Chapter 12: Performing More Logic With Perceptrons

12.1 TWO-VALUED ALGEBRA AND PATTERN SPACES

Chapter 11 provided a brief introduction to two-valued algebra, culminating in the presentation of a functionally complete set of logic gates (AND, OR, INVERT). A moment's reflection, though, reveals that there must be many more logic gates that can be expressed for a two-valued algebra, and that this functionally complete set is merely a sample of a larger set of primitive logic gates.

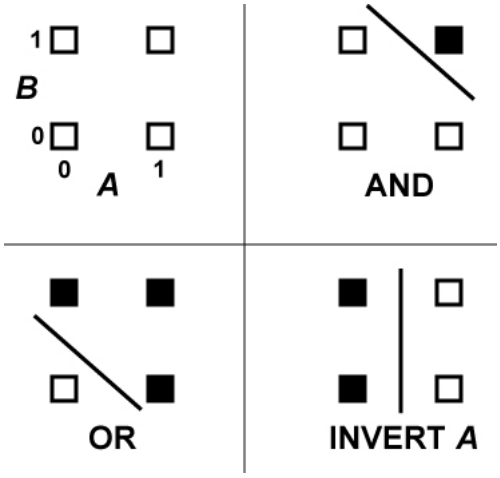
In the two-valued algebra, a logic gate is an operator that combines the values of two input variables, and makes some resulting judgment. The input values of two variables ***A*** and ***B*** can only be one of two values, true or false. This means that, in the two-valued algebra, a logic gate can be only be presented one of four possible stimulus patterns: (***~A***, ***~B***), (***A***, ***~B***), (***~A***, ***B***), or (***A***, ***B***). The response of the logic gate when presented one of these patterns can also only be true or false. By looking at the pattern of responses that a logic gate makes to the set of four possible stimuli, one can determine what relationship between ***A*** and ***B*** is being computed. For instance, if the logic gate responds "true" to the stimulus (***A***, ***B***) and "false" to the other three stimuli, then it must be computing the AND relationship (or, in some of the historic notation seen in Chapter 11 (Boole, 1854), it must be computing ***AB***).

How many different logic gates can be defined for the two-valued algebra? The answer to this question is the number of different patterns of responses that can be generated to the set of four stimuli. Given that the output of a logic gate must be binary (i.e., true or false), and that there are four different stimulus patterns that must be responded to, the number of different patterns of responses that can be generated is 2^4, or 16. Therefore the two-valued algebra must consist of 16 different primitive logical operations. The first of these operations is CONTRADICTION, where the logic gate generates a "false" response to each of the four stimuli. The last of these operations is TAUTOLOGY, where the logic gate generates a "true" response to each of the four stimuli. The other 14 possible logic gates generate "false" to some of the stimuli, and "true" to the others. The complete set of logic gates is provided in the table later in this section.

For the purposes of the exercises in the current chapter, it will be convenient to think of these logical operators from a slightly different perspective. Imagine a logic gate to be a perceptron with two input units (one for ***A*** and one for ***B***) and one output unit (for computing the relationship between ***A*** and ***B***). As was the case in Chapter 11, let an input activation value of 1 represent "true", and a value of 0 represent "false". So, when this perceptron is being trained, it will be presented four different sets of numbers as inputs: (0, 0), (1, 0), (0, 1), and (1,1). In Chapter 4 we saw that in some instances it is useful to represent sets of numbers as points.

The figure below shows a graph in which each of these four patterns involved in the two-valued algebra is represented as a point in a two-dimensional space, where the x-axis represents the value of ***A***, and the y-axis represents the value of ***B***. We will call this the *pattern space* for the two-valued algebra. In processing this pattern space to perform two-valued logic, a perceptron must learn to carve the space into different decision regions (Lippmann, 1989). For example, one straight cut through this pattern space will separate it into two regions. If the point representing a pattern falls in one region of a pattern space so divided, then the point will be classified as being "true" (i.e., the perceptron would turn its output unit on to this pattern). If the point falls in the other region of the pattern space, then the point will be classified as being "false" (i.e., the perceptron would turn its output unit off to this pattern). Clearly there must be 16 different ways in which the pattern space for two-valued algebra can be carved up to separate true points from false points. The figure below illustrates three of these, for AND, OR, and INVERT. In these examples, a straight line is drawn to represent the straight cut that separates the space into decision

regions, false points are represented as outline squares, and true points are represented as solid squares.



Given that there are 16 different primitive operations in the two-valued algebra, it is not surprising that in logical formalisms each operation is represented with its own notation. The table below represents the output responses of each of the 16 operations to each of the 4 possible input patterns; the first column in this table provides a traditional logical notation for these operations (Mendelson, 1970). A second notational approach is to represent each operation with its own graphical symbol (a "jot"), which is occasionally seen in the literature (McCulloch, 1988). For example, McCulloch would represent tautology as ***A*※*B***. I have developed my own system of jots for these 16 operations, where the appearance of each jot is modeled after the appearance of the pattern space as drawn above. My system is provided in the second column of the table below; to see how it works, one need only compare the jot for AND ( ), OR (,) or INVERT ( ) to the pattern space drawings in the figure above. This notational system may be convenient to use when considering answers to the questions below. The appendix to this chapter provides some information about obtaining the font set that I created for this jot notation.

