Cluster analysis of deep embeddings in real-time strategy games

Haley, Joshua, Wearne, Adam, Copland, Cameron, Ortiz, Eric, Bond, Amanda, et al.

Joshua Haley, Adam Wearne, Cameron Copland, Eric Ortiz, Amanda Bond, Mike van Lent, Rob Smith, "Cluster analysis of deep embeddings in real-time strategy games," Proc. SPIE 11413, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, 114131I (21 April 2020); doi: 10.1117/12.2558105

SPIE.

Event: SPIE Defense + Commercial Sensing, 2020, Online Only

Cluster Analysis of Deep Embeddings in Real-Time Strategy Games

Joshua Haley[a], Adam Wearne[a], Cameron Copland[a], Eric Ortiz[a], Amanda Bond[a], Mike van Lent[a] and Rob Smith[b]

[a]Soar Technology Inc,3600 Green Ct 600, Ann Arbor, MI, USA
[b]CCDC Ground Vehicle Support Center, Warren MI, USA

ABSTRACT

Given that many readily available datasets consist of large amounts of unlabeled data,[1] unsupervised learning methods are an important component of many data-driven applications. In many instances, ground-state truth labels may be unavailable or obtainable only at a costly expense. As a result, there is an acute need for the ability to understand and interpret unlabeled datasets as thoroughly as possible. In this article, we examine the effectiveness of learned deep embeddings via internal clustering metrics on a dataset comprised of unlabelled StarCraft 2 game replays. The results of this work indicate that the use of deep embeddings provides a promising basis for clustering and interpreting player behavior in complex game domains.

Keywords: Machine Learning, Unsupervised Clustering, Neural Embeddings

1. INTRODUCTION

Most data generated from video games and simulation environments are unlabeled sequences of actions. This is not a surprise given the additional human intervention often needed to parse and label individual samples of a given scenario or game instance. Certain scenarios may require multiple experts with substantial domain knowledge to produce accurate labels for data samples, while others may be made manageable task by leveraging the crowd-sourcing the work via services like Amazon's MTurk.

However, in many instances, this human-labeled approach is very difficult to do, such is the case in the analysis of complex human behavior. One such effort, Early Synthetic Prototyping (ESP), seeks to uncover conceptual operations and tactics by constructing a physics-based game environment to rapidly assess how technologies might be employed on the battlefield. ESP is presently led and funded by the Army Capabilities and Integration Center (ARCIC) and supported by U.S. Army Research, Combat Capabilities Development Command (CCDC) labs[2] . The first effort is a small unit first person shooter entitled Operation Overmatch.

Given that ESP and the Overmatch efforts are ongoing, early prototyping required a rich data source that still had the highly cognitive decision making aspects as Overmatch. A common arena in which this occurs is in the space of Real-Time Strategy (RTS) games. With the success of Google's AlphaGo, deep learning applications aimed at RTS environments have become one of the most rapidly growing areas of research interest due to the complexity of the game space. In this article, we examine the efficacy of deep embeddings applied to game play data via metrics for internal clustering. Several earlier articles have shown the successful usage of deep embeddings[3] using metrics for which ground-state truth labels are known for given data samples. In the present work, we seek to evaluate their efficacy when such labels are either unavailable or highly subjective. Classifying the specific strategy used by a given player in an RTS game is an endeavor for which there may not be a clear class label. Using game play logs gathered from professional StarCraft 2 players, we examine the ability of clustering via learning deep embeddings, and evaluate the efficacy of this system in terms of separating and identifying

Further author information: (Send correspondence to Joshua Haley)
Joshua Haley: E-mail: Joshua.haley@soartech.com, Telephone: 1 407 602 6118

Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, edited by Tien Pham, Latasha Solomon, Katie Rainey, Proc. of SPIE Vol. 11413, 114131I
 · doi: 10.1117/12.2558105

distinct player strategies.

