Machine Learning and Pattern Recognition Project

Das Debojit
Davletberdin Azamat
Mammadli Shahriyar
Mehta Tushar
Shah Bansi

Supervised by
Dr David Nebel
M. Sc. Alexander Engelsberger

Fakultät Angewandte Computer- und Biowissenschaften

Hochschule Mittweida

January 2023

Introduction

Task:

Given a 10 dimensional dataset with 512 data points.

To solve:

1. Determine a reasonable number of clusters inside the data. Decide by which measure this evaluation will proceed

2. Fit a GMM model with the data and choose the number of components regarding your outcome of the first part.

Dataset Initial Preparation

After importing the given dataset and needed libraries to the Python, we started with checking the dataset for possible missing values and outliers. There were no missing values (NA).

To check for outliers we turned the data into Z-scores (columnwise), and found out two outliers:

	feat1	feat2	feat3	feat4	feat5	feat6	feat7	feat8	feat9	feat10
42	-3.02306	7.95507	4.57720	29283.12004	-7.86535	-6.37606	-9.36859	6.53065	1.91527	3.12621
108	91.31056	102.31521	92.20416	108.78853	95.24764	103.03824	95.28230	100.47884	101.19196	92.45533

Fig. 1: Outliers

We will have two copies of the same dataset. First, original version X, including outliers. Second, modified version X_withoutOutliers, which consists of all the data points except the above two.

After initial data preparation we could move to next part, dimensional reduction of the dataset.

Dimensional reduction to 2D by PCA method

We are going to reduce dimensions of the given dataset, so that it could be possible to visualize it. For that we will use PCA method.

At first we standardized the datasets ("X" and "X_withoutOutliers") using built-in StandardScaler function from sklearn.preprocessing. Then we applied a PCA model with 2 components and applied it to both datasets.

We will keep using the whole data with 10 features for clustering. Reduced versions will only serve to visualize the clustering results.

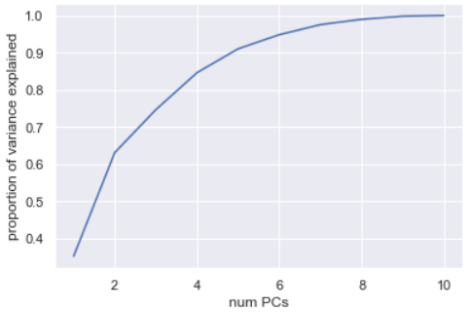

Fig. 2: Fraction of variance explained for X

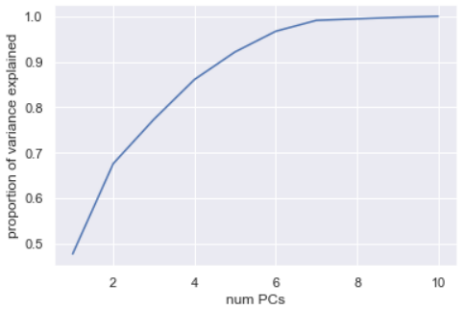

Fig. 3: Fraction of var. exp. for X_withoutOutliers

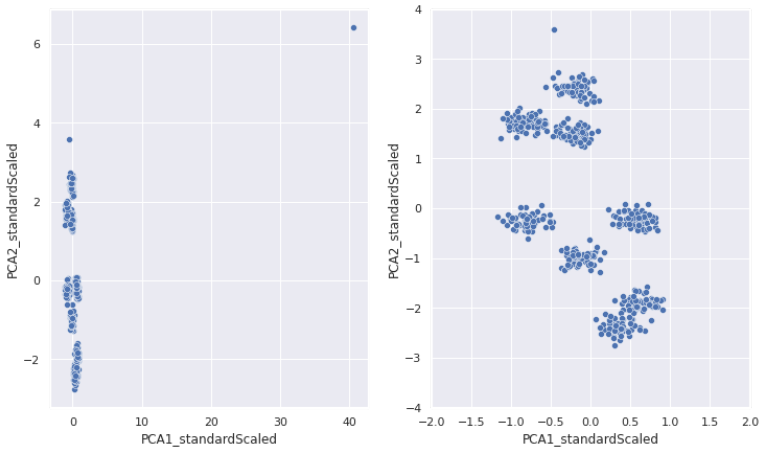

Fig. 4: PCA reduced dataset with outliers (the right figure is the zoomed version of the left figure)

