01 May 2021

FuzzyART: An R Package for ART-based Clustering

Louis Steinmeister

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

FuzzyART: An R Package for ART-based Clustering.

Louis Steinmeister* Donald C. Wunsch II*

2021-06-09

Abstract

Adaptive Resonance Theory (ART) was introduced by Steven Grossberg as a theory of human cognitive information processing (Grossberg 1976, 1980). Extending the capabilities of the ART 1 model, which can learn to categorize patterns in binary data, fuzzy ART as described in (Carpenter, Grossberg, and Rosen 1991) has become one of the most commenly used Adaptive Resonance Theory models (Brito da Silva, Elnabarawy, and Wunsch 2019). By incorporating fuzzy set theroy operators, fuzzy ART is capable of learning from binaray and bounded real valued data. Its advantage over other unsupervised learning algorithms lies in the flexibility of the learning rule. If a given input feature does not resemble a known category satisfactorily, as determined by the vigilance test, a new category is initialized. Hence, the total number of categories (or clusters) is not determined a-priori, like k-means, but chosen in accordance with the data and the context of already learnt representations. This vignette explores the use of the fuzzy ART implementation as provided by the FuzzyART R package.

Contents

Packacke Usage

Feel free to use this package as specified by the license (MIT). However, please consider citing this work in any publication that this package may contribute to.

To install the package run

```
devtools::install_gitlab(repo = "acil-group/rFuzzyART", host = "git.mst.edu")
```

and run

```
library(FuzzyART)
```

to load the package for use.

Training

Before the fuzzy ART model can be trained, one needs to determine a minimum set of parameters: - rho: Vigilance parameter in (0,1). - alpha: Choice parameter alpha > 0. Can be viewed as a regularization

*Missouri University of Science and Technology, Applied Computational Intelligence Lab

parameter penalizing large weights. - beta: Learning rate in (0,1).

Input Pre-processing

Before training, it is important to remember scaling the inputs to lie in the d-dimensional unit hypercube ($[0, 1]^d$), where d is the dimension of the inputs. In other words, each input variable needs to be normalized to the interval $[0, 1]$. This can easily be done with the normalize() function.

```
library(FuzzyART)
library(mclust)
#> Warning: package 'mclust' was built under R version 4.1.0
#> Package 'mclust' version 5.4.7
#> Type 'citation("mclust")' for citing this R package in publications.
print("Original data:")
#> [1] "Original data:"
summary(iris)
#>   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
#>  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
#>  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
#>  Median :5.800   Median :3.000   Median :4.350   Median :1.300
#>  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
#>  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
#>  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
#>        Species
#>  setosa    :50
#>  versicolor:50
#>  virginica :50
#>
#>
#>
print("Normalized data:")
#> [1] "Normalized data:"
iris.normalized = normalize(df = subset(iris,select = -Species))
summary(iris.normalized)
#>   Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
#>  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
#>  1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333
#>  Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000
#>  Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.45806
#>  3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833
#>  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
```

