Introduction to Machine Learning

TRSM R Team

What is Machine Learning?

- The idea that we can automate processes
- It is done via algorithms that help **Machines** learn
- In this context machines can be robots, computers, etc.

How do they work?

- Machine Learning models have to be trained.
- There is a labeled training data, we will have an input and an output.
- ML model will relate the input to output.

Terminology

The values that we are trying to find are denoted as **y__true**, and the values that we have predicted using our model will be named **y__pred** (our predicted y). The difference between these two instances will be the error. If you look at the dataframe sum of our predictions with error will be the actual y values (y__true). The goal is minimize this error value in the dataframe.

```
library(tidyverse)
library(ggplot2)

y_true <- rnorm(10, 0, 5)
y_pred <- y_true + rnorm(10, 1, 5e-1)

error <- y_true - y_pred

data.frame(y_true, y_pred, error)
```

```
##        y_true    y_pred      error
## 1   2.095309  3.336877 -1.2415683
## 2  -4.239787 -3.715024 -0.5247632
## 3   6.086174  6.750774 -0.6646005
## 4   0.170588  1.418469 -1.2478809
## 5   1.306781  2.020130 -0.7133497
## 6  -4.947981 -4.582400 -0.3655808
## 7  -6.396923 -5.300041 -1.0968820
## 8   5.768146  6.964501 -1.1963555
## 9   5.400303  6.257196 -0.8568927
## 10  6.170643  7.151055 -0.9804119
```

Loss Function The loss function is used to determine how accurate our predictions are. They are some popular loss function choices for regression:

1. **Mean Absolute Error (MAE)**: Sum of absolute error divided by the number of entries (rows).
2. **Mean Square Error (MSE)**: Sum of square of error divided by the number of entries (rows).
3. **Root Mean Square Error (RMSE)**: Root of sum of sqaured error divided by the number of entries (rows).

```
# MAE
mae <- function(y_true, y_pred) {
  sum(abs(y_true - y_pred)) / NROW(y_true)
}

# MSE
mse <- function(y_true, y_pred) {
  sum((y_true - y_pred) ** 2 / NROW(y_true))
}

# RMSE
rmse <- function(y_true, y_pred) {
  sqrt(sum((y_true - y_pred) ** 2 / NROW(y_true)) )
}

data.frame(mae=mae(y_true, y_pred), mse=mse(y_true, y_pred), rmse=rmse(y_true, y_pred))
```

```
##         mae       mse      rmse
## 1 0.8888286 0.8788175 0.9374527
```

These loss functions can be used interchangeably. There is no specific recipe to determine when to use which function? But our goal is to minimize values obtained from these functions so we can have a more accurate model.

Case Study: Minimizing Error

We are given a dataset (Sample.csv) where there are three features: x1, x2, and x3. We will assign a weight to each of them: w1, w2, w3. Then we alter the weights to get the least error when we compute the y using these features: $w1 * x1 + w2 * x2 + w3 * x3$.

```
s <- read_csv('./Sample.csv')

y_pred_1 <- 1 * x1 + 1 * x2 + 1 * x3
y_pred_2 <- 2 * x1 + 2 * x2 - 1 * x3
y_pred_3 <- 3 * x1 + 3 * x2 - 2 * x3
y_pred_4 <- 2 * x1 + 5 * x2 - 3 * x3

preds <- data.frame(s$y_true, y_pred_1,y_pred_2,y_pred_3,y_pred_4)
head(preds, 5)
```

```
##     s.y_true y_pred_1  y_pred_2   y_pred_3  y_pred_4
## 1 -1.1360664 4.777022 0.6635112 -0.4864887 -2.781978
## 2 10.1462763 7.101787 5.6194380  6.9984677  9.547768
## 3 -1.5353528 5.104542 0.9216330 -0.1654590 -2.274132
## 4  0.9237372 5.367845 2.0913543  1.6963088  0.760168
## 5  6.4072602 6.625494 3.9782532  4.4219240  5.443880
```

The model with the lowest amount of error will be the most accurate one, hence we will look for the weights that yield the minimum error. Looking at the table it is easy to see that the fourth prediction is the most accurate one.

```
error <- c("MAE", "MSE", "RMSE")
pred_1_error <- c(mae(s$y_true, y_pred_1), mse(s$y_true, y_pred_1), rmse(s$y_true, y_pred_1))
pred_2_error <- c(mae(s$y_true, y_pred_2), mse(s$y_true, y_pred_2), rmse(s$y_true, y_pred_2))
pred_3_error <- c(mae(s$y_true, y_pred_3), mse(s$y_true, y_pred_3), rmse(s$y_true, y_pred_3))
pred_4_error <- c(mae(s$y_true, y_pred_4), mse(s$y_true, y_pred_4), rmse(s$y_true, y_pred_4))

errors <- data.frame(error, pred_1_error, pred_2_error, pred_3_error, pred_4_error)
head(errors, 10)
```

```
##   error pred_1_error pred_2_error pred_3_error pred_4_error
## 1   MAE     3.507502     2.296096     1.640520     1.011016
## 2   MSE    18.538004     7.838876     4.039742     1.242105
## 3  RMSE     4.305578     2.799799     2.009911     1.114498
```

Mathematical Functions vs Machine Learning

Consider three parts:

- ***Input***: Data (number) that enters into a function usually denoted as **x**.
- ***Output***: Value(s) that come out of a function, denoted as **y**.
- ***Weight***: Variables within the function that transform the input to output, denoted as **w**.

For instance in $f(x) = x^2 - 2x + 10$, any value of x is the input, f(x) is the output and 1, -2, 10 are the weights.

Machine Learning algorithms have the same three components, so ***what is the difference between a ML algorithm and a function?*** In functions, we have the x and w but we are looking for y, in ML models, we have x and y, we want to find w. Meaning that we have the input and output but we want to related in a way that we could automate our processes.

Different Type of Machine Learning Algorithm

Classification: We have a number of classes and we want to put the given data into a class. - We are working on animal classifier algorithm, where any picture is given to us and we decide which animal is present in the picture. The input is a picture and after running through the ML model, we decide if there is a dog, cat, or etc in the picture. - Detecting financial fraud.

Regression: The goal is to predict one or more numeric values. - Using data we predict the stock/Bitcoin price for tomorrow.