To Appear In The 2nd Knowledge Extraction from Games Workshop 2019. AAAI.

# Discretization of Game Space by Environment Attributes

**Alexander Braylan and Risto Miikkulainen**
The University of Texas at Austin

## Abstract

Game AI is difficult to program, especially as games are frequently changing due to updates from the designers and the evolving behavior of human players. It would be useful if AI agents were able to automatically learn to reason about their environment. A major part of the environment is geospatial information. An agent's geospatial coordinates can suggest likelihoods of encountering important objects such as items or enemies, even when those objects are not in sight. Difficulties arise when these probabilities are not nicely demarcated into areas predefined and provided by the game API, creating the need to learn geospatial models automatically. This paper argues for models that divide game environments into discrete areas, proposes appropriate evaluation measures for such models, and tests a few clustering approaches on detailed creature sighting data extracted from a large number of players of a modern multi-player first-person shooter game. Two methods are shown to work better than simple baselines, demonstrating how these techniques can be used to automatically divide the game environment by its observed attributes.

## Introduction

Programming intelligent agents by hand is increasingly difficult as games become more complex and evolve faster with the changing needs and behavior of human players. Therefore, a major goal for programming game AI is to move toward agents that learn from data on their own. One powerful ability would be to automatically learn rich representations of the environmental dynamics.

Video game-playing agents that model their environment require high-level abstraction of their observations. Abstractions representing the nature of an agent's precise location should be particularly useful. For example, an agent might be able to infer what kind of items might be obtained in the vicinity after encountering a certain type of enemy. Or the agent might be able to assess the threat level of nearby enemies based on some of the fauna observed nearby. Using such spatial knowledge, the agent should be able to make accurate predictions as its location changes over time. Spatial knowledge therefore enhances gameplay by allowing the agent or player to strategize about where to hide, where to hunt, where to find quests or items, where to relax and enjoy the scenery, and so on.

Characteristics of the various locations in the game space might not be explicitly coded but emerge rather as a result of player interactions or the natural progression of the game mechanics. Previous work has explored ways to automatically represent such game characteristics that cannot be directly extracted from the game code. One example is extracting *interaction modes* representing subsets of reasonable actions for a given situation out of a much larger set of actions allowed by the code (Fulda et al. 2018). Along those lines, observed interactions between agents and objects can be organized into a graph structure used for high-level statistical reasoning and decision-making (Tomai 2018). Other examples include extracting hierarchies of concepts formed from observations of game objects and their attributes (Winder and desJardins 2018) and learning deep embeddings of game moments from screenshots (Zhan and Smith 2018).

Complementing prior work, the focus of this paper is to explore incorporating spatial location into abstract representations of the game. Representing space is a distinct challenge from representing objects, events, characters, and interactions in that the input is continuous rather than discrete, with nearby locations assumed to be more similar than far ones, and in that it is constantly observable and concurrent with other observations. Therefore, some of the existing methods for extracting and using abstract game concepts may not be adequate for spatial locations without first converting the spatial input into useful discrete representations.

There are several ways to characterize game locations. One way is to simply use the high-level information about the current section of the game map predefined and provided by the game API. However, such predefined areas are themselves often divided into several more specific areas of varying attributes whose borders are not provided by the API, possibly because they arise from dynamics that are not explicitly programmed, or possibly because they are affected by human players. Furthermore, it may be desirable for the agent to know only as much as could be observed by a human player rather than "cheat" by querying the game state. An alternative approach then is to learn to reason about possible observations from the precise $(x, y)$ coordinates of the agent's location. These coordinates can be input into a *geospatial model* that outputs useful information such as the probabilities of encountering various types of enemies, non-player characters, items, etc.

In specifying a geospatial model, an important decision is whether to use a kernel approach or a clustering approach. A kernel-based model outputs attributes that vary smoothly over fine changes in location, often through the use of Gaussian process regression (Williams 1998), although other machine learning algorithms could also serve this purpose. This type of model is used in real-world applications such as geology and mining, which require precise estimates of resource availability over a range of locations, similarly to how a game agent might wish to predict what objects might appear at a location. Such kernel-based models are often *non-parametric*, in the sense that the past observations themselves parametrize the model, as opposed to a parametric model whose number of parameters are fixed despite making additional observations. The implication of using a non-parametric model for characterizing locations in a video game is that the model learns not only by changing its parameters but by growing its parameters, which can be prohibitively expensive.