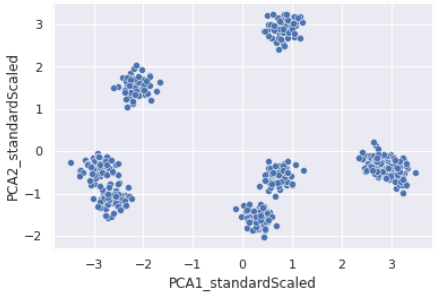

Fig. 5: PCA reduced dataset without outliers

We will observe 3 cases of training and clustering:
- training and clustering the dataset without outliers
- training and clustering the dataset with outliers
- training dataset without outliers but clustering dataset with outliers

Case 1. Training and clustering the dataset without outliers

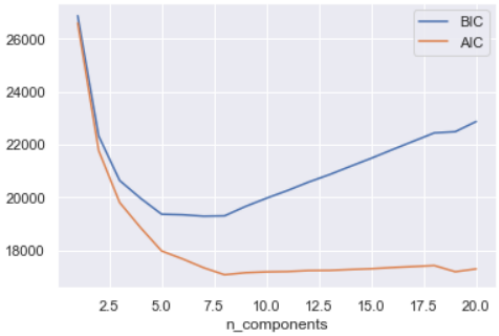

Fig. 6: BIC loss is minimized in the range between 5 and 8.

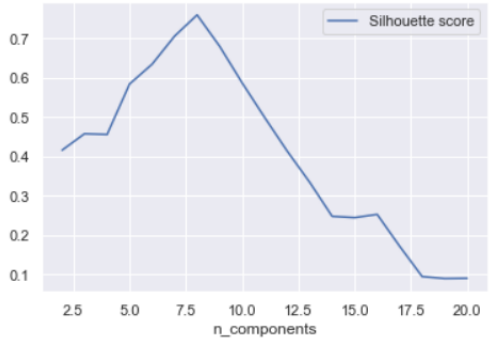

Fig. 7: Silhouette scores peaking at 8 indicates that 8 could be a good choice.

So we will apply GMM to each option of possible number of clustering in the range.

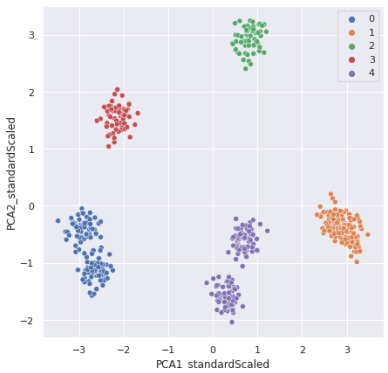

Fig. 8: Clustering with 5 components

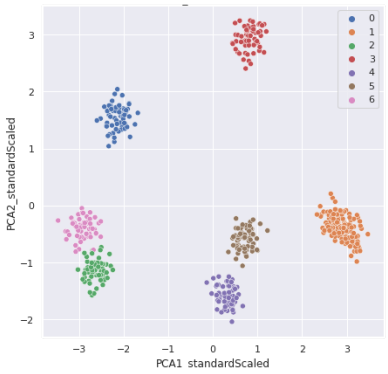

Fig. 9: Clustering with 7 components

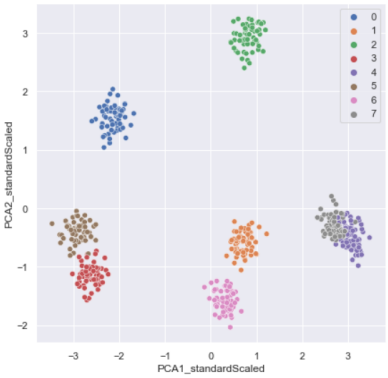

Fig. 10: Clustering with 8 components

Case 2. Training and clustering the dataset with outliers

With outliers, clustering becomes more complicated. From BIC loss we see that optimal number of clusters increase and now lies between 7 and 11.

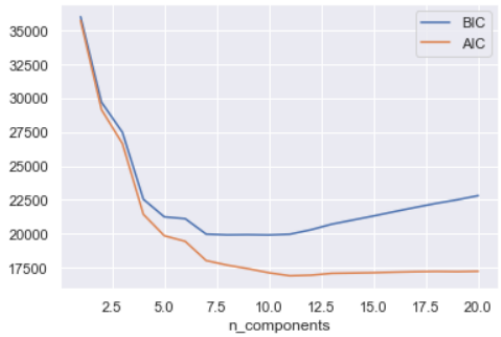

Fig. 11: BIC loss is minimized in the range between 7 and 11.

