Unsupervised Pokémon

Mihai Matei
Practical Machine Learning Course
Data Science Master, 2019
University of Bucharest, Faculty of Mathematics and Computer Science

Abstract

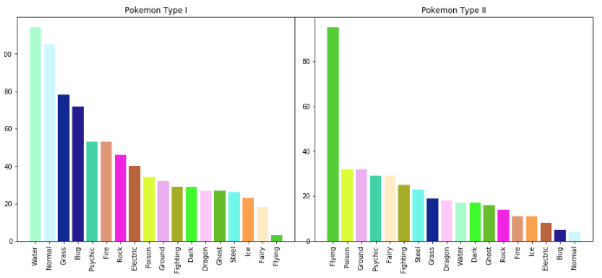
This paper tries to apply unsupervised machine learning and statistic methods to reduce the number of features in the data set represented by Pokémon images as well as to apply clustering techniques on those pruned features. Different metrics and graphs are being plotted for each studied model. To compare the results to the supervised classification models the Pokémon type is considered as before as a target label.

1. Introduction

Pokémons are very well known fictional creatures made popular around the world through movies and popular computer game series, like the whole madness of 2016. The Wikipedia page [1] describes the "890 fictional species of collectible monsters, each having unique designs and skills. Apart from their fantasy appearance each Pokémon has one or two "types", such as Fire, Water, or Grass. In battle, certain types are strong against other types. For example, a fire-type attack will do more damage to a grass-type Pokémon than a water-type attack, in a rock-paper-scissors-style relationship. In a unsupervised approach the target is not known and thus we cannot consider the known Pokémon type in training the model. First the features represented by each RGB image is being reduced and selected by applying different statistical and machine learning models: Anova filtering, Principal Component Analysis, Histogram of Oriented Gradients, a state of the art previously trained CNN network. I will use the above generated features to try clustering techniques on them such as Kmeans and DBSCAN and notice the impact of clustering parameters as well as feature selection has on the results. These results are being compared against each other and with some supervised approaches implemented in the last project. The same matmih own library was used.

2. Pokémon Dataset

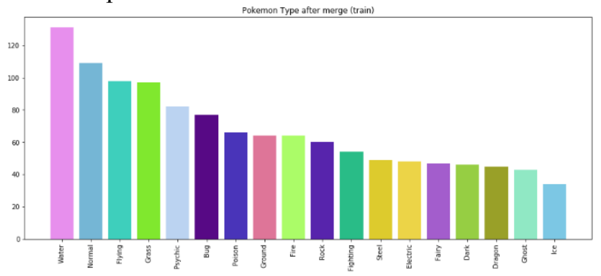
The data set [2] was chosen to contain all pokémons character images one time only and was taken from the popular Kaggle repository. It consists of a list of 809 characters, with their specific two types of attack, as well as 809 120x120x3 RGB images, one for each. All pokémons contain type1 information, but only 405 have the additional type2 provided. There are 18 target classes of pokémons types to use in the multi label classification(Bug, Dark, Dragon, Electric, Fairy, Fighting, Fire, Flying, Ghost, Grass, Ground, Ice, Normal, Poison, Psychic, Rock, Steel, Water).



You can notice that the target class distribution in the above histogram is quite unbalanced, being dominated by the Water and Normal class for type1 and for Flying for Type2.

2.1. Data set merging

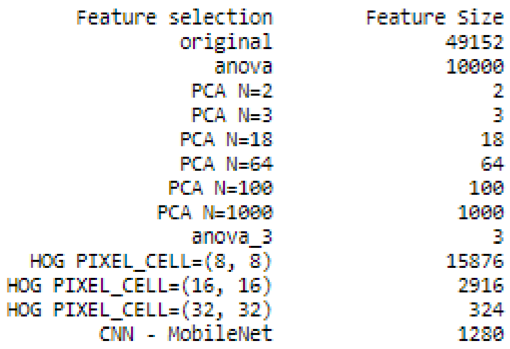
In order to balance the data set I first augment the data by also adding the type2 information as a new type1 entry. As such the total resulting data will be 1214 samples with a much more balanced distribution:



2.2 Unsupervised data set

Since I am applying unsupervised method I will not need to have a validation or test set. Hence all of the data will be used giving a total of 1214 training samples. Each sample will have 49152 features. I will also keep the type as the target value but it will only be used in computing supervised metrics for the clustering algorithms. Below one can see a sample of each type of pokemon.

