Brennan McFarland
9/21/18
Problem Set 1.

My analytic computation consistently matches the numerical approximation estimates returned by the numer_est_Wji and numer_est_Wkj functions, as evidenced by this sample of iterations' error estimates when running the fitting algorithm:

Iteration 101:

```
dWL_cum =
  -0.0594   -0.0224   -0.0081   -0.0302
est_dWkj =
  -0.0594   -0.0224   -0.0081   -0.0302

dW_Lminus1_cum =
         0.0015         0.0014
         -0.0016        -0.0018
         0.0055         0.0048
         -0.0014        -0.0005
est_dWji =
         0.0015         0.0014
         -0.0016        -0.0018
         0.0055         0.0048
         -0.0014        -0.0005
```

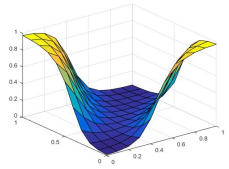
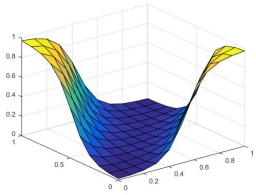
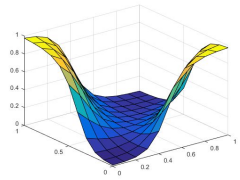
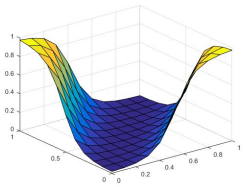
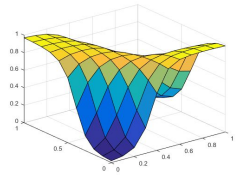
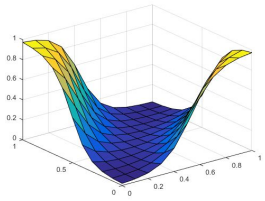
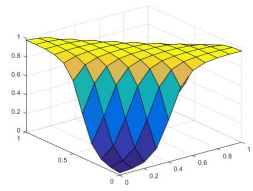
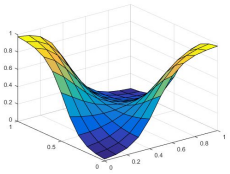
For every dimension of dWL_cum vs est_dWkj and dW_Lminus1_cum vs est_dWji, the difference between the result of each algorithm is less than the digits displayed. This is also true for later iterations:

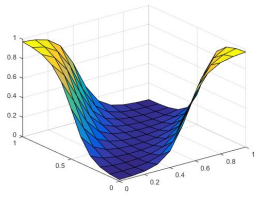
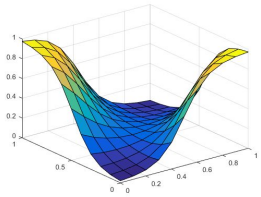
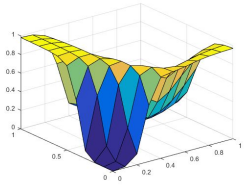
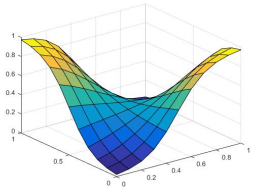
Iteration 6060:

```
dWL_cum =
  -0.0003       0.0011 0.0015   -0.0031
est_dWkj =
  -0.0003       0.0011 0.0015   -0.0031
dW_Lminus1_cum =
  -0.0004       0.0000
  -0.0004       -0.0001
  -0.0002       -0.0030
   0.0008       -0.0006
est_dWji =
  -0.0004       0.0000
  -0.0004       -0.0001
  -0.0002       -0.0030
   0.0008       -0.0006
```

Again, the difference between the two error algorithms is less than the precision of the values displayed.

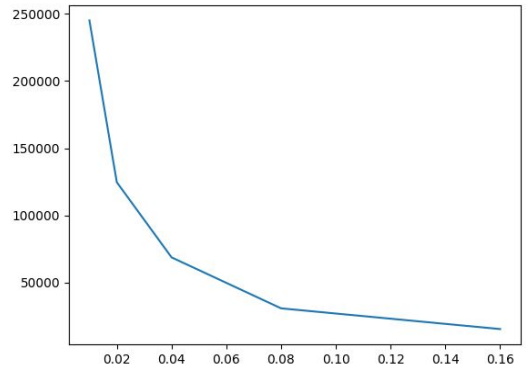
To evaluate the performance of the number of interneurons and choice of epsilon, I had the network train until the rms error was under .025 and kept track of the number of iterations it took to get there. The results for various values of the hyperparameters are as follows:

eps=.01,nnodes_layer1=4: iterations = 245167 	Eps = .16, nnodes_layer1=4: iterations = 15340 
Eps = .02, nnodes_layer1=4: iterations = 124659 	Eps = .16, nnodes_layer1=8: iterations = 13002 
Eps = .04, nnodes_layer1=4: iterations = 68708 	Eps = .16, nnodes_layer1=16: iterations = 10650 
Eps = .08, nnodes_layer1=4: iterations = 30763 	Eps = .16, nnodes_layer1=32 : iterations = 8998 
Eps = .16, nnodes_layer1=4: iterations = 15340	Eps = .16, nnodes_layer1=64 : iterations = 9525

	
Eps = 20, nnodes_layer1=4: iterations = 239 	Eps = .16, nnodes_layer1=256 : iterations = 7552 
At approximately eps=30 (with nnodes_layer1=4), the network appears to run indefinitely as the error never dips below .025, indicating the step size is so large that the error function is oscillating around the minimum instead of going towards it.	Note: though the number of iterations tended to decrease as the number of interneurons increased, the time each iteration took gradually increased, overtaking the time savings from fewer iterations.

As evidenced by the following graph, increasing the epsilon consistently decreased the number of iterations (and time) needed to train to within a given error, with the exception of extremely large epsilon increasing training time ad infinitum which is not shown here as it would blow the rest of the graph out of proportion. This indicates that until the step size is so large it causes oscillation in attempting to decrease the error function, a larger step size will lead to faster convergence.

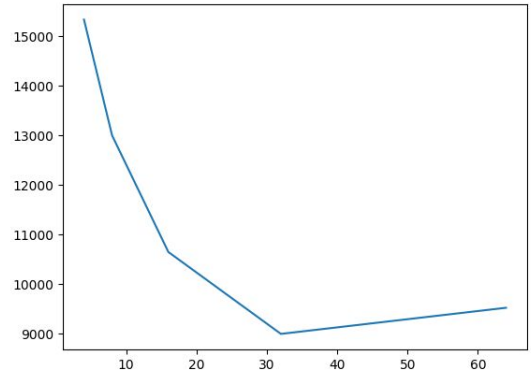
Training iterations



epsilon

Additionally, increasing the number of interneurons also seems to decrease training time up until the number becomes so large that adjusting the weights for each neuron overtakes the gain in performance per neuron.

Training iterations



nnodes_layer1