Artificial Neural Networks (RWeka)

CIS400/600 Fundamentals of Data and Knowledge Mining

1. Install and load `RWeka` library

   ```
   install.packages("RWeka")
   library(RWeka)
   ```

2. `RWeka` has a list of classifiers that can be checked using

   ```
   ?Weka_classifiers
   ```

3. There are five standard interface functions viz.

   (a) `Weka_classifier_functions`

   (b) `Weka_classifier_lazy`

   (c) `Weka_classifier_meta`

   (d) `Weka_classifier_rules`

   (e) `Weka_classifier_trees`

   None of these five interfaces have **Artificial Neural Networks** function implemented. But, **Weka** has Artificial Neural Networks classifier among the list of classifiers that are implemented. ANNs exist in `weka.classifiers.functions.MultilayerPerceptron`

4. Let us create the classifier by importing the same

   ```
   md <-
       make_Weka_classifier("weka/classifiers/functions/MultilayerPerceptron")
   ```

`make_Weka_classifier` method helps in creating user defined classifiers

5. Load **IRIS** dataset (which is provided along with lab exercise)

```
iris_data <- read.csv("iris.csv")
```

6. Splitting the dataset into training and test sets. Let us split the dataset as 80% training and 20% test data. Let us use random sampling without replacement. There are total of 150 records, 20% of 150 records is 30 records

```
# indexes of rows that are selected for training and testing
test_set_indexes <- sample(1:nrow(iris_data), 30)
train_set_indexes <- setdiff(1:nrow(iris_data),
    test_set_indexes)

# fetching rows with above indexes
test_data <- iris_data[test_set_indexes, ]
train_data <-iris_data[train_set_indexes, ]
```

7. Classifiers in **RWeka** take `formula` as input. These formulas indicate the variables on which model gets trained and the necessary class variable. Class variable in IRIS dataset is `species`. Model gets trained on attributes `sepal_length, sepal_width, petal_length, petal_width`. The formula is

```
iris_data$species ~ iris_data$sepal_length +
    iris_data$sepal_width + iris_data$petal_length +
    iris_data$petal_width
```

   It can be simply written as

```
iris_data$species ~ .
```

   More details about formula can be found at `https://faculty.chicagobooth.edu/richard.hahn/teaching/FormulaNotation.pdf`

8. Training the model on training dataset

```
trained_model <- md(iris_data$species ~ ., data = iris_data,
    subset <- train_set_indexes, control = Weka_control(G
    = T))
```

`subset` helps to train the model `md` on training set which is provided as input as `train_set_indexes`
To output GUI, `control = Weka_control(G = T)` is enabled. Other options for `control` can be found at

```
WOW(md)
```

9. View the trained model

```
trained_model
```

10. Test the (trained) model on test dataset

```
test_results <- predict(trained_model, newdata =
    test_data[1:4])
```

`newdata <- test data[1:4]` ensures that the actual output or test in-stances is not used while predicting output of test instances

11. Printing results

```
# prediction results
test_results

# actual test dataset results
test_data[ ,5]
```

12. Printing confusion matrix

```
table(test_results, test_data[ ,5], dnn = list("predicted",
    "actual"))
```

13. For the purpose of model selection, **Cross-validation** can also be used rather than **test-train split** which is used above. Cross-validation can

be implemented as

```
evaluate_Weka_classifier(trained_model, class = TRUE,
    numFolds = 10)
```