Decision Trees and Random Forests

Shaurya Jauhari, Mora Lab, GMU. (Email: shauryajauhari@gzhmu.edu.cn)

2019-06-11

Welcome to the second workshop in the series of **Machine Learning Fundamentals**. In this session, we shall explore the theory of **Decision Trees** and scale it to the broader concept of **Random Forests**. Decision trees epitomize the divide and conquer strategy to accomplish classification tasks (although they could also be implemented for regression chores as well) (Quinlan 1986, Rokach and Maimon (2005)). From the previous session on **Logistic Regression**, you may recall that if the response variable is catergorical in nature (factors in R), we aim classification, or else, if the response variable is continuous, it denotes regression. Classification and regression methodologies are both categorized under supervised machine learning. The theme of machine learning was realized in 1950s and since then, coupled with the data deluge and upheaval in computational prowess, has exhibited stronghold in data analysis domain (See Figure 1).

Random forests enlighten on the dogma of democracy, i.e. *majority wins*. Decision Trees are rudimentary classification algorithms that, at a low-level, are synonymous to *if-then* conditional statements in programming languages. They follow the strategy of iterative recursion, and intuitively the leaf nodes hold the final verdict. The highest aggregate from all leaf nodes (terminals) is graded as the output of that decision tree.

In this module, we shall delve into creation of basic decision trees to have an understanding of it. For the purpose, we shall load the package **party** and make use of the function **ctree()** to calculate and analyze decision trees.

```
install.packages("party", dependencies = TRUE, repos = "https://mirrors.tuna.tsinghua.edu.cn/CRAN/")
```

```
The downloaded binary packages are in
    /var/folders/hm/c3_fjypn62v5xh5b5ygv267m0000gn/T//RtmpmkhzfC/downloaded_packages
```

```
library(party)
```

```
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: 'zoo'
The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
Loading required package: sandwich
```

Dataset Profile

Let us now, pick up a dataset. The dataset pertains to the soft computation of the **enhancer prediction** module in bioinformatics. We certainly take into cognizance the biological implications about enhancer

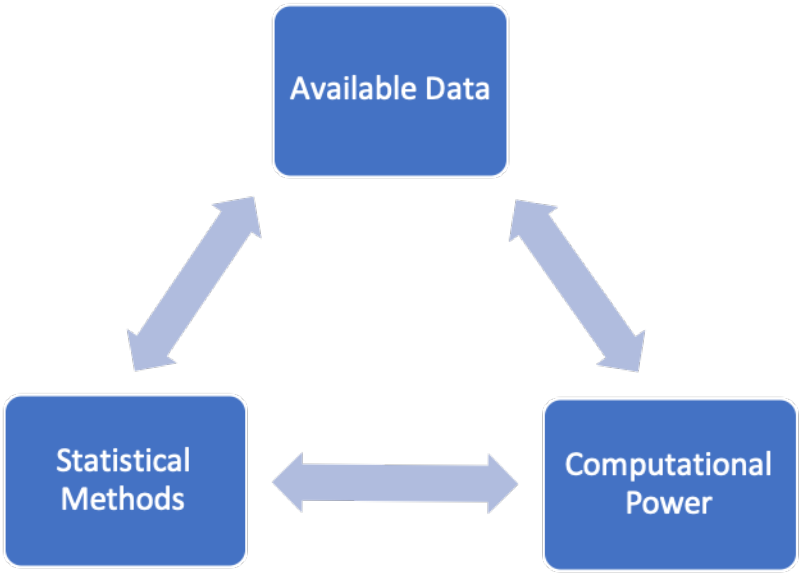


Figure 1: Machine Learning Genesis

regions (See Figure 2).
Certain known classes of proteins/ cis-regulatory elements called Transcription Factors (TFs) and Transcription Co-Activators (TCoAs) are programmed to bind to regions in the genome called *Enhancers*, that remotely orchestrate the phenomena of gene regulation. They are at a distal location to the *Promoters*, regions associated with genes and respective Transcription Start Sites (TSS). On stimulus from TFs, the enhancer and promoter sequences reciprocate and actuate the transcription machinery.

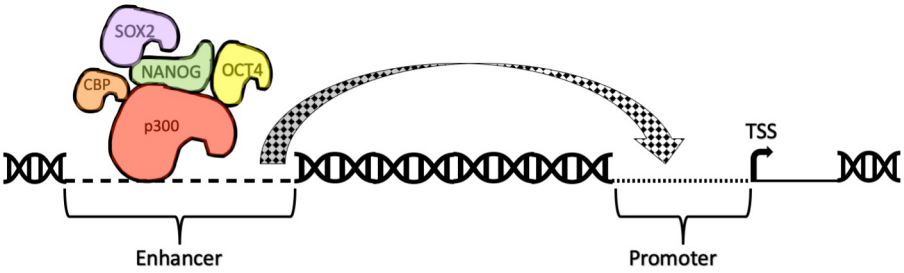
For use in this session, we have chosen the dataset from the study, "Genome-wide mapping of p300 binding in resting and activated human naïve and memory CD8 T cells", sample dataset deposited to GEO labeled GSM2445787. The dataset is originally available as a WIG file. At first, it is converted to BED and is then subsequently cleaned for our use as illustrated in the following code[1].

```
mydata_enhancers <- read.csv("./GSM2445787.bed", sep = '\t', header = FALSE)

## We choose to incorporate only those peaks with occurence of 1 and above. Additionally, a column for

mydata_enhancers <- mydata_enhancers[mydata_enhancers$V5 > 0,]
mydata_enhancers <- mydata_enhancers[,c(1,2,3)]
mydata_enhancers$Class <- "Enhancer"
colnames(mydata_enhancers) <- c("Chrom", "Start", "End", "Class")
```

[1] See Notes, 3



To add to the negative class data, we consider regions that are non-enhancers. Under this category, we include some random background data as well as equal number of TSS. The reason for soing such is because enhancer sites are lower in number in comparison to the promoter sites (Rajagopal et al. 2013). So, in order to normalize our example dataset we plan such a structuring.

Notes:
1. The authors (Rajagopal et al. 2013) construed p300 (a transcription co-activator) binding sites overlapping DNase-I hypersentitive sites and distal to annotated transcription start sites (TSS) as active p300 binding sites representative of enhancers. 2. Clusters with presence or absence of H3K36me3 were hypothesized to represent genic and inter-genic enhancers respectively. 3. While pre-processing the genomic ranges data if you need to convert WIG file to BED file, you may want to refer to BEDOPS -> wig2bed() function. This is available as a command line utility.

P.S. During intermediary partitioning, if the node has the lowest Gini Index, it becomes leaf node. That will most likely be the case when all remaining non-root nodes have been exhausted, checking for impurity score.

All workshop study material is available at my github page (Jauhari, n.d.).

References

Jauhari, Shaurya. n.d. "My Github Repository." https://github.com/shauryajauhari/Machine-Learning.

Quinlan, J R. 1986. "Induction of Decision Trees." *Machine Learning* 1: 81–106.

Rajagopal, Nisha, Wei Xie, Yan Li, Uli Wagner, Wei Wang, John Stamatoyannopoulos, Jason Ernst, Manolis Kellis, and Bing Ren. 2013. "RFECS: A Random-Forest Based Algorithm for Enhancer Identification from Chromatin State." *PLoS Computational Biology* 9 (3). doi:10.1371/journal.pcbi.1002968.

Rokach, Lior, and Oded Maimon. 2005. "Decision Tree." *Data Mining and Knowledge Discovery Handbook*, pp 165–92. doi:10.1007/978-0-387-09823-4_9.