Notes: 'Hands-On Machine Learning with R'

Steven Marcel Bißantz

2023-03-25

Disclaimer: The script contains my extended summary of the book "Hands-On Machine Learning with R" by Bradley Boehmke & Brandon Greenwell. That is, most of the information you read here is not mine. Nevertheless, I have also reworded sections and incorporated my own thoughts. For a clear delimitation, please read the original text.

Chapter 1: Introduction

The essence of machine learning: Develop algorithms/models that use information, such as data or prior knowledge, to approximate or extract functional relationships between variables.

Learning: ... means adaption; i.e. the learner uses the data (or evidence) to (self) optimize the algorithmic steps (e.g., the parameters, θ, of a model) to make accurate prediction.

Example: In supervised learning we adapt to a known ("labeled") target. In unsupervised machine learning we need to adapt to an unknown ("unlabeled") target.

Loss function: ... but 'accuracy' in terms of what? Our goal is encapsulated in a loss (or utility) function, that specifies the difference to a target.

Example: The Kullback-Leibler Divergence minimizes the (info) distance between our model distribution **q** and the true frequency distribution of events **p**

Optimization: With the loss function we optimize (often: minimize, rarely: maximize) a (often: loss, rarely: utility) function using an optimization algorithm, like stochastic gradient descent (SGD) algorithm. The algorithm tells us at each iteration the difference to a prespecified optimum (often: the minimum, e.g. 0).

https://upload.wikimedia.org/wikipedia/commons/a/a3/Gradient_descent.gif Figure: An intuitive visualization of the stochastic gradient descent method. The picture is taken from: https://commons.wikimedia.org/wiki/File:Gradient_descent.gif

Side note: If AIML someday manages to learn causal relationships among variables (DAGs), it might be a data-driven replacement for theory development.

Important: "Causal predictions" vs. "mere predictions"

Machine learning paradigms We could classify ML models according to the amount of supervision needed during training. However, do not forget reinforcement learning:

1. Supervised learning (~predictive models)
2. Unsupervised learning (–descriptive models)
3. Reinforcement Learning

Supervision: ... means, to provide the algorithm with predefined "labeled" training data. No supervision implies, the algorithm needs to do the assignment of the labels itself.

Labelling: ... refers to the process of assigning a specific category or value to each data point (e.g., based on experts opinion, or a specific criterion)

Example: Experts diagnosed (i.e. "labeled") 500 patients with Major Depression Disorder (MDD): y. Now we can train an algorithm based on these data, $\mathcal{D}_{train} : \{X, y\}_{i=1}^{n_{train}}$, together with patient characteristics, $X : f(X) = y$. Then we can use f, the learner, to make predictions for new, unseen patients, for example, i in $\mathcal{D}_{test} : \{X, y\}_{i=1}^{n_{test}}$.

Data quality & Validity I have seen few authors discussing this point: *Do always critically reflect on the quality of your labels.* In unsupervised machine learning we do this, because we want to check the quality of the machine predictions. In supervised machine learning the labels are preassigned. Still, we must check the validity of the scores assigned by experts or criteria. Do they make sense or are they high quality?

The quality of the approximation ($y = f(X)$) is only of importance if y is valid. If y is trash then we get "a good approximation of trash". *Garbage in garbage out* = it is that simple.

Supervised learning (~predictive models)

TODO: Add: A major goal of the machine learning process is to find an algorithm f(X) that most accurately predicts future values (^Y) based on a set of features (X).

Supervised learning: Develop an algorithm or predictive model that uses labeled data $D : \{X, y\}_{i=1}^{n}$ to produces an accurate (e.g., non- or piecewise linear) approximation of the mapping function "f": $y = f(X)$ – and make accurate predictions. (*Note*: The labels are often assigned based on a criterion or expert knowledge)

Why is supervised learning "supervised"?

Because the *training* data ($\mathcal{D}_{train} : \{X, y\}_{i=1}^{n_{train}}$) you feed the algorithm includes the target values (y). Consequently, the solutions can be used to help supervise the training process to find the optimal algorithm parameters.

Underlying function assumption: Given a dataset comprised of inputs and outputs $D : \{X, y\}_{i=1}^{n}$, we assume that there is an unknown underlying function that is consistent in mapping inputs to outputs in the target domain and resulted in the dataset. We then use supervised learning algorithms to approximate this function.

Open Question: Does it need to be a function, or are also relations possible? e.g. $y = x^2$?. I think relations are not possible, since there is no unique element in the codomain or range for each element in the domain.

The two problems in supervised learning

1. Regression problem
2. Classification problem

Regression problem **Regression problem**: ... means the objective of our supervised learning is to predict a numeric outcome that falls on a continuum $(-\infty, \infty)$.

$$y_{\in \mathbb{R}} = f(X_{\in \mathbb{R}})$$

Example: Predict height based on persons characteristics, like sex.

Classification problem **Clasification problem**: ... means the objective of our supervised learning is to predict a categorical outcome (i.e., a binary or multinomial response measure)

Example: Classify customer reviews from 0 to 5 (multinomial) or "like", "don't like" (binary).

Important: In ML often ordinal responses are modeled as categorical processes and thus with a multinomial distribution.

Classification problems are kind of regression problems: Every classification is a (logistic) regression problem. In a general regression problem can be formalized as $y_{\in \mathbb{R}} = f(X_{\in \mathbb{R}})$. In a classification problem

the situation is as follows: $y_{\in[0,1]} = f(X_{\in\mathbb{R}})$. *Note that* $[0,1]$ *is a subset of* $\mathbb{R}^+$. The problem is thus: $y_{\in[0,1]\subset\mathbb{R}^+} = f(X_{\in\mathbb{R}})$. What about the categories? In this binary case, the labels for the categories are assigned ex post according to 0/1 assignment rule.

0/1 Assignment rule: If we predict the probability of a class ($P(X = 1)$); by default, the class with the highest predicted probability becomes the predicted class: if $p \geq 0.5$ it get the label "1", else $p* < 0.5$, and "0".

Unsupervised learning

(~descriptive models)

Unsupervised learning: Develop an algorithm that uses unlabeled information to discover functional relationships between variables (rows, columns).

Underlying function assumption: Given a dataset: D:Z_i, we assume that there is an unknown functional relationship between the variables (rows, columns). We then use unsupervised learning algorithms to discover this function.

In essence, unsupervised learning is concerned with identifying groups in a data set. The groups may be defined by the rows (i.e., clustering) or the columns (i.e., dimension reduction); however, the motive in each case is quite different.

Chapter 2: Modeling Process

```
# Helper packages
pkgs <- c("rsample", # Resampling procedures
          "caret",   # Resampling and model training
          "h2o")     # Resampling and model training

# lapply(pkgs, library, character.only=TRUE)

# h2o
h2o::h2o.no_progress() # Avoid the progress bars in the output
h2o::h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         5 days 14 minutes
##     H2O cluster timezone:       Europe/Berlin
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.40.0.1
##     H2O cluster version age:    1 month and 21 days
##     H2O cluster name:           H2O_started_from_R_steven_qdt481
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.97 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.2.3 (2023-03-15)
```

```r
?AmesHousing::ames_raw

# Ames housing data
ames <- AmesHousing::make_ames()

# Coerce to a h2o frame
ames_h2o <- h2o::as.h2o(ames)

# Job attrition data
# Note: the data are now in the modeldata package
churn <- modeldata::attrition
```

Important: h2o cannot handle ordered factors so we need to coerce them to unordered factors before the analysis.

```r
# Coerce ordered factors
unorder_if <- function(x) ifelse(is.ordered(x), factor(x, ordered = FALSE), x)
# Trick: Use '[]' to keep the data frame structure
churn[] <- lapply(churn, unorder_if)
```

Data splitting

Simple random sampling and stratified sampling

Simple random sampling Use `base` and no additional packages

```r
# Reproduceable results
set.seed(123)
# Number of rows
n_rows <- nrow(ames)
# Row index
row_index <- seq(n_rows)
# Index for the training cases
train_index <- sample(row_index, size = round(n_rows) * 0.7, replace = FALSE)
# Training set
train <- ames[train_index,] ; cat("Training set: ", dim(train), "\n")
```

```
## Training set:  2051 81
```

```r
# Testing set
test <- ames[-train_index,] ; cat("Testing set: ", dim(test), "\n")
```

```
## Testing set:  879 81
```

```r
# Plausibility check
cat("Split: ", dim(train)[1] / n_rows * 100, "/", dim(test)[1] / n_rows * 100)
```

```
## Split:  70 / 30
```

Use the `caret` package

```r
set.seed(123)
```

Use the `rsample` package

```r
set.seed(123)
```