The Meta-Generalized Delta Rule: A New Algorithm for Learning in Connectionist Networks

Dean A. Pomerleau

September 1987
CMU-CS-87-165

Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15217

ARPA-NET: Pomerlea@F.CS.CMU.EDU

Abstract

Currently the most popular learning algorithm for connectionist networks is the generalized delta rule (GDR) developed by Rumelhart, Hinton & Williams (1986). The GDR learns by performing gradient descent on the error surface in weight space whose height at any point is equal to a measure of the network's error. The GDR is plagued by two major problems. First, the progress towards a solution using the GDR is often quite slow. Second, networks employing the GDR frequently become trapped in local minima on the error surface and hence do not reach good solutions.

To solve the problems of the GDR, a new connectionist architecture and learning algorithm is developed in this thesis. The new architectural components are called meta-connections, which are connections from a unit to the connection between two other units. Meta-connections are able to temporarily alter the weight of the connection to which they are connected. In doing this, meta-connections are able to tailor the weights of individual connections for particular input/output patterns. The new learning algorithm, called the *meta-generalized delta rule* (MGDR), is an extension of the GDR to provide for learning the proper weights for meta-connections.

Empirical tests show that the tailoring of weights using meta-connections allows the MGDR to develop solutions more quickly and reliably than the GDR in a wide range of problems.

Acknowledgments

This report is a revised version of a thesis submitted to Williams College as partial fulfillment of the requirements for the degree of Bachelor of Arts with honors in computer science.

I thank Prof. Kim Bruce for his patience and support throughout this research. Despite the fact that he was originally unfamiliar with the subject matter of this thesis, he has willingly overseen my work and has constantly provided me with keen insights I would have otherwise missed. I cannot hope to thank him sufficiently for the opportunity he has provided me. I am also in debt to Prof. Donald House, who has selflessly shared with me much of his time and expertise in the area of biological computation. I also thank the professors for my other classes, especially Tom Murtagh and Pat Carroll, whose understanding during the last few weeks of this thesis work has helped immensely.

At CMU, I thank Roni Rosenfeld, Barak Pearlmutter, Jay McClelland, Scott Fahlman and especially Dave Touretzky for their help in revising this report.

My family and friends have given me warmth, support and for this I am grateful. I cannot hope to thank them properly; thank goodness I don't need to.

Dean Pomerleau

Table of Contents

Chapter 1. Introduction.. 1

Chapter 2. Connectionist Basics... 6

Chapter 3. The Generalized Delta Rule...................................... 12

Chapter 4. The Meta-Generalized Delta Rule................................ 25

Chapter 5. Simulation Results.. 33

Chapter 6. Analysis of the MGDR... 45

Chapter 7. Conclusion... 56

References.. 59

Chapter 1
Introduction

This thesis describes, demonstrates and analyzes a new method for learning in connectionist systems called the *meta-generalized delta rule* (MGDR). The MGDR is an extension of the generalized delta rule (GDR) (Rumelhart, Hinton & Williams, 1986), which allows the present context of the network to affect the weights of individual connections. The central claim of this thesis is that by allowing the weights of connections to be partially determined by the state of their environment, the MGDR can offer dramatic improvements in learning speed and reliability over the GDR.

This chapter describes the need for connectionist systems in general and provides a general overview of the remainder of this thesis.

The goal of connectionism is to develop massively parallel computational networks. Towards this end connectionist researchers have adopted a brain-like architecture for their networks. The resemblance of connectionist systems to neural networks results from the fact that there are a number of attributes inherent in neural networks that connectionists hope to capture in their systems.

Some of the primary benefits of computing using a brain-like architecture result from the highly parallel nature of these networks. For instance, the human brain can perform the computations required to do visual object recognition in less than 200 msec, which allows for the sequential firing of fewer than 100 neurons (Thompson, 1986). Serial computers, performing many millions of sequential instructions, are still unable to approach the proficiency of the brain at visual recognition tasks. Somehow the 100 step "program" the brain is using to perform visual analysis is very powerful. The power of the human visual system results from the fact that during those 100 steps, the brain is performing a huge amount of parallel computation. This parallel computational power is one aspect of neural networks that connectionists hope to duplicate in their systems.

A second important property that emerges from the interaction of many processing units is the tendency such systems have towards graceful degradation. The brain is in fact, an amazingly resilient computational system. For example, it is not uncommon for stroke patients, who have had large portions of their brains destroyed, to retain a large part of their former capacities. It appears that either no one specific area of the brain is crucial for many of our abilities or else the brain is able to reorganize itself to circumvent the loss of one of these crucial areas. Either way, the resulting fault tolerance is a very desirable property missing from most traditional computational systems.

The ability to spontaneously generalize in new situations is another aspect of neural networks that connectionists hope to take advantage of in their systems. When we confront something new, we are usually able to draw on analogous past experiences to guide us. Computers are notoriously bad at reasoning by analogy, primarily because of their address based memory scheme. Unless one is willing to contend with a time consuming search, one must tell the computer exactly where to look in its memory for the information appropriate to the situation. As a result, traditional computers are terrible at dealing with incomplete or misleading information.

Perhaps the most important appeal the brain-like computational systems have is their ability to learn. One of the major problems with traditional artificial intelligence is the emphasis placed on the "blocks world approach". In the blocks world approach to developing artificial intelligence, a researcher first chooses an environment or situation in which he would like his computer to act "intelligently". Then the researcher compiles all the knowledge he considers necessary for functioning in the environment or solving the particular problem at hand. The final steps of the process are to program this information into a model of intelligence and test the model's performance to determine if it acts "intelligently".

The troubles with such an approach are twofold. First, a large part of intelligence involves the ability to learn from one's mistakes. If you bump into a block in your blocks world environment, you should learn that blocks are solid and not to run into them. The traditional approach to AI however, has the model "in-born" with this kind of knowledge.

But a more important criticism of this technique concerns its practicality. While it is possible to directly plug in the knowledge necessary for functioning in a toy environment like blocks world, such an approach quickly becomes impossible as the complexity of the target environment increases. In short, traditional artificial intelligence is faced with what has been termed the knowledge bottleneck problem: how to get all the information necessary for functioning in a complex environment into the computer in a usable form.

The hope of connectionism is to put together a relatively general network of interconnecting neuron-like computational units, and by using a learning algorithm to alter the pattern of interaction between these units, develop a system that can learn the knowledge required for intelligent behavior through experience.

This introduction was meant to justify the work being done in connectionism. The remainder of this thesis explores specific aspects of connectionism, and particularly learning in connectionist networks. Chapter 2 presents the specific components of connectionist models, including such aspects as the nature of the neuron-like computational units and the means of communication between them.

Chapter 3 consists of two parts. In the first, a technique developed by Rumelhart, Hinton & Williams (1986) for learning in connectionist networks, called the generalized delta rule (GDR) is described. In the second part, the need for a better learning algorithm is justified by pointing out the shortcomings of the GDR.

Chapter 4 describes a new solution to some of these problems, the MGDR. In this chapter, the modifications to the traditional network architecture needed by the MGDR are described. Specifically, the concept of meta-connections, which are connections from one unit to the connection between two other units, are illustrated. Next comes a mathematical presentation of the means by which meta-connections interact with the traditional network components. In addition, it is shown that the MGDR is really an extension of the GDR by proving that in a traditional network, that is, one without meta-connections, the MGDR behaves exactly like the GDR. Despite the fact that meta-connections alter the strength of what may be considered the traditional weight of a connection, it is shown that the change to the traditional weight of a connection on any given trial using the MGDR is identical to that in the GDR.

Chapter 5 explores the MGDR in action. Specifically, the results of simulations performed to test the learning ability of the MGDR are presented. Throughout chapter 5 the MGDR and the GDR are compared in terms of how quickly and reliably they develop solutions to problems. My major finding is that, relative to the GDR, the MGDR is more consistently able to quickly develop a set of weights to solve a relatively wide range of problems. In addition, preliminary results indicate that the solutions developed by the MGDR are more generalizable in that when faced with an input pattern never encountered before, a network with a solution developed using the MGDR more readily returns the correct "answer" than does a network with a solution developed using the GDR.

In Chapter 6 the reasons for the performance difference between the GDR and the MGDR are described. The MGDR's learning advantage is primarily due to the fact that meta-connections allow the weights of individual "normal" connections to be tailored for individual trials. Since the network can tailor connection weights for individual trials, it less frequently encounters the problems inherent in trying to find a single set of weights which satisfies all facets of a problem. Chapter 6 also contains a comparison between the MGDR and other extensions of the traditional network architecture, including adding more units as a substitute for meta-connections and adding what Rumelhart, Hinton & McClelland (1986) call sigma-pi units. The conclusion drawn in Chapter 6 is that meta-connections provide a combination of attributes which is missing from these other extensions, namely a flexible architecture which can learn quickly.

Finally, in Chapter 7, we conclude by recapping the major findings concerning the MGDR, discussing some of its shortcomings and outlining some of the important questions this thesis leaves unanswered.

Chapter 2
Connectionist Basics

According to Rumelhart, Hinton & McClelland (1986), there are at least six major aspects of a connectionist computational network:

1) A set of processing units
2) An activation state for each unit
3) A connectivity pattern between the units
4) An activation rule for determining the activation state of a unit from the inputs impinging on it from other units
5) A learning rule by which the pattern of connectivity is modified by experience
6) An overall system architecture

The components in connectionist networks which perform computation are neuron-like elements called units. The i^{th} unit in a particular network is denoted with the notation u_i. Associated with each unit u_i is an activation state, o_i. The activation state of a unit represents how active that unit is, or from a neural standpoint, the neuron's rate of firing. The activation level is usually represented by a real number between 0.0 and 1.0.

The pattern of connectivity of a network is composed of uni-direction links between units called connections. These connections are similar in many ways to synapses in the brain, in that it is through these connections that units communicate. The connection from u_i to u_j is denoted c_{ji}. Each connection has associated with it a signed real number called its weight. The weight from u_i to u_j is denoted w_{ji}. A positive weight signifies that the connection is excitatory and a negative weight signifies that the connection is inhibitory. The magnitude of the weight determines the degree to which a connection is excitatory or inhibitory. An excitatory connection from u_i to u_j (see Figure 2-1) means that if u_i has a relatively high activation level then it will excite u_j, causing its activation level, o_j, to increase. Conversely, if the connection from u_i to u_j is inhibitory, then if u_i has a high activation level it will decrease o_j. The magnitude of the influence u_i has on the activation level of u_j at any given time depends on o_i, w_{ji}, and the activation function f, which maps a unit's net input from other units to an activation level for that unit.

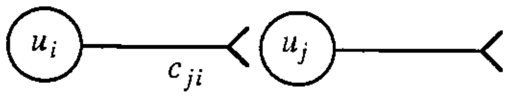

Figure 2-1. Two connected units.

In all connectionist systems,

$$o_j = f(net_j)$$

where

$$net_j = \sum_i w_{ji}\, o_i,$$

for all u_i with connections to u_j.

The specific activation function f varies from model to model. Perhaps one of the most popular activation functions, and the one used exclusively in this paper, is the logistic activation function, represented mathematically by

$$o_{pj} = \frac{1}{1 + e^{-\left(\sum_i w_{ji}\, o_{pi} + \theta_j\right)}}$$

where θ_j is a bias term similar in function to a threshold.

The logistic activation function has a number of advantages. First, the function mimics the activation profile of actual neurons (House, personal communication) in that as the magnitude of the net input to a unit gets large, the function asymptotically approaches its boundary (see Figure 2-2). In addition, the logistic activation function is differentiable, which will be seen to be a requirement for all interesting learning rules.

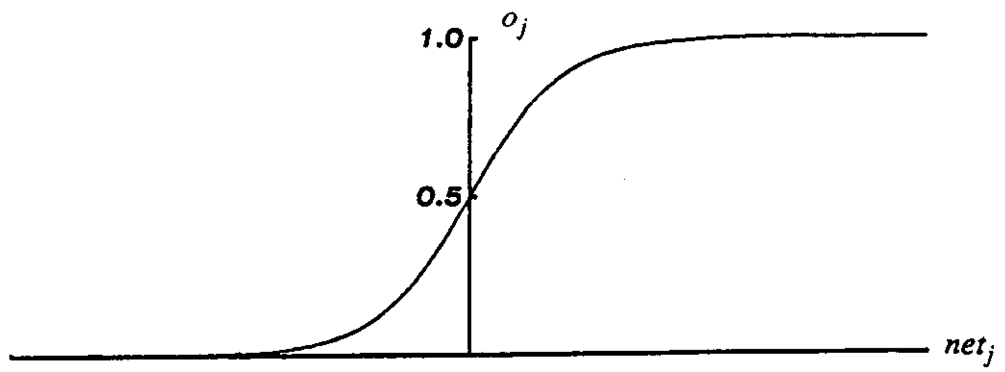

Figure 2-2. The logistic activation function

Before discussing learning rules in connectionist models, it is important to understand two more aspects of these networks. The first is the traditional network architecture employed in connectionist systems. The most common architecture, and the one used almost exclusively in this thesis, involves three layers of units; an input layer, a hidden layer and an output layer (see Figure 2-3).

