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].

- 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?
- 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?

[1] like splitting the grid into 4 quadrants.
[2] be careful not to divide by zero ☺

- 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. Start by playing with the values of the annealing function in the current chain.

-
-

Next, build your own annealing functions using your acceptance functions from Section 2.

-