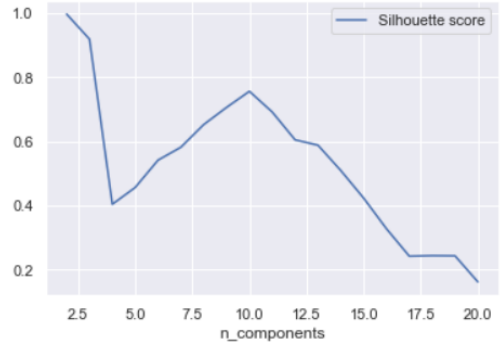

Fig. 12: Silhouette scores peaks around 10.

We fit GMM model with the above mentioned range of components (from 7 to 11) and find that among them possible optimal number of clusters could be 10.

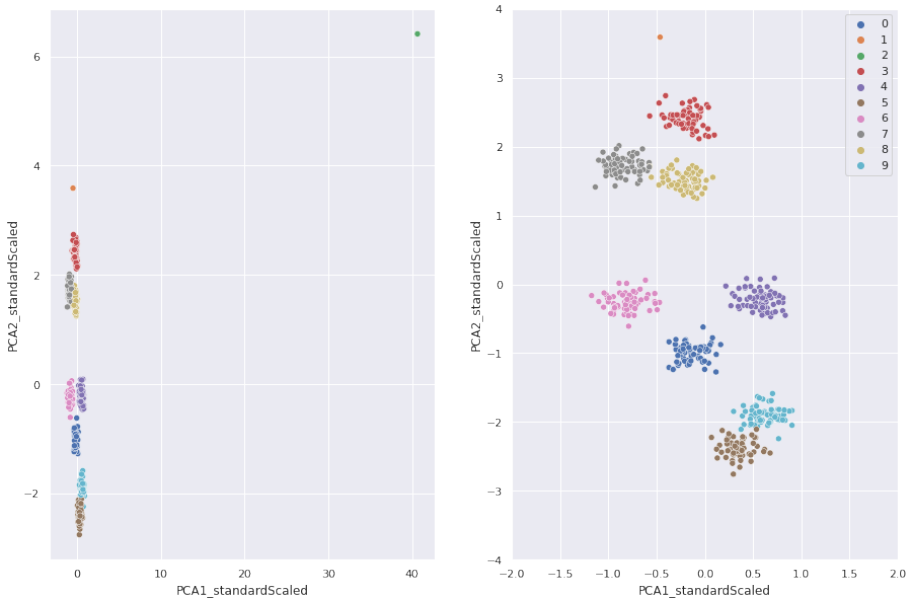

Fig. 13: Clustering the dataset with outliers into 10 clusters (zoomed version on the right side)

But we can observe and should mention that the two previously identified outliers are forming two distinct clusters of their own, for all models.

Case 3. Training dataset without outliers but clustering dataset with outliers

If we train the models on the dataset with outliers (such as in case 1) and then try to predict the original dataset including outliers, GMM tries to assign the outliers to

one of the main clusters.

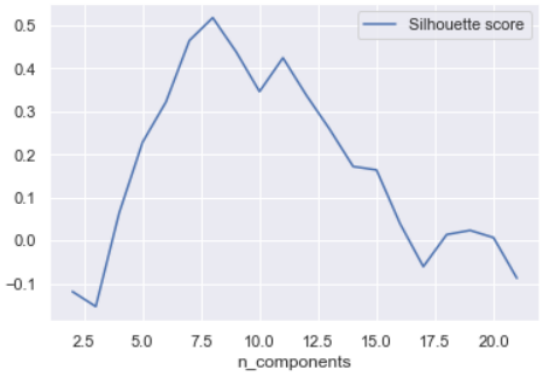

Fig. 14: Silhouette scores peaks at 8.

In this case, BIC/AIC loss minimization seems to be useless, however silhouette scores indicate that 8 could be ideal number of clusters.

So as the optimal numbers of clustering in the case 1 (dataset without outliers) were 5 and 7, we will use them for this case of clustering of dataset with outliers.

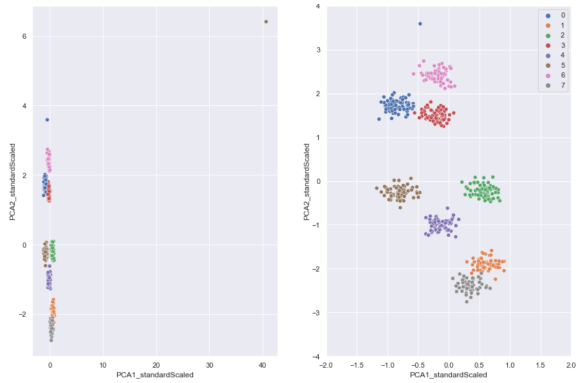

Fig. 15: Clustering of whole dataset with outliers into 8 clusters trained on dataset without outliers.