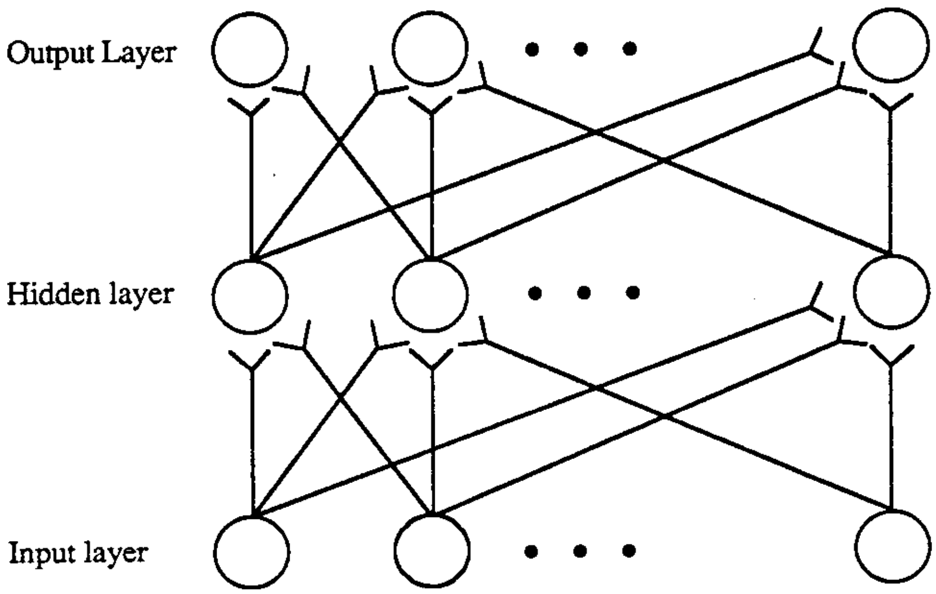

Figure 2-3. A three-layered connectionist network.

The input layer is comprised of units whose activation levels are determined externally to the system. This layer can be thought of as providing the system with sensory input. The input layer is generally connected only to the hidden layer. The hidden layer can be envisioned as the place at which input is processed and categorized. The hidden layer usually only has connections to the output units, although in some networks there may be connections between hidden units.

Finally, the output layer is the set of units to which the results of the processing of the input by the hidden units is relayed. The general goal of this type of system is to transform a given input pattern, represented by a vector of activation levels, into an output pattern represented by another vector of activation levels.

For example, the network in Figure 2-4 might be used to compute the exclusive-or (XOR) of the two input units. For this example, the desired activation level of the output unit is the exclusive-or of the activation level of the two input units. The activation levels of the input units would be externally set to one of the four XOR patterns (0 and 0, 0 and 1, 1 and 0, or 1 and 1) and the goal would be to have the two hidden units transform this input into the appropriate answer, (0, 1, 1 or 0, respectively, for each of the input patterns), represented by the activation level of the output unit. The network in Figure 2-4 could be taught a proper set of connection weights to compute the XOR function using a back propagation algorithm like the Generalized Delta Rule. This learning rule will be discussed in detail in the next chapter.

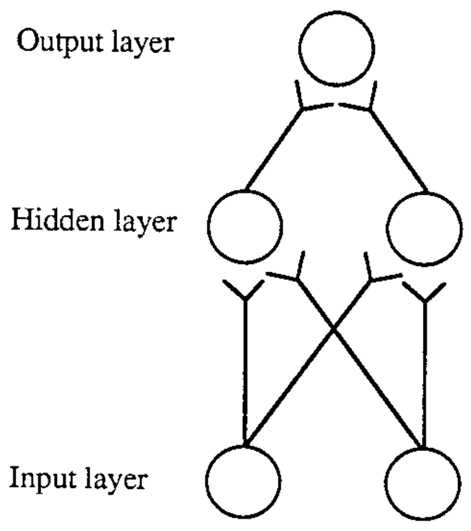

Figure 2-4. A network for computing XOR.

Because of the asymptotic nature of the activation function, it is obvious that the activation level of the output unit can never reach a perfect "answer" (an activation level of 0.0 or 1.0). Therefore a means of measuring how close a network is to "solving" a problem is needed.

Rumelhart, Hinton & Williams (1986) used the error function

$$E = \sum_p E_p$$

where E_p is the measure of the networks error on input/output pattern p as given by

$$E_p = \frac{1}{2}\sum_p (t_{pj} - o_{pj})^2$$

where t_{pj} is the "answer" activation level that u_j should have on input/output pattern p and o_{pj} is the level of activation that u_j actually has on input/output pattern p.

A useful way of conceptualizing the error in a given network is by considering the error in the network as a function of weights of the network's connections. Figure 2-5 is an illustration of such a conceptualization in which the height of the surface at any point in weight space represents the value of the error function *E*. At point *A* in weight space, *E* is quite high, so the network is far from a "correct" solution. At point *B* however, the network's error is minimized, and hence the system is as close to the "correct" solution as possible. Note that at point *A*, any changes made to the weights of the network's connections will decrease the network's error, while at point *B*, any changes made to the weights of the network's connections will increase the network's error.

The learning rules upon which this thesis will focus can be thought of as techniques for changing the weights of connections in order to move efficiently from point *A* to point *B*.

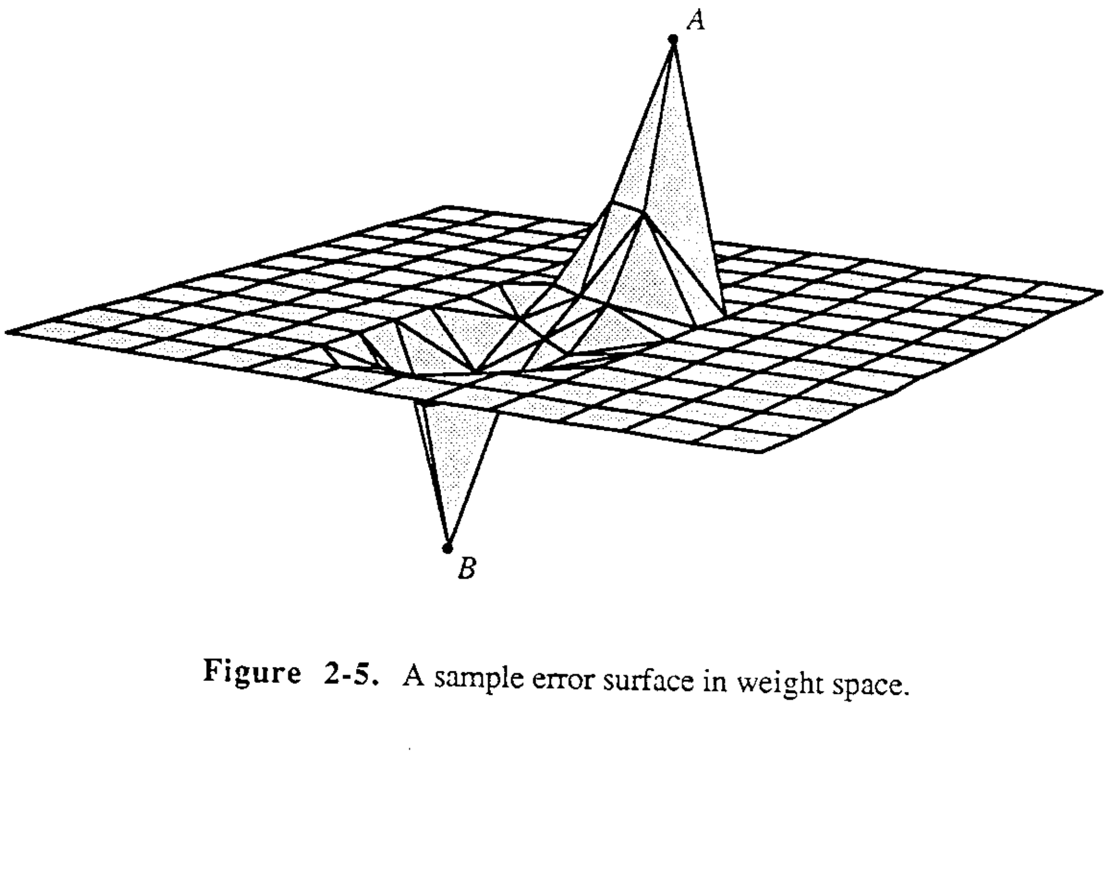

Figure 2-5. A sample error surface in weight space.

Chapter 3
The Generalized Delta Rule

This chapter describes the connectionist learning rule upon which this thesis is based. It is called the generalized delta rule and it was developed by Rumelhart, Hinton & Williams (1986). Conceptually, the GDR can be thought of as a technique for starting at point *A* and getting to point *B* in Figure 2-5 by gradient descent. In other words, the GDR attempts to reduce the error in a given network by altering the network's weights in the direction with the steepest downward slope.

In this section, the GDR will be described mathematically and the proof developed by Rumelhart et al. illustrating that the GDR does gradient descent in error space will be presented. Finally, the problems with the GDR will be described as a lead-in to the MGDR. The purpose of this section is twofold. It is intended to give the reader an idea of the work others are doing in the area of connectionist learning systems. More importantly, an understanding of the concepts behind the GDR is crucial if one is to understand the MGDR, since the latter is a direct extension of the former.

The GDR is itself an extension of the perceptron learning rule developed by Minsky and Papert (1969). The perceptron learning rule was originally developed for networks with only two layers of units, input and output units.[1] In their classic book *Perceptrons*, Minsky and Papert showed that there are severe limitations on the computational power of these simple two-layered networks. In short, there is a large class of often trivial input/output mappings that perceptrons cannot solve. The problem is that with only two layers, perceptrons are unable to

[1]Actually, perceptrons had three layers, but since the weights between the first and the second layers were fixed, it is easier to conceptualize them as two-layered networks.

recode the input pattern. Therefore, a perceptron can only make the correct mapping from input to output pattern if the output pattern is linearly dependent on the input pattern. For a more detailed description of perceptrons and their limitations see Minsky and Papert (1969).

Perhaps the best example of the limitations of perceptrons is the exclusive-or (XOR) problem, in which the activation of the one output unit should be the XOR of the activation level of the two input units (see Figure 3-1). A perceptron, with connections only from input to output units cannot compute the XOR of the input units. To understand why, consider the necessary mappings. When one of the two input units has an activation level of 1.0, then the output unit should also have an activation level of 1.0. Therefore each of the input units should have excitatory connections to the output unit. But when both of the input units have an activation level of 1.0, these excitatory connections will cause the output units to have a high activation level, when its activation level should really be 0.0. The problem with perceptrons is that since they have no way of recoding the input pattern, they can only map similar input patterns to similar output patterns. In other words, the output pattern must be a linear combination of the input units. In the XOR case, since a high level of activation of either input unit should produce a high output level, there is no way for the network to learn that when both input units have a high activation level, the output unit should have a low activation level.

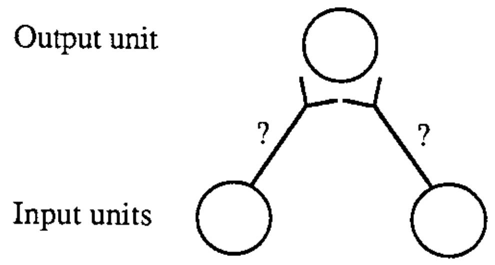

Figure 3-1. A perceptron unable to compute XOR.

For our purposes, it is enough to know that hidden units are necessary so that a network can recode the input pattern when the output pattern is not an obvious linear function of the input pattern. The GDR is a generalization of Minsky & Papert's perceptron learning rule to networks with hidden units.

The learning procedure for the GDR involves the presentation of a set of input and output patterns. The rule initially uses the input pattern and the logistic activation function to produce an output pattern in the output layer. If the output pattern produced is identical to the pattern dictated by the target output pattern, then no weight changes take place. Otherwise, the weights are changed to reduce the difference between the actual and target output pattern. The rule for changing weights following the presentation of input/output pattern p is as follows:

$$\Delta_p w_{ji} = \eta \, \delta_{pj} \, o_{pi} \qquad (1)$$

where $\Delta_p w_{ji}$ is the change in the weight of the connection from u_i to u_j following presentation of input/output pattern p, η is a learning rate constant, o_{pi} is the actual activation level of u_i, and δ_{pj} is a measure of the distance the activation level of u_j is from its target.

The value of δ_{pj} in the GDR depends on whether u_j is an output unit or a hidden unit. If u_j is an output unit:

$$\delta_{pj} = (t_{pj} - o_{pj}) \, f'_j(net_{pj}) \qquad (2)$$

where t_{pj} is the target activation level for u_j on input/output presentation p, and $f'_j(net_{pj})$ is the derivative of the activation function for u_j with respect to net_{pj}, the net input to u_j for input/output presentation p.

The major innovative aspect of the GDR is its formula for computing the delta value for hidden units. The problem prior to the development of the GDR, with the perceptron learning rule in particular, was that there was no good way to calculate the distance the activation level of a hidden unit is from its "proper" level. The reason for this is simple; it is very difficult to know the "proper" activation level for a hidden unit because there is no clear target activation level for the hidden unit as there is for an output unit. Rumelhart, Hinton & Williams (1986) solved this problem by using a recursive formula to compute the distance. The GDR's mathematical formula for δ_{pj} when u_j is a hidden unit is:

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk}\, w_{kj} \qquad (3)$$

The concept behind the computation of the delta value for hidden units is relatively straightforward. The value of δ_{pj} for hidden unit u_j is calculated recursively by determining u_j's influence on all units u_k to which it is connected and for which we have already calculated the value δ_{pk}. For instance, if hidden unit u_j makes excitatory connections to output units which should have a lower activation level and hence have a negative delta value, then

$$net_{pj} = \sum_k \delta_{pk}\, w_{kj}$$

will be negative. Since the logistic activation function is a nondecreasing function of net_{pj}, the value of δ_{pj} will be negative, indicating that the activation of u_j should be lower. Intuitively this seems correct in that, if u_j is exciting output units which should have lower activation levels, the activation level of u_j should be decreased.

Learning using the GDR involves two phases. First, activation is propagated forward through the network from the input units to the hidden units and finally to the output units. Then, in the second phase, error is propagated backward through the network using the following procedure. First, Equation 2 is used to compute the delta value for each output unit by comparing the actual activation levels of the output units with the desired levels. The weights from hidden to output units are then altered using these delta values and Equation 1. The delta value for each hidden unit is then calculated according to the formula in Equation 3 and the weights to the hidden units from the input units are altered using these delta values and Equation 1.

