Capítulo 3

Clasificadores Débiles - AdaBoost

El término *boosting* hace referencia a un tipo de algoritmos cuya finalidad es encontrar una hipótesis fuerte a partir de utilizar hipótesis simples y débiles. Durante este trabajo se hablará del algoritmo AdaBoost, el cual fue creado por Freund y Schapire y es un diseño mejorado del *boosting* original; y de sus diferentes variantes [Freund and Schapire E., 1996].

AdaBoost es una contracción de "Adaptive Boosting", en donde el término *Adaptive* hace alusión a su principal diferencia con su predecesor. En términos de funcionalidad son iguales, ambos algoritmos buscan crear un clasificador fuerte cuya base sea la combinación lineal de clasificadores "débiles simples" $h_t(x)$. Sin embargo, AdaBoost propone entrenar una serie de clasificadores débiles de manera iterativa, de modo que cada nuevo clasificador o "*weak learner*" se enfoque en los datos que fueron erróneamente clasificados por su predecesor, de esta manera el algoritmo se adapta y logra obtener mejores resultados.

En este punto cabe aclarar que el tipo de AdaBoost sobre el que se trabajó en esta investigación es el binario; esto significa que sólo se trabaja con dos tipos de datos los

cuales se representan con $+1$ y -1, por lo tanto el resultado se expresa como "pertenece a x clase/no pertenece a x clase". La contraparte del tipo binario es el tipo multiclase, cuya característica es que puede diferenciar entre numerosos tipos de datos pero tiene una complejidad de análisis mayor.

Una de las formas más sencillas de entender la finalidad de este tipo de algoritmos es mediante el ejemplo que se da en [Freund and Schapire E., 1999] y en [Corso,]. El primer ejemplo considera a una persona experta en apuestas de carreras de caballos, dicho personaje puede tener varias estrategias para determinar al posible ganador en una carrera específica, por ejemplo basándose en mayor número de carreras ganadas, mejor tiempo en dar una vuelta, el caballo con mayor número de apuestas, etc. Sin embargo, para obtener una hipótesis que permita predecir con mayor seguridad el resultado de cualquier carrera, se necesita discernir cuáles carreras aportan mayor información. El segundo ejemplo consiste en considerar un filtro capaz de diferenciar entre un correo spam y uno que no lo es, para esto se crean hipótesis acerca de qué tipo de palabras clave contienen este tipo de archivos y así generar un clasificador que logré diferenciar de manera más precisa entre estos dos tipos de correos. Para lograr esto, primero se necesita de un "*weak learner*", este es un algoritmo que nos ayuda a encontrar las hipótesis débiles. El método boosting llama t veces al weak learner, cada vez dándole una distribución diferente de los datos para el entrenamiento.

A todos los datos se les asigna inicialmente el mismo peso de $(1/m)$, este peso se va actualizando con cada iteración según los ejemplos mal clasificados, de esta manera se busca minimizar el error esperado y enfocarse en clasificar correctamente los datos