An alternative geospatial modeling approach is to use clustering algorithms to divide groups of neighboring location coordinates into discrete categories, or *biomes* (Udvardy 1975). A cluster-based model treats output attributes as constant within each biome but subject to sudden change upon crossing into a different biome. As a result, its predictions are less precise than those of a kernel-based model, especially near the edges of biomes. However, its number of parameters are fixed with the number of clusters, which may only grow as the agent explores far outside the spatial bounds of its past observations. Furthermore, common approximate clustering algorithms are often in practice more computationally efficient than the Gaussian processes, especially when observations are high-dimensional.

In contrast to many geological applications where new observations are relatively infrequent and plenty of compute time and memory are available before making a decision, a video game agent is constantly moving around the environment making new observations and having to act quickly based not only on geospatial knowledge but many other relevant variables. Cluster-based models can be preferable to kernel-based models due simply to being the more economical choice. Additionally, it could be argued that they are more interpretable as well. Consider the following explanations made by the candidate models:

**Kernel-based:** "I am at coordinates $(x, y)$, which by interpolation from observations at $(x-3, y-2), (x-1, y+1), (x+2, y+4), ...$, the probability of encountering a wolf each minute is 0.12"

**Cluster-based:** "I am at coordinates $(x, y)$, which are in biome type 7, where the probability of encountering a wolf each minute is 0.12"

The knowledge required for the cluster-based explanation is more compact and transferable. A non-player character that needs to explain its goals and plans in terms of objects and relationships (Molineaux, Dannenhauer, and Aha 2018) could enhance its explanations by adding this latter representation of location to its terminology.

Given a preference for cluster-based models that demarcate the game space into discrete biomes of varying characteristics, the remaining questions are what methods to use for learning the model parameters from observations and how to evaluate and compare methods. The remainder of this paper proposes evaluation criteria and methods that are then experimentally shown to outperform baselines. The experiments are conducted on data from an anonymous multiplayer first-person shooter game taking place over a large world map inhabited by hundreds of species of non-player *creature* characters.

## Approach

The following kind of data is assumed to be collected by the agent: A set of $n$ observation vectors $\mathbf{V} = V_1, V_2, ..., V_n$, where each $V_i$ is of size $j+2$: $j$ for environment attributes of interest and the other two for $x$ and $y$ coordinates. Each observation $V_i$ is therefore composed of $x_i, y_i, z_i^1, z_i^2, ...z_i^j$. Other notations include $X = x_1, x_2, ..., x_n$, likewise for $Y$, and the matrix $\mathbf{Z}$ with rows as observations and columns as attributes. The goal is for the algorithm to assign labels $B = b_1, b_2, ..., b_n$ which represent the biomes each observation belongs to, thereby also yielding a lookup function $C(b)$ returning the set of all observations pertaining to biome $b$. The possible values for $b$ are integers up to $K$, the total number of biomes.

### Evaluation measures

While developing candidate algorithms for discretizing the game space into biomes, progress was initially made by looking at the resulting maps and manually judging whether the biome divisions made sense. Qualitatively, the most attractive maps were ones where the biome boundaries were clear and distinct from each other in the compositions of their attribute populations. However, it is necessary to establish how to objectively evaluate a candidate result and why that evaluation metric is important. The following are the two main criteria that a successful discretization algorithm must meet.

**Usefulness for modeling** The algorithm should minimize variation in attributes of interest within biomes while maximizing said variation across biomes. Doing so improves the agent's predictive accuracy of its environment. For example, if an area populated with $z^1$ (e.g. wolves) and a neighboring area populated with $z^2$ (e.g. turtles) are treated as a single biome, each species would have comparable probability of occurring, whereas if the areas are treated as two distinct biomes, each species would have a probabilities closer to zero and one, leading to fewer surprises. For categorical attribute variables, information entropy in the attribute occurrence rates suffices instead of standard deviation. Low entropy implies the information is more useful in that it can be predicted more successfully. Biomes with low entropy are ones in which an agent is less likely to be surprised by an observation, so an algorithm which produces low-entropy biomes is more useful for agent reasoning. There may also be preferences for weighing some attributes more than others, but in these experiments all attributes are treated equally. The total entropy score $E$ for a biome discretization algorithm is calculated by grouping observed attributes by biome and summing the entropy $H$ of the frequency of each attribute in each biome, or more formally as follows:

$$E = \sum_{b=1}^{K} \sum_{t=1}^{j} H(P(t,b))$$

$$P(t,b) = \frac{\sum_{i \in C(b)} z_i^t}{|C(b)|}$$

$$H(p) = p\log(p) + (1-p)\log(1-p)$$

**Representation efficiency** As the number of attributes $j$ could be as large as the number of enemy types plus item types and so on, one important objective is to be able to handle such high-dimensional predictions with a relatively compact abstraction. To evaluate compactness, one measure is the number of clusters used. Eventually the approximate boundaries of the clusters also need to be stored efficiently. Therefore, another important objective is to maximize $S(X,Y,B,m)$, the average fraction of $m$ nearest neighbors in geographical space belonging to same biome, in other words a *non-overlapping condition*.

## Baselines

Two simple approaches tested do not achieve a satisfactory level of the above objectives. They make use of the *K-means* clustering algorithm (Hartigan and Wong 1979) implemented in scikit-learn (Pedregosa et al. 2011).

**K-Means on X, Y** One baseline is to run a K-means algorithm on only $X$ and $Y$ - effectively just a partitioning of the geographical space without regard to the composition of the clusters in terms of environment attributes, as seen in Figure 1. This method should not be expected to minimize and maximize attribute variation within and across clusters, respectively, therefore being less useful for predictive modeling.

**K-Means on Z** Another baseline method is to run K-means on **Z**, perhaps after normalizing or choosing weights to assign each column. However, this method should only be suitable to very specific cases where geospatial distributions of observations are already very homogeneous. For example, if an area only contains wolves, then a single cluster suffices to cover this area. However, if the area contains both wolves and turtles, K-means on Z can assign each species to different clusters which overlap in space as seen in Figure 2, violating the non-overlapping condition. This example illustrates the purpose of the non-overlapping condition: it is more efficient to store a definition for each biome containing several species than to store coverage definitions for each species inhabiting the game.

## Geospatially-constrained clustering (GCC)

The first solution attempts to meet the proposed objectives by clustering **Z** while imposing a constraint based on $X$ and $Y$. Effectively, this approach works by clustering over environment attributes like the second baseline above, but only allowing any observation to belong to a cluster if that cluster contains another observation that is a nearest neighbor in

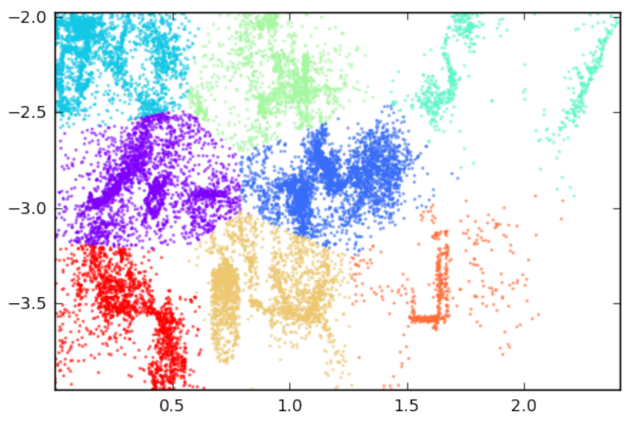

Figure 1: K-means on X,Y. Biomes determined only by location, allocating attribute information sub-optimally.

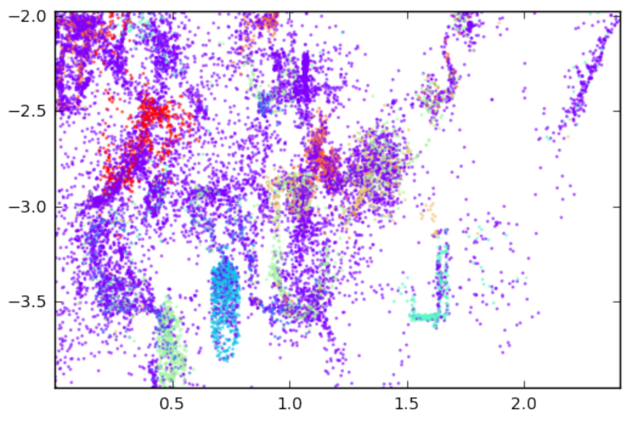

Figure 2: K-means on Z. Biomes determined only by attributes, preventing demarcation into contiguous areas.

geographic space. A map resulting from GCC can be seen in Figure 3. The implementation of constrained clustering used in this paper is the scikit-learn library's *agglomerative clustering*, a form of hierarchical clustering (Ward Jr 1963) where each observation is initialized in its own cluster before a merging process groups them into gradually larger clusters. The constraint is set to merging observations within $q = 3$ nearest neighbors. Setting it much larger than this will result in violations of the non-overlap constraint.

