John Ortiz

Video Games K-Means Clustering

Videogames were created before I was born but I definitely grew up in one of the best times to experience the best of video game culture. Alongside the exponential growth of the videogame industry I was able to experience a lot of different games. My question of interest stemmed from my love of videogames. I have played video games almost my entire life (earliest I remember was 7), it started as competition with my brother to see which one of us would be the best at each videogame we play. It got to the point where I could beat him at every game except one, NBA-2k. Now in all my time gaming, not once have I ever asked which video games are most closely related to my top games? I think the question of interest can be applied generally to help newcomers to the gaming community. By presenting the list of the videogames clustered closely as it could present insight on other games they might like if they recognize any. For me it could help identify which games to try out that I have not seen. To answer this question, we need to find a data set and clustering algorithm to go along with it.

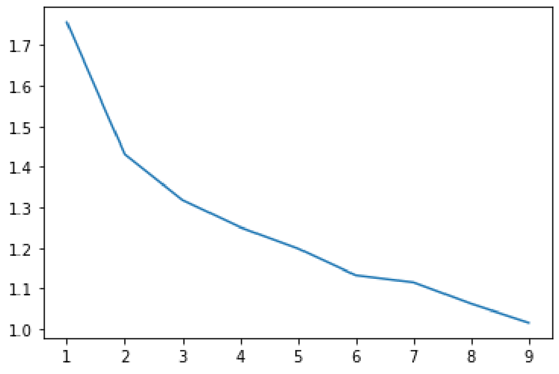
The source of the videogame data set comes from Kaggle (https://www.kaggle.com/ashaheedq/video-games-sales-2019?select=vgsales-12-4-2019.csv), but the owner used a scrape on vgChartz.com to get the data. In total there were 23 columns and 37102 values; later we will find out that for a couple reasons we have to reduce the dataset. The video game data set includes columns on rank(of overall sales), name, baseName(there are repeats based on platform),genre(action, shooting, role playing), ESRB Rating(ex. T is for teens), platform, publisher, developer, criticScore (vgChartz critics),userScore(rating 1-10),totalShipped(shipped copies),global Sales(total of following Sales), NA_Sales (north America sales), PAL_Sales(Europe sales),JP_Sales(Japan Sales), Other Sales(rest of world), year(when game got released),first VGchartz_Score (empty list- assuming it forgot to be deleted as there are two columns for it), lastUpdate (when all the columns have last been updated), url (location of game on VGchartz.com), Vgchartzscore(given by VGchartz),and last is an img_url(which only some of the links works). Couple points worth mentioning is all sales are in millions, there are scarce columns that will need to be dropped, and we will only be using the numerical data that has no missing values, so we can properly run k-Means.

The problem I am proposing I do not believe has been solved by real world companies, but there are some academic papers. The closest article I have found in terms of the problem is a article form ScienceDirect, Game Data Mining: Clustering and Visualization of Online Game Data in Cyber-Physical Worlds. The article analyzes Overwatch data on gameplayer statistics to create “high win-rate cluster”, by looking for teams that have a full team (tend to have a higher win percentage) and then comparing the clusters members characteristics/playstyle. If there are similarities in members, the characteristics/playstyle would be recommended to a new player as they would be able to reproduce the same tactics. The goal of the clusters where to help new players by giving suggestions on characters (in Overwatch) based on the characteristics from clusters with highest wining player strategy. This is similar to the problem I am trying to solve as the insight offered from our results are more in line with what other games an individual might want to consider trying vs specific playstyle. I do not think there is any specifics on videogame sale clustering as whole, but there is a lot of clustering revolving a specific game or library (like steam). There is an public published paper, called *Clustering steam user behavior data using K-Prototypes algorithm.* *K*-Prototypes combines K-means and K-modes (For categorical variables), which seems to be really cool idea to try an implement.

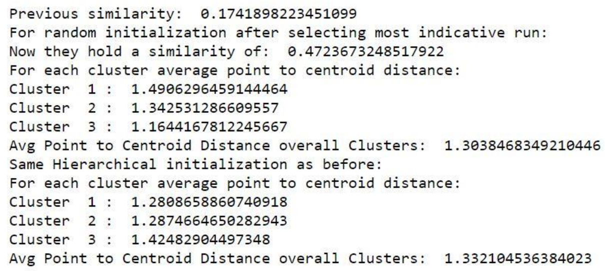
After loading our dataset and taking a look, I decided to drop the NaN values regarding all the numerical data. It turns out that in this dataset that we collected that the column *Total_Shipped* only contained values in which sales (Global, NA,PAL,JP, Other) did not. So to decide which column(s) would stay would be based on which ever data frame had more videogames as after reducing just by NaN values, the data frame was reduced to only 108 videogames if we use sales and 64 if we use Total_Shipped. This left an easy decision to drop this column so we can use more data points. It turns out when we dropped NaN values Vgchartzscore it reduced the list from 108 to 58. I thought the list size was already quite reduced from what it was to keep such a scarce column. So, I dropped it as well. The last thing I would mention is that there was a collinear relationship between every sales column except for Japan Sales. Due to this fact if we are split in to two data frames to compare at the end. The decision to focus on just these four columns 'Critic_Score', 'User_Score', 'Global_Sales', 'Year' comes from the fact the Global Sales are the total of all the sales, seems counterproductive to have both in here.

