Anthony Bugatto

CS 455: Mobile Sensor Networks

May 8, 2018

Project 2: Consensus Filters

For this project we had to finish two tasks: design a consensus filter that finds the value of a single data cell, and design a consensus filter that finds the values of a 25x25 grid of data cells with an embedded Gaussian mixture.

For the first part we show the convergence of the weighted consensus algorithm 1. This response is in figure 1 and as we can see it converges very nicely after roughly 300 iterations. We can also note that it is a mere .025 off the target value, which makes for a very accurate

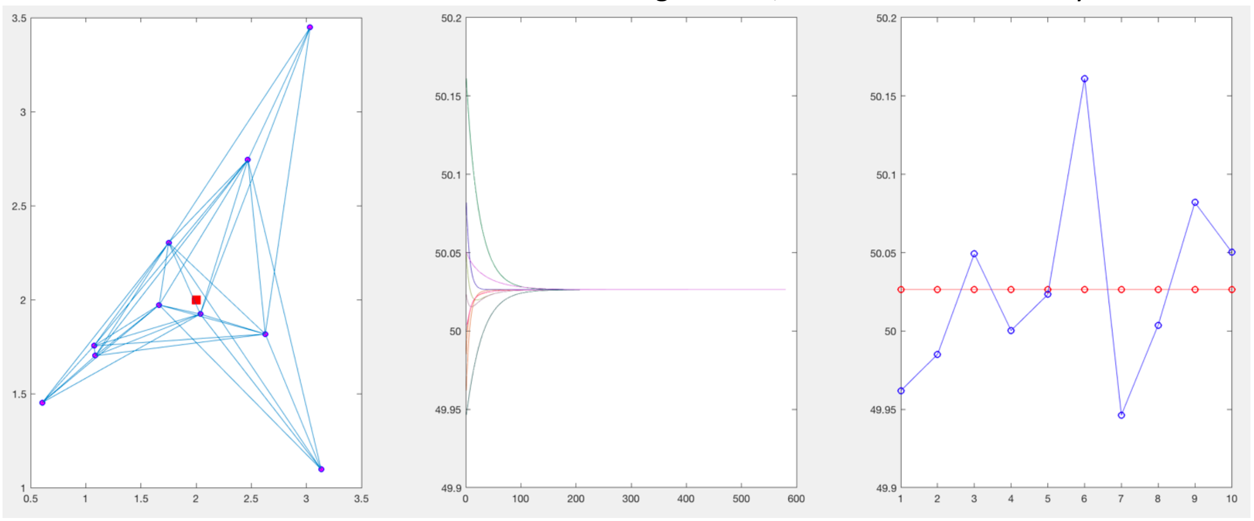

Figure 1: a) Graph b) Graph of each node's convergence c) Graph of initial and final values

approximation. The second part was to fins the convergence of the weighted consensus algorithm 2. Shown in figure 2, this algorithm also converges nicely but at a much faster rate of 30ish iterations and the error from the target value is negligible. So overall, this algorithm is much better than than former. The third part of the project was to use the metropolis