So error propagation, the concept behind the generalized delta rule, is really quite simple. The interesting aspect of the formulation for error propagation developed by Rumelhart, Hinton & Williams (1986) is that it can be shown to perform gradient descent in the error space which was described earlier. The proof goes as follows:

Since we want to prove that the GDR does gradient descent in error space, we have to show that

$$\Delta_p w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}}.$$

where E is the sum-squared error function defined earlier. This equation mathematically embodies the idea behind gradient descent in error space that the more changing a weight in a given direction increases the system's error, the more that weight should be changed in the opposite direction. This concept will be explained more fully after the derivation.

It is helpful to view the above derivative as the result of the product of two parts: one reflecting the change in error as a function of the change in net input to the unit and the other part reflecting the effect of changing a particular weight on the net input. Thus we can write

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \qquad (4)$$

Recall earlier we defined

$$net_{pj} = \sum_k w_{jk}\, o_{pk}$$

for all u_k with connections to u_j. So

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk}\, o_{pk} = o_{pi} \qquad (5)$$

Now for hidden units let us define

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} \qquad (6)$$

By substituting Equations 5 and 6 into Equation 4, we get

$$\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj}\, o_{pi}.$$

Therefore, to implement gradient descent, we should make changes in the network's weights such that

$$\Delta_p w_{ji} = \eta\ \delta_{pj}\, o_{pi}.$$

This is in fact, the exact formula the GDR uses to alter connection strengths. Now all that remains to complete the proof is to show that Rumelhart's definition of δ_{pj} is such that

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}.$$

Again it is useful to consider the above derivative as the result of the chain rule. That is,

$$\frac{\partial E_p}{\partial net_{pj}}$$

is the product of a factor representing how much the activation level of u_j affects the network's error on input/output pattern p and how much the net input to u_j affects the activation level of u_j. In other words,

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \qquad (7)$$

The second factor is easy. The derivative of the effect the net input to u_j has on u_j's activation level is just the derivative of the activation function. That is,

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f'_j(net_{pj}) \qquad (8)$$

where $f'_j(net_{pj})$ is merely the derivative of the logistic activation function described earlier.

The first factor in Equation 7 must be considered in two cases. First, if u_j is an output unit, the derivative of that unit's effect on the network's error is easy to compute because u_j contributes directly to the network's error. Therefore, the derivative of u_j's effect on the network's error on input/output pattern p is merely the derivative of the error function with respect to o_{pj}. In other words,

$$\begin{aligned}\frac{\partial E_p}{\partial o_{pj}} &= \frac{\partial}{\partial o_{pj}}\left(\frac{1}{2}\sum_j (t_{pj} - o_{pj})^2\right) \\ &= -\,(t_{pj} - o_{pj}) \qquad (9)\end{aligned}$$

Substituting Equations 8 and 9 into Equation 7 results in

$$\delta_{pj} = (t_{pj} - o_{pj})\, f'_j(net_{pj})$$

when u_j is an output unit. Not surprisingly, this is the exact equation the GDR uses to compute the delta value for output units.

The case in which u_j is a hidden unit is not so simple because the error function does not depend directly on the activation level of u_j. This means the first factor in Equation 7 is difficult to determine. But we do know that the error function depends directly on the activation level of the units that the hidden unit u_j connects to, since in three-layered systems, hidden units are only connected to output units.[2] We also know how each u_j affects each of the hidden units to which it is connected. Using these two pieces of knowledge the derivative of hidden unit u_j's effect on the network's error can be derived. Specifically,

$$\begin{aligned}\frac{\partial E_p}{\partial o_{pj}} &= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} \\ &= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ki}\, o_{pi} \\ &= \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{kj} \\ &= -\sum_k \delta_{pk}\, w_{kj} \qquad (10)\end{aligned}$$

Therefore, substituting Equations 8 and 10 into Equation 7, results in the equation

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk}\, w_{kj}.$$

Again this is the exact equation the GDR uses to calculate the delta value for a hidden unit u_j. Since the formulas for computing δ_{pj} in the GDR are exactly those needed to guarantee that

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$$

for both output and hidden units, we know that this equation holds true for all units in a network employing the GDR.

[2]This statement is somewhat misleading, since the recursive derivation of δ_{pj} for hidden units does not depend on the fact that a hidden unit is connected only to output units. The derivation proves that the generalized delta rule will perform gradient descent in networks with more than three layers, as long as units in each layer are only connected to units in the layer above, i.e. to the layer one closer to the output units.

Therefore the GDR changes weights in accordance with the formula

$$\Delta_p w_{ji} = \eta \; \delta_{pj} \, o_{pi}$$

with

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}.$$

Since it was proven earlier that if weights are changed in accordance with this formula, a network will perform gradient descent in error space, we know that the GDR will perform gradient descent in error space.

Actually this is not entirely correct. What has been proved is that the GDR will change weights on each pattern in the direction of the steepest downward gradient <u>in that pattern's error space</u>. If however, the network only makes small changes to weights after presentation of each pattern p, then the GDR will closely approximate gradient descent in the total error space defined earlier as

$$E = \sum_p E_p.$$

This point is very important in two respects. First of all, it is an important limitation on the effectiveness of the GDR for reasons that will be discussed shortly. Second, the concept of changing a network's weights after presentation of any one input/output pattern in the direction of the steepest downward gradient on that pattern's error surface is a crucial idea that will have major importance later in the discussion of the MGDR.

In fact, the idea of gradient descent on an error surface is of such importance that it deserves a pictorial explanation. Consider Figure 2-5 to be a graphical representation of a network's total error surface. In other words, consider Figure 2-5 to be E, the summation of a number of error surfaces from individual trials. Recall that in Figure 2-5 a method for getting from point A, the global maximum on the total error surface, to point B, the global minimum on the total error surface, was desired. The GDR is such a technique and it works in the following manner. Starting at point A, after presentation of each input/output pattern the GDR changes the weights in the network a small amount in the direction of the steepest downward slope *on the error surface for that individual input/output pattern.*

Consider changing one of the two weights, call it *weight*$_1$, that define the x and y axes of Figure 2-5. Changing *weight*$_1$ corresponds to moving along the error surface in the direction of one of the two planar axes in weight space. If changing *weight*$_1$ on a given input/output presentation by a given amount in a given direction decreases the error by twice the amount resulting from changing *weight*$_2$ by the same amount, then the GDR dictates changing *weight*$_1$ by twice the amount *weight*$_2$ is changed. In this way the system will change the weights in proportion to each weight's ability to reduce the error, leaving unchanged the weights that have little effect on the error for this particular input/output pattern. Intuitively, this is an important attribute for a learning rule because if a rule made a relatively large change to a weight which has little bearing on the error for this particular input/output pattern pair, the learning rule would very likely disturb the delicate weight system developed to solve some other input/output pairing in which this particular connection is very important. In short, if a given connection is not important for a given input/output pattern pair, leave it alone because it might be important for some other pair.

The fact that the GDR performs gradient descent in total error space is crucially dependent on the fact that each weight is changed during each pattern presentation in proportion to its ability to decrease the network's error for that particular pattern. In this way, if an increase to a given weight reduces the error for a given pattern by a relatively large amount, while a decrease in that same weight will decrease the error on another pattern by only a relatively small amount, the net change in the weight of this particular connection over the two input/output pattern pairs will be positive. This is exactly the result desired because it insures that network's weights will be altered in proportion to how much a given alteration decreases the error for the system as a whole over all patterns.

These are the appealing mathematical underpinnings of the GDR and gradient descent. However this technique for minimizing error is not as effective as it might first appear. In fact it has two major problems, one resulting from the particular implementation of gradient descent by the GDR and one resulting from a limitation of gradient descent in general.

The first problem results from the fact mentioned earlier that the GDR does not perform perfect gradient descent on the total error surface because weights in the network are altered after each pattern presentation. Consider Figure 3-2 in which the steepest gradient in the total error space from point A is toward point B. If upon presentation of the first pattern, the gradient of that particular pattern's error space dictates changing the network's weights not in the direction of point *B*, but instead in the direction of point *C*, the GDR is in trouble. The rule is in trouble because by moving towards *C* after this first presentation, the network enters an area of the error surface in which the steepest gradient is away from the global minimum, point B, and towards a local minimum, point *C*. As a result the network will never reach point *B*; but will remain stuck at *C* with a relatively large amount of error.

One possible solution to this problem is "off-line" learning in which the weight changes dictated by each pattern are summed before any weight changes are actually made. Although this method will guarantee gradient descent in the total error space, it is somewhat implausible from a biological standpoint. Rumelhart, Hinton & Williams (1986) use a different method to minimized this problem with the GDR. First, they insure that the changes made to weights on any given pattern are relatively small by using small learning rate constants (the η in the Equation $\Delta_p w_{ji} = \eta\ \delta_{pj}\, o_{pi}$). This minimizes the likelihood of entering an area of error space with a totally different gradient on a single pattern presentation, but at the same time it greatly reduces the speed at which the network can move towards the solution. In order to maintain a relatively rapid rate of progress down the gradient while avoiding wild weight oscillations on each pattern presentation, Rumelhart, Hinton & Williams (1986) employed a momentum term in their learning rule.

Momentum in the GDR was accomplished by altering the learning rule to:

$$\Delta w_{ji}(p + 1) = \eta\ \delta_{pj}\, o_{pi} + \alpha\, \Delta w_{ji}(p),$$

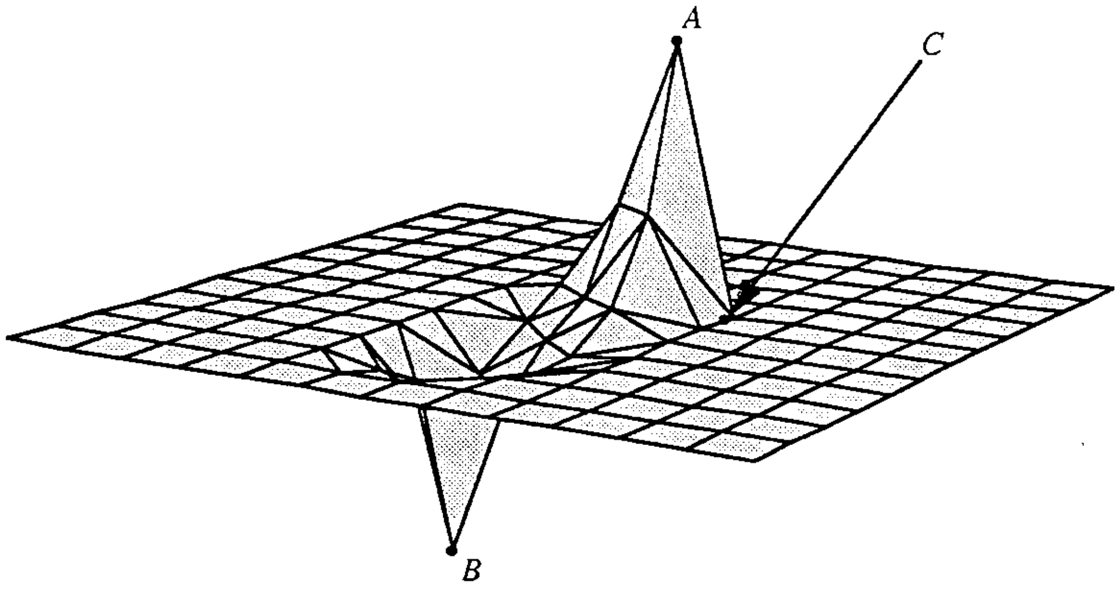

Figure 3-2. Another sample error surface in weight space. If the network starts at point *A* and on the first I/O pattern moves in the direction of point *C*, even though the steepest gradient over all I/O patterns may be in the direction of *B*, then the network will continue in the direction of *C* and become trapped at this local minimum.

where the index p refers to the pattern presentation number and α is a constant which determines the effect of past weight changes on the current change to be made. In the MGDR a momentum term is also frequently used. In fact, unless otherwise noted, it can be assumed that a momentum term was used with an α of 0.90. The findings in this research and in that of Rumelhart's group have been that a momentum term brings the network much more quickly to the same solution that is reached using no momentum term and a much smaller learning rate constant.

Despite the fact that the small learning rate constant coupled with a momentum term reduces the problem of wild oscillation while maintaining good progress towards the solution, the GDR is still plagued by slow learning. Just how slowly Rumelhart's GDR learns will be elaborated on in Chapter 5 where simulations using the MGDR are compared with the simulations using the GDR.

The tendency of networks employing gradient descent to become stuck at local minima is a problem that is not so easily avoided. The problem results from the fact that moving from the present point on the error surface in the direction with the steepest downward slope does not guarantee that the network is headed towards the point on the error surface with minimum error. All that gradient descent insures is that by moving in the dictated direction the total error in the network will be reduced from its current level. This concept is clearly illustrated in Figure 3-3, where the steepest gradient from point A is in the direction of point C. The GDR will cause the network to move in this direction until it reaches point C. At point C, all weight changes in the network will stop because there is no longer a downward slope for the network to follow. Thus the network will never reach point B, but will instead be stuck with a relatively large amount of total error.

Rumelhart, Hinton & Williams (1986, pp. 324) suggest that "the apparently fatal problem of local minima is irrelevant in a wide variety of learning tasks," but it will be seen that the problems of slow learning and local minima are major shortcomings of the GDR which the MGDR appears to effectively overcome.

