K Means Clustering of StarCraft strategies

ANTHONY OU & BRENDAN ANDERSON

HTTPS://GITHUB.COM/CMPUT350UALBERTABOTSTRATEGY/REPLAY-PARSER

Our motivation

The current UalbertaBot currently makes no effort to match or predict strategies.

When compared to the behavior of professionals, this is extremely naïve. Strategic choices are mapped on a strategy-to-strategy basis. Rock beats scissors, scissors beats paper. If you can understand that your opponent is setting up to play scissors, you can prepare some rocks.

Starcraft in particular is designed with hard counters.

Our Solution

- To solve this problem, we have attempted to impliment a strategy clustering system. The goal is for our module to be able to answer this question at runtime of the bot:
- "Does what my opponent is doing look like a Rock or a Scissors or a Paper?"
- **The System:**
- Develop a database of parsed out strategies from professional replays
- Cluster into some manageable number of similar looking strategy clusters
- Provide a similarity rating for a Live strategy to the clusters.

Replay parsing to file implemented into UAlbertaBot

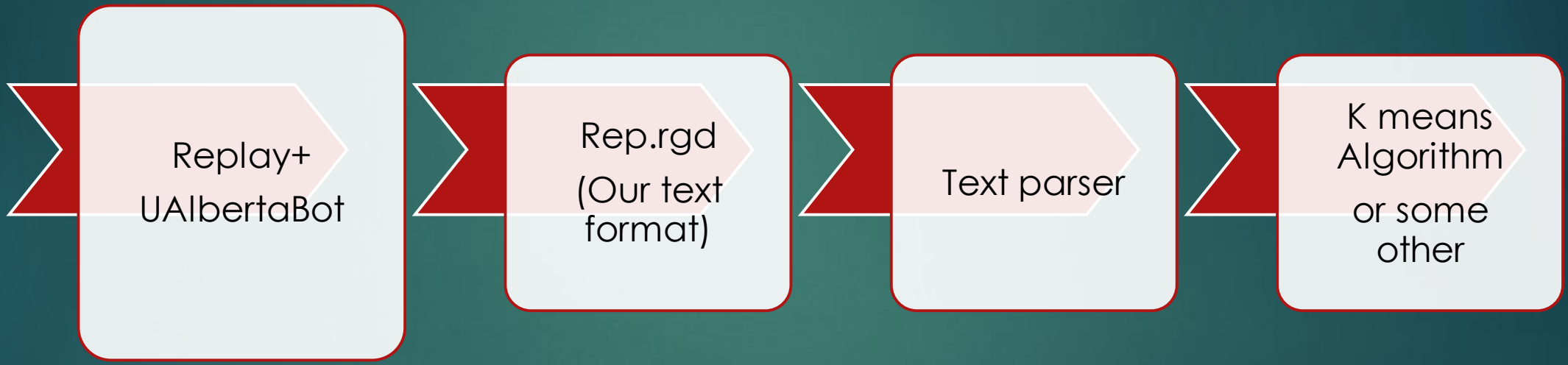

```
[Replay Start]
RepPath: MAPS\REPLAYS\GG26630.rep
MapName: | iCCup | Fighting Spirit 1.3
NumStartPositions: 4
The following players are in this replay:
0, TSL-Kr, Protoss, 2
1, TSL)Mondragon(, Zerg, 1
Begin replay data:
0 1 Created 2 Zerg Hatchery (288,3760)
0 0 Created 4 Protoss Probe (3784,328)
0 0 Created 7 Protoss Probe (3832,328)
0 1 Created 8 Zerg Larva (288,3802)
0 0 Created 10 Protoss Probe (3760,328)
```

```
36275 1 Morph 1059 Zerg Scourge (335,3809)
36276 1 Discovered 321 Zerg Overlord (2640,4023)
36276 1 Discovered 535 Zerg Lurker (2406,3737)
36288 1 Discovered 321 Zerg Overlord (2640,4023)
36288 1 Discovered 535 Zerg Lurker (2406,3737)
36290 1 Created 1085 Zerg Scourge (344,3768)
36290 1 Created 1086 Zerg Scourge (240,3717)
36290 1 Created 1087 Zerg Scourge (348,3773)
36300 1 Discovered 321 Zerg Overlord (2640,4023)
36300 1 Discovered 535 Zerg Lurker (2406,3737)
[ELAPSED TIME]: 2268
[UNITS LOST]
0 Protoss High Templar 1 [0-8000]
0 Protoss Probe 1 [0-8000]
0 Protoss Zealot 1 [0-8000]
1 Zerg Drone 3 [0-8000]
1 Zerg Zergling 2 [0-8000]
0 Protoss Dark Templar 3 [12000-18000]
0 Protoss High Templar 2 [12000-18000]
```

Why??