The data science technique we will be implementing is the K-means from class. It is intended to only work with numerical data. The goal of any clustering algorithm is to cluster into groups which is exactly what we need: as ours is to cluster together similar videogames. K-means require us to define a specific distance measure, we will use Euclidean Distance for ease of use and simplicity. Euclidean Distance is defined by $\mathrm{d}(\mathrm{x}, \mathrm{y}) = \sqrt{(\sum_i^n (x_i - y_i)^2}$. K-means requires two other things to run before running the algorithm. To fix a value K and to decide on an initialization. To find a value for K, it is recommended we use an elbow plow of average distance from point to there respective centroid. It will act as are “measure of goodness” of our K-means algorithm. As discussed in class initialization does matter, and as such what better way to see that than is by practicing it. There were the types of initializations given as options for K-means. The first is by random, pick k points randomly to be in each different cluster. This is currently the fastest to implement as it just calling a random function. The second option was hierarchical plan: in which we take a subset of the data and perform Hierarchical clustering to get K clusters, then pick randomly from each cluster. The third is picking subsequent furthest points from all previous points. We will perform random initialization and compare it the hierarchical plan for initialization. As I never was able to implement plan for the homework, I would like to see how it compares to the random initialization. To perform the hierarchical plan, we need to perform hierarchical clustering. Hierarchical clustering starts by assigning every point to there own cluster than combining the two closest to be one (centroid or point, depending). The next iteration is n-1 clusters and would continue to iterate until K clusters where reached. With are k and initialization set we just need to run it. The K-means algorithm works on a two-step iteration till convergence. Step 1 is for each point assign to the cluster with closest centroid. After assigning all subsequent points, for each cluster update centroids. Then repeat until points stop moving (compare cluster assignment) or centroids stop moving (compare with previous centroids).

To start as I said I needed to determine which is a good K value for the number of clusters. As such after implementing both versions of initialization, I decided to use hierarchical as it would produce closer results than a single run of random initialization. The elbow plot (**shown on the right**) does not look great as it seems to not have a definitive "elbow", however we also have to consider visually how many clusters it should be. As such 2 groups will not tell us enough so the other elbow, I would argue k is at 3 as when k equals 4, when visualizing groups, a very small amount into the 4th cluster. This follows the trend I observed that with higher k values the avg point to centroid distance did shrink; however, after certain amount of increases in k some of the clusters are sparse which does not seem advantageous in getting an accurate representation of the data.

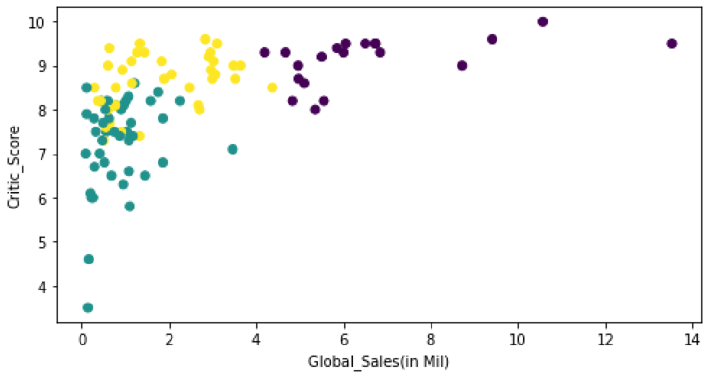


After knowing which K value to use I started by comparing the two different initializations. The random initialization was a bit worse in terms of cluster average point to centroid distance as the hierarchical initialization had a tighter grouping. After comparing the visuals of the cluster, we can see clearly that the hierarchical created more in line clusters. I used sklearns adjusted_rand_score to check similarity between cluster assignment for each point. The similarity score is 0.17418. As this proves the importance of initialization as random initialization could have the chance to pick really tuff starting cluster points. As such I used what we were taught from the homework and run the kMeans 100 iterations to form a similarity matrix using the adjusted_rand_score for the 100 iterations. From that we can take the most ideal run to now compare against the same hierarchical initialization for k-means as before. The similarity score between the two is now up to .47237.

