So I was relatively confident my code was correct and I didn’t want to wait a really long time for the estimations to calculate 20000 times so I turned them off first and let it run to approx. convergence. Here is the plot from that run to 25755 iterations:

I ran it again in the background while writing this document to get a screenshot of the estimated weights for the proof that my calculation matches the approximation:

“rmserr =

0.9411

dWL\_cum =

0.0020 -0.0913 0.0494 0.0349

0.0045 -0.0976 0.0502 0.0352

0.0066 -0.0767 0.0500 0.0351

0.0088 -0.0772 0.0505 0.0352

est\_dWkj =

0.0020 0.0045 0.0066 0.0088

0 0 0 0

0 0 0 0

0 0 0 0

dW\_Lminus1\_cum =

-0.0141 -0.0140

-0.0318 -0.0319

-0.0063 -0.0064

-0.0028 -0.0029

est\_dWji =

-0.0231 -0.0229

-0.0823 -0.0825

0.0512 0.0514

0.0699 0.0716

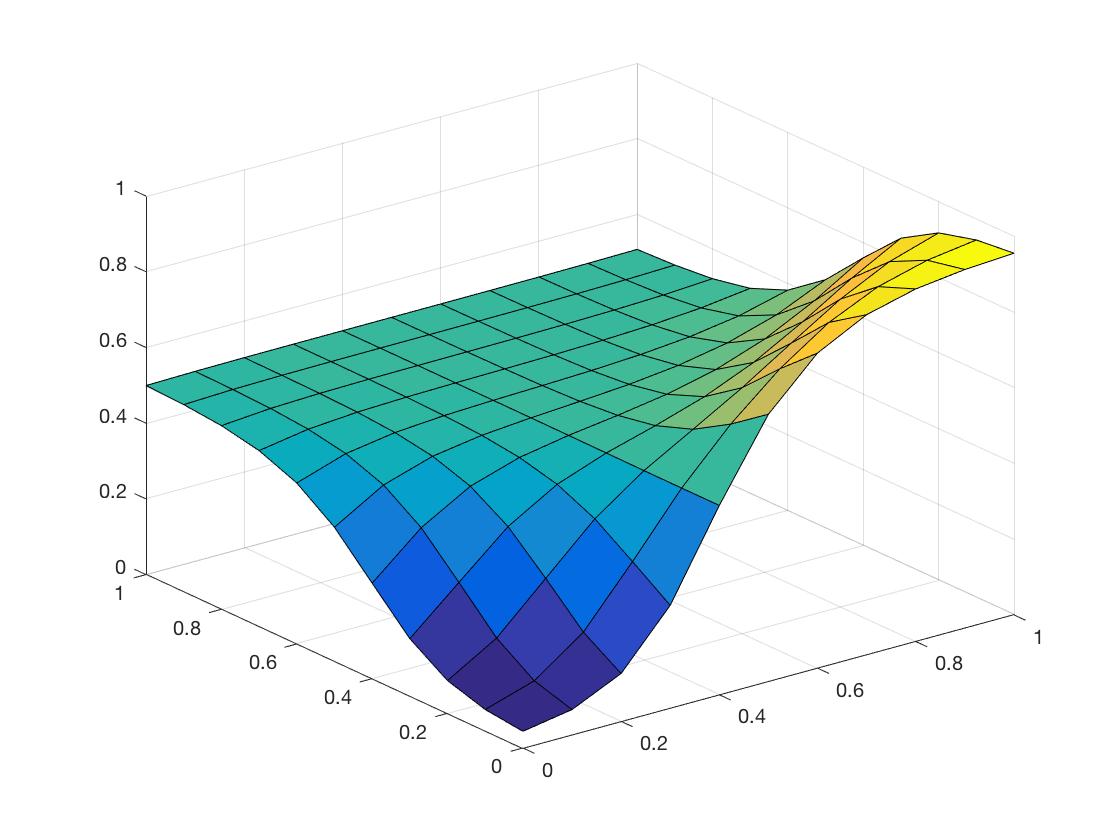
iteration =

14241”

This was only run to 10000 iterations so the estimations don’t actually match my computed result precisely. This is likely why the guess at the corners hasn’t converged on 1 or 0, but more like .7 or .3.

Because convergence was extremely slow and I was impatient I tried turning the learning rate up to .1. This had some interesting results (which I unfortunately forgot to save a plot for). The first time it overfit and produced a flat plane at z = 1 with a sharp drop down to (0,0). Basically what I’m trying to say was it was correct, but not smooth at all. The second time I ran it, with the high learning rate, gradient descent must have over shot the minimum and it converged on an incorrect result.