To test different feature reduction techniques on the feature set and apply the resulting data to clustering techniques several methods are being tried: Anova, PCA, HOG, CNN. The resulted features sets are:



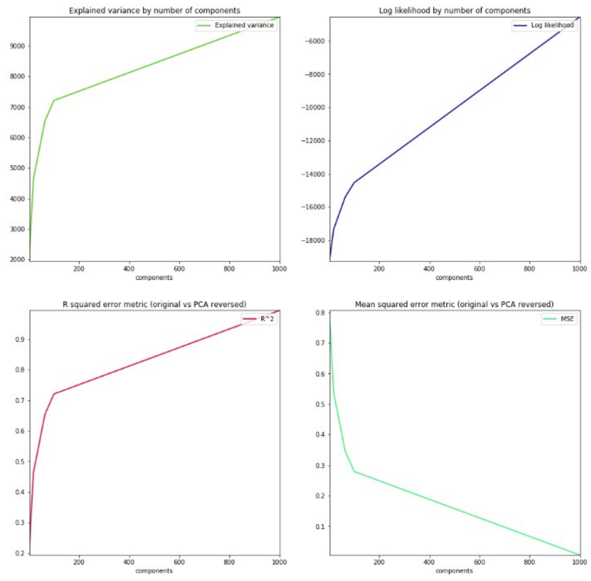
Feature selection	Feature Size
original	49152
anova	10000
PCA N=2	2
PCA N=3	3
PCA N=18	18
PCA N=64	64
PCA N=100	100
PCA N=1000	1000
anova_3	3
HOG PIXEL_CELL=(8, 8)	15876
HOG PIXEL_CELL=(16, 16)	2916
HOG PIXEL_CELL=(32, 32)	324
CNN - MobileNet	1280

3. Principal component analysis

PCA [3] is a statistical procedure that converts a set of observations into a set of values of linearly uncorrelated variables called principal components. This is done by either eigenvalue decomposition on the covariance matrix of the data or a singular value decomposition of the data. Each eigenvalue accounts for how much a vector is modified when the matrix is multiplied with it in the space denoted by base of eigen vectors. Since our matrix is the covariance data this implies that the first principal component will represent the projection with the highest variance of the data.

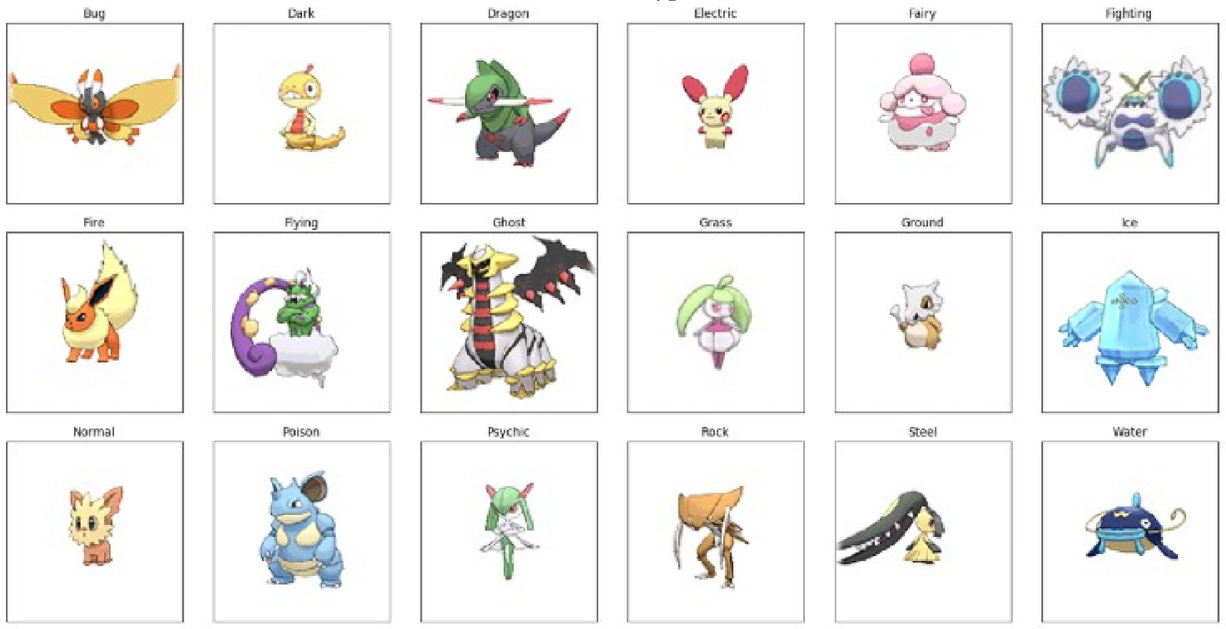
To do feature reduction using PCA I used scikit-learn library with the full solver. First I filtered the original feature set to get the top 10000 features with the highest variance using Anova and Constant variance filtering (to remove white spaces) and then normalized the features to 0 mean and 1 standard deviation.

