Problem 1.

1. The neural network gets the correct answer most of the time. I trained 100 networks for each of the Boolean operators and tested them against the correct outputs with the following results.

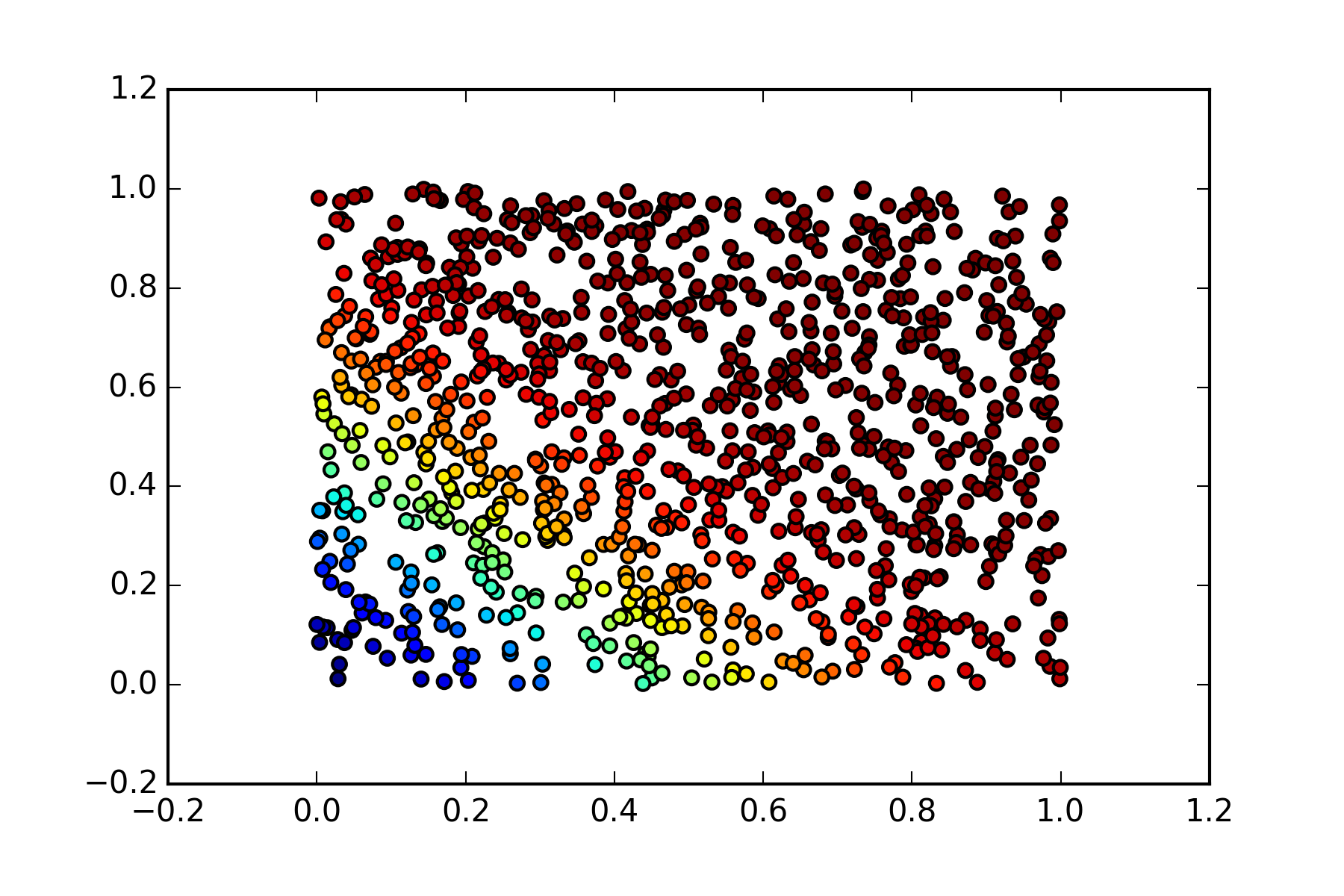
|  |  |  |  |
| --- | --- | --- | --- |
| not | and | or | nor |
| Fully accurate in 100 runs | Iteration 6  Error 0.750044  0.252743  0.252743  0.252743  0.252771  Iteration 88  Error 0.500115  0.489517  0.489311  0.005619  -0.008091 | Iteration 35  Error 0.751248  0.764003  0.764009  0.764010  0.764310 | Fully accurate in 100 runs |