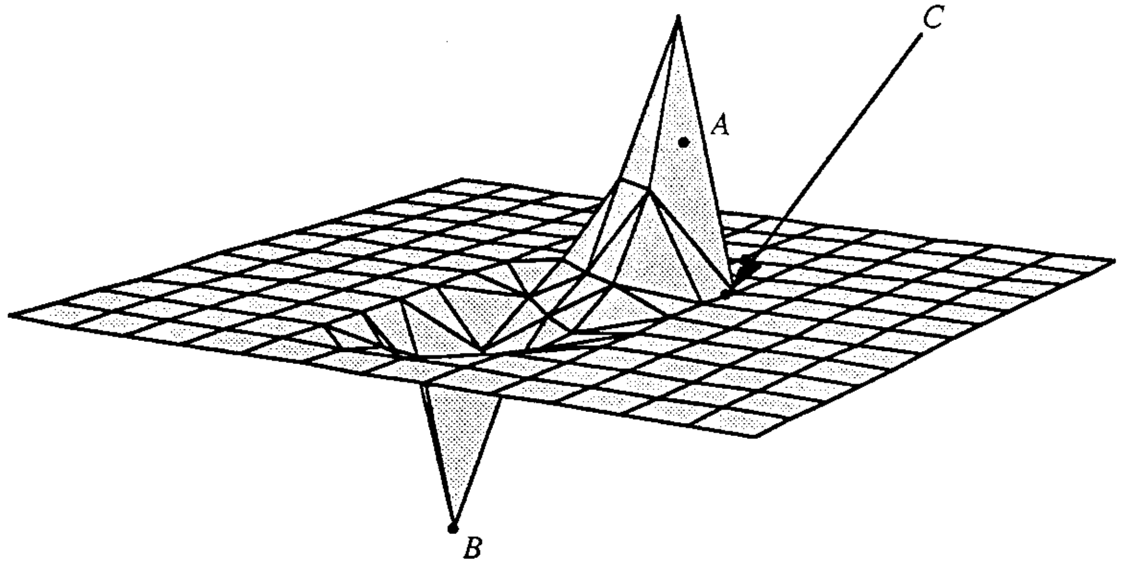

Figure 3-3. Another sample error surface in weight space. If the network starts at point *A*, gradient descent will move the network to point *C*, a local minimum, instead of point *B*.

Chapter 4
The Meta-Generalized Delta Rule

To solve the problems inherent in straight gradient descent and the generalized delta rule we have developed the concept of a meta-connection. A meta-connection is a connection from a unit to the connection between two other units. Meta-connections have the ability to alter the weight of the connection to which they connect in a manner somewhat similar to that of sigma-pi units[1] (Rumelhart, Hinton & McClelland, 1986). The interaction between meta-connections and normal connections is most easily understood through an example.

Consider Figure 4-1. In this simple network, there is a normal connection between u_i and u_j denoted c_{ji}. There is also a meta-connection from u_k to c_{ji}. Such a meta-connection will be referenced as $c_{(ji)k}$. The underlying concept behind meta-connections involves treating each normal connection as a unit, with the connection's "activation level" being the weight of that connection at a given moment. The total weight of c_{ji} on a given I/O pattern presentation, denoted $W_{p(ji)}$ is just the normal weight of the connection, w_{ji} plus a nondecreasing function, f, of the net input to c_{ji} from meta-connections. Specifically,

$$W_{p(ji)} = w_{ji} + \sum_k w_{(ji)k} \sqrt{o_{pk}}$$

for all u_k with meta connections to c_{ji}, where $w_{(ji)k}$ denotes the weight of the meta-connection from u_k to c_{ji}. The necessity of the root in the above equation will later be shown to result from the desire to maintain a correspondence between the GDR and the MGDR.

[1]In chapter 6 an explicit comparison is made between meta-connections and sigma-pi units.

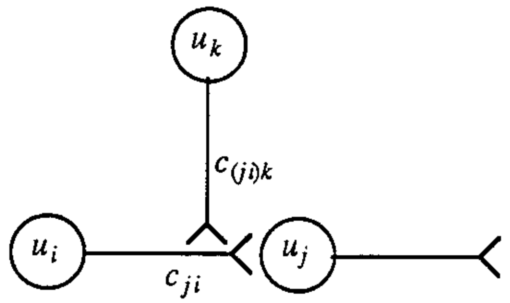

Figure 4-1. A meta-connection to the connection between two units.

As a concrete example, compare the solution to the XOR problem in a network without meta-connections in Figure 4-2 to the solution with meta-connections in Figure 4-3.[2] The solution without meta-connections involves two input units, one hidden unit and one output unit. Either of the two input units, when firing alone, is able to activate the output unit, despite the output unit's negative bias. The hidden unit in this network is basically a detector for the situation in which the two input units are active, in which case it inhibits the output unit.

The solution using meta-connections depicted in Figure 4-3 is quite different.[3] In this solution, as in the solution without meta-connections, each input unit, is able to activate the output unit when firing alone. But when both are firing, the meta-connections from unit *A* to the "*B-C*" connection turns off the "*B-C*" connection by decreasing its weight. In this way, when both input units are active, the net input to the output unit is zero, so the activation level of the output unit, with its negative bias, remains around zero.

In the the MGDR, the activation function is the same as in the GDR except it takes into account the new idea of the total weight of a connection $W_{p(ji)}$, being a function of the normal weight of the connection w_{ji}, plus a factor to account for the influence of the meta-connections to that connection. Specifically, the activation function, f_j, for a unit j, is

[2]The actual networks depicted in Figures 4-2 and 4-3 are somewhat misleading in that neither is in the 3-layer form discussed in the previous chapter since each has connections from input units to output units.
[3]There is actually a solution to the XOR problem using a single meta-connection from one unit to the connection between the other input unit and the output unit, but the solution in Figure 4-3 is easier to understand and is more analogous to the solution without meta-connections.

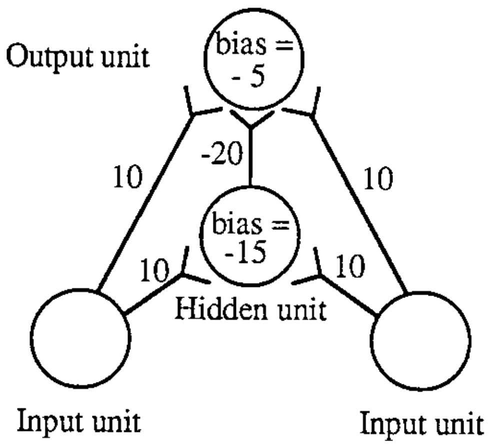

Figure 4-2. A solution to the XOR problem without meta-connections.

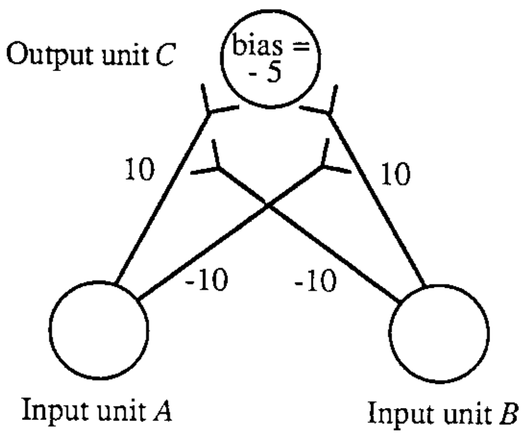

Figure 4-3. A solution to the XOR problem with meta-connections.

$$o_{pj} = \frac{1}{1 + e^{-\left(\sum_i W_{p(ji)}\, o_{pi} + \theta_j\right)}} \qquad (1)$$

where

$$W_{p(ji)} = w_{ji} + \sum_k w_{(ji)k} \sqrt{o_{pk}} \qquad (2)$$

The delta values for units are computed using the method described by Rumelhart, Hinton & Williams (1986) in their derivation of the generalized delta rule, except that again we use the concept of total weight $W_{p(ji)}$ where they used the normal weight w_{ji}. Specifically, when *j* is an output unit,

$$\delta_{pj} = (t_{pj} - o_{pj})\, f'_j(net_{pj}) \qquad (3)$$

where

$$net_{pj} = \sum_i W_{p(ji)}\, o_{pi} + \theta_j \qquad (4)$$

and $f'_j(net_{pj})$ is the derivative of the activation function for unit *j* with respect to its net input, net_{pj}. Therefore,

$$\delta_{pj} = (t_{pj} - o_{pj})\, o_{pj}\, (1 - o_{pj}) \qquad (5)$$

since

$$f'_j(net_{pj}) = o_{pj}\, (1 - o_{pj}). \qquad (6)$$

When *j* is not an output unit,

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk}\, W_{p(kj)}. \qquad (7)$$

In the MGDR, $\delta_{p(ji)}$, the deviation of the weight of c_{ji} from what it should be, is

$$\delta_{p(ji)} = \delta_{pj}\, o_{pi}. \qquad (8)$$

The value of $\delta_{p(ji)}$ is proportional to the total weight change that c_{ji} would have experienced in the normal generalized delta rule. Therefore, $\delta_{p(ji)}$ can be thought of as the desired change in the total connection strength, $W_{p(ji)}$. This desired change in total connection strength is distributed to both the pure weight of c_{ji} and the meta-connections connecting to c_{ji} through the following equations:

$$\Delta_p w_{ji} = \eta \; \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}} \tag{9}$$

and

$$\Delta_p w_{(ji)k} = \eta_m \; \delta_{p(ji)} \frac{\sqrt{o_{pk}}}{net_{p(ji)}} \tag{10}$$

where

$$net_{p(ji)} = o_{pi} + \sum_k o_{pk}$$

for all units k with meta-connections to c_{ji} and η_m is the learning rate constant for meta-connections.

These equations differ substantially from the weight changing equations for the GDR, and hence they deserve further explanation. The driving factor behind the differences is the goal that the total weight change to a given connection between units ($W_{p(ji)}$) on any given trial equal the change dictated by the GDR. But in the MGDR this total change is divided between alterations to the normal connection between two units and alterations to the meta-connections to that normal connection. The quantity $net_{p(ji)}$ is a measure of the "total activation" projecting to a connection between two units and it is used for this division process. In the next section it will be shown that the root in Equation (10), when coupled with the root in Equation (2), enables the MGDR to achieve the goal as stated above.

To demonstrate that the MGDR works towards a proper set of weights, it is useful to show that it alters a network's weights in the direction of the steepest gradient of the surface in weight space whose height at any point is equal to the error measure. To show the learning system performs gradient descent, it is merely necessary show to that

$$\Delta_p W_{p(ji)} \propto \delta_{pj}\, o_{pi}, \qquad (11)$$

since we showed earlier that if the weight changes are proportional to $\delta_{pj}\, o_{pi}$, using δ_{pj} as defined above, then the learning system will perform gradient descent. From Equation 2, we know that

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} + \sum_k \Delta_p w_{(ji)k} \sqrt{o_{pk}}. \qquad (12)$$

But from Equations (9) and (10) we know that

$$\Delta_p w_{ji} = \eta\ \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}}$$

and

$$\Delta_p w_{(ji)k} = \mu\ \delta_{p(ji)} \frac{\sqrt{o_{pk}}}{net_{p(ji)}}.$$

Therefore,

$$\begin{aligned}\Delta_p W_{p(ji)} &= \eta\ \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}} + \sum_k \mu\ \delta_{p(ji)} \frac{\sqrt{o_{pk}}}{net_{p(ji)}} \sqrt{o_{pk}} \\ &= \eta\ \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}} + \sum_k \mu\ \delta_{p(ji)} \frac{o_{pk}}{net_{p(ji)}}.\end{aligned}$$

If we let the two learning rate constants be equal, (that is $\eta = \mu$) we get

$$\Delta_p W_{p(ji)} = \left(o_{pi} + \sum_k o_{pk}\right) \eta\ \frac{\delta_{p(ji)}}{net_{p(ji)}}.$$

But since

$$net_{p(ji)} = o_{pi} + \sum_k o_{pk},$$

we know that

$$\begin{aligned}\Delta_p W_{p(ji)} &= \frac{net_{p(ji)}}{net_{p(ji)}}\ \eta\ \delta_{p(ji)} \\ &= \eta\ \delta_{p(ji)}.\end{aligned}$$

Also, recall from Equation 8 that

$$\delta_{p(ji)} = \delta_{pj}\, o_{pi}.$$

Therefore,

$$\Delta_p W_{p(ji)} = \eta\ \delta_{pj}\, o_{pi}.$$

This means that the total change to the weight of a connection from unit *i* to unit *j* upon the presentation of input-output pair *p* will equal the product of the learning rate constant, a measure of the deviation of the activation of unit *j* from what it should be and the activation level of unit *i*. This total weight change, $\eta\ \delta_{pj}\, o_{pi}$, is not only proportional to $\delta_{pj}\, o_{pi}$, but it is in fact equal to the weight change the simple connection from unit *i* to unit *j* would have experienced in Rumelhart's generalized delta rule. The only difference is that in the MGDR this weight change is divided between the "normal weight" of the connection from unit *i* to unit *j* and the meta-connections connecting to the connection from unit *i* to unit *j*. This completes the proof that with equal learning rate constants for normal and meta-weights, the weight alterations made to a network after presentation of a given I/O pattern pair using the MGDR will move the network in the direction of the steepest downward gradient *on that particular I/O pattern's error surface*. How the MGDR performs in the total error space will be discussed later in Chapter 6.

An important point to be proven is that, in a network with no meta-connections, the MGDR will behave exactly like Rumelhart's GDR, making the identical weight changes. The total change to a given weight on a given trial is

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} + \sum_k \Delta_p w_{(ji)} \ \sqrt{o_{pk}}$$

from Equation 12.

Since the network has no meta-connections, the equation simplifies to

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji}.$$

From Equation 9,

$$\Delta_p w_{ji} = \eta\ \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}}.$$

Therefore,

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} = \eta\ \delta_{p(ji)} \frac{o_{pi}}{net_{p(ji)}}.$$

But recall,

$$net_{p(ji)} = o_{pi} + \sum_k o_{pk}$$

for all units *k* with meta-connections to the connection from unit *i* to unit *j*.

