Machine Learning in R and Other Languages

Introduction

Alright, we need to talk about machine learning in R. There is great need for machine learning these days, and everyone is trying to enter into the scene with their favorite languages. So, how does R fare compared to other languages? If machine learning is the future of all technologies, how is R going to do in the future? In this post, I will compare R and Python and how they do in the world of machine learning. And finally, I will show code examples of machine learning in R.

R and Python

We gotta talk about the background of each language a little bit. My last post was about just differences between R and Python in general, so I will put a short summary of the differences.

Python:

- Simple syntax and structure, meaning easy learning curve.
- Very versatile and general-purpose.
- Python's data analysis libraries are developing rapidly.
- Graphing in Python is not bad. There are some rough spots in their popular graphing libraries.

R:

- Written by hard-core statisticians.
- Not really optimized for speed or efficiency, but have many libraries.
- Statistical research is made easy in R.
- Graphing in R is pretty easy and sophisticated.

Ok, so these are some basic differences between the two languages. Now let's actually get to our topic of machine learning. It is generally agreed upon that Python is ahead of most other programming languages in terms of availability and popularity of machine learning libraries like Tensorflow, scikit-learn, and others. And yes, R is catching up a lot in implementing ways to accommodate a lot of these libraries, but still, there's long ways to go. The community of machine learning experts in R is just much smaller than that of Python or Java, which are just conventional programming languages for the tech industry. This makes sense, because it is true that more machine learning experts like to stick to their favorite programming languages, rather than venture into a language like R.

Still, it is true that a solid background in mathematics and statistics is necessary for anybody trying to do machine learning, so this may be why R is doing pretty well in the world of machine learning. One website reports that R is the 3rd most popular language used for machine learning, surveyed from the industry's experts. And there is a community of "believers" in R machine learning who really appreciates some of the ways that R makes manipulating rows and tables of data easy. So the future of machine learning in R doesn't seem to be too bad, either.

Examples

Let's dive into some examples of machine learning in R (Reference 4). We are going to use a popular machine learning library in R called Caret. So install the library along with a library called 'e1071' with the following and load the library.

```
#install.packages("caret", repos="http://cran.rstudio.com/")
#install.packages('e1071', dependencies=TRUE)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2017c.
## 1.0/zoneinfo/America/Los_Angeles'
```

And now we will bring in the famous iris dataset and create partitions of the data so that we can train our models with 80% of the data and test with the rest. Coming from mainly doing machine learning in Python, I must say that this is pretty elegant coding.

```
data(iris)
dataset <- iris
validation_index <- createDataPartition(dataset$Species, p=0.80, list=FALSE)
validation <- dataset[-validation_index,]
dataset <- dataset[validation_index,]
```

And now, we are going to set up a variable for cross-validation and make five different models that are widely used. And I will print the accuracy of each. (Ignore the Kappa table and other print output that appears in between code chunks.)

```
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"

# linear algorithms
set.seed(7)
fit.lda <- train(Species~., data=dataset, method="lda", metric=metric, trControl=control)

# nonlinear algorithms

# CART
set.seed(7)
fit.cart <- train(Species~., data=dataset, method="rpart", metric=metric, trControl=control)

# kNN
set.seed(7)
fit.knn <- train(Species~., data=dataset, method="knn", metric=metric, trControl=control)

# advanced algorithms

# SVM
set.seed(7)
fit.svm <- train(Species~., data=dataset, method="svmRadial", metric=metric, trControl=control)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
# Random Forest
set.seed(7)
fit.rf <- train(Species~., data=dataset, method="rf", metric=metric, trControl=control)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
## lda  0.9166667 1.0000000 1.0000000 0.9833333       1    1    0
## cart 0.8333333 0.9166667 0.9166667 0.9333333       1    1    0
## knn  0.8333333 0.9375000 1.0000000 0.9666667       1    1    0
## svm  0.8333333 0.9166667 0.9583333 0.9416667       1    1    0
## rf   0.8333333 0.9375000 1.0000000 0.9583333       1    1    0
##
## Kappa
##       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
## lda  0.875 1.00000 1.0000 0.9750       1    1    0
## cart 0.750 0.87500 0.8750 0.9000       1    1    0
## knn  0.750 0.90625 1.0000 0.9500       1    1    0
## svm  0.750 0.87500 0.9375 0.9125       1    1    0
## rf   0.750 0.90625 1.0000 0.9375       1    1    0
```

Before going into this kind of machine learning in R, I was expecting long unnecessary syntax and hard-to-understand function logic; however, going through some of the code myself, I am convinced that machine learning in R is up to the industry standards and that it's very easy for even beginners to use.

Conclusions

Even though other languages are more popular for machine learning, R has a lot to offer to the industry as well. It's actually catching up to the standards of other languages and in some ways, and it supports more elegant code syntax for applying statistics models to your data which adds to the language's value. R will most likely continue to excel in machine learning in the near future.

References

1. https://fossbytes.com/popular-top-programming-languages-machine-learning-data-science/
2. https://tensorflow.rstudio.com
3. https://en.wikipedia.org/wiki/TensorFlow
4. https://machinelearningmastery.com/machine-learning-in-r-step-by-step/
5. https://www.forbes.com/sites/adelynzhou/2017/11/21/key-qualities-to-look-for-in-ai-and-machine-learning-experts/#14338c365e8a
6. https://www.datacamp.com/community/tutorials/machine-learning-in-r
7. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/