These results are not reproducible because I did not set seed before the simulations, but they do show that the neural networks model the Boolean operations well.

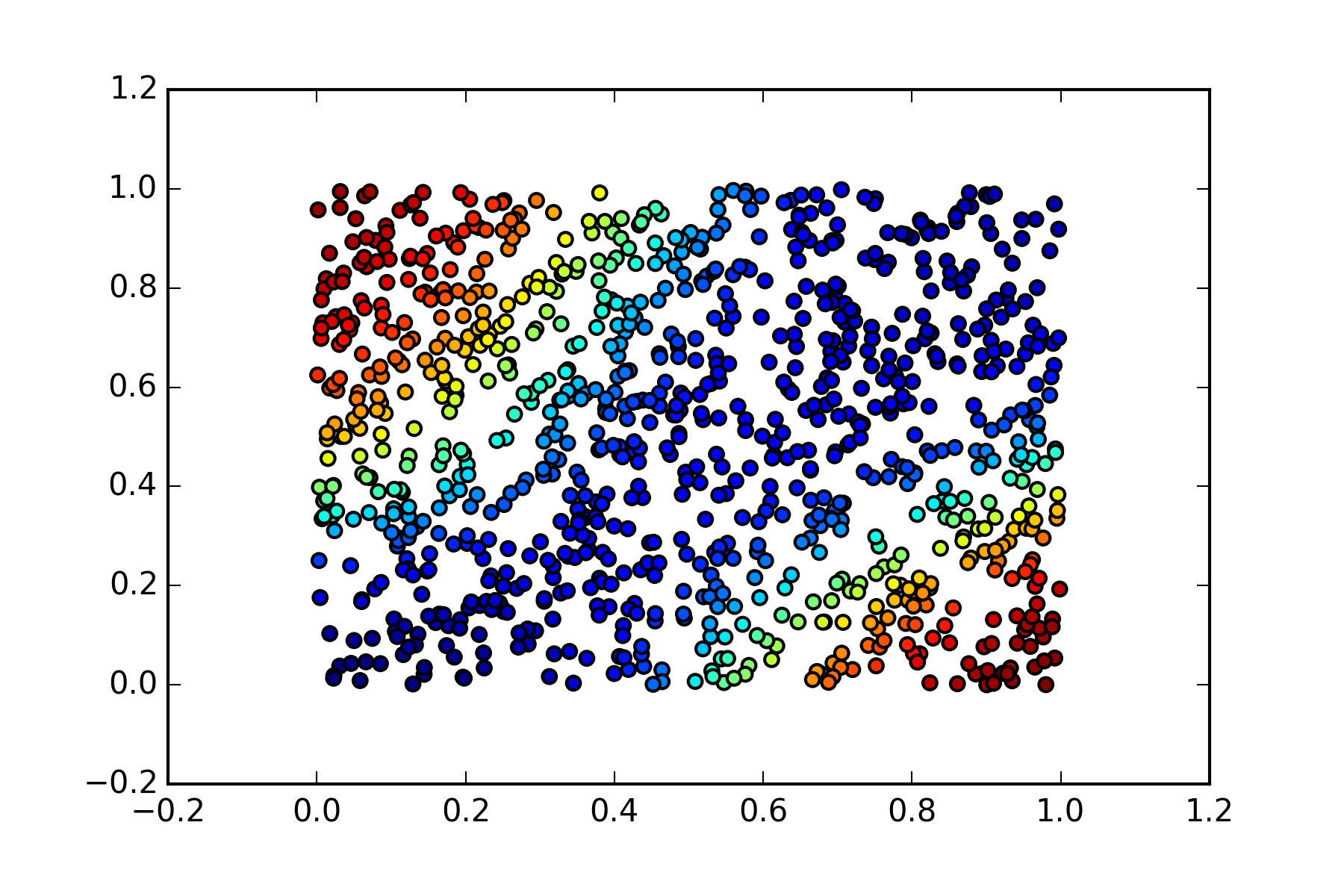
The code we were given uses a neural network with 2 input nodes, one hidden layer with 4 nodes and 1 output node, denoted (2, 4, 1). In addition it has bias nodes. Hidden layers allow the network to create nonlinear decision boundaries. Because OR, NOR, AND, and NOT are linearly separable, they can be modeled with a simpler network without a hidden layer. For example a (2,1) network would suffice. XOR, however is not linearly separable, so a hidden layer is required.

