About the method - cla\_mlp: Multilayer Perceptron (feedforward neural network) for classification. - Hyperparameters: size (hidden layer units), decay (L2 regularization to reduce overfitting).

Environment setup.

# Classification using Multilayer Perceptron  
  
# installation   
#install.packages("daltoolbox")  
  
# loading DAL  
library(daltoolbox)

Load data and inspect.

iris <- datasets::iris  
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

Target Species levels.

# extracting the levels for the dataset  
slevels <- levels(iris$Species)  
slevels

## [1] "setosa" "versicolor" "virginica"

Building train and test samples via random sampling Random train/test split.

# Building train and test samples via random sampling  
set.seed(1)  
sr <- sample\_random()  
sr <- train\_test(sr, iris)  
iris\_train <- sr$train  
iris\_test <- sr$test

Class distribution by split.

tbl <- rbind(table(iris[,"Species"]),   
 table(iris\_train[,"Species"]),   
 table(iris\_test[,"Species"]))  
rownames(tbl) <- c("dataset", "training", "test")  
head(tbl)

## setosa versicolor virginica  
## dataset 50 50 50  
## training 39 38 43  
## test 11 12 7

Model training Train MLP: set hidden layer size and regularization.

# Model training  
model <- cla\_mlp("Species", slevels, size=3, decay=0.03)  
model <- fit(model, iris\_train)

Training evaluation

# Checking fit on training data  
train\_prediction <- predict(model, iris\_train)  
  
# Model evaluation (training)  
iris\_train\_predictand <- adjust\_class\_label(iris\_train[,"Species"])  
train\_eval <- evaluate(model, iris\_train\_predictand, train\_prediction)  
print(train\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 0.9833333 39 81 0 0 1 1 1 1 1

Test evaluation

# Model test  
test\_prediction <- predict(model, iris\_test)  
  
iris\_test\_predictand <- adjust\_class\_label(iris\_test[,"Species"])  
  
# Test evaluation  
 test\_eval <- evaluate(model, iris\_test\_predictand, test\_prediction)  
print(test\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 0.9666667 11 19 0 0 1 1 1 1 1

References - Rumelhart, D., Hinton, G., Williams, R. (1986). Learning representations by back-propagating errors. - Bishop, C. M. (1995). Neural Networks for Pattern Recognition.