Chapter 13: Value Units And Linear Nonseparability

13.1 LINEAR SEPARABILITY AND ITS IMPLICATIONS

The main implication from Exercise 12.1 is that that perceptrons that use the step activation function are unable to learn to solve linearly nonseparable problems. This will also be the case for perceptrons that use the logistic function, because the logistic is best thought of as a continuous approximation of the step function. While we have demonstrated this limitation empirically in Chapter 12, proofs of this limitation also exist (Minsky & Papert, 1988). These proofs are elegant formalizations of some of the intuitions that you explored when answering questions 3 and 4 of Exercise 12.1. Many researchers have claimed that the original publication of such proofs in 1969 led to the near extinction of research on artificial neural networks (Medler, 1998). Minsky and Papert (p. xii) have a slightly different view of this history: "One popular version is that the publication of our book so discouraged research on learning in network machines that a promising line of research was interrupted. Our version is that progress had already come to a virtual halt because of the lack of adequate basic theories." It is certainly the case that perceptrons are unable to represent solutions to some problems that seem very elementary to us, and as a result this restricts our interest in perceptrons as models in psychology and cognitive science. However, the specific problems that perceptrons have depend crucially on their activation function. We'll begin to explore this issue in this chapter by considering how value units cope with the logic problems that were posing problems to the network in Chapter 12.

13.1.1 WHY TYPICAL PERCEPTRONS CAN'T SOLVE NONSEPARABLE PROBLEMS

Why can't perceptrons that use the step activation problem solve linearly nonseparable problems? One answer to this question comes from considering a perceptron to be a tool for a geometric projection. In the case of the two-valued algebra, the connection weights of a perceptron are used to project a two-dimensional representation of the pattern space onto a one-dimensional representation. That is, the connection weights convert the (x, y) coordinates of a pattern in a two-dimensional plane into a single number (the net input) that is the coordinate of the point on a one-dimensional number line. The threshold of the perceptron represents a divider on this number line. If a net input falls on one side of this divider, then the pattern that generates this net input is false. If the net input falls on the other side of this divider, then the pattern is true. However, when a perceptron projects a linearly nonseparable logic problem onto this one-dimensional line, the single divider cannot separate all of the false patterns from all of the true patterns. In order to do this, the perceptron would need two dividers. However, it only has one, because its output unit only has one threshold – in the typical architecture (Rosenblatt, 1962).

13.1.2 WHAT VALUE UNITS HAVE TO OFFER

When the perceptron architecture was introduced in Chapter 9, one of the points that was stressed was that different activation functions could be used in the output units. One of these activation functions was the Gaussian that was used to create networks of value units (Dawson & Schopflocher, 1992b). Value units are of particular interest in the context of learning the linearly nonseparable operations of the two-valued algebra because they can be described as if they have two thresholds, a lower threshold and an upper threshold. If the net input of a pattern falls between the two thresholds, then the unit's activation will be high. If the net input falls below the lower threshold, or above the upper threshold, then the unit's activation will be low. It would seem that this kind of activation function is ideally suited to deal with the linearly nonseparable problems that posed challenges to the perceptron that was studied in Chapter 12. The purpose of the next exercise is to generate some data that is relevant to this hypothesis.

13.2 VALUE UNITS AND THE EXCLUSIVE-OR RELATION

One of the logical operations that was shown to be problematic for a perceptron that used the step function was exclusive-or (XOR, or $A \otimes B$, or $A \veebar B$). In the following exercise, we will train a perceptron with two input units and only one output unit. The output unit is intended to compute XOR, and differs from the perceptron investigated in Chapter 12 by being a value unit. The issue of interest is whether this change affects the ability of the perceptron to compute this logical relation.

13.2.1 Procedure For XOR

Using the Rosenblatt program, load the file "XOR.net". On the setup page, choose the gradient descent rule (Gaussian output), and keep the remaining settings at their default values:

- End after a maximum number of training epochs
- End when there are all "hits" and no "misses"
- Randomize patterns each epoch
- Train thresholds during learning
- Default starts for weights
- Default starts for thresholds
- Maximum number of epochs = 1000
- Number of epochs between printouts = 100
- Learning rate = 0.1
- Minimum level of squared error to define a "hit" = 0.01

Press the "Start Training" button to begin training. Keep training the network until it generates 4 hits and 0 misses. Then, have the program build an Excel spreadsheet to summarize the results. This spreadsheet will contain all of the information required to answer the following questions. If you are using a version of the Rosenblatt program that does not use Excel, then save the results of training to a file that you can examine later to answer the questions.