To illustrate the decision making by the neural networks, I plotted the the predictions made by OR and XOR networks trained on the lattice points of the ((0,0), (1,1)) square and then tested on a uniform random sample of points from that square.

Here is a plot of a (2,1) neural network with sigmoid output layer trained for OR operator and tested on a uniform random sample from the square ((0,0), (1,1)). This network is essentially the same as a logistic regression with two predictors.

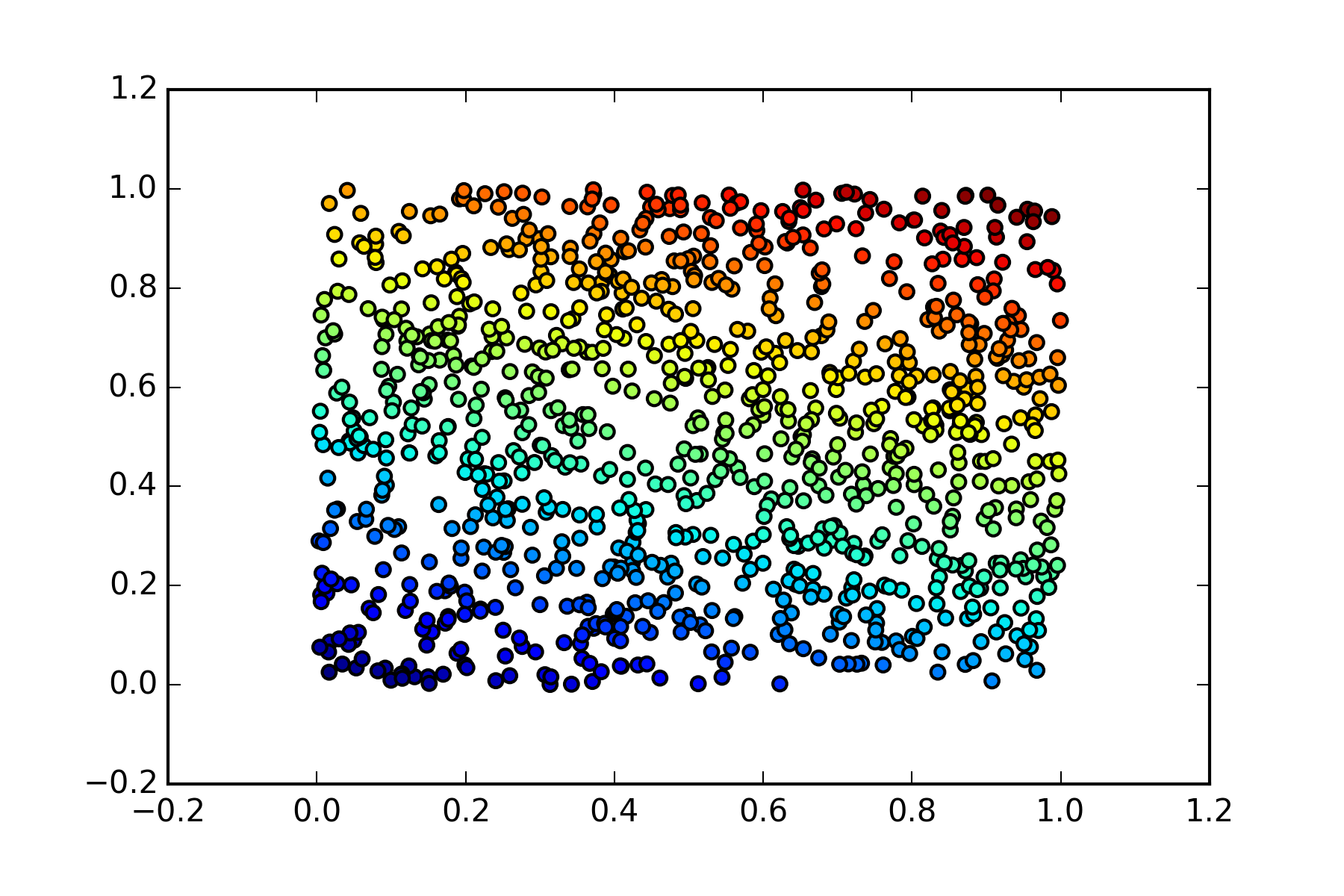


Predictions computed by a (2,4,1) XOR network with a sigmoid output layer can be seen to be not linearly separable in the following image.



1. Changing the numbers of hidden layers for OR, NOR, and AND does not significantly change the prediction quality, but adds computational expense if the number of hidden nodes is increased. The network for XOR becomes ineffective if the hidden node is taken away.

The image below shows the prediction made by a linear XOR network. Its prediction is a complete failure: 0.5020612, 0.50036557, 0.50138001, 0.49968437.



1. Composite Boolean operators
   1. (A AND B) OR C

The output of a (3,4,1) network is given below. It is a successful approximation of the correct result.

(1, 1, 1) [ 1.02140037]

(1, 1, 0) [ 0.98898213]

(1, 0, 1) [ 0.99397006]

(0, 1, 1) [ 0.99427853]

(1, 0, 0) [ 0.00162767]