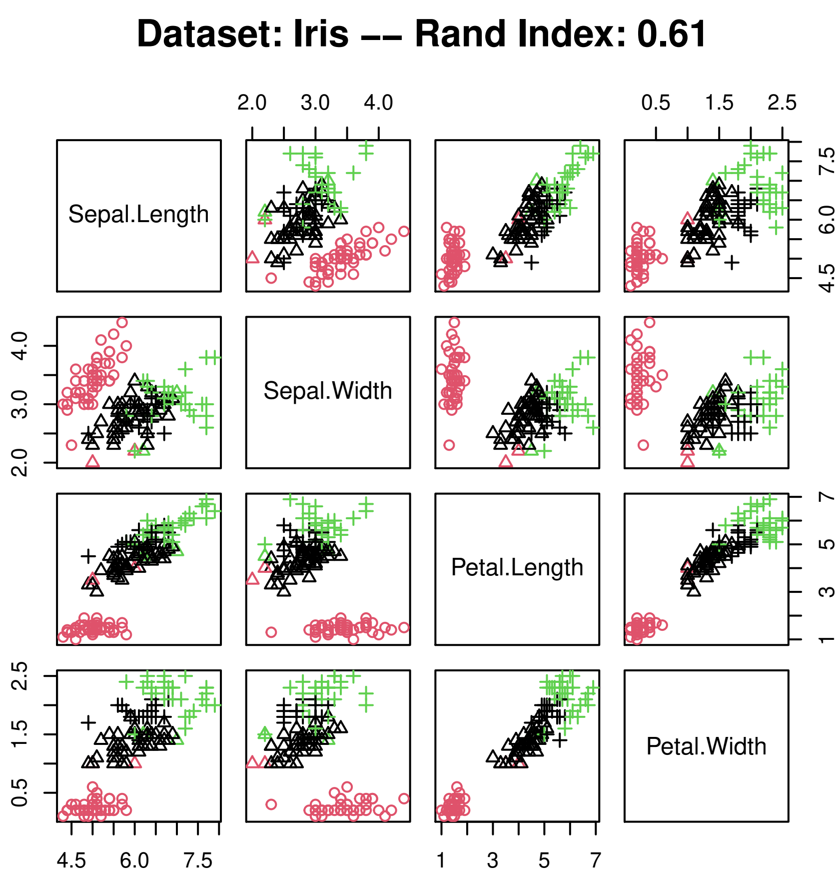
Training the ART model

In our Iris example we shall use parameters close to the ones used in (Hoa and Bui 2012); that is alpha $\approx$ 0.8, beta $\approx$ 0.1, rho $\approx$ 0.5. For the wine dataset, the parameters as specified in (Elnabarawy, Tauritz, and Wunsch 2017) appear superior. We will demonstrate to power of this implementation on a number of popular datasets.

The true membership of individual observations is indicated by the symbol while the color corresponds to the category membership according to our trained fuzzy ART model. The Rand Index measures the similarity between two sets of clustering partitions. In this case, we benchmark the performance of the unsupervised fuzzy ART model against the ground truth, the labels associated with each observation.

Iris

```
# load data
inputs = subset(iris,select = -Species)
labels.true = as.numeric(unlist(iris$Species))
normalized_inputs = normalize(df = inputs)
# train model
mod  = FuzzyART_train(normalized_inputs,alpha = .8,rho = .5,
                      beta = .12, max_epochs = 1000,max_clusters =20,
                      random_seed = 4, show_status = FALSE, beta_decay = .9)
plot(inputs, col = mod$Labels, pch = labels.true,
     main = paste0("Dataset: Iris -- Rand Index: ",
                   round(adjustedRandIndex(mod$Labels,labels.true),digits = 2)))
```

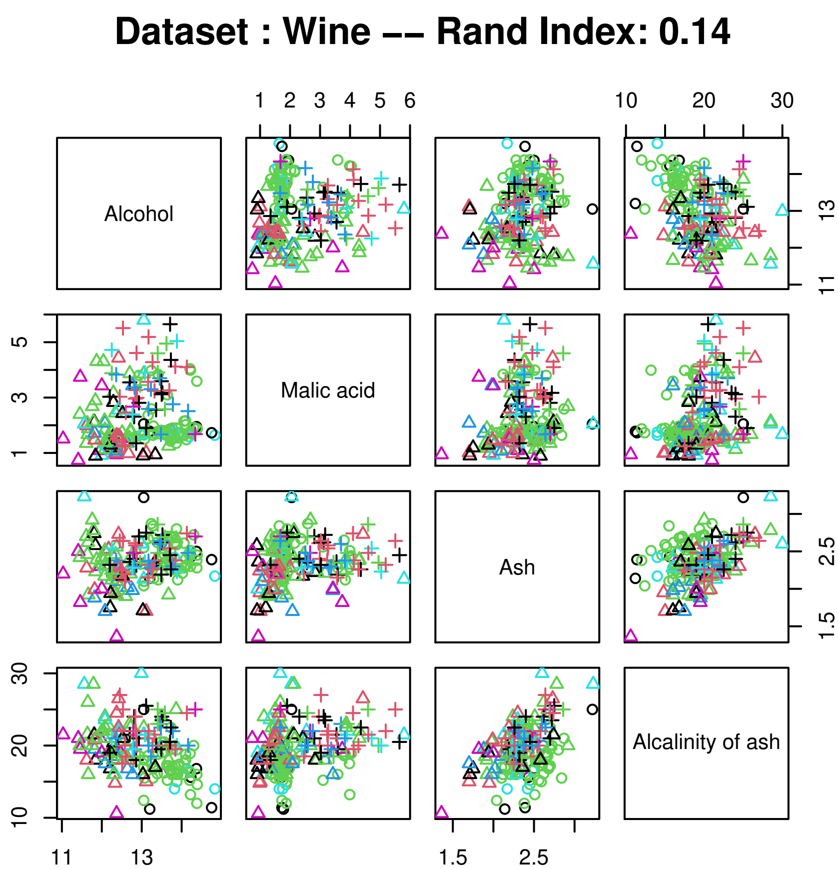


Wine

Note that for a better presentation, we will only be displaying the first four features of the wine dataset. However, all features were used during training.

```
# load the wine dataset
wine.address <- "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
wine <- read.csv(wine.address,header = FALSE)
wine.colnames = c("Label","Alcohol", "Malic acid", "Ash", "Alcalinity of ash", "Magnesium",
"Total phenols", "Flavanoids", "Nonflavanoid phenols", "Proanthocyanins",
```

```
"Color intensity", "Hue", "OD280/OD315 of diluted wines", "Proline")
colnames(wine)<-wine.colnames
# prepare model inputs
inputs = subset(wine,select = -Label)
labels.true = wine$Label
normalized_inputs = normalize(df = inputs)
# train the model
mod  = FuzzyART_train(normalized_inputs,alpha = .8679,rho = .375,
                      beta = .9797,max_epochs = 2000,max_clusters =20,
                      random_seed = 4,show_status = FALSE, beta_decay = .9)
plot(inputs[1:4], col = mod$Labels, pch = as.numeric(labels.true),
     main = paste0("Dataset : Wine -- Rand Index: ",
                   round(adjustedRandIndex(mod$Labels,labels.true),digits = 2)))
```



Evaluation of package performance

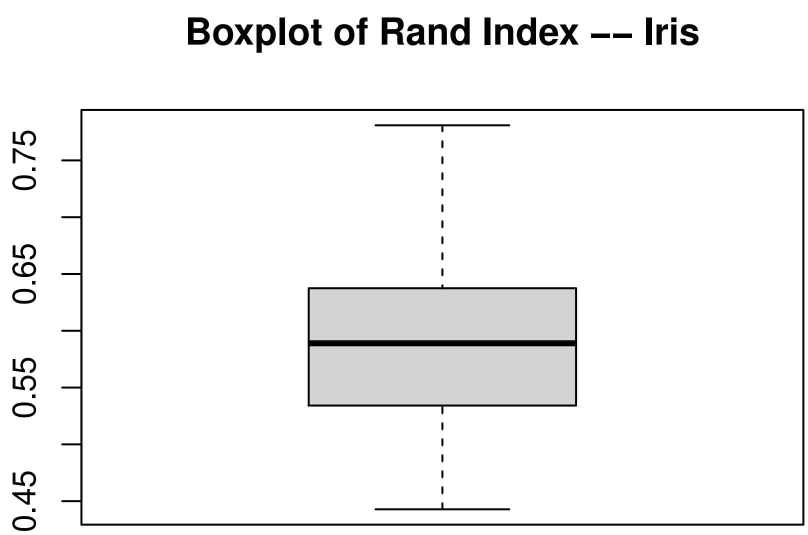
The examples as shown here certainly have further potential for fine tuning. Nevertheless, in reference to results achieved in (Illetskova et al. 2019), the achievable performance as demonstrated here appear to be on par with the baseline algorithm (as described in the cited work). This can be verified via:

```
#Iris
inputs = subset(iris,select = -Species)
labels.true = as.numeric(unlist(iris$Species))
```

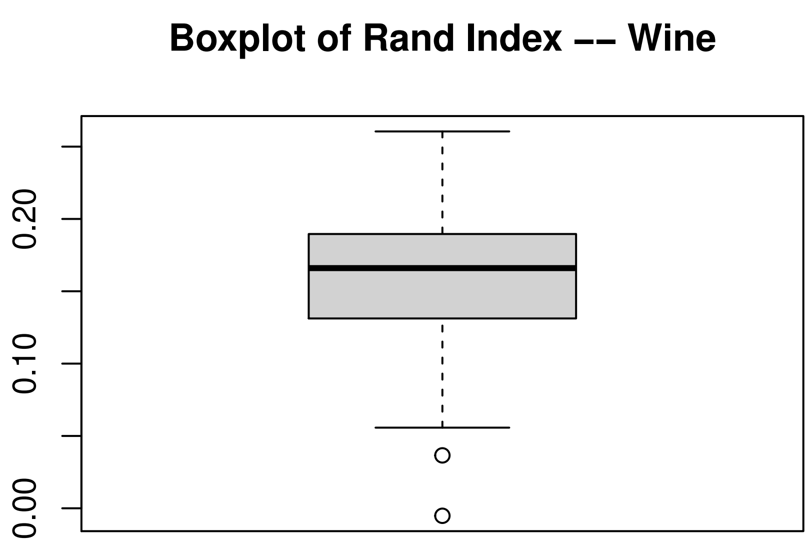
```r
normalized_inputs = normalize(df = inputs)
test_iris = function(seed)
{
  mod  = FuzzyART_train(normalized_inputs,alpha = .8,rho = .5,
                        beta = .12,max_epochs = 1000,max_clusters =20,
                        eps = 10^-8,random_seed = seed, show_status = FALSE,
                        beta_decay = .9)
  return(adjustedRandIndex(mod$Labels,labels.true))
}
res_iris = sapply(X = 1:50,FUN = test_iris)
print(paste0("Mean: ", mean(res_iris)))
print(paste0("StD: ",sqrt(var(res_iris))))
boxplot(res_iris, main = "Boxplot of Rand Index -- Iris")


#Wine
inputs = subset(wine,select = -Label)
labels.true = wine$Label
normalized_inputs = normalize(df = inputs)
test_wine = function(seed)
{
  mod  = FuzzyART_train(normalized_inputs,alpha = .8679,rho = .375,
                        beta = .9797,max_epochs = 2000,max_clusters =20,
                        eps = 10^-8,random_seed = seed,show_status = FALSE,
                        beta_decay = .9)
  return(adjustedRandIndex(mod$Labels,labels.true))
}
res_wine = sapply(X = 1:50,FUN = test_wine)
print(paste0("Mean: ", mean(res_wine)))
print(paste0("StD: ",sqrt(var(res_wine))))
boxplot(res_wine, main = "Boxplot of Rand Index -- Wine")
```

```
#> [1] "Mean: 0.588421981887455"
#> [1] "StD: 0.0825407082831491"
#> Registered S3 method overwritten by 'GGally':
#>   method from
#>   +.gg   ggplot2
#> Registered S3 method overwritten by 'sets':
#>   method        from
#>   print.element ggplot2
#> Warning: replacing previous import 'GGally::%>%' by 'sets::%>%' when loading
#> 'bootcluster'
```




```
#> [1] "Mean: 0.157224686186454"
#> [1] "StD: 0.0490131046535643"
```



References

Brito da Silva, Leonardo Enzo, Islam Elnabarawy, and Donald C. Wunsch. 2019. "A survey of adaptive resonance theory neural network models for engineering applications." *Neural Networks* 120 (December): 167–203. https://doi.org/10.1016/j.neunet.2019.09.012.

Carpenter, Gail A., Stephen Grossberg, and David B. Rosen. 1991. "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system." *Neural Networks* 4 (6): 759–71. https://doi.org/10.1016/0893-6080(91)90056-B.

Elnabarawy, Islam, Daniel R. Tauritz, and Donald C. Wunsch. 2017. "Evolutionary computation for the automated design of category functions for fuzzy ART." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1133–40. ACM. https://doi.org/10.1145/3067695.3082056.

Grossberg, Stephen. 1976. "Adaptive Pattern Classification and Universal Recoding: II. Feedback, Expectation, Olfaction, Illusions." *Biological Cybernetics* 23 (4): 187–202. https://doi.org/10.1007/BF00340335.

———. 1980. "How does a brain build a cognitive code?" *Psychological Review* 87 (1): 1–51. https://doi.org/10.1037/0033-295X.87.1.1.

Hoa, Nong Thi, and The Duy Bui. 2012. "A New Effective Learning Rule of Fuzzy ART." In *2012 Conference on Technologies and Applications of Artificial Intelligence*, 224–31. IEEE. https://doi.org/10.1109/TAAI.2012.60.

Illetskova, Marketa, Islam Elnabarawy, Leonardo Enzo Brito da Silva, Daniel R. Tauritz, and Donald C. Wunsch. 2019. "Nested monte carlo search expression discovery for the automated design of fuzzy ART category choice functions." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 171–72. 1. ACM. https://doi.org/10.1145/3319619.3322050.