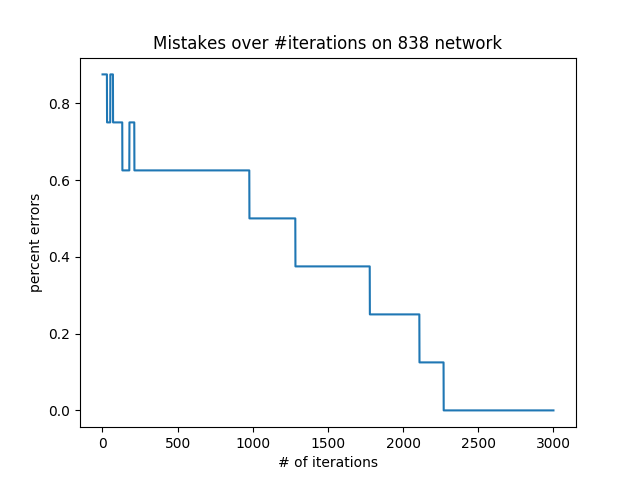
Ethan Hartzell

Machine Learning PP4

 For the 838 dataset, below is the graph of the error rate over the number of iterations on the 838 network, as well at the representation at the hidden nodes at the final iteration. (Input one-hot code, then the hidden nodes’ values). I used the following naming convention for hidden names ‘h’ followed by the depth followed by the position. They use zero-based indexing so the third hidden node in layer 1 will be “h02.”

[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h02 = 0.105409399498

h01 = 0.358657093213

h00 = 0.976533956116

[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h02 = 0.760768348191

h01 = 0.970426035146

h00 = 0.0354614254648

[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]

h02 = 0.928111637458

h01 = 0.0259855907025

h00 = 0.950211128926

[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]

h02 = 0.374218903241

h01 = 0.324616491019

h00 = 0.423112296566

[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]

h02 = 0.367327292136

h01 = 0.407492945142

h00 = 0.339518447745

[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]

h02 = 0.977379717311

h01 = 0.291660459409

h00 = 0.153797482913

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]

h02 = 0.0432857054803

h01 = 0.974279283216

h00 = 0.639391755489

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]

