Michael Janov

EECE5644 Introduction to Machine Learning and Pattern Recognition

Professor Jennifer Dy; MW 2:50pm-4:30pm

December 14, 2017

Techniques for K-Means Clustering Initialization

Abstract

K-Means clustering is generally reported in the literature as a robust and simple-to-implement algorithm that nevertheless tends to fall into local minima when identifying class centroids [1]. While using randomized means is very effective and the easiest to implement [1], it is inelegant and performs inconsistently. Especially with particularly large data sets, this unstable performance is undesirable. Because of the usefulness and portability of K-Means clustering, a more predictable initialization method is desired. This paper evaluates two initialization techniques from the past five years, Linear Assignment Initialization by K.L. Cheng, et al., [2] and Initialization Based on Density by Q. Yuan, et al., [3] against the supervised data classification and randomized initializations. The Iris [4], Wine [5], and Glass [6] datasets from UCI's Machine Learning Repository as well as a simple two-class gaussian with identity covariance are used to evaluate the initialization techniques.

Final results for these initialization techniques show that Linear Assignment and Density both perform consistently, although their computation times to reach this result are considerably larger than randomized initialization scale. Against larger sets of data, the computation time gets exponentially larger than the random initialization. That said, both algorithms led to a consistent 90 percent accuracy clustering with the Iris data set while the randomized initialization ranged from 70 to 85 percent accuracy.

Table of Figures

Background

Optimizing initializations for k-means clustering is, fortunately, a popular problem to address by the machine learning community; there is a wide variety of algorithms going back at least twenty years and new algorithms are continuously being created. A 1999 paper by J.M. Peña, et al., evaluated three popular k-means initialization algorithms being used at the time, Forgy, MacQueen, and Kaufman, and evaluated their effectiveness against randomized starts. [1] They concluded that Kaufman and Randomized starts were nearly the same effectiveness while Forgy and MacQueen sat below Randomized in terms of tradeoff between computation time, added value, and robustness. These results were taken into consideration when beginning this project, such that Randomized starts might be considered more effective than existing algorithms due to this time-versus-added-value evaluation.

The two algorithms reimplemented and analyzed in this paper, Linear Assignment [2] and Density [3], are from conference proceedings in 2012 and 2015, respectively, and thus represent algorithms that are approximately 15 years newer than J.M. Peña's original survey paper. The four data sets used in this paper were selected to measure against a breadth of different data set parameters, namely sample size, number of attributes, and number of classes. This distribution can be found in Table 1 on the next page and all algorithms described in this paper are assumed to have a known number of classes prior to execution.

	SAMPLES	CLASSES	ATTRIBUTES	KEY PARAMETER
2GAUSS	1000	2	2	Samples
IRIS [4]	150	3	4	*Intermediate*
WINE [5]	178	3	13	Attributes
GLASS [6]	214	7	11	Classes

Table 1 - Parameters of data sets used

The supervised data from these data sets was imported and plotted in MATLAB, as seen in the figure below. Graphed attributes for each set remain consistent throughout this paper and were chosen because they represented the more-segmented attribute pairs. In the case of Glass data, most of the classes are highly overlapping and the RI: Refractive Index attribute was the attribute that separated them most. More details on the attributes and class list can be found on the appropriate UCI Machine Learning Database website found in the references at the end of this paper. It should be noted while that only two attributes are plotted, all attributes are considered when clustering.

