IEMS 454 Final Project

Collin Erickson

March 18, 2016

1 Introduction

Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies by Krause, Singh, and Guestrin presents a solution to the problem of placing sensors in an environment that maximizes the mutual information. The paper was published in the Journal of Machine Learning Research in 2008. The paper has been influential in its field and has been has been cited 522 times on Google Scholar. The most important contribution of the paper is a greedy approximation algorithm that gives near-optimal results in polynomial time. This algorithm can be sped-up with further approximations to be scalable, which is required for big data sets. I implemented this greedy algorithm myself in R and some examples of results are shown below. Some of the point selections seem poor, which is either a problem in my interpretation of the problem or my code. The topic of this paper are closely related to my own research projects. There are many useful ideas that I learned from this paper that I will try to incorporate into my next project, which is creating a sequential experimental design algorithm. I will consider using mutual information in point selection since it makes more sense than other options. The speed-ups presented in this paper also seem like they will work for what I am trying to do, specifically lazy evaluation and inducing sparsity via a compact correlation function. However there are many differences in our assumptions, such as using a continuous space, using the responses gathered at previous points, and not knowing the correlation parameters.

2 Summary of Krause et al.

The goal of the paper is to tackle the problem of placing sensors at a set of points in an efficient way. The sensors take measurements at their location, and we want to be able to have a good estimate of the response over the entire surface. In experimental design terms, we want a space-filling design, in contrast to searching for an optimum. The authors choose to use Gaussian processes to fit the surface and make predictions. This is an extremely common choice in experimental designs since we not only want predictions but also the errors on the predictions. For this paper the authors say that the space they are

considering consists of a set of points at which they can place sensors and another set of points they are interested in but cannot place sensors at.

The other main decision is picking the optimization criteria. Entropy and alphabetical criteria are often used, but the authors decide to use mutual information based on their knowledge of the problem. A common problem in sequential design is that points are chosen near boundaries. This makes sense because those points will be farthest from other design points, and thus their uncertainty will be highest. However this is not what we often want because we can only use the information it adds on the side inside our design region, the other half is wasted. Mutual information measures the uncertainty over the rest of the space, which is what we are really interested in, and will help avoid placing sensors at the boundary. With this set up in place, the authors provide useful methods for solving this problem.

The authors prove that solving this problem is NP-hard. It is important that they prove this since it shows that no efficient algorithm exists to solve the problem exactly, so instead the problem becomes finding an efficient algorithm that provides approximate solutions. Nowadays data set sizes have become huge, making efficient algorithms necessary for it to be used in practice. The first approximation algorithm provide is a simple greedy algorithm, which means that the best option is chosen at each step even if it leads to inferior results further on. So in this problem the next sensor is chosen by finding which one gives the largest increase in mutual information. The authors describe this as picking a location that is central to the unobserved points. This algorithm is obvious and is not useful by itself. If the algorithm can give terrible results compared to the optimum result, then it provides no value to us. Thus we need a guarantee of worst-case performance. The authors prove that the greedy algorithm gives a result within $1 - 1/e \approx 0.63$ of the optimal design using the properties of submodularity since the mutual information function is submodular. This bound does not sound great, but on average the algorithm could be much better. An example they included found the greedy algorithm was within 5% of the optimal solution for small samples. They provide a way to calculate a more useful bound when actually running the algorithm that gives much tighter results. The complexity of the greedy algorithm is $O(kn^4)$, where n is the number of total points and k is the number of points already selected. Most of the complexity comes from maximizing the mutual information which is $O(n^3)$, which is too slow for large problems.

The complexity of greedy algorithm does not scale well, so heuristics are needed to make it a practicable algorithm. The first shortcut the authors provide reduces it by a factor of n. This is done by simply storing the mutual information difference for all potential points and only updating the values for those that could actually be selected. For example, a candidate point could be right next to another point already selected when there are still candidate points fairly isolated. It is a waste of time to update the mutual information increase of this bad candidate each iteration. This shortcut works because the mutual information increase is monotonic. If the gain in mutual information we would have by adding the candidate point is small, it will only get smaller after adding

other points. However, its complexity, $O(kn^3)$, still doesn't scale well, we usually want the complexity linear or log-linear for k and n.

