EE 565 - Machine Learning I Project 5 Report

Jorge A. Garcia

***Abstract*—In this project, Self Organized Maps are used to explore unsupervised learning, identifying similarity in data. Density estimation using a histogram and Gaussian Mixture Models is done.**

***Index Terms*—Machine Learning, Data Science**

I. Introduction

Kohonen Self-Organizing Maps are a type of neural network that perform unsupervised learning. They essentially map any input into a new discrete space that is self converged. Due to this learning process, similar objects are grouped together and as such this type of network is useful for clustering or dimensionality reduction.

In previous projects we had approached the task of classification by finding the decision boundary that best explains the data. Another approach can be estimating the distribution of the data, and through a probabilistic test one can define the class to which new points belong. One method to approximate the data distribution is through the usage of Gaussian Mixture Models, in which a number of Gaussian models can be used in conjunction as a universal density estimator.

II. Animal Contextual Map Using a SOM

A. Tool Used: SOMPy

The tool chosen to implement the Self Organized Map was *SOMPy* [1] for Python 3. Besides the choice of node dimensions, it possible to choose between Gaussian and circular neighborhoods for weight updating, rectangular and hexagonal lattices, and use of either random initialization or using PCA. Only planar mapping is currently available, which is the desired one anyways. Gaussian neighborhood and rectangular lattice are chosen, with the default initialization using PCA. The models are trained for 2000 epochs, and the default learning rate is used.

B. Contextual Map and Attribute Distribution

A contextual map of the trained SOM can be seen in Figure 1. Given the original data set, there was trouble trying to separate the Horse and Zebra, and the Owl and Eagle instances due sharing the exact same traits. There was also an issue in separating the Duck and Goose instance, despite having one different attribute. Due to this, some slight modifications in attribute values were done in the data set to achieve complete separation of the instances. This modified data set is the one used for Figure 1. Even after modification of the data and playing with various SOM sizes, there was difficulty in effectively separating all of the classes.

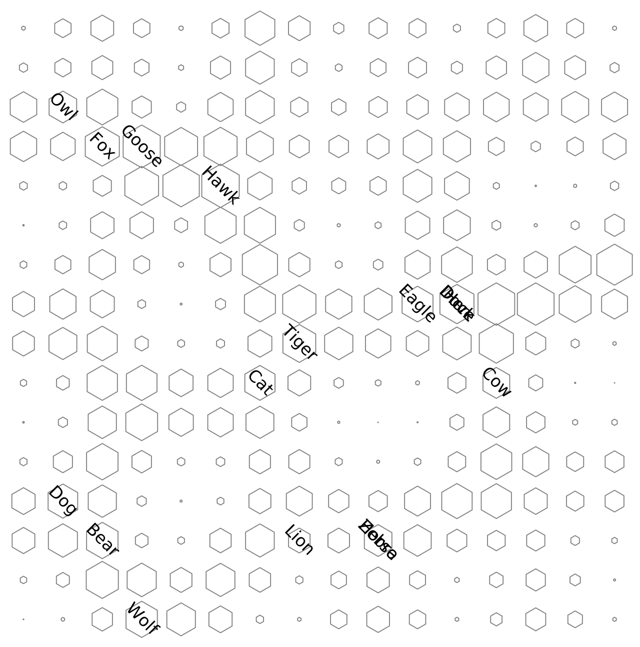

Fig. 1. Contextual map of the “animals.txt” data set.

Three different sets of attributes are queried and displayed in Figure 2. The first (top left) displays the three possible animal sizes, the second (top right) compares quadrupeds with bipeds, and the last one (bottom) shows the regions of animals that hunt and those that don’t.

III. Star Wars Sentient Species Contextual Map Using a SOM

A data set of various sentient species from the Star Wars franchise was created, using binary attributes to describe their physiological characteristics. The data set spans 14 species described through 15 different attributes. The resulting contextual map can be seen in Figure 3. From a personal point of view, this is an awesome map and not far off from something I would consider very accurate. The only ones that seem a bit ill-placed in my opinion are Sullustans and Ewoks, which the former I would group closer to the humanoids and the latter closer to Wookies. As the number of descriptors increase, the model would likely generate much better and sensible groupings.