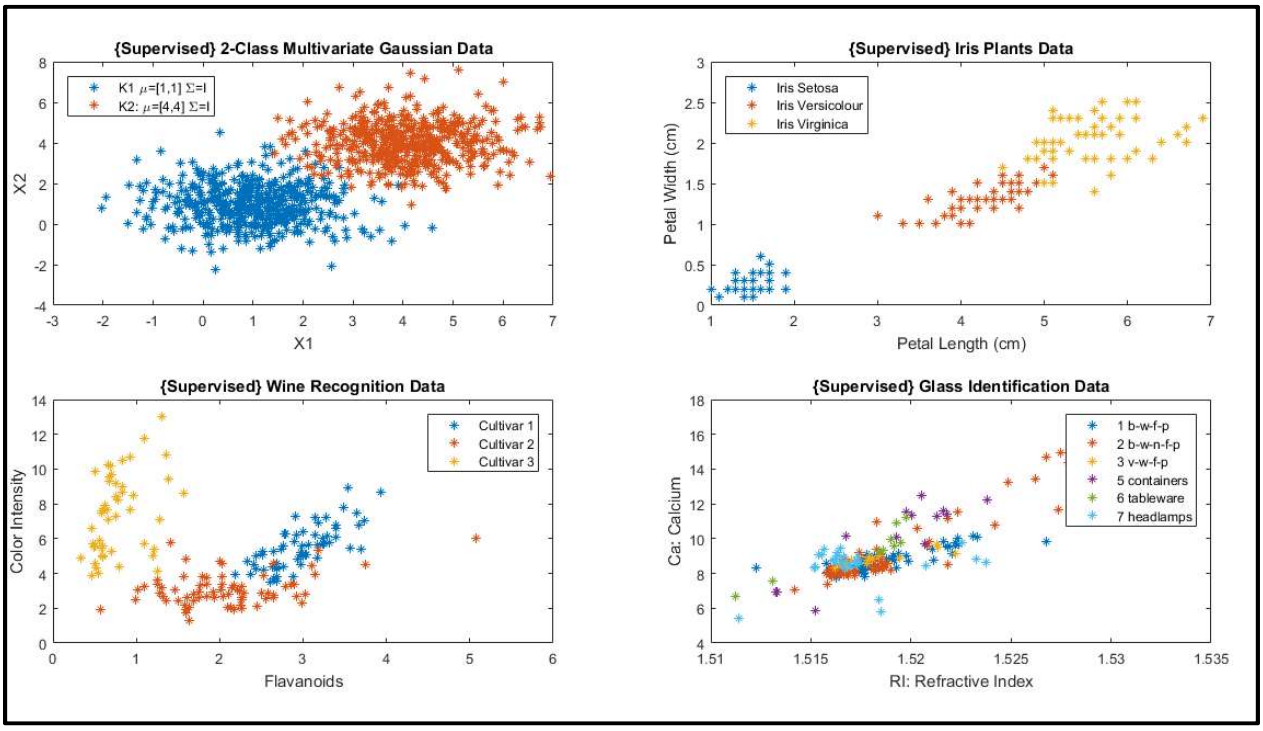

Figure 1 - Supervised data sets plotted in MATLAB

Algorithms

This section highlights the three algorithms analyzed in this paper, including randomized initialization. Below in Figure 2 is a simplified, plain-words description of the randomized start implementation. The number of random restarts used in this paper is ten, though this number can be modified depending on desired closeness to true result.

1. Randomly select a k data points to serve as class means
2. Assign label based on distance between data point and class mean
3. Store sum-squared-error of current solution
4. Repeat steps 1-3 ten times
5. Keep the solution with the lowest sum-squared-error

Figure 2 - Randomized Start generalized algorithm

Below in Figure 3 highlights the plain-words description of the Linear Assignment Initialization algorithm [2], of which the more detailed and clear-to-follow algorithm can be found in the original paper. The implementation for this algorithm was kept as authentic to the source as possible. The described cluster representatives are a sort of travelling pseudo-centroid based on existing sample data. These representatives eventually get superseded into non-sample-restricted true-means of each cluster.

1. Compute distance between each data point
2. Choose the two farthest points as the initial **cluster representatives**
3. For each additional class, choose the farthest point from currently-existing representatives
4. Classify data based on distance from representatives
5. Given current classifications, calculate the true cluster mean for each and reassign labels based on distance
6. Repeat 5 until convergence

Figure 3 - Linear Assignment Initialization algorithm [2]

Finally, the below Figure 4 highlights the plain-words description of the Density-Based Initialization algorithm [3]. The reimplementation used for this paper is not a fully-authentic recreation of the algorithm, which instead uses a Minimum Density Distance (MDD) metric to evaluate whether a point is truly in a highly-dense area or instead is a singleton. Additionally, the paper's described method for generating a local radius size was not performing adequately; the radius used by this paper is hard-tailored for the data sets being used and is likely to behave differently given different data sets or attributes.