## Geospatially-aggregated clustering (GAC)

A second possible solution is to first convert **Z** into a representation that is aggregated in the geospatial neighborhood. For each observation $V_i$, any other observations within a specified radius $r$ are collected, and their attributes are averaged into vector $\hat{Z}_i = \hat{z}_i^1, \hat{z}_i^2, ...\hat{z}_i^j$. K-means is then run on $\hat{\mathbf{Z}}$ rather than **Z** as in the second baseline. The initial aggregation step changes the attribute representation at each observation by including nearby observations. The second baseline's problem of finding separate overlapping biomes for wolves and turtles occupying the same area is therefore overcome by grouping wolves and turtles together in the attribute space. Figure 4 depicts a map resulting from GAC.

Compared to the GCC method, GAC allows a biome to encompass distant patches in the map because it does not require a geographical constraint during the clustering phase. For this reason, it may ultimately require a more complex representation of the biome boundaries, though this paper does not investigate further into that issue.

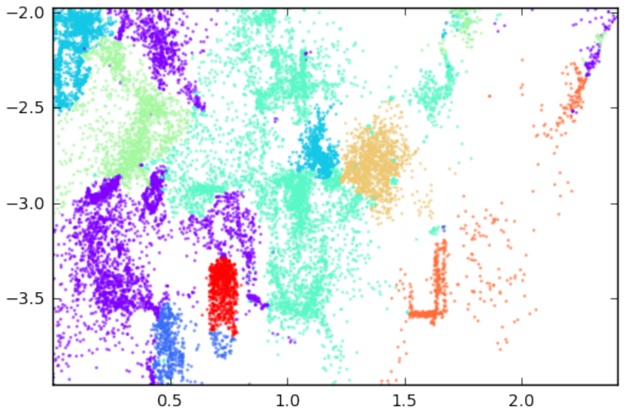

Figure 3: Geographically-constrained clustering. Clusters form out of neighboring observations, so biomes are rarely separated into different areas.

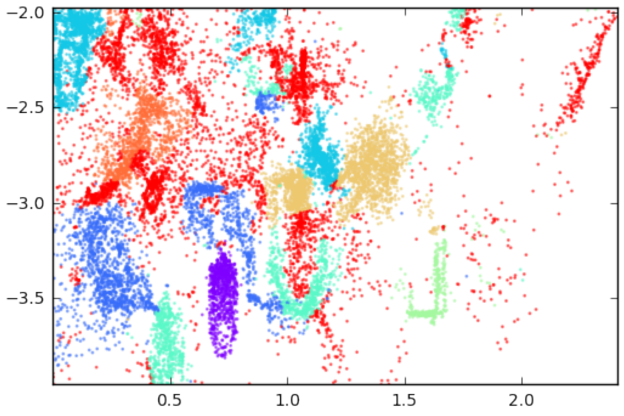

Figure 4: Geographically-aggregated clustering. Clusters form from patches of observations that may be in distant areas.

## Experiments

The above four methods are tested on data from an anonymous modern multi-player first-person shooter game, describing the locations of observed non-player creatures. Experiments are conducted and reported on three maps comprising about a fifth of the whole game area each. The size of the attribute vectors in each map is the number of creatures inhabiting it, ranging between 200 and 400. The $n$ number of observations in each map ranges between 5000 to 18000.