Since there are no meta-connections,

$$net_{p(ji)} = o_{pi}\ .$$

Therefore,

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} = \frac{net_{p(ji)}}{net_{p(ji)}} \eta\ \delta_{p(ji)}$$

$$= \eta\ \delta_{p(ji)}.$$

But recall from Equation 8,

$$\delta_{p(ji)} = \delta_{pj}\, o_{pi}.$$

Therefore,

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} = \eta\ \delta_{pj}\, o_{pi}.$$

In other words, the weight change in the connection from unit *i* to unit *j* is $\eta\ \delta_{pj}\, o_{pi}$, which is the exact change made by the GDR.

To summarize, two important theorems about the meta-generalized delta rule have been proved in this chapter. First, its been shown that the MGDR is equivalent to the GDR in a network without meta-connections. Second, we have shown that a network with meta-connections employing the MGDR behaves in a manner somewhat analogous to a network without meta-connections in that, for each I/O pattern pair, the weights in the network are altered to move the network in the direction of the error surface's steepest gradient. In the next two chapters, the important differences between the MGDR and the GDR are illustrated both empirically and analytically.

Chapter 5
Simulation Results

The major goal behind the development of the meta-generalized delta rule was to create a weight-learning scheme for connectionist networks which avoids or at least minimizes the problems inherent in the generalized delta rule. Specifically, the MGDR was designed to reach a solution more quickly and to become trapped in local minima less frequently. A second important characteristic of the GDR that we hoped to maintain or expand upon in the MGDR was the ability to develop solutions to problems which generalize to new situations. In other words, it was hoped that the MGDR would develop solutions which produce the correct output pattern for input patterns the network has never encountered before. Our results indicate that the MGDR successfully accomplishes these goals. This section describes the simulation results which support this claim.

All simulations were done using the rules described for the MGDR. When simulating the GDR, the meta-connections were merely removed from the network. As proved in Chapter 4, this makes the MGDR equivalent to the GDR. The networks employed in the simulations were all three-layered networks as described in Chapter 2. All the important parameters are discussed in the results for each individual simulation except for the fact mentioned in Chapter 3 that a momentum term with an α value of 0.90 was used in all the simulations.

Also, recall that the logistic activation function prohibits activation levels of 0.0 or 1.0 without infinitely large weights. In the simulations where the desired outputs are binary (0 or 1), the system can never achieve these values. Therefore, the values 0.1 and 0.9 have been used as the targets.

Finally, since the networks used to test the MGDR involved varying degrees of connectedness, it is important to understand the concept of meta-connectivity. In the networks tested, meta-connections only originated from input units and terminated on normal (non-meta) connections which did not originate from the same input unit as the meta-connection. Therefore, a network with 100% meta-connectivity is a network with a meta-connection from each input unit to every non-meta connection in the network except those originating from the same input unit. Figure 5-1 illustrates a fully-connected network with two input units, one hidden unit and one output unit. Notice there are no meta-connections originating from the hidden unit. On the problems tested, these extra meta-connections only served to increase the network's complexity and did not increase the network's learning speed or ability to avoid local minima. In a network with 50% meta-connectivity, each meta-connection from the fully-connected network has a 0.5 probability of being present.

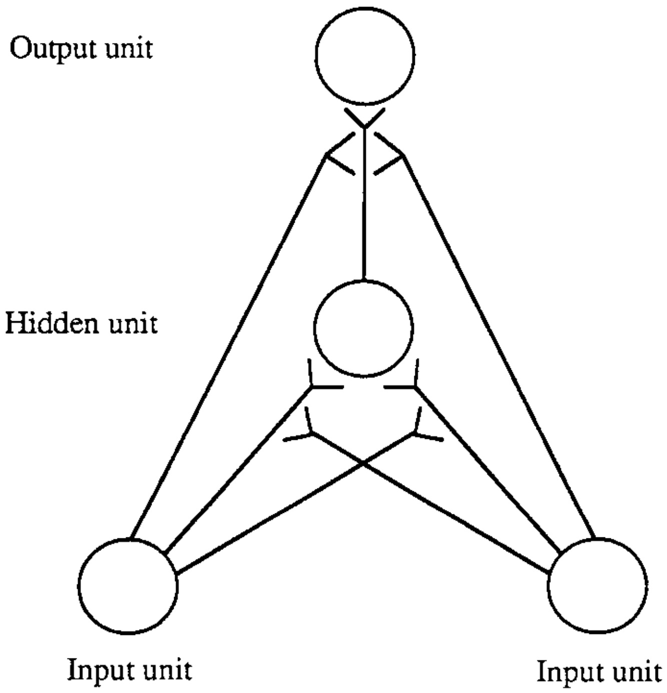

Figure 5-1. A fully-connected network with two input units, one hidden unit and one output unit.

The MGDR was first tested on the XOR problem discussed earlier. This is considered by Rumelhart to be the "classic" problem for connectionist networks because it is one of the simplest which requires hidden units (at least for the networks without meta-connections) and because many other difficult problems include XOR as a subproblem (Rumelhart, Hinton & McClelland 1986). The XOR problem was run many times, varying the number of hidden units from 0 to 8, the normal learning rate constant from 0.1 to 1.0, the meta-learning rate constant from 0.5 to 4.0 and the degree of meta-connectivity from 0% (the GDR) to 100%. A representative sample of the results obtained on the XOR problem is depicted in Figure 5-2.

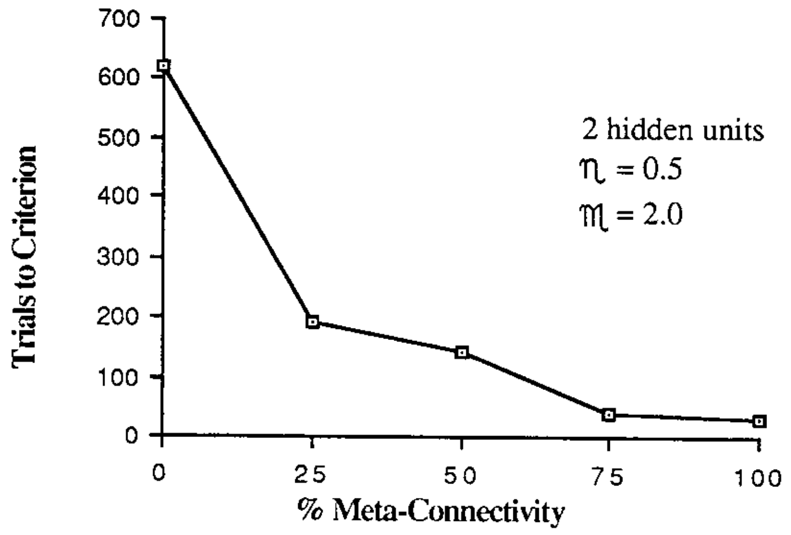

Figure 5-2. Trials to criterion on the XOR problem for networks of varying degrees of meta-connectivity. This graph includes trials upon which the network became trapped in local minima.

The way the values for the learning rate constants in this and all the examples in this chapter were determined was through the following procedure. First, a particular network with no meta-connections was tested to determine how quickly it could learn a solution to the problem using a number of different normal learning rate constants and various random initial weight configurations.[1] Using these results, the optimum learning rate constant for the network without meta-connections was determined. The optimum learning rate constant was

[1]The initial weights in all tests were randomly distributed between -0.5 and 0.5.

defined to be the value at which the network learned most quickly as averaged over all the various random initial weight configurations.[2]

Then, using the optimum learning rate constant, the network was tested to determine how quickly it could learn using a number of meta-learning rate constants and various degrees of meta-connectivity. One interesting discovery was that networks with meta-connections generally learned more quickly if the meta-learning rate constant, $m\eta$, was larger than the normal learning rate constant, η. This finding, which seems to contradict the earlier idea that $m\eta$ should be set equal to η for the network to reach a solution quickly through gradient descent, is discussed in more detail in Chapter 6. In tests with meta-connections, the initial normal weight configurations were identical to the configurations used to test the network without meta-connections. In addition, the meta-connection weights were initially set to 0.0.

The number of random initial weight configurations and random meta-connectivity patterns on test configurations with partial meta-connectivity varied across test problems. Solutions to the XOR problem were learned relatively quickly, and as a result, it was possible to test a relatively large number of initial configurations. Each point in the graph of Figure 5-2 therefore represents the mean number of trials over 25 random initial configurations for a network with two hidden units, a normal learning rate constant of 0.5 and a meta learning rate constant of 2.0 to solve the XOR problem.

As will be seen in all the test problems, a network with meta-connections can reach a solution much more quickly than a network without them. In the XOR problem in fact, the network with 100% meta-connections was able reach a solution over 20 times faster than the network employing the GDR and no meta-connections. This comparison is somewhat misleading however, because the system with no meta-connections became stuck in local minima more frequently than did networks with meta-connections, hence shifting upward the

[2]Occasionally, especially in networks without meta-connections, the network would become stuck in a local minimum. In these cases, a large, arbitrary cutoff point was used as the number of trials to reach a solution. In the case of XOR, the cutoff point was 2000 trials.

averaging in trials upon which the GDR became trapped in local minima, the networks with meta-connections were able to solve the problem much more quickly.

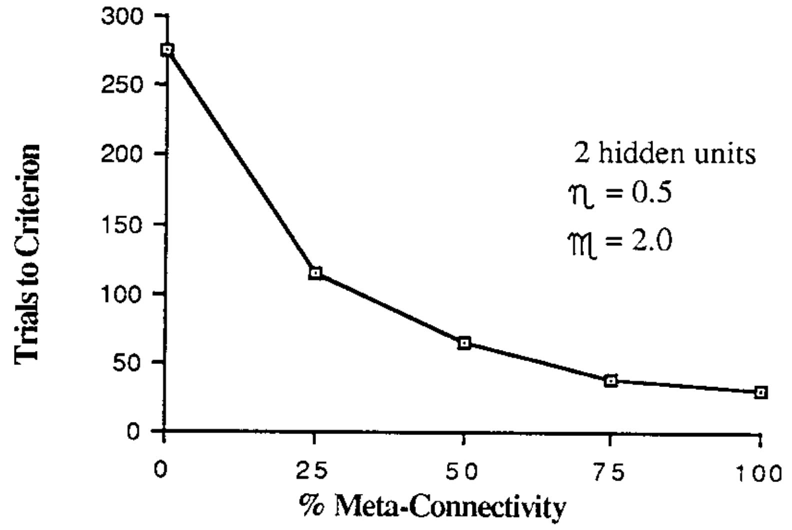

Figure 5-3. Trials to criterion on the XOR problem for networks of varying degrees of meta-connectivity. This graph does not include trials upon which the network became trapped in local minima.

In addition, the general trend in this and the other examples tested was that the higher the degree of meta-connectivity, the more quickly the system could reach a solution, almost entirely without regard to which meta-connections were present.

A second problem used to test the MGDR was the parity problem, in which the activation level of the one output unit should be 1.0 if the input pattern contains an odd number of active units and it should be 0.0 otherwise. The XOR problem is actually the parity problem with two input units. The parity problem with 3 input units was run thousands of times using the method for finding optimum learning rate constants described above. Figure 5-4 represents the findings for the problem in the network with three hidden units. The results are quite similar to those from the XOR problem: generally, the higher the meta-connectivity the faster the learning, although there was a slight upturn in the graph from 75 to 100% meta-connectivity. It is difficult to explain this slight decrease in learning speed in the 100% meta-connectivity condition, because obviously all the meta-connections present in the 75% network were present in the 100% network. One possible reason for this anomaly results from the fact that it is often the learning of weights for the normal connections, and especially biases, that slows down learning in networks with meta-connections.[3] What might have happened as the meta-connectivity increased from 75% to 100% was that the total weight change to a given connection became distributed over more meta-connections, and hence the normal weight for that connection changed more slowly. This decrease in learning speed of the normal weights may have been more than enough to overcome any speed increases due to higher meta-connectivity, since at 75% meta-connectivity the network has reached the point at which only asymptotic increases in learning speed are possible with higher meta-connectivity.

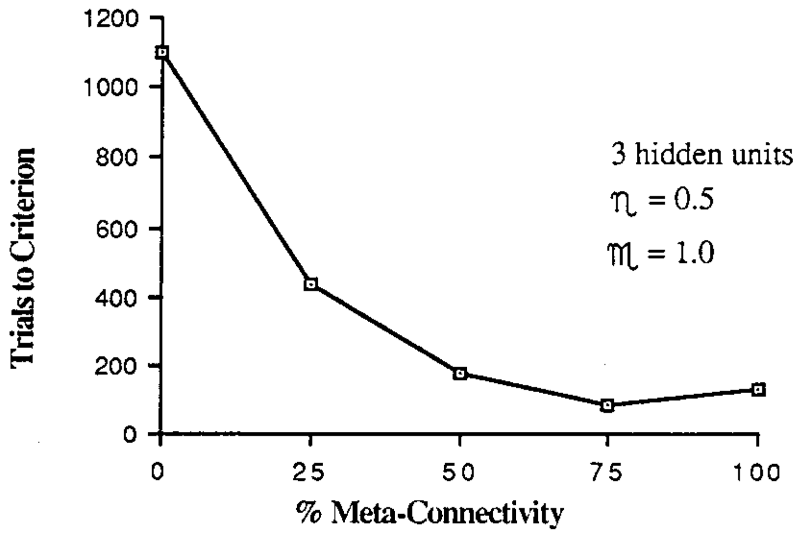

Figure 5-4. Trials to criterion on the 3 unit parity problem for networks of varying degrees of meta-connectivity.

We have also tested the MGDR on the parity problem with four input units. Unfortunately, computational limits prohibited the detailed analysis afforded the three unit

[3]In fact, it was almost always the case that networks with meta-connections were slowest to learn the pattern in which no input units were active. On this pattern, no meta-connections contribute to the weights of connections, and hence the system has to derive a solution using only biases and normal connections.

parity test.[4] However the general trend of increasing learning speed with increasing meta-connectivity continued for the network configurations that were tested.

