GerryChain Practice Day 4: Acceptance Functions

Daryl DeFord

June 26, 2019

1 Introduction

The main purpose of today's session is to explore the use of acceptance functions in GerryChain, in order to transform our Markov chains into MCMC samplers. This is one of the more complicated aspects of GerryChain and it incorporates all of the skills we have practiced this week. Like constraints, acceptance functions take as input a partition and return a True or False value, although they usually compare some value of the partition currently under consideration to the previous state of the chain. For more details about the theoretical aspects of acceptance functions, see Sections 5 and 8.5 of my notes. There are several additional example acceptance and annealing functions implemented here for grids.

One of the overall goals of these sessions is preparing you for the YOUR STATE project. Hopefully some of the functions you design in these sessions will make it in to your presentation ☺

1.1 Python Code

The initial code for these investigations is in a separate repo in the VRDI organization. The template for this session is a slightly modified version of the complicated grid chain from the first week templates. You should be able to do everything in a Jupyter Notebook or whichever other IDE you used on Friday for the original templates. The more detailed guide is a good reference for more examples.

2 Acceptance Functions

One of the most natural uses of an acceptance function is optimizing for a specific metric, say increasing compactness or decreasing population imbalance or county splits. If you read about these methods online it is easy to get the false impression, particularly from those who come from a continuous background, that this "just magically works" and will solve all of your problems rapidly. For our problems, tuning the values in order to navigate the space of plans is difficult, particularly when you are trying to optimize for an integer valued measure that that is centrally distributed. For each of the examples below, focus on optimizing the cut edges score. Then try them for the county split function, I added a couple of silly county definitions to the grid constructor, you might wish to pick a more reasonable one[1].

You should also think about how to balance populations with acceptance functions. One approach is to measure the difference between the maximum and minimum population district but you could also measure the maximum distance from ideal or the average absolute deviation from ideal. Each of these leads to different acceptance rates – which one works best?

- First, implement an acceptance function that always accepts an improving value and accepts a worse value with a fixed probability of .05. Run some chains with this acceptance and then vary the .05 to .1, .01, and .001. How do these modifications change the output?

[1]like splitting the grid into 4 quadrants.

- Next, modify your acceptance function so that it depends on the ratio of the old score to the new score[2]. Run some chains with this acceptance and then try raising the ratio to some integer exponents. Does it seem to impact the types of outputs you see?
- Finally make the acceptance function exponential in the difference between the score of the previous state and the proposed state. Again, run some chains and then vary the base of the exponential. How does this change the runs?

3 Annealing

Another use of the acceptance function changes the acceptance probabilities with the length of the chain. A common approach is to do some number of unconstrained steps and then slowly increase the effect of the bound on the acceptance function. This is implemented in the template in the function `annealingcutaccept2`. This version uses an exponential weighting on the number of cut edges multiplied by a value β, which represents the temperature. For the first 10,000 steps the temperature is zero, so the chain is moving towards uniformity. The next 30,000 steps linearly interpolates $0 \leq \beta \leq 3$ and the last 10,000 steps have temperature 3.

You should start by modifying the values of the annealing function in the current chain. Note that you may want to shrink the size of the grid or the total number of steps [3] in order to work through more examples.

- Vary the amount of time that the chain spends in each temperature. How does this change the cut edges plot? How long does it take to get to a minimal value, as a function of the number of steps that you spend with $\beta = 0$?
- Try replacing the part that linearly interpolates $0 - 3$ with a fixed value of 1.5. How does this change the cut edges plot?
- Next, split the interval up into a series of discrete steps .5, 1, 1.5, 2, 2.5. How does this change the cut edges plot?
- What happens if the minimum value of β is greater than zero?
- Does the chain ever find a plan that is more compact than the number of cut edges in the stripes plan?

Next, build your own annealing functions using your acceptance functions from Section 2[4]. You should try varying the lengths and the functions you are trying to optimize for. In particular, an interesting question is how many steps it takes to both reach maximally bad and maximally good plans for each measure. Another thing to try is repeated annealing where instead of simply cooling the temperature from infinity, you instead heat and cool and heat and cool in succession. Is this enough to save the Flip proposal?[5]

- What happens if you try to do annealing on the county splitting values?
- What happens if you try to anneal for both population and cut edges?
- In Section 2, we experimented with different types of functions for determining the acceptance probability. What happens when you try to do annealing with the ratio of the cut edges instead of the exponential version?
- Can you convert a plan with 20% population imbalance to one with 5% imbalance? How many steps does it take?
- Most of our examples above use the flip step - what happens if you try it with ReCom instead?

[2]be careful not to divide by zero ☺

[3]and the corresponding number of steps in each temperature regime.

[4]One potentially nice application of this approach would be to use annealing to create a population balanced plan from one that is initially unbalanced.

[5]No ☹