Caleb Kalisher

Comp Bio

5/10/16

Final Project Report:

I chose to do the Gibbs Sampler, as it was an extension of a Random Sampler, which gave me trouble earlier in the semester. By completing this, I saw it as fixing a past failing and taking it one step further.

For my data I used the toy set provided in the book and the larger set we used in HW6. I was planning on creating my own data set, but I did not get around to accomplishing that.

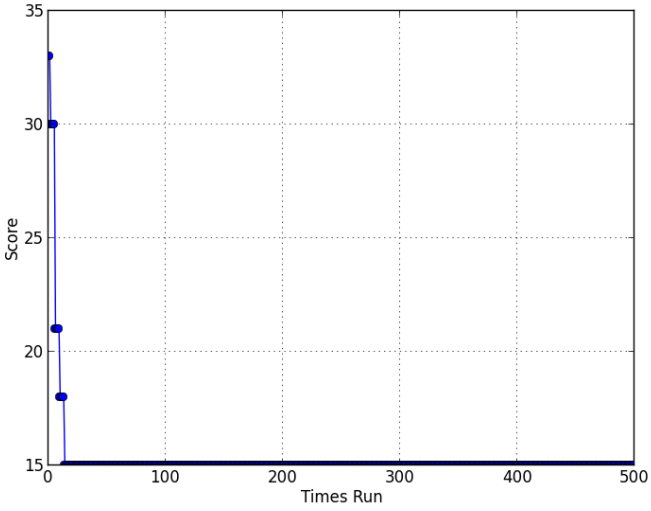
First, my program selects random k-mers (of a given length) on each string in a given dataset. Then, it goes into a while loop that allows for an adjustable number of randomizations. In the while loop, it selects a random string, and then a random kmer on that string. After it accomplishes that, it checks the score of the new, one-k-mer-adjusted set of kmers against the old set. If the new score is greater than the old score it retains the changed kmer. After it has run through loop as many times as you have asked it to, it will report the lowest score that it found as well as how long it took to find it.

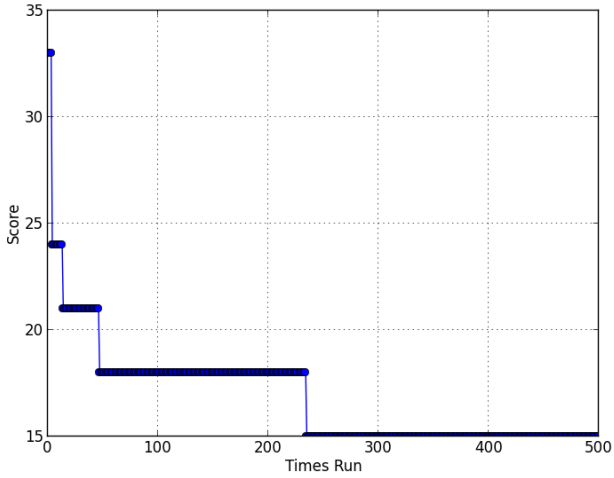
I attempted to give this program a contingency loop that would stop randomizing knew kmers after it had found a given number of unchanged kmers in a row. It did this by adding an additional loop on the outside of the Gibbs loop mentioned above. The contingency loop found a “lowest score” at about five times

as fast as the regular Gibbs sampler. So in that way, it was a success. However, if you set the contingency too low, it would not find *the* lowest score everytime.

Finally I also just have a basic random sampler that changes all selected kmers for each string and compares it against the best one found so far.

I tried to collect various forms of data to determine accuracy/efficiency of the programs. The primary measures were time it took to run each program, and the size of smallest kmer found. The Gibbs Sampler and the Random Sampler both were able to their lowest score at about the same time. The Contingency Gibbs Sampler was about five times as fast as the other two. However, the Gibbs and Random sampler regularly found the lowest score, however the Contingency Gibbs Sampler was highly dependent on if you gave it enough trials so it wouldn't get hung up on a "plateau. "





For example, if you look at the graph above and to the left, the Contingency Sampler would accurately find the lowest score (15) and cut it off as long as you gave it a contingency counter of above fifty. However, if you look at the graph to the right,

that same counter of 50 would have the sampler stopping in the middle of the plateau of the score 18, and therefor be less accurate than the regular Gibbs.

Unfortunately, these are all based in relatively small data sets. However, it seems safe to assume that the efficiency found by the Contingency Gibbs Sampler would just show more impactful differences in time improvement. That being said, the trick would be finding where the appropriate cut off point would be, as to avoid losing the potential best score in the name of efficiency. While the regular Gibbs sampler and the Random sampler did not differ substantially in their ability to find the best score, it may be that differences would arise if we were working with a larger data set.

Overall, the Gibbs Sampler worked, and has the potential to be improved by a contingency parameter. The tricky part is figuring out a reliable number for the cut off point for that contingency.