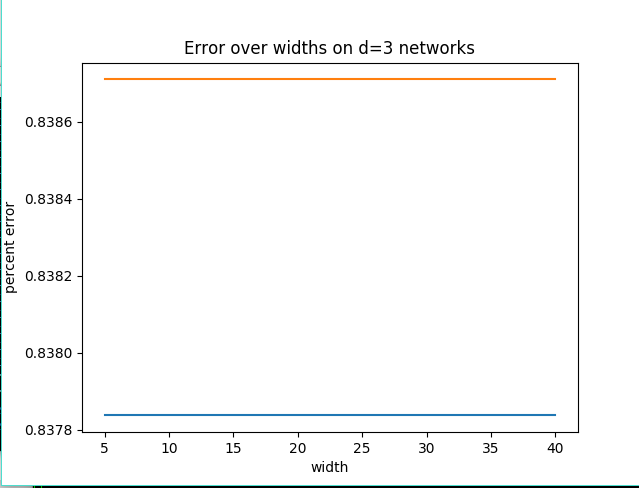
h02 = 0.369794735873

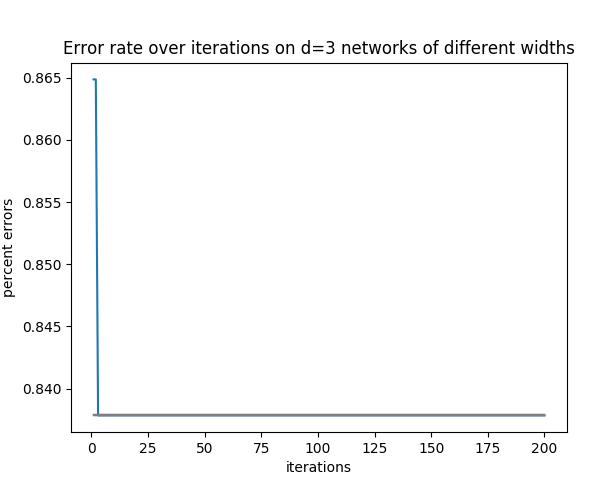
h01 = 0.362755973258

h00 = 0.36503332911

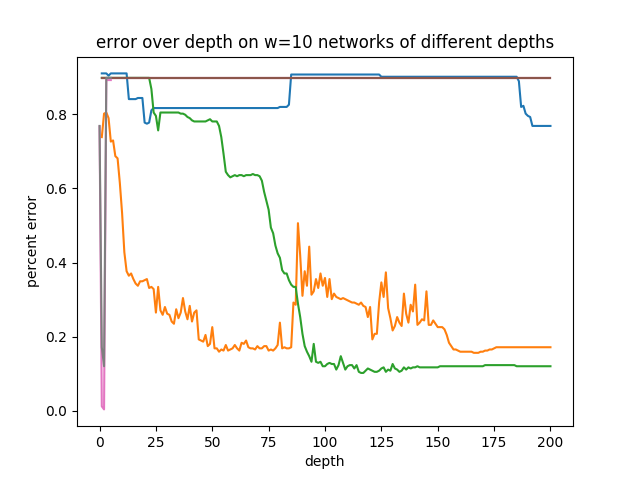
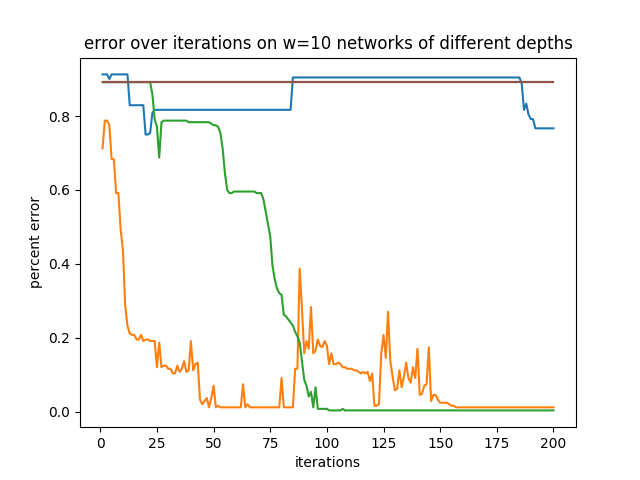
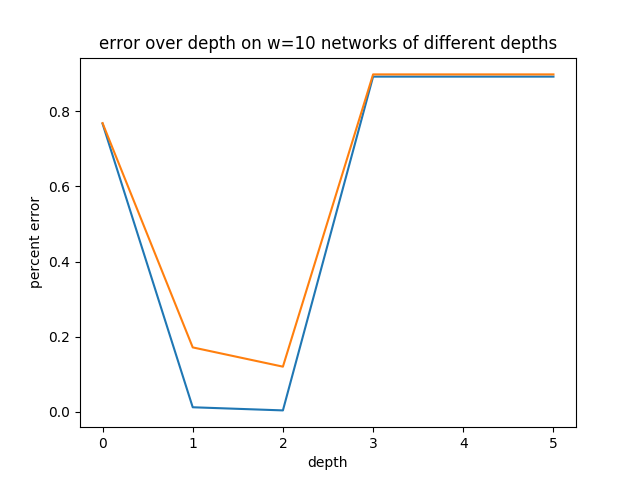
It seems like some sort of representation is discovered that somewhat resembles a binary representation, but it doesn’t match (it doesn’t need to) the actual numbers in 10nary. The curve goes down in steps: this makes sense because our learning rate is a bit slow, and when a representation that finally allows more than one choice to be made, it can be quickly changed back, but with enough iterations we reach a point where all the weights are optimized and something resembling a 3 digit binary representation is learned by the hidden layer. For example, 8 is represented by 3 values close to zero, while 1 is represented by two values close to zero and one value close to 1, and 3 is represented by two values close to 1 and one close to 0. Despite this, some nodes contain values that are all intermediate (such as 5), but what makes it so mistakes are not made is that enough others have gotten close to a binary representation, and that it can be said some of the values produced are closer to 1 than others may be helpful.

Below are graphs for experiment two, showing error over iterations on the test set, and also as a function of width. We might expect the error to decrease as width increases, but depth of 3 does not work well on the data. Note that due to time constraints this experiment was run on a shortened version of the optdigits dataset.

 (blue = training, orange = test)



Below are graphs for experiment three, showing error over iterations on the training and test sets, and also as a function of depth. Some depths are too much to even change the error rate significantly in 200 iterations, whereas some converge on zero at different times. Blue = depth 0. Orange = depth 1. Green = depth 2. Depths 3,4 and 5 seem to have blended into purple. Those networks were so deep that it would’ve taken much longer for any representation to be learned (if possible). Depth of 2 seems to have been the best in terms of getting to zero errors first, although depth=1 also worked. These two both do not perform at 0 mistakes on the test set, but green does significantly better after less iterations. It seems d=2 is the best choice for this data. Note the x-axis on the second graph is mislabeled—this is the error over iterations on the test set (but it says depth). The first graph is the same but on the training set.

 (Graph3- Blue = training, orange = test)