1. Compute distance between each data point
2. Calculate local density at all data points
   - $density(x_i) = \sum_{j=1}^{n} exp(-\frac{d(x_i,x_j)^2}{2R^2})$
   - Where x_j indicates a point in the local radius
3. Choose highest density point as first cluster mean. Remove all points it its local radius from further consideration.
4. Repeat 3 until k means are reached
5. Assign labels based on distance
6. Given current classifications, calculate the true cluster mean for each and reassign labels based on distance
7. Repeat 6 until convergence

Figure 4 - Density-Based Initialization algorithm [3]

Results and Analysis

For the Gauss2 and Iris datasets, the algorithms worked well in correctly clustering data together. One pitfall, which is immediately evident in the Wine and Glass Accuracy data from Table 2 on the next page, is that accuracy is routinely extremely low. Observation of the raw data confirms that, while not exceptionally well-clustered, the low accuracy comes from the data being mislabeled, but consistently so. This is an issue that many papers do not address and instead opt to give Root-Square-Error or similar metrics because while a cluster might be correct, it is mislabeled simply because the clustering algorithm is agnostic of how classes should be named. In my code implementation, which is available at the GitHub link provided at the end of this paper, I use a novel label normalization function to shift labels around based on frequency. For instance, if the clear majority of data points are labelled incorrectly, but consistently, for a given class, the two labels are swapped and correct naming is reached while avoiding cheating by maintaining the incorrect classifications. This works very well for well-segmented data sets like Gauss2 and Iris, but fails on Wine and Glass, which have a considerable number of misclassifications and overlap and provide no clear indication as to the correct label.

ACCURACY (%)	GAUSS2	IRIS	WINE	GLASS
RANDOMIZED	98.70%	84.67%	71.35%*	2.80%*
LINEAR ASSIGNMENT	98.60%	89.33%	6.74%*	4.67%*
DENSITY	98.60%	89.33%	53.37%*	11.68*

Table 2 - Algorithm accuracy metrics

Computation time was also analyzed and the results of which can be seen in Table 3 and the closely-related number of convergences in Table 4. The algorithms were run on an Intel Core i7-4700MQ CPU at 2.40GHz to achieve these numbers, though the importance here lies in the relative values. The most important metric to observe is the disparity between the Gauss2 data set and the others. Gauss2 has approximately four times as many data points, but well over ten times as much computation time is necessary to complete, thus making sample size by far the most important parameter when optimizing for time as it is trends exponentially. The computation time for the non-random initializations is well higher than randomized start, but does not differ much between themselves despite Density having more convergence iterations on average. This suggests that the initial, common steps in both algorithms drive the computation time up. Most likely this is Step 1 for each, where the distances between all points are calculated and stored, which would correlate with the exploding computation time for increasing sample size.

TIME (MS)	GAUSS2	IRIS	WINE	GLASS
RANDOMIZED	44.8	8	9.9	15.6
LINEAR ASSIGNMENT	703.6	19.1	30.4	48.1
DENSITY	696.6	20.2	32.2	39.2

Table 3 - Algorithm computation time metrics

CONVERGENCE ITERATIONS	GAUSS2	IRIS	WINE	GLASS
RANDOMIZED	10	10	10	10
LINEAR ASSIGNMENT	2	3	7	6
DENSITY	6	8	12	5

Table 4 - Algorithm number of recursions to reach convergence

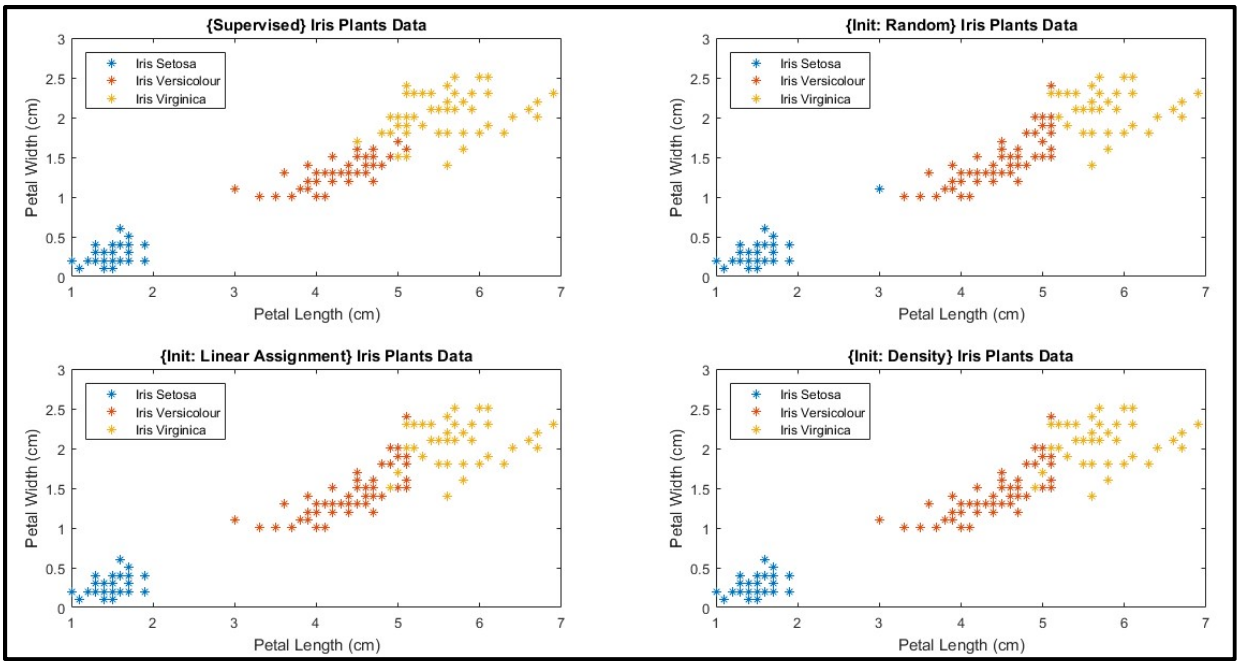

Figure 5 - Iris Plants Data across all algorithms and supervised labels

Looking now at a side-by-side graphical comparison for all three algorithms and the supervised labelling for the Iris data set in Figure 5, the resulting classification is clean and consistent throughout. There is one section for Iris Virginica (yellow) that is routinely misclassified for all algorithms, though this is likely because of the data point distributions in the other two attributes for this data set.

The next two pages contain graphs looking at algorithm performance for each data set--Figure 6 Randomized, Figure 7 Linear Assignment, and Figure 8 Density-Based. The main takeaway is the consistent performance for Gauss2 and Iris data sets, but generally sloppy classification for Wine and Glass, even if the label normalization function described previously (which is not used for the Wine and Glass data sets) worked as intended. The original researchers for these data sets had more success than me with these data sets, so perhaps a next step would be to be more picky when selecting attributes from those available.

