RNPS No. 2142
ISSN 2072-6287
Versión Digital

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Reconocimiento de Patrones

SERIE AZUL

Métodos de clasificación de locutores utilizando clasificadores Boosting

Flavio J. Reyes Díaz,
Gabriel Hernández Sierra, y
José Ramón Calvo de Lara

RT_041 **febrero 2011**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Reconocimiento de Patrones

Métodos de clasificación de locutores utilizando clasificadores Boosting

Flavio J. Reyes Díaz,
Gabriel Hernández Sierra, y
José Ramón Calvo de Lara

RT_041 **febrero 2011**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Métodos de clasificación de locutores utilizando clasificadores Boosting

Flavio J. Reyes Díaz[1], Gabriel Hernández Sierra[2], y José Ramón Calvo de Lara[2]

[1] Dpto. Ingeniería en Sistemas, Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), Ciudad de La Habana, Cuba
freyes@cenatav.co.cu
[2] Dpto. Reconocimiento de Patrones, Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), Ciudad de La Habana, Cuba
{gsierra, jcalvo}@cenatav.co.cu

RT_041, Serie Azul, CENATAV
Aceptado: 14 de diciembre de 2010

Resumen. Los algoritmos de clasificación para el reconocimiento del locutor se han venido desarrollando hasta la actualidad. No obstante, aun no alcanzan una efectividad o eficiencia máxima, sobre todo en condiciones de ruido y ante variabilidad de canal o sesión. Debido a la redundancia existente en los modelos de los locutores, abordaremos el reconocimiento del locutor mediante el uso del clasificador Boosting para reducir esa redundancia y elevar la robustez. Aquí se recoge el estado del arte de la familia Boosting, con las variantes y mejoras existentes. También proveemos del estado de arte de métodos del reconocimiento del habla o del locutor donde se ha usado el método Boosting o alguna de sus variantes.

Palabras clave: boosting, adaboosting, variantes del boosting, reconocimiento del locutor.

Abstract. The classification algorithms for speaker recognition have been developed at the present time, however they don't reach maximum effectiveness and efficiency when facing noise and channel and session mismatch. Due to redundancy in speaker's models, we propose to use boosting techniques in speaker recognition to reduce redundancy and increase robustness. This report presents a state of the art of Boosting methods, with variants and existing improvements. We also provide a state of art in speaker and speech recognition where Boosting classifier or some of its descendants is used.

Keywords: boosting, adaboosting, boosting's variants, speaker recognition.

1 Introducción

El reconocimiento automático de personas por la voz es el proceso mediante el cual se verifica o identifica a un individuo utilizando una señal de voz desconocida.

Dentro del RAL[1] podemos distinguir dos tareas diferenciadas:

[1] Reconocimiento automático del locutor

- *Verificación Automática del Locutor (VAL).* El objetivo es verificar la identidad reclamada por el locutor, a partir de una muestra de voz con identidad desconocida y un individuo que clama ser el emisor de esta señal; la respuesta del sistema, por lo tanto será, binaria: identidad aceptada o rechazada.
- *Identificación Automática del Locutor (IAL).* Aquí el objetivo es, dada una muestra de voz, señalar, dentro de un grupo de personas, los propietarios más probables de la muestra.

Estos procesos se dividen en dos fases: el entrenamiento y la prueba (identificación o verificación), y en cada una de ellas se realiza la extracción de rasgos. Hasta la actualidad se han venido desarrollando algoritmos con enfoques estadísticos y discriminativos.

- Entrenamiento: Obtención de los modelos y umbrales correspondientes a cada uno de los integrantes de la población de locutores.
- Prueba: Comparación entre los modelos de la población y las muestras de la señal de voz analizada, que permitirá tomar decisiones acerca de la identidad del mismo.
- Extracción de rasgos: Consiste en obtener una representación matemática de los rasgos acústicos de la señal de voz conservando la información.

Estas fases internamente realizan una serie de pasos de gran importancia para el reconocimiento de las personas por la voz, los cuales son mencionados a continuación: Entrenamiento:

1. Extracción de rasgos
    a. Se adquiere la voz.
    b. Se extraen los rasgos acústicos.
    c. Se realiza la normalización de rasgos.
2. Aplicación de métodos de reconocimiento de patrones a los rasgos extraídos de las muestras, para entrenar al sistema**,** creando los modelos y clases necesarias para la posterior clasificación.

Prueba:

1. Extracción de rasgos
    a. Se adquiere la voz.
    b. Se extraen los rasgos acústicos.
    c. Se realiza la normalización de los rasgos.
2. Comparación de los rasgos extraídos con los modelos creados en la fase de entrenamiento.

En nuestro trabajo nos centraremos en las etapas de clasificación y prueba, sin introducirnos en el proceso de extracción de rasgos.

En la sociedad se han venido introduciendo aplicaciones para el reconocimiento de las personas por su voz y se han desarrollado una serie de trabajos donde para resolver el problema de reconocimiento se han utilizado una serie de clasificadores como son: la distorsión dinámica en el tiempo *(DTW)* [1, 2, 4, 6], cuantificación vectorial *(VQ)* [1, 2, 4, 6], los modelos ocultos de Markov *(HMM)* [2, 4, 6], los modelos de mezclas gaussianas *(GMM)* [4, 5, 6], adaptaciones de las GMM *(MAP)* [4, 5, 6] y las máquinas de soporte vectorial *(SVM)* [4, 5].

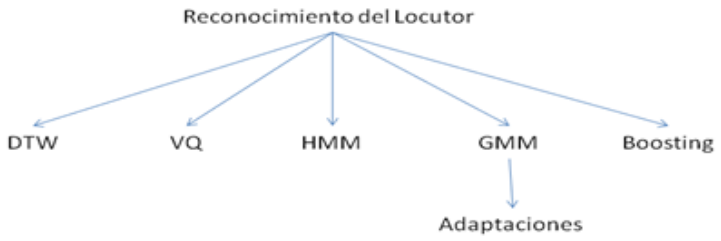

Fig. 1. Algoritmos de clasificación utilizados para el reconocimiento del locutor

A continuación describiremos brevemente el proceso de clasificación que se utiliza como línea base en los sistemas de reconocimiento del locutor en la actualidad.

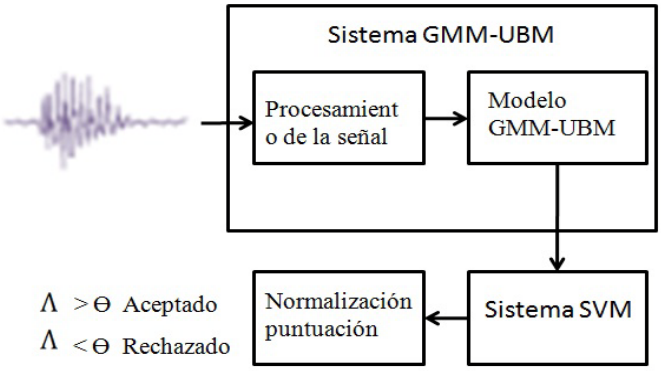

Fig. 2. Estructura de los sistemas GMM-SVM

La etapa de clasificación consiste en obtener los modelos del locutor, los súper vectores[2] y entrenar la máquina de soporte vectorial.

Sistema de adaptación del modelo del locutor al modelo universal de Background

Los *GMM* representan con un alto grado de fidelidad un amplio margen de distribuciones muestrales, tales como los diferentes coeficientes cepstrales [4] que puede generar una locución concreta.

La adaptación del modelo del locutor al modelo universal de Background *(GMM-UBM)* [5, 6] modela los distintos vectores de rasgos de una locución, realizando una suma ponderada o mezcla de funciones de densidad de probabilidad Gaussiana adaptadas del modelo universal de Background[3] *(UBM)* [4, 5, 6].

Como resultado se obtiene un modelo $\lambda = \{\bar{p}, \mu, \Sigma\}$ donde: $\bar{p}$ representa el vector de pesos, μ la matriz de medias y Σ la matriz de varianzas. Cada locutor es representado por un modelo λ obtenido utilizando *GMM-UBM* adaptado [2].

Cada modelo tendrá la misma cantidad de componentes Gaussianos que el UBM. Por lo general en los sistemas actuales son 512, 1024 o 2048 componentes. Esto hace que la matriz de medias μ tenga k componentes de la misma dimensión de los Coeficientes Cepstrales en frecuencia Mel *(MFCC)*.

Creación de súper vectores de medias

Los súper vectores son construidos a partir de una concatenación de todos los vectores de la matriz de media obtenidos en el entrenamiento del modelo utilizando *GMM-UBM*. Estos súper vectores tendrán la

[2] Son vectores de gran dimensión obtenidos a partir de las matrices de medias de los modelos del locutor.

[3] No es más que el modelo obtenido a partir de rasgos que nos representen a toda una población.

dimensión de la cantidad de componentes Gaussianos por la dimensión de los rasgos MFCC. Provocando la creación de un vector de gran dimensión, el cual representa al locutor.

Sistemas con máquina de soporte vectorial (GMM-UBM-SVM)

Para el entrenamiento de las máquinas de soportes SVM se utilizan como datos de entrada los súper vectores obtenidos en el paso anterior, estos no son más que un punto en un espacio de altas dimensiones, donde cada punto representa a un locutor. Se entrena utilizando dicha información de locutores clientes, y un grupo de súper vectores de impostores, ver [4].

En la prueba con otras locuciones de locutores clientes e impostores se obtienen los modelos del locutor a partir de la GMM-UBM adaptadas y luego se construyen los súper vectores, por último se utiliza la SVM entrenada para obtener los valores de similitud entre las señales de voz [4].

Teniendo en cuenta que los modelos obtenidos del GMM-UBM adaptado para el reconocimiento del locutor contienen todas las componentes Gaussianas del UBM, provoca una redundancia en la información contenida en el modelo de cada locutor, la cual afecta en ocasiones la efectividad de los clasificadores. Con la realización de una inteligente selección de las componentes Gaussianas que presenten mayor y menor verosimilitud sobre los rasgos del locutor desconocido y su posterior combinación, pudiera brindar una mejora en la efectividad y la eficiencia de los resultados del reconocimiento.

Para esto, las técnicas Boosting aplicadas al espacio de las mezclas Gaussianas sería una interesante línea de trabajo para encontrar una nueva combinación de los componentes Gaussianos, capaz de mejorar la efectividad de los resultados en el reconocimiento del locutor.

2 Métodos ensambladores

2.1 ¿Por qué métodos ensambladores?

Estos métodos consisten en combinar un conjunto de clasificadores para obtener mayor precisión de la que cada uno de éstos logra de manera individual. Cada método de clasificación para la obtención de los clasificadores débiles se basa en conceptos o procedimientos de estimaciones diferentes. Además, como todos los métodos de clasificación tienen algún punto fuerte o ventaja, es lógico intentar asociar las mejores propiedades de cada uno de ellos combinándolos de alguna manera. Para ello en ocasiones se trabaja con la clase etiquetada por los clasificadores individuales mientras que en otras, lo que se utiliza son las probabilidades condicionales, asignadas a cada clase por los distintos clasificadores.

Combinar la salida de varios clasificadores es útil únicamente si hay desacuerdo entre ellos, es decir si existe alguna diferencia entre ellos.

Obviamente combinando varios clasificadores idénticos no se obtiene ningún beneficio. Hansen y Salomon en [7] probaron que si la tasa de error media para una observación es menor del cincuenta por ciento y los clasificadores débiles utilizados en el universo son independientes en la producción de sus errores, el error esperado para una observación puede ser reducido a cero cuando el número de clasificadores combinado se acerca a infinito.

Por su parte, Krogh y Vedelsby en [8] probaron más tarde que el error conjunto puede dividirse en un término que mide el error de generalización medio de cada clasificador individual y un término que recoge el desacuerdo entre los clasificadores. Lo que demostraron formalmente fue que la combinación ideal consiste en clasificadores débiles con alta precisión que estén el mayor número de veces posible en desacuerdo.

Se considera únicamente las combinaciones con un número impar de clasificadores, como suele hacerse en la práctica para evitar empates. A medida que aumenta el número de clasificadores utilizados aumenta la precisión del conjunto para un valor dado de la precisión individual.

En [9] se plantean las siguientes tres razones que apoyan el uso de los métodos de combinación:

1. La primera razón denominada razón estadística. Un sistema de aprendizaje puede entenderse como la búsqueda dentro de un determinado espacio de hipótesis de aquella que sea la más adecuada. Si se combinan las distintas hipótesis, se pueden promediar los resultados y reducir el riesgo de elegir un clasificador que obtenga una menor precisión a la hora de generalizar.
2. En segundo lugar se considera una razón computacional. Viene causada por la propia naturaleza de determinados sistemas de clasificación que realizan algún tipo de búsqueda local, lo que puede provocar que estos queden atrapados en un óptimo local. Una combinación de clasificadores que realice una búsqueda local desde puntos iniciales distintos logrará una mejor aproximación a la hipótesis o función objetivo buscada, que cualquiera de los clasificadores individuales.
3. La tercera, la función de representación de un problema. En muchos problemas de clasificación la verdadera función no se puede representar mediante ninguna de las hipótesis existentes en el espacio de hipótesis disponibles (universo). Sin embargo, mediante combinaciones de hipótesis relativamente sencillas de ese espacio se puede llegar a una mejor aproximación a la función real mediante la expansión del espacio de funciones representables. Por ejemplo, utilizando una suma ponderada de las hipótesis simples.

Estas son las tres razones fundamentales en que nos apoyaremos para justificar la superioridad de la combinación de clasificadores sobre los clasificadores individuales, aunque en el caso que nos interesa, el reconocimiento del locutor, se utiliza la combinación de todos los clasificadores débiles obteniendo un clasificador fuerte, lo cual podemos verlo como la solución bruta, "aunque hasta ahora es la solución más utilizada en estos métodos". No se debe olvidar que algunos de estos algoritmos también presentan algunas desventajas como son la pérdida de comprensibilidad (una estructura más compleja y de mayor tamaño), la mayor lentitud en la construcción de la combinación (se necesita una mayor capacidad de memoria) y también un mayor costo computacional a la hora de clasificar una nueva observación.

2.2 Principales métodos

Dentro de los métodos de combinación de clasificadores que utilizan el re-muestreo[4], los más utilizados son los métodos Bagging y Boosting.

2.2.1 Métodos Bagging

El método Bagging fue propuesto por Breiman en [10] y se basa en los métodos de Bootstrapping [11] y de agregación [12]. Tanto los métodos de Bootstrapping como los de agregación presentan propiedades beneficiosas.

Bootstrapping realiza un re-muestreo que consiste en obtener muestras aleatorias con de igual tamaño que el conjunto original. Partiendo del conjunto de entrenamiento $X = (x_1, x_2, \dots, x_n)$, mediante la extracción aleatoria con re-muestreo con el mismo número de elementos que el conjunto original de n elementos, se obtienen B muestras bootstrap $X_b = (x_1^b, x_2^b, \dots, x_n^b)$ donde $b = 1, 2, .., B$. En algunas de estas muestras se habrá eliminado o al menos reducido la presencia de observaciones ruidosas, teniendo en cuenta que estos métodos realizan una selección de los rasgos más discriminativos, por lo que el clasificador construido en ese conjunto presentará un mejor comportamiento que el clasificador construido en el conjunto original. Así pues Bagging puede ser útil para construir un mejor clasificador cuando el conjunto de entrenamiento presente observaciones ruidosas.

La combinación de clasificadores, obtiene frecuentemente mejores resultados que los clasificadores individuales utilizados para construir el clasificador final. Esto puede entenderse si se considera que al combinar los clasificadores individuales se están combinando las ventajas de cada uno de ellos en un solo clasificador.

---

[4] Las técnicas de re-muestreo pueden utilizarse para extraer muestras distintas a partir del conjunto de datos original de tal forma que los clasificadores individuales entrenados en cada una de ellas sean utilizados posteriormente en la combinación.

En concreto, el método Bagging se aplica del siguiente modo:

Datos de entrada: Conjunto de entrenamiento: $X = (x_1, x_2, \dots, x_n)$ y $b = 1,2, \dots, B$

1 Repetir para $b = 1,2, \dots, B$.
   a. Realizar una réplica bootstrap X_b del conjunto de entrenamiento X.
   b. Construir un clasificador sencillo $C_b(x)$ en X_b (con un umbral de decisión igual a $C_b(x) = 0$.
2 Combinar los clasificadores básicos $C_b(x)$, b=1,2,...B usando el voto mayoritario (la clase predicha más frecuente) en la regla de decisión final

$$\beta(x) = \arg \max_{y \in \{-1,1\}} \sum_b \delta_{sgn}(C_b(x), y) \text{ donde } \delta(i,j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Las diferentes versiones del Bagging han sido estudiadas por varios investigadores, aunque quizás con más frecuencia en el ámbito de la regresión que desde el punto de vista de la clasificación. En [10] se muestra que Bagging puede reducir el error tanto de regresión como de clasificación en los árboles de decisión, aunque es beneficioso para clasificadores inestables. Mientras que para procedimientos estables puede incluso llegar a perjudicar el comportamiento del clasificador, como por ejemplo, en el método de clasificación del vecino más cercano. En [13] el autor defiende que la estabilidad de los clasificadores lineales depende del tamaño del conjunto de entrenamiento. Por lo tanto, no se podría decir que Bagging no es útil para un determinado clasificador, sino que dependerá de la aplicación concreta a la que se enfrente.

2.2.2 Método Boosting

El método Boosting [14, 15] se encuadra dentro de lo que se ha dado a conocer como métodos *"ensemble"* (de combinación).

Para este método de combinación de clasificadores es necesario tomar dos decisiones fundamentales. La primera es como se obtiene el conjunto de clasificadores (universo de hipótesis).

Una de las maneras más naturales de obtener el conjunto de clasificadores es utilizando varios métodos de aprendizaje distintos, para este caso nos encontramos ante un método de combinación de clasificadores heterogéneos.

También es posible combinar clasificadores homogéneos, que son todos los clasificadores obtenidos utilizado un mismo método, al que se denomina clasificador débil. Para que la combinación de clasificadores tenga sentido, es necesario que los clasificadores a combinar difieran en sus predicciones.

