Random Forest and Deep Learning

Carlos E Martínez-Rodríguez

14 de noviembre de 2023

Contents

1 Random Forest: Explanation in Mathematical Terms

Random Forest is a supervised learning algorithm used for both classification and regression problems. The main idea behind Random Forest is to build multiple decision trees during training and combine their results to obtain a more robust and accurate prediction.

Let's assume we have a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the features and y_i is the target variable.

1. **Construction of Decision Trees:**
   - For each tree, a random subset of features is selected (randomly sample features), and a random subset of training data is chosen (randomly sample observations with replacement).
   - A decision tree is constructed using the subset of data and features.
   - The decision tree is represented as $h_i(x; \theta_i)$, where i denotes the tree index, x is the input feature vector, and θ_i represents the parameters of the tree.
   - At each node t of the tree, a feature j_t is selected from the random subset, and the split is determined based on minimizing impurity:

$$\theta_{i,t} = \arg\min_{j_t, s_t} \Big[\text{Impurity}(D_t) - p_{\text{left}}\text{Impurity}(D_{\text{left}}) - p_{\text{right}}\text{Impurity}(D_{\text{right}})\Big]$$

     where D_t is the dataset at node t, D_{left} and D_{right} are the datasets in the left and right child nodes, p_{left} and p_{right} are the proportions of data in the left and right child nodes, and $\text{Impurity}(\cdot)$ is a measure of impurity, such as Gini index for classification or mean squared error for regression.

2. **Voting or Averaging:**
   - For classification problems, the final prediction is obtained by majority voting among the trees:

$$H(x) = \text{mode}\{h_1(x; \theta_1), h_2(x; \theta_2), \ldots, h_n(x; \theta_n)\}$$

   - For regression problems, the final prediction is the average of predictions from all trees:

$$H(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x; \theta_i)$$

3. **Variance Reduction:**
   - By building decision trees in a random manner, the variance of the model is reduced, leading to a more robust and generalizable model.

The final prediction is obtained by combining the predictions of all trees.

2 Deep Learning: Overview

Deep Learning is a subfield of machine learning that focuses on neural networks with multiple layers (deep neural networks). These networks can automatically learn hierarchical representations of data, allowing them to capture intricate patterns and features.

1. **Neural Network Representation:**
   - A neural network consists of layers of interconnected nodes (neurons) organized into an input layer, one or more hidden layers, and an output layer.
   - The input layer represents the features of the data, and each neuron in the layer processes a specific feature.
   - Hidden layers perform nonlinear transformations on the input data, learning hierarchical representations.
   - The output layer produces the final prediction based on the learned representations.

2. **Training a Neural Network:**
   - During training, the network's parameters (weights and biases) are adjusted to minimize the difference between predicted and actual outputs.
   - This optimization is typically done using backpropagation and gradient descent.
3. **Activation Functions:**
   - Activation functions introduce nonlinearity to the neural network, enabling it to learn complex patterns.
   - Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (tanh).
4. **Forward Pass:**
   - The forward pass of a neural network involves computing the output of the network for a given input.
   - Given an input vector x, the output y is computed by passing x through the network's layers using learned weights and biases.
   - The output of each layer is computed as:

     $$a^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

     where $W^{(l)}$ is the weight matrix, $a^{(l)}$ is the activation of layer l, $a^{(l-1)}$ is the activation of the previous layer, and $b^{(l)}$ is the bias vector.
   - The activation function is then applied to $a^{(l)}$ to introduce nonlinearity.
5. **Loss Function:**
   - The loss function measures the difference between the predicted output and the true output.
   - Common loss functions include mean squared error for regression and cross-entropy for classification.
   - The goal during training is to minimize the loss by adjusting the network's parameters.
6. **Backpropagation:**
   - Backpropagation is a training algorithm that computes the gradient of the loss with respect to the network's parameters.
   - The gradient is used to update the parameters in the direction that reduces the loss.
   - It involves computing the gradient of the loss with respect to the output of each layer and using the chain rule to propagate these gradients backward through the network.

3 Example of Implementation in R:

3.1 Install and Load the Keras Library:

```
install.packages("keras")
library(keras)
```

3.2 Build a Simple Neural Network:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = 'relu', input_shape = c(10)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

summary(model)
```

3.3 Compile and Train the Neural Network:

```
model %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = c('accuracy')
)

# Assuming you have training data X_train and labels y_train
history <- model %>% fit(
  X_train, y_train,
  epochs = 10, batch_size = 32,
  validation_split = 0.2
)
```

3.4 Evaluate the Model:

```
# Assuming you have test data X_test and labels y_test
evaluate_result <- model %>% evaluate(X_test, y_test)
print(evaluate_result)
```

4 Impurity Measures in Random Forest

In the context of Random Forest, impurity measures play a crucial role in the construction of decision trees. The two commonly used impurity measures are the Gini index for classification and the mean squared error for regression.

4.1 Gini Index for Classification

The Gini index is a measure of impurity used in classification problems. Given a node in a decision tree that contains data points from different classes, the Gini index quantifies how often a randomly chosen data point would be incorrectly classified.

For a node t with K classes and a set of data points D_t, the Gini index ($Gini(t)$) is calculated as follows:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

where p_i is the proportion of data points in class i at node t. A lower Gini index indicates a purer node with predominantly one class.

In the context of Random Forest, the decision tree split is determined by minimizing the weighted sum of Gini indices for the left and right child nodes. The split that results in the lowest overall Gini index is chosen.

4.2 Mean Squared Error for Regression

For regression problems, the impurity measure used is the mean squared error (MSE). Unlike classification, where impurity is related to the purity of classes in a node, regression impurity is a measure of the variability of target values within a node.

For a node t with data points D_t, the MSE ($MSE(t)$) is calculated as follows:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

where y_i is the target value of data point i, $|D_t|$ is the number of data points in node t, and $\bar{y}_t$ is the mean target value of all data points in node t.

Similar to the Gini index, in Random Forest, the decision tree split is determined by minimizing the weighted sum of MSE for the left and right child nodes.

These impurity measures guide the construction of individual decision trees within the Random Forest ensemble, contributing to the overall robustness and predictive power of the model.

5 Construcción del Árbol de Decisión

5.1 Teoría:

Un árbol de decisión es una estructura de datos que representa un conjunto de decisiones y sus posibles consecuencias. En el contexto de Random Forest, la construcción de un árbol de decisión sigue el principio de "aprendizaje supervisado", donde el algoritmo aprende patrones a partir de un conjunto de datos etiquetado.

La construcción del árbol se realiza a través de divisiones recursivas basadas en características del conjunto de datos. Cada nodo del árbol representa una pregunta sobre una característica, y las ramas que surgen de ese nodo son las respuestas a esa pregunta. El proceso continúa hasta que se alcanza un criterio de parada, como la profundidad máxima del árbol o el número mínimo de muestras en un nodo.

5.2 Elementos Matemáticos:

1. **Función de Impureza:**

En cada nodo del árbol, se elige la característica y el umbral que minimizan la impureza en los nodos hijos resultantes. La impureza se mide mediante funciones como el Índice de Gini para clasificación o el Error Cuadrático Medio (MSE) para regresión.

Para clasificación:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

Donde p_i es la proporción de ejemplos de la clase i en el nodo t.

Para regresión:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

Donde D_t es el conjunto de datos en el nodo t, y_i es la etiqueta del ejemplo i, y $\bar{y}_t$ es la media de las etiquetas en el nodo t.

2. **Criterio de División:**

La elección de la mejor característica y umbral se basa en la reducción de la impureza. Se busca el par (j, s) que minimiza la expresión:

$$\theta_{i,t} = \arg\min_{j,s} \left[\text{Impureza}(D_t) - p_{\text{left}} \text{Impureza}(D_{\text{left}}) - p_{\text{right}} \text{Impureza}(D_{\text{right}}) \right]$$

Donde D_t es el conjunto de datos en el nodo t, D_{left} y D_{right} son los conjuntos de datos en los nodos izquierdo y derecho después de la división, y p_{left} y p_{right} son las proporciones de datos en esos nodos.

3. **Criterios de Parada:**

Para evitar sobreajuste, se utilizan criterios de parada, como la profundidad máxima del árbol o el número mínimo de muestras requeridas para realizar una división.

La construcción de cada árbol en Random Forest implica este proceso iterativo y se repite para cada árbol en el bosque. La diversidad en la construcción de árboles se logra mediante el uso de diferentes subconjuntos aleatorios de características y datos en cada árbol. La combinación de predicciones de estos árboles mejora la generalización del modelo y su capacidad predictiva en nuevos datos.

6 Reducción de Varianza en Random Forest

6.1 Teoría:

La reducción de la varianza es uno de los objetivos clave en la construcción de árboles de decisión en Random Forest. Esta técnica busca mejorar la generalización del modelo al reducir la sensibilidad a pequeñas variaciones en los datos de entrenamiento.

La varianza se refiere a la variabilidad de las predicciones de un modelo respecto a diferentes conjuntos de datos de entrenamiento. En Random Forest, la reducción de la varianza se logra mediante dos técnicas principales: Bagging y Random Subspace.

6.1.1 Bagging (Bootstrap Aggregating):

Bagging es una técnica que consiste en entrenar múltiples modelos en diferentes subconjuntos de datos generados mediante muestreo con reemplazo (bootstrap). Cada árbol de decisión en Random Forest se entrena en un conjunto de datos ligeramente diferente, lo que introduce diversidad en los modelos.

6.1.2 Random Subspace:

Random Subspace es otra técnica utilizada para reducir la correlación entre los árboles de decisión. En lugar de usar todas las características para cada árbol, se selecciona un subconjunto aleatorio de características para entrenar cada árbol. Esto ayuda a que cada árbol se especialice en diferentes aspectos de los datos, mejorando la diversidad y reduciendo la correlación entre las predicciones.

6.2 Elementos Matemáticos:

La reducción de la varianza no se expresa directamente con fórmulas matemáticas específicas, pero los conceptos clave son fundamentales:

1. **Bagging:** - Cada árbol T_i se entrena en un conjunto de datos D_i generado por muestreo con reemplazo (bootstrap). - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

2. **Random Subspace:** - Cada árbol T_i se entrena utilizando un subconjunto aleatorio de características. - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Estos enfoques combinados contribuyen a reducir la varianza del modelo, lo que resulta en un modelo más robusto y generalizable.

7 Votación y Promedio en Random Forest

7.1 Teoría:

La técnica de "Votación" (para problemas de clasificación) o "Promedio" (para problemas de regresión) es crucial en Random Forest para combinar las predicciones de múltiples árboles de decisión y obtener una predicción final más robusta y precisa.

7.1.1 Para Problemas de Clasificación:

En problemas de clasificación, el enfoque de votación se utiliza. Cada árbol en el bosque emite una predicción de clase, y la clase final se determina por mayoría de votos.

7.1.2 Para Problemas de Regresión:

En problemas de regresión, se utiliza un enfoque de promedio. Cada árbol realiza una predicción numérica, y la predicción final es el promedio de todas las predicciones.

7.2 Elementos Matemáticos:

1. **Votación para Clasificación:** - La predicción final para un ejemplo x se obtiene por mayoría de votos:

$$H(x) = \text{mode}\{T_1(x), T_2(x), ..., T_N(x)\}$$

2. **Promedio para Regresión:** - La predicción final para un ejemplo x se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Donde $T_i(x)$ representa la predicción del árbol i para el ejemplo x, y N es el número total de árboles en el bosque.

Estos enfoques de votación y promedio permiten que Random Forest combine la información de múltiples árboles de decisión, mejorando la generalización y la capacidad predictiva del modelo.

8 Resumen

Random Forest es un algoritmo de aprendizaje automático que se utiliza para resolver problemas de clasificación y regresión. En lugar de utilizar un único árbol de decisión, Random Forest construye varios árboles de decisión y los combina para obtener una predicción más precisa. Cada árbol de decisión se construye utilizando un subconjunto aleatorio de las características del conjunto de datos original. Al final, las predicciones de cada árbol se promedian para obtener una predicción final.

El algoritmo de Random Forest se basa en dos técnicas: Bagging y Random Subspace. Bagging es una técnica que se utiliza para reducir la varianza de un modelo al entrenar múltiples modelos en diferentes subconjuntos de datos. Random Subspace es una técnica que se utiliza para reducir la correlación entre los modelos al entrenar cada modelo en diferentes subconjuntos de características.

Aquí hay algunos elementos matemáticos que se utilizan en Random Forest:

- **Árbol de decisión:** Un árbol de decisión es una estructura de datos que se utiliza para modelar decisiones y sus posibles consecuencias. Cada nodo en el árbol representa una decisión, y cada rama representa una posible consecuencia de esa decisión. Los árboles de decisión se construyen utilizando un conjunto de reglas que se utilizan para tomar decisiones.
- **Bootstrap:** Bootstrap es una técnica que se utiliza para generar múltiples conjuntos de datos a partir de un conjunto de datos original. Cada conjunto de datos se genera mediante muestreo con reemplazo, lo que significa que cada elemento del conjunto de datos original tiene la misma probabilidad de ser seleccionado en cada conjunto de datos generado.
- **Out-of-Bag Error:** Out-of-Bag Error es una técnica que se utiliza para estimar el error de validación de un modelo sin la necesidad de un conjunto de datos de validación separado. El error se estima utilizando los datos que no se incluyeron en el conjunto de datos de entrenamiento para cada árbol de decisión.

8.1 Ejemplo en R: Random Forest

Aquí hay un ejemplo de cómo construir un modelo de bosque aleatorio en R:

```
# Cargamos el paquete necesario para este ejemplo
library(randomForest)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Entrenamos el modelo de bosque aleatorio
rf_model <- randomForest(Species ~ ., data = iris[trainIndex,])

# Realizamos predicciones en el conjunto de prueba
predictions <- predict(rf_model, iris[-trainIndex,])
```

8.2 Deep Learning

Deep Learning es un subcampo del aprendizaje automático que se centra en la creación de redes neuronales artificiales profundas. Estas redes neuronales están diseñadas para imitar el cerebro humano y son capaces de aprender patrones complejos en los datos.

Aquí hay un ejemplo de cómo construir una red neuronal profunda en R utilizando el paquete keras:

```
# Cargamos los paquetes necesarios
library(keras)
library(tensorflow)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Creamos la red neuronal
model <- keras_model_sequential() %>%
  layer_dense(units = 4, input_shape = c(4)) %>%
  layer_activation("relu") %>%
  layer_dense(units = 3) %>%
  layer_activation("softmax")

# Compilamos la red neuronal
model %>% compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accuracy")

# Entrenamos la red neuronal
history <- model %>% fit(
  x = iris[trainIndex, 1:4],
  y = to_categorical(as.numeric(iris[trainIndex, 5])),
  epochs = 100,
  batch_size = 10,
  validation_split = 0.2
)
```