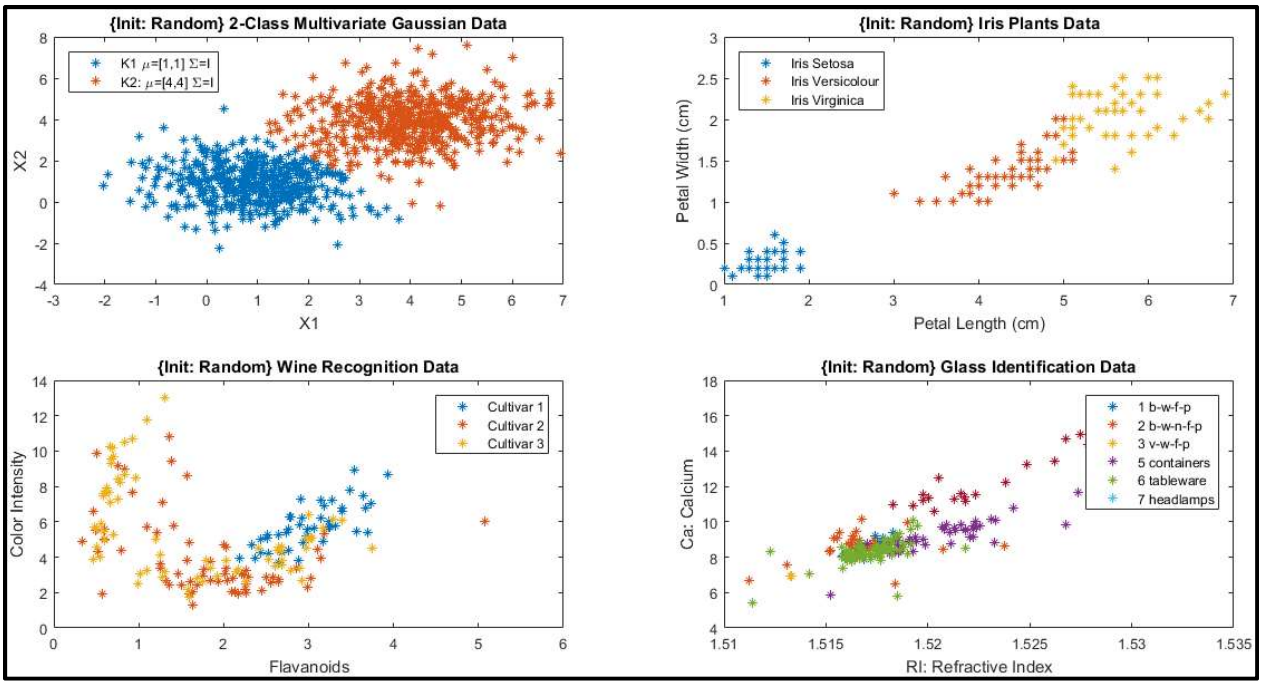

Figure 6 - Randomized Initialization graphs

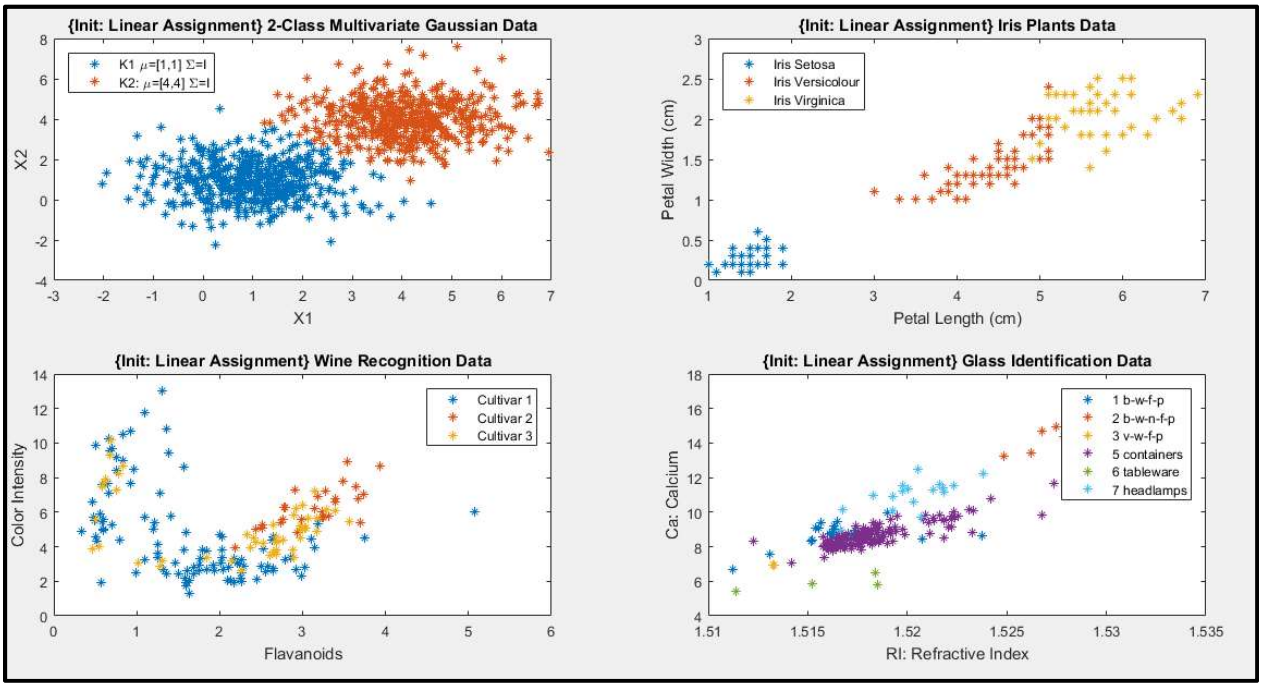

Figure 7 - Linear Assignment graphs

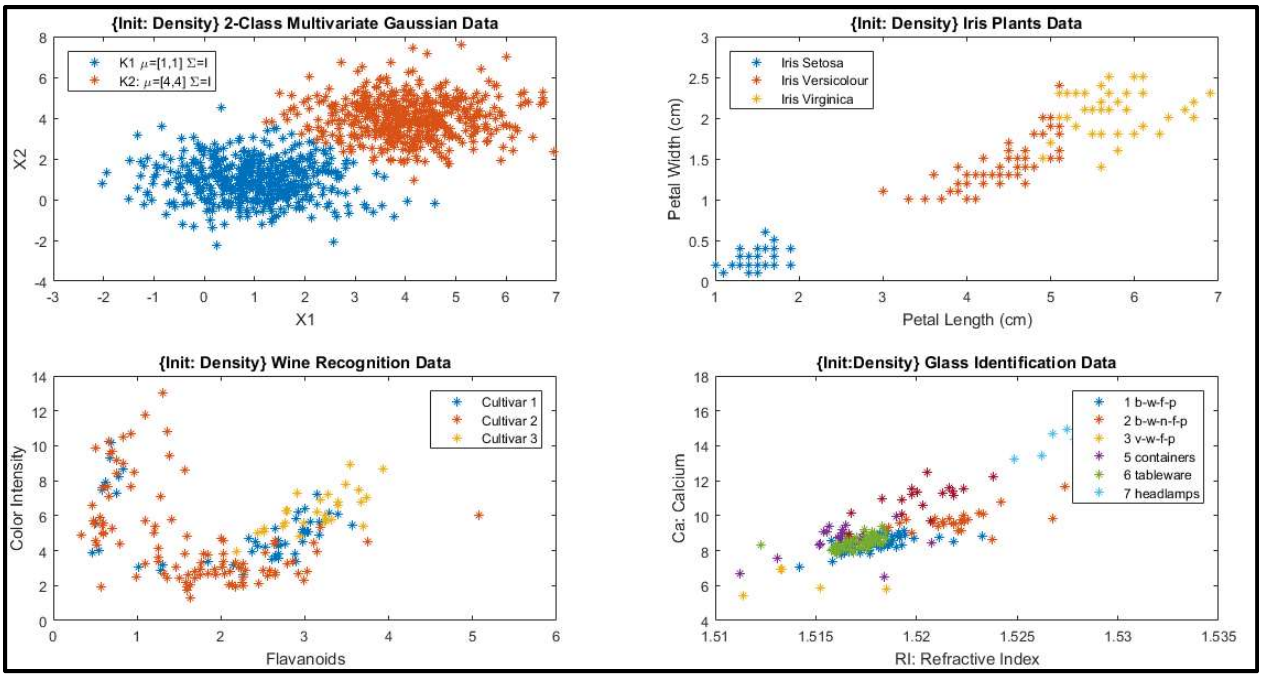

Figure 8 - Density-Based Initialization graphs

Conclusions

For much larger data sets (>1000 samples), randomized means initialization is still a viable alternative to more computationally-heavy logical initialization algorithms if time is the primary consideration. Otherwise, Linear Assignment is generally the best of the three in terms of consistency, accuracy, and robustness. For both logical initialization algorithms, the result was consistent and predictable each time, which could not nearly be said for randomized initialization. Density-based initialization has a quirk that makes it less desirable: the radius value used to determine local density is extremely sensitive to the answer of the solution. Too-large or too-small values will give wildly incorrect results and the paper-proposed method for finding this radius is inconsistent and unreliable.

The full MIT-licensed MATLAB code implementation for these algorithms is available on my public GitHub at https://github.com/mikejanov/k-means-clustering-initialization. Simply download the full directory, source all folders, and run the code from `main.m`. All data sets and configurations are auto-populated.

References

[1] J. L. J. &. L. P. Pena, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognition Letters,* pp. 1027-1040, 2010.

[2] J. F. a. J. W. K. L. Cheng, "A two-pass clustering algorithm based on linear assignment initialization and k-means method," in *5th International Symposium on Communications, Control, and Signal Processing*, Rome, 2012.

[3] H. S. a. X. Z. Q. Yuan, "An optimized initialization center K-means clustering algorithm based on density," in *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, Shenyang, 2015.

[4] R. Fisher, "UCI Machine Learning Repository: Iris Data Set," University of California, School of Information and Computer Science, Irvine, CA, 1936.

[5] M. e. a. Forina, "UCI Machine Learning Repository: Wine Data Set," University of California, School of Information and Computer Science, Irvine, CA, 1991.

[6] B. German., "UCI Machine Learning Repository: Glass Identification Data Set," University of California, School of Information and Computer Science, Irvine, CA, 1987.