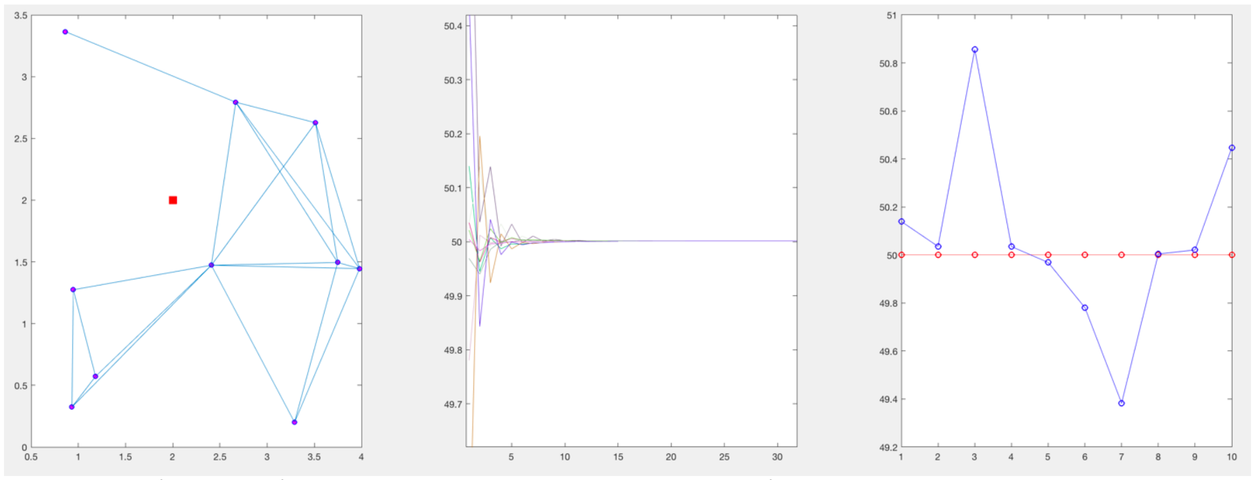

Figure 2: a) Graph b) Graph of each node's convergence c) Graph of initial and final values

weighting algorithm to find the average consensus and convergence of the algorithm. Seen in figure 3, the metropolis algorithm is a little slower to converge than the weighted 2 algorithm though like it, the consensus value is negligibly close to the target value. The fourth part is to

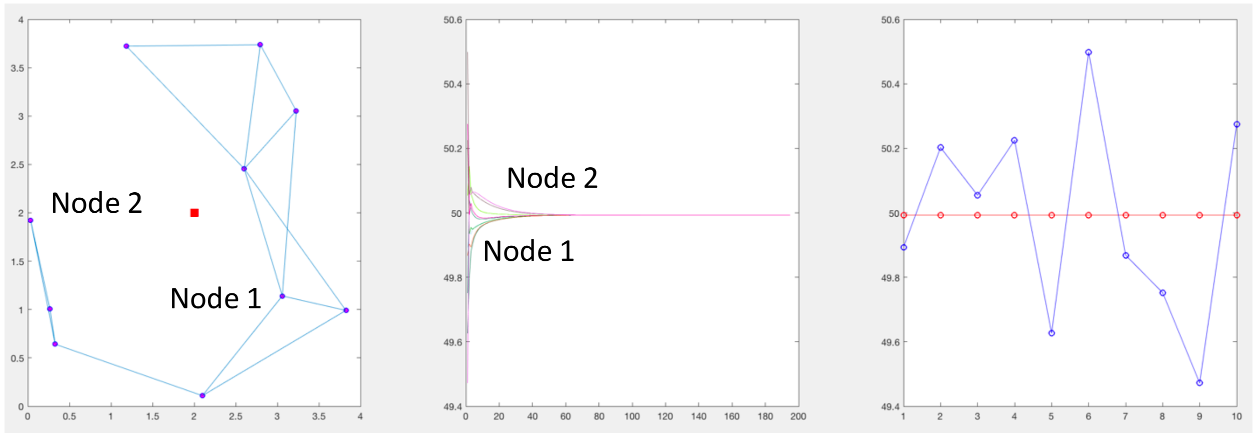

Figure 3: a) Graph b) Graph of each node's convergence c) Graph of initial and final values

find the consensus and convergence of the max degree average consensus algorithm. From figure 4 we can see that it converges faster than the metropolis convergence, converging at around 70 iterations. We can also see that like the metropolis and weighted 2 algorithms the difference in consensus values is negligible. In this particular example I included the convergence of the most connected node (node 1: green) and the least connected node (node 2: pink). We can see that node 1 converges much faster as it's consensus is probably more accurate and we can see that node 2 converges the slowest but over time it is clear that even the least connected of nodes will converge as long as it's part of the main graph.