La segunda: es como se combinan dichos clasificadores. Esta etapa se divide en dos ramas:

- Votaciones: Se consideran las predicciones de cada clasificador débil como un voto, y la predicción del clasificador combinado será el resultado de dicha votación. El proceso puede ser simplemente asignar la clase con más votos, pero se puede ponderar el voto de cada clasificador débil de acuerdo a algún criterio o tener en cuenta también la confianza que cada clasificador tiene en sus propias predicciones.
- Stacking [16]: Se utiliza un método de aprendizaje adicional cuya entrada son los resultados de los clasificadores débiles.

Boosting es un método de combinación de clasificadores homogéneos por votación.

Descripción del método Boosting

Sea X el dominio. Un concepto es una función booleana $c: X \to \{-1, +1\}$.

Un algoritmo de aprendizaje tiene acceso a un conjunto de muestras etiquetadas $(x, c(x))$ denominado conjunto de entrenamiento, donde x es escogido aleatoriamente acorde a una distribución fija pero desconocida del dominio X y c es la etiqueta correspondiente a la muestra. El algoritmo debe retornar una hipótesis o clasificador:

$$h(x): X \to \{-1, +1\} \quad (1)$$

El error de *h, e,* es medido por la probabilidad de que $h(x) \neq c(x)$

Un algoritmo fuerte es un algoritmo de aprendizaje que dado $e, \delta > 0$ y acceso a muestras, es capaz, con una probabilidad $1 - \delta$ de devolver una hipótesis h con un error no mayor que e.

Un algoritmo débil cumple las mismas condiciones pero solo para $e \geq \frac{1}{2} - \gamma$, donde γ es un valor constante o que disminuye de manera polinomial con respecto a parámetros significativos, como la cantidad de elementos de la muestra.

Esto quiere decir que mientras que el algoritmo fuerte debe generar una hipótesis con un error arbitrariamente pequeño, el débil solo puede generar hipótesis con errores "algo menores" que ½, que es la elección aleatoria, en caso que se tengan dos clases.

La tarea del Boosting es estimar una función $F: X \rightarrow \{-1, +1\}$, utilizando como base un algoritmo de aprendizaje, de la siguiente manera:

- Recibe un conjunto de entrenamiento donde $x \in X$ y c toma los siguientes valores $\{-1, +1\}$ representando la clase a que pertenece cada muestra.
- Iterar desde $t = 1$ hasta T, donde T es el parámetro de la cantidad de iteraciones deseadas.
    - Entrenar un clasificador débil mediante el algoritmo de aprendizaje base en función de la distribución y se obtiene una hipótesis $h_t: X \rightarrow \{-1, +1\}$.
    - Se calcula el error e_t de la hipótesis.
    - Se actualiza la distribución en función de e_t.
- Combinar los clasificadores débiles obtenidos en función de la regla de combinación escogida $h_t(i = 1 \ldots T)$.

El objetivo final es, por supuesto, que el error de la combinación de las T hipótesis sea menor que el de cualquiera de éstas por separado.

No obstante Boosting presenta una serie de dificultades dentro de las cuales se encuentran:

- Es susceptible al ruido aditivo.
- El rendimiento en un problema particular, es dependiente de los datos y el clasificador débil.
- Necesita a priori el conocimiento de la etiqueta y asociada a cada ejemplo x.
- Es un método enfocado a la clasificación binaria $c = 2$.
- No es resistente al sobreajuste.
- El algoritmo Boosting requiere que los clasificadores bases trabajen con los pesos.

Presenta agotamiento *(underflow)* demostrado por Bauer y Kohavi en [17], los cuales plantean que el problema aparece cuando se tienen instancias bien clasificadas en bastantes iteraciones. Si, además, el error de clasificación es pequeño, entonces las instancias bien clasificadas en t iteraciones verán reducido su peso en aproximadamente un factor 2^t, lo que puede llevar al agotamiento.

2.3 Comparación entre los métodos Bagging y Boosting

Ambos métodos construyen los clasificadores débiles utilizando versiones modificadas del conjunto de entrenamiento y los combinan después con la regla de clasificación final. Sin embargo, se diferencian en el modo de obtener los clasificadores básicos. En concreto las dos principales diferencias son:

1. El Boosting utiliza todas las muestras del conjunto de entrenamiento y no utiliza la técnica Bootstrapping.
2. El clasificador obtenido en cada paso del Boosting depende de todos los anteriores, mientras que en Bagging son independientes.

Características del Bagging

- Genera diversos clasificadores sólo si los algoritmos de aprendizaje base son inestables, es decir, que pequeños cambios en el conjunto de entrenamiento producen grandes cambios en el clasificador final.

- Los modelos están correlacionados.
- La Varianza de modelos entrenados con menos casos de entrenamiento generalmente es mayor.
- Los modelos estables tienen varianzas bajas así que Bagging puede no ayudar mucho.
- Cada clasificador débil se entrena con menos datos, aproximadamente 63.2% de los datos.
- Bagging reduce la varianza al promediar y tiene poco efecto en sesgo.

Características de Boosting

- El funcionamiento de la metodología Boosting en cualquier conjunto de aprendizaje es mejor que la predicción aleatoria (> 50%).
- Se puede utilizar cualquier clasificador débil como: árboles de decisión, redes neuronales, etc.
- Funciona bien con modelos estables.
- El re-muestreo es mucho mejor que Bagging.

3 Familia de clasificadores Boosting (Extensiones y nuevos métodos a partir del Boosting)

3.1 Adaboosting

La variante más usada del método Boosting es Adaboost (Adapting Boosting) [18, 19]. Desarrollada por Schapire y Freund en 1996 con el objetivo de mantener una distribución $(D_t(i))$ o conjunto de pesos $w_{(i)}$ sobre la muestra i en la iteración t. Cuya función es penalizar los ejemplos mal clasificados y así obligar al clasificador débil a enfocarse en estos ejemplos, más difíciles de clasificar. Este algoritmo genera al igual, que el algoritmo Boosting, un conjunto de clasificadores débiles para su posterior combinación, con la diferencia que Adaboosting los genera secuencialmente.

Adaboosting puede utilizar enfoques como:

1. Seleccionar un conjunto de ejemplos basados en las probabilidades de los ejemplos.
2. Usar todos los ejemplos y el peso (que se actualiza dinámicamente) del error de cada ejemplo de la probabilidad (es decir, ejemplos con mayores probabilidades de tener más efecto sobre el error).

Este último enfoque tiene la ventaja evidente que cada ejemplo se incorpora (al menos en parte) en el conjunto de entrenamiento. Además, los autores han demostrado que esta forma de Adaboosting puede ser vista como una forma de modelado aditivo para optimizar la pérdida de una función logística.

Este algoritmo resuelve muchas de las dificultades prácticas de los anteriores algoritmos Boosting. El algoritmo toma como datos de entrada para el entrenamiento a $(x_1, y_1), \dots, (x_m, y_m)$, donde cada x_i pertenece a un dominio o espacio X, y cada y_i pertenece al dominio $Y = \{-1, +1\}$. Adaboost llama a un algoritmo de aprendizaje débil o al algoritmo de aprendizaje base en una serie de iteraciones $t = 1, \dots, T$. Una de las ideas principales del algoritmo es mantener una distribución o conjunto de pesos en el conjunto de entrenamiento. El peso de la distribución en el modelo i del entrenamiento en la iteración t se denota $D_t = (i)$. Inicialmente, todos los pesos se establecen por igual, pero en cada iteración, los pesos de los modelos que son clasificados de forma incorrecta se incrementan de modo que el clasificador débil se ve obligado a concentrarse en los modelos duros en el conjunto de entrenamiento.

El trabajo del clasificador débil es encontrar una hipótesis $h_t: X \to \{-1, +1\}$ apropiada para la distribución D_t. La confidencialidad de la hipótesis débil es medida a través del error:

$$e_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_i(i) \tag{2}$$

Observe que el error se mide en relación con el D_t de distribución en la que el clasificador débil fue entrenado. En la práctica, el clasificador débil puede ser un algoritmo que puede utilizar los pesos D_t en los modelos de entrenamiento. En cambio, cuando esto no sea posible, un subconjunto de los modelos de entrenamiento se puede probar según el D_t, y estos modelos se pueden utilizar para entrenar a los clasificadores débiles.

Descripción del método Adaboosting

Este algoritmo fija la probabilidad de la distribución y toma inicialmente a $1/N$ donde N es el número de muestras de entrenamiento. Este método calcula nuevamente las probabilidades después de cada clasificador entrenado, y se añade al conjunto. En Adaboosting, el error E_k es la suma de las probabilidades de los casos mal clasificados para el clasificador actualmente entrenado C_k. Las probabilidades para la siguiente prueba son generadas por la multiplicación de las probabilidades de C_k 's mal clasificadas por el factor $B_k = (1 - E_k)/E_k$ y re-normalizadas las probabilidades para que su suma sea igual a 1. Adaboosting combina los clasificadores $C_1, \dots, C_k$ mediante el peso seleccionado, donde C_k tiene el registro de peso (B_k). Estos pesos permiten al Adaboosting la predicción de los clasificadores que no son muy precisos sobre el problema general.

Pseudocódigo del algoritmo:

1. Datos de entrada: Muestras de entrenamiento: $(x_1, y_1), \dots, (x_N, y_N)$ donde $x_i \in X,\ y_i \in Y = \{-1, +1\}$
2. Inicialización: $w_i(i) = 1/N$
3. Desde $t = 1\ hasta\ T$ donde T es el número de iteraciones.
   a. Se obtienen el o los clasificadores débiles $h_t : X \to \{-1, +1\}$ con un error, $\varepsilon_t = Pr[h_t(x_i) \neq y_i]$ basados sobre la distribución w_t.
   b. Se elige $\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
   c. Se actualizan los pesos o distribuciones: $w_{t+1}(i) = \frac{w_t(i)}{Z_t} \times \begin{cases} e^{-\propto} & si\ h_t(x_i) = y_i \\ e^{\propto} & si\ h_t(x_i) \neq y_i \end{cases}$ $donde\ Z_t = \sum_{i=1}^{N} w_t(i)$
4. Salida: Hipótesis final: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Este algoritmo calcula el error de entrenamiento ε, que es el error de h en el conjunto de entrenamiento, de la forma como se demuestra en [15], donde se plantea que ε esta acotado por:

$$\prod_t \left[\sqrt[2]{e_t(1-e_t)}\right] = \prod_t \sqrt{1 - 4\gamma_t^2} \leq exp\left(-2\sum_t \gamma_t^2\right) \tag{3}$$

Teniendo en cuenta que $e_t = 1/2 - \gamma_t$, y siendo $\gamma_t \geq \gamma$ para algún $\gamma > 0$, trae consigo que: e cae exponencialmente a 0. Obteniendo un resultado independiente de γ_t, incluso si esta cambia de una distribución a otra, convirtiéndose en una de principales ventajas del algoritmo.

En [19] los autores mostraron una característica importante, donde el error en los conjuntos de prueba continuaba descendiendo en la medida que se añadían más clasificadores débiles, incluso cuando el error de entrenamiento se encontraba muy cerca de cero. En respuesta a esta cuestión, exponen un importante análisis sobre las propiedades del error de generalización del AdaBoost, en términos de los márgenes en el conjunto de entrenamiento.

El margen de una muestra (x, y) se define como

$$m(x, y) = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \tag{4}$$

Nótese que es un número entre $[-1,+1]$ que es positivo si y solo si h clasifica correctamente la muestra. Esta magnitud puede ser interpretada como una medida de la confianza en la predicción. Los autores demostraron que el error de generalización puede ser acotado por

$$\widehat{Pr}\,[m(x,y)<\theta]+\ \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right) \tag{5}$$

para cualquier $\theta > 0$ con una gran probabilidad.

Este algoritmo puede combinar cualquier clasificador con el objetivo de obtener la hipótesis débil como por ejemplo: redes neuronales [20], árboles de decisión [21], entre otros.

3.2 Extensiones para la solución de los problemas

A partir del surgimiento del algoritmo de clasificación Adaboosting se han venido desarrollando una serie de variantes, modificaciones y nuevos métodos con el propósito de mejorar su funcionamiento y rendimiento, utilizarlo en diferentes casos o aplicaciones de la vida real y con esto resolver una serie de problemas presentes en el mismo como son:

- Problemas de regresión.
- Problemas ante la presencia de múltiples clases.

Teniendo en cuenta que Adaboosting presentaba problemas ante la presencia de múltiples clases y la regresión, Freund y Schapire [15] le incluyeron algunas mejoras al mismo, y surgen dos extensiones del Adaboosting: una para el caso de varias clases en la cual Y es cualquier conjunto finito de clases y otra para los problemas de regresión donde Y es un intervalo definido real.

3.2.1 Problema de múltiples clases

Teniendo en cuenta que hasta el momento solo se había trabajado sobre problemas de clasificación binaria, Schapire y Freund en [18, 19] nos brindan dos posibles extensiones para la resolución de problemas de múltiples clases, es decir la presencia de una clasificación para un dominio $Y > 2$ donde Y es el numero de etiquetas de clases. Estos dos algoritmos son iguales ante una clasificación binaria, solo difieren en el caso de $Y > 2$.

1. Adaboosting.M1
2. Adaboosting.M2

Adaboosting.M1

Este algoritmo toma como datos de entrada un conjunto de entrenamiento de m muestras $X = \{(x_1, y_1), \dots, (x_m, y_m)\}$ donde x_i es la instancia extraída de un espacio X y representa de alguna manera un vector de valores de atributos y $y_i \in Y$ es la etiqueta de la clase asociada con x_i.

Adaboosting.M1 tiene acceso al igual que Boosting a un algoritmo débil no específico, al cual lo llama repetidamente en varias iteraciones. En cada iteración la extensión le da al algoritmo débil la distribución D_t sobre el conjunto de entrenamiento S. En respuesta a esto el algoritmo o clasificador débil devuelve una hipótesis $h_t: X \to Y$ que debe clasificar correctamente una fracción del conjunto de entrenamiento que tiene gran probabilidad con respecto a la D_t.

El objetivo del clasificador débil es encontrar una hipótesis $h_t$que minimice el error de entrenamiento

$$e_t = Pr_{i\sim p^t}[h_t(x_i) \neq y_i] \tag{6}$$

Este error es medido con respecto a la distribución D_t que se le dio con anterioridad al clasificador débil. El proceso se repite durante T iteraciones, obteniendo finalmente un grupo de hipótesis $h_1, \dots, h_T$ las cuales son combinadas para la obtención del clasificador fuerte.

Pseudocódigo del algoritmo Adaboost.M1

Datos de entrada:

- Secuencias de m muestras de la forma: $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$ donde las etiquetas $y_i \in Y = \{1, \dots, k\}$
- Algoritmo Débil: Algoritmo de aprendizaje base.
- T es el número de iteraciones.

1. Inicializa la distribución $D_1(i) = 1/m$ para toda i.
2. Se realiza un ciclo desde $t = 1 \; hasta \; T$
   a. Se llama al Clasificador débil previéndole con la distribución D_t
   b. Se regresa a la hipótesis $h: X \to Y$
   c. Se calcula el error de h_t: $e_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$

Si $e_t > \frac{1}{2}$ entonces $T = t - 1$ y termina el ciclo.

   d. Se determina $\beta_t = e_t/(1 - e_t)$
   e. Actualización de las Distribuciones D_t

$$D_{t+1(i)} = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & si \; h_t(x_i) = y_i \\ 1 & otherwide \end{cases}$$

Donde Z_t es un valor constante de normalización.

Salida:

- Hipótesis final

$$h_{fin}(x) = arg \max_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1}{\beta_t}$$

Varios aspectos no especificados son: (a) la manera de calcular las distribuciones D_t en cada iteración y (b) como se obtiene el clasificador fuerte. Para responder estas dos interrogantes los algoritmos Boosting presentan diferentes maneras. El Adaboosting.M1 usa una simple regla la cual se observa en el pseudocódigo del algoritmo descrito anteriormente, donde la inicialización de la D_t es uniforme sobre S de la forma $D_1(i) = 1/m$ para todas las i y se calcula la distribución D_{t+1} a partir de la distribución D_t y de la última hipótesis débil, que se multiplica su peso por un numero $\beta_t \in [0, 1)$ si h_t clasifica correctamente a x_i, de lo contrario el peso no se modifica. El número β_t se calcula basándose en el valor del error como se muestra en el paso 2.d.

La propiedad teórica más importante de este algoritmo está dada por que si la hipótesis débil presenta un error ligeramente menor que ½, el error de la hipótesis final decrece exponencialmente a cero de forma rápida.

Adaboosting.M2

Este algoritmo es otra extensión del algoritmo Adaboosting para el problema de múltiples clases. Esta extensión se basa en extender la comunicación entre el algoritmo Boosting y el algoritmo de aprendizaje débil, lo que trae consigo la ventaja de que el clasificador débil es mucho más flexible para tomar sus predicciones y permite que este pueda hacer contribuciones útiles a la exactitud de la hipótesis final, incluso cuando la hipótesis débil no predice la etiqueta correcta con una probabilidad mayor que ½. En primer lugar, permitir que el aprendiz débil genere una hipótesis más expresiva, es decir, que en lugar de identificar una sola etiqueta en *Y,* elegir un conjunto plausible de etiquetas.

En este algoritmo los aprendices débiles generan sus hipótesis de la forma:

$$h: X \times Y \to [0,1] \qquad (7)$$

En este caso cada hipótesis débil genera un vector $[0,1]^k$ donde k es el número de clases y cada componente con valores finales 1 o 0 corresponden a las etiquetas consideradas estimables y no

estimables respectivamente. Este vector de valores no es el vector de probabilidades, es decir no necesitan que la suma de los valores sea 1.

Al ser el algoritmo de aprendizaje débil más expresivo, también existe un requisito más complejo en la creación de las hipótesis débiles. Es decir que en lugar del error de predicción normalmente usado, le pedimos que la hipótesis débil obtenga un error más sofisticado, medida que se denomina Seudo-pérdida.

Esta medida del error a diferencia del error común, es que se calcula con respecto a una distribución sobre los ejemplos, la Seudo-pérdida se calcula con respecto a una distribución en el conjunto de todos los pares de ejemplos y las etiquetas incorrectas. Mediante la manipulación de esta distribución, el algoritmo Boosting en el aprendizaje puede centrarse no sólo en los ejemplos débiles o difíciles de clasificar, sino más específicamente, en las etiquetas incorrectas que son más difíciles de discriminar.