13.2.2 Exercise 13.1

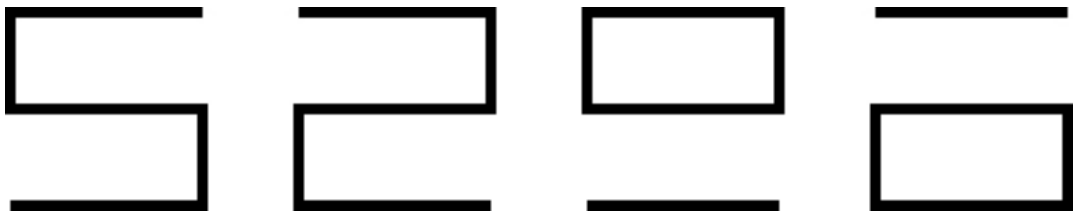
1. **What was the sum of squared error at the end of training?**
2. **How many of training were required before the program stopped training the network?**
3. **Given your answers to questions 1 and 2, what are the implications of this particular network for the claim that perceptrons are unable to represent solutions to linearly nonseparable problems?**
4. **Remembering that this network uses the Gaussian equation described in Chapter 9, and that the bias or threshold of the output unit is equal to the value of μ in the equation, examine the two connection weights that feed into the output unit, as well as the bias of the unit. How does this perceptron compute this particular logical operation?**

13.3 VALUE UNITS AND CONNECTEDNESS

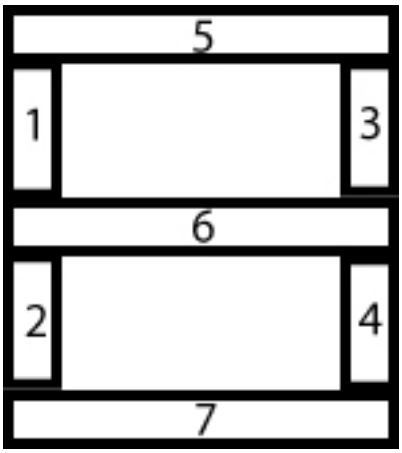
In their detailed analysis of the limitations of perceptrons, one of the geometric properties studied intensively by Minsky and Papert (1988) was connectedness. "We chose to investigate connectedness because of a belief that this predicate is nonlocal in some very deep sense; therefore it should present a serious challenge to any basically local, parallel type of computation" (p. 70).

One of their more straightforward treatments of connectedness involved proving that a limited order perceptron could not learn to correctly classify the four figures below, where the two figures on the left are connected, and the two figures on the right are not connected (Minsky &

Papert, 1988, pp. 12-14). For our purposes, a limited order perceptron is one in which no single connection carries information about all of the elements that make up a stimulus pattern.



In order to create a training set that includes these four figures, we need to find some way in which they can be represented by a perceptron. One approach to doing this is illustrated below. The building blocks for the four figures above are 4 short, vertical line segments and 3 long, horizontal line segments. The figure below illustrates how each of these components can be represented with a single input unit. If a component is part of the figure, then the input unit that corresponds to the component is given a value of 1. If the component is not part of the figure, then its input unit is given a value of 0. With this kind of representation, each of the figures above can be presented to a perceptron by turning 5 of the input units on, and 2 of the input units off. As can be seen from the figure below, for any of the four patterns, the first four input units will indicate the states of the vertical line segments, and the final three input units will indicate the states of the horizontal line segments.



13.3.1 Procedure For Connectedness

Using the Rosenblatt program, load the file “Minsky.net”. On the setup page, choose the gradient descent rule (Gaussian output), and keep the remaining settings at their default values:

- End after a maximum number of training epochs
- End when there are all “hits” and no “misses”
- Randomize patterns each epoch
- Train thresholds during learning
- Default starts for weights
- Default starts for thresholds
- Maximum number of epochs = 1000
- Number of epochs between printouts = 100
- Learning rate = 0.1
- Minimum level of squared error to define a “hit” = 0.01

Press the “Start Training” button to begin training. Keep training the network until it generates all hits and no misses. Then, have the program build an Excel spreadsheet to summarize the results. This spreadsheet will contain all of the information required to answer the following questions. If you are using a version of the Rosenblatt program that does not use Excel, then save the results of training to a file that you can examine later to answer the questions.

13.3.2 EXERCISE 13.2

1. **What was the sum of squared error at the end of training?**
2. **How many of training were required before the program stopped training the network?**
3. **Given your answers to questions 1 and 2, what are the implications of this particular network for the claim that perceptrons are unable to represent solutions to linearly nonseparable problems?**
4. **Remembering that this network uses the Gaussian equation described in Chapter 9, and that the bias or threshold of the output unit is equal to the value of μ in the equation, examine the two connection weights that feed into the output unit, as well as the bias of the unit. How does this perceptron compute this particular predicate?**
5. **Many would argue – legitimately – that this exercise is just a parlour trick. To get a sense of what I mean by this, answer the following question: What is the relationship between how this network solves the connectedness problem and how the previous perceptron solved the XOR problem? To answer this question, you should have already generated answers to question 4 in both this exercise and in the previous one.**