Conclusion

In process of clustering the given dataset we had three different cases of scenarios on how to treat the outliers. And we conclude that even small number of outliers have pretty significant impact in the form of increasing in the number of clusters while training of the dataset. Therefore, it is very important to pay attention to the possible outliers in the dataset during the training and subsequent clustering of the dataset.

Part II

Introduction

Task:

Given a data set called classification data set.

1. To implement the Generalized Learning Vector Quantization (GLVQ) algorithm to find an optimal number of prototypes using cross validation.

2. To classify another set of unlabeled data with our trained result along with the prediction.

Implementing the Generalized Learning Vector Quantization (GLVQ) algorithm to find an optimal number of prototypes using cross validation.

First, we import libraries such as NumPy, matplotlib, pandas and scikit learn and load the given datasets. We convert the names to numbers to visualize it and we do it in 2D for better visualization.

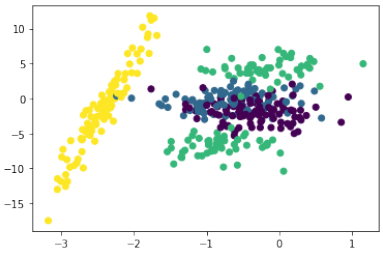

Fig. 16: Dataset with respective classes

Next, we standardize and normalize the dataset.

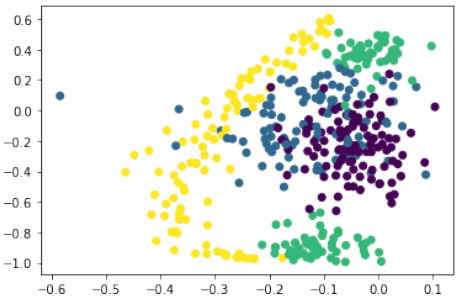

Fig. 17: Normalized Dataset

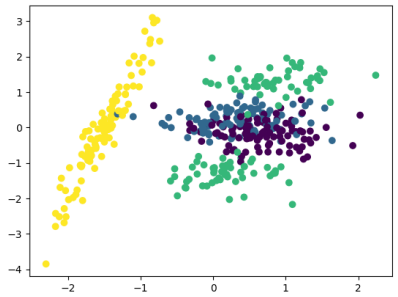

Fig. 18: Standardized data

As we can see from figure 10 and 11, it appears preferable to standardize the dataset.

Next we will initialize the prototypes. Here we have tried 4 different methods. These are -

1. rng.uniform
2. rng.random
3. random data point from the dataset
4. hybrid initialization (choosing a random prototype with 1 prototype per class and then using KMeans to choose 1 prototype per class).

Since the *rng.random* method gives the best results for accuracy, we choose this method for initializing prototypes.

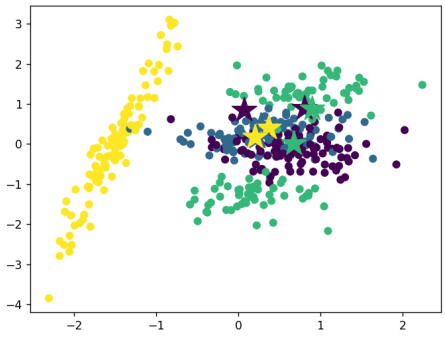

Fig. 19: Initialized the prototypes with rng.random.

Further, we define a function GLVQ which is our main function to train the prototype and then later predict the new labels based on these prototypes. Next, we use GridSearchCV to find the optimal number of prototypes, epoch, learning rate and learning time. Here are a few good results :

```
# Normalized data with random initialization
# Best Parameters:  {'epoch': 1100, 'lt': 4, 'lr': 0.4, 'ppc': 2}
# Best Score:  0.8780487804878049
# Best Parameters:  {'epoch': 900, 'lt': 4, 'lr': 0.4, 'ppc': 4}
# Best Score:  0.8536585365853658
# Best Parameters:  {'epoch': 1100, 'lt': 4, 'lr': 0.4, 'ppc': 2}
# Best Score:  0.8536585365853658
# Best Parameters:  {'epoch': 1000, 'lt': 3, 'lr': 0.5, 'ppc': 4}
# Best Score:  0.9024390243902439
# Best Parameters:  {'epoch': 1100, 'lt': 3, 'lr': 0.5, 'ppc': 4}
# Best Score:  0.8780487804878049
```

Here ppc:1 denotes 2 prototypes per class and if it was ppc: 2 then it denotes 4(this because of the hybrid initialization).