Even if the full solver was used I applied the model for different number of resulting PCA components, (2, 3, 18, 64, 100, 1000), using my own implemented matmih library hyper parameter grid search for a model. Several metrics were plotted for each decomposition like the Explained variance, the log likelihood of train data under the PCA model, and the R squared and mean squared error regression metrics for the images that were reconstructed by reversing the PCA transformation. Results are below:



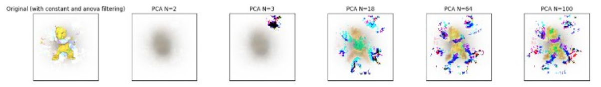
One interesting fact is the R2 score is basically the same, but on a different scale, as the explained variance. This seems to go along with the definition of

Pokemon Types

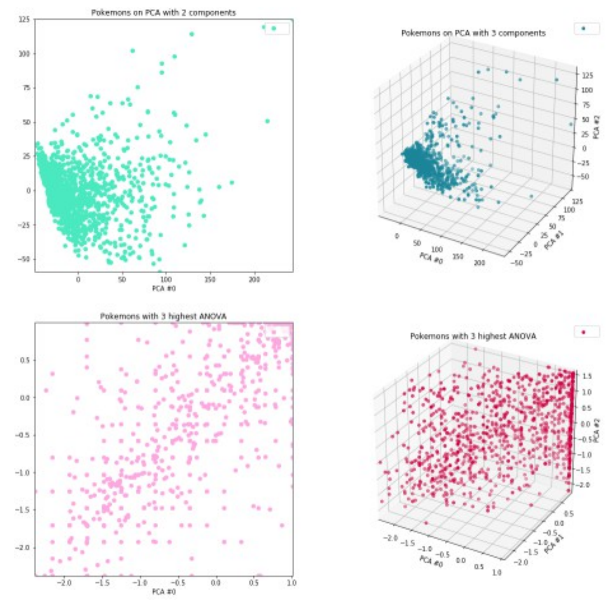


R2 as the explained prediction variance over the data variance. One can also see that MSE decreases and the R2, Explained variance and Log Likelihood increases as we increase the number of PCA components.

The reconstructed images are also shown to get a feeling on how much PCA compresses the data and the information loss of this compression:



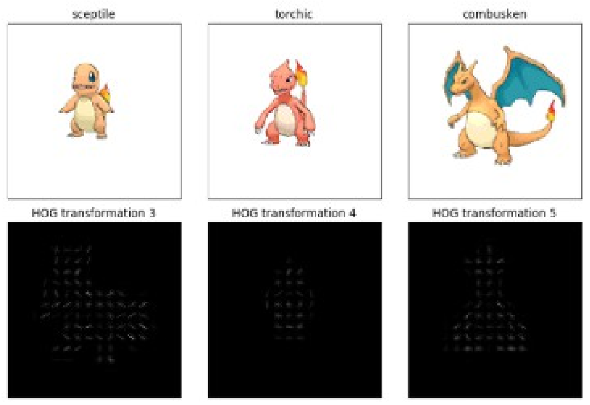
Since the computed PCA features will be used later on in clustering analysis I plotted the data reduced PCA with 2 and 3 components as well as Anova reduced features also to 2 and 3. One can notice that the resulting PCA variance is less than the Anova normalized variance, and in case of the PCA the presence of some outliers may be possible:



4. HOG and CNN reduction

To select more feature sets for the clustering analysis I used Histogram of Oriented Gradients and Convolutional Neural Networks to extract more differentiated features.

HOG is best explained in Practical Machine Learning course 10 [4] and the model was implemented using scikit-learn library. The hyper parameters used was both the directions of the gradients as well as the number of pixels per cell and cells per block. For our feature sets I used (8,8), (16,16), (32,32) pixels per block parameters. HOG's generates both a flatten feature vector that can be used later on as data for other machine learning models, in our case for clustering, and an image of how the new data looks like under the gradient transformation:



For the CNN model I used the state of the art MobileNetV2 [5] model trained on the imagenet data set. I took the logits output from the last dense layer prior to classification as the new CNN feature set, yielding 1280 new features. The same CNN model was used in Project 1 for doing transfer learning for a supervised pokemon type prediction.

5. Clustering analysis

The above feature sets were further used in 2 different unsupervised cluster algorithms namely KMeans and DBSCAN. The following new feature sets were used to compare the performance of the cluster algorithms under different hyper parameters: PCA 18, PCA 100, PCA 1000, HOG pixel_cell=(8, 8), HOG pixel_cell=(32, 32), CNN – MobileNet. Each feature set was further flattened and normalized to 0 mean and standard deviation of 1 prior to being used.

To do some software engineering a common clustering model was implemented that extends the mm.Model class. For each hyper parameter search that was done several unsupervised or supervised cluster metrics was computed as well as a custom made accuracy metric discussed below. For each model all metrics were plotted as well as several helper methods to get the best cluster hyper parametrization.

5.1 Cluster metrics

Computing cluster fitness is a difficult task and since this is an unsupervised problem the overall good fit of the model can also depend on how we would like the clustering to look like. Both supervised and unsupervised metrics are being computed as follows:

Unsupervised metrics

The metrics do not rely on any clustering information and compute several distance related metrics between cluster centers or cluster variance or probabilistic statistical information. Some metrics that are being computed are:

Silhouette Coefficient a higher Silhouette Coefficient score relates to a model with better defined clusters

Calinski-Harabasz Index known as Variance Ratio Criterion and higher score relates to a model with better defined clusters.

Davies-Bouldin Index signifies the average similarity between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves. Values closer to zero indicate a better partition.

Supervised metrics

This can be constructed when the true value of the cluster are known. I will use the Pokemon type information from the original data set as the truth value. Since the labels are between 0-17 (18 classes) some metrics might be monotonous.

Adjusted Rand index measures the similarity of the two assignments, ignoring permutations and with chance normalization. Between [-1, 1] negative values signifies independent labeling, similar clustering have a positive ARI.

Normalized Mutual Information used in clustering literature. Between [0, 1] values close to zero indicate two label assignments that are largely independent, while values close to one indicate significant agreement.

Homogeneity each cluster contains only members of a single class, between [0, 1] with greater values signifies better assignment.

Completeness all members of a given class are assigned to the same cluster, [0, 1] with greater values signifies better assignment.

V-measure harmonic measure of Homogeneity and Completeness 2*H*C / (H+C).

Fowlkes-Mallows scores uses classification metrics such as TP, FN to compute the index, between [0, 1]. A high value indicates a good similarity between two clusters.

Outliers measure the number of samples that are not part of any cluster (especially for DBSCAN).