A number of related articles provided inspiration for this work. In,[4] Weber and Mateas take a collection of Starcraft game play data, which are labeled according to the strategy used by the player. Here, a player's strategy refers to the path taken through the technology tree. The authors applied several classification algorithms using the Weka toolkit to predict the strategy label from their raw input vectors. Their results showed that over certain time horizons, the predictive ability of their classification techniques outperformed a rule-based classifier which followed the exact rules used to produce the original strategy labels. A second approach was demonstrated by Synnaeve and Bessiere[5] in which they applied Bayesian methods to predict player strategies. Their work explores a probabilistic model for calculating the distribution of possible opponent technology tree trajectories given partial observations of the opponent's behavior. Their results demonstrate the ability to make advanced predictions regarding certain classes of player actions.

1.1 Domain Description

StarCraft 2 is a popular RTS game in which players play as one of three alien races, and compete to gather resources, develop technology, build an army, and ultimately destroy their opponents. Each of the race's military units have their own unique strengths and weaknesses requiring the player to quickly and continually adapt to opponent responses and counter-offensives. The adaptive nature of the game play in StarCraft 2 means that players must make complex and rapid decisions at a wide variety of timescales, each of which yield different insights into the player's behavior. This is not unlike the decision making made within tactical domains of non-video game nature.

Figure 1. game play from StarCraft 2

On the smallest timescales, one could consider player actions that might include orders to individual units or groups of units that comprise low-level tactics of the player's game play strategy. A step above this would be to consider the creation of buildings and production of units. Finally, at the highest level, one could consider the path a player takes down their chosen race's technology tree. The technology details the dependence relationship between advanced end-game buildings and their rudimentary prerequisites. It is this level of description in which we are interested. Knowledge of a player's path down their faction's build tree provides a great deal of information with regards to the overall strategy chosen. By examining features at this level, it is then possible to distinguish players who favor advanced aerial units and research upgrades, versus players who might focus on ground units, or some combined strategy. While low-level tactical information is very useful in understanding fine-grained strategic information with respect to combat, it has the tendency to miss out on overall strategies as the game plays out over a longer time horizon.

Figure 2. StarCraft 2 Technology Tree

1.2 Overview of Dataset

The dataset was compiled by scraping one-versus-one game replays from professional gaming websites.[6] The data was then separated based on the races used in a given match, and then each game was split to isolate the actions of individual players. The breakdown of match types in our dataset is given in table 1.

Table 1. Breakdown of Dataset by Match Type

	Protoss	Terran	Zerg
Protoss	88304	83799	83890
Terran	-	83629	84536
Zerg	-	-	86268

To create initial feature vectors from this data, we follow the approach taken by Weber.[4]

$$f(x) = \begin{cases} t & \text{Time when action } x \text{ was first taken} \\ 0 & \text{Otherwise} \end{cases}$$

In this context, the action, x refers to potential actions a player may take. These actions include producing a unit, constructing a building, and researching unit upgrades or new technologies. Following this approach, the raw feature vectors typically have 40 to 60 features depending on the race under consideration. The intuition behind this representation of game play is to attempt to trace the player's technological progress down a given path of their chosen race's technology tree. For instance, players using strategies that emphasize aerial combat would be expected to have lower values for those features that correspond to producing and upgrading air units.

2. METHODOLOGY

Given the representation of the raw input data described above, the next goals of the created embedded representation of this data are two-fold. To reduce the number of input variables that will be handed-off to our down-stream clustering task, and to learn a more meaningful representation of the raw features which captures

nonlinear relationships between them. This is accomplished using a deep autoencoder. Autoencoders have a symmetric network architecture which is comprised of an encoding and decoding module. In the first half of the network, the input is transformed through a series of layers into a compressed representation that captures the most prominent characteristics of the data. The latter half of the network then attempts to "undo" this transformation and reconstruct the original input from this compressed representation. The intuition here is that by training the network in this way, all the essential features of the data are then contained in this latent representation, reducing the dimensionality of the problem, and providing a more nuanced embedding of the original raw feature vectors, which can improve the performance of down-stream machine learning tasks. In one sense, the Autoencoder is approximating a Principal component Analysis in the latent dimension while side stepping the "Curse of Dimensionality."