As in the previous section 4, three different traits are queried and plotted in Figure 4. The species type (top left), number of fingers (top right) and species height (bottom) are shown.

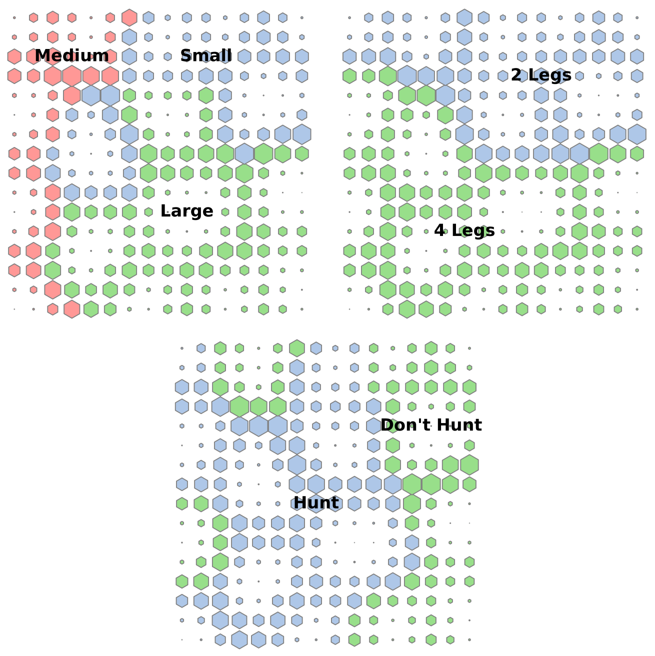

Fig. 2. Feature maps of animal size (top left), number of legs (top right) and predator status (bottom).

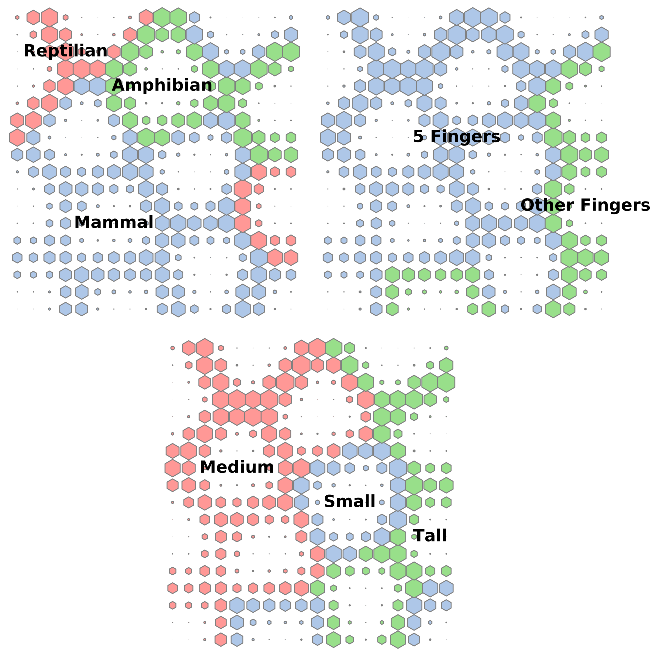

Fig. 4. Feature maps of species type (top left), number of fingers (top right) and species height (bottom).

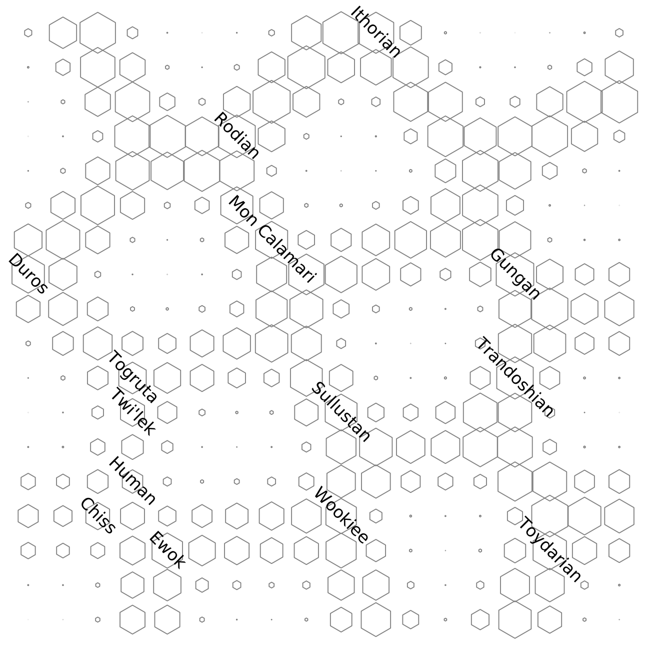

Fig. 3. Contextual map of the "starwars.csv" data set.

IV. Density Estimation

A function *histogram* was designed, with the capability of finding the number of data points in a given set that fall in a a range of N bins. This function was then used to estimate the density of the provided "DatasetA.csv" data. Bin sizes of 20, 200 and 2000 were used, and the resulting histograms are shown in Figure 5. While the details and the number of counts in the distribution change with bin size, the overall shape that envelopes the data can be appreciated throughout. It appears that there are 3 different Gaussian distributions describing this data set (2 small ones to the side and a more prominent one in the center).

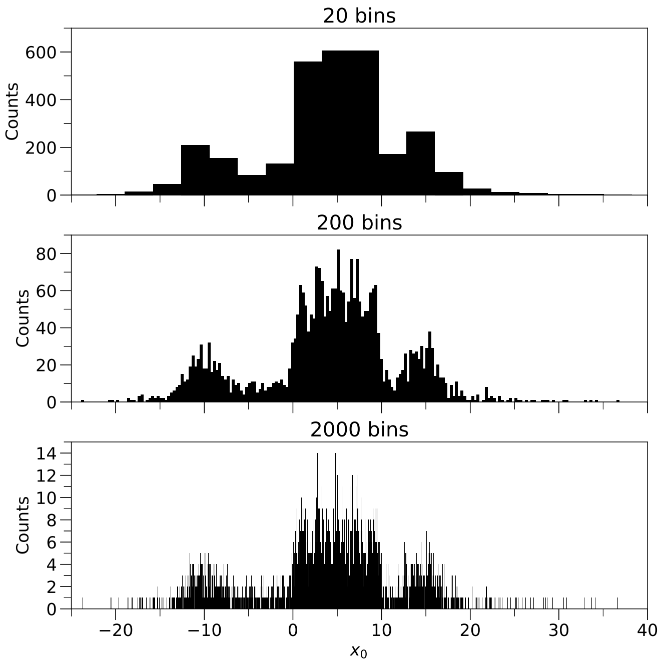

Fig. 5. Histograms of the "DatasetA.csv" data set using 20 (top), 200 (middle) and 2000 (bottom) bins.

V. Classification using Density Estimation

A. Tool Used: Scikit-learn

The *scikit-learn* library [2] contains a class *GaussianMixture*, which is used for this section. Besides allowing for a varying number of components the type of covariance can be chosen, as well as a choice between K-means and random initialization methods. Full covariance was considered, and the default initialization method with K-means was used.

A double moon distribution with $r = 1$, $w = 0.6$, $d = -0.5$ and $N = 1000$ points was generated and used a training set. Gaussian mixture models using $K = 1, 2, 3$ and 5 components were fitted to each class individually. From these fitted models, 3000 points are drawn and plotted in Figure 6. Given a single component, this results in two overlapping Gaussian distributions. As the number of components in the mixture model increases, it approximates the actual double moon distribution very nicely.

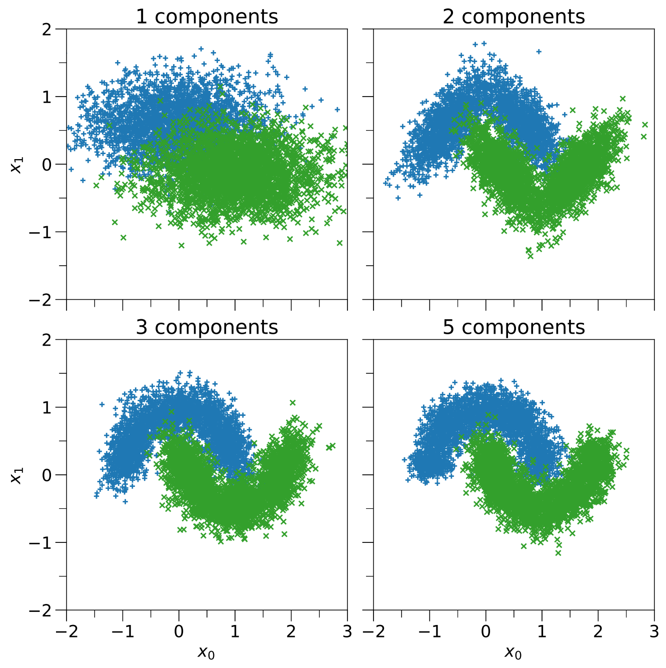

Fig. 6. Plots of 3000 randomly sampled points from the fitted GMM. The overall shape of the double moon distribution is better captured with an increasing number of components.

A new testing data set with the same r, w and d parameters and $N = 3000$ points was generated. The likelihood of the generated points belonging to each mixture model is calculated, and the maximum likelihood indicating the corresponding class. The prediction for the testing set, along with the decision surface for the model, is shown in Figure 7. Prediction accuracy increases with the number of components, achieving 86.4%, 99.8%, 100.0% and 100.0% accuracy at 1, 2, 3 and 5 components respectively.

VI. Conclusion

In this report, we implemented a Self Organizing Map to obtain contextual maps for an animal data set and a Star Wars sentient species data set. The results obtained were not the best with the animals data set, and this could be either due to the limited information in the data set or the implementation of the algorithm from the library used. A histogram algorithm was developed and used to estimate the density of a provided data set. The performance of Gaussian Mixture Models was tested on the double moon data set, which is able to achieve perfect performance given enough components to approximate the distribution.

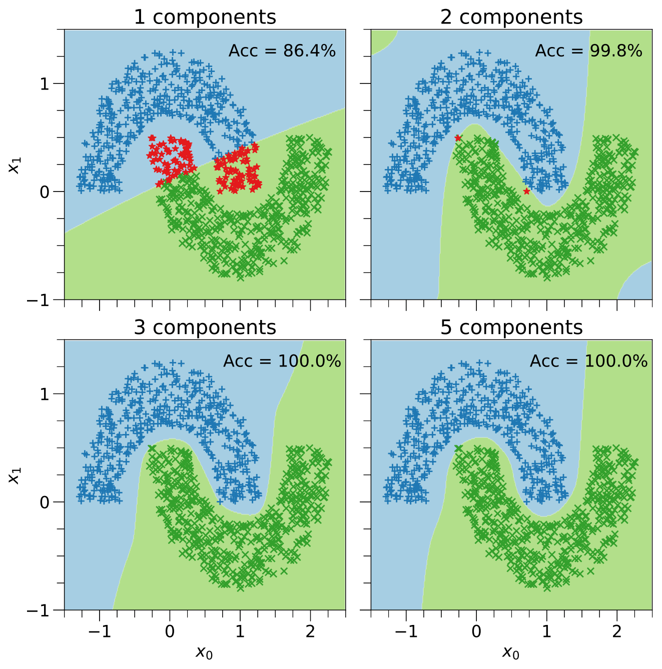

Fig. 7. Prediction for the testing set and the decision region for the various mixture models used.

References

[1] V. Moosavi, S. Packmann, and I. Vallés, “Sompy: A python library for self organizing map (som),” 2014, gitHub.[Online]. Available: https://github.com/sevamoo/SOMPY.

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