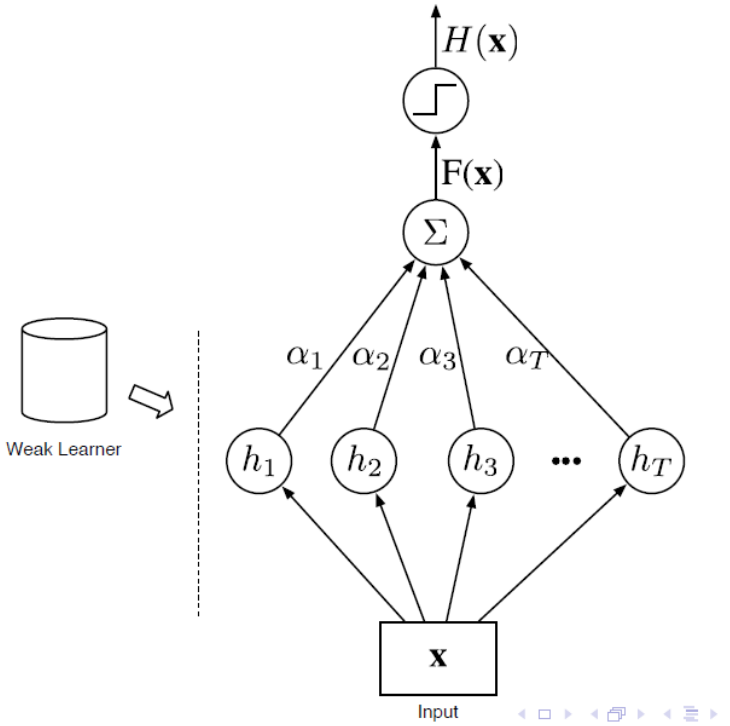

Figura 3.1: Diagrama de flujo del funcionamiento de AdaBoost

que ahora tienen mayor peso. De este modo, al final se realiza la suma de todos los clasificadores previamente generados y se obtiene una hipótesis cuya predicción es más acertada. La Figura 3.1 muestra un diagrama de flujo que explica el funcionamiento de AdaBoost.

Para que un clasificador sea efectivo y preciso en sus predicciones, se requiere que cumpla las siguientes tres condiciones [N. Vapnik, 2013]:

- Debe de haber sido entrenado con "suficientes" ejemplos-datos.
- Debe de tener un error de entrenamiento bajo.
- Debe de ser "simple"

La Figura 3.2 muestra un ejemplo del procedimiento que un algoritmo AdaBoost

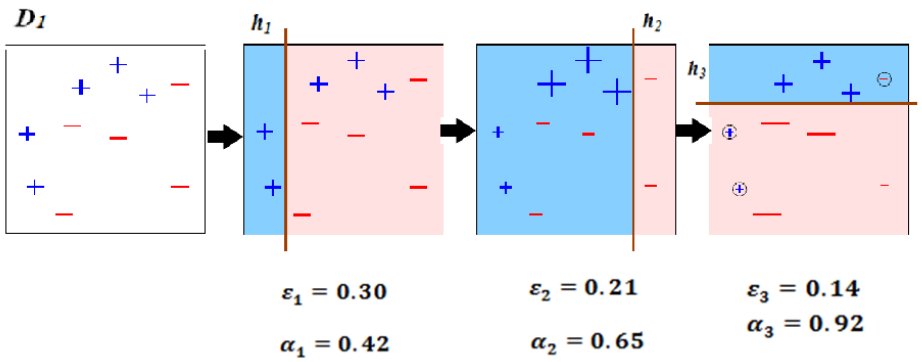

Figura 3.2: Ejemplo de procedimiento de AdaBoost

simple sigue para formar el clasificador final, para este caso se utiliza como *weak learner* un árbol de decisión de un sólo nivel. Se tiene el siguiente conjunto de datos en donde un tipo de datos se representa por símbolos "+" azules y el otro por símbolos "-" en rojo. En su primer intento, el clasificador realiza un "corte" tratando de separar los datos pero podemos observar como realiza tres errores: deja dos datos negativos dentro del conjunto positivo y un dato positivo no lo toma en cuenta. En su siguiente intento, el peso de estos tres datos en donde hubo error es incrementado, de modo que el nuevo clasificador le de más importancia a resolver bien estos puntos.

En su segundo intento logra discriminar correctamente a los datos negativos que tenían más peso pero ahora incluye aquellos que tenían un bajo peso. Debido a esto, se incrementa el peso de estos errores y, por el contrario, aquellos datos que ya fueron correctamente clasificados reciben un peso menor. Se realiza un tercer clasificador que de igual manera

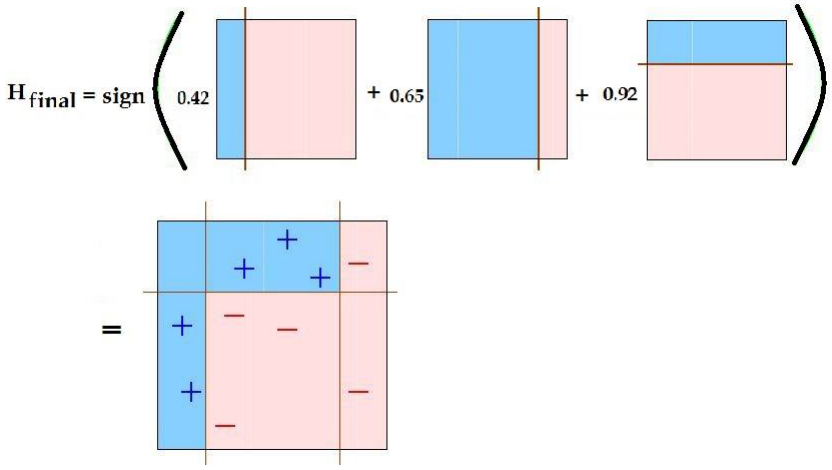

Figura 3.3: Clasificador final

tratará de hacer el mejor " corte " para resolver el problema. Finalmente se sumarán todos los clasificadores creados para crear la solución que, teóricamente, debería de ser mejor que cada clasificador por separado.

A continuación se explicará el pseudocódigo de Adaboost, lo primero es definir sus entradas, que en este caso son el conjunto de datos a utilizar junto con sus etiquetas para determinar de que tipo son, determinar el número de iteraciones a realizar e inicializar el la distribución de pesos otorgados a cada uno de los datos. Dado:$(x_1, y_1), ..., (x_m, y_m)$ donde $x_i \in X, y_i \in -1, +1$.

Inicializar: $D_i = 1/m$ para $i = 1, ..., m$.

Para $t = 1, ..., T$:

- Entrenar el clasificador débil (*weak learner*) utilizando la distribución D_i.
- Obtener una hipótesis débil $h_t : X \to -1, +1$.

- Meta: Obtener una h_t con un error bajo.

$$\epsilon_t = Pr_{i\ D_t}[h_t(x_i) \neq y_i]. \tag{3.1}$$

- Calcular el peso del clasificador débil actual:

$$\alpha_t = \frac{1}{2} ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right). \tag{3.2}$$

- Actualizar la distribución D_t, para $i = 1, ..., m$, según los errores que se hayan realizado en la etapa previa.

$$D_{t+1}(i) = \frac{D_t(i) exp\left(-\alpha_t y_i h_t\left(x_i\right)\right)}{Z_t} \tag{3.3}$$

En donde Z_t es un factor de normalización (elegido de modo que D_{t+1} sea una distribución). La hipótesis final se expresa de la siguiente manera:

$$H\left(x\right) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{3.4}$$

Es decir, se realiza una sumatoria de todas las hipótesis débiles calculadas para poder obtener la hipótesis fuerte. En [Emer, 2012] se explican las ventajas y desventajas que tienen los algoritmos AdaBoost, esta información se puede visualizar en la Tabla 3.1.

Desventajas	Ventajas
Clasificadores débiles complejos pueden llevar a *overfitting*	Simple y sencillo de programar
Clasificadores débiles demasiado débiles pueden producir un bajo margen y *overfitting*	El único parámetro a establecer son las iteraciones
Es vulnerable al rudio	El clasificador débil no requiere conocimiento previo
	Versátil Rápido

Tabla 3.1: Ventajas y Desventajas de AdaBoost

3.1. GML AdaBoost

Durante este trabajo el algoritmo que se utilizó para realizar la clasificación de las características se encuentra en la ToolBox "GML AdaBoost" [Vezhnevets, 2007], este archivo utiliza como *weak learner* a un árbol de decisión CART junto con uno de tres algoritmos disponibles: Real, Gentle y Modest AdaBoost.

Este Toolbox utiliza el tipo de aprendizaje supervisado, este tipo de aprendizaje se utiliza cuando se tiene conocimiento pleno acerca del tipo al que pertenecen cada uno de los datos que se van a utilizar. Por lo tanto, se generaron mediante una estructura *for* dos vectores, estos elementos contienen etiquetas que nos permiten identificar mediante un $+1$ cierto tipo de llanto, por ejemplo asfixia o sordera, y con un -1 un tipo de llanto diferente.

Las funciones más importantes dentro de este algoritmo son: stump, tree node, XAdaBoost, Classify. Los primeros dos se utilizan para generar el árbol e ir agregando los nodos conforme avanza el programa. XAdaBoost es el algoritmo que realiza el entrenamiento de los nodos del árbol a través de un algoritmo *boosting*. Para utilizarlo se debe de inicializar al árbol, cargar los datos de interés junto con sus respectivas etiquetas y determinar el

número de iteraciones que va a realizar, como resultado entrega un conjunto de nodos entrenados. Classify utiliza los nodos entrenados por la función AdaBoost para clasificar los datos de interés. Esta función entrega como resultado un vector con valores de +1 y -1, posteriormente se comparan estos valores con el vector de etiquetas para así poder determinar el número de aciertos que tuvo.

Los parámetros que se requieren establecer para utilizar este algoritmo son el número de nodos que tendrá el árbol y el número de iteraciones que realizará el algoritmo. El valor para ambos parámetros fue determinado mediante métodos heurísticos, por lo que el número de nodos fue de 30 y se fijaron 100 iteraciones por prueba.

3.1.1. Modest AdaBoost

El primer algoritmo de AdaBoost fue propuesto en 1996, a partir de entonces se han diseñado numerosas versiones modificadas que buscan tener mejores características que su versión original. Entre los métodos que se han creado se pueden mencionar los siguientes: Real AdaBoost, Gentle AdaBoost, Margin AdaBoost, Float AdaBoost, entre otros. Sin embargo, durante este trabajo se hará especial énfasis en el método de Modest AdaBoost [Vezhnevets and Vezhnevets, 2007].

A continuación se explicara el pseudocógido de Modeste AdaBoost, al igual que el AdaBoost original se utilizan como entrada los siguiente parámetros: Z= $(z_1, z_2, ..., z_N$ con $z_i = (x_i, y_i)$ como conjunto de entrenamiento. M, que representa el máximo numero de clasificadores. La salida, por otro lado, es la siguiente: H(x), la cual representa a clasificador adaptado al conjunto de entrenamiento.

- Inicializar los pesos de los datos, al inicio todos son iguales $\omega_i = \frac{1}{N}$,i$\in 1, ..., N$
- De m=1 hasta M y mientras $H_m \neq$.
  - Entrenar $H_m(x)$ utilizando la distribución de ω_i.
  - Calcular la distribución "invertida" $\varpi = (1 - \omega_i)$ y re-normalizar a $\sum_i \varpi_i = 1$.
  - Se recalculan los pesos del clasificador previo :

$$\begin{aligned} P_m^{+1} &= P_\omega(y = +1, H_m(x)). \\ \bar{P}_m^{+1} &= P_\varpi(y = +1, H_m(x)). \\ P_m^{-1} &= P_\omega(y = -1, H_m(x)). \\ \bar{P}_m^{-1} &= P_\varpi(y = -1, H_m(x)). \end{aligned} \tag{3.5}$$

  - Poner $H_m(x) = (P_m^{+1}(1 - P_m^{+1}) - P_m^{-1}(1 - P_m^{-1}))$
  - Actualizar $\omega_i \leftarrow \omega_i exp(-y_i H_m(x_i))$ y re-normalizar a $\sum_i \omega_i = 1$.
- La salida se expresa como:

$$H\left(x\right) = sign\left(\sum_{m=1}^{M} H_m(x)\right). \tag{3.6}$$

Las ecuaciones 3.5 son las que logran hacer diferente a AdaBoost, estos valores representan que tan bien está funcionando un clasificador. Dicho valor se decrementa si es que un clasificador está trabajando "demasiado bien" en datos que ya se han clasificado correctamente, de esta manera se da un poco de holgura al clasificador para seguir evaluando los datos y encontrar una mejor manera de clasificarlos.

Modest AdaBoost es un método que, a diferencia del algoritmo original; aumenta el peso de los elementos correctamente clasificados y disminuye aquellos en los que hubo errores. Las características que lo convierten en un buen método para este experimento son: su capacidad de evadir el *overfitting* y su menor porcentaje de error a cambio de tener un error mayor durante el entrenamiento.

El *overfitting*, también llamado sobreajuste; es el efecto que se da al entrenar de más un algoritmo de aprendizaje, de este modo el algoritmo queda muy ajustado a características muy específicas y por lo tanto su respuesta a nuevos datos empeora.

Para lograr esto, el algoritmo le da un valor a cada clasificador que representa la eficiencia al trabajar con los datos, en caso de que este funcionando "demasiado bien" su valor se decrementa para evitar que pase por alto información importante por "confiarse".

Los otros dos algoritmos considerados durante este trabajo son Real y Gentle AdaBoost. Real AdaBoost es una versión mejorada del AdaBoost original, su nombre se debe a que obtiene como resultado un valor real, el cual representada la probabilidad de que cierto patrón de entrada pertenezca a una clase. Por otro lado, Gentle AdaBoost es un algoritmo cuyo resultado es un conjunto más estable y confiable, esto se logra al realizar una optimización con respecto a H_m mediante métodos numéricos.

3.1.2. CART (Classification and Regression Trees

Los árboles de decisiones son estructuras que permiten llegar a un resultado después de pasar por numerosas "reglas" que describen el comportamiento de su entrada a través de una serie de atributos. Para que sean más fáciles de entender para el ser humano pueden

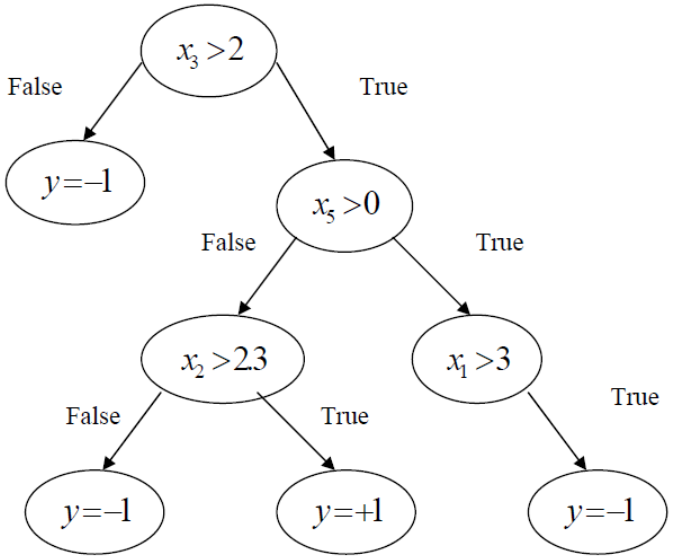

Figura 3.4: Ejemplo de un árbol CART

ser representados mediante reglas tipo "si-entonces".

En este caso se utilizó un árbol tipo CART, el cual es un acrónimo de *Classification and Regression Trees*. Este tipo de árboles realiza tres tareas importantes: como dividir los datos en cada nodo, sabe cuando debe detenerse y cómo predecir el valor de y para cada partición de los datos de entrada.

La Figura 3.4 muestra el funcionamiento de un árbol CART, se ingresa un dato a la raíz. En la raíz del árbol se evalúa el dato y según el valor de un atributo se puede tomar el camino derecho o el camino izquierdo, sin importar el nodo al que llegue se vuelve a repetir el proceso. Al final, llega a alguna hoja del árbol en donde puede tener un valor de +1 o -1, dependiendo del cual es la categoría a la que pertenece el llanto.