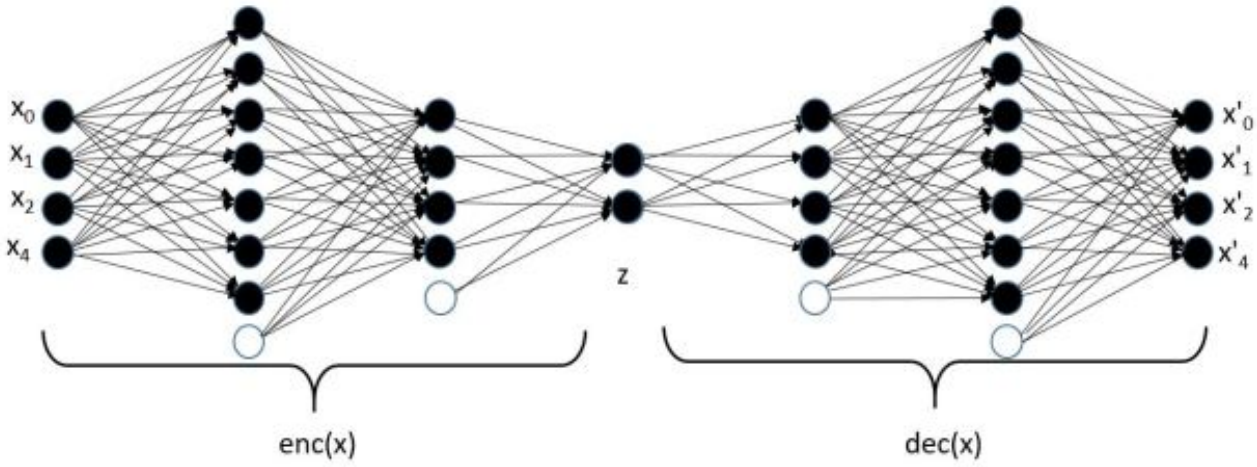

Figure 3. Autoencoder Architecture

2.1 CLUSTERING

Upon learning the embedding representation of the original data, three clustering methods were used. To start, as a benchmark we employed random clustering. That is, given the set of latent representations of game replays, we randomly assigned each instance to one of k groups. While simple and naive, this method provides a useful benchmark to which we can compare other methods. The second method of choice was the simple k-means clustering algorithm. Here we have elected to use the same value for k as in the random clustering case. As a final method, we employed a more modern method, Hierarchal Density Based Spatial Clustering of Applications with Noise (HDBSCAN).[7] HDBSCAN is an extension of the widely popular Density Based Spatial Clustering of Applications with Noise (DBSCAN) method; however, this extension has the added benefit of accounting for changes in density of data points. Additionally, compared to methods like k-means, which is perhaps better described as a partitioning algorithm, the number of clusters need not be specified apriori. Details regarding how HDBSCAN works can be found in the original paper by Campello, Moulavi, and Sander[7] but the basic operating principles are as follows. To start, one estimates the mutual reachability distance between points. For two points x_i and x_j this is defined as

$$d_k(x_i, x_j) = \max(\text{core}_k(x_i), \text{core}_k(x_j), d(x_i, x_j))$$

Where $d(\cdot, \cdot)$ is taken to be the usual distance metric. The function $\text{core}_k(\cdot)$ defines the minimum radius for which k neighbors are in the vicinity of a given point. Defining core distances in this way allows for changes in local density of points in the data set under study. As a whole, the mutual reachability distances can be viewed as a weighted undirected graph. Given this, the next step is to construct the minimum spanning tree between all points in this graph. A dendrogram is then constructed from the minimum spanning tree, and then further aggregated into a smaller hierarchal structure with more data points per node, using the minimum cluster size

as a threshold. Searching this condensed dendrogram, one can then identify the most persistent clusters based on when and where bifurcations occur. The result is then a hierarchal based clustering method which is robust to noise, and accounts for changes in local density.

2.2 Internal Clustering Analysis

To gain an understanding of the efficacy of our system's results, we examined the resulting clusters using both qualitative and quantitative means. First, as a qualitative measure, two-dimensional t-SNE plots were produced to gain some intuition for how the data is distributed in the higher dimensional space, and how well the clustering performed on partitioning this data. Rather than using more traditional methods of dimensionality reduction like PCA or SVD, which are based on projecting points in the dataset onto a set of eigenvectors, t-SNE is a probabilistic method which uses the distance between nearby points as a proxy for the probability that two points are similar. Details can be found in van der Maaten's original paper,[8] but the basic idea is as follows.

Consider a single point in the high-dimensional space, x_i. Centering a Gaussian at that point in the space with variance σ_i^2, one can then use the distance of other points, x_j, to compute the probability that it be considered a neighbor of x_i. That is,

$$p_{j|i} = \frac{\exp{(x_j - x_i)^2/(2\sigma_i^2)}}{\sum_{i \neq k} \exp{(x_k - x_i)^2/(2\sigma_i^2)}}$$

The goal is to map the distribution from this high-dimensional space to one in the latent space. Empirical studies have shown that using fat-tailed distributions tends to improve the performance and prevent points in the latent space from crowding near the origin. Thus, the distribution for points in the latent space is taken to be a t-distribution. Constructing a similar probability measure then in the latent space we have,

$$q_{j|i} = \frac{(1 + (z_i - z_j)^2)^{-1}}{\sum_{k \neq l}(1 + (z_k - z_l)^2)^{-1}}$$

With the form of these two distributions now in hand, one can then perform standard gradient descent on the corresponding KL-divergence to learn the latent space representations, z_i.

$$KL(P||Q) = \sum_{i,j} p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$

Notice that given the form of the KL-divergence and the fact that we are using it as a loss means that t-SNE is best suited for preserving local structure when projecting from the high dimensional space to the latent space, i.e., data points that are close to one another in the high-dimensional space should remain close in the latent space. Visualizing the learned projection provides scientists and researchers a way to then qualitatively understand the structure of their data, and discover if there are "natural" clusters of points that occur.

While t-SNE provides a nice means to gain intuition about the data under study, one also requires some quantitative metrics to better understand the efficacy of clustering. To address this, the silhouette score was employed to gauge how well separated our identified clusters were. The calculation was done as follows. For a given point in our dataset, x_i, we first computed the average distance x_i is to all other members of its assigned cluster. Let this quantity be denoted as a_i. Likewise, we then calculate the average distance between x_i, and the elements of all other clusters. The lowest average distance of x_i to a neighboring cluster was then denoted by b_i. With these two values, the silhouette score for the point x_i is then given by

$$s(x_i) = \frac{b_i - a_i}{\max(a_i, b_i)}$$

Average Silhouette Score	Interpretation
0.76 - 1.0	Indication of Strong Structure
0.51 - 0.75	Indication of Moderate Structure
0.26 - 0.50	Indication of Weak Structure
<0.25	No Indication of Structure

Table 2. Interpretation of Average Silhouette Score

In effect, what this quantity attempts to measure is how close a given datum fits within its assigned cluster, compared to how close it is to the nearest neighboring cluster. This provides a score for how well separated the clusters tend to be. One nice feature of the silhouette score compared to other internal clustering evaluation metrics is that $s(x_i)$ lies in the interval $[-1, 1]$. In many applications, it is then standard to report this quantity as an average across all data points. On the topic of interpretation of the silhouette score, and what constitutes a "good" score, there are no hard rules, but a common set of guidelines are listed in table 2.[?]

3. RESULTS

Here we examine the results of our clustering procedure, using the Protoss versus Protoss dataset as an example. Figures 4 and 5 display the t-SNE visualizations of our results using various clustering methods. Searching over a range of values for k, we found that the highest average silhouette score was obtained for $k = 2$. While k-means outperforms HDBSCAN in this sense, this disagrees with what one might expect for the number of player strategies in this domain.