The target number of clusters is set to eight in all experiments. The $q$ nearest neighbor constraint for GCC is set to three, which worked best on a separate development map. The $r$ aggregation radius parameter for GAC is similarly roughly optimized on the development map. The total information entropy measure $E$ is calculated for the evaluation of usefulness for modeling, while the $S$-measure is calculated for the evaluation of representation efficiency.

| Map | Method | S | E | Time(s) |
|---|---|---|---|---|
| 1 | K-means (X,Y) | .99 | 33 | 2.3 |
| 1 | K-means (Z) | .64 | 5 | 1.4 |
| 1 | GCC | .98 | 28 | 3.6 |
| 1 | GAC | .96 | 28 | 7.0 |
| 2 | K-means (X,Y) | .99 | 36 | 2.3 |
| 2 | K-means (Z) | .50 | 6 | 1.4 |
| 2 | GCC | .97 | 27 | 3.6 |
| 2 | GAC | .97 | 31 | 9.0 |
| 3 | K-means (X,Y) | .99 | 33 | 1.4 |
| 3 | K-means (Z) | .55 | 5 | 0.7 |
| 3 | GCC | .98 | 26 | 2.0 |
| 3 | GAC | .96 | 25 | 4.6 |

Table 1: Results of running the two baselines and two proposed methods to divide three maps into eight biomes each. K-means (X,Y) performs worst on entropy, while K-means (Z) gets an impermissibly low $S$-score, violating the non-overlap condition. GCC and GAC are about tied for entropy, but GCC is faster.

## Results

Table 1 lays out the results of the experiments. As expected, K-means on X,Y consistently performs worst on the entropy measure for usefulness, while K-means on Z fails the non-overlapping condition.

Of the two remaining proposed methods, the winner is not as clear. GCC outperforms GAC on one map, and they perform similarly on the other two. This result is slightly surprising as GAC should attain better entropy due to its increased flexibility over GCC. Further experiments are needed to understand this result better. Both GCC and GAC achieve high enough $S$-scores that the difference between them is unimportant. However, due to GAC's ability to divide each biome into distant areas, the representation efficiency should be a bit worse, although additional measures of representation efficiency are needed to further confirm that. GAC takes about twice as long to run as GCC. The extra slowness is due to the determination of neighbors within each observation's radius.

## Conclusion

Game AI is increasingly difficult to program by hand due to the complexity of the game environments, contributed to by the various types of enemies, friends, items, structures, and other objects. While some environmental knowledge might sometimes be provided by the game API, many of the useful dynamics need to be learned from observation.

Representing space in a lower-dimensional abstraction is an important ability for game agents that need their knowledge to be light-weight and explainable. Sectioning the geography into discrete biomes and knowing their boundaries and distributions of environment attributes is one way to accomplish this goal.

A desirable method for discretizing the game space maximizes the usefulness of the resulting indicators for the purpose of predictive modeling while also maintaining representation efficiency. This paper presents two methods that accomplish both goals and demonstrates their benefits against methods that only aim for one goal or the other.

These methods may be further refined by taking what works from both and figuring out how to overcome their weaknesses. Future work must also evaluate and optimize representation efficiency more rigorously, and test how well predictions generalize near borders and from fewer total observations. Another interesting direction is to increase to complexity of the spatial $(x, y)$, and attribute $(z)$ dimensions. For example, $x$ and $y$ could be augmented with a third spatial dimension for interstellar games or with a time dimension to capture dynamic or seasonal biomes. Attributes could consist not only of observed objects but of interactions between them, allowing for agents to understand how their location could affect complex game behaviors.

Ultimately these methods to discretize game environments can be used in a variety of applications besides just training game agents. For example, they can be used to create indicator variables for design optimization problems such as difficulty balancing and match-making. They can also be used as predictors for player modeling. Biome distributions and layouts learned from successful games may even give insights into map design and content generation.

## References

Fulda, N.; Ricks, D.; Murdoch, B.; and Wingate, D. 2018. Threat, explore, barter, puzzle: A semantically-informed algorithm for extracting interaction modes. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.

Hartigan, J. A., and Wong, M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1):100–108.

Molineaux, M.; Dannenhauer, D.; and Aha, D. W. 2018. Towards explainable npcs: A relational exploration learning agent. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Tomai, E. 2018. Extraction of interaction events for learning reasonable behavior in an open-world survival game. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.

Udvardy, M. D. 1975. *A classification of the biogeographical provinces of the world*, volume 8.

Ward Jr, J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58(301):236–244.

Williams, C. K. 1998. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In *Learning in graphical models*. Springer. 599–621.

Winder, J., and desJardins, M. 2018. Concept-aware feature extraction for knowledge transfer in reinforcement learning. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.

Zhan, Z., and Smith, A. M. 2018. Retrieving game states with moment vectors. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.