The next method provided makes the algorithm scale much better by exploiting locality. Instead of using all sample points to make estimates using the Gaussian process, only a small number of points are used. This works well because the correlation between points quickly drops to zero. The mutual information calculation, which requires a matrix inverse, can be reduced from $O(n^3)$ to $O(d^3)$, where d is the number of points used in the correlation instead of all n points. This greedy algorithm is now $O(nd^3 + nk + kd^4)$ which is much faster if $d \ll n$. The optimality guarantee is reduced further and depends on d and the correlation between points that are no longer used in the correlation.

This paper is exceptional because it investigates the problem of sensor placement extensively and attacks it on multiple fronts. They proved that the problem is NP-hard and also provided heuristic algorithms. The greedy algorithm they provide is proven to be within a constant of the optimal solution, and then they provided speed-ups that also have bounds. Their speed-ups should make the algorithm scalable to large problems and they also show that placing sensors is robust against failure of nodes. They provide examples at the end of the paper that show their methods give better predictions at least as good as other heuristics in many cases.

3 Optimization

There is not a lot of advanced optimization in this paper. To solve the problem of finding the k best design points out of n options requires checking all sets of size k. Branch and bound can be used to get exact solutions, but it is still not practical. They mention other methods that can be used as heuristics, such as simulated annealing and pairwise exchange, but these approximations don't give performance guarantees. Similarly, linear relaxation of integrality constraints can be used, or the problem could be solved over a continuous space, but then there is no performance guarantee. Thus the purpose of the paper is to find a way to avoid any difficult optimization problem with a guarantee of a good solution. The greedy algorithm is the simplest possible optimization problem, just finding the best out of a list of options.

I tried to see how the large-scale optimization topics we learned in class can be applied to this problem but I couldn't find any way to do it. The objective function, mutual information, is highly complex and does not lead to subproblems. Also there is no useful way to generate constraints since the only constraint is a limit to the number of sensors selected. Moreover, if there were a useful large-scale optimization algorithm that I could find then the authors probably would have found it too.

4 My implementation

To see how the algorithm works I implemented the greedy algorithm myself in R. It was challenging to figure some of the notation, but eventually I got it working. The only problem is that they do not specify how to select the first point. I was able to get it working doing what I think it should be, but it was not clear from the paper. Problems easily arise from finding matrix inverses in R. An error is return when more than about 8 points in one dimension or 40 points in two dimensions are used because of matrix singularity. I have included graphs of results for one and two dimensions, and for designs where the points were selected uniformly and from an LHS maximin design, and $\theta = 10$ and $\boldsymbol{\theta} = (10, 10)$ are used in one and two dimensions, respectively.

Figure 1 shows five 1-D uniform designs, where each row is a replicate and 3 points are selected out of 8. The red points are the points selected and they have the number above to show the order they were picked. It seems like the first point is often a central point, which is good. Some of the other choices seem odd, such as in replicate 1 where all three points are near each other. The algorithm favors points that are close to others since we can get information on them. When I changed θ the points selected changed minimally, but it can affect whether it picks points spread out or points near other points. The value of θ would depend on the response function, and often we would estimate in an experiment. Figure 2 shows the 1-D LHS results, which gives no surprising results.

Figure 3 shows a single 2-D design with 50 points from a uniform design where 8 are chosen. Note that the dots are points not selected, and the red numbers represent the points and the order they were chosen. Some of these placements seem very bad, such as 3 being near 1 and 5 being near 6 while the upper right and lower left areas remain unexplored. This could be caused by the choice of $\boldsymbol{\theta}$, or by problems in my code. Figure 4 shows the same thing except the points were from a maximin LHS design. These results seem better, but points 4, 5, and 6 all seem too close to the edge, and 3 was placed close to 2 and 8 is close to 6.