The third test of the MGDR involved determining whether the input pattern was symmetric about its center. It is interesting that this problem can be solved for any number of input units by a network without meta-connections using only two hidden units (see Rumelhart, Hinton & Williams, 1986 for solution). Because of this simplicity, the network with four input units was a manageable test case.[5] Figure 5-5 depicts the results of the four unit symmetry tests. Again, the learning speed was an increasing function of the meta-connectivity.

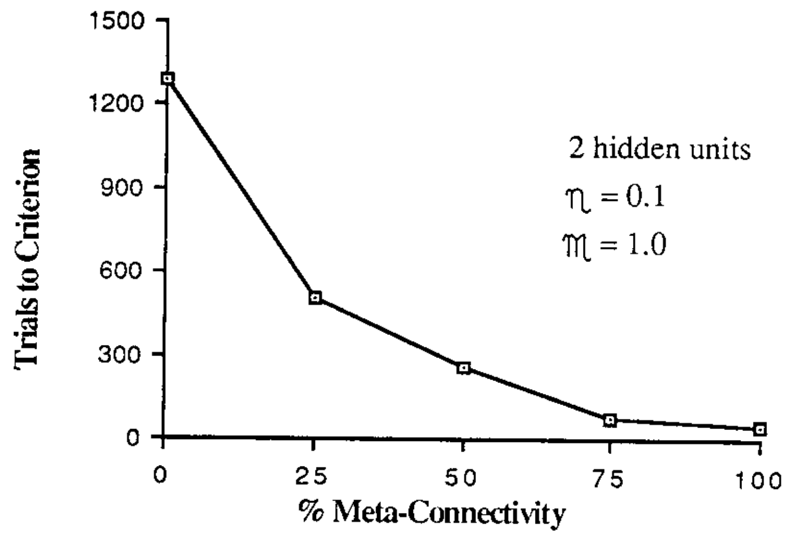

Figure 5-5. Trials to criterion on the 4 unit symmetry problem for networks of varying degrees of meta-connectivity.

One of the most interesting functions used to test the MGDR was the two bit addition problem. In this test, there were four input units, representing two two-bit binary numbers. The goal for the network was to activate the three output units such that, if interpreted as a three

[4]The problem is that both the number of input/output patterns and the number of meta-connections increases exponentially with the number of input units. These additions, when coupled with the fact that problems with more input units generally take more trials to solve, increased simulation running times at a prohibitively fast rate.

[5]A similar learning curve was obtained for the three input units symmetry problem, which in fact merely involves computing the AND of the two outside units and ignoring the middle one.

bit binary number, the output pattern would represent the sum of the two two-bit input numbers. The network required to learn the solution without meta-connections involves four hidden units, so obviously this is a rather complex network. As a result, only limited tests were computationally feasible for this problem. The results that were obtained are illustrated in Figure 5-6. Each point in the figure represents the mean number of trials to learn a solution for nine random network configurations at the given level of meta-connectivity. Again the graph slopes downward, indicating an increase in learning speed with increased meta-connectivity. The fact that this graph is somewhat more linear than those previous is probably a result of testing limitations. The network's complexity necessitated placing a limit on the number of trials during which each network configuration was allowed to learn to 2000. This resulted in a relatively large percentage (33%) of configurations in the 0% meta-connectivity condition being arbitrarily stopped after 2000 trials, when in fact they may have been able to develop a solution given more trials.

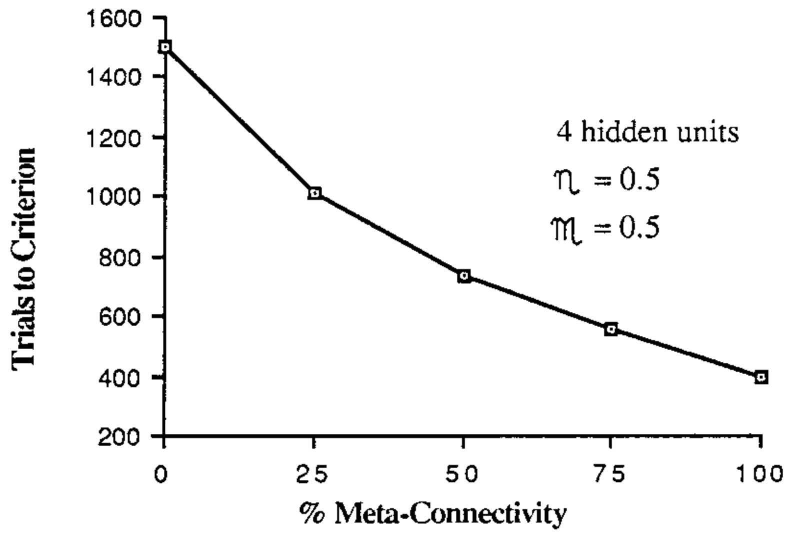

Figure 5-6. Trials to criterion on the 2 bit addition problem for networks of varying degrees of meta-connectivity.

The next problem tested using the MGDR was primarily done to explore the important ability of developing solutions which will generalize to patterns never encountered previously. The problem used to test this ability was the three unit encoder problem, in which the pattern in the input units must be encoded into two hidden units, which must in turn duplicate the input pattern in the output units. This pattern was chosen to test the MGDR's ability to develop generalizable representations, since intuitively it should be a problem amenable to such generalization. Specifically, the network needs to develop a system of weights which causes each input unit to activate the corresponding output unit. Such a network should easily produce the correct output pattern for any new input pattern.

The way the test for generalizability worked was to first randomly select six of the eight input/output pattern pairs and then train the network until it developed a solution for them. Then the two other patterns were presented to the network to determine how well the representation developed generalized to new patterns. The network's error in generalization was defined to be the average distance the activation level of each output unit was from the target level (1.0 or 0.0) upon presentation of the new input patterns.

The results are depicted in Figures 5-7 and 5-8. Figure 5-7 illustrates that the learning speed again increased rapidly with increasing meta-connectivity. Figure 5-8 is the more significant of the two graphs for the question at hand. Note that the ability of the GDR (the 0% meta-connectivity case) to develop representations for this problem which generalize was quite poor. In fact, its solutions don't generalize at all, since the average difference in activation level of the output units from their targets was 0.48 and one would expect a network with merely random connections to have an average output unit activation error of only 0.50.

Note also that as the meta-connectivity increased, the output activation error decreased to the point where, with 100% connectivity, the average error was less than 0.1, the level which, as indicated earlier, was the criterion level used to determine if a system had reached a solution. These findings indicate that, not only does the MGDR develop solutions more quickly than the straight GDR, but the solutions that it does develop are in a sense more useful since they can be used to solve problems not encountered previously.

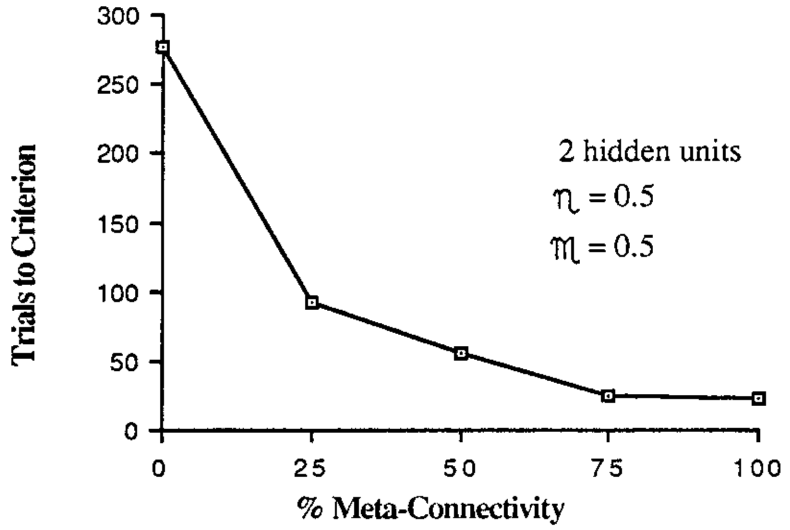

Figure 5-7. Trials to criterion on the 3 unit encoder problem for networks of varying degrees of meta-connectivity.

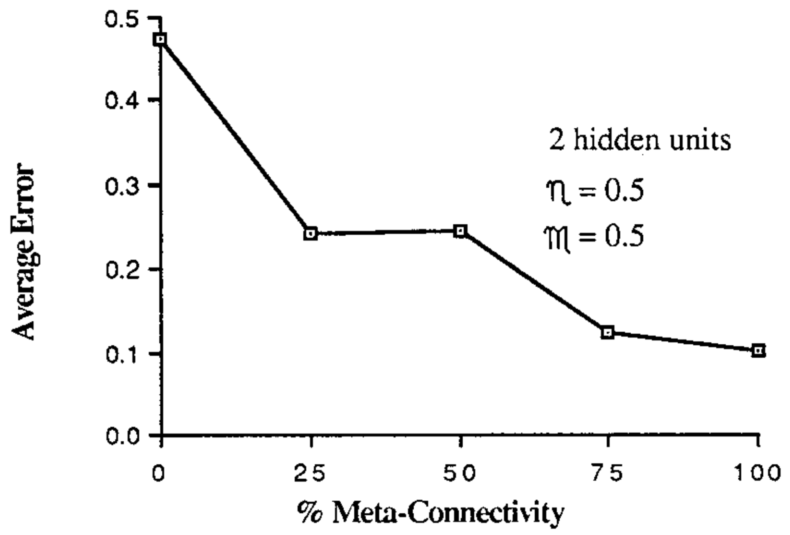

Figure 5-8. Average error per unit on the 3 unit encoder problem for networks of varying degrees of meta-connectivity. The average error was the mean distance the activation levels of the output units were from their target levels.

The final problem upon which we tested the MGDR was in direct contrast to the abstract mathematical nature of most of the problems discussed earlier. It is called the T/C problem and it involves learning to discriminate between the letters T and C, regardless of their translation and rotation.

The architecture employed to learn the T/C problem is slightly different from that of earlier models. In this problem, the input layer consisted of a 5 by 5 two-dimensional "retina". The input units were connected to a layer of hidden units which in turn projected to a single output unit. The input to the network consisted of a pattern representing either a T or a C with some amount of translation and rotation. The eight possible rotations of the T figure and the C figure are shown in Figure 5-9. A representative input pattern is depicted in Figure 5-10. The dark squares in Figure 5-10 represent input units with an activation level of 1.0 for this particular input pattern and the white squares represent input units with an activation level of 0.0 for this particular input pattern. The desired activation level of the output unit was 1.0 if the input pattern represented a T and 0.0 if the input pattern represented a C.

In total, the set of input patterns presented to the network on each trial consisted of the 84 possible retinal positions and rotations of Ts and Cs. Because of the large size of both the input pattern set and the network itself, extensive comparisons between the learning done by the MGDR and the GDR were impossible.[6] However Rumelhart, Hinton & Williams (1986) reported that a network employing the GDR requires between 5000 and 10,000 trials to learn the T/C problem.

Again the MGDR developed a solution to the problem in fewer trials than the GDR. In this case, using arbitrary values for the number of hidden units (4), the normal learning rate constant (0.5) and the meta-learning rate constant (1.0), a network with 100% meta-connectivity was able to develop a solution to the problem in 1350 trials.

[6]In fact, the solution alluded to using meta-connections required 27 hours of CPU time on a Sun 3/50 to learn.

In short, the findings in this and the earlier simulations suggest that the meta-generalized delta rule can be successfully employed in a wide range of problems to develop better solutions more quickly than with the generalized delta rule.

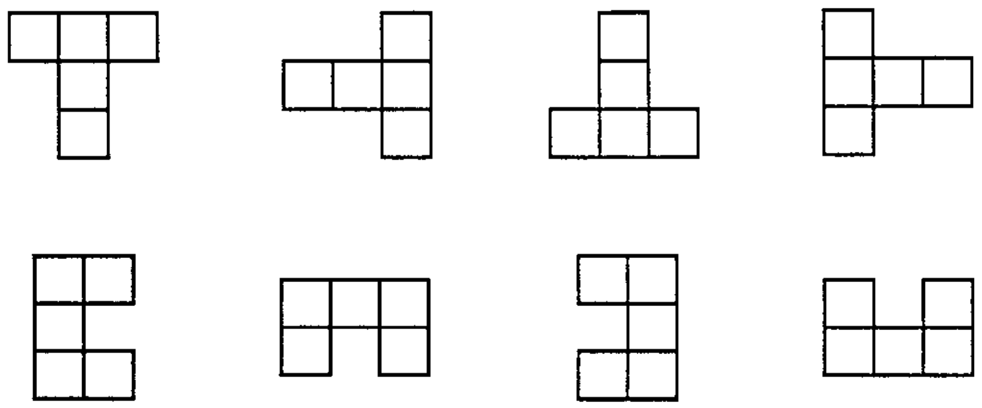
Figure 5-9. The eight rotations of the block letters T and C used as stimuli in the T/C problem.

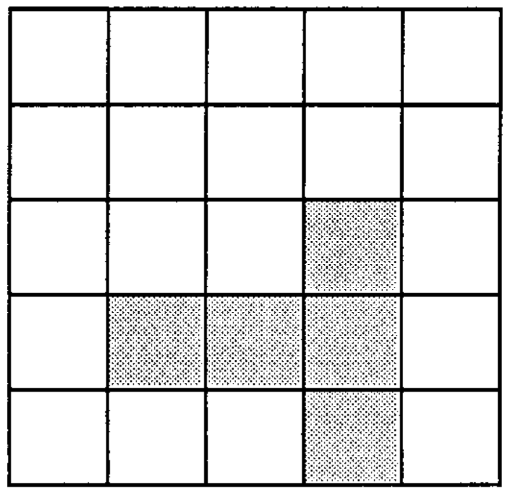
Figure 5-10. A sample input pattern from the T/C problem. The gray squares represent input units with an activation level of 1.0 and the white squares represent input units with an activation level of 0.0.