	Inputs	Pattern 1	Pattern 2	Pattern 3	Pattern 4
	A	False	False	True	True
	B	False	True	False	True
Logic Gates (Standard Notation)	**Logic Gates (Jots Notation)**	**Output 1**	**Output 2**	**Output 3**	**Output 4**
Contradiction	***A B***	False	False	False	False
~*A* ∧ ~*B*	***A B***	True	False	False	False
~*A* ∧ *B*	***A B***	False	True	False	False
~*A*	***A B***	True	True	False	False
A* ∧ ~*B	***A B***	False	False	True	False
~*B*	***A B***	True	False	True	False
A* ⊗ *B	***A B***	False	True	True	False
~(*A* ∧ *B*)	***A B***	True	True	True	False
A* ∧ *B	***A B***	False	False	False	True
~(*A* ⊗ *B*)	***A B***	True	False	False	True
B	A B	False	True	False	True
A* ⊃ *B	***A B***	True	True	False	True
A	***A B***	False	False	True	True
B* ⊃ *A	***A B***	True	False	True	True
A* ∨ *B	***A B***	False	True	True	True
Tautology	***A B***	True	True	True	True

12.2 PERCEPTRONS AND LINEAR SEPARABILITY

12.2.1 LINEAR SEPARABILITY

If you examine the 16 jots in the table above, then you should see that they fall into different classes. 12 of the jots have a single line in them that separates outline squares from solid squares. Two of the jots have no line in them, because all of the squares in the jot are of the same type. However, we could easily add a single line to these jots (outside of the four points that the jot illustrates), so one could argue that these two jots are just special cases that belong to the same set as the 12 that we have already discussed. The remaining two jots have two lines in them to separate the two types of squares, and because of this stand out as being different.

Each of the jots used in the table above is based upon a plot of a pattern space, such as those that were illustrated in Section 12.1. The differences between jots that we have just seen reflects an important property of the different ways in which the pattern space for two-valued algebra can be carved into decision regions. 14 of these carved pattern spaces (including the pattern spaces for CONTRADICTION and TAUTOLOGY) are said to be *linearly separable*. In a linearly separable pattern space, all of the points that belong to one class can be separated from all of the points that belong to the other class by carving a single, straight cut through the pattern space. If a pattern space is *linearly nonseparable*, then it is impossible to separate one set of points from the other using a single straight cut. The two jots that stand out as being different from the other 14 are different because they represent two operations in the two-valued algebra that are linearly nonseparable.

The exercise below is an extension of the exercises that were performed in Chapter 11. Its first goal is to explore the ability of perceptrons to learn more logical operations than just the four that were taught in Exercise 11.1. As was the case in this previous exercise, you will train a perceptron with 2 input units to represent the four possible stimuli in the two-valued logic. However, instead of only having 4 output units, you will be working with a perceptron that has 16 output units. Each output unit corresponds to one of the logical operations in the table above; output unit 1 corresponds to the first logical operation in the table, output unit 2 corresponds to the second logical operation in the table, and so on. The second goal of this exercise is to determine whether the difference between linearly separable and linearly nonseparable logical operations.

12.2.2. PROCEDURE

Using the Rosenblatt program, load the file “Boole16.net”. This file represents the table above in a format that can be processed by the perceptron architecture. On the setup page, choose the Delta rule, and keep the remaining settings at their default values, with the exception that the learning rate can be reduced to a value of 0.1:

- 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. *The network will not converge to a solution in which the number of misses drops to 0*. Keep training the network until there are 60 hits and 4 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.

12.2.3 EXERCISE 12.1

1. **Examine the responses of the network to the training set, as well as the errors computed for each output unit and each training pattern. Your network should be making 4 misses. What logical operations are causing the perceptron difficulty? (Note: if an output unit is generating three correct responses for a logical operation, but is generating an error for the fourth response, then we will say that this operation is posing a problem, simply because the output unit has failed to respond correctly to all of the input patterns.)**
2. **Focus on the errors being made by the perceptron in more detail. For each output unit that is making at least one error, describe the output unit's response to each of the four input patterns.**
3. **For each output unit that is making at least one error, examine the threshold for that unit as well as the two connection weights that are feeding into that unit. Use this information, in the context of your description of responses in question 2, to explain why the output unit is not responding correctly. (Remember – if you used the Delta rule, then the output units are using the step activation function.)**
4. **On the basis of your answer to question 3, is it possible in principle for these incorrect output units to eventually learn to respond correctly, or are they doomed to eternal failure? If you think that they cannot learn to respond correctly, then explain this belief. If you think that they can respond correctly, then explain why, and return to the program to try to validate this belief empirically. (Hint: I don't recommend this latter approach!)**

12.3 APPENDIX CONCERNING THE DAWSONJOTS FONT

The DawsonJots font was created with Macromedia Fontographer 4.1 for Windows. Versions of the font files are available for download from the site that provides web support for this book.