NegSelReport

Name: Aleksandar Yordanov User-name: jvgw34

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

Implementation:
I have chosen to implement a basic version of the V-Detector algorithm detailed in the lectures. My enhancements for NegSel and NatAlg focus on optimisations using NumPy, such as faster Euclidean norm calculations and efficient random population generation.

Experimentation and insight:
The goal of my experimentation was to identify optimal values for c0, c1 and self-radius to achieve a high detection rate and low false positive rate. Specifically, a detection rate of over 80% and a false positive rate under 20% as a baseline. To meet time constraints, I modified the algorithm to output the detector set when a time limit of 12 seconds was hit.

I started with initial values of 0.001, 0.1 and 0.9. These base values led to a detection rate of 87.05% but a false positive rate of 26.95%, rendering the detector set unreliable. The first variable that I decided to modify was the self-radius. Decreasing radius below 0.001 increased false positives without improving the detection rate. Increasing the radius led to reductions in false positive rate while keeping good detection rates. In this report I test radii from [0.005, 0.01, 0.1].

My testing methodology involved, for each radius value, running tests for c0 = [0.01, 0.1, 0.2], c1 = [0.8, 0.9, 0.99]. Taking the best result where false positive rate is minimised and fine-tuning c0 ± 0.05, c1 ± 0.05, identifying any potential improvements.

A radius of 0.005, starting at c0 = 0.1, c1 = 0.9 returned a detection rate of 77.95% and false positive rate of 16.80%. Increasing c1 to 0.99 led to a detection rate of 84.45% and false positive rate of 20.85%. Further testing of combinations of c0 between [0.01, 0.2] and c1 between [0.9,0.99] yielded a detection rate of 84.40% and false positive rate of 19.40% for c0 = 0.2 and c1 = 0.95. This was the best result achieved with a radius of 0.005.

In all of these trials, the number of detectors found hit the imposed 12-second maximum instead of terminating when providing adequate coverage of the data space. Evaluating these results without a time limit led to a detector set of ~1900 being returned with detection rates of 90.25% and false positive rates of 29.75%. Indicating that there is redundant coverage and the detector set is unreliable.

A radius of 0.1 provided 1 to 4 detectors regardless of c0 and c1. Leading to poor coverage and early termination. For this reason, I went with a radius of 0.01

A radius of 0.01 provided a detection rate of 76.05% and 12.45% with a c1 of 0.99, and 48.95% and 3.05% with a c1 of 0.9. Testing further combinations of c0 = [0.01,0.1], c1 = [0.9,0.99]. A value of c0 = 0.02 and c1 = 0.99 led to a detection rate of 79.40% and false positive rate of 12.95%. For a radius of 0.01, this was the best value I achieved. In each test case, for a radius of 0.01, the algorithm consistently terminated when an adequate coverage was reached, reducing redundant coverage and providing good detection and false positive rates. For this reason, I decided to use a radius of 0.01 with c0 = 0.02 and c1 = 0.99

Performance constraint testing:
Testing on low power mode in macOS (approx. 40% CPU slowdown). A radius of 0.01 terminated due to time and provided coverage rates of 74.55% and false positive rates of 9.70%, indicating that adequate coverage and error rates are still achieved under performance constraints.