But we reject 4 prototypes per class since after performing the GLVQ algorithm on it and when we visualize it, the results are not good enough as can be seen in the figure below.

```
#Initializing random prototypes, standardized data(stopped after 61 mins).
# Best Parameters:  {'epoch': 1200, 'lt': 5, 'lr': 0.5, 'ppc': 2}
# Best Score:  0.975609756097561
# Best Parameters:  {'epoch': 1000, 'lt': 2, 'lr': 0.5, 'ppc': 4}
# Best Score:  0.975609756097561
# Best Parameters:  {'epoch': 1100, 'lt': 3, 'lr': 0.5, 'ppc': 4}
# Best Score:  0.975609756097561
# Best Parameters:  {'epoch': 1100, 'lt': 4, 'lr': 0.4, 'ppc': 4}
# Best Score:  1.0
# Best Parameters:  {'epoch': 900, 'lt': 5, 'lr': 0.5, 'ppc': 4}
# Best Score:  0.975609756097561
```

```
#Normalized data with hybrid initialization
# Best Parameters:  {'epoch': 1100, 'lt': 5, 'lr': 0.5, 'ppc': 1}
# Best Score:  0.9024390243902439
# Best Parameters:  {'epoch': 800, 'lt': 2, 'lr': 0.4, 'ppc': 1}
# Best Score:  0.9024390243902439
# Best Parameters:  {'epoch': 1000, 'lt': 2, 'lr': 0.5, 'ppc': 1}
# Best Score:  0.926829268292683
# Best Parameters:  {'epoch': 1000, 'lt': 3, 'lr': 0.5, 'ppc': 1}
# Best Score:  0.926829268292683
# Best Parameters:  {'epoch': 800, 'lt': 2, 'lr': 0.5, 'ppc': 1}
# Best Score:  0.9024390243902439
```

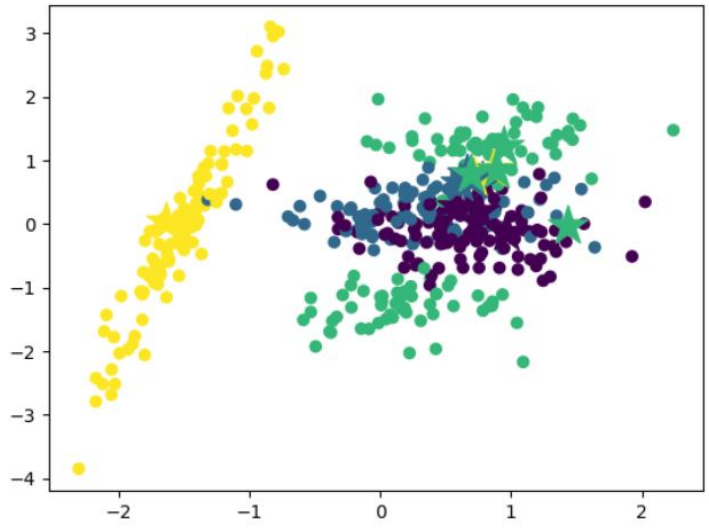

Fig. 20: 4 Prototypes per class(Prototypes were not well divided among the respective classes)

So then we use only 2 prototypes per class and perform the GridSearchCV again and below are the results: For standardized data with hybrid initialization the range of maximum score is between 82% and 92% for different epoch, lt, lr and 2 prototypes per class.

For normalized data with hybrid initialization the range of maximum score is again between 82% and 92% for different epoch, lt, lr and 2 prototypes per classand the best result we got was for standardized data with random initialization with the maximum scores between 95% and 97%.

```
# Initializing random prototypes, standardized data
# Best Parameters:  {'epoch': 1100, 'lt': 3, 'lr': 0.5}
# Best Score:  0.975609756097561
# Best Parameters:  {'epoch': 1100, 'lt': 3, 'lr': 0.5}
# Best Score:  0.9512195121951219
# Best Parameters:  {'epoch': 900, 'lt': 3, 'lr': 0.5}
# Best Score:  0.9512195121951219
# Best Parameters:  {'epoch': 1000, 'lt': 4, 'lr': 0.5}
# Best Score:  0.975609756097561
# Best Parameters:  {'epoch': 1000, 'lt': 5, 'lr': 0.5}
# Best Score:  0.975609756097561
```

