A K-Medians Clustering Algorithm for Somatic Mutation Detection

Scott Kramer

November 30, 2015

Abstract

I present a method for predicting somatic mutations as an entry into the ICGC-TCGA DREAM Somatic Mutation Calling Challenge (SMC-DNA). My algorithm uses a combination of k-medians clustering and statistical analysis to make its predictions.

Introduction

Somatic mutations are “alterations in the DNA of a cell that can be passed to the progeny of the mutated cell in the course of cell division”[1]. These alterations are often the root cause of cancer. A variant caller is an algorithm that predicts which bases in a DNA sequence are the result of a somatic mutation. If a reliable variant caller algorithm existed, scientists would know which somatic mutations have occurred in each cancer patient before their tumor has grown large enough that it can be analyzed. This would bring scientists one step closer to developing improved, customized cancer treatments for each patient.

The ICGC-TCGA DREAM Mutation Calling challenge was a crowd-sourced competition to develop the best variant caller. While the contest produced many good variant calling algorithms, the accuracy of these algorithms was not perfect. Heavy dependence on parameters as well as customized data calibration and filtering make it risky to fully trust the predictions by a single caller[2]. Thus, the organizers of the contest realized that an algorithm that incorporates the results of multiple variant callers will produce better predictions, and created the ICGC-TCGA SMC-DNA Meta Challenge to crowd-source the creation of such an algorithm.

Running a large number of variant callers is computationally expensive. Therefore, while the algorithms developed for this contest can look at the past performance of all of the variant callers, they can only choose up to five variant callers to incorporate into future predictions. Contestants are given the output of various variant callers on four synthetic tumors to use as training data. Once the contest deadline has past, algorithms will be ranked according to how they perform on ten real tumors. The scoring metric for the ranking will be the median F_1 score[3].

K-Percentile Vote

The first step in my algorithm is to determine what percentage of variant callers need to predict a given somatic mutation before one should predict the somatic mutation. While intuitively one

[1] http://www.britannica.com/science/somatic-mutation)
[2] https://www.synapse.org/#!Synapse:syn4588939/wiki/233672
[3] F_1 = 2 * precision * recall / (precision + recall)

might think that it makes the most sense to go with a majority-vote, this might not yield the best results. Below you'll find the percentage that optimized the F_1 score for each of the four synthetic tumors.

Tumour	Best Percentile	F_1 Score at Best Percentile
1	40	0.9665
2	40	0.9689
3	23	0.9770
4	21	0.8925

Overall, the percentile with the greatest median adjusted F_1 score is 40, with a median F_1 score of 0.9677. Thus, it is optimal to predict a given somatic mutation if and only if at least 40 percent of variant callers predict it.

K-Medians Clustering

As mentioned in the introduction, it is not computationally feasible to run every variant caller in order to make somatic mutation predictions. Thus, my algorithm needs to determine which callers are representative of the entire set. To do so, it uses the K-medians clustering algorithm with K=5. The K-medians algorithm computes K clusters with a centroid point in each cluster equal to the median of all of the points in the cluster. As a result, it minimizes error over all clusters with respect to the L^1-norm distance metric[4]. In my implementation, the set of all of the predictions a variant caller makes is considered one point. Thus, if there are V variant callers, then there will be V points that are used to determine the five clusters.

I use the K-medians clustering algorithm over the more popular K-means algorithm because I want each prediction that differs between two variant callers to have equal weight. For example, imagine there are two spots in the sequence at which two different variant callers (VC1 and VC2) are asked to predict whether a somatic mutation has occurred. At spot 1, VC1 predicts that there is a somatic mutation, while VC2 predicts that there is not a somatic mutation. At this point, both the L^1-norm and L^2-norm[5] distances between VC1 and VC2 are 1. However, if at spot 2 VC1 again predicts that there is a somatic mutation, while VC2 predicts that there is no somatic mutation, the L^1-norm distance will be $(1 + 1) = 2$, but the L^2-norm distance is only $\sqrt{1^2 + 1^2} = \sqrt{2}$.

Once the K-medians algorithm runs, I choose the point in each cluster that has the shortest L^1-norm distance to the centroid of the cluster as being the variant caller that is most

[4] For an explanation of the L^1-norm distance: http://mathworld.wolfram.com/TaxicabMetric.html
[5] more commonly known as Euclidean distance

representative of the cluster. I now only need to run these five variant callers to predict future tumors. To make a prediction, I add a weight of |Ci| for each of the five variant callers that predict a somatic mutation, where Ci is the set of points in cluster i. If the total weight is greater or equal to 0.40, then I predict a somatic mutation. Thus, my algorithm is able to mimic a 40th percentile vote while only needing to run five variant callers on each new tumor. As you can see in the table below, my algorithm performs almost as well as a 40th or 50th percentile vote, and performs much better than the median variant caller.

Tumour	40th Percentile Vote	50th Percentile Vote	Median Variant Caller	My Algorithm
1	0.9665	0.9655	0.8818	0.9748
2	0.9688	0.9661	0.9130	0.9401
3	0.9728	0.9637	0.9319	0.9645
4	0.8606	0.8375	0.7901	0.8725
Median	0.9677	0.9649	0.9225	0.9523

Conclusion

I was able to use the skills I developed during this course to solve a real-world bioinformatics problem. My algorithm for predicting somatic mutations from only five variant callers performs well on the synthetic tumors, and currently performs better than the two other entries that have been submitted as of December 2nd, 2015[6].The contest runs until February 28, 2016, at which point the leaderboard for the real tumors will be released. I plan on continuing to improve my algorithm and entering a submission before the deadline.

[6] Leaderboard located at https://www.synapse.org/#!Synapse:syn4588939/wiki/233694