5.2 Accuracy computation

One research task of this project was how to measure accuracy of a model so that I can compare the cluster model against a supervised classification approach. For a cluster assignment this is hard to compute as we do not know how to assign the cluster identifier to a true label. There is also the problem that the cluster numbers can be greater that the cluster labels and hence a classification label can be assigned to multiple clusters. I used 2 important prior elements to create the algorithm: contingency matrix and maximum matching in bipartite graphs.

Contingency matrix is a supervised method to create a classification matrix similar to the supervised confusion matrix where each row i represents a true label and each of the N j columns are each of the predicted clusters. Ci,j represents the number of true labels i present in cluster j. A column will sum up to the number of cluster assigned to predicted cluster, where a row will sum up to the total true labels of that class in the set.

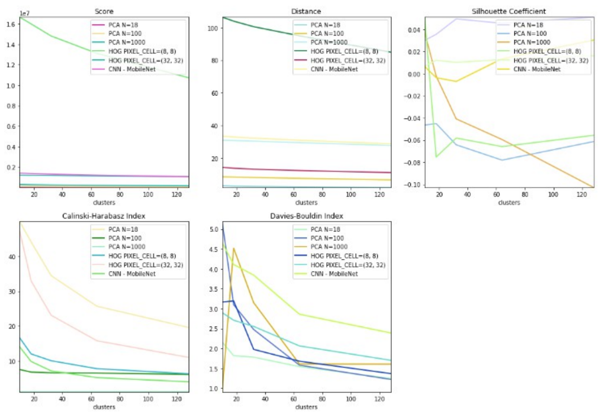
We can see the true label to cluster id assignment as a bipartite graph where on the left we have the true labels and on the right the predicted cluster ids. The graph edge matrix is represented by the contingency matrix. The maximum matching in this graph will represent the highest number of true labels that are being correctly assigned by considering that a cluster id represents a specific classification label. The maximum weight matching problem [6] can be computed in $O(V^2E)$ in a general graph but in bipartite graphs it is shown in Koning's theorem that is equal to the minimum vertex cover. This is solved using the Hungarian method implemented in scipy's linear_sum_assignment using the negative contingency matrix as a cost matrix.

The one problem that remains to be solved is that the above steps only compute TL assignments where TL is the number of true labels. To solve the case where the number of clusters is greater than the number of labels we can remove the samples that are in the current assigned clusters and re-apply the process until all clusters are assigned to a true label.

```
Algorithm Cluster accuracy:
  - compute contingency matrix (TL, PC)
  - solve bipartite maximum matching (– contingency)
  - add current matching to total predictions
  - remove all samples that are predicted to belong to previously assigned clusters from TL and PC
  - repeat process until all clusters are assigned
  - return total matching / total samples
```

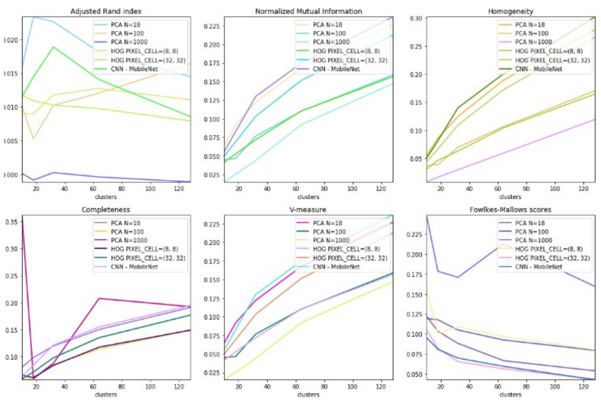
5.3 KMeans

Kmeans clustering is an expectation maximization algorithm that tries to minimize the total distance of the samples to a fixed number of centroids (cluster centers). The unsupervised model was implemented using scikit-learn Kmeans model and uses euclidean distance with the number of cluster as hyper parameters. Apart from the unsupervised and supervised metrics previously discussed I also compute the Distance and Score metrics. Distance represents the sum of the minimum distance between all samples and all centroids (hence the distance between the sample and its assigned cluster) and Score is scikit-learn internal metric that measures the inertia of the model. Both metrics can be used to apply the Elbow method to compute the best number of clusters. Silhouette Coefficient is also plotted and one can use the highest value for to select K from the tested 10, 18, 32, 64, 128 values:



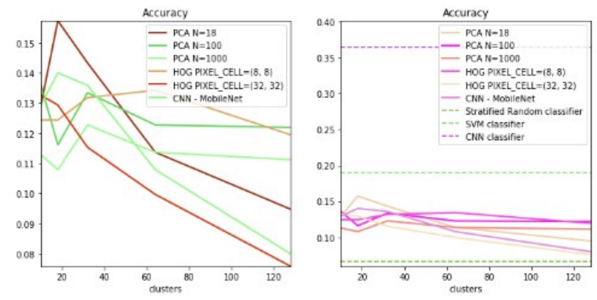
One can see that the metrics differ quite well between different feature set but for feature sets with larger amount of features a maximum Silhouette coefficient can be found around 40 or even larger.

For supervised methods we plot the cluster coefficients and notice that both the Completeness and Homogeneity increases with the number of clusters:



The Adjusted Rank Index seems to be bigger around 40 clusters but with PCA 100 this seems to increase with the number of clusters.

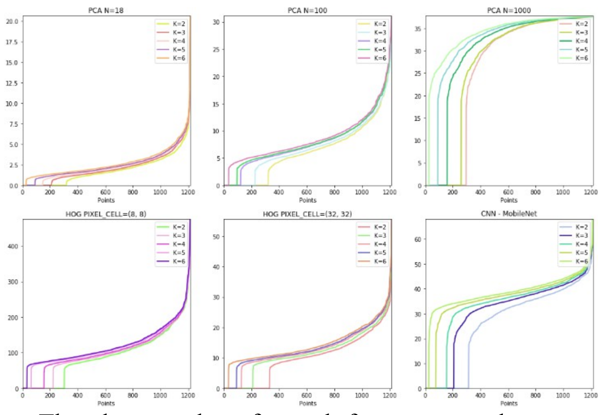
Probably the most interesting metric is the accuracy plotted as independent and against the Random model accuracy of 0.06 and SVM and supervised CNN of 0.19 and 0.365 computed previously in Project 1. One can see that depending on the feature set the best accuracy seems to be 18 and 32 clusters but starts to decrease for all feature sets when we increase the number of clusters. The overall accuracy seems to be between the random model and the SVM classifier:



5.4 DBSCAN

DBSCAN is a clustering algorithm than tries to group data points together by a chain link model where each point is added to a cluster if it is inside previously set up epsilon distance from a core point, where a core point contains at least N other data points inside the epsilon sphere. Though it requires minimum domain knowledge and can discover clusters of arbitrary shapes [7] it does not work well for clusters of variable density and it is hard to tune, especially the epsilon distance and the number of samples. The model is implemented using scikit-learn DBSCAN cluster and uses euclidean distance and has epsilon and the number of data points to define a core point as the hyper parameter of the model.

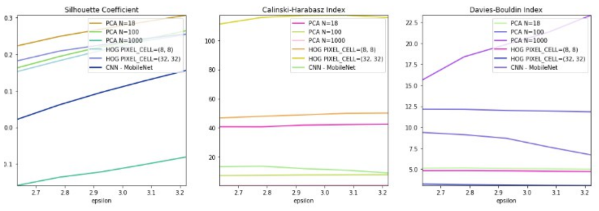
To start a good hyper parameter space search for the model one can plot the Kth distance between each data point and choose the inflection point with the largest slop change. This is done by computing the distance between each data point, plotting the sorted Kth distances for each feature set and choosing the point where the gradient is largest. A linear interval is then chosen between the found epsilon for the grid hyper parameter search. The K number of samples to define a core point is kept fixed to the found best point.