Una mejor descripción de este algoritmo es que las etiquetas incorrectas son representadas por el par $(i, y))$ donde i es el índice de los muestra a entrenar y y es la etiqueta incorrecta asociada al ejemplo i. Y B es el conjunto de todas las etiquetas incorrectas:

$$B = \{(i, y): i \in \{1, \dots, m\}, y \neq y_i\}. \tag{8}$$

Pseudocódigo del algoritmo Adaboost.M2

Datos de entrada

- Secuencias de m muestras de la forma $\langle (x_1, y_1), \dots, (x_m, y_m)\rangle$ con etiquetas $y_i \in Y = \{1, \dots, k\}$.
- Algoritmo para la obtención de los clasificadores débiles.
- T=Valor entero que representa el número de iteraciones.
- $B = \{(i, y): i \in \{1, \dots, m\},\ y \neq y_i\}$

1 Se inicializan las distribuciones:

$$D_1(i, y) = {}^{1}/_{|B|} \ para\ (i, y) \in B$$

2 Hacer desde $t = 1\ \ hasta\ T$

a. Se llama al clasificador débil, al cual se le provee las D_t mal etiquetadas.
b. Se regresa a la hipótesis $h_t : X \times Y \rightarrow [0, 1]$
c. Se calculan las perdidas falsas de h_t:

$$\epsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i, y)\big(1 - h_t(x_i, y_i) + h_t(x_i, y)\big)$$

d. Se determina $\beta_t = e_t/(1 - e_t)$
e. Se actualizan las distribuciones D_t

$$D_{t+1}(i, y) = \frac{D_i(i, y)}{Z_t} \cdot \beta_t^{({}^{1}/_{2})(1 + h_t(x_i, y_i) - h_t(x_i, y))}$$

Donde Z_t es la constante de normalización (elegida de tal manera que D_{t+1} será una distribución).

Salida:

- Hipótesis final

$$h_{fin}(x) = arg \max_{y \in Y} \sum_{t=1}^{T} \left(log \frac{1}{\beta_t}\right) h_t(x, y)$$

Como se muestra en el pseudocódigo del algoritmo Adaboost.M2, donde en cada iteración $\boldsymbol{t}$ este algoritmo le prevé al clasificador débil una distribución de las etiquetas incorrectas $\boldsymbol{D_t}$ de la forma:

$$D_1(i, y) = {}^{1}/_{|B|}\ para\ (i, y)\ \in B \tag{9}$$

y se realiza la llamada al algoritmo de entrenamiento base pasándole la distribución D_t como parámetro. La hipótesis débil se obtiene de la forma $h_t: X \times Y \rightarrow [0,1]$ de la cual se le calcula la Seudo-perdida de la forma como se muestra en el paso 2.c del seudo código, esto no es una restricción para $\sum_t h_t\,(x, y)$. En particular la predicción del vector no tiene que definir una distribución de probabilidad.

La hipótesis débil es interpretada de la siguiente manera: si $h_t(x_i, y_i) = 1$ y $h_t(x_i, y) = 0$, entonces $h_t$predice correctamente que la etiqueta de x_i es y_i, no y. Esta interpretación llevo a definir la Seudo-pérdida de la hipótesis h_t con respecto a la distribución mal etiquetada de la siguiente manera:

$$e_t = \frac{1}{2} \sum_{(i,y)\epsilon\, B} D_t\,(i,y)\big(1 - h_t(x_i, y_i) + h_t(x_i, y)\big) \tag{10}$$

Se puede verificar sin embargo, que la Seudo-pérdida se minimiza cuando las etiquetas correctas y_i se le asignan valor 1 y las etiquetas incorrectas $y \neq y_i$ se le asigna 0. Además la Seudo-perdida ½ es trivialmente alcanzada por cualquier hipótesis h_t de valor constante.

En este algoritmo el objetivo principal del clasificador débil es encontrar una hipótesis débil con pequeña Seudo-perdida.

Nota: posteriormente a que este algoritmo obtiene la hipótesis débil h_t y se va a actualizar las distribuciones se utiliza los mismos pasos del algoritmo Adaboosting.M1. Y solo requiere de un valor de Seudo-perdida ligeramente menor que ½.

3.2.2 Problemas de regresión

En esta extensión denominada Adaboosting.R el espacio de los valores de las etiqueta esta dado por $Y = [0, 1]$, como en las extensiones anteriores este algoritmo recibe muestras (x, y) elegidas aleatoriamente según las distribuciones y el objetivo es encontrar una hipótesis $h: X \to Y$ que con cierto valor de x prediga aproximadamente el valor de y que pueda ser visto. Con más precisión el aprendiz trata de encontrar una hipótesis con pequeño error cuadrático medio.

$$\varepsilon_{(x,y)\sim \mathrm{D}}[(h(x) - y)^2] \tag{11}$$

Estos métodos pueden aplicarse a cualquier medida razonable de error acotado, pero, se concentraron para esta variante en la medida de error cuadrático.

Inicialmente este algoritmo le provee al algoritmo de clasificación un conjunto de entrenamiento $(x_1, y_1), \ldots, (x_N, y_N)$ de muestras distribuidas acorde la distribución D y se enfocaron solo en la disminución empírica del error cuadrático medio *(MSE)*:

$$\frac{1}{N}\sum_{i=1}^{N}(h(x_i) - y_i)^2 \tag{12}$$

Enfocando el algoritmo Boosting en este contexto se logra reducir el problema de regresión dado, a un problema de clasificación binaria y posteriormente se le aplica el algoritmo Adaboost.

Para cada muestra (x_i, y_i) en el conjunto de entrenamiento, se definieron índices continuos por pares (i, y) para todas las $y \in [0, 1]$: la instancia asociada es $\tilde{x}_{i,y} = (x_i, y)$ y la etiqueta es $\tilde{y}_{i,y} = [\![y \geq y_i]\!]$. Cada hipótesis débil $h: X \to Y$ es reducida a hipótesis de valores binarios $\tilde{h}: X \times Y \to \{0,1\}$ la cual está definida de la forma:

$$\tilde{h}(x, y) = [\![y \geq h(x)]\!] \tag{13}$$

Por lo tanto $\tilde{h}$ responde las preguntas binarias de manera natural con valores estimados.

Pseudocódigo del algoritmo Adaboosting.R:

Datos de entrada

- Secuencias de m muestras $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ con etiquetas $y_i \in Y = [0, 1]$.
- Distribución D sobre las muestras.
- Algoritmo denominado clasificador débil.

- Valor entero que representa el número de iteraciones.

1 Inicializar el vector de pesos

$$w_{i,y}^{t} = \frac{D_{(i)}|y - y_i|}{Z}$$

2 Desde $i = 1, \dots, N,\ y \in Y$ donde

$$Z = \sum_{i=1}^{N} D_{(i)} \int_0^1 |y - y_i|\, dy.$$

3 Hacer desde $t = 1\ hasta\ T$

a. Se establece la densidad $p^t = \frac{w^t}{\sum_{i=1}^{N} \int_0^1 w_{i,y}^t dy}$

b. Se realiza la llamada al clasificador débil pasándole la densidad p^t que devuelve la hipótesis $h_t: X \times Y$.

c. Se calcula la perdida de h_t:

$$e_t = \sum_{i=1}^{N} \left| \int_{y_i}^{h_t(x_i)} p_{i,y}^t dy \right|$$

Si $\varepsilon_t > \frac{1}{2}$, se establece $T = t - 1$ y se aborta el ciclo.

d. Se determina $\beta_t = e_t/(1 - e_t)$.

e. Se determina el nuevo vector de pesos

$$w_{i,y}^{t+1} = \begin{cases} w_{i,y}^{t} & \text{si } y_i \le y \le h_t(x_i) \text{ o } h_t(x_i) \le y \le y_i \\ w_{i,y}^{t}\beta_t & En\ cualquier\ otro\ caso \end{cases}$$

Salida:

- Hipótesis fuerte

$$h_f(x) = inf\left\{ y \in Y: \sum_{t:h_t(x) \le y} log(1/\beta_t) \ge \frac{1}{2} \sum_{t} log(1/\beta_t) \right\}$$

Una gran desventaja de este algoritmo es que no hay manera trivial de generar una hipótesis cuya pérdida sea menor de ½, no obstante se podría solucionar obteniendo una hipótesis mediante dos funciones: $h: X \to [0, 1]$ y $k: X \to [0, 1]$ la cual está asociada a una medida de confianza para cada predicción de la hipótesis.

3.3 Mejoras y generalizaciones del algoritmo Adaboosting

3.3.1 Discrete Adaboosting

Este algoritmo desde el punto de vista teórico, muestra las propiedades originales del Boosting y, además, trae interesantes mejoras computacionales y numéricas que hacen significativamente más fácil su manejo. Conceptualmente hablando, se prevé una nueva reducción y atractivo a los$\mathbb{R}$. Tal vez la más popular es un mecanismo de peso iterativo modificado, de acuerdo con los ejemplos que han disminuido su peso si y sólo si reciben la clase correcta, por hipótesis.

La generalización de los valores reales hace que los pesos de las muestras que han disminuido su error sólo afectan a las muestras en los que el margen de la hipótesis excede de su margen promedio. Así, mientras que las dos propiedades coinciden en el caso discreto, las muestras que reciben la clase aún pueden ser mejor re-ponderadas con valores reales de hipótesis débil. Desde el punto de vista experimental, la generalización muestra la capacidad de producir fórmulas de bajo error con particulares distribuciones de margen acumulado, y que proporciona un manejo agradable de los dominios de ruido que representan el talón de Aquiles para los algoritmos de adaptación. Este algoritmo realiza una adaptación de la actualización de los pasos de Newton para minimizar los pesos y puede ser interpretada como un procedimiento de estimación para el ajuste de los modelos de regresión logística aditiva.

En esta variante del Adaboost la forma de actualizar los pesos es la siguiente:

$$D_{k+1}(i, y) = \frac{D_k(i, y)}{Z_k} \beta_k^{(1/2)(1 + h_k(x_i, y_i) - h_k(x_i, y))} \tag{14}$$

donde Z_k es una constante de normalización elegida de modo que la suma de los pesos siga valiendo la unidad.

Después los pesos son re normalizados, los ejemplos sencillos (los cuales son correctamente clasificados por las primeras reglas débiles) van quedando en un segundo plano, para darles una mayor importancia (con pesos mayores) a aquellas muestras que resultan difíciles, y que por tanto suelen ser incorrectamente etiquetadas por el clasificador.

Pseudocódigo del algoritmo:

Datos de entrada

- Datos de entrenamiento→ D
- Numero de iteraciones→ T
- Distribución inicial sobre las muestras→$\mathcal{W}_0(i)$

1 Hacer desde $t = 1\ hasta\ T$

a. Entrenar el clasificador débil $h_t = arg\ min_h \sum_{i=1}^{n} \mathcal{W}_{t-1}(i)\ 1\big(y_i \neq h(x_i)\big)$.

b. Calcular el error de la hipótesis débil (h_t): $\epsilon_t = \sum_{i=1}^{n} \mathcal{W}_{t-1}(i) 1\big(y_i \neq h_t(x_i)\big)$.

c. Se determina $\alpha_t = \frac{1}{2} log\ \frac{1-\epsilon_t}{\epsilon_t}$

d. Se determina $\mathcal{W}_{t+1}(i) = \begin{cases} \frac{\mathcal{W}_t(i)}{2(1-\epsilon_t)} & si\ y_i = h_{t+1}(x_i) \\ \frac{\mathcal{W}_t(i)}{2\epsilon_t} & si\ y_i \neq h_{t+1}(x_i) \end{cases}$

Salida

- Clasificador fuerte → $H(x) = \sum_{k=1}^{T} \alpha_k h_k(x)$.

La importancia de este algoritmo reside en el teorema que demuestra que si el error de cada una de las hipótesis débiles es sólo ligeramente inferior a 0.5, entonces el error de la hipótesis final convergerá exponencialmente a cero.

3.3.2 Real Adaboosting

Esta variante del Adaboosting es similar a la anterior con una sola diferencia en la forma de actualización de los pesos, que es la siguiente:

$$D_{t+1}(i, y) = D_t(i, y) exp\big(y_i \cdot\ h_k(x_i, y_i)\big) \tag{15}$$

Siempre con la precaución de normalizar todos los pesos tras actualizarlos posterior a la primera iteración. Para de esta forma obtener una distribución discreta de los pesos.

3.4 Logitboost

Dado que el error cuadrático no es una buena opción para la clasificación [22] provoca que el ajuste de residuales no funcione bien para ese caso. También se demostró que Adaboosting se adapta a un modelo aditivo mejor usando una función de pérdida para la clasificación, para mayor especificación, muestran que Adaboosting se ajusta a un modelo de regresión logístico aditivo, utilizando un criterio similar, pero no el mismo que el del logaritmo de la verosimilitud binomial. Por tal motivo desarrollaron un nuevo procedimiento del Boosting "Logitboost", que optimiza directamente el logaritmo de la verosimilitud binomial y ajusta los modelos de regresión logísticos aditivos mediante los pasos de Newton.

El algoritmo se basa sobre un modelo de regresión logística aditiva de los datos de entrenamiento. Y el modelo aditivo es una aproximación a la función $F_{(x)}$ de la forma:

$$F_{(x)} = \sum_{m=1}^{M} c_m f_m(x) \tag{16}$$

donde c_m es la constante a determinar y f_m es la función básica.

Si $F(x)$ es el mapeo adecuado que se busca entonces es la hipótesis fuerte agregada y f_m es nuestra hipótesis débil, luego se muestra que esta ajustado al modelo por la minimización del criterio:

$$J(F) = E\left(e^{-yF(x)}\right) \tag{17}$$

donde y es la etiqueta de la clase verdadera. Este algoritmo minimiza el criterio mediante el uso de los pasos Newton para adaptar el modelo de regresión logística aditiva para una directa optimización del logaritmo de la verosimilitud binomial.

$$-log\left(1 + e^{-2yF(x)}\right) \tag{18}$$

En este algoritmo la variable y toma valores de 0 y 1 como resultado final y representa la probabilidad de la forma $y = 1$ para $p(x)$, donde $p(x)$ se obtiene de la forma:

$$p(x) = \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}} \tag{19}$$

Pseudocódigo del algoritmo Logitboost para la clasificación binaria:

1. Inicialización de los datos
   $w_i = 1/N \;\; i = 1,2,\dots,N, \quad F(x) = 0$ y la probabilidad se estima $p(x_i) = \frac{1}{2}$
2. Se itera desde $t = 1 \; hasta \; T$
   a. Se calcula el peso w_i y z_i que es el trabajo
   $$w_i = p(x_i)\left(1 - p(x_i)\right)$$
   $$z_i = \frac{y_i^* - p(x_i)}{p(x_i)\left(1 - p(x_i)\right)}$$
   b. Se ajusta la función $f_m(x)$ mediante la regresión de mínimos cuadrados pesada de z_i usando en peso.
   c. Se actualiza la probabilidad y $F(x)$ de la forma:
   $$F(x) \leftarrow F(x) + \frac{1}{2} f_m(x) \quad y \;\; p(x) \leftarrow \left(e^{F(x)}\right)/e^{F(x)} + e^{-F(x)}$$
3. Obtención del clasificador fuerte
   $$[F(x)] = sign\left[\sum_{m=1}^{M} f_m(x)\right]$$

3.5 Multiboosting

Dado el estudio y las comparaciones mostradas en [23]; los autores han llegado a la decisión que: los métodos Bagging y Adaboosting actúan de diferentes formas y tienen efectos diferentes. De ahí que sugieren que sería posible obtener un gran beneficio mediante la combinación de los dos. Dado que los mecanismos son diferentes, su combinación puede triunfar, teniendo en cuenta que el método Bagging sobre todo reduce la varianza, mientras que Adaboosting reduce tanto el sesgo como la varianza.

Con esta combinación los autores en [23] buscan mejorar los resultados en la clasificación basándose en la reducción de la varianza de Bagging, la cual está presente en Adaboosting pero en menor grado.

Teniendo en cuenta que la solución al problema es una combinación de dos algoritmos en uno y teniendo en cuenta que, una pregunta no trivial sería, ¿Cómo las hipótesis de dos conjuntos de hipótesis diferentes deben ser combinados?; en este caso Adaboosting presenta pesos sobre los votos de los miembros de la comisión mientras que Bagging no. De ahí que escogieron para desarrollar el algoritmo la variante Adaboosting de la familia Boosting y Wagging que es una variante de Bagging descrita en [10], para desarrollar el método Multiboosting.

El método Multiboosting [24] consiste en combinar los algoritmos Wagging y Boosting respectivamente. En este algoritmo se realizan varias inicializaciones de los pesos usando la distribución de Poisson:

$$Poisson() = -log\left(\frac{Random(1{,}999)}{1000}\right) \tag{20}$$

y a partir de cada una de ellas, realiza un proceso de Boosting. A cada proceso de Boosting independiente que parte de una inicialización aleatoria de los pesos se le llama comité. La clasificación del conjunto utiliza el voto ponderado de todos los clasificadores obtenidos.

Multiboosting [24] demostró, que obtiene los clasificadores finales con menor error que Adaboosting o Wagging y ofrece la ventaja adicional sobre Adaboosting que satisface a una de ejecución en paralelo.

La aplicación actual de Multiboosting toma como argumento un único tamaño T de comités, de la que por defecto se establece el número de sub-comités y el tamaño de subcomités $\sqrt{T}$. Ambos valores deben ser números enteros.

El mejoramiento de su implementación se logra, mediante el establecimiento de un índice en los miembros de cada uno de los subcomités finales, lo que cada uno de los miembros de la comisión final emite una probabilidad, a partir de 1.

Pseudocódigo del algoritmo:

Datos de entrada

- S, secuencia de m muestras etiquetadas $\langle (x_1, y_1), \ldots, (x_m, y_m)\rangle$ con etiquetas $y_i \in Y$.
- Algoritmo de aprendizaje base.
- Numero de iteraciones T
- Vector de enteros I_i, especificando las iteraciones en las cuales cada subcomité $i \geq 1$ puede terminar.