The next part of the project was to repeat the metropolis and max degree average consensus parts from the first system except that we use a 20x20 grid size and a network of 50 nodes. For the metropolis part of this we can see from figure 4 that the consensus value is

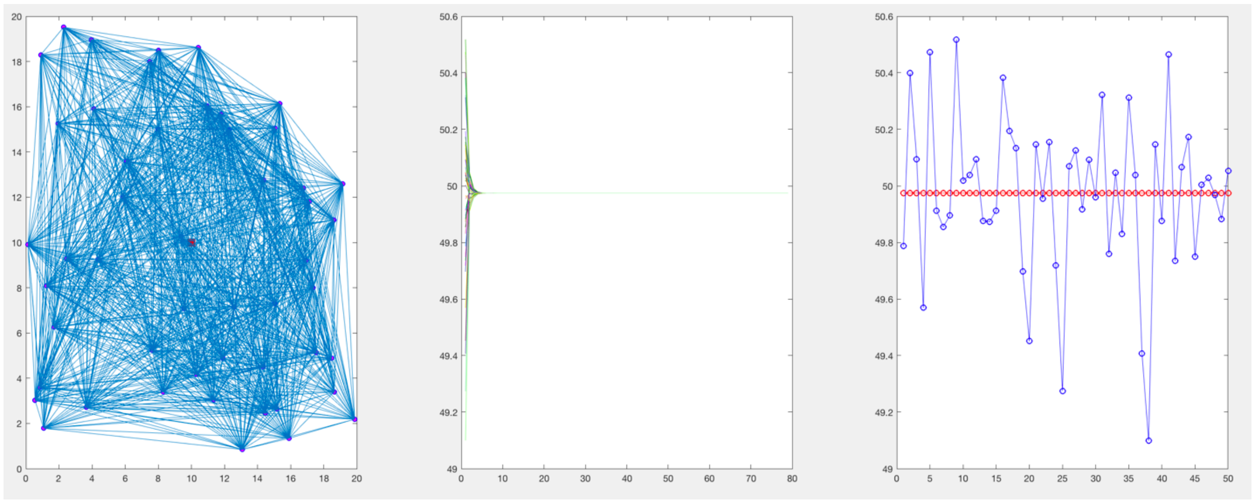
Figure 4: a) Graph b) Graph of each node's convergence c) Graph of initial and final values

converged to in less than 10 seconds, which is by far the fastest convergence of any of the

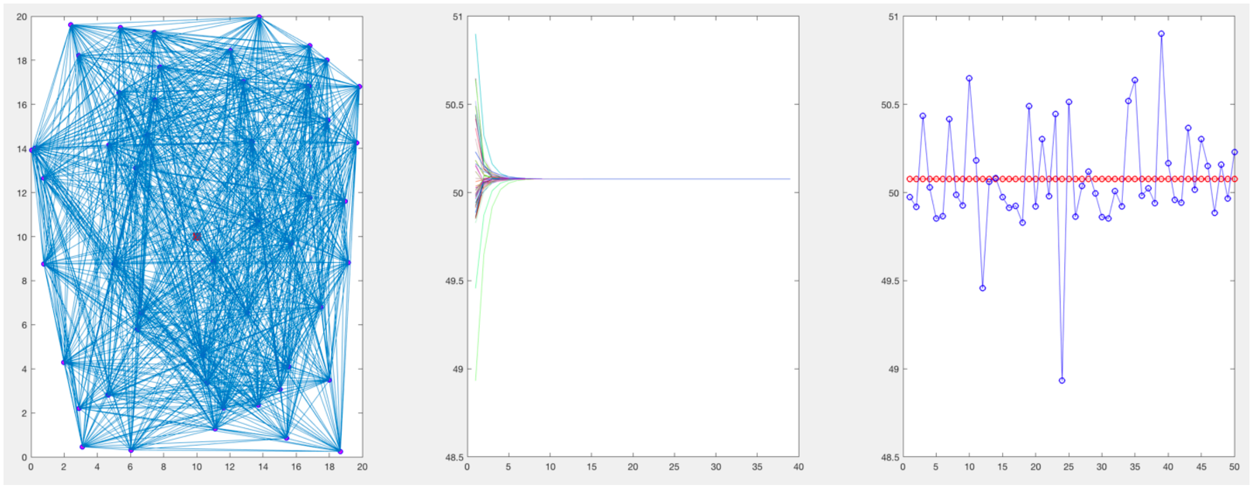
Figure 5: a) Graph b) Graph of each node's convergence c) Graph of initial and final values

algorithms used previously. However, the value is very close but not negligible to the target value. For the last part of section 1 we use the same dimensions and number of nodes but instead use the max degree average consensus filter. From figure 5 we can see that the response is a little slower than the metropolis version with a convergence over about twice as many iterations. However, the consensus value is roughly the same at being very close to the target but not quite negligibly different.

For part two we needed to design the same consensus algorithm but instead of measuring one cell we measured a grid of 25x25 cells. I ran two simulations with this model using first the weighted design 2 algorithm and second the average consensus metropolis algorithm. For the grid I used a Gaussian mixture modeled in both 2D and 3D with the graph overlaid. This is shown in figure 6. For the weighted design 2 algorithm the output was rather

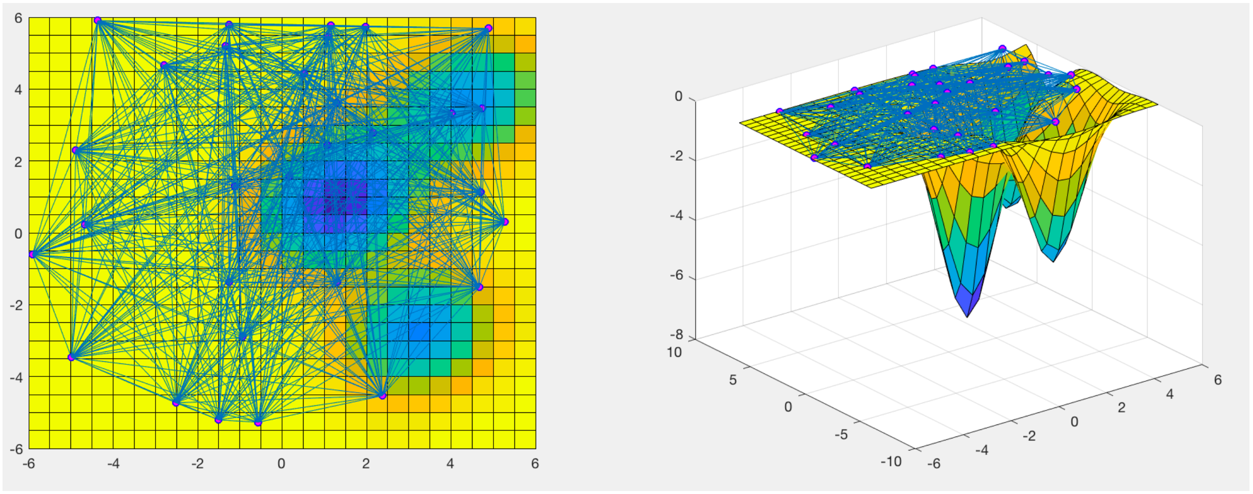

Figure 6: a) 25x25 grid with graph b) 3D 25x25 surface with graph

interesting in that the graph was nearly the same except that the areas if high intensity experienced a large error. We can see the output data in figure 7 and the error data in figure 8. The error function seems like it works badly for the high intensity parts and I'd be willing to bet that the consensus is better in a dynamic graph case. As a note I did do some convergence testing and it appeared that the system converged to these values every time I ran the

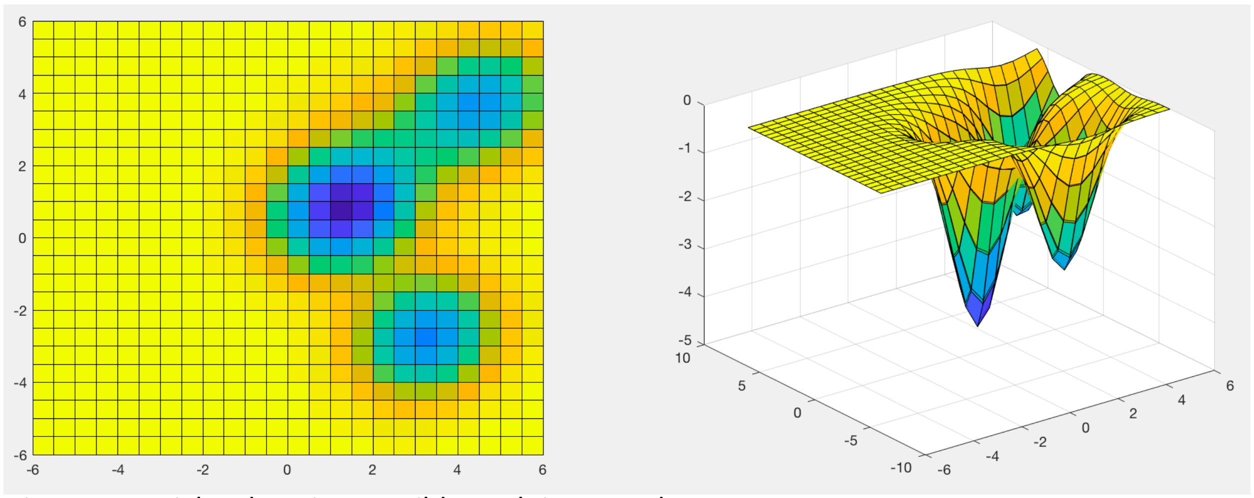

Figure 7: Weighted Design 2 Build Graph in 2D and 3D

simulation. We can see that the error is roughly 20-30% which is not bad for a static system. As for the imperfections in this data it seems to me that the high error is probably caused by a misplaced use of Matlab cells or maybe a wrong iteration call at some point because if you look through the code the algorithm is very clearly correct in terms of the equations. My current version of the code was redone to get rid of the cells and to iterate over every part of the arrays individually. Unfortunately, this doesn't appear to be working currently it seems as if all the cells are converging to the same value with a coupling factor of sorts. Because of this I used my original data as it clearly does solve the problem to some degree, just not 100% correctly. For the average consensus

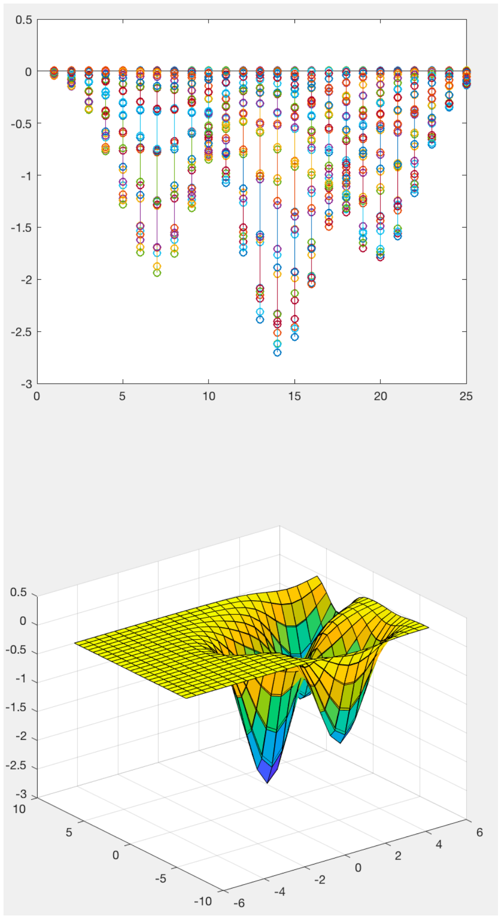

Figure 8: Weighted Design 2 Error in 2D and 3D

metropolis grid algorithm, we see much of the same response from figures 9 and 10 except that the error is larger. This does make sense given that it is an average consensus filter.

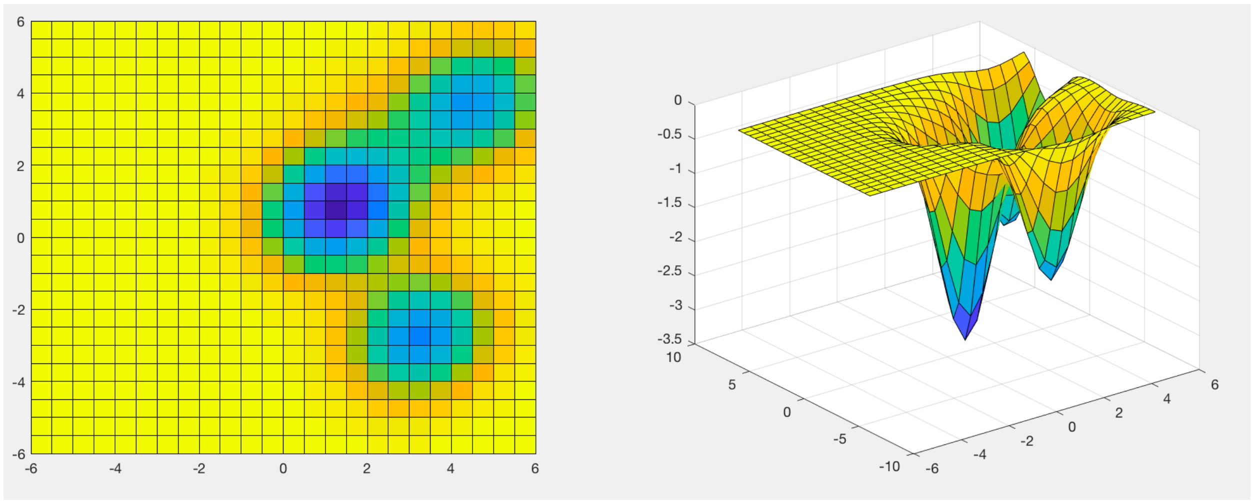
Figure 9: Metropolis Build Graph in 2D and 3D

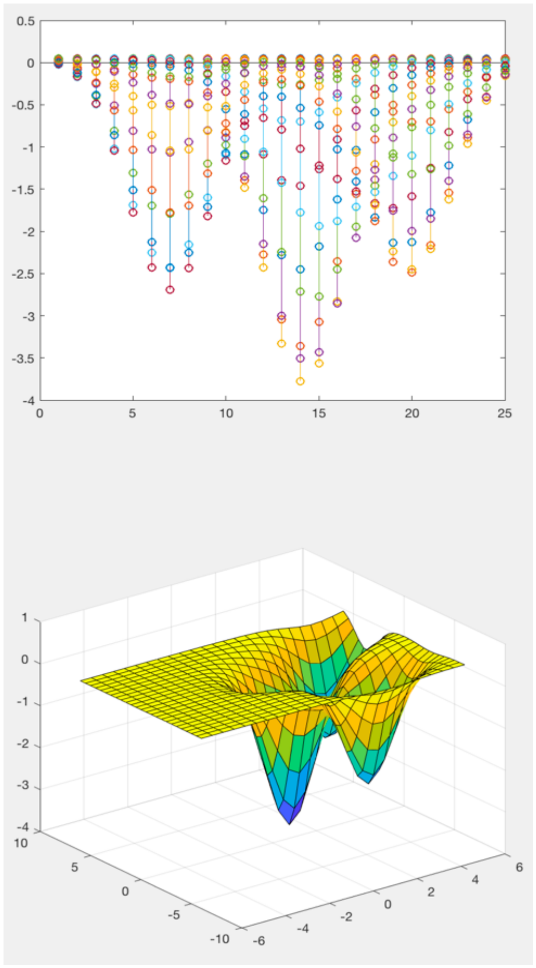
Figure 10: Metropolis error in 2D and 3D