The plots are done for each feature set and one can see that the distances are very similar for different values of K between 2 and 6. Greater values of K around 60 yield the same result. Then by computing the gradients I can find the best epsilon and K for each feature set:

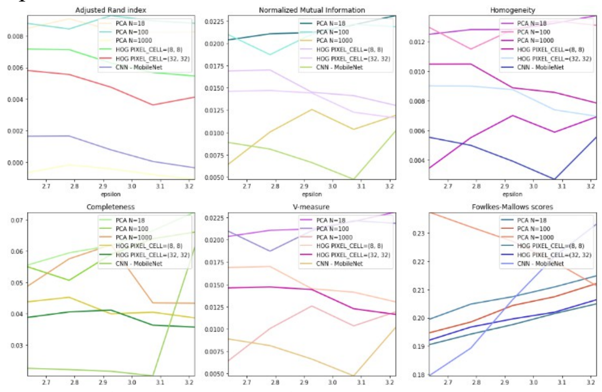
```
 ================Best Found Epsilon Value================
                  FEATURE SET              EPSILON       K
                     PCA N=18               2.9277       6
                    PCA N=100               9.2879       6
                   PCA N=1000               27.395       6
          HOG PIXEL_CELL=(8, 8)             128.68       6
        HOG PIXEL_CELL=(32, 32)              15.64       6
              CNN - MobileNet               41.023       6
```

The hyper parameter model grid search will look for 5 values of epsilon between [0.9*eps, 1.1*eps] and keep K constant to the selected best value. This is done for each feature set. The unsupervised metrics are shown below but please keep in mind that the X values of the epsilon are not the same but they were plotted together to better highlight differences:

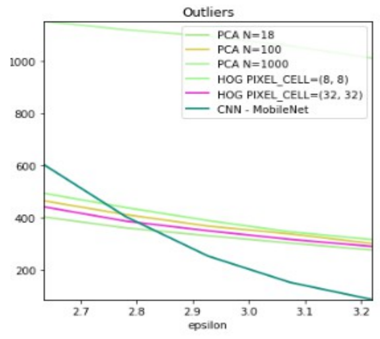


One interesting fact is that the Silhouette coefficient seems to be increasing with epsilon suggesting that perhaps even larger values should be used. Silhouette is always higher for DBSCAN.

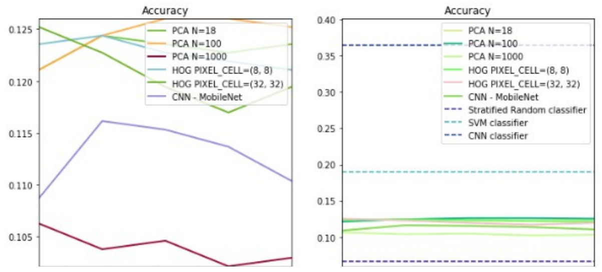
The unsupervised seem to be different among the different feature set either increasing or decreasing as epsilon is increased:



Another interesting plot are those for the Outliers that for all feature sets seem to be decreasing as epsilon increases meaning that some points are outliers from a core cluster center. For higher values of epsilon the amount of data points that have not been clustered is under 400.



The last plot that also interest us is accuracy:



One thing to note that all outliers in computing accuracy was assigned to a true label. The clustering performance is around 0.125 for the best cluster and varied a lot but tends to be the biggest around the best value of epsilon.

6. Conclusions

Clustering is hard but both Kmeans and DBSCAN seem to the same accuracy levels for the supervised classification analysis. Kmeans is a bit better producing 0.16 on PCA with 18 components. DBSCAN is slightly less performing with the best accuracy of 0.126 on PCA with 100 components. The Silhouette Coefficient seems to be higher for DBSCAN (0.3) than for Kmeans 0.04. Homogeneity and Completeness seems to be always increasing for Kmeans but random and kind of constant for DBSCAN. All the Adjusted Rand Index and Normalized Mutual Information seem to be low for both models on all the feature sets. DBSCAN could be used for outlier class detection as around best epsilon we have close to ¼ of the data set.

7. References

[1]https://en.wikipedia.org/wiki/List_of_Pok%C3%A9mon

[2]https://www.kaggle.com/vishalsubbiah/pokemon-images-and-types

[3]https://en.wikipedia.org/wiki/Principal_component_analysis

[4]https://practical-ml-fmi.github.io/ML/

[5]https://arxiv.org/abs/1801.04381

[6]https://en.wikipedia.org/wiki/Matching_(graph_theory)

[7]https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf