 General Homogeneous MAS

Homogenuous MAS are considered to be several different agents with identical structure (sensors, effectors, domain knowledge, and decision functions), but they have different sensor input and effector output. These agents are situated differently in the environment and they make their own decisions. We must keep in mind that having different effector output is a necessary condition for MAS, because if all agents act as a unit, then they will be considered as a single agent [14], and [15].

3.4 General Heterogeneous MAS

In addition to homogenous MAS, we have heterogeneous MAS. Agents are situated differently in the environment; they have different sensory inputs, and take different actions. However in this scenario, the agents have much more significant differences. Thus, they may have different goals, actions, and/or domain knowledge. This condition of heterogeneity among agents adds a great power for the system designer in MAS [16].