Beyond this we can also look at the distribution across clusters to view what actions were favored by players in different clusters. Alternatively, we can view a single cluster to get a more local view of what actions were chosen. We can do this by first removing points classified as noise, and then viewing the corresponding box-plot across different cross-sections of our data. To properly interpret these plots, one must also keep in mind the method by which the raw feature vectors were constructed. By their very nature, many individual features may be zero, which happens when a player never undertakes a given action.

As an example, Figure 6 demonstrates such a cross-section across all clusters for the production of the Fleet Beacon. The Fleet Beacon is an advanced structure in the Protoss technology tree which enables a player to produce high-level aerial units and advanced research upgrades. Examination of Figure 6 illustrates that the creation of this structure was an important feature for cluster 2.

Going further with this example, we can then examine the distribution across all player actions for cluster 2 to gain further insight into the player strategies captured by this cluster. Examination of Figure 7 reveals that the main features that typify this cluster would be an early focus on basic structure as one might expect (Pylon, Gate, Assimilator), but also a focus on aerial combat, which can be seen from their focus on the production of Stargates, Fleet Beacons, and Carriers. What members of this cluster tend *not* to build is also telling. For instance, in no case did a member of this cluster research weapons upgrades for ground units.

4. CONCLUSION

In this article, we discussed and demonstrated the feasibility for clustering macro-level strategies and behaviors in StarCraft 2 using learned feature vectors from a deep autoencoder. Of the clustering methods used, k-means with $k = 2$ performs best in terms of average silhouette score; however, the results of HDBSCAN look qualitatively more impressive when visualized with t-SNE. Given the conflict between the quantitative results shown with k-means and the qualititative separations of HDBSCAN with t-SNE, it is important to pair the clustering with robust visualizations to allow for analyst decision support. Automated machine learning in this domain acts as a force multiplier for analyst evaluation by reducing the number of instances that must be viewed, but human understanding and intuition is still required.

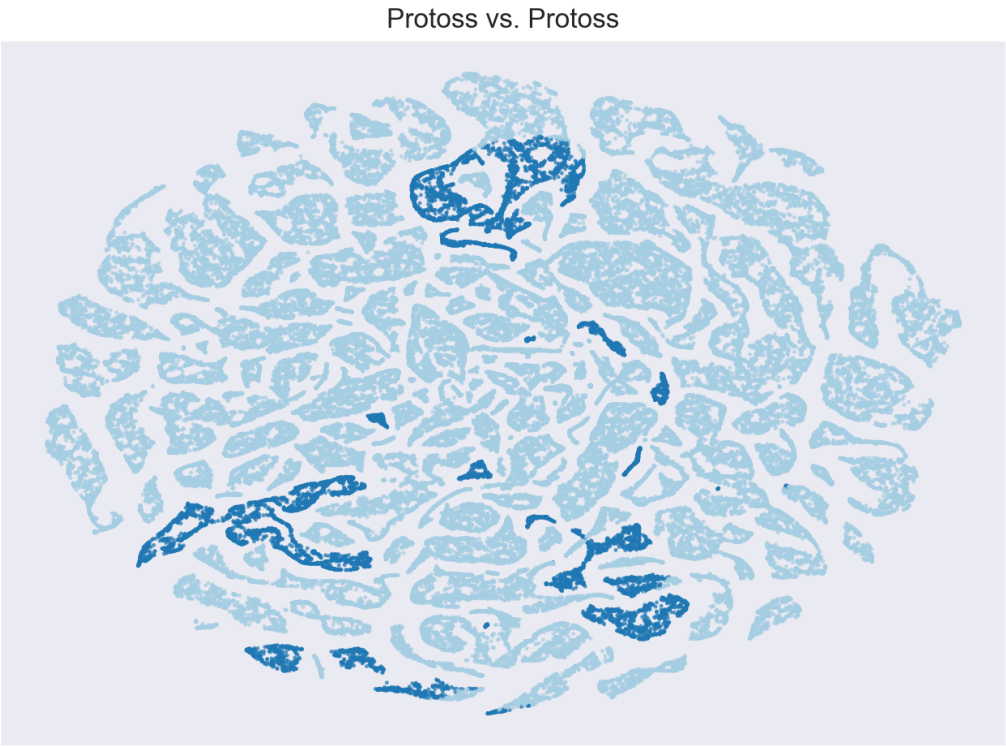

Figure 4. Method: k-means Number of clusters: 2 Silhouette Score: 0.705

REFERENCES

[1] Torralba, A., Fergus, R., and Freeman, W. T., "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence* **30**(11), 1958–1970 (2008).

