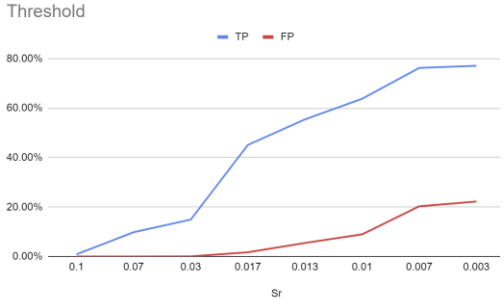
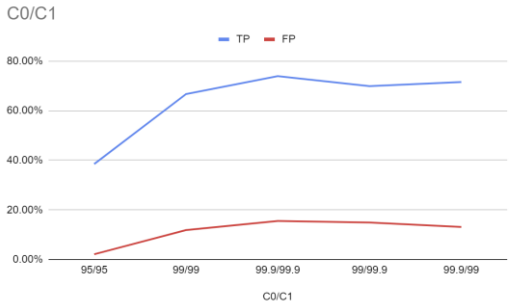
NegSelReport

Name: Damian Elkins User-name: frcg69

Two-letter code for your chosen negative selection algorithm: VD

I started off with coding up the base algorithm and tweaking the various parameters to see what kind of balance I could strike. My methodology was as follows: 1. Start by making adujustments of relatively large amounts to the variables to understand how each changed the performance. 2. From this make fine tuning tweaks to the individual variables one at a time to gradually increase performance. 3. Create 5-10 test detector sets of the final candidate solutions and select the highest performer. From this I selected my best set which will act as a baseline for future. For the value I selected 0.01 as it had the best tradeoff between TP and FP in my opinion though 0.013 is also quite good. From there I altered C0 and C1 in tandem keeping them both equal and then altering independently when I felt close to the best with the 0.01 self radius. From this I selected the best to be 99.9/99 which has roughly the best ratio without the extra time that comes from 99.9/99.9. The previous Sr had been looked at with 99/99 so I went back and retested with the new ratio and 0.01 was still the strongest choice. Upon further investigation I decided to try some more combos that made sense. I felt that by increasing the number of attempts to place detectors would obviously increase the TP rate but by increasing the self radius I would be able to offset the increase in the false positive rate. So I tried a few values a bit above 0.01 with the 99.9/99.9 and found that 0.02 was best giving me a TP of 70.95% and an FP of 7.45%. This wasn't determined through testing but built off of my intuition developed from the testing.





From here I wanted to look at the research and see what I could do to improve. I discovered tolerant v-detector algorithm which aims to ignore outlier members of self by using the K nearest self in the training set. The algorithm requires different detector representation making it an n+2-dimensional vector. I decided to adapt this to fit into the framework required. My adaptation finds the k closest elements of self and then averages them out to get the tolerant distance. This is then cut down to the distance to the nearest self with a much tighter self-radius if the tolerant distance is greater than the distance from the closest to the detector. The aim is to increase coverage in much more sparsely populated areas of non-self while maintaining the accuracy on the denser borders. This immediately created an increase in TP with the previous parameters and a k of 3. This, however, was met with a tripling of the FP with only a 1.14x increase of TP so I started experimenting. I found that the increase in TP was roughly equal in % to an increase in FP however FP being lower would have a much larger jump. This remained true for all new parameters I changed, that being the scaling factor of the threshold and k. With the scaling factor having the largest impact. Within my testing I was not able to improve upon my previous score. I feel that while this modification would work well on a sample set with a large number of outliers, the provided dataset doesn't have enough for the increased coverage to offset the impact of false positives.