Chapter 6
Analysis of the MGDR

This chapter addresses two questions: "Why does the meta-generalized delta rule develop solutions so much more quickly than the generalized delta rule?" and "How does the meta-generalized delta rule compare with other extensions of the generalized delta rule?".

One way to conceptualize the advantages derived from meta-connections is by realizing that meta-connections allow the network to tailor the total weights of its connections for particular input/output patterns. Recall that the total weight of a particular connection depends on the present activation level of the input units, since meta-connections from active input units can influence the total weight of a connection. As a result, the network does not have to find a single set of weights to solve all the input/output patterns.

To understand how the MGDR tailors the total weights of individual connections for specific I/O patterns, consider the following situation. Suppose on a given I/O pattern, error minimization dictates that the weight of connection *c* should be negative, and suppose that on another pattern, the weight of *c* should be positive. In the GDR, this conflict will result in a small net change to the connection's weight in the direction that minimizes the total error over the two patterns. In a sense, the two I/O patterns fight against each other to determine the weight of *c*. This rivalry between patterns results in slow gradient descent in total error space.

But with meta-connections, the network is in some sense able to "remember" the proper weight of connection *c* for each input pattern. Since each I/O pattern has a different set of active input units, a difference in the total weight of *c* on the two input patterns can be achieved by altering the weights of meta-connections impinging on *c* from input units which are active in one I/O pattern but not in the other.

Perhaps the best way to understand how the MGDR tailors the weights of connections for the individual input/output pattern pairs of a problem is to reconsider the concept of the error surface discussed in Chapter 2. Recall that the error surface for a particular network is a surface in weight space whose height at any point is equal to the error measure. Also recall that the GDR develops a solution to a problem by performing gradient descent on the network's total error surface in which the height at any point in weight space equals the sum of the error from each input/output pattern. Gradient descent in total error space was guaranteed by the fact that on each I/O pattern, the change to each weight was proportional to that weight's contribution to that pattern's error. Therefore, across all patterns, the net change to a given weight would be in the direction which minimizes the total error.

The MGDR behaves differently. Instead of performing gradient descent on the total error surface, the MGDR performs a modified form of gradient descent in the error space for each individual I/O pattern pair. Since meta-connections allow tailoring of the total weights of connections in the network, they allow the network to explore different regions of the error surface on different I/O patterns.

This phenomenon is best seen through an example. Suppose we had a very simple network with two weights and a very simple problem with two I/O patterns. Suppose further that planes depicted in Figures 6-1a and 6-1b are the error surfaces for the two I/O patterns. The resulting total error surface would be the plane shown in Figure 6-1c. Starting at point A in weight space, the GDR dictates that on the first presentation of pattern 1, the network weights should be changed so as to move the system in the direction depicted by vector v_1 in Figure 6-1a. After moving in this direction, the network will be at point B in weight space. Upon presentation of pattern 2, the network will move in the direction of vector v_2 to point C (see Figure 6-1b). As can be seen from Figure 6-1c, the net result of these two pattern presentations is a change to the network's weights in the direction of vector v_3, the steepest gradient in total error space.

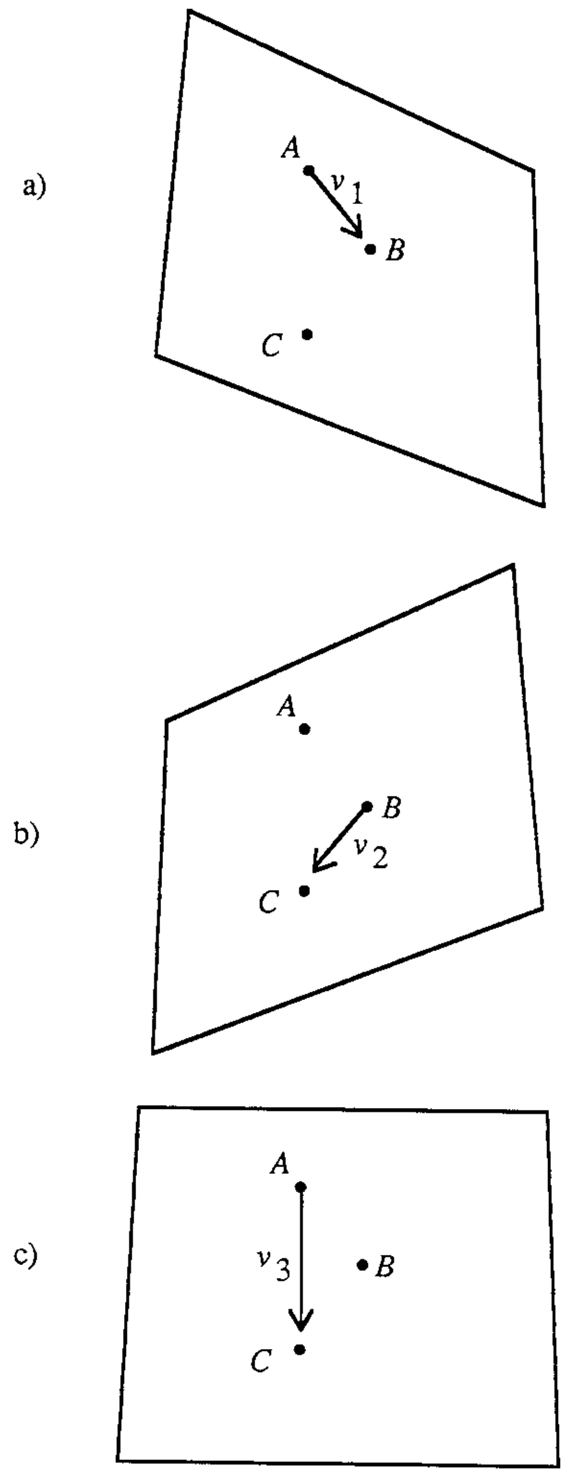

Figure 6-1. An illustration of gradient descent on an error surface in weight space. Starting at point A, the network follows the steepest gradient on the first I/O pattern and moves in the direction of v_1 to point B. Then, on the second I/O pattern, the steepest gradient is in the direction of v_2, so the network moves to point C. Notice that the movement in total error space, illustrated in part c, is in the direction v_3, the steepest gradient in total error space.

The MGDR works somewhat differently. Upon presentation of input/output pattern 1, the total change in the weights of the network's connections would be the same as in the GDR, so the network would move from point *A* to point *B* in error space (see Figure 6-2a). But some of that movement would result from changes in the weights of normal connections and some of it would result from changes in the weights of meta-connections.

Therefore, upon presentation of I/O pattern 2, which has a different set of active input units and hence a different set of active meta-connections, the network will not be at point *B*. Instead, the difference in meta-connection influence between the two patterns will place the network at some other location in error space, call it point *D* (see Figure 6-2b). From here, the MGDR dictates movement in the direction of vector v_2, to point *E*. It is not appropriate to determine the progress the network has made towards a solution by examining movement in total error space, because there is no longer a single point in total error space corresponding to the total weights of the current network. Instead, we have a network whose total weights, and hence whose position in error space, depends on the pattern of activity in the input units.

The only way to determine progress towards a solution is to look at the network's new position in error space on each of the individual input/output patterns. Upon the next presentation of input pattern 1, the network will no longer be at point *B*, since the normal weights and some of the meta-weights altered originally in moving from point *A* to point *B* have been altered again during processing of input/output pattern 2. But some of the meta-connections altered in moving from *A* to *B* may not have been active on pattern 2. It is primarily through these non-overlapping meta-connections that the network can "remember" some of the change made on the last presentation of I/O pattern 1. The net result will be that on the second presentation of pattern 1, the network will be at a point in error space in the vicinity of point *B*, call it point *F* (see Figure 6-2c). From this position, the network will again move in the direction of the steepest gradient, v_1, to point *G*.

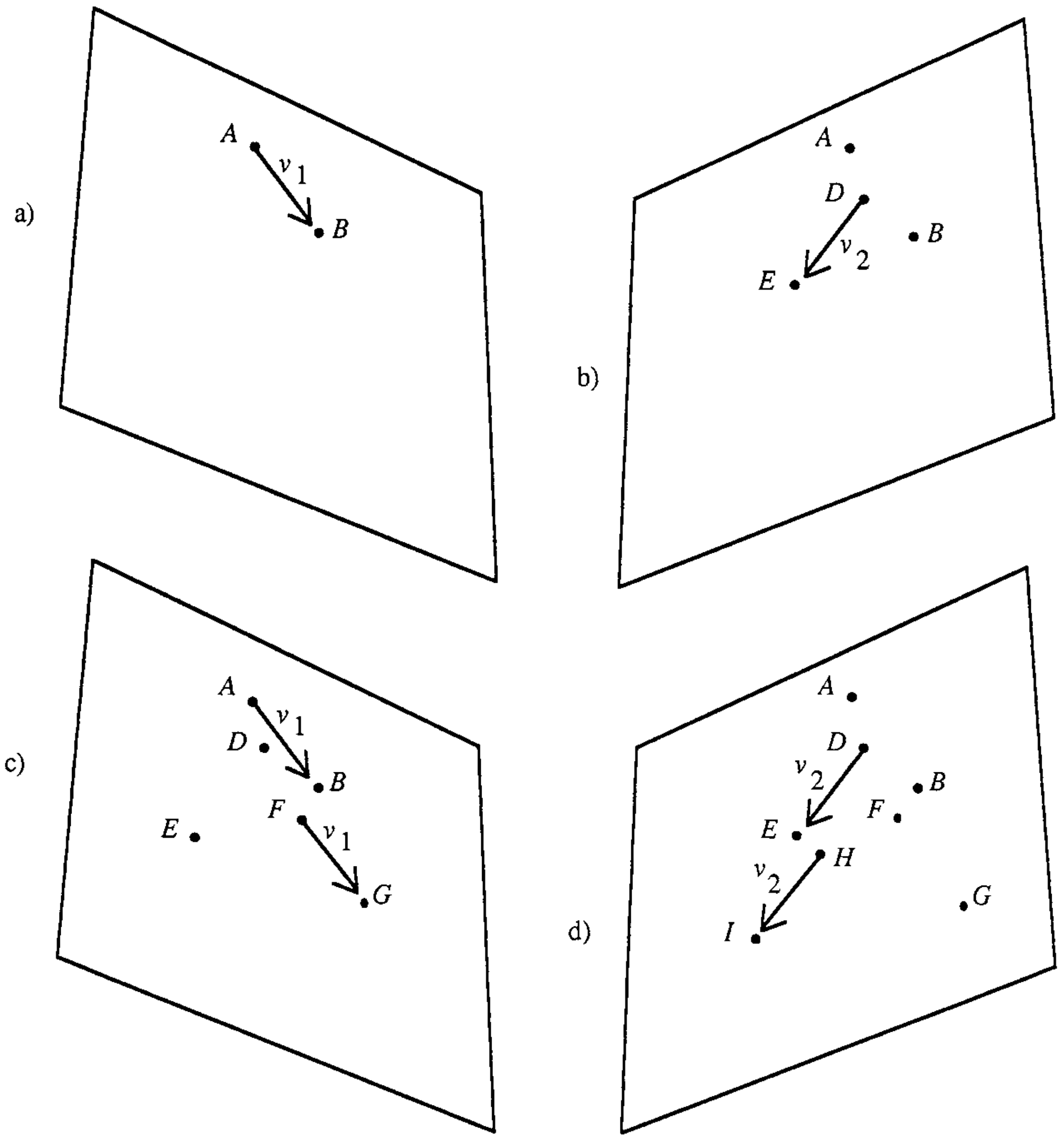

Figure 6-2. An illustration of the path taken across the error space for a network employing meta-connections. See text for explanation.

When pattern 2 is presented again, the network will no longer be at point E, because of the reason given above. It should however, be in the vicinity of point E, at a point like H (see Figure 6-2d). From point H, the network will again move in the direction of vector v_2.

Although there are minor perturbations between each presentation of a given input/output pattern, its clear that the overall pattern of movement in the network over a number of presentations of the given input/output pattern pair is in the general direction of the steepest gradient for that input/output pair.[1] In this manner, the MGDR performs gradient descent on the error surface for each individual input/output pattern of a problem. A network employing the MGDR need only continue this descent until it has found a system of normal and meta-connection weights which maps each input pattern to the appropriate output pattern.

One possible argument against the significance of meta-connections is that a network with meta-connections can be transformed into a network without meta-connections, but which is computationally as powerful as the original by replacing each meta-connection with a "hidden" unit as depicted in Figure 6-3. While the network without meta-connections can perform any computation possible in the network with meta-connections, the asymmetry inherent in the network with meta-connections makes learning much more feasible.

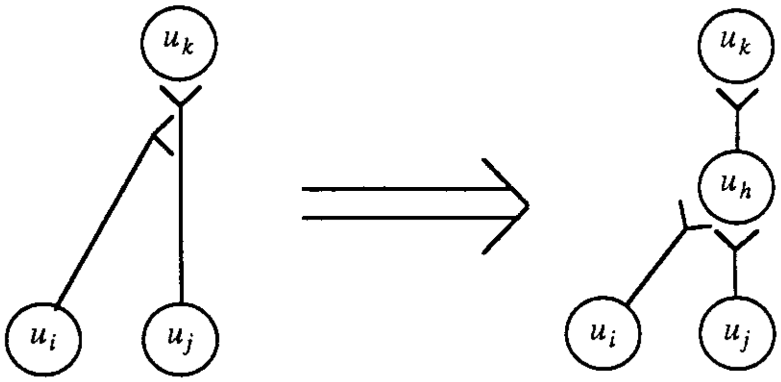

Figure 6-3. An example of the strategy for converting a network with meta-connections to a network without them by adding extra hidden units.

[1]The planar error surfaces in Figure 6-2 are somewhat misleading in that they suggest that the MGDR consistently moves in a direction which is different from the direction of the steepest gradient. In fact, if the error surface has a bowl shape, then the perturbations experienced between presentations of an I/O pattern will be corrected for because the network will alter the direction of its next step to head towards the bottom of the bowl.