When learning rate was set to .05 we get an interesting result. After ~15000 iterations, the plot has converged to an incorrect result:



The calculated result compared to the estimated result can be seen here:

dWL\_cum =

-0.0011 -0.0017 0.0000 0.0004

0.0006 -0.0011 -0.0001 0.0014

0.0017 0.0013 0.0006 0.0014

-0.0008 -0.0013 0.0000 0.0003

est\_dWkj =

-0.0011 0.0006 0.0017 -0.0008

0 0 0 0

0 0 0 0

0 0 0 0

dW\_Lminus1\_cum =

0.0022 -0.0011

-0.0022 -0.0022

-0.0007 -0.0005

0.0019 -0.0008

est\_dWji =

0.0036 0.0033

-0.0059 -0.0000

-0.0009 0.0003

-0.0029 0.0001

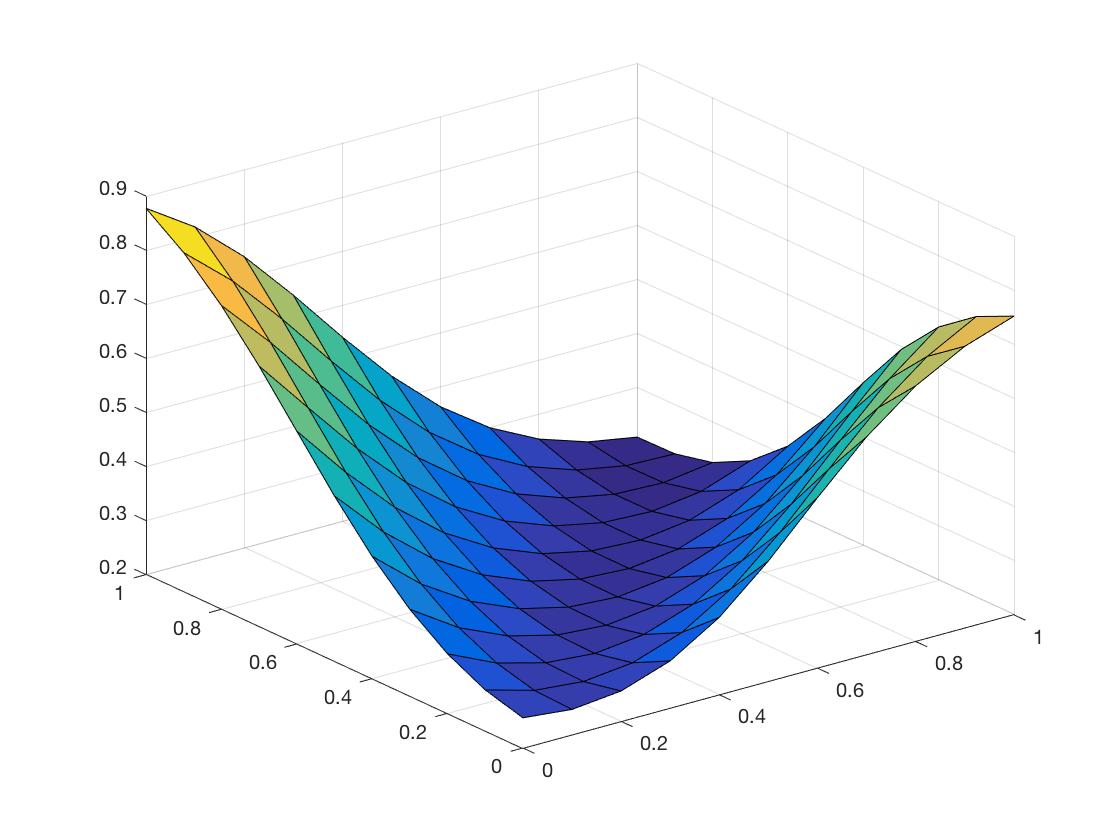
iteration =

15453

Interestingly the computed weights are quite close to the estimated weights, but the plot doesn’t reflect this.

With 16 inner layer neurons instead of 4…

**SO** I got this far into the writeup before I realized there was something wrong. Target was supposed to be 0 with input (1,1), but I was converging to 1. I found my mistake in calculating the sensitivity of the hidden layer neurons. After fixing that I get the following results (~25k iterations):



Which is much better.

