

Handmade Random Forest

Álvaro Orgaz Expósito

THEORY

The aim of the *AdaBoost* algorithm is to predict the variable y (categorical for *classification* problem or continuous for *regression problem*) knowing the explanatory variables $x = \{x_1, \dots, x_p\}$.

Firstly, you need to understand the concept of *boosting*. The idea of *boosting* is to repeatedly apply a weak algorithm on various distributions of the data (in each iteration the data is changed based on the error of the algorithm in the previous iteration, in such a way that the weak model of the iteration focuses on the incorrect previous predictions) and to ensemble the individual models into a single overall model. Then, the aim is to combine multiple models, which individually have a larger error, for getting a final model with a smaller global error.

Note: A weak model is any unstable predictive model whose learning algorithms are sensitive to changes in the training data.

Secondly, let's see the main steps of the *AdaBoost* algorithm, which is one of the available algorithms that use *boosting*.

1. Define the inputs $x_{[NxP]}$ and $y_{[Nx1]}$.
2. Define the number of iterations T and the weak model used in each iteration.
3. Initialize the data weights of the N observations uniformly as

$$w_i^1 = 1/N \quad \text{for } i = 1, \dots, N$$

4. For each iteration $t = 1, \dots, T$:

4.1 Train the weak model using the weights $w_{[Nx1]}^t$, and the inputs $x_{[NxP]}$ and $y_{[Nx1]}$. Then, compute the predictions $\hat{y}_{[Nx1]}^t$.

4.2 Compute the global error ϵ^t of the weak model t as

- In a *regression* problem

$$\epsilon^t = \sum_{i=1}^N w_i^t \times \begin{cases} 1 & \text{if } \frac{|\hat{y}_i^t - y_i|}{y_i} \geq \text{threshold} \\ 0 & \text{if } \frac{|\hat{y}_i^t - y_i|}{y_i} \leq \text{threshold} \end{cases}$$

Note: *threshold* is a parameter of the model that the user defines.

- In a *classification* problem

$$\epsilon^t = \sum_{i=1}^N w_i^t \times \begin{cases} 1 & \text{if } \hat{y}_i^t \neq y_i \\ 0 & \text{if } \hat{y}_i^t = y_i \end{cases}$$

4.3 Compute the confidence α^t of the weak model t as

$$\alpha^t = \frac{1}{2} \left(\log \left(\frac{1 - \epsilon^t}{\epsilon^t} \right) \right)$$

Note: In this formula, the confidence increases when the error decreases.

4.4 Update the weights as

- In a *regression* problem

$$w_i^{t+1} = \frac{w_i^t}{Z^t} \times \begin{cases} e^{\alpha^t} & \text{if } \frac{|\hat{y}_i^t - y_i|}{y_i} \geq \text{threshold} \\ e^{-\alpha^t} & \text{if } \frac{|\hat{y}_i^t - y_i|}{y_i} \leq \text{threshold} \end{cases} \quad \text{for } i = 1, \dots, N$$

where Z^t is a normalization factor ensuring that $\sum_{i=1}^N w_i^{t+1} = 1$. *Note:* In this formula, the weight of each observation increases when the error increases.

- In a *classification* problem

$$w_i^{t+1} = \frac{w_i^t}{Z^t} \times \begin{cases} e^{\alpha^t} & \text{if } \hat{y}_i^t \neq y_i \\ e^{-\alpha^t} & \text{if } \hat{y}_i^t = y_i \end{cases} \quad \text{for } i = 1, \dots, N$$

where Z^t is a normalization factor ensuring that $\sum_{i=1}^N w_i^{t+1} = 1$. *Note:* In this formula, the weight of each observation increases when the error increases.

5. Compute the single overall model ensembling the T models as

- In a *regression* problem

$$\hat{y}_i = \frac{\sum_{t=1}^T \alpha^t \hat{y}_i^t}{\sum_{t=1}^T \alpha^t} \quad \text{for } i = 1, \dots, N$$

Note: The predictions of models with lower global error (higher confidence α^t) have more weight in the final prediction.

- In a *classification* problem with K categories in y

$$\hat{y}_i = \max_k \left(\sum_{t=1}^T \alpha^t \times \begin{cases} 1 & \text{if } \hat{y}_i^t = k \\ 0 & \text{if } \hat{y}_i^t \neq k \end{cases} \right) \quad \text{with } k = 1, \dots, K \quad \text{for } i = 1, \dots, N$$

Note: The overall classification of each observation in the boosted final model is the class k more voted after aggregating the votes of the T models, giving more confidence to the models with a lower global error.

Thirdly, this paper covers the *AdaBoost* with *Classification Decision Tree (using Binary Splitting)* as the weak model in each iteration. Then, it is for *classification* problems and you can find all the necessary information about this model in the paper *Handmade_Decision_Tree.pdf*.

However, this model can be very complex if we build a *full tree*. It means that, if we do not set limitations when training, the tree grows until the *leaf nodes impurity* cannot be improved by a new split. Then, as we need a weak classifier in each *boosting* iteration, we will define the *maximum depth* of the tree as 2 (one split from the *root node* and one more split from each *son nodes*).

Note: We will use the implemented function *rpart* for *Decision Trees* in the R package *rpart*, setting all the parameters by default except of *maxdepth* equals to 2. It is appropriate because it includes the *weights* parameter, and we need it for the *boosting*.

CODE

The data used is the popular dataset *iris*. Let's predict the categorical variable *Species* (then it is a *classification* problem with 3 categories) with the explanatory features: *Sepal.Width* and *Petal.Width*.

Let's define the inputs x and y . Also, let's define x^{test} by creating artificial data with all combinations of both features from the minimum to the maximum values in x by 0.02, it will be interesting for observing the *Random Forest decision boundaries* in the following plots.

```
x <- iris[,c("Sepal.Width", "Petal.Width")]
y <- iris[, "Species"]
x_test <- expand.grid(list(Sepal.Width=seq(min(x[,1]), max(x[,1]), 0.02),
                          Petal.Width=seq(min(x[,2]), max(x[,2]), 0.02)))
```

Let's create the *weak_model* function which uses the inputs *x* and *y* for training a *Classification Decision Tree*, with *maxdepth* 2 and also computes the predictions for the input x^{test} .

```
weak_model <- function(x,y,x_test){
  library(rpart)
  decision_tree <- rpart(y~., data=x, method="class", control=rpart.control(maxdepth=2))
  predictions <- predict(object=decision_tree, newdata=x_test, type="class")
  return(list(decision_tree=decision_tree, predictions=predictions))
}
```

Let's create the *Random Forest* algorithm.

```
RandomForest <- function(x,y,x_test,weak_model,iterations,bootstrap_fraction){
  N <- nrow(x)
  P <- ncol(x)
  K <- length(unique(y))
  N_test <- nrow(x_test)
  predictions <- rep(0, times=N)
  # Create a matrix for the single overall model prediction of x_test
  predictions_test <- matrix(0, nrow=N_test, ncol=K)
  # Iterate the T iterations
  for(t in 1:iterations){
    # Bootstrap sample
    set.seed(t)
    samples_t <- sample(1:N)[1:round(N*bootstrap_fraction)]
    # Features sample
    set.seed(t)
    features_t <- sample(1:P)[1:round(sqrt(P))]
    # Train the weak model t and compute the predictions for x and x_test
    predictions_t <- weak_model(x[samples_t, features_t, drop=FALSE], y[samples_t], x)$predictions
    predictions_test_t <- weak_model(x[samples_t, features_t, drop=FALSE], y[samples_t], x_test)$predictions
    # For each x_test observation, sum the weak model t confidence to the predicted class
    for(i in 1:N_test){
      k <- predictions_test_t[i]
      predictions_test[i, k] <- predictions_test[i, k] + 1
    }
  }
  # Compute the single overall model prediction for x_test
  y_test <- rep(0, times=N_test)
  for(i in 1:N_test){
    y_test[i] <- which.max(predictions_test[i,])
  }
  return(y_test)
}
```

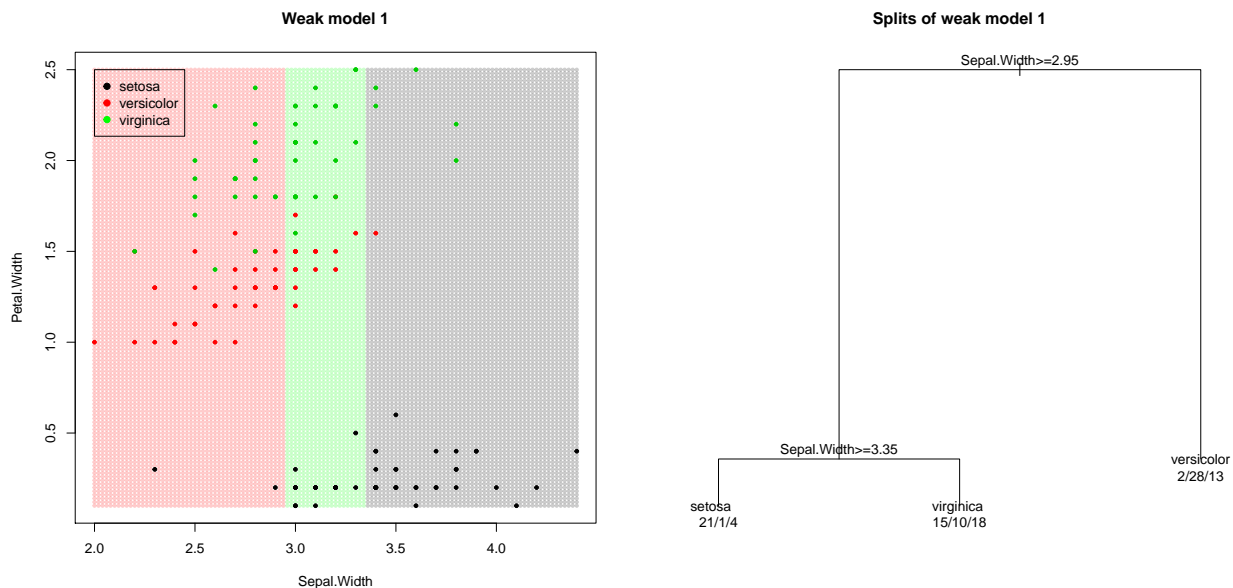
Now, let's apply the first weak model (iteration 1 of the *Random Forest*) and plot the result.

```
par(mfrow=c(1,2), xpd=TRUE)
# Bootstrap sample
bootstrap_fraction <- 0.75
N <- nrow(x)
```

```

set.seed(1)
samples_1 <- sample(1:N)[1:round(N*bootstrap_fraction)]
# Features sample
P <- ncol(x)
set.seed(1)
features_1 <- sample(1:P)[1:round(sqrt(P))]
# Train the weak model 1
weak_model_1 <- weak_model(x[samples_1,features_1,drop=FALSE],y[samples_1],x_test)
# Decision boundaries
predictions <- weak_model_1$predictions
transparent_colors <- scales::alpha(c("black","red","green"),0.2)
plot(x_test[,1],x_test[,2],col=transparent_colors[as.numeric(predictions)],
     pch=19,cex=0.5,xlab="Sepal.Width",ylab="Petal.Width",main="Weak model 1")
points(x[,1],x[,2],col=y,pch=19,cex=0.6)
legend(2,2.5,legend=unique(y),col=c("black","red","green"),pch=19)
# Splits
plot(weak_model_1$decision_tree,main="Splits of weak model 1")
text(weak_model_1$decision_tree,use.n=TRUE)

```



Now, let's apply the *AdaBoost* with 10 and 50 iterations, and plot the result.

```

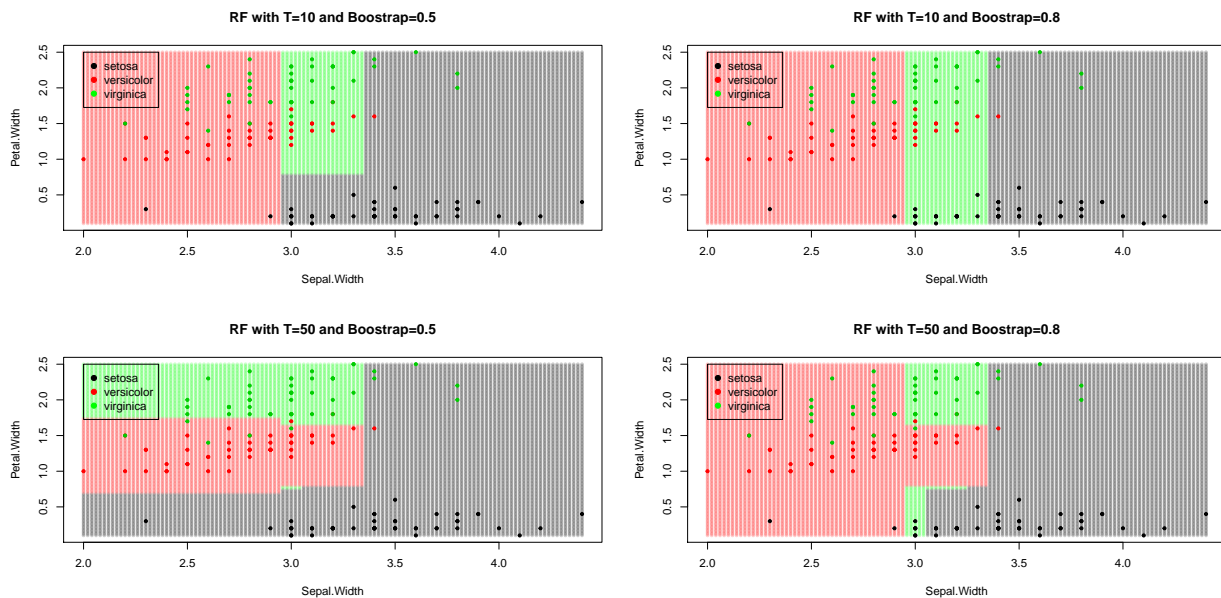
par(mfrow=c(2,2))
# Random Forest with 10 iterations and bootstrap fraction 0.50
predictions <- RandomForest(x,y,x_test,weak_model,iterations=10,bootstrap_fraction=0.50)
transparent_colors <- scales::alpha(c("black","red","green"),0.2)
plot(x_test[,1],x_test[,2],col=transparent_colors[as.numeric(predictions)],
     pch=19,cex=0.5,xlab="Sepal.Width",ylab="Petal.Width",main="RF with T=10 and Bootstrap=0.5")
points(x[,1],x[,2],col=y,pch=19,cex=0.6)
legend(2,2.5,legend=unique(y),col=c("black","red","green"),pch=19)
# Random Forest with 10 iterations and bootstrap fraction 0.80
predictions <- RandomForest(x,y,x_test,weak_model,iterations=10,bootstrap_fraction=0.8)
transparent_colors <- scales::alpha(c("black","red","green"),0.2)
plot(x_test[,1],x_test[,2],col=transparent_colors[as.numeric(predictions)],
     pch=19,cex=0.5,xlab="Sepal.Width",ylab="Petal.Width",main="RF with T=10 and Bootstrap=0.8")

```

```

points(x[,1],x[,2],col=y,pch=19,cex=0.6)
legend(2,2.5,legend=unique(y),col=c("black","red","green"),pch=19)
# Random Forest with 50 iterations and bootstrap fraction 0.50
predictions <- RandomForest(x,y,x_test,weak_model,iterations=50,bootstrap_fraction=0.50)
transparent_colors <- scales::alpha(c("black","red","green"),0.2)
plot(x_test[,1],x_test[,2],col=transparent_colors[as.numeric(predictions)],
     pch=19,cex=0.5,xlab="Sepal.Width",ylab="Petal.Width",main="RF with T=50 and Bootstrap=0.5")
points(x[,1],x[,2],col=y,pch=19,cex=0.6)
legend(2,2.5,legend=unique(y),col=c("black","red","green"),pch=19)
# Random Forest with 50 iterations and bootstrap fraction 0.80
predictions <- RandomForest(x,y,x_test,weak_model,iterations=50,bootstrap_fraction=0.8)
transparent_colors <- scales::alpha(c("black","red","green"),0.2)
plot(x_test[,1],x_test[,2],col=transparent_colors[as.numeric(predictions)],
     pch=19,cex=0.5,xlab="Sepal.Width",ylab="Petal.Width",main="RF with T=50 and Bootstrap=0.8")
points(x[,1],x[,2],col=y,pch=19,cex=0.6)
legend(2,2.5,legend=unique(y),col=c("black","red","green"),pch=19)

```



In conclusion, in this plots, we can observe the predictions for x^{test} (transparent coloured points) and the real values y (solid coloured points). Also, we can observe the *decision boundary* for each class in the target variable *Species* in the *iris* data.

Comparing the plots of three models (first weak model and two versions of *AdaBoost*), the decision boundaries are more complex and smooth when increasing the number of *boosting* iterations and this is because each iteration improves the previous predictions. Then, the more iterations, the more chances to classify well the misclassified observations.

Note: Remember that we are using always the same weak model (*Classification Decision Tree* with *maxdepth* 2) and we get smoother decision boundaries and higher accuracy in the single overall model just *boosting* it. It is really useful because it enables us to improve our model without changing it for another more complex.