Specifically, in the network with meta-connections, the weight of the meta-connection $c_{(kj)i}$ will only be altered if both u_i and u_j are active. This however is not true in the normal network. The weight of the connection c_{hi} will be altered whenever unit u_i is active, regardless of unit u_j's activation level. This decreased specificity results in a large decrease in the learning speed of the network without meta-connections for two reasons. First, the learning rate decreases because the rate of movement in the weight of c_{hi} towards its proper level (dictated by the desired activation level of the output u_k on the input pattern in which both u_i and u_j are active) is decreased by the fluctuations in the weight of c_{hi} on the pattern in which unit u_i is active but unit u_j is off.

But perhaps more importantly, the higher specificity of meta-connections means that fewer I/O patterns are dependent on the weight of any given meta-connection, so larger changes can be made in the weight of the given meta-connection on those trials that do depend on it without seriously disrupting the solution developed for other I/O patterns. This increased specificity of meta-connections translates into networks that perform well using a high meta-learning rate constant and hence take large steps towards a solution upon presentation of each I/O pattern. For this reason networks employing the MGDR can learn more quickly than networks without meta-connections.

The learning speed advantage of a network employing meta-connections can in fact be quite significant. For example, the network with meta-connections depicted in Figure 6-4 can learn to solve the XOR problem in fewer than 20 presentations of the four input/output patterns. The corresponding network without meta-connections, depicted in Figure 6-5, requires an average of nearly 150 trials to reach a solution to the XOR problem.[2]

[2]This average is in fact misleadingly low because it does not take into account test runs on which the network became stuck in a local minima.

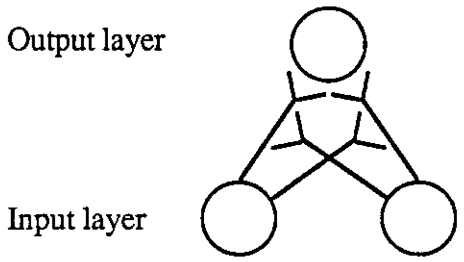

Figure 6-4. A network with meta-connections for computing XOR.

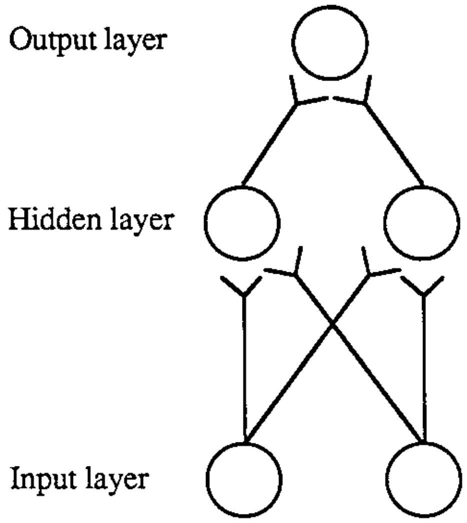

Figure 6-5. A network without meta-connections for computing XOR.

A second modification of the normal network configuration with which meta-connections may be compared is the concept of sigma-pi units developed by Rumelhart, Hinton and McClelland (1986). Networks with sigma-pi units are somewhat similar to networks with meta-connections, in that they use the pattern of activation over a set of units, called a sigma-pi conjunct, to determine the net input to a given unit resulting from a given connection. A sigma-pi connection is graphically represented in the network of Figure 6-6. Mathematically, the net input to unit k from the sigma-pi connection in the figure is

$$net_k = w_{k(ji)}\, o_i \;\, o_j$$

where $w_{k(ji)}$ is the weight of the sigma-pi connection from the conjunct of u_i and u_j to u_k. More generally, the net input to a unit from a sigma-pi connection is the weight of the connection times the product of the activation levels of the units in the sigma-pi conjunct.

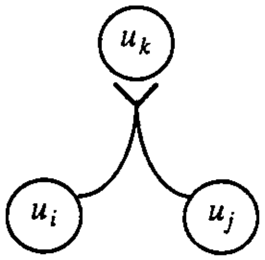

Figure 6-6. An example of a sigma-pi connections.

In certain respects, a network with a normal connection between u_j and u_k with a single meta-connection projecting to it from u_i can be considered similar to a network with a connection between u_j and u_k and a sigma-pi connection from u_i and u_j to u_k (see Figure 6-7).

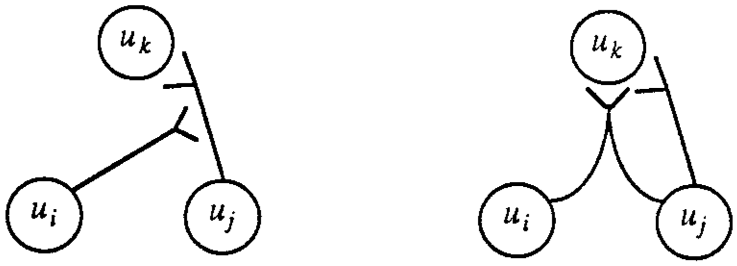

Figure 6-7. An example of analogous networks, one with a meta-connection and the other with a sigma-pi connection.

Mathematically however, the behaviors of these two networks differ substantially. One difference involves the forward propagation of information. In the MGDR network, the net input to u_k is

$$\begin{aligned} net_{pk} &= \left(w_{kj} + w_{(kj)i}\sqrt{o_i} \right) o_j \\ &= w_{kj}\,o_j + w_{(kj)i}o_j\sqrt{o_i} \end{aligned}$$

while in the GDR network, the net input to u_k is

$$net_{pk} = w_{kj}\,o_j + w_{(kj)i}o_jo_i .$$

A second difference involves the formulas for changing a network's weights. In the MGDR network, it has been shown that the total change in the weight of the connection from u_j to u_k will be equal to the change dictated by the GDR. Therefore, when the network with the meta-connection is presented with the pattern again, the change in net input to u_k will be equal in the MGDR and the GDR. Using the generalization of the GDR for sigma-pi units developed by Rumelhart, Hinton & Williams (1986), the net change in w_{kj} would equal that dictated by the GDR and the change to the weight of the sigma-pi connection will be some additional amount. Without even considering the specific formula for weight changes in the sigma-pi GDR, it is clear that the change in the net input to u_k when the given pattern is presented again will be greater than that dictated by the GDR for the network with the sigma-pi connection. Therefore, the change in the net input to u_k in the sigma-pi network will be greater than the change in the corresponding network with meta-connections.

This is not meant to be a thorough comparison of networks employing sigma-pi connections and networks employing meta-connections. In particular, it only points out some of the differences between the two architectures and algorithms, without addressing which is more effective. We are now conducting a more complete theoretical and empirical comparison between networks with meta-connections, sigma-pi networks, the Cascaded Networks of Pollack (1987) and the higher order networks of Maxwell, Giles & Lee (1987).

There is a more general version of the MGDR which is interesting to consider. The generalized equations are

$$W_{p(ji)} = w_{ji} + \sum_k w_{(ji)k}\, g_1(o_{pk})$$

and

$$\Delta_p w_{(ji)k} = \eta\, \delta_{p(ji)} \frac{g_2(o_{pk})}{net_{p(ji)}}$$

where $g_1(x) \cdot g_2(x) = x$ for all $x \in [0, 1]$.

With these equations,

$$\Delta_p W_{p(ji)} = \Delta_p w_{ji} + \sum_k \Delta_p w_{(ji)k}\, g_1(o_{pk})$$
$$= \Delta_p w_{ji} + \sum_k \mathfrak{M}\, \delta_{p(ji)} \frac{g_2(o_{pk})}{net_{p(ji)}} g_1(o_{pk}).$$

In this paper we have chosen

$$g_1(x) = g_2(x) = \sqrt{x}$$

but there are other possibilities. In the sigma-pi GDR, $g_1(x) = x$ and $g_2(x) = x\ net_{p(ji)}$. Since $g_1(x) \cdot g_2(x)$ does not equal x, the sigma-pi GDR does not keep the total weight change equal to the change dictated by the GDR. Another example is to let $g_1(x) = x$ and $g_2(x) = 1$. In this case, the total weight change will equal the change dictated by the GDR, but the change in the weight of a meta-connection will no longer depend on the activation level of the unit from which it projects. Similiarly, if $g_1(x) = 1$ and $g_2(x) = x$, the total weight change will be correct, but the influence of a meta-connection on the total weight of a connection will be independent of the activation level of the unit from which the meta-connection projects. When $g_1(x) = x^{1/3}$ and $g_2(x) = x^{2/3}$, both the desired total weight change and the desired dependencies are maintained. Therefore this version of the MGDR might be an interesting one to test.

In summary, networks with meta-connections can learn quickly because they allow the weights of connections to be tailored for individual input/output patterns. This rapid learning cannot easily be achieved by replacing meta-connections with hidden units. In addition, there does not appear to be a simple equivalence relationship between networks with meta-connections and networks with sigma-pi connections. Further comparisons are necessary with sigma-pi networks and other complex architectures to determine the proper perspective for viewing the MGDR.

Chapter 7
Conclusion

In review, this thesis is an exploration of a new architecture and learning algorithm for connectionist networks involving meta-connections and the meta-generalized delta rule. Our major finding is that by using meta-connections to sample the state of the network, individual connection weights can be tailored for individual input/output patterns. Networks with meta-connections not only learn more quickly and reliably than networks without, but also reach solutions which appear to generalize more easily to new inputs.

However, the advantages of meta-connections do not come without costs. First, meta-connections greatly increase the complexity of networks. In fact, the results in Chapter 6 suggest that there is a tradeoff between network complexity, in the form of increased meta-connectivity, and the learning efficiency of a network. However, the tradeoff does not seem to be linear. At least for the relatively simple problems tested here, a small number of meta-connections usually results in a large increase in learning speed. Perhaps one of the most important findings of this thesis is that there is a positive relationship between network complexity and learning efficiency. Previously, it was very difficult to speed up learning in connectionist networks, regardless of the complexity one was willing to introduce.[1]

The added complexity introduced by meta-connections is troublesome in another respect. The actual solutions derived by the MGDR have been described for only the very simplest networks. In the other cases, only the theory underlying the means by which the

[1]Rumelhart, Hinton & Williams (1986) found that exponential increases in the number of hidden units were necessary to produce a linear increase in learning speed on the XOR problem. Plaut, Nowlan & Hinton (1986) found that increasing either the number of hidden layers or the number of units in a hidden layer actually decreased the rate of learning in the random vector pairing problem.

MGDR reached a solution has been presented. The reason for this is simple: the MGDR almost never develops solutions amenable to easy explanation. This solution complexity could be viewed as a problem for two reasons. First, one can argue that if we knew exactly how a network processed information, we might be able to recognize its shortcomings and improve it. Second, it seems somewhat unsatisfying to develop a network which solves a problem without understanding how the solution works. It is part of human nature to desire knowledge and understanding.

But perhaps this is the direction in which work in artificial intelligence must head to solve the knowledge bottleneck problem. The magnitude of the information required for functioning in our complex world precludes a detailed understanding of its every aspect in man or in any machine we construct to effectively function in man's environment. As was mentioned in the introduction, the only way for a true artificially intelligent system to acquire and process this mass of information may be to learn it from experience.

It remains to be seen whether the approach of connectionism in general, and the meta-generalized delta rule specifically, will help the discipline in this direction. There remain important conceptual questions concerning the meta-generalized delta rule left unanswered by this thesis. Specifically, the behavior of networks employing the MGDR on the error surface in the weight space of all connections, including meta-connections, is still largely unknown. It would be interesting to know if the MGDR performs gradient descent on this higher-dimensional error surface. If it doesn't, then an obvious extension of the MGDR would be a learning algorithm with meta-connections which does.

In addition, major hurdles in general connectionist research need to be overcome before the hopes of connectionism can become a reality. Among the hurdles are large scale tests of systems like the meta-generalized delta rule to determine if the encouraging preliminary results will generalize to more difficult problems.

It is my hope that the results of this thesis will have a twofold effect on connectionist research. First, the positive results obtained using the meta-generalized delta rule should encourage further research into networks with meta-connections. Second, the increased learning rate and reliability of the meta-generalized delta rule should facilitate exploration of more difficult tests of connectionist theory.

References

Hinton, G.E. & Anderson, J.A. (1981). *Parallel Models of Associative Memory.* Hillsdale, NJ: Erlbaum.

House, D. (1987). Williams College Dept. of Computer Science, Williamstown, MA. Personal communication.

Maxwell, T., Giles, C.L., & Lee Y.C. (1987). Generalization in neural networks: The contiguity problem. In *Proc. IEEE Int. Conf. on Neural Networks.* San Diego, CA.

Minsky, M., & Papert, S. (1969). *Perceptrons.* Cambridge, MA: MIT Press.

Pollack, J.B. (1987) On connectionist models of natural language processing. PhD Thesis. New Mexico State U. Dept. of Computer and Cognitive Science, MCCS-87-100.

Plaut, D.C., Nowlan S.J., Hinton, G.E. (1986). Experiments on learning by back propagation. Carnegie-Mellon technical report # CMU-CS-86-126.

Rumelhart, D.E., Hinton, G.E., & McClelland, J.L. (1986). A general framework for parallel distributed processing. In D.E. Rumelhart & J.L. McClelland (Eds.) *Parallel Distributed Processing: Explorations in the microstructures of cognition. Volume 1: Foundations.* Cambridge, MA: Bradford Books/MIT Press.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.) *Parallel Distributed Processing: Explorations in the microstructures of cognition. Volume 1: Foundations.* Cambridge, MA: Bradford Books/MIT Press.

Thompson, R.F., (1986). A Conversation with Richard F. Thompson. In A.B. Crider, G.R. Goethals, R.D. Kavanaugh, & P.R. Solomon, *Psychology.* Glenview Illinois: Scott, Foresman & Co.