My results:

rmserr =

0.7481

dWL\_cum =

-0.0355 -0.0860 -0.0436 -0.0965

0.0041 -0.0019 -0.0008 0.0023

0.0026 -0.0171 -0.0025 -0.0086

0.0004 -0.0567 -0.0105 -0.0355

est\_dWkj =

-0.0355 0.0041 0.0026 0.0004

0 0 0 0

0 0 0 0

0 0 0 0

dW\_Lminus1\_cum =

0.0184 -0.0202

0.0061 -0.0032

0.0099 -0.0022

-0.0350 0.0358

est\_dWji =

0.0184 -0.0202

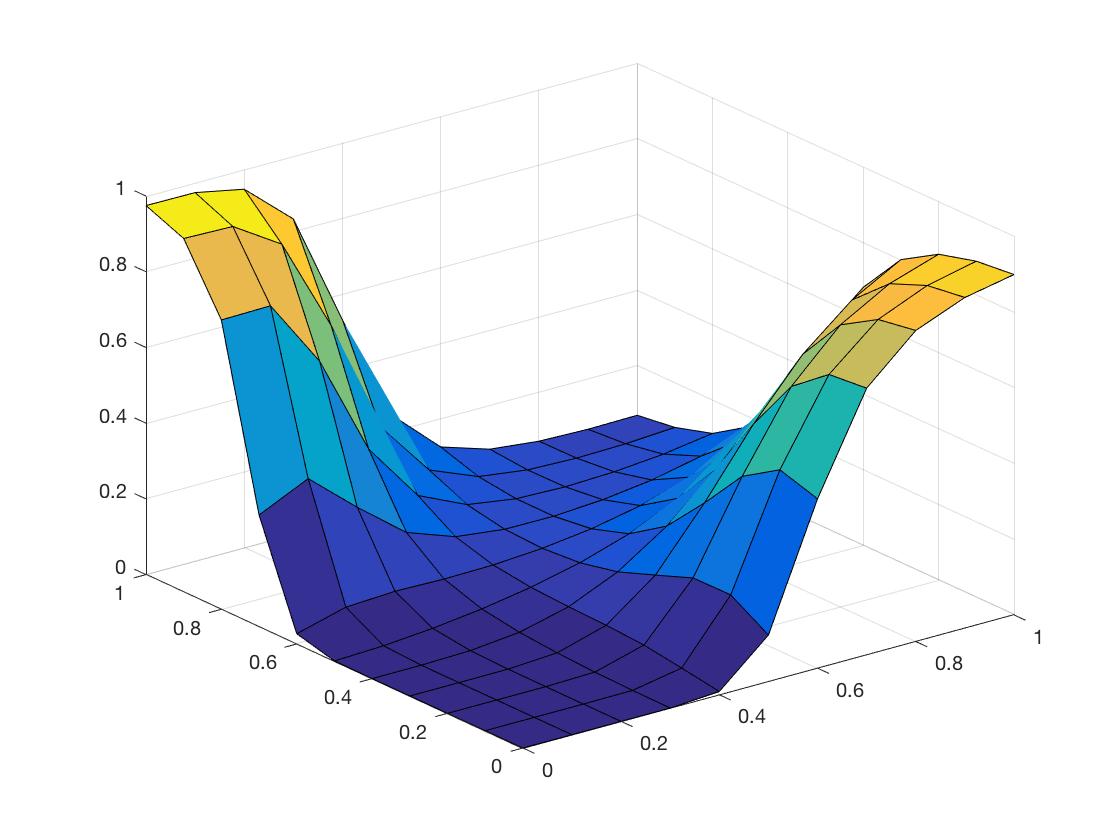
0.0061 -0.0032

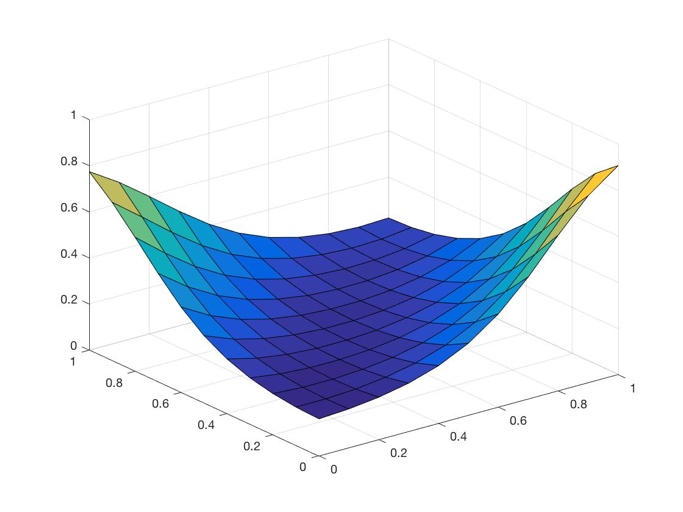
0.0099 -0.0022

-0.0350 0.0358

Are a lot more consistent now too.

Increasing the learning rate to .1, we get an ugly (unsmooth) result in about 5000 iterations:



Increasing the number of hidden neurons to 16 it takes forever to converge (as expected since we just roughly squared the number of calculations. At ~30k iterations we have: