MARS, or Multivariate Adaptive Regression Splines, constructs a linear combination of basis functions for logistic regression.

The required packages for this chapter include:

```r
library(rattle)  # The weather dataset and normVarNames().
library(randomForest)  # Impute missing values using na.roughfix().
library(dplyr)  # Data munging: tbl_df(), %>%.
library(ROCR)  # Use prediction() to convert to measures.
library(earth)  # An implementation of mars.
```

As we work through this chapter, new R commands will be introduced. Be sure to review the command’s documentation and understand what the command does. You can ask for help using the `?` command as in:

```r
?read.csv
```

We can obtain documentation on a particular package using the `help=` option of `library()`:

```r
library(help=rattle)
```

This chapter is intended to be hands on. To learn effectively, you are encouraged to have R running (e.g., RStudio) and to run all the commands as they appear here. Check that you get the same output, and you understand the output. Try some variations. Explore.
1 Data Preparation—Load and Configure

We use the \texttt{weather} dataset from \texttt{rattle} (Williams, 2014) to illustrate. Refer to Chapter Data for details.

\begin{verbatim}
library(rattle) # Provides \texttt{weather} and \texttt{normVarNames()}. library(dplyr) # Provides \texttt{\%\%} and \texttt{tbl_df()}. dsname <- "weather" ds <- get(dsname) %>% tbl_df() names(ds) <- normVarNames(names(ds)) vars <- names(ds) target <- "rain_tomorrow" risk <- "risk_mm" id <- c("date", "location")

ds
## Source: local data frame [366 x 24]
##
## date location min_temp max_temp rainfall evaporation sunshine
## 1 2007-11-01 Canberra 8.0 24.3 0.0 3.4 6.3
## 2 2007-11-02 Canberra 14.0 26.9 3.6 4.4 9.7
## 3 2007-11-03 Canberra 13.7 23.4 3.6 5.8 3.3
## 4 2007-11-04 Canberra 13.3 15.5 39.8 7.2 9.1
## 5 2007-11-05 Canberra 7.6 16.1 2.8 5.6 10.6
## 6 2007-11-06 Canberra 6.2 16.9 0.0 5.8 8.2
## 7 2007-11-07 Canberra 6.1 18.2 0.2 4.2 8.4
## 8 2007-11-08 Canberra 8.3 17.0 0.0 5.6 4.6
## 9 2007-11-09 Canberra 8.8 19.5 0.0 4.0 4.1
## 10 2007-11-10 Canberra 8.4 22.8 16.2 5.4 7.7
## .. ... ... ... ... ... ...
## Variables not shown: wind_gust_dir (fctr), wind_gust_speed (dbl),
## wind_dir_9am (fctr), wind_dir_3pm (fctr), wind_speed_9am (dbl),
## wind_speed_3pm (dbl), humidity_9am (int), humidity_3pm (int),
## pressure_9am (dbl), pressure_3pm (dbl), cloud_9am (int), cloud_3pm
## (int), temp_9am (dbl), temp_3pm (dbl), rain_today (fctr), risk_mm (dbl),
## rain_tomorrow (fctr)
\end{verbatim}

Copyright © 2013-2014 Graham@togaware.com
2 Data Preparation—Variables to Ignore

Here we identify variables that we probably do not want to play a part in the modelling.

```r
# Ignore the IDs and the risk variable.
ignore <- union(id, if (exists("risk")) risk)

# Ignore variables that look like identifiers.
ids <- which(sapply(ds, function(x) length(unique(x))) == nrow(ds))
ignore <- union(ignore, names(ids))

# Ignore variables which are completely missing.
mvc <- sapply(ds[vars], function(x) sum(is.na(x))) # Missing value count.
mvn <- names(ds)[(which(mvc == nrow(ds)))] # Missing var names.
ignore <- union(ignore, mvn)

# Ignore variables that are mostly missing – e.g., 70% or more missing
mvn <- names(ds)[(which(mvc >= 0.7*nrow(ds)))]
ignore <- union(ignore, mvn)

# Ignore variables with many levels.
factors <- which(sapply(ds[vars], is.factor))
lvls <- sapply(factors, function(x) length(levels(ds[[x]])))
many <- names(which(lvls > 20)) # Factors with too many levels.
ignore <- union(ignore, many)

# Ignore constants.
constants <- names(which(sapply(ds[vars], function(x) all(x == x[1L]))))
ignore <- union(ignore, constants)

# Initialise the variables
vars <- setdiff(vars, ignore)
```

```r
vars
## [1] "min_temp"   "max_temp"   "rainfall"
## [4] "evaporation" "sunshine"   "wind_gust_dir"
## [7] "wind_gust_speed" "wind_dir_9am" "wind_dir_3pm"
## [10] "wind_speed_9am" "wind_speed_3pm" "humidity_9am"
## [13] "humidity_3pm"  "pressure_9am"  "pressure_3pm"
## [16] "cloud_9am"    "cloud_3pm"    "temp_9am"
## [19] "temp_3pm"     "rain_today"   "rain_tomorrow"
```

```r
ignore
## [1] "date"  "location" "risk_mm"
```
3 Data Preparation—Clean and Finalise

The dataset has missing values and the implementation of the algorithm does not support missing values so we impute the missing values here.

```r
ds[vars] <- na.roughfix(ds[vars])
```

Now we finalise the meta-data.

```r
# Variable roles.
inputc <- setdiff(vars, target)
inputi <- sapply(inputc, function(x) which(x == names(ds)), USE.NAMES=FALSE)
numi <- intersect(inputi, which(sapply(ds, is.numeric)))
numc <- names(numi)
cati <- intersect(inputi, which(sapply(ds, is.factor)))
catc <- names(cati)

# Remove all observations with a missing target.
ds <- ds[!is.na(ds[target]),]

# Normalise factors.
factors <- which(sapply(ds[vars], is.factor))
for (f in factors) levels(ds[[f]]) <- normVarNames(levels(ds[[f]]))

# Ensure the target is categoric.
ds[target] <- as.factor(ds[[target]])

# Number of observations.
nobs <- nrow(ds)
```
4  Build Model

We use `earth` (Milborrow, 2014).

```r
library(earth) # Model builder

# Formula for modelling.
form <- formula(paste(target, "~ ."))

# Training and test datasets.
seed <- sample(1:1000000, 1)
set.seed(seed)
train <- sample(nobs, 0.7*nobs)
set.seed(seed)
test <- setdiff(seq_len(nobs), train)
actual <- ds[test, target]
risks <- ds[test, risk]

# Build model.
m.earth <- earth(form, data=ds[train, vars])
mtype <- "earth"
model <- m.earth

model
## Selected 21 of 94 terms, and 11 of 62 predictors
## Importance: wind_gust_speed, humidity_3pm, min_temp, max_temp, ...
## Number of terms at each degree of interaction: 1 20 (additive model)
## GCV 0.08528  RSS 15.4  GRSq 0.4259  RSq 0.5919
```
5 Evaluate Model with Error Matrix

```r
library(ROCR)  # prediction()

classes <- predict(model, ds[test, vars], type="class")
acc <- sum(classes == actual, na.rm=TRUE)/length(actual)
err <- sum(classes != actual, na.rm=TRUE)/length(actual)
predicted <- predict(model, ds[test, vars], type="response")
predicted <- rescale(predicted, 0:1)  # TRY THIS THEN READ DOCS
pred <- prediction(predicted, ds[test, target])
ate <- attr(performance(pred, "auc"), "y.values")[[1]]

round(table(actual, classes, dnn=c("Actual", "Predicted"))/length(actual), 2)
## Predicted
## Actual No Yes
## No 0.78 0.04
## Yes 0.12 0.06
```
6 Evaluate Model with Riskchart

library(rattle)  # riskchart()

riskchart(predicted, actual, risks)
7 Further Reading and Acknowledgements

The Rattle Book, published by Springer, provides a comprehensive introduction to data mining and analytics using Rattle and R. It is available from Amazon. Other documentation on a broader selection of R topics of relevance to the data scientist is freely available from http://datamining.togaware.com, including the Datamining Desktop Survival Guide.

This chapter is one of many chapters available from http://HandsOnDataScience.com. In particular follow the links on the website with a * which indicates the generally more developed chapters.

Other resources include:

8 References


