WORKSHEET FOR EXERCISES FROM CHAPTER 11

EXERCISE 11.1

1. **What is the total SSE for the network after training has finished?**

The total SSE for the network after training was finished was 0.00

2. **How many epochs of training occurred before the program stopped training the network?**

The program stopped training the network after 10 sweeps, because after this small amount of training the network was generating the correct response to every pattern in the training set.

3. **When the Delta rule is used in the Rosenblatt program, the step activation function is being used in the output units. The unit will only turn on when its net input exceeds the unit's threshold or bias. Armed with this knowledge, look at the two connection weights that feed into the first output unit, and look at the bias of this output unit. Explain how this output unit computes the AND of *A* and *B*.**

The first output unit computes AND. This unit has a bias of -1.5, and weights of 1.47 and 0.5. The net input to this unit is equal to the bias plus the signals coming through the weights from the input units. The net input must be greater than 0 for the output unit to turn on. The only time this will occur for this output unit is when both input units are on, leading to a net input of -1.5 + 1.47 + 0.5 = 0.47. For all other possible states of the two input units, the net input will be less than zero. As a result, this configuration of bias and weights combines to produce an AND unit.

4. **Look at the two connection weights that feed into the second output unit, and look at the bias of this output unit. Explain how this output unit computes the OR of *A* and *B*.**

The second output unit computes OR. It has a bias of -0.50, and weights of 0.98 and 0.92. If both input units are off, then the net input for the unit will be equal to the bias, which is below zero, and the unit will not turn on. However, each weight is large enough so that if either one of the input units, or both of the input units, are turned on, the net input (bias plus signal) will be greater than zero, which is required for the OR relation.

5. **Look at the two connection weights that feed into the third output unit, and look at the bias of this output unit. Explain how this output unit INVERTS the signal from *A*.**

The third output unit computes the inversion of the first input, A. It has a bias of 0.5, a weight of -0.99 from A, and a weight of 0.48 from B. If A is on (regardless of B), the signal will be negative enough to drive net input below zero when combined with the bias, turning the output unit off. When A is off, the bias is large enough (regardless of B) to keep the net input above zero, turning the unit on. Thus the output unit will generate the opposite behaviour of A.

6. **Look at the two connection weights that feed into the fourth output unit, and look at the bias of this output unit. Explain how this output unit INVERTS the signal from *B*.**

The fourth output unit has nearly the same structure as the third, except the connection weights are reversed – the signal from A is weighted by 0.49, the signal from B is weighted by -1.03, and the bias is 0.5. Using a similar logic to the answer in question 5, this configuration of weights will

cause the output unit's net input to be above 0 when B is off, and below 0 when B is on, regardless of what is going on with A. Thus this configuration inverts B, just as the reverse configuration inverted A for the third output unit.

EXERCISE 11.2

1. **What is the total SSE for the network after training has finished?**

The total SSE at the end of training was approximately 0.05.

2. **How many epochs of training occurred before the program stopped training the network?**

The program required 938 epochs to train the network to this level of performance.

3. **How do your answers to questions 1 and 2 above compare to your answers to questions 1 and 2 in Exercise 11.1? If the answers are different, provide a brief explanation of why this is to be expected.**

Basically, the level of error was identical for both networks – they both learned how to solve the problem. However, the gradient descent network took much longer with these settings. This is likely due to the fact that with a continuous activation function, unit error has more possible values, slowing learning down (e.g. a unit can generate a small, nonzero, error). For the delta rule, with a step function in the output units, the unit is either perfect or totally wrong. Without smaller values of error possible, learning can be quite sudden.

4. **When the gradient descent rule is used in the Rosenblatt program, the logistic activation function is being used in the output units. Armed with this knowledge, look at the two connection weights that feed into the first output unit, and look at the bias of this output unit. Explain how this output unit computes the AND of *A* and *B*. How does this explanation compare to the explanation of AND that you provided for the perceptron that was trained with the Delta rule?**

The bias of the output unit for AND was -6.98, and the two connection weights feeding into this unit were both equal to 4.59. Net input equals the total weighted signal coming from the input units, plus the bias. If net input is sufficiently higher than zero then the unit will turn on. With this pattern of connectivity, this will only happen when both input units are on. If only one input is on, or if both are off, net input will be negative – well below zero – and the output unit will be off. This is how the unit computes AND. This explanation is very similar to the AND unit in the previous network, which is to be expected, because the step function and the logistic function are very similar (the latter is a continuous approximation of the former).

5. **Look at the two connection weights that feed into the second output unit, and look at the bias of this output unit. Explain how this output unit computes the OR of *A* and *B*. How does this explanation compare to the explanation of OR that you provided for the perceptron that was trained with the Delta rule?**

The bias for the OR output unit was -2.39, and the weights feeding into it were both 5.3. When both input units are off, the net input will be negative (well below zero) and the unit will be off. When one input unit is on, or both are on, the net input will be driven to a positive value well above zero, and the unit will be on. This results in OR being computed. This account is functionally very similar to the account of the OR circuitry in the previous network that used a step function in the output units.

6. **Look at the two connection weights that feed into the third output unit, and look at the bias of this output unit. Explain how this output unit INVERTS the signal from *A*. How does this explanation compare to the explanation of INVERT that you provided for the perceptron that was trained with the Delta rule?**

The bias of the third output unit is 2.99; the weight from A to this unit is -6.43 and the weight from B to this unit is a near zero 0.23. With this small weight, B's signal has little effect. The large negative weight from A drives the net input below zero when A is on, causing the output unit to turn off. When A is off, the bias of the unit is high enough to have the output unit turn on. As a result, the output unit behaves in an opposite fashion to A. This circuitry is very similar to that found in the previous network for INVERTING an input.

EXERCISE 11.3

1. **What is the total SSE for the network after training has finished?**

At the end of training, the total SSE was 3.

2. **How many epochs of training occurred before the program stopped training the network?**

This was accomplished after 3000 sweeps.

3. **Examine the responses of the network to each pattern, and the errors computed for each output unit for each pattern. In what way is the network behaving correctly? In what way is the network making mistakes?**

The network is correctly inverting the input signals – that is, output units 3 and 4 are generating the correct responses for every training pattern. However, the network is having trouble with AND and OR. It generates two errors for AND: it incorrectly turns on when both input units are off, and it incorrectly turns off when both input units are on. It generates correct responses for the other two input patterns. It generates one error for OR: it turns on when both input units are off. It correctly turns on to the remaining three input patterns.

4. **With the default settings, and with thresholds held constant during training, every output unit always has a threshold of 0. Armed with this knowledge, examine the connection weights that feed into any output unit that is generating errors. Explain why any errors are being made.**

For the AND network, the threshold is zero, and both weights are slightly negative. As a result, the network turns on when both input units are off, because net input is 0 (i.e. exactly equal to the threshold to turn the unit on). When both input units are on, the net input is -.11, so the unit (incorrectly) turns off.

For the OR network, one weight is 0.57 and the other is 0.41. So an input from either unit (or both) will turn the network on. The trouble is that with a 0 threshold, when both inputs are off, net input is 0, and the network will incorrectly turn on!

5. **Given your answer to question 4, speculate on the role of the threshold in the perceptron, and speculate on why it might be important to let the learning rule train thresholds in addition to training the connection weights.**

The threshold appears to need to be adjusted to correctly differentiate on patterns from off patterns – in other words, to shift the "cut" that linearly separates one type of pattern from another. If thresholds are not trained, then the cut cannot be correctly shifted. It would appear

that if you don’t train the threshold, then a perceptron cannot learn to correctly respond to simple logical patterns.