[2] Murray, K. L., "Early synthetic prototyping: Exploring designs and concepts within games," tech. rep., NAVAL POSTGRADUATE SCHOOL MONTEREY CA (2014).

[3] Xie, J., Girshick, R., and Farhadi, A., "Unsupervised deep embedding for clustering analysis," in [*International conference on machine learning*], 478–487 (2016).

[4] Weber, B. G. and Mateas, M., "A data mining approach to strategy prediction," in [*2009 IEEE Symposium on Computational Intelligence and Games*], 140–147, IEEE (2009).

[5] Synnaeve, G. and Bessiere, P., "A bayesian model for plan recognition in rts games applied to starcraft," in [*Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*], (2011).

[6] "Ggtracker."

[7] Campello, R. J., Moulavi, D., and Sander, J., "Density-based clustering based on hierarchical density estimates," in [*Pacific-Asia conference on knowledge discovery and data mining*], 160–172, Springer (2013).

[8] Maaten, L. v. d. and Hinton, G., "Visualizing data using t-sne," *Journal of machine learning research* **9**(Nov), 2579–2605 (2008).

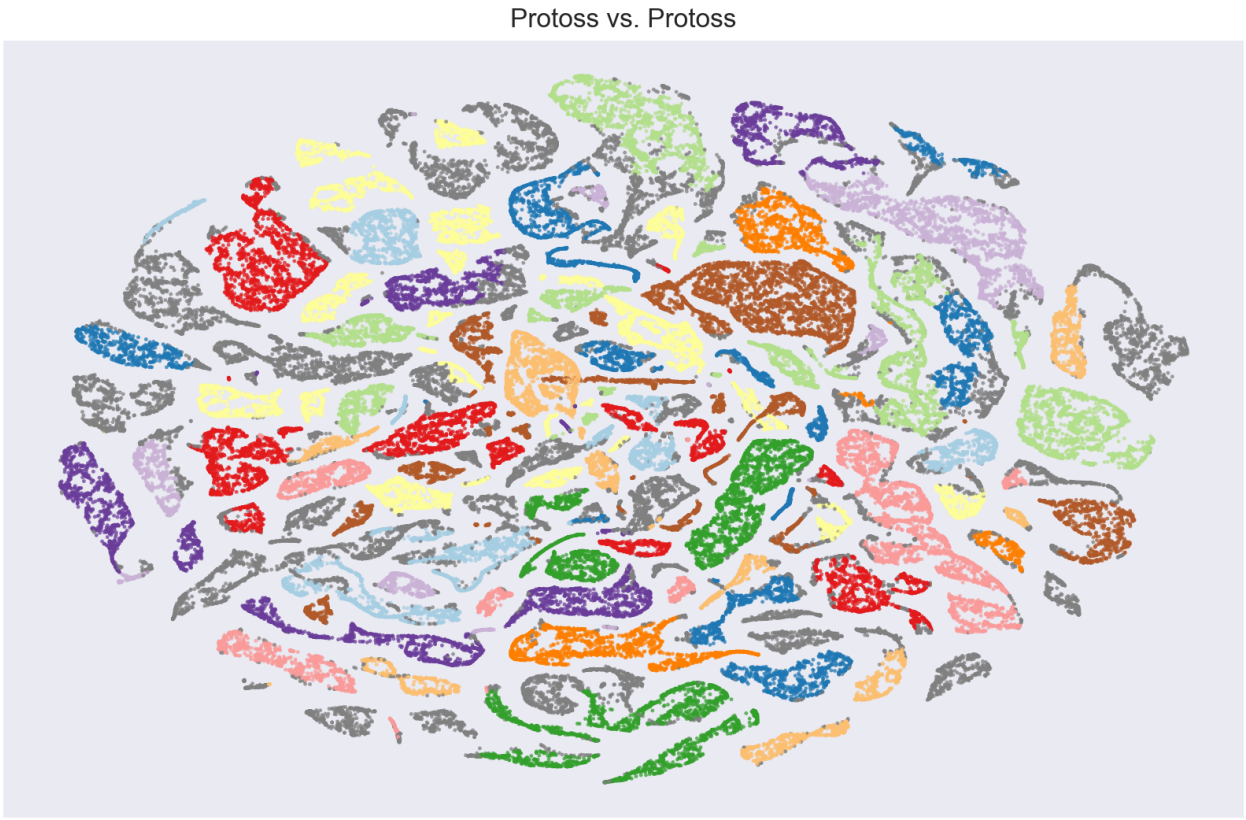

Figure 5. Method: HDBSCAN Number of clusters: 28 Silhouette Score: 0.442

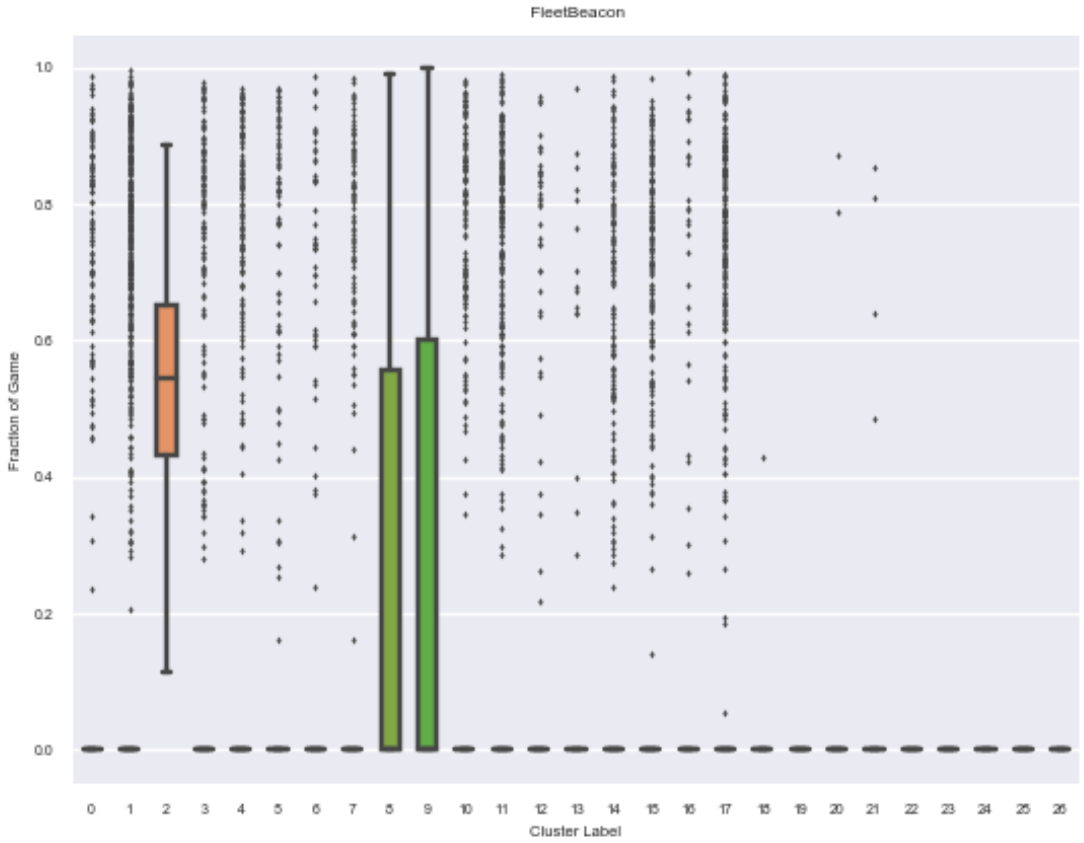

Figure 6. Cross-section across all clusters for Fleet Beacon production

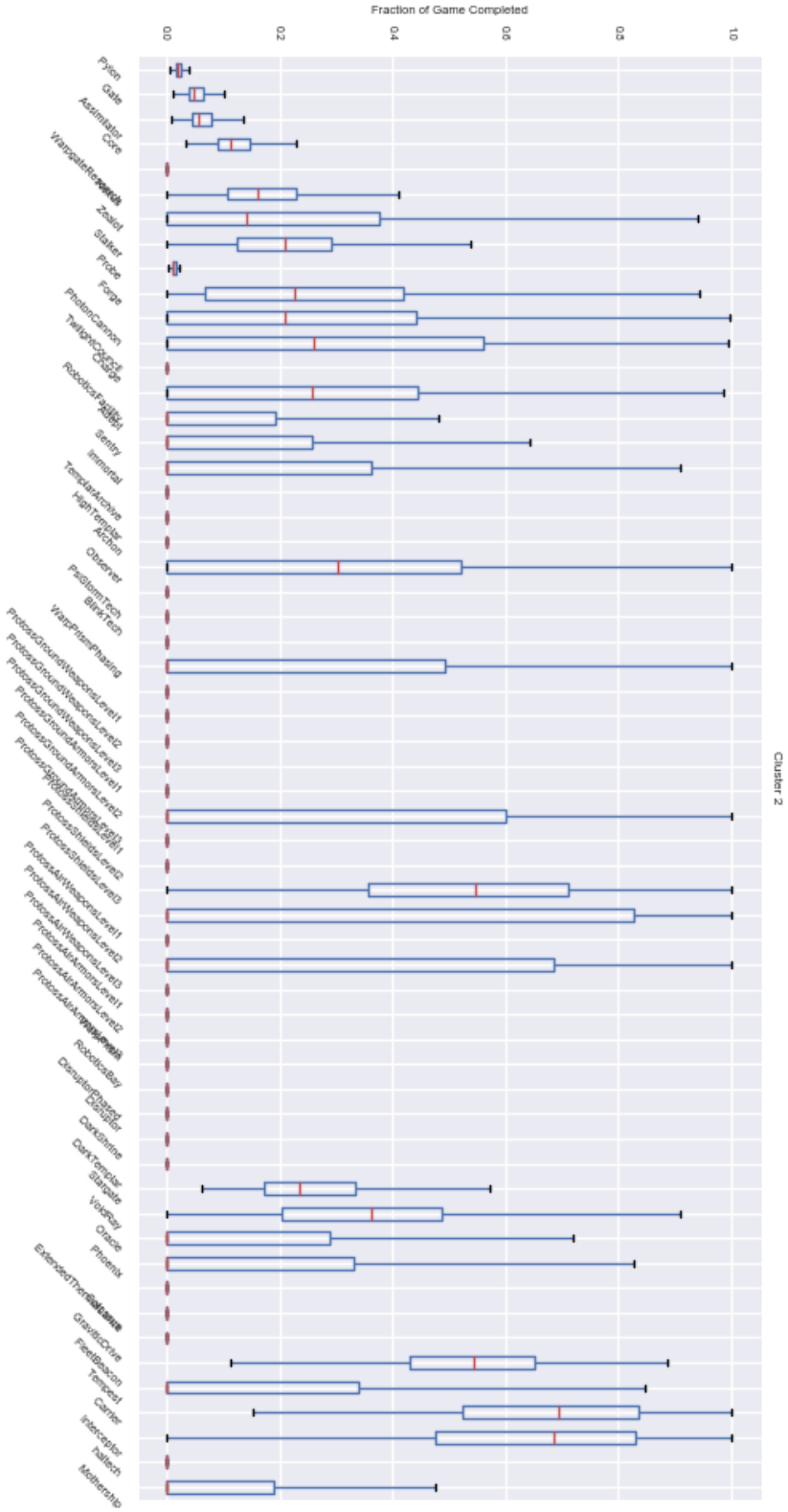

Figure 7. Cross-section across all player actions for a single cluster