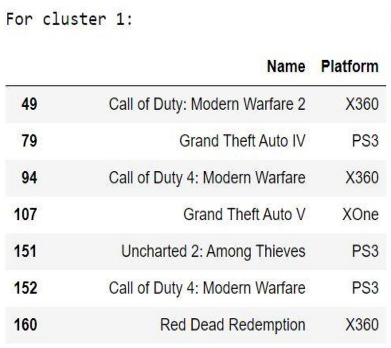


```
Previous similarity:  0.1741898223451099
For random initialization after selecting most indicative run:
Now they hold a similarity of:  0.4723673248517922
For each cluster average point to centroid distance:
Cluster  1 :  1.4906296459144464
Cluster  2 :  1.342531286609557
Cluster  3 :  1.1644167812245667
Avg Point to Centroid Distance overall Clusters:  1.3038468349210446
Same Hierarchical initialization as before:
For each cluster average point to centroid distance:
Cluster  1 :  1.2808658860740918
Cluster  2 :  1.2874664650282943
Cluster  3 :  1.42482904497348
Avg Point to Centroid Distance overall Clusters:  1.332104536384023
```

Comparing both forms of initialization with two distances: within each cluster the average point to centroid distance and the overall avg point to centroid distance. Hierarchical Initialization had lower distance measures for each cluster. However, I selected initialization with the lower avg point to centroid Distance overall, which in this case is **random initialization** with the most indicative run. As such we will use it as our final clusters and visualizations.

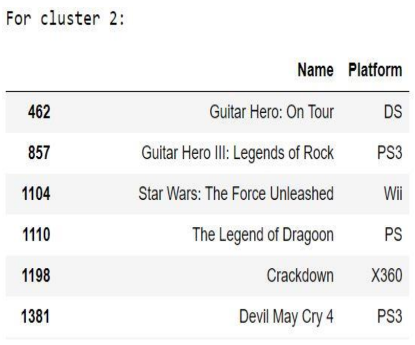


So now that we can see our clusters pretty clearly, it would be useful to actually see which videogames got classified into which cluster. Below I will display some videogames that belong to each Cluster.



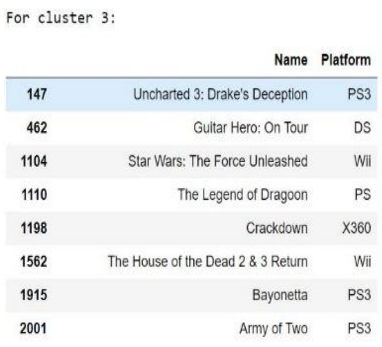
For cluster 1:

	Name	Platform
49	Call of Duty: Modern Warfare 2	X360
79	Grand Theft Auto IV	PS3
94	Call of Duty 4: Modern Warfare	X360
107	Grand Theft Auto V	XOne
151	Uncharted 2: Among Thieves	PS3
152	Call of Duty 4: Modern Warfare	PS3
160	Red Dead Redemption	X360



For cluster 2:

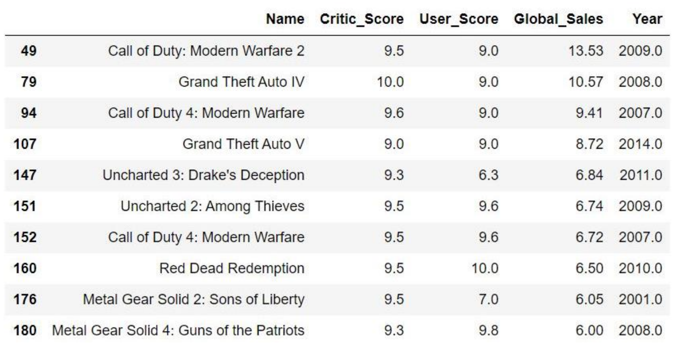
	Name	Platform
462	Guitar Hero: On Tour	DS
857	Guitar Hero III: Legends of Rock	PS3
1104	Star Wars: The Force Unleashed	Wii
1110	The Legend of Dragoon	PS
1198	Crackdown	X360
1381	Devil May Cry 4	PS3



For cluster 3:

	Name	Platform
147	Uncharted 3: Drake's Deception	PS3
462	Guitar Hero: On Tour	DS
1104	Star Wars: The Force Unleashed	Wii
1110	The Legend of Dragoon	PS
1198	Crackdown	X360
1562	The House of the Dead 2 & 3 Return	Wii
1915	Bayonetta	PS3
2001	Army of Two	PS3

Before we classify each cluster, I think we need a little sanity check that these games should be grouped together. Below show the first 10 values of cluster1, with most of the features displayed. I think a good check is looking at our numerical categories we can all the users and critic scores have extremely high scores (8-10), there is two relatively low user_scores which are for Uncharted3 and metal Gear Solid 2, which still made over 6.84 and 6.05 million in total sales. Now in terms of the entire cluster the lowest in global sales is 4.19 million. With all this information it looks as if the videogames have been split by quality of the game. By that I mean Cluster 1 seems to representative of top tier videogames. (I am not displaying the other two too not take up too much space with images. However, they are displayed at the end of my notebook.) Using the same logic cluster 2 holds videogames that seem to hold average ratings (5-8) for both critic and user scores. Another note is that the global sales does not reach over 3.46 million in sales for any videogame in the cluster. Cluster 2 seems to be representative of mid-tier (average) video games. Cluster 3 is representative of high user/critic scores (7-10) and low global sales (nothing higher than 4.37 million, average would be closer 2 million). Cluster 3 I would group as the videogames that you would buy that are on sale. They are not the best but are quality. It seems that year played the least influential role in the cluster categorization as I do not see a definitive pattern. This is sign that my K-mean might not be representing the data the best.



	Name	Critic_Score	User_Score	Global_Sales	Year
49	Call of Duty: Modern Warfare 2	9.5	9.0	13.53	2009.0
79	Grand Theft Auto IV	10.0	9.0	10.57	2008.0
94	Call of Duty 4: Modern Warfare	9.6	9.0	9.41	2007.0
107	Grand Theft Auto V	9.0	9.0	8.72	2014.0
147	Uncharted 3: Drake's Deception	9.3	6.3	6.84	2011.0
151	Uncharted 2: Among Thieves	9.5	9.6	6.74	2009.0
152	Call of Duty 4: Modern Warfare	9.5	9.6	6.72	2007.0
160	Red Dead Redemption	9.5	10.0	6.50	2010.0
176	Metal Gear Solid 2: Sons of Liberty	9.5	7.0	6.05	2001.0
180	Metal Gear Solid 4: Guns of the Patriots	9.3	9.8	6.00	2008.0

I think I found very interesting results as the first cluster displays some of my favorites MW, MW2, Red Dead Redemption, are all in the same boat, it really makes me want to go try out other games in each cluster that I had not gotten the chance to play yet. Cluster 2 hold the original assassin's creed (rip to no fast travel), and cluster three contains two of my old favorite games one is Star Wars the Force Unleashed (great character arc),other is Army of Two (coop campaign with my brother, the race for more eliminations). From each cluster if I could only choose one to explore that I have not played it would be Uncharted 2, Resident Evil 5, Guitar Hero: On tour for each subsequent cluster.

I think my K-mean implementation might be not be the best implementation, first off with exclusion of the different sales (NA, PAL, JP, Other) types to just using the global sales. I think this would have been a perfect time to implement PCA on it to reduce the number of dimensions. In terms of prebuilt libraries that give access to running algorithms like K-means, specifically sklearn: my 4022 K-means implementations might perform worse. Sklearns implementation depend on which K-means called (K-means++), could be optimized in terms of their code as our 4022 technique I think loops more than it needs to. In other words, it's great to know for learning purposes I am not sure well implementing each of these by hand is better. And in terms of work still to do, there is a lot of directions I could take the data set. After learning about k-prototypes it would a be very interesting doing a k means with categorical variables included. It would be really cool to see which videogames get clustered from that. There is another way I believe that could improve K-means is to use the Mahalanobis distance, as it focuses on distance from the "structure", which would capture the variance of the structure of each cluster. Both these are out of the scope for this project due to the timing of this, however, could be interesting to try out. Finding videogames with all the numerical data proved to be tuff from the data as there were a lot issues with missing values and made some easy choices of which columns to focus keep/drop. It really reduced the total amount of videogames I was going to run on K-Means. If the data set was more complete not just the reduced list of 108, I think the impact of the results would be bigger. If I were to do any data set on Videogame cluster based on sales, I would make sure I get a more filled in data set. The next move would be to utilize PCA to reduce the dimensions so I would not need to be concerned with choosing a certain dimension to focus on (like sales). Overall, even if my K-means implementation is not the best, it led to me learning about different implementations of the algorithm that are beyond the scope of the class and produced some new games I will have to go buy.