And we chose to use epoch, lt, lr, = 1000, 4, 0.5 and with these values we use KFolds to find the accuracy score and we find that it gives the best accuracy score.

```
# Standard data random init 1000, 4, 0.5
# Accuracy: 0.79 (+/- 0.15)
# Accuracy: 0.85 (+/- 0.11)
# Accuracy: 0.81 (+/- 0.15)
# Accuracy: 0.78 (+/- 0.16)
# Accuracy: 0.80 (+/- 0.16)
# Accuracy: 0.81 (+/- 0.16)
# Accuracy: 0.81 (+/- 0.15)
# Accuracy: 0.80 (+/- 0.16)
# Accuracy: 0.81 (+/- 0.22)
# Accuracy: 0.77 (+/- 0.17)
```

Classifying another set of unlabeled data with our trained result along with the prediction.

Now we initialize the prototypes and run the GLVQ algorithm and obtain the trained prototypes and then we predict the classes for the test data provided and visualize it in 2D for better visualization.

After training the model using GLVQ algorithm, the following are the predicted classes for the test data provided.

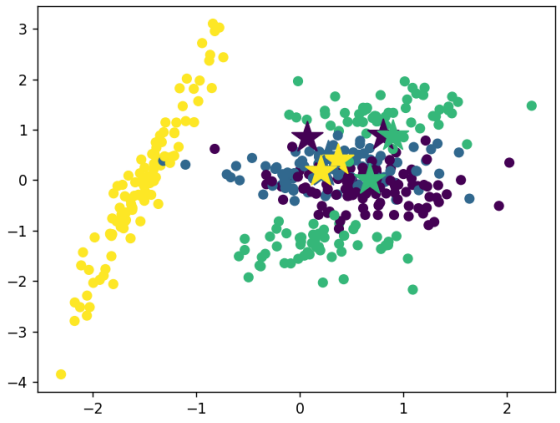

Fig. 21: Initialized prototypes

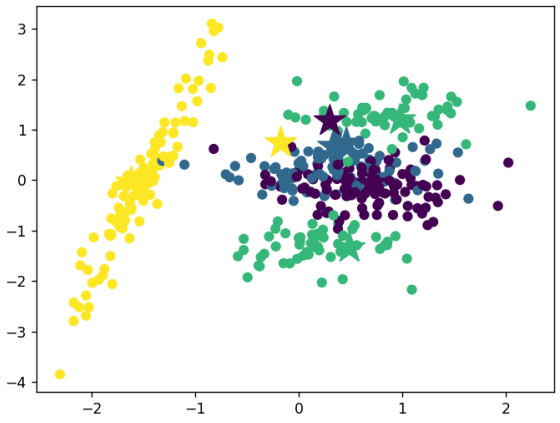

Fig. 22: Prototypes after GLVQ algorithm

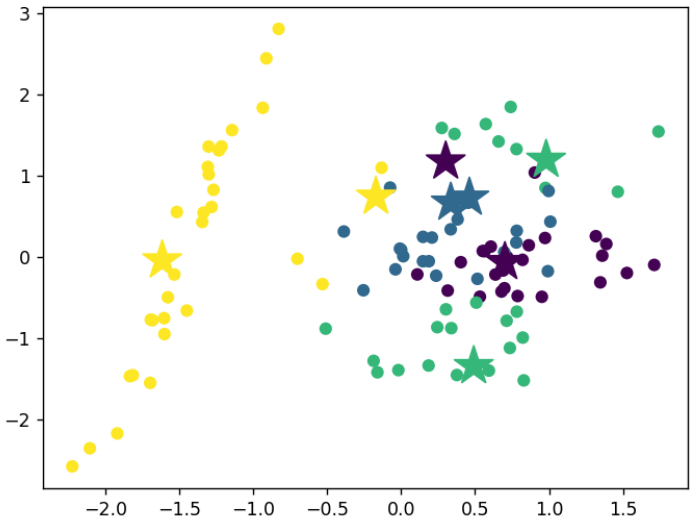

Fig. 23: Predicted Classes

Conclusion

In conclusion, we can say that we found out that the best method was to standardize data and to randomly initialize prototypes for the dataset provided. Since, it has the best accuracy score for epoch, lt, lr = 1000, 4, 0.5 and those were the best valued among what we have generated using the GLVQ algorithm.