1 $S' = S$

2 Inicializa $k = 1$.

3 Se itera desde $t = 1\ hasta\ T$

a. Si $I_k = t$ entonces:
- Se reinicia S' a los pesos aleatorios extraídos de las distribuciones de Poisson continuas.
- Se estandariza S' de tal forma que su suma sea n.
- Se incrementa k.

b. Se ejecuta el clasificador de aprendizaje base pasándole S'

c. Se calcula el error del conjunto de entrenamiento:

$$\epsilon_t = \frac{\sum_{x_j \in S' : C_t(x_j) \neq y_j} pesos(x_j)}{m}$$

d. Si $\epsilon_t > 0.5$ entonces:
- Se reinicia S' a los pesos aleatorios extraídos de las distribuciones de Poisson continuas.
- Se estandariza S' de tal forma que su suma sea n.
- Se incrementa k.
- Se retorna al paso 3.b

o En el caso de que $\epsilon_t = 0.5$ entonces:
- Se determina $\beta_t = 10^{-10}$.
- Se reinicia S' a los pesos aleatorios extraídos de las distribuciones de Poisson continuas.
- Se estandariza S' de tal forma que su suma sea n.
- Se incrementa k.

o En cualquier otro caso entonces:
- Se determina $\beta_t = \frac{\epsilon_t}{(1-\epsilon_t)}$
- Para cada $x_j \in S'$,

e. Se divide los $pesos(x_j)$ entre $2\epsilon_t$ si $C_t(x_j) \neq y_j$ y entre $2(1-\epsilon_t)$ en otro caso.

f. Si $pesos(x_j) < 10^{-8}$, se determina $pesos(x_j) = 10^{-8}$.

Salida

- Clasificador fuerte:

$$C^*(x) = arg\max_{y \in Y} \sum_{t:\, C_t(x)=y} log\frac{1}{\beta_t}$$

Además de las propiedades de reducción del sesgo y la reducción de varianza, Multiboosting presenta la ventaja potencial de cálculo sobre Adaboosting que los clasificadores débiles pueden entrenarse en paralelo, aunque esto requeriría un cambio en el manejo de terminación anticipada del aprendizaje de un clasificador débil. El proceso de Adaboosting es intrínsecamente minimizado por el potencial de la computación paralela. Sin embargo, cada clasificador débil con Wagging es independiente del resto, lo que permite la computación en paralelo.

Con este algoritmo se obtuvieron resultados experimentales sobre el aprendizaje de los comités (árboles de decisión) dentro de los cuales están:

- Multiboosting consigue una mayor reducción del error que cualquier método de Adaboosting, Wagging.
- Multiboosting logra la mayor reducción del sesgo que Adaboosting junto con mayor reducción de la varianza que Wagging.

3.6 Brownboost

Esta variante del Boosting es una versión adaptada del algoritmo *"Boost by Majority"* (BBM) [15] que es una de las primeras versiones del Boosting original desarrollado por Freund. Que combina los objetivos delimitados por el algoritmo BBM y la adaptabilidad de Adaboost.

El método usado para adaptar el BBM considera el límite en cada una de las iteraciones de Boosting. Este límite se puede modelar usando las ecuaciones diferenciales que gobiernan el movimiento browniano. El Brownboost [25], se basa en la búsqueda de soluciones a estas ecuaciones diferenciales.

El algoritmo mantiene para cada muestra (x, y) en el conjunto de entrenamiento sus márgenes $r(x, y)$, no obstante la estructura del algoritmo es similar a la del Adaboosting. Este método recibe inicialmente como parámetros de entrada números positivos reales c, el cual determina la tasa de error, que mientras más grande sea, menor será el mismo y se utiliza para inicializar la variable s, esta puede ser vista como el decrecimiento de un reloj y cuando $s = 0$ el algoritmo se detiene.

En cada iteración el peso $W_i(x, y)$ es asociado con cada muestra y posteriormente se llama al clasificador débil para generar la hipótesis $h_i(\cdot)$ cuya correlación es y_i. Dado la hipótesis h_i el algoritmo escoge el peso α_i y además el número positivo t_i (el cual representa la cantidad de tiempo que puede restar de la cuenta atrás del reloj, para más detalles ver sección 4 de [25]). Para calcular t_i y α_i el algoritmo resuelve la ecuación diferencial:

$$\frac{dt}{d\alpha} = \gamma = \frac{\sum_{(x,y)\in T} exp\left(-\frac{1}{c}(r_i(x,y)+ \alpha h_i(x)y+s_i-t)^2\right)h_i(x)y}{\sum_{(x,y)\in T} exp\left(-\frac{1}{c}(r_i(x,y)+ \alpha h_i(x)y+s_i-t)^2\right)} \tag{21}$$

Ya obtenidos los valores del peso y el valor del parámetro t_i se actualizan los pesos y el tiempo del reloj.

Inicialmente esta versión toma como datos de entrada un conjunto de datos de entrenamiento de tamaño m, cantidad de muestras; una constante $\upsilon > 0$, la cual es utilizada para evitar los casos degenerados al resolver la ecuacion diferencial; el parámetro c, explicado anteriormente; y el clasificador débil.

Si se define la pérdida de la muestra de entrenamiento como $1 - h(x)y$, posteriormente la pérdida esperada de la hipótesis fuerte o final obtenida de la combinación de todas las muestras se plantea de la siguiente forma:

$$1 - \frac{1}{m}\sum_{j=1}^{m} p(x_j)y_j \tag{22}$$

La pérdida es un valor en el rango [0, 2], si se desea medir el error en un rango de [0,1] se hace necesario dividir por 2.

El parámetro c es utilizado para el cálculo del error de pérdida de entrenamiento de la hipótesis final:

$$e = 1 - erf\left(\sqrt{c}\right) \tag{23}$$

La salida de este algoritmo toma cualquier forma de medida de confianza en el rango [-1, 1] o la etiqueta $\{-1, 1\}$:

Para el primer caso la forma exacta de la hipótesis final es seleccionada tal que la predicción de los datos de entrenamiento tienen una pérdida esperada de exactamente ϵ la cual es definida en términos de c. Esto explica por qué la función de error aparece en la definición de $p(x)$(la función de error es, en efecto, simplemente un mapeo no linear dentro del rango $[-1, 1]$[-1, 1].

En el segundo caso retorna una predicción binaria directa tomando el signo de la suma ponderada de las hipótesis débiles.

Pseudocódigo de la versión Binaria del método Brownboost:

Datos de entrada:

- Conjunto de entrenamiento→es un conjunto de muestras etiquetadas de la forma: $T = (x_1, y_1), \dots, (x_m, y_m)$ donde $x_i \in \mathbb{R}^d$ y $y_i \in \{-1, +1\}$.
- Algoritmo de clasificacion utilizado para obtener los clasificadores debiles.
- c→ parametro de valor real positivo.
- $\upsilon > 0$ → pequeña constante para evitar la degeneración de los casos.

1. Estructura de los datos:
   - Valor de prediccion→ cada muestra se le asocia un margen de valor real. El margen de la muestra (x, y) sobre la iteracion i es denotado $r_i(x, y)$. Los valores de prediccion inicial de todas las muestras es 0.
2. Inicializacion
   - Tiempo restante $s_1 = c$.
3. Hacer para $i = 1,2, \dots T$
   a. Asociar con cada muestra un pesos positivo: $W_i(x, y) = e^{-(r_i(x,y)+s_i)^2/c}$
   b. Llamada al clasificador debil con la distribución definida por la normalización $W_i(x, y)$ y recibe la hipotesis $h_i(x)$ que tiene alguna ventaja sobre adivinar al azar $\sum_{(x,y)} W_i(x, y)h_i(x)y = \gamma_i > 0$.
   c. Siendo $\gamma, \alpha \; and \; t$ variables escalares de valores reales que obedecen a la siguiente ecuacion diferencial:

$$\frac{dt}{d\alpha} = \gamma = \frac{\sum_{(x,y)\in T} exp\left(-\frac{1}{c}(r_i(x,y)+ \alpha h_i(x)y+s_i-t)^2\right)h_i(x)y}{\sum_{(x,y)\in T} exp\left(-\frac{1}{c}(r_i(x,y)+ \alpha h_i(x)y+s_i-t)^2\right)}$$

Donde $r_i(x, y)$, $h_i(x)y$ y s_i son constantes en este contexto.

Determinando las condiciones limites $t = 0$, $\alpha = 0$ encuentra la solución del conjunto de ecuaciones para encontrar $t_i = t^* > 0$ y $\alpha_i = \alpha^*$ de tal manera que $\gamma^* \leq \upsilon$ o $t^* = s_i$.

   d. Se actualiza el valor de predicción de cada muestra de la forma:

$$r_{i+1}(x, y) = r_i(x, y) + \alpha_i h_i(x)y$$

   e. Se actualiza el tiempo restante $s_{i+1} = s_i - t_i$ hasta que $s_{i+1} \leq 0$.

Salida

- Hipótesis final

Si $p(x) \; \epsilon \; [-1, +1] \; entonces \; p(x) = erf\left(\frac{\sum_{i=1}^{N} \alpha_i h_i(x)}{\sqrt{c}}\right)$

Si $p(x) \; \epsilon \; \{-1, +1\} \; entonces \; p(x) = sign\left(\sum_{i=1}^{N} \alpha_i h_i(x)\right)$

3.6.1 Extensión de Brownboost para clasificación con múltiples clases

La extensión de Brownboost para la clasificación de múltiples clases [26] está basada en la matriz de codificación M con las entradas en {-1, 0, 1}, donde las filas de M corresponden a los conjuntos de clases originales, mientras que las columnas representan a las particiones binarias de este conjunto que son distinguidos por el clasificador débil.

Este algoritmo realiza los mismos pasos que la versión para la clasificación binaria solo modificando numéricamente los pasos 2 y 3 como se muestran en el Seudo código, que no son más que la forma de salida de la hipótesis débil y la forma de calcular la ecuación diferencial. (Para mayor detalles remitirse al apéndice B de [26]).

Pseudocódigo de la versión para múltiples clase del método Brownboost:

Datos de entrada:

- Matriz de rasgos
- Muestras de entrenamiento: es un conjunto de muestras etiquetadas de la forma: $T = (x_n, y_n), n = 1, \dots, m$ donde $x_n \in \mathbb{R}^d$ y $y_n \in \{y_1, y_2, \dots, y_k\}$. Cada y_n es asociada a traves de la M con un conjunto de ℓ etiquetas binarias $\{\lambda_1^n, \dots, \lambda_\ell^n\}$, donde $\lambda_j^n \epsilon \{-1, +1\}. j = 1, \dots, \ell$.
- Weaklearn: Clasificador débil.
- $c \rightarrow$ parámetro de valor real positivo.
- $v > 0 \rightarrow$ pequeña constante para evitar la degeneración de los casos.

1. Estructura de los datos:
   - Valor de predicción$\rightarrow$ cada muestra se le asocia un margen de valor real. El margen de la muestra (x_n, y_n) sobre la iteración i es denotado $r_{i,j}(x_n, y_n)$. Los valores iniciales de todas las muestras es 0: $r_{i,j}(x_n, y_n) = 0, n = 1, \dots, m, j = 1, \dots, \ell$.
2. Inicialización:
   - Tiempo restante $s_1 = c$.
3. Hacer para $i = 1,2, \dots$
   a. Asociar con cada muestra un pesos positivo: $W_{i,j}(x_n, y_n) = e^{-\left(r_{i,j}(x_n, y_n) + s_i\right)^2 / c}, n = 1, \dots, \ell$.
   b. El aprendiz base binario: Realiza la llamada al Weaklearn ℓ veces, cada tiempo de transición de la distribución ponderada defina por la normalización $W_{i,j}(x_n, y_n)$ para cada j fija, y el conjunto de datos de entrenamiento al lado de las etiquetas binarias definidas por columnas j de la matriz M.

   El aprendiz bases para multiples clases: Realiza la llamada al algoritmo Weaklearn, pasándole los datos de entrenamiento y el conjunto completo de pesos.

   Se obtiene del Weaklearn un conjunto de ℓ hipotesis $h_{i,j}(x)$ las cuales presentan ventaja sobre la eleccion al azar.

$$\frac{\sum_{n=1}^{m}\sum_{j=1}^{\ell} W_{i,j}(x_n, y_n)\left(h_{i,j}(x_n)\lambda_j^n\right)}{\sum_{n=1}^{m}\sum_{j=1}^{\ell} W_{i,j}(x_n, y_n)} = \gamma_i > 0$$

   c. Se determinan las variables $\gamma, \alpha \; and \; t$ que obedecen la siguiente ecuacion:

$$\frac{dt}{d\alpha} = \gamma = \frac{\sum_{n=1}^{m}\sum_{j=1}^{\ell} exp\left(-\frac{1}{c}\left(r_{i,j}(x_n, y_n) + \alpha h_{i,j}(x_n)\lambda_j^n + s_i - t\right)^2\right) h_{i,j}(x_n)\lambda_j^n}{\sum_{n=1}^{m}\sum_{j=1}^{\ell} exp\left(-\frac{1}{c}\left(r_{i,j}(x_n, y_n) + \alpha h_{i,j}(x_n)\lambda_j^n + s_i - t\right)^2\right)}$$

   Donde $r_{i,j}(x_n, y_n)$, $h_{i,j}(x_n)$ y s_i son constantes en este contexto.

   Dadas las condiciones limites $t = 0$, $\alpha = 0$ encuentra la solucion del conjunto de ecuaciones para encontrar $t_i = t^* > 0$ y $\alpha_i = \alpha^*$ de tal manera que $\gamma^* \leq v$ o $t^* = s_i$.

   d. Se actualiza el margen:

$$r_{i,j}(x_n, y_n) = r_{i,j}(x_n, y_n) + \alpha_i h_{i.j}(x_n)\lambda_j^n$$

   e. Se actualiza el tiempo restante:

$$s_{i+1} = s_i - t_i$$

   Hasta que $s_{i+1} \leq 0$.

Salida:

- Hipotesis final:

$$p_j(x) = erf\left(\frac{\sum_{i=1}^{N}\alpha_i h_{i.j}(x)}{\sqrt{c}}\right), j = 1, \dots, \ell.$$

Cuando el clasificador débil obtiene como resultado una predicción de múltiples clases, los autores asumen que la matriz de codificación es de $k \times k$ dimensiones donde la diagonal de entrada es 1 y los otros valores de entradas son -1. También como en la versión para clasificación binaria se toman como parámetros de entrada a c y v.

3.7 Floatboost

El algoritmo Adaboosting necesita procedimientos más efectivos para el aprendizaje de los clasificadores débiles, ya que presenta dificultades ante la presencia de datos de gran dimensión. Esto está basado en que para el aprendizaje se requiere de la estimación de la densidad del espacio de los datos de entrada, pero cuando la dimensión de los mismos es alta, se convierte en un problema para la efectividad del algoritmo.

Otras dificultades del algoritmo Adaboosting son:

- El aprendizaje del clasificador fuerte en Adaboosting no es el mejor para el criterio de que siempre minimiza la tasa de error aunque minimice la función exponencial del margen (para mayor detalle ver [27]).

Para la solución de la segunda dificultad desarrollaron un nuevo procedimiento para un mejor aprendizaje de los clasificadores Boosting, al cual denominaron Floatboost [28].

Este algoritmo usa el mecanismo "marcha atrás" después de cada iteración del Adaboosting para eliminar los clasificadores débiles que presentan un rango de error grande. El objetivo del clasificador Floatboost consiste en que mientras menos clasificadores débiles con tasa de error alta, los resultados serán mejores que en Adaboosting tanto en el entrenamiento como en el Test. Floatboost le introduce al algoritmo Adaboosting la idea de la búsqueda flotante originalmente propuesta en [29] para la selección de rasgos.

El mecanismo "marcha atrás" elimina los clasificadores débiles que no son efectivos o no favorables en términos de tasa de error, esto está dirigido al clasificador fuerte que mejora el error de clasificación.

Para la solución del segundo problema planteado, los autores en [28] proponen un modelo estadístico para el aprendizaje de los clasificadores débiles y eficientes en la selección de rasgos en espacios de altas dimensiones.

Datos de entrada

- Muestras de entrenamiento $Z = \{(x_1, y_1), \dots, (x_N, y_N)\}$, donde $N = a + b$. Donde las muestras a presentan las $y_i = +1$ y b las muestras $y_i = -1$
- El número máximo M_{max} de clasificadores débiles.
- El rango de error es $\varepsilon(H_M)$, y el umbral de aceptación ε^*

1. Inicialización
   - $w_i^{(0)} = \frac{1}{2a}$ para las muestras con $y_i = +1$.
   - $w_i^{(0)} = \frac{1}{2b}$ para las muestras con $y_i = -1$.
   - $\varepsilon_m^{min} = max - valor\ (para\ m = 1\ hasta\ M_{max})$
   - $M = 0$. Numero de Clasificadores débiles más característicos.
   - $\mathcal{H}_0 = \{\}$. Conjunto de M Clasificadores débiles.
2. Incorporación adelantada
   a. $M \leftarrow M + 1$
   b. Selección de h_M acorde con la ecuación del cálculo del rango de error.
   c. Actualizar los pesos $w_i^{(M)} \leftarrow exp[-y_i H_M(x_i)]$ y se normaliza con $\sum_i w_i^{(M)} = 1$
   d. $H_M = H_{M-1} \cup \{h_M\}$; si $\varepsilon_m^{min} > \varepsilon(H_M)$, entonces $\varepsilon_m^{min} = \varepsilon(H_M)$.
3. Excepción condicional
   a. $h' = arg\ min_{h\ \epsilon\ H_M}\ \varepsilon(H_M - h)$
   b. Si $\varepsilon(H_M - h') < \varepsilon_{M-1}^{min}$ entonces
      - $H_{M-1} = H_M - h'$;
      - $\varepsilon_{M-1}^{min} = \varepsilon(H_M - h');\ M = M - 1$;
      - $H_M = \sum_{h\ \epsilon\ \mathcal{H}_M} h$
      - Ir al paso 3.a
   c. Si no
      - Si $M = M_{max}$ o $J(\mathcal{H}_M) < J^*, entonces\ ir\ al\ paso\ 4.$
      - $w_i^{(M)} \leftarrow exp[-y_i H_M(x_i)]\,; ir\ al\ paso\ 2.a;$

Salida

- $H(x) = sign\left[\sum_{h(x)\ \epsilon\ \mathcal{H}_M} h(x)\right]$

Para este algoritmo tomaron como base a la variante del Adaboosting (Real Adaboosting), el cual fue modificado. Inicialmente se le adicionan dos parámetros mas como datos de entrada, como son el número máximo de clasificadores débiles M_{max}, la tasa de error del conjunto de clasificadores débiles seleccionados $\varepsilon(H_M)$ y el umbral de aceptación ε^*. La inicialización de los parámetros no tiene ninguna modificación y se mantiene como mismo se realiza en el algoritmo Real Adaboosting, posteriormente se realiza el envío de los clasificadores débiles seleccionados donde se calcula la hipótesis débil según la ecuación propuesta por el algoritmo Real Adaboosting. En el tercer paso aparece otra modificación, es el paso donde se eliminan los clasificadores débiles que no son factibles basándose en el ε_{M-1}^{min}. Esto se repite hasta que no se pueda eliminar más y el procedimiento termina cuando el conjunto de

entrenamiento este por debajo de J^*, que nos es más que la función exponencial utilizada en la variante Real Adaboosting; o cuando el número máximo M_{max} se alcance.

Finalmente se obtiene el clasificador fuerte de la misma forma que en el algoritmo base utilizado para incluirle la propuesta.

3.8 Aveboost

Teniendo en cuenta que el paso clave del algoritmo Adaboosting es la creación de la distribución sobre un conjunto de muestras para crear cada clasificador débil. Esta distribución es representada por un vector, el cual puede ser un vector ortogonal; creado a partir del clasificador débil anterior. La idea fue utilizar el error del clasificador débil no correlacionado con los clasificadores débiles anteriores. De ahí que en [30] se desarrolló un nuevo algoritmo denominado Aveboost.

Este algoritmo perteneciente a la familia Boosting presenta los mismos objetivos que el algoritmo de corrección total (TCA) [31]. Principalmente el Aveboost calcula la distribución del clasificador débil por el primer cálculo de la distribución igual que en Adaboosting, pero posteriormente se obtiene el porciento de los elementos convenientes del cálculo de los clasificadores débiles anteriores, es decir que toma todos los clasificadores débiles anteriores y construye el próximo clasificador débil.

Datos de entrada

- Muestras de la forma: $\{(x_1, y_1), \dots, (x_m, y_m)\}$
- L_p→ Algoritmo para la obtención de los clasificadores débiles.
- T → Máximo de Iteraciones

1. Inicialización
   - $d_{1,i} = 1/m$ para todas las $i \in \{1, 2, \dots, m\}$. → distribuciones
2. Iteraciones desde $t = 1,2, \dots, T$:
   a. $h_t = L_b(\{(x_1, y_1), \dots, (x_m, y_m)\}, d_t)$.
   b. Calcular el error de $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} d_{t,i}$.
   c. Si $\epsilon_t \geq \frac{1}{2}$ entonces se determina $T = t - 1$ y se aborta el lazo.
   d. Se calcula la distribución ortogonal.
   - Desde $i = 1,2, \dots, m$
     - $c_{t,i} = d_{t,i} \times \begin{cases} \frac{1}{2(1-\epsilon_t)} & si\ h_t(x_i) = y_i \\ \frac{1}{2\epsilon_t} & otro\ caso \end{cases}$
     - $d_{t+1,i} = \frac{t d_{t,i} + c_{t,i}}{t+1}$

Salida

- Hipótesis final

$$h_{fin(x)} = arg \max_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1 - \epsilon_t}{\epsilon_t}.$$

En este algoritmo como se observa en el pseudocódigo se inicializan los parámetros de igual forma que el algoritmo Adaboosting, posteriormente se llama al algoritmo de aprendizaje de modelos base L_b con el conjunto de entrenamiento y la distribución d_1 y se calcula el error de modelo base resultante h_1. En el siguiente paso se calcula el parámetro c_1, que es la distribución que Adaboosting usa para construir el próximo modelo. Este parámetro no necesita ser normalizado porque se calcula de la misma forma que la distribución en Adaboosting.

Cuando las muestras de entrenamiento están afectadas por el ruido el algoritmo Adaboosting tiende a aumentar los pesos y sobre ajustarlos, lo que no es así en Aveboost ya que tiende a aplacar el sobre ajuste mediante el merito del proceso del porciento.

3.9 Modest Adaboost

En [32] los autores proponen un nuevo algoritmo para la familia Boosting, con el objetivo de disminuir el error de generalización en comparación con las variantes desarrolladas anteriormente, presentando un costo algo más elevado en el error de entrenamiento.

Como en el Adaboosting original esta variante recibe un conjunto de datos de entrenamiento $(x_1, y_1), \dots, (x_n, y_n)$ donde x_i es el vector de entrada y y_i la etiqueta correspondiente al ejemplo $y = \{-1, 1\}$. También presenta a (H) que es una familia de funciones de clasificación débil, que puede llevar a cabo la asignación de espacio para las etiquetas de clase, $h: X \to \{-1, +1\}$ o puede ser una función de valores reales, para el cálculo de las predicciones de confianza. El único requerimiento para $h(x) \epsilon\, H$ es que para cualquier valor de salida se deberá estimar la probabilidad de la siguiente forma:

$$P_D\big(y = 1 \cap h(x)\big) \tag{24}$$

Donde D es la distribución de las muestras de entrenamiento.

Esta variante obtiene el clasificador fuerte mediante la función:

$$F(x) = sign\left[\sum_{m=1}^{M} f_m\big(h_m(x)\big)\right] \tag{25}$$

Donde $h_m(x)\ \epsilon\ H$ son los clasificadores débiles entrenados y f_m son las funciones de valores reales.

Pseudocódigo del método Modest Adaboost:

Datos de entrada

- Muestras de entrenamiento → $(x_1, y_1), \dots, (x_N, y_N)$
- Máximo de Iteraciones → M

1. Inicialización
   - Los pesos: $D_0(i) = \frac{1}{N}$
2. Iterar desde $m = 1$ hasta M y mientras $f_m \neq 0$
   a. Entrenar el clasificador débil $h_m(x)$ usando la distribución $D_m(i)$.
   b. Calcular la distribución invertida: $\bar{D}_m(i) = \big(1 - D_m(i)\big)\bar{Z}_m$.
   c. Calcular
      - $P_m^{+1}(x) = P_{D_m}\big(y = +1 \cap h_m(x)\big)$
      - $\bar{P}_m^{+1}(x) = P_{\bar{D}_m}\big(y = +1 \cap h_m(x)\big)$
      - $P_m^{-1}(x) = P_{D_m}\big(y = -1 \cap h_m(x)\big)$
      - $\bar{P}_m^{-1}(\mathrm{x}) = \mathrm{P}_{\bar{\mathrm{D}}_\mathrm{m}}\big(\mathrm{y} = -1 \cap \mathrm{h_m(x)}\big)$
   d. Determinar:

$$\mathrm{f_m(x)} = \big(\mathrm{P_m^{+1}}(1 - \bar{\mathrm{P}}_\mathrm{m}^{+1}) - \mathrm{P_m^{-1}}(1 - \bar{\mathrm{P}}_\mathrm{m}^{-1})\big)(\mathrm{x})$$

y se actualizan las distribuciones:

$$\mathrm{D_{m+1}(i)} = \frac{\mathrm{D_m(i)}\exp\big(-\mathrm{y_i f_m(x_i)}\big)}{\mathrm{Z_m}}$$

3. Construye el clasificador final:

$$\mathrm{sign}\left[\sum_{\mathrm{i=1}}^{\mathrm{i=M}} \mathrm{f_m(x)}\right]$$

Los parámetros $\mathrm{Z_m}$ y $\bar{\mathrm{Z}}_\mathrm{m}$ son los valores de normalización utilizados en la actualización de las distribuciones y en el cálculo de las distribuciones invertidas respectivamente.

Esta variante de Adaboosting supera en eficiencia variantes del método Adaboosting enfocadas al procesamiento de imágenes en términos de error de generalización y el sobre ajuste.

3.10 SAMME Algoritmo para clasificación con múltiples clases

Hasta el momento se han realizado una serie de modificaciones y extensión del algoritmo Adaboosting para cuando esté presente una clasificación de múltiples clases, como por ejemplo las más tempranas extensiones son: Adaboosting.M1 y M2 [18, 19], Logitboost [22], la extensión desarrollada con Brownboost [25, 26], entre otras. Este es un tema que ha seguido mejorando hasta la actualidad utilizando diferentes variantes.

Ji Zhu, Hui Zou, Saharon Rosset y Trevor Hastie en [33] desarrollaron un algoritmo para el trabajo con múltiples clases, mediante un modelado aditivo por pasos usando funciones de pérdidas exponenciales para la clasificación con múltiples clases, al cual denominaron SAMME.

Pseudocódigo del algoritmo SAMME:

Datos de entrada

1. Se inicializan los pesos:
   - $w_i = \frac{1}{n}$
   - $i = 1, 2, \dots, n$
2. Iterar desde $m = 1$ hasta $M \rightarrow$ Máximo numero de iteraciones
   a. Se entrenar el clasificador débil $T^{(m)}(x)$ con los datos de entrenamiento usando los pesos w_i.
   b. Calcular el error de cada clasificador débil entrenado: $err^{(m)} = \sum_{i=1}^{n} w_i II\left(c_i \neq T^{(m)}(x_i)\right) / \sum_{i=1}^{n} w_i$
   c. Calcular: $\alpha^{(m)} = \log\frac{1-err^{(m)}}{err^{(m)}} + \log(k-1)$
   d. Actualización de peso: $w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(x_i)\right)\right)$;

   Iteración desde $i = 1$ hasta n
   - Re-normalizar w_i.

Salida

$$C(x) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}\left(T^{(m)}(x) = k\right).$$

Este algoritmo parte de la misma estructura modular simple que Adaboosting pero con sencillas diferencias en el paso 1 del algoritmo, paso donde se inicializan los pesos w_i donde $i = 1, 2, \dots, n$ y especialmente en el término adicional $\log(K-1)$. En el caso de que $K = 2$, donde K es el número de clases; el algoritmo SAMME se reduzca simplemente al método Adaboosting, pero para el caso $K > 2$, se le adiciona el término $\log(K-1)$ a la ecuación del cálculo de parámetro $\alpha^{(m)}$ utilizado en la actualización de los pesos. Un requerimiento inmediato es que para que α^m sea positivo, solo se necesita $\left(1 - err^{(m)}\right) > 1/K$, o la exactitud de cada clasificador débil tiene que ser mejor que adivinar a azar ½, > 50%.

En este algoritmo el error de la prueba decrece rápidamente hasta tomar un valor pequeño y posteriormente continúa decreciendo después de las 600 iteraciones.

4 Descripción de los clasificadores débiles utilizados en el reconocimiento del habla como del locutor

4.1 Redes neuronales

Las redes neuronales (Artificial Neural Networks, ANN) son modelos computacionales que emulan el comportamiento del cerebro por medio de topologías que reflejan la interconexión de las células nerviosas [21]. Una ANN consiste en un grupo de "neuronas" (nodos o celdas) que se distribuyen en capas y se interconectan por medio de caminos pesados. Cada neurona es un elemento procesador que entrega una salida simple a partir de múltiples entradas, dicha salida usualmente es controlada por una función de activación.

Estructura de las ANN:

Poseen una capa de entrada, una capa de salida y una o más capas ocultas, estando conectadas las salidas de una capa a las entradas de la próxima. Una ANN "se entrena" ajustando los pesos de los caminos de las uniones entre neuronas. El entrenamiento es supervisado si se conocen los patrones a aplicar a la capa de entrada y las clases correspondientes a obtener en la capa de salida, las neuronas correspondientes a las capas ocultas se entrenan de forma tal que cuando se cargue el patrón a la entrada, la salida de la clase a la cual corresponde el mismo.

Estas redes también se conocen como máquinas de aprendizaje o “perceptrones”. El entrenamiento es no supervisado si no se conocen las clases de salida.

Las ANN más usadas para clasificación en reconocimiento del locutor, y que han obtenido relativamente buenos resultados en tareas de moderada complejidad son:

4.1.1 Multi-Layer Perceptron (MLP)

Se trata de una red de varias capas, cuya función de activación de las neuronas es del tipo “sigmoide”, cada salida corresponde a una clase. El entrenamiento supervisado se lleva a cabo de acuerdo a una función de costo, la forma más usual de entrenamiento es por medio del algoritmo de retro-propagación (“back propagation”). El MLP es robusto al ruido y permite tener en cuenta el contexto de la señal.

4.1.2 Radial Basis Functions (RBF)

Constituye una alternativa de MLP, que reduce el tiempo de entrenamiento con las mismas propiedades discriminativas de MLP, es una red de dos capas con funciones Gaussianas en las neuronas de la primera capa y funciones lineales en la capa de salida, teniendo un comportamiento similar al modelo de mezclas Gaussianas (GMM).

4.1.3 Learning Vector Quantization (LVQ)

La arquitectura LVQ utiliza varias celdas de salida para cada clase, es similar a la VQ y su entrenamiento supervisado se basa en el principio de clasificación del vecino más cercano. Sin embargo, los vectores de referencia no se escogen de los patrones de entrada sino que se entrenan de acuerdo al objetivo de la clasificación, permitiendo identificar las fronteras de las clases con más precisión que la VQ con muchas menos referencias.

Otras alternativas y variantes de las ANN utilizadas en el reconocimiento de locutores son:

- Task Decomposition Neural Network (TDNN)
- Task Decomposition Neural Network (TDNN)
- Recurrent Neural Networks (RNN)
- Neural Tree Networks (NTN)
- Probabilistic Neural networks (PNN)

4.2 Árboles de decisión

De todos los métodos de aprendizaje, los sistemas de aprendizaje basados en árboles de decisión son quizás el método más fácil de utilizar y de entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Los árboles de decisión se utilizan desde hace décadas, y son especialmente apropiados para expresar procedimientos médicos, legales, comerciales, estratégicos, matemáticos, lógicos, etc.

Una de las grandes ventajas de los árboles de decisión es que, en su forma más general, las opciones posibles a partir de una determinada condición son excluyentes. Esto permite analizar una situación y, siguiendo el árbol de decisión apropiadamente, llegar a una sola acción o decisión a tomar. Existen una serie de algoritmos para la creación de los árboles de decisión como son CART, C4.5, entre otros. Donde el más popular y más usado en el reconocimiento del locutor y del habla es el C4.5.

4.2.1 Clasificador C4.5

En la década del 90, se proponen nuevas estrategias para la construcción de árboles. Entre las estrategias propuestas se destaca por su popularidad y simpleza el C4.5 [34]. Este método es una extensión del algoritmo ID3 (Induction Decision Trees) [22], que permite el trabajo con varias clases, modificando convenientemente las ecuaciones a continuación pertenecientes al algoritmo ID3.

$$I(p,n) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right) \tag{26}$$

donde p y n representan la cantidad de objetos pertenecientes a las clases P y N respectivamente.

Al utilizar el rasgo $A = \{A_1, A_2, \dots, A_m\}$ en un nodo, se particiona en m subconjuntos MI_i, con $i = (1, \dots, m)$. La información requerida para el subárbol de MI_i es $I(p_i, n_i)$. La información esperada al utilizar el rasgo A es la siguiente:

$$E(A) = \sum_{i=1}^{m} \frac{p_i + n_i}{p+n} I(p_i, n_i) \tag{27}$$

La información que se gana al dividir por A es:

$$gain(A) = I(p,n) - E(A) \tag{28}$$

Así, para cada caso se escoge el rasgo que tiene una mayor ganancia de información para construir el árbol, y maximizar la ganancia es equivalente a minimizar la información esperada.

Las modificaciones realizadas a las 3 ecuaciones anteriores se observan de la siguiente forma:

$$I(T) = -\sum_{j=1}^{l} \frac{freq(K_j, T)}{|T|} \log_2\left[\frac{freq(K_j, T)}{|T|}\right] \tag{29}$$

Al utilizar el atributo $A = \{A_1, A_2, \dots, A_m\}$ en un nodo, se particiona T en m subconjuntos T_i con $i = 1, \dots, m$. La información esperada al utilizar el rasgo A es la siguiente:

$$E(A) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} I(T_i) \tag{30}$$

La información que se gana al dividir por A es como sigue:

$$gain(A) = I(T) - E(A) \tag{31}$$

Sin embargo, debido a que el criterio de la ganancia de información tiende a favorecer a los rasgos con mayor cantidad de valores, el C4.5 utiliza una nueva medida, la proporción de ganancia, que no es más que el cociente entre la ganancia y la información que se obtiene al dividir por el rasgo que se analiza.

$$gain\ ratio(A) = \frac{gain(A)}{E(A)} \tag{32}$$

La información esperada de dividir utilizando un rasgo A se calcula como:

$$E(A) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \log_2\left[\frac{|T_i|}{|T|}\right] \tag{33}$$

El C4.5, a diferencia del ID3, sí concibe el trabajo con rasgos numéricos. Para ello, cuando se analiza un rasgo numérico, éste es discretizado de la siguiente forma: sea un rasgo A numérico, se

toman todos los valores de éste que aparecen en los objetos del subconjunto que se analiza y se ordenan de forma ascendente. Luego, se calcula la proporción de ganancia que se obtendría al discretizar el rasgo A utilizando cada uno de estos valores como punto de corte.

El algoritmo C4.5 toma en cuenta las ausencias de información para decidir qué rasgo deberá formar el nodo a construir, esa ausencia de información reducirá la medida de proporción de ganancia de acuerdo al número de casos que la tengan. Para el tratamiento de los valores incompletos, el C4.5 modifica la fórmula de la ganancia. Sea F la cantidad de objetos donde el valor del rasgo A es conocido, la fórmula queda de la siguiente forma:

$$gain(A) = \frac{F}{|T|} I(T) - E(A) \tag{34}$$

Al dividir el conjunto de entrenamiento T, en el caso de tener ausencias de información, se asigna una probabilidad de pertenencia al subconjunto. Para ello, a cada objeto que pertenece a un subconjunto T_i, si su valor es conocido, se le asigna un peso $w(o)$ igual a 1, de lo contrario, se le asigna la probabilidad de obtener A_i en ese subconjunto $\left(P(A_i)\right)$. Como el proceso de división es iterativo, de manera general, a un objeto con peso $w(o)$ con un valor desconocido, será asignado a cada subconjunto T_i con peso $w(o) = w(o) \cdot P(A_i)$.

$$P(A_i) = \frac{\sum_{0 \in T, A(0) = A_i} w(o)}{\sum_{0 \in T, A(0) \neq ?_i} w(o)} \tag{35}$$

Es decir, el peso del objeto estará determinado por la razón entre la suma de los pesos de los objetos que tienen valor A_i, y la suma de los pesos de los objetos que tienen un valor conocido para el atributo A.