Thus I have an implementation of the greedy algorithm that seems to be working well. For further study I could see how the changing $\boldsymbol{\theta}$ affects the points selected. It would also be useful to have the output data, y, to see if the points selected make sense. I could also try to implement the speed-ups, such as lazy evaluation, and see if I find the running time to agree with the complexity they give. This program will be useful for my future research since I will have to implement a similar algorithm that solves our sequential design problem.

5 My research

My research with Dr. Ankenman is related to sequential design of experiments. The project I am finishing now is just a study of Gaussian processes so I am familiar with how they work. The next project we will work on is creating a

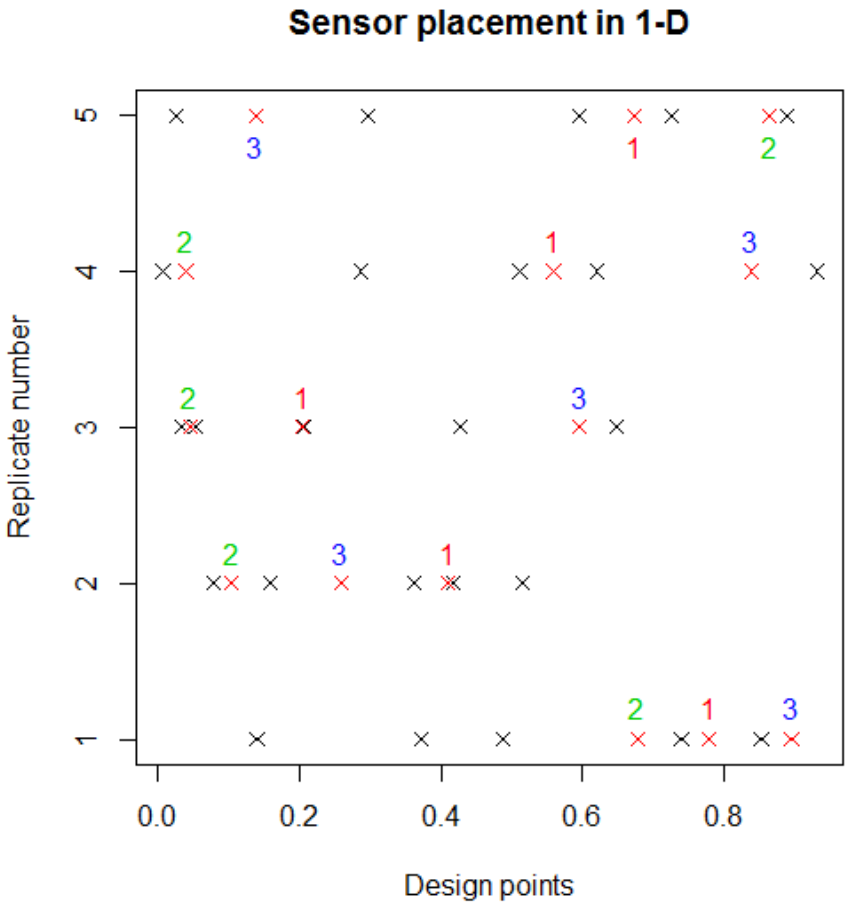

Figure 1: Sensors placed in 1-D using my R code using the greedy algorithm. Each row is a separate replicate where there are 8 design points and 3 are selected. The points were chosen uniformly.