(0, 1, 0) [ 0.00071983]

(0, 0, 1) [ 1.00415994]

(0, 0, 0) [ 0.00085638]

* 1. (NOT (A OR B)) AND C

The output of a (3,4,1) network is given below. It is successful.

(1, 1, 1) [-0.00046274]

(1, 1, 0) [ 0.00061117]

(1, 0, 1) [ 0.00018912]

(0, 1, 1) [ 0.00039458]

(1, 0, 0) [ 0.0001218]

(0, 1, 0) [-0.00019796]

(0, 0, 1) [ 1.0001051]

(0, 0, 0) [ 7.74848717e-05]

* 1. NOT ((A OR B) AND C)

The output of a (3,4,1) network is given below. It is successful.

(1, 1, 1) [ 0.]

(1, 1, 0) [ 1.]

(1, 0, 1) [ 1.11022302e-15]

(0, 1, 1) [ -1.11022302e-15]

(1, 0, 0) [ 1.]

(0, 1, 0) [ 1.]

(0, 0, 1) [ 1.]

(0, 0, 0) [ 1.]

Although a network with one hidden layer fit all of the composite operators in this assignment, and a number of other three dimensional Boolean operators, I do not believe the (*n*, 4, 1) network, where n is the number (or dimension) of inputs is universal. A paper by Martin Anthony (<http://www.cdam.lse.ac.uk/Reports/Files/cdam-2003-01.pdf>) states that any Boolean function can be computed by a 2-layer neural network with no hidden layers, but that would require a larger input layer than simply the dimension of the input of the Boolean function. There is also a lower limit on the number of nodes in a network of any architecture that would not be satisfied for some Boolean functions by (*n*, 4, 1) networks for a high enough *n*.