Para clasificar un objeto con descripción incompleta, si se llega a un nodo donde el valor del rasgo que se analiza es desconocido en el objeto en cuestión, se explorarán todas las ramas del nodo y se combinarán los posibles resultados de la clasificación de forma aritmética, es decir, se calcula una tupla con las probabilidades frecuenciales de cada una de las posibles clases, y se asigna la clase de mayor probabilidad.

Una de las desventajas del ID3 es el sobre entrenamiento, para evitarlo, el C4.5 utiliza la poda basada en el error. Esta estrategia se basa en la estimación del error que se cometería en un nodo. Para ello, considera que el error está acotado por una distribución binomial para un nivel de confianza dado α. Para calcular el error en una hoja, se estima una distribución binomial $U_\alpha(E, |T|)$ teniendo como argumentos la cantidad de objetos clasificados de forma incorrecta (E) y el total de objetos en la hoja. Así, el error predicho será $|T| \cdot U_\alpha(E, |T|)$. En cada subárbol, se calcula su error como la suma del error asociado a sus ramas. En caso de que un subárbol tenga un mayor error que si fuera reemplazado por una hoja, es podado. El procedimiento termina cuando no se pueden realizar más podas.

4.3 Clasificador HMM

Los modelos ocultos de Markov (HMM) [2, 4, 6] son modelos estadísticos que pueden representar procesos aleatorios paramétricos. Estos son el enfoque estocástico más popular y con mayor éxito en el ámbito del reconocimiento del habla.

Estos modelos están compuestos por dos elementos básicos: un proceso Markov y un conjunto de distribuciones de probabilidad de salida. Los estados del proceso de Markov están ocultos pero son observables de una manera indirecta a partir de la secuencia de vectores con información espectral extraídos de la señal de voz de entrada.

Los HMMs en reconocimiento de voz se utilizan teniendo en cuenta dos hipótesis:

La voz se puede dividir en segmentos, estados, en los que la señal de voz se puede considerar estacionaria, es decir que la señal mantiene una estructura de principio a fin en la ventana de análisis.

La probabilidad de observación de que un vector de características se genere depende solo del estado actual y no de símbolos anteriores. Esto es una definición de Markov de primer orden denominado hipótesis de independencia.

Mientras que las aplicaciones de reconocimiento del habla presentan una topología clásica de modelado conocida como "de izquierda a derecha" (ver Fig. 3), los modelos utilizados por los sistemas HMM para caracterizar la identidad de un locutor independiente del texto son los denominados ergódicos[5], los cuales no presentan un orden consecutivo de las transiciones entre los distintos estados del modelo y, por lo tanto, resulta factible cualquier combinación de transición entre estados.

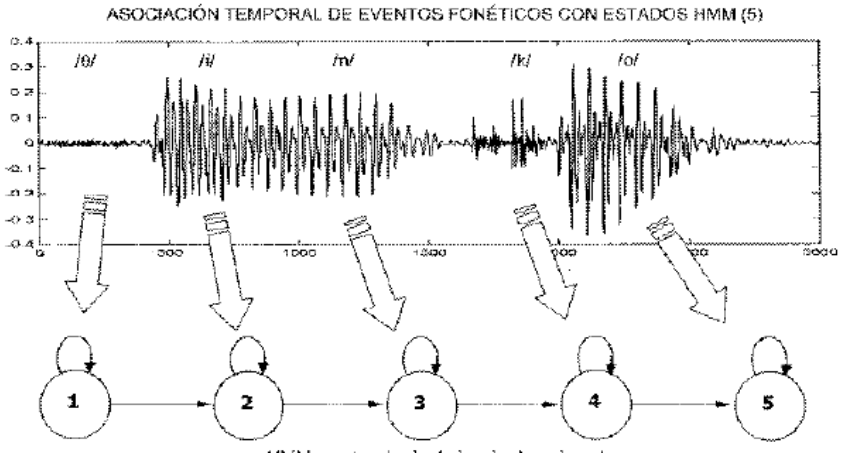

Fig. 3. Representación de los modelos ocultos de Markov

La principal ventaja de los sistemas de reconocimiento basados en HMM respecto a otros tipos, la constituye su gran versatilidad, tanto en lo que se refiere a los procesos de entrenamiento como a ciertas características variables de la muestra: duración, contenido fonético o lingüístico, contexto, etc. A todo ello, hemos de añadir su gran adaptabilidad a la variación de las condiciones de registro o del canal de trasmisión.

4.4 Modelos de mezclas Gaussianas (GMM)

Las GMM [4, 5, 6] modelan los distintos vectores de rasgos de una locución, realizando una suma ponderada o mezcla de funciones de densidad de probabilidad Gaussiana.

Pueden entenderse como un sistema en el que se aglutinan las virtudes de aquellos otros basados en técnicas *VQ* y los denominados clasificadores Gaussianos uni-modales. También pudieran apreciarse como *HMM* de un solo estado.

El uso de las *GMM* representa con un alto grado de fidelidad un amplio margen de distribuciones muestrales, tales como los diferentes coeficientes cepstrales que puede generar una locución concreta.

A diferencia de los *HMMs*, las *GMMs* no precisarán en la fase de entrenamiento segmentar en estados ni entrenar la matriz de probabilidades de transiciones. Además, en la etapa de reconocimiento, no será necesario buscar la secuencia de estados de máxima verosimilitud, sino que bastará con acumular las probabilidades que asocia el modelo con cada uno de los vectores de entrada.

Descripción del modelo

[5] Cuando la probabilidad de que el modelo que se encuentre en un nodo determinado sea constante para cualquier nodo escogido.

El modelo de densidades de mezclas Gaussianas es una suma pesada de M (número de mezclas) componentes de densidad descrita por:

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{36}$$

En la siguiente figura se muestra gráficamente la suma de las componentes de densidad:

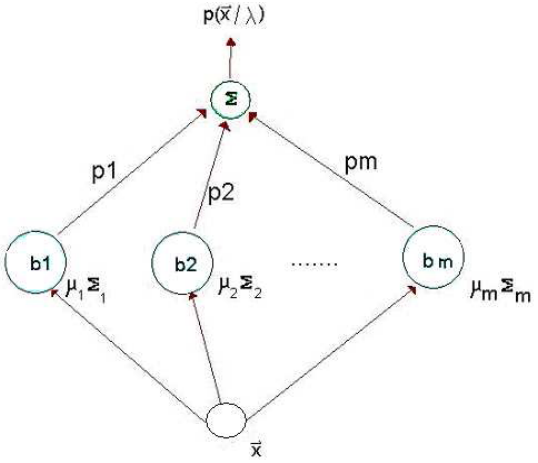

Fig. 4. Descripción de un modelo de mezclas Gaussianas de M componentes

donde:

- $\bar{x}$ es un vector D-dimensional

$$X = \begin{Bmatrix} \frac{\vec{x}_1}{\downarrow} & \frac{\vec{x}_2}{\downarrow} & \cdots & \frac{\vec{x}_T}{\downarrow} \\ c_{1,1} & c_{1,2} & \cdots & c_{1,T} \\ c_{2,1} & \cdots & & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ c_{Z,1} & \cdots & & c_{Z,T} \end{Bmatrix}$$

Matriz de rasgos

La matriz X es una sucesión de variables aleatorias indexadas por una variable discreta, el tiempo ($t = 1, \ldots, T$). Cada una de las variables aleatorias del proceso tiene su propia función de distribución de probabilidad y asumiremos que son independientes. X es una matriz de rasgos Mel (MFCC-Delta) obtenida de una expresión de voz de un locutor.

- $b_i(x)$ son las componentes de densidad, con $i = 1, 2, \ldots, M$. Cada componente es una función Gaussiana de la forma:

$$b_i(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_i)' \sum_i^{-1} (x-\mu_i)\right\} \tag{37}$$

donde μ_i y Σ_i son el vector de medias y el vector de covarianza correspondientes a la i-esima mezcla, los cuales son inicializados desde la matriz de rasgosX.

- p_i son los pesos de las mezclas con $i = 1, 2, \ldots, M$ y satisfacen la condición $\sum_{i=1}^{M} p_i = 1$.

$\lambda = \{\bar{p}, \mu, \Sigma\}$ Representa el modelo de mezclas Gaussianas, donde $\bar{p}$ es el vector de pesos, μ es la matriz de media Σ y es la matriz de varianzas. Cada locutor es representado por un modelo λ obtenido utilizando el algoritmo EM [4, 5].

Componentes de densidad Gaussiana

Un factor crítico en el entrenamiento de las GMM es el número de las mezclas a seleccionar. Decidir el número de mezclas M para el modelo del locutor es un problema importante, ya que el objetivo es elegir el número mínimo de componentes de densidades Gaussianas necesarias para modelar adecuadamente un locutor.

- Elegir pocos componentes de la mezcla puede producir un modelo del locutor que no modele exactamente las características que lo distinguen entre el grupo de locutores.
- Elegir demasiados componentes puede llevar a que el costo computacional sea excesivamente alto a la hora del entrenamiento o la clasificación.

Varios aspectos del uso de las GMM para la verificación del locutor independiente del texto a tener en cuenta son:

- Son insensibles al método de inicialización que se utilice para crear el modelo, logrando la misma verificación del locutor.
- La limitación de la varianza es importante en el entrenamiento para evitar las singularidades del modelo.
- Mantienen un alto desempeño en la verificación con el aumento del tamaño de la población de los datos.
- Se obtienen mejores resultados experimentales que con técnicas usadas con anterioridad como la *DTW, VQ, ANN*y *HMM*.

Estos resultados indican que las GMM proporcionan una representación robusta del locutor para la difícil tarea de la verificación a partir de locuciones independientes del texto. Las GMM se pueden utilizar fácilmente en sistemas de ejecución en tiempo real (online).

5 Aplicaciones de la familia Boosting en el reconocimiento del habla y del locutor

Los algoritmos pertenecientes a la familia Boosting se han venido aplicando en diferentes ramas del reconocimiento de patrones, teniendo en cuenta que estos algoritmos realizan una combinación de clasificadores para la obtención de un clasificador final más fuerte.

Estas aplicaciones pueden verse en las ramas del procesamiento de imágenes [35-38], en la clasificación de patrones temporales [39] y en aplicaciones de minería de datos [40]. También se han utilizado en enfoques del reconocimiento del habla y del locutor, lo cual se abordará a continuación en este capítulo.

5.1 Aplicaciones del Boosting en el reconocimiento del habla

En el reconocimiento automático del habla se han desarrollado algunos trabajos usando las variantes del Boosting desde la década de los 90. Dentro de esos trabajos existen aportes en el perfeccionamiento en el rendimiento, la optimización de los algoritmos, la creación de algoritmos para los sistemas de reconocimiento automático del habla, la creación de algoritmos para la detención de ruido en la señal

del habla, entre otros. A continuación se presenta una descripción y análisis de algunos de estos trabajos.

5.1.1 El Boosting para mejorar el rendimiento del reconocimiento del habla de gran vocabulario

Los autores en [41] tienen en cuenta que los modelos acústicos conexionistas suelen utilizar considerablemente menor cantidad de parámetros que *HMM*, permitiendo el funcionamiento en tiempo real, sin una degradación significativa del rendimiento. Sin embargo, el reducido del número de parámetros en modelos acústicos conexionistas es considerado un problema, dado que en la disminución de parámetros se puede eliminar información necesaria. Este trabajo propone realizar una selección de los datos de entrenamiento para incrementar el funcionamiento del sistema de reconocimiento del habla.

Para esto desarrollan 3 nuevos procedimientos utilizando el algoritmo Boosting con los modelos acústicos de las redes neuronales:

Boost1: Entrena una red sobre un subconjunto seleccionado aleatoriamente de las sentencias de entrenamiento. Usa esta red para filtrar los datos restantes para producir un conjunto de entrenamiento para una segunda red. Calcula la tasa de reconocimiento de la trama de la primera red sobre las nuevas sentencias de los datos de entrenamiento. Selecciona los datos de entrenamiento para una segunda red de tal manera que la mitad son las sentencias en que la tasa de reconocimiento de la trama de la primera red es más bajo, y la otra mitad son las sentencias en que la tasa de reconocimiento de la trama de la primera red es la más alta.

Boost2: Forma una red en un subconjunto seleccionado al azar de las sentencias de entrenamiento. Calcula la tasa de reconocimiento por trama de la primera red de nuevas sentencias de los datos de entrenamiento. Selecciona las sentencias en las que la tasa de reconocimiento de la trama de la primera red es la más baja para formar una segunda red.

Boost3: Entrena una red de un subconjunto seleccionado al azar de las sentencias de entrenamiento. Usa esta red para producir probabilidades a posteriores de los fonemas. Estas probabilidades de los fonemas se utilizan posteriormente como las probabilidades de observación dentro de un marco *HMM*. Decodificar las nuevas sentencias de los datos de entrenamiento, y se calcula a través de la tasa de error por palabra. Se selecciona las sentencias para los que la tasa de error por palabra es más alta para entrenar una segunda red.

Nota: En todos los casos el número de sentencias elegido para el entrenamiento de la segunda red es la misma que se utiliza para formar la primera red.

Este trabajo presentó una mejora de un 15,2 % si se utilizara una sola red neuronal y un 8,7% si los datos de entrenamiento fueran seleccionados aleatoriamente. Además que estos procedimientos se pueden utilizar para los modelos acústicos dependientes e independientes del contexto.

5.1.2 Mezclas gaussianas con Boosting para el sistema de reconocimiento del habla continua de gran vocabulario

El objetivo en [42] es optimizar el funcionamiento del algoritmo Adaboosting ante los sistemas de reconocimiento del habla. Los autores se centraron principalmente en la disminución del costo computacional ya que el principal resultado alcanzado con este trabajo fue la paralelización del algoritmo Adaboosting, el cual utiliza las mezclas Gaussianas para la creación de los clasificadores débiles. También desarrollaron una variante jerárquica del método Adaboosting.

Para la realización de este trabajo se basaron en el algoritmo Adaboosting.M2 (mencionado en epígrafes anteriores). El cual recibe como datos de entrada conjuntos de pares de entrenamientos etiquetados, (x_i, y_i); donde x_i representa los rasgos asociados a la ith muestra, que para el caso de este trabajo es un vector de rasgos acústicos y y_i es la etiqueta de cada muestra.

Para la obtención de los clasificadores débiles utilizan el algoritmo GMM, las cuales son mezclas de Gaussianas, con una mezcla para cada fonema dependiente del contexto.

Los autores demostraron con este trabajo que la utilización del Boosting es una manera eficaz de construir sistemas para grandes bases de datos. Y experimentalmente, encontraron que las variantes

jerárquica y de restricción del algoritmo Adaboosting básica permiten una gran mejora en los sistemas de reconocimiento del habla.

5.1.3 Mejoramiento del rendimiento de los sistemas reconocimiento del habla continua de gran vocabulario mediante combinación de modelos acústicos

En [43] los autores describen el trabajo sobre la aplicación de conjuntos de modelos acústicos para el problema de reconocimiento de un amplio vocabulario del habla continua (LVCSR). Nos proponen tres algoritmos para la construcción de ensembles. Los dos primeros tienen su origen en el algoritmo Bagging, y el tercer algoritmo es basado en las técnicas Boosting.

El Algoritmo 3 presenta una solución al estilo del Boosting que incorporan el parámetro C_t para describir la diferencia de la importancia entre los modelos. El algoritmo se basa en la función de costo siguiente:

$$L = \sum_{i=1}^{|X|} \exp\left(-\sum_{t=1}^{T} c_t e_t(X_i)\right) \tag{38}$$

donde:

$$e_t(X_i) = \begin{cases} \alpha & si\ \varepsilon_i^t = 0 \\ -\varepsilon_i^t & otro\ caso \end{cases} \tag{39}$$

La ε_i^t es el reconocimiento incorrecto mediante el rango de error de la palabra y α es el reconocimiento correcto.

A diferencia de otros métodos Boosting donde la demanda de recursos para el cálculo y almacenamiento es más grande, el método propuesto presenta una solución más eficaz, idónea para la formación del modelo acústico.

Los tres algoritmos logran mejoras significativas en el rendimiento del sistema. La reducción relativa de la tasa de error por palabra en la prueba es de 15,56%, 11,71% y 14,38% respectivamente. Los resultados muestran el potencial de este enfoque como un método significativo para mejorar la calidad de los modelos acústicos entrenados.

Los autores plantean que se hace necesario mejorar la función de costo definida en el tercer algoritmo para incorporar más información.

5.1.4 Boosting y HMMs para una aplicación de reconocimiento del habla

Si bien el algoritmo de Adaboost original se ha definido para tareas de clasificación, los autores presentan como objetivo en [44] examinar la aplicabilidad del algoritmo Adaboosting a las secuencias de aprendizaje, centrado en el reconocimiento del habla.

Los autores exploran el uso de Boosting para HMMs que emplean los modelos de mezclas Gaussianas pero a diferencia de los enfoques anteriores, se aplicará el algoritmo Boosting a nivel de fonemas. Esto se traduce en una serie de modelos de base para cada fonema.

La forma más sencilla de aplicar el método Boosting al entrenamiento del HMM es convertir el problema en un marco de clasificación binaria. Esto es posible a un nivel de clasificación de fonemas, donde cada entidad de la clase corresponde a uno de los fonemas posibles. Mientras que la formación disponible en los datos son anotados en el tiempo de manera que solo los datos que contiene sub-secuencias de fonema puede ser extraído, es natural para adaptar cada Modelo Oculto Markov m_i de una sola clase n_c. Un clasificador Bayesiano puede ser utilizado como un experto o clasificador débil en el marco Adaboost.

