CSC665 Homework 4 (Project Proposal)

Aaron Blumenfeld

March 28, 2017

What you are going to implement

I would like to implement k-means clustering and k-means clustering with constraints to study when the problem of (extremely) unbalanced clusters comes up. According to some sources (such as the MSR paper linked to below), this situation specifically comes up for large values of k in high-dimensional space. They specifically mention the values of $k \geq 20$ and $d \geq 10$ (d is the dimension of $\mathbb{R}^d$). I would also like to know if and when this situation occurs for smaller values of k (say, $k \approx 4$).

It would be good to know if this happens as a global property of the data set, or if it happens because we made poor initial choices of centroids of our clusters. Below are a few papers that describe the relevant algorithms:

`https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2000-65.pdf`
`http://users.cis.fiu.edu/~taoli/pub/clustering-size-constraint.pdf`
`https://public.fh-wolfenbuettel.de/~klawonn/Papers/hoeppnerklawonn08.pdf`

How it relates to your work

Last year I put together a website (which can be seen here: `http://www.blumenfeldcalc.com/series`) that generates and grades calculus (specifically sequences/series, for now at least) exams that adapt to the level of the student. That is, if a student does poorly on a test, the next test s/he gets is easier; conversely, if a student does well on a test, the next test is harder. I built this in the simplest way possible, without any knowledge of machine learning or data science or anything related.

So I thought it would be interesting to use something I learned from this class to improve the back-end. In particular, to dynamically change how I determine which problems are easy or hard. At present, each problem has a statically set difficulty level between 1 and 4 (1 being easy, 4 being hard). I did this based on years of experience teaching calculus, and while I think my judgment on this was fairly good, it's hardly objective.

So one technique would be to associate a vector to each problem describing its difficulty (which would change based on how users do on specific problems) and use 4-means. But a number of sticky situations could theoretically arise, not the least of which is having empty or near-empty clusters. Thus I would want to guarantee lower bounds on the size of each cluster.

Here is one possible technique for creating a vector for each problem. Partition the interval $[0, 4]$ (the range of grade levels) into r subintervals. For example, we could choose $r = 5$ and write $[0, 4] = [0, 2) \cup [2, 2.8) \cup [2.8, 3.2) \cup [3.2, 3.5) \cup [3.5, 4.0]$. Then whenever someone with grade level in subinterval i gets a problem right, add 1 to the ith component of the vector (which is r-dimensional). Subtract 2 if they get it wrong. (Separate weights to account for lucky guesses since the problems are multiple-choice.) Initialize vectors to $(0, \ldots, 0)$. Using k-means here would be slightly easier in that we have initial cluster assignments and may be able to easily come up with initial values for the centroids. We could also set r (and even k if we wanted finer-grained difficulty levels) higher and see what happens in higher-dimensional space since the MSR paper claims this really becomes an issue for large d.

How will you know that you succeeded?

By answering the question of when and why certain clusters in k-means are small (or even empty), by implementing k-means with constraints to fix this problem, and by determining whether this algorithm will work for the vector construction described above (or if perhaps another method is needed to map a problem to a vector).

How will you know that the technique is good?

By coming up with simulations that would be realistic based on the application. This could include simulations where the initial clusters are all correct, and ones where they need to be corrected. For example, maybe some level-3 and level-4 problems need to be swapped.

It would also be good to use knowledge of why highly unbalanced clustering can occur and determine if this would realistically happen in this application of k-means.