- Reading text is faster than real time parsing of the game state
  - So small changes to learning algorithms or clustering algorithms does not take days for re-computing data
    - It took me 2 days to go through 200 replays, and I still had 1200 left
- Machine learning requires a lot of data.
- Technical, Broodwar API unloads injected .dlls after a match.
- To Obtain feature vectors.

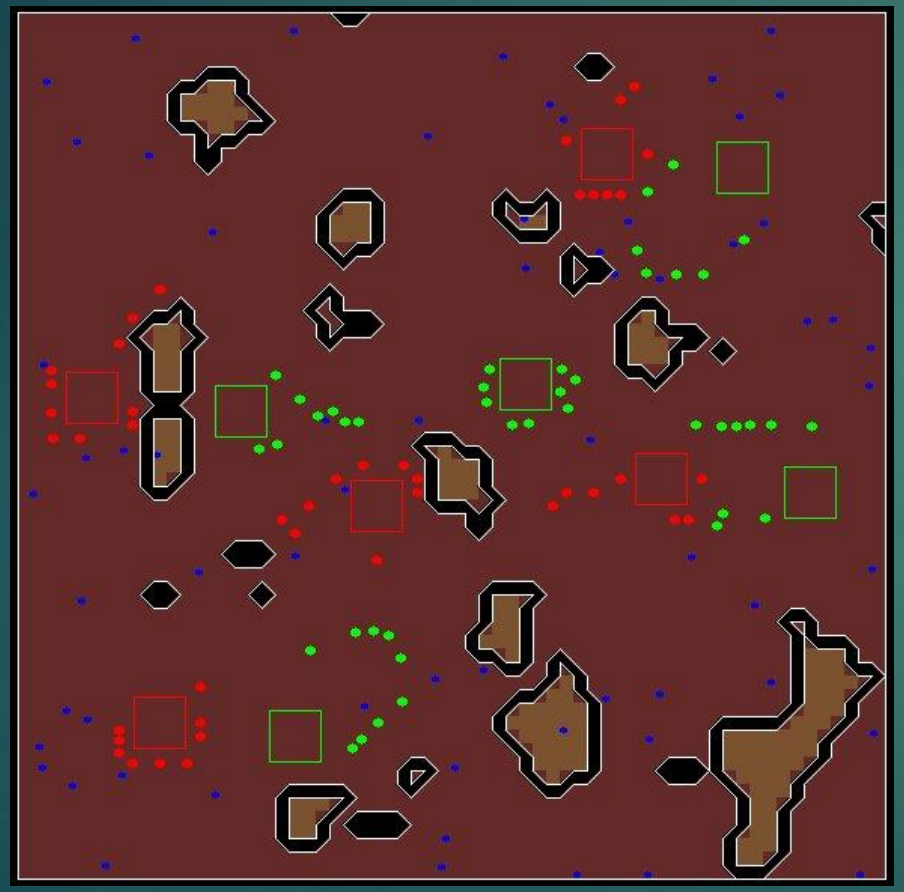
Overview of our process



Huge amounts of data in text format

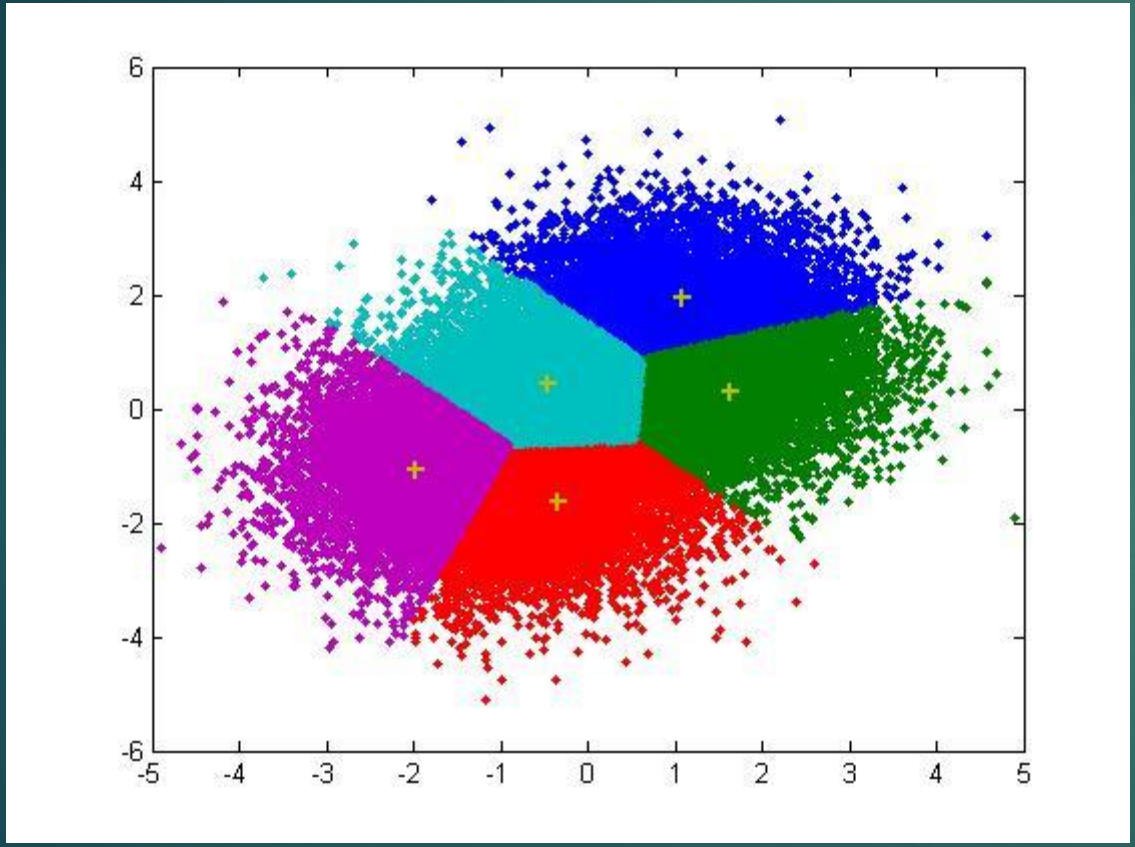
- What could be analyzed.
  - Where and when combat is taking place
  - The types of conflict. E.g. hit and run tactics (lots of detection) mass encounters (lots of destroyed units)
  - Where do players position their units.
  - Timings of battles
  - Creating a library of opening moves

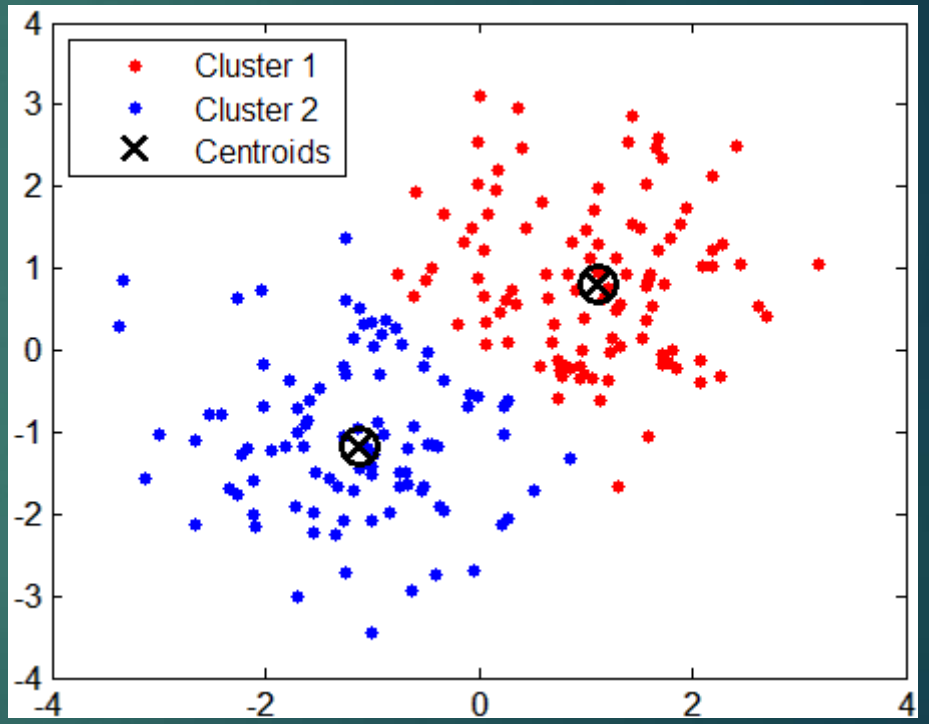
K means is flexible



Feed it feature vectors and it will produce centroids and clusters or any RTS game like ANTS and ORTS

Clustering algorithm





How K-means works

- First, select k arbitrary points to act as initial clusters.
- Assign all points to the cluster they are closest to.
- Find the centroid of these clusters.
- Repeat assignment of all strategies to these new clusters.
- Repeat until convergence condition is satisfied.

Kmeans can work flexibly with any sort of data, as long as the distance function is valid. We used a dot product-like approximation of similarity.

- Treat the strategies like vectors, and evaluate their projection onto each other.

How can our work be improved upon?

- A better similarity function or encoding function.
- More data!
- Optimal K is unknown

- Learning based off our returns is undone