Los autores están estudiando la utilización de varios métodos para combinar los modelos para su uso en el reconocimiento de voz. Y plantean que otro tema de investigación en el futuro es la aplicación del

Boosting a nivel de frases. Esto trae consigo dificultades adicionales, ya que tratar el problema como una tarea de clasificación ya no es factible.

5.1.5 Optimización del diseño del clasificador de habla y no habla usando Adaboosting

En este trabajo [45] los autores proponen un nuevo método diseñado para la clasificación de voz o no voz en expresiones acústicas. Este método utiliza una combinación de clasificadores bases simples a través de algoritmo Adaboosting y un conjunto de rasgos del habla optimizados mediante la combinación de las sustracciones espectrales.

Este trabajo utiliza el algoritmo Adaboosting para el entrenamiento de hiperplanos (parámetro que delimita los impostores de los target) en un principio y de manera automática. Para esto utilizan los perceptrones como clasificadores débiles mediante una función sigmoidal:

$$f(x) = \tanh(\gamma x) \tag{40}$$

donde x es el vector de rasgos y $\gamma = 4$ es el controlador del rango de las regiones limites. Usan los clasificadores bases mediante los límites de decisión para hacer más rápido y simple el proceso de aprendizaje.

Para la hipótesis final $h_t(x)$ se eligió para usar los stump[6] donde solo un único rasgo es usado para cada perceptron. Dado que experimentos realizados por los autores demuestran que la diferencia en el funcionamiento entre los stump y los perceptrones en general no son significantes.

El clasificador final está dado por:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t (h_t(x) + \delta)\right) \tag{41}$$

donde α_t es el peso del clasificador base y δ es un parámetro de control del rango de falsas alarmas.

El beneficio de este método se encuentra en su simple implementación y poca complejidad computacional, en comparación con el VAD G.729 [46].

5.1.6 Selección de rasgos discriminantes para los modelos ocultos de Markov usando Boosting segmental

Los autores teniendo en cuenta que persisten los problemas para la selección de rasgos para los modelos de Markov en la clasificación secuencial, en [47] nos proponen el método Segmentally-Boosted-HMMs (SBHMMs) donde la optimización de los rasgos es realizada de manera segmental y discriminante.

En este trabajo desarrollan un nuevo algoritmo para la selección de rasgos y nos muestran el algoritmo SBHMM que mejora consistentemente el reconocimiento tradicional de HMM en varios dominios.

Para este nuevo algoritmo utilizan el Adaboosting y como clasificadores débiles los árboles de decisión (*stump*).

Para la obtención del nuevo espacio de rasgos discriminantes V del SBHMMs usan las hipótesis del Adaboosting, donde $V = \left(H^{(1)}, \dots, H^{(S)}\right)$, y S es el total de clases. Es decir toman el valor de las hipótesis como coordenadas en el espacio.

En conclusión este trabajo presenta un algoritmo Boosting segmental que funciona con una selección de rasgos discriminativos para secuencias de tiempo y este algoritmo construye un nuevo espacio de rasgos discriminativos.

Nos plantean que en un futuro sería interesante integrar la técnica de selección de rasgos segmental con el entrenamiento discriminativo segmental para la tarea de reconocimiento de gesticulaciones.

---

6 Árbol de decisión un solo nodo

5.1.7 Detección de ruido y voz con Adaboosting

Los autores en [48] presentan como objetivo principal la detención de voz y ruido repentino introducido en la señal, teniendo en cuenta que sobre la reducción de ruido existen una serie de trabajos pero presentan dificultades ante la presencia de ruido repentino como se observa en la figura 5.

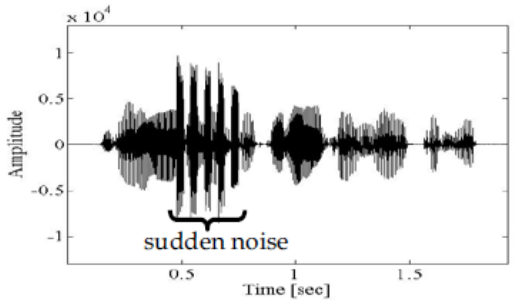

Fig. 5. Señal de audio que presenta ruido repentino

Para esto utilizan el algoritmo Adaboosting en sus dos versiones (Clasificación binaria y Clasificación con múltiples clases) utilizando los árboles de decisión como clasificadores débiles. Para esta propuesta los autores realizan tres modificaciones principales:

El cálculo del error:

$$\epsilon_t = \sum_{i=1}^{n} w_t(z_i) \frac{I(h_t(x_i) \neq y_i) + 1}{2} \tag{42}$$

La actualización de los pesos:

$$w_{t+1} = \frac{w_t(z_i)\exp\{\alpha_t \cdot I(h_t(x_i) \neq y_i)\}}{\sum_{j=1}^{n} w_t(z_i)\exp\{\alpha_t \cdot I(h_t(x_i) \neq y_i)\}} \tag{43}$$

La ecuación para la hipótesis fuerte:

$$f(x) = \frac{1}{|\alpha|}\sum_{t} \alpha_t \cdot h_t(x) \tag{44}$$

donde α es el parámetro que mide la importancia que se le asigna a la hipótesis.

También proponen un método para la clasificación del ruido con la versión extendida del algoritmo Adaboosting para la clasificación con múltiples clases, lo que para este caso utilizan como clasificador débil al método uno-vs-el resto [49] de la familia de los arboles de decisión.

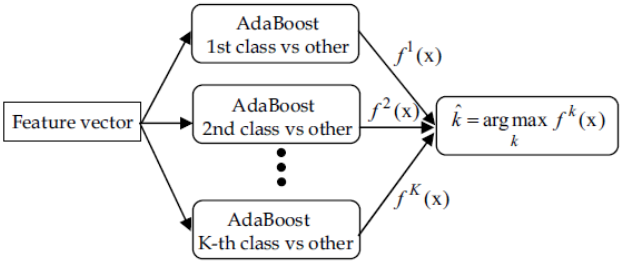

Fig. 6. Esquema que describe la estructura del sistema

La estructura del sistema se observa en la figura anterior, donde se modifica la forma de obtener la hipótesis fuerte.

Los experimentos desarrollados devolvieron mejores resultados en el funcionamiento de la clasificación y detección de ruido repentino usando Adaboosting que el método *GMM*. La razón es que el método Adaboosting puede hacer que los datos de entrenamiento se ajusten a las fronteras no lineales complejas, mientras que *GMM* no puede expresar la frontera compleja, porque los métodos *GMM* se basan en el cálculo de la media y la covarianza de los datos de entrenamiento.

En la Fig. 7 se observa una taxonomía de los sistemas de reconocimiento del habla en función de los clasificadores débiles y las características principales presentes en estos sistemas:

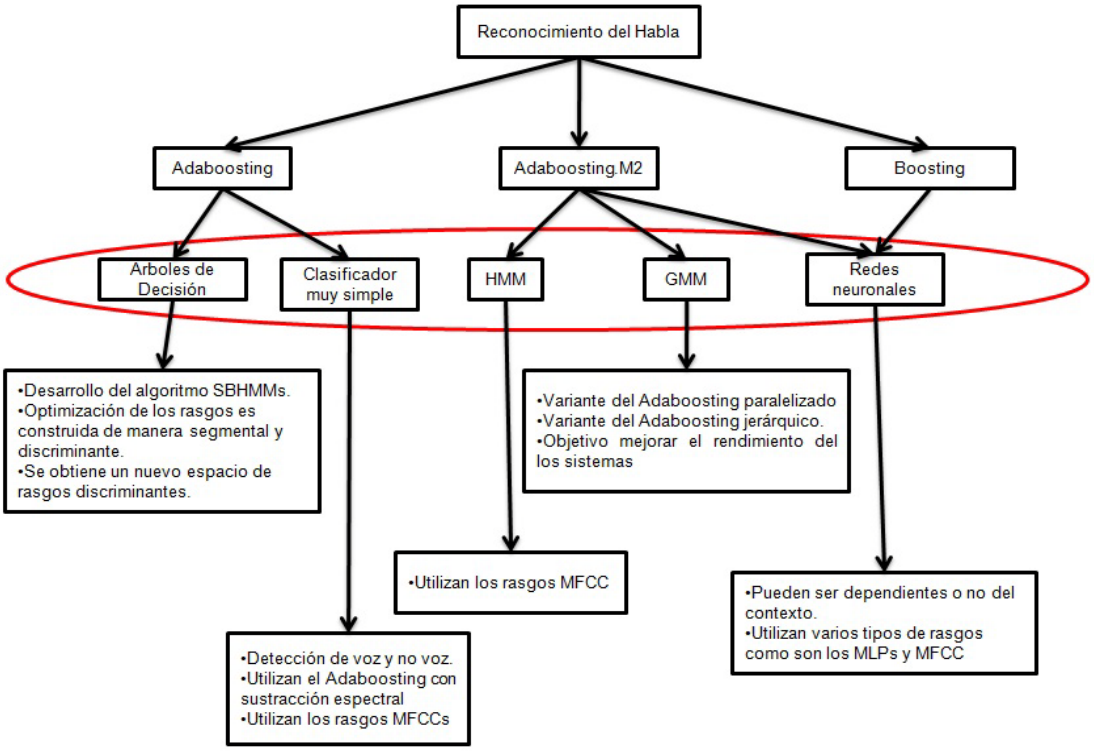

Fig. 7. Taxonomía de los sistemas de reconocimiento del habla que usan las variantes Boosting como algoritmo de clasificación

5.2 Reconocimiento del locutor

En la rama del reconocimiento del locutor se ha explorado muy poco la utilización del Adaboosting. Los primeros resultados fueron obtenidos en el año 2000 y en lo adelante se ha venido trabajando en base a mejorar el rendimiento (en función al tiempo de ejecución y el costo computacional), existiendo muy pocos detalles sobre la introducción de nuevos algoritmos para el reconocimiento del locutor mediante el uso de Adaboosting o cualquier otra variante del Boosting. En los siguientes epígrafes se realiza una descripción y análisis de los trabajos más importantes enfocados al reconocimiento automático del locutor.

5.2.1 Reconocimiento de locutor usando Adaboosting con clasificadores a partir de los árboles de decisión

La investigación plasmada en [50], desarrollada en el 2002 utiliza el algoritmo Adaboosting y toma como clasificadores débiles arboles de decisión para el reconocimiento de un conjunto cerrado de locutores dependientes del texto. Para esto utilizaron la versión del Adaboosting para una clasificación binaria y el clasificador C4.5 como clasificador débil.

El reconocimiento del locutor con Adaboosting y C4.5 fue representado en un sistema de la siguiente manera:

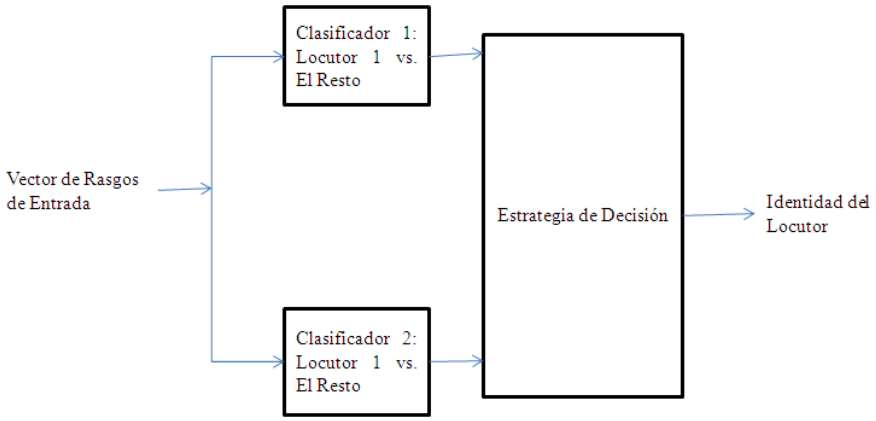

Fig. 8. Sistema de Reconocimiento del locutor utilizando Adaboosting y C4.5

Como rasgos iniciales se utilizan los Coeficientes Ceptrales de predicción Linear *(LPCC)*, de los cuales los coeficientes de orden 10 y sus respectivos coeficientes cepstrales Deltas componen el vector de rasgos de 20 componentes.

La estructura de este sistema brinda una gran ventaja, la cual divide un complejo problema de múltiple clasificación en M simple problemas de clasificación binaria.

En esta aplicación el algoritmo Adaboosting para una clasificación binaria toma como datos de entrada un conjunto S de N muestras a entrenar denotadas $S = \{x_i, y_i\}_{i=1}^{N}$, con $x_i \in X$, $y_i \in Y = \{0,1\}$, al igual que en su versión original. El resto de los procesos se realizan de igual manera que el Adaboosting en sus inicios. Las adaptaciones en esta aplicación caen sobre el clasificador débil C4.5 referenciado anteriormente, en el cual se cambia la forma de calcular el número total de instancias de cualquier conjunto i. Estas se redefinieron de la siguiente forma:

$$|T_i| = \sum_{i=1}^{n} w_i \tag{45}$$

y con esto se ha modificado $\text{freg}(C_j, T_i)$ para convertirse en la suma de pesos de todos los ejemplos que pertenecen a la clase de C_j.

A la hora del test se obtiene el locutor más probable a partir de una medida de cuán bien el modelo reclamado se expresa en términos de la relación entre la puntuación que produce el modelo solicitado y la puntuación que produce el modelo más competitivo siguiente.

$$R = \frac{S_v}{S_c} \tag{46}$$

donde S_v es la puntuación del modelo reclamado y S_c es la puntuación del modelo siguiente más competitivo. La relación entonces se compara con un umbral predefinido T_r, que los locutores pueden

ser específicos o independientes. Para esta aplicación los autores nos proponen umbrales independientes a los locutores, la cual es definida:

$$V = \begin{cases} 1 \; if \; R \geq T_r \\ 0 \; if \; R \geq \; T_r \end{cases} \tag{47}$$

Los experimentos realizados con el algoritmo propuesto alcanzaron mejores resultados en comparación con el clasificador C4.5.

5.2.2 Aprendizaje basado en GMM y Boosting para la verificación de locutores

En [51] los autores, basándose en la eficiencia de los modelos de mezclas Gaussianas para la verificación de locutores, y que son referenciadas como las más usadas en el estado de arte, proponen un nuevo método utilizando el aprendizaje de una de las variantes de Adaboosting basada sobre las *GMM*. Los autores, además de lo mencionado anteriormente se basaron en la idea intuitiva de que las *GMM* combinan linealmente una serie de modelos Gaussianos de acuerdo a un conjunto de mezclas ponderadas y que existen mejores medias a partir de la combinación de los distintos modelos de mezcla de Gaussianas.

Para el desarrollo de la propuesta los autores decidieron tomar la variante *"Discrete Adaboosting"*, la cual toma como datos de entrada un conjunto de N muestras de entrenamiento etiquetadas de la forma $\left(x_1, y_1, \ldots, (x_N, y_N)\right)$, donde $y_i \in \{1, -1\}$ son las etiquetas de las clases. En el cálculo del clasificador fuerte utilizan una versión normalizada:

$$H(x) = \frac{\sum_{m=1}^{M} \alpha_m h_m(x)}{\sum_{m=1}^{M} \alpha_m} \tag{48}$$

En el aprendizaje del clasificador débil los autores definieron utilizar como clasificadores débiles un conjunto de parámetros (μ_k, Σ_k) del modelo provenientes de 2 mezclas Gaussianas, las cuales representan 2 vectores de medias y 2 vectores de covarianza. Estos clasificadores débiles los definen a partir de la razón de verosimilitud (LLR).

$$h_k^{(M)}(x) = \log \frac{p(x|\Lambda_{SP}, w^{M-1})}{p(x|\Lambda_{UBM}, w^{M-1})} - \tau_k^{(M)} \tag{49}$$

Seleccionan el mejor clasificador débil de la forma:

$$h_M(x) = h_{k^*}^{(M)}(x) \tag{50}$$

donde k^* es el valor que minimiza las falsas alarmas $k^* = \arg\min_k FA\left(h_k^{(M)}(x)\right)$.

Los autores obtuvieron buenos resultados en los experimentos desarrollados utilizando la base de datos de la competencia NIST del 1996 para el reconocimiento del locutor. Estos resultados fueron comparados con las GMM adaptadas para el manejo de varios canales mejorándolos un 10%.

Uno de los aspectos pendiente en este trabajo es evaluar el algoritmo Adaboosting para otros niveles de rasgos, como los cepstrales.

5.2.3 Modelos ocultos de Markov utilizando el algoritmo Boosting como entrada y salida para la clasificación secuencial

En este trabajo [52], publicado en el 2005, los autores se basan en que los modelos ocultos de Markov logran cierta eficacia en el procesamiento de datos secuenciales ante un aprendizaje supervisado, sin embargo presenta dificultades a la hora de la selección de los modelos, teniendo en cuenta que la complejidad computacional es alta y el óptimo local no es estable.

Para la solución de dichos problemas utilizan los modelos ocultos de Markov de entrada y salida simple con estructuras topológicas diferentes como clasificadores bases del algoritmo Boosting, teniendo en cuenta que los (IOHMMs) simples tienden a tratar los problemas de clasificación secuencial sin la necesidad de una selección de rasgos.

En este trabajo modifican la variante Adaboosting.M1 donde los datos de entrada son:

1. Entrada:
   - Conjunto de P muestras de la forma $\left\{\left(u_1^{T_p}, y_p\right)\right\}_{p=1}^{P}$ donde y_p es la etiqueta, la cual $\epsilon\ \{0,1\}^M$ con $M \geq 2$.
   - Un repositorio de los IOHMMs simples ordenados en función de la complejidad topológica.
   - Condición de parada τ la es el numero de iteraciones.
2. La forma de inicialización es la misma descrita en el epígrafe que aborda esta variante del Adaboosting.
3. Posteriormente se llamada al clasificador base y se provee con el conjunto de entrenamiento $\left\{\left(u_1^{T_p}, y_p\right)\right\}_{p=1}^{P}$ y la distribución inicializada en el paso anterior.
4. Este paso se le aplica el algoritmo de máxima expectancia EM para capacitar los IOHMMs para la producción de la hipótesis h_t.
5. Otras de las modificaciones se encuentran en la ecuación utilizada en el cálculo del error:

$$\epsilon_t = \sum_{p:C(p)} D_t(p)$$

   - Siempre y cuando $D_t(p)$ sea correcto.

   $C(p) \equiv \arg\max_{1\leq m\leq M} h_t\left(u_1^{T_p}\right) \neq \arg\max_{1\leq m\leq M} y_p$ y $h_t\left(u_1^{T_p}\right)$ es el vector de salida de la hipótesis de la secuencia entrante.

El algoritmo desarrollado fue simulado para la identificación de locutores dependientes del texto.

5.2.4 Optimización del flujo con peso por LDA y Adaboosting para la verificación de locutor multi-flujos

En [53] podemos encontrar un método para la verificación de locutores mediantes los múltiples flujos de *HMM*. Los autores utilizan el algoritmo *LDA* (Análisis Discriminante Lineal) para estimar los pesos de los flujos automáticamente y optimizan esta operación mediante el uso del algoritmo Adaboosting. Como rasgos utilizan las informaciones espectrales y prosódicas.

La efectividad de este método se encuentra centrada en la integración de los rasgos segmentales y prosódicos donde:

- Cada vector de rasgos segmentales está constituido por 12 *MFCC*, sus deltas y el delta de la energía logarítmica.
- Cada vector de rasgos prosódicos presenta el $\log F_0$ y $\Delta \log F_0$. Donde la función F_0 es extraído mediante un método robusto al ruido basado en las transformaciones de Hough [54].

Estos dos tipos de rasgos son combinados en cada trama obteniendo un vector de rasgos segmentales-prosódicos.

Para alcanzar el modelo *HMM* segmentales-prosódicos *(SP-HMM)*, inicialmente se entrenan los vectores de rasgos por separado obteniendo un modelo por cada tipo de rasgos, *S-HMM* y *P-HMM*.

La novedad de este algoritmo se encuentra a la hora de la verificación de SP-HMM y toma de decisión, donde el algoritmo *LDA* se utiliza para la estimación del peso de los flujos de *HMM*.

Anteriormente para este paso se usaba la función discriminante como nos muestran en el epígrafe 3.1 de este trabajo donde:

$$z = q_{sp}(O_{sp}) - \theta = \lambda_s q_{sp}(O_s) + \lambda_s q_{sp}(O_p) - \theta \qquad (51)$$

donde O representa los rasgos y $q(O)$ al conjunto de rasgos. Y utilizan para la estimación de los pesos de los flujos, λ_s y λ_p; el método *LDA* como se muestra a continuación.

- Inicialmente la puntuación segmental y prosódica calculada del locutor reclamado y de los datos impostares son representadas en un espacio bidimensional.
- Posteriormente LDA se le aplica al espacio para obtener la función discriminante z.

A continuación se optimizan los pesos mediante el uso del algoritmo Adaboosting, el cual toma como clasificadores débiles las funciones discriminantes obtenidas en la estimación.

Esta nueva propuesta toma como datos iniciales a n el cual representa el número de datos del conjunto de entrenamiento (funciones discriminantes) y T el número de iteraciones. También recibe a $\{x_i\}\, donde\ (i = 1, \dots, n)$ que son las muestras de entrenamiento etiquetadas y $\{w_i\}(1, \dots, n)$ que representan los pesos de cada dato.

Se inicializan los pesos como normalmente lo realiza Adaboosting, e inicia el proceso de iteraciones comenzando por elegir *n* muestras con el duplicado de $\{x_i\}$ usando los pesos como distribuciones de probabilidad. Posteriormente se llama al método *LDA* con el objetivo de obtener la función discriminante lineal mencionada anteriormente. Con z_t se verifican todo el conjunto de entrenamiento y se calcula el error discriminante ponderado:

$$\epsilon_t \coloneqq \sum_{i:misclassify\ x_i} w_i \tag{52}$$

Con las condiciones de que si $0 < x_i \leq {}^1/_2$. En caso de que $\epsilon_t > {}^1/_2$ se invierte la decisión de la función discriminante lineal (z_t) y $\epsilon_t \coloneqq 1 - \epsilon_t$. Ahora cuando $\epsilon_t = 0$ se inicializa el peso y se retorna al paso de la elección. Posteriormente se obtiene el peso de z_t el cual es $c_t \coloneqq \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$.

Las actualizaciones de los pesos están dadas por:

$$w_i \coloneqq \begin{cases} w_i \times \epsilon^{-c_t}\ (i: clasifica\ a\ x_i\ con\ precision\ ) \\ w_i \times \epsilon^{c_t} (i: clasificaca\ a\ x_i\ incorrectamente) \end{cases}$$

Finalmente el clasificador final se obtiene como se muestra a continuación:

$$z = \sum_{t=1}^{T} \{c_t \times sign(z_t)\} \tag{53}$$

Los autores teniendo en cuenta que el resultado obtenido del algoritmo Adaboosting no se puede usar directamente para estimar el flujo de pesos, debido que no es una función discriminante lineal; realizan una aproximación de z por $z = \sum_{t=1}^{T} c_t z_t$, posteriormente z es representada como una función discriminante lineal de la forma:

$$z = \lambda_s^{(boost)} q_s(O_s) + \lambda_p^{(boost)} q_p(O_p) + \theta^{(boost)} \tag{54}$$

donde $\lambda_s^{(boost)} = \sum_{t=1}^{T}\left(c_t \lambda_s^{(t)}\right)$, $\lambda_p^{(boost)} = \sum_{t=1}^{T}\left(c_t \lambda_p^{(t)}\right)$ y $\theta^{(boost)} = \sum_{t=1}^{T}\left(c_t \theta^{(t)}\right)$.

Estas modificaciones traen consigo que para la estimación del valor de los flujos de pesos segmental y prosódicos son normalizados $\frac{\lambda_s^{(boost)}}{\lambda_s^{(boost)} + \lambda_p^{(boost)}}$, $\frac{\lambda_p^{(boost)}}{\lambda_s^{(boost)} + \lambda_p^{(boost)}}$.

Para la realización de los experimentos utilizaron varias sesiones y al conjunto de entrenamiento se le adicionaron 30db de ruido blanco. Ya a la hora de Test se contaminaron las señales con 5, 10, 15, 20 y 30db. Los resultados obtenidos demostraron que en el rango entre 20 y 30db de condición SNR, el método de optimización de los pesos usando Adaboosting es efectivo.

No obstante plantean para un trabajo futuro:

- La investigación y desarrollo de un método para la estimación de los umbrales de verificación.
- Investigar el método de estimación del flujo de pesos usando el puntaje $q_{sp}(O_{sp})$ y compararlo con los resultados de este trabajo.
- Comprobar la efectividad del método propuesto ante la presencia de otras variedades de ruido.

5.2.5 Método de estimación del flujo de pesos y el umbral usando Adaboosting para la verificación de locutores con multi-flujos

Este trabajo [55] es la continuación de la investigación mencionada en el epígrafe 5.2.4, por los mismos autores, publicada en el año 2006.

Este artículo propone un método automático para optimizar la estimación de los pesos de los flujos y el umbral de decisión usando los múltiples flujos del *HMM*, donde se observa la efectividad en la integración de las informaciones segmentales y prosódicas. Para optimizar los umbrales de decisión y los pesos de los flujos se utiliza el algoritmo Adaboosting, el cual toma como clasificador débil las funciones discriminantes obtenidas mediante el método *LDA*, el cual presenta la función de optimizar el umbral de decisión al igual que el trabajo anteriormente publicado en el 2005 por los autores.

La diferencia viene dada por el algoritmo Adaboosting, mediante el uso del rango de falsas aceptaciones *(FAR)* y el rango de falsos rechazos *(FRR)* para optimizar el rango de error del objetivo buscado. Para esto realizaron una serie de modificaciones y adiciones en las ecuaciones del Adaboosting, como se observa a continuación:

En este método se van a realizar los mismos primeros pasos que el método del artículo mencionado anteriormente (la inicialización y la selección de las muestras). Ya a partir de la llamada al algoritmo *LDA* para la obtención de la función discriminante linear comienzan las modificaciones, en primer lugar modifican la ecuación de la función discriminante linear sumándole un valor δ_{t-1}, el cual se utiliza para balancear *FAR* y el *FRR*. Otras de las modificaciones se presenta a la hora de calcular el error, además de obtener el error de entrenamiento de igual forma; se calculan los errores discriminantes de *FAR* y *FRR*, ecuaciones c) y d) respectivamente; teniendo en cuenta que anteriormente se obtienen los valores de estas dos variables como se observa a continuación en las ecuaciones a) y b):

$$FAR_t = \frac{\sum_{i:} x_i \ \ es\ la\ FA^1}{\sum_{i:} x_i \ \ es\ un\ \ impostor^1} \tag{55}$$

$$FRR_t = \frac{\sum_{i:} x_i \ \ es\ el\ FR^1}{\sum_{i:} x_i \ \ es\ el\ locutor^1 \ \ reclamado} \tag{56}$$

$$\epsilon_{FA} = \frac{\sum_{i:} x_i \ \ es\ la\ FA^{w_i}}{\sum_{i:} x_i \ \ es\ un\ \ impostor^{w_i}} \tag{57}$$

$$\epsilon_{FR} = \frac{\sum_{i:} x_i \ \ es\ el\ FR^{w_i}}{\sum_{i:} x_i \ \ es\ el\ locutor^{w_i} \ \ reclamado} \tag{58}$$

En este método la forma de actualizar los pesos w_i es modificada ya que para este método utilizan las funciones de costo:

$$w_i = \begin{cases} w_i \ \times \ \epsilon^{-C_{cost_t}} \ \ (i: x_i \ clasificada \ correctamente) \\ w_i \ \times \ \epsilon^{C_{cost_t}} \ \ (i: x_i \ clasificada \ incorrectamente) \end{cases} \tag{59}$$

donde C_{cost_t} se define a partir de las funciones de costo de la forma:

$$C_{cost_t} = \frac{1}{2}\log\frac{1 - cost_t}{cost_t} \tag{60}$$

donde $cost_t$ se obtiene mediante la siguiente ecuación:

$$cost_t = (1-\alpha)\cdot\epsilon_{FA} + \alpha\cdot\epsilon_{FR} \quad (61)$$

El método propuesto ajusta la tasa de falsas aceptaciones y falsos rechazos a razón de la tasa del EER[7] y optimiza la estimación de los umbrales de decisión y el flujo de pesos.

5.2.6 Detector de voz robusto al ruido para el reconocimiento del locutor

Este trabajo [56] fue desarrollado en el 2006 por los investigadores del Centro de Aplicaciones de Tecnología de Avanzada (CENATAV).

En este trabajo se aborda el problema del reconocimiento del locutor ante la presencia de muestras de voz distorsionada por ruido blanco aditivo. Para esto se seleccionan las muestras de voz mediante el detector de voz *(VAD-Boost)*, basado en el algoritmo Adaboosting.

Este detector de voz utiliza las técnicas Boosting con árboles de decisión como clasificadores débiles al cual combinan con una función de periodicidad, que no es más que una función que devuelve el grado de periodicidad que tiene un vector determinado, sea X un conjunto de vectores w de tamaño no necesariamente fijo, se define la clase de funciones:

$$f_d^p : X \to [-1, 1] \quad (62)$$

Para evaluar un vector $w[k]$ de tamaño k, este se divide en p partes de tamaño d y debe expresar hasta que punto dividiendo w de esta manera se obtienen sub-vectores similares, o lo que es lo mismo, debe decir si w tiene un período p de tamaño d.

El detector de voz VAD-Boost recibe la señal de voz representada de forma diferente, teniendo la idea de no tener en cuenta las características innecesarias del habla. De ahí que para esto utilizan los cruces por cero, los cuales consisten en convertir el segmento de la señal que se está analizando en un vector de tamaño igual a la cantidad de cruces por cero de ésta. En cada elemento de este vector se almacena la suma de los valores bajo la curva desde el anterior cruce por cero (Ver Figura 9).

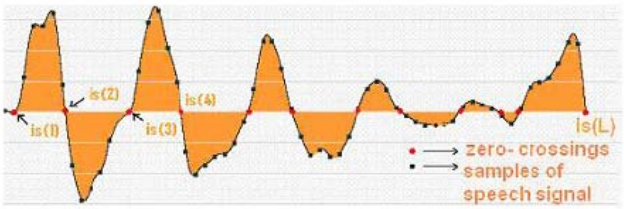

Fig. 9. Representación en cruces por cero

La idea es obtener un clasificador h_d^p mediante una función de periodicidad para entrenar con sonidos sonoros como muestras positivas y sonidos sordos para las negativas. Para esto utilizan un umbral determinado según:

$$h_d^p = \begin{cases} +1 & f_d^p \geq \theta \\ -1 & f_d^p \leq \theta \end{cases}$$

Esto solo lo utilizan para la selección de los sonidos sonoros o sordos de la señal del habla, porque para crear el árbol de clasificación se utiliza el algoritmo Adaboosting utilizado en [57], con criterio de decisión en función al número de clasificadores débiles.

Este trabajo trajo mejoras en función al costo computacional y al tiempo de ejecución, además de obtener mejores resultados relativamente en función del EER (tasa del rango de error), que no es más que el valor del error del sistema cuando el umbral de decisión es tal que el porcentaje de falsas aceptaciones es igual al de falsos rechazos.

[7] Consiste en medir el error del sistema cuando el umbral decisión es tal que el porcentaje de falsas aceptaciones es igual al de falsos rechazos

5.2.7 Plataforma discriminativa-generativa usando métodos de combinación para la verificación de locutores dependientes del texto

En este trabajo [58] publicado en el 2007, los autores proponen un acercamiento más robusto mediante el uso del acoplamiento entre los modelos discriminativos y modelos generativos para la verificación de locutores dependientes del texto.

En esta propuesta utilizan rasgos como:

- El resultado de la verosimilitud inicial, que es normalizada en función de las tramas, obtenida en cada uno de los pasos en el reconocimiento.
- La diferencia entre la verosimilitud inicial y la razón de la verosimilitud.
- La duración.

En el entrenamiento utilizaron las técnicas Boosting utilizando como clasificador débil a los árboles de decisión.

La estrategia para la selección de rasgos discriminativos nos es más que en cada iteración se selecciona un elemento (la dimensión) de X(matriz de rasgos) y a representa el umbral que minimiza el error de entrenamiento. Estos dos parámetros (dimensión y umbral) son la articulación de la selección para minimizar el error. En este caso la hipótesis se obtiene con la siguiente ecuación:

$$h_p(x) = I\left(x^p > K_p\right) \tag{63}$$

donde x^p es el elemento de X, K_p es el umbral correspondiente y I es la función indicador que retorna 1 si es correcta la clasificación o 0 en cualquier otro caso.

El algoritmo propuesto presenta una mejora del 36,41% con respecto a la tasa del EER con la utilización de las técnicas Boosting en comparación con el uso de la razón de la verosimilitud con la puntuación de los modelos generados.

Los autores dejan pendientes investigaciones sobre los otros rasgos derivados del modelo generativo, como por ejemplo las medias de las mesclas Gaussianas.

5.2.8 Modelos de mezclas gaussianas y Boosting a través del análisis discriminante

Este trabajo [59], publicado en el 2008 se basa en que las GMM pueden aproximarse arbitrariamente a una distribución de probabilidades, la cual es una herramienta fuerte para la representación de rasgos y la clasificación. No obstante es insuficiente para el entrenamiento de los datos ante la presencia de grandes dimensiones. Por estos motivos los autores proponen una integración con los métodos Boosting, *GMM* y la variante de *LDA (MDA)* denominado *"BoostGMM-DA"*. El método propuesto es evaluado para el reconocimiento de emociones en el habla.

La propuesta realizada inicialmente toma las muestras de entrenamiento de la forma $\{x_i, c_i\}$, donde $x_i \epsilon X$ y $c_i \epsilon \{1, \dots, C\}$ y T es el numero de iteraciones. Posteriormente en cada iteración las muestras serán proyectadas mediante el algoritmo *MDA* en un sub-espacio $Y_t = W_t^T X_t$, los cuales son entrenados las *GMM* usando el algoritmo F-J:

$$p_t(x|W_t, c) = p_t(y|c), c = 1, \dots, C \tag{64}$$

y el algoritmo *MER* (rango de error mínimo):

$$h_t(x) = \arg \max_{c\epsilon\{1,\dots,C\}} p_t(x|W_t, c)p_t(c|W_t) \tag{65}$$

El error se obtiene mediante una sumatoria de los pesos desde $i = 1$ hasta n, en caso de que el error sea mayor que 0,5 se detiene, en caso contrario $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$. Para actualizar los pesos se utiliza la siguiente fórmula:

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) e^{2\alpha_t [c_i \neq h_t(x_i)]} \tag{66}$$

donde Z_t es el factor de normalización.

El resultado final de cada iteración es una función de densidad probabilística de la forma:

$$p(x|c) = \sum_{t=1}^{T} \alpha_t p_t(x|W_t, c) \tag{67}$$

y para obtener el clasificador final o fuerte lo hacen de la forma:

$$H(x) = \arg\max\ p\,(c|x) = \arg \max_{c \in \{1,\ldots,C\}} p(x|c)p(c) \tag{68}$$

Para comprobar este algoritmo experimentan con el reconocimiento de emociones en el habla y lo comparan con el algoritmo *GMM-MER* obteniendo un 6% de mejoras con relación a la línea base demostrando que el algoritmo propuesto en este artículo es más eficiente y además solo depende del número de clases y es independiente de las dimensiones de los rasgos.

5.2.9 Rasgos binarios a partir del Boosting para la verificación de locutores robusto al ruido

En [60], los autores proponen un método para la selección de rasgos robustos para la verificación del locutor mediante la utilización del algoritmo Adaboosting.

Para esto escogieron una de las variantes más usadas del Adaboosting, Discrete Adaboosting; que realiza una combinación linear simple de los rasgos seleccionados para la creación del clasificador final.

En primer paso las señales de habla de entrada son interceptadas en tramas y se le aplica una transformación espectral T, para crear una secuencia de vectores de magnitud espectral, donde los vectores se encuentran de la forma $\vec{X} = [X(1), \ldots, X(N)]^T$.

La transformación espectral T puede ser realizada de dos formas:

1. Una simple transformación de Fourier Discreta *(DFT)*.
2. DFT seguido de filtros Mel [5]. En este caso $\vec{X}$ representa las salidas de los filtros Mel y N el número de filtros.

Los rasgos binarios son obtenidos de los vectores $\vