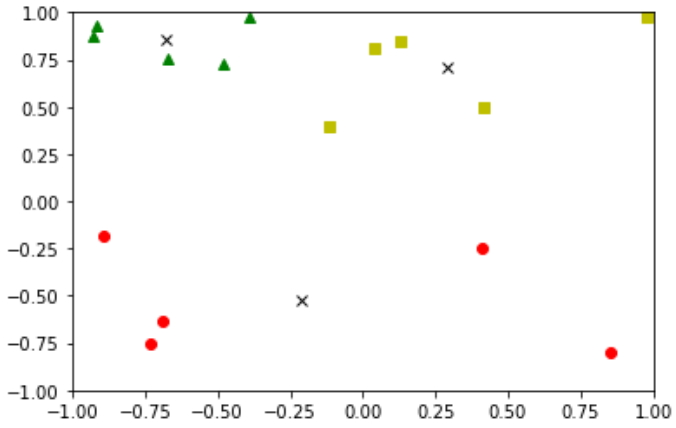
Practical 7

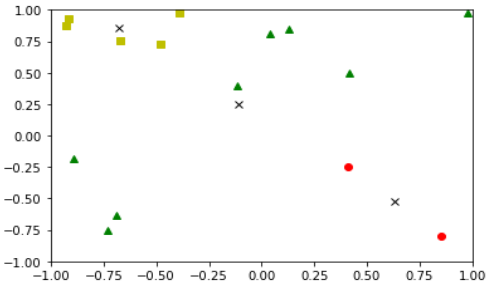
1.) The random points and graph created is in figure one. Already it is notable that the three Xs sort of fall in the middle (more or less) of the clusters anyway. The data points have been saved and graphs of all ten iterations are shown on the next page.

```
[[-0.68973752 -0.62950155]
 [ 0.12902673  0.84649287]
 [ 0.03962326  0.81191641]
 [ 0.8511069  -0.79890579]
 [ 0.98026066  0.97676941]
 [-0.11443423  0.39994209]
 [ 0.41684659  0.49571083]
 [-0.91612358  0.93281501]
 [-0.38914322  0.97723658]
 [ 0.41071898 -0.24651302]
 [-0.73028746 -0.74933859]
 [-0.89137343 -0.18091723]
 [-0.67163264  0.75235193]
 [-0.47800762  0.7271211 ]
 [-0.92746244  0.87628712]]
<class 'numpy.ndarray'>
```

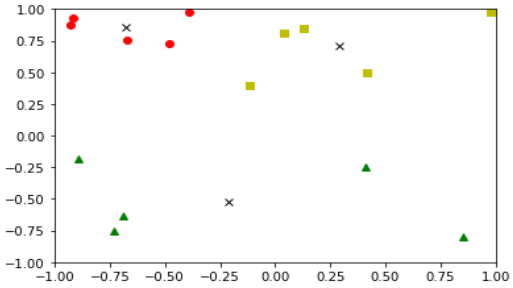


It is interesting to actually see the k-means classification algorithm in action (in this case k=3). Here I have given it a set of data point and told it to find three clusters (represented by the Xs at the center of the cluster). Each time, it is guessing a center to these clusters, attempting to find the correct one. It divides the data points into three clusters, takes the average (mean) distance of the points in these clusters and considers that several times (means – aha! Get it?) to improve the accuracy of the group/cluster. So here we have a data set, I have selected 3 random points as a starting position, then those points are compared to all other points in terms of distance. It determines which starting point is the closest. This is the first attempt at clustering. So now we have groups. But these are probably a bit off because the starting points were chosen randomly. Here is where the mean comes in. The average of all points in one appointed group and that is now used as the new center or starting point for that group. Then the clustering happens again to acertain new groups. Because the average pulls the X (guessed center) closer to the majority of each cluster, on the second time (k=2), the clusters will be more defined (hopefully) by their (hopefully) more accurate centers.

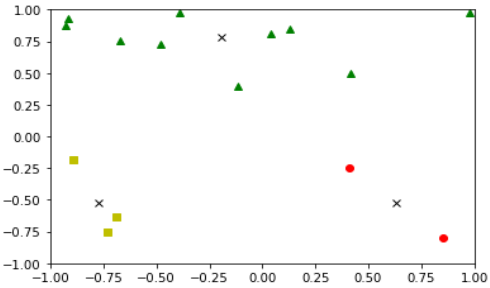
1:



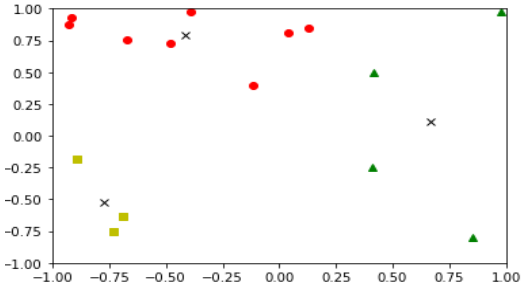
2:



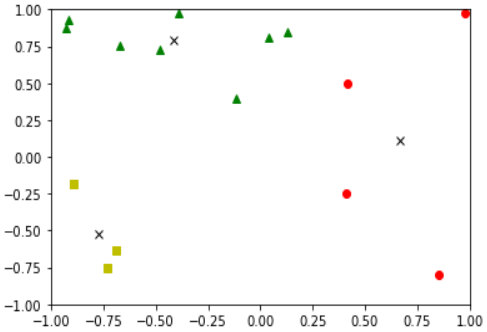
3:



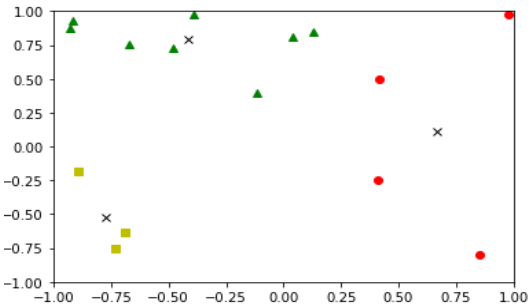
4:



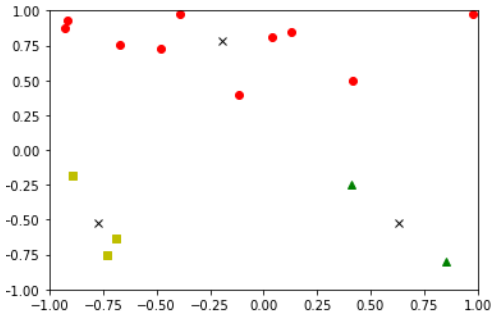
5:



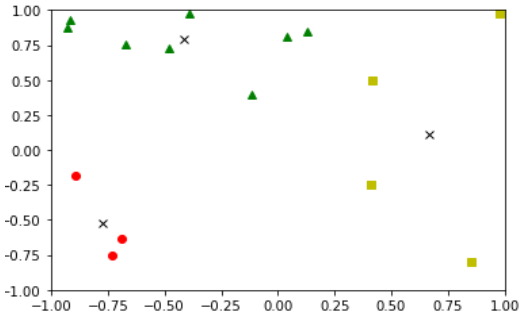
6:



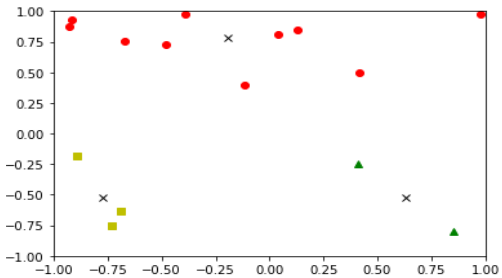
7:



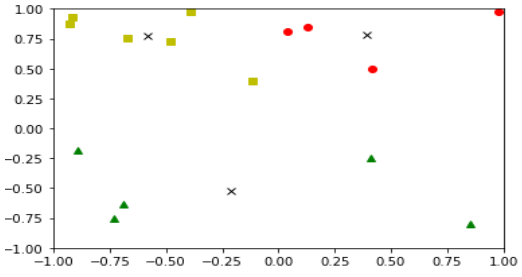
8:



9:



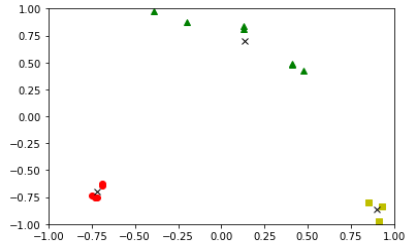
10:

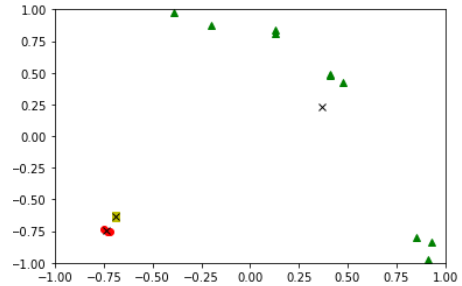


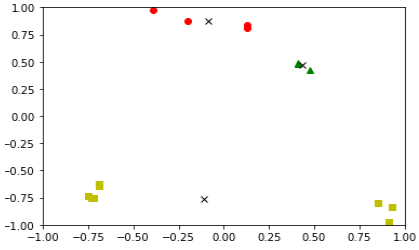
Considering the mechanics behind k-means, it is interesting to see which clusters each iteration produces. As humans, we are built to recognize patterns and can look at a graph and essentially interpret clusters (whether they be empiracally correct or not) but the algorithm might not find said clusters as can be seen in iteration 1 where one X is just placed in the middle of the graph attempting to capture the points on the bottom left and top right corners. Sometimes, the graphs of centers found are identical (3,7,and 9; 4,5,6, and 8) or complete inverses of one another (See 7 and 9 vs. 10 and 2). This indicates a problem with between-group distance versus in-group distance. Looking at graph 10, it is clear that the bottom x (center of the green cluster) started randomly far away from any of the points as this current position encapsulates points that are further from each-other than from other clusters, thus implying that, from its intial starting point, the average distance to the nearest points was somewhere in the middle of that empty space. Should the point have started say in the top left corner, it would have taken the much closer distances of the nearest points and had averaged them together and not have moved so far, attempting to be closer to the points and thus making a smaller cluster. Over-all, identical graphs produced/ clusters found were 1/10 (1), 2/10 (2, 10), 3/10 (3,7,9) and 4/10 (4,5,6,8) for an average of .25 or ¼ of the time same clusters are found.

2.) The self created data set of 15 points and the resulting graph is seen in figure 2. There are three very clear clusters, although I thought it would be interesting to try to spread one out slightly (see the green cluster) to see what clusters it finds. The algorithm was run 20 times on these points. Usually, the clusters it found were the "correct" ones or rather the ones that a human being would identify by lookin at the graph. These are the same displayed in the intial graph of the points. However, on some iterations, completely different clusters were found. Figure 3 depicts a variation sometimes found wherein one center point encapsulated the entierty of the top and right part of the graph. This is probaby because the point first appeared somewhere in the space between the top and bottom right clusters. This pattern was actually found 3/20 times. This pattern particularly surprised me as two centers of two clusters were found so very close to one another r(both in the bottom left). This is perhaps due to the points randomly generating very closely to one another in that same area then essentially splitting the nearest points between them. Another odd clustering can be seen in figure 4. Here, the entire set of bottom points (wherein y < -.25) are grouped together. The points above this line are divided into two groups. Again, this is absolutely due to starting placement of the center points. This clustering occurred 4/20 or 1/5 of the time. As can be seen, there is less varierty of clusters

```
[[-0.689      -0.629     ]
 [ 0.12923     0.8423    ]
 [ 0.127234    0.81234   ]
 [ 0.85234    -0.798234  ]
 [ 0.91234    -0.97      ]
 [-0.688      -0.639     ]
 [ 0.41        0.49      ]
 [ 0.93       -0.83457   ]
 [-0.389       0.977     ]
 [ 0.41071898  0.48      ]
 [-0.73028746 -0.74933859]
 [-0.75137343 -0.73091723]
 [-0.72163264 -0.75235193]
 [ 0.47800762  0.4271211 ]
 [-0.2         0.87628712]]
<class 'numpy.ndarray'>
```







found with this data set than the one used in part A – I would imagine this is due to the increase in nearness of some of the data points.

3.) It is well noted that although the k-means algorithm will always terminate, it does not necessarily find the most optimal configuration in terms of the clusters on a global level (it will find its local clusters but not necessarily all of them). The clusters found really depend on the initial values for the means as well as the starting points, and it frequently happens that suboptimal partitions are found (Matteucci, M.). It is difficult to check if your center is the global center and not just locally unless you know all points being considered by the problem (which, then I suppose you could brute force place a center on a data point and iterate through every single data point) but usualy this is not the case. The standard solution is to try different starting points, run the algorithm, and compare the findings. The starting points don't always have to be random, so if they were selected with purpose, this might help find the optimal number of clusters, although in unsupervised learning, prior knowledge might not be possible. On approach to this change of starting points is to put the first center on a data point, and then second center on a data point that is as far as possible from the first center. Continue this until the kth center which will be put on a data point that's as far away from the closest of the already established centers (Moore, 2001).

How can we compare clusters? One option is to make clusters and then use the rand index and adjusted rand index (Im Walde, S. S., 2006). The rand index is a measure of the similarity between two data clusterings and the index adjusted for the chance of grouping elements, respectively (Rand index, 2017). Essentially it is just a pairwise comparison between two sets of data resulting in a number between 0 and 1 with 0 being totally different and 1 being exactly the same. Should two clusters resulting from a k-means clustering have a rand index close to 1, it is logical to determine that these so-called clusters are probably just one cluster divided. Another option is to find the nearest neighbor cluster distance and further neighbor cluster distance, distance between cluster centroids, and average distance between clusters for each iteration and compare them (Im Walde, 2004). Another approach is Ward's method which defines the distance between two clusters as the loss of information when merging the two (Im Walde, 2004).

Another problem that exists is, in an unsupervised problem, how can we know how many clusters there are and, as a result, how can we know the optimal k-value (Matteucci, M.)? One approach is to run the same test with multiple values of k and take the one with the best results according to a given criterion. According to Moore, an oft-used criterion is the k value that minimizes the schwarz criterion (distortion + (lambda *num dimentions * k * log of the number of data points)) (Moore, 2001). However, it is worth noting that when increasing k you increase the chance of overfitting.

Works Cited

Im Walde S. S. (2004). Ph.D Dissertation. Universitat Stuttgart. From http://www.ims.uni-stuttgart.de/institut/mitarbeiter/schulte/theses/phd/algorithm.pdf

Im Walde, S. S. (2006). Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2), 159-194.

Matteucci, M. (n.d.). A Tutorial on Clustering Algorithms: K-Means Clustering. Retrieved November 01, 2017, from https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html#moore

Moore, A. (2001, November 16). K-means and Hierarchical Clustering. Lecture presented in School of Computer Science Carnegie Mellon University, Pittsburgh. Retrieved November 02, 2017, from https://www.autonlab.org/tutorials

Rand index. (2017, May 28). Retrieved November 03, 2017, from https://en.wikipedia.org/wiki/Rand_index