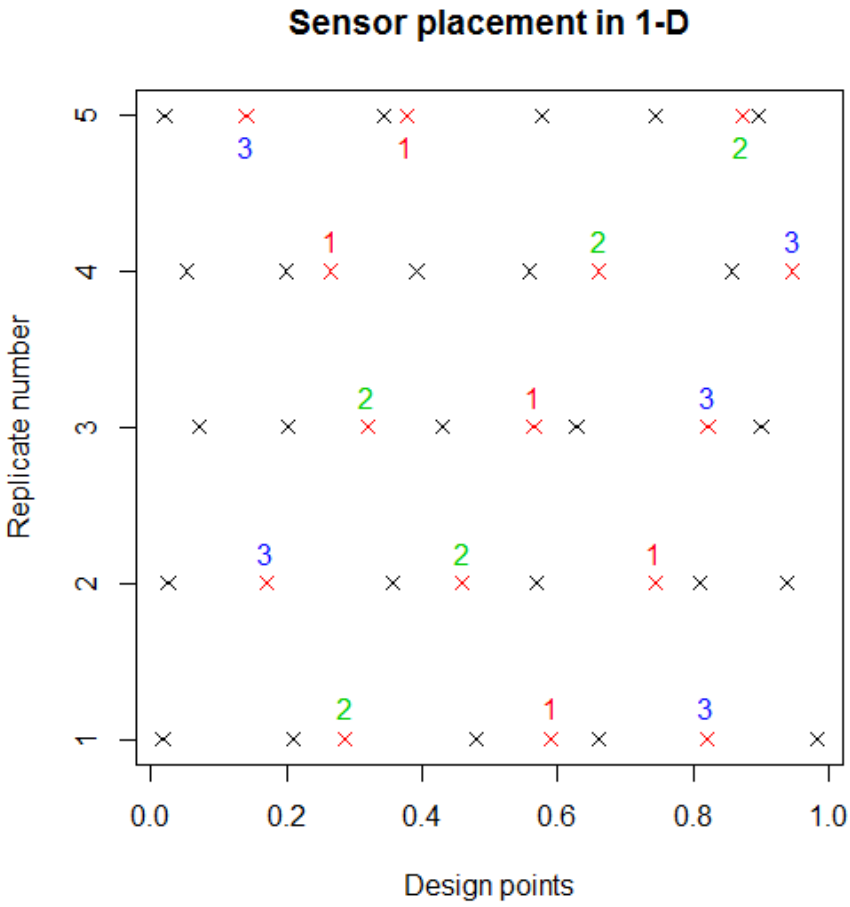

Figure 2: Sensors placed in 1-D using my R code using the greedy algorithm. Each row is a separate replicate where there are 8 design points and 3 are selected. The points were chosen with a maximin LHS design.

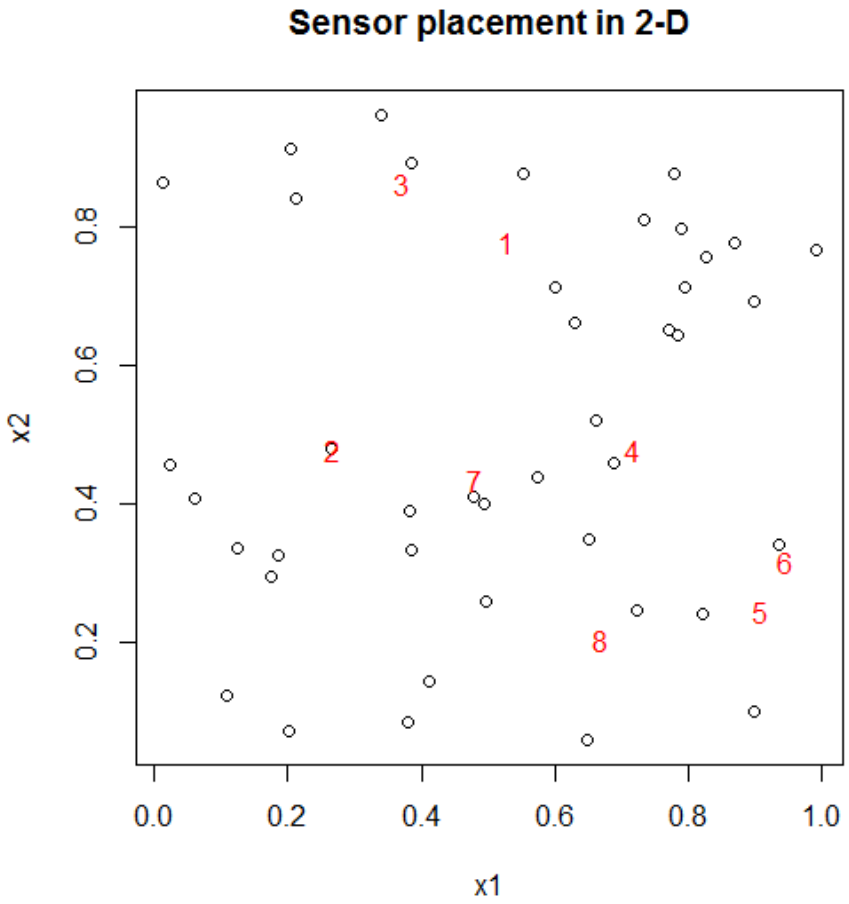

Figure 3: Sensors placed in 2-D using my R code using the greedy algorithm. There are 50 points and 8 sensors are placed, represented by the red numbers, the x's are points that were not selected. The points were chosen uniformly.

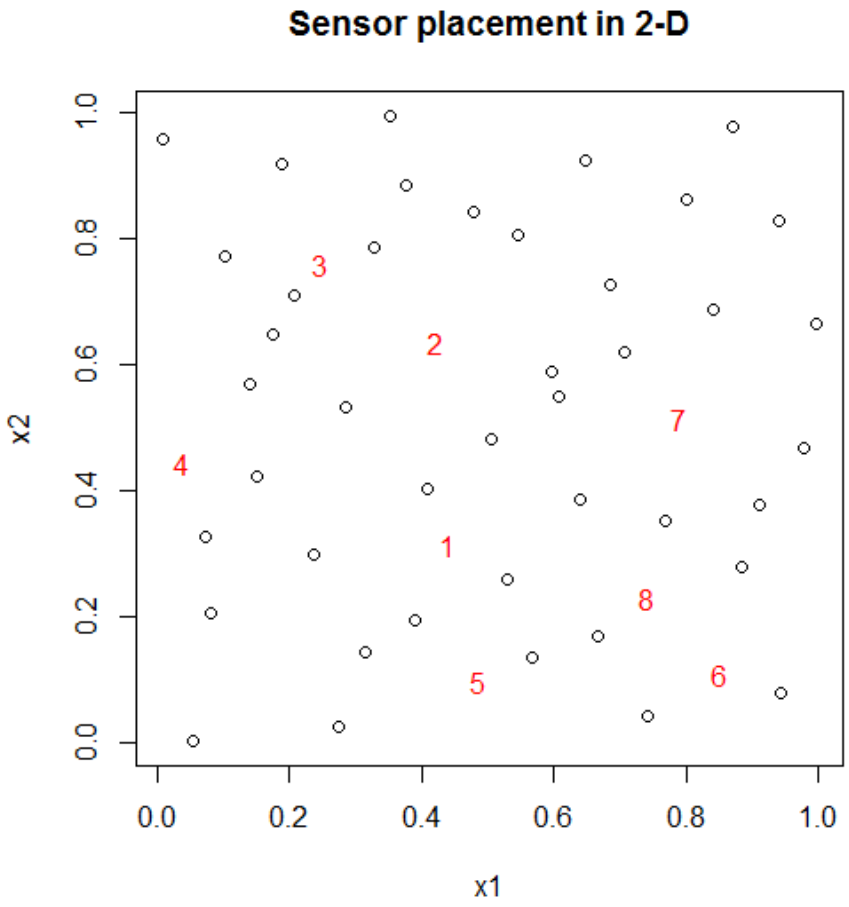

Figure 4: Sensors placed in 2-D using my R code using the greedy algorithm. There are 50 points and 8 sensors are placed, represented by the red numbers, the x's are points that were not selected. The points were chosen with a maximin LHS design.

sequential algorithm experimental design that maintains orthogonality, balance, and other desirable properties. I think I can use some of the ideas in this paper to help me on this project.

The greedy algorithm presented in the paper is essentially the same as a sequential design since a single point is selected in each step. Thus the ideas, especially for speed-ups, will apply for what we are trying to do. The lazy evaluation implemented in the first speed-up, where the improvement in mutual information is only at a point if it potentially can be selected, is a great but simple idea that could be useful. The setting for this paper is a little different since they are constrained to a fixed set of points while we will use a continuous space, but this speed-up idea can still be useful for us. We might keep a queue of candidate points, in which case we would not have to update the value for every point in the queue. The major speed-up they introduce is from exploiting locality. My study of Gaussian processes so far has been limited to the Gaussian correlation function, which is never zero between two points. The covariance function they use to induce sparsity, created by Storkey, could be useful for us to consider using. Our plan for the sequential design is to have it run until a stopping criteria is met, which could require huge amounts of points. If we used the Gaussian correlation, the correlation would get bigger with each point added, so the algorithm would be slower as it goes along. This is highly undesirable, so we will consider using a method like this to limit the number of points used in the correlation. A similar idea we were going to investigate was that of Gramacy and Apley in *Local Gaussian process approximation for large computer experiments* (2014). Their algorithm finds a subset of points that gives the best predictions at a given points, and scales very well.

The concept of mutual information is also new to me. Papers I have read usually use maximum entropy or an alphabetical optimality criteria, or simply picking the point with the highest predicted variance. It seems that mutual information is better in concept since it gives the most information at points not selected, which is generally what we want. However, this paper assumed they cared only about a certain set of points, while we will care about the entire space. We could pick representative points, essentially a MC integral approximation, but this would be expensive. The sparse correlation function would make this easier. We will have to consider this more in-depth when we have a better statement of our problem. Another idea we are thinking of using the information from points already run is to use the idea of Roshan et al. in *Sequential Exploration of Complex Surfaces Using Minimum Energy Designs* (2015). In this paper they provide a way to select points that encourages space-filling. We could use the uncertainty (instead of the density) to repel points from being too close in areas that have low uncertainty. I'm not sure if this is compatible or contrary to mutual information, but it is something I will look into more later.

A major difference between what we'll do and this paper is that they assume they know the correlation parameters, $\boldsymbol{\theta}$, for the Gaussian process. We will not know these parameters since they are inherently unknowable. Instead we will estimate the parameters using the data. This is a major difference: Krause

et al. pick their points before making any measurements, whereas we will be taking measurements at each point selected and using this information. The parameters they use, which are for the standard Gaussian process, assume that the correlation parameters are the same over the whole space. There are many flexible models, such as Gramacy and Apley's, that allow these parameters to vary over the surface, which is often what happens in practice. Thus the points we select will be dependent on the previous responses since they affect the correlation parameters, especially for local models.

6 Conclusion

In this project I have studied the work of Krause et al. which tries to solve the problem of placing sensors efficiently. The goal is to, given a set of points that we care about and can place sensors at, pick the optimal set of sensors so that the mutual information is maximized. They prove that the problem is NP-hard, and give the next best thing: a polynomial-time algorithm that performs within a constant of the optimum using the properties of submodularity. The algorithm is greedy, meaning it selects points iteratively, picking the point that will give the largest increase in mutual information at each stage. They also provide some speed-ups to this algorithm to make it scalable for large data sets, which is necessary in practice. They also discuss many other useful topics related to their algorithms, such as its robustness against failure of nodes and parameter uncertainty.

I implemented the greedy algorithm in R and got it to work, albeit with questionable results. This is good for me to do since I will have to do something for my own research in sequential designs that maintain favorable properties such as orthogonality. My own research is closely related to this paper by Krause et al since a sequential design is often essentially the same as a greedy algorithm. Thus I hope to use some of the ideas for mutual information and algorithm speed-ups in the algorithm I design to solve my problem. The major difference in sequential design is that the responses gathered up to the current time can be used to guide the current selection, so I will find a way to incorporate this knowledge. There are also other differences in my work and this paper, such as having a continuous space instead of a set of points and not knowing the correlation parameters, that I will have to look at other papers for ideas on how to tackle them. The work of Krause et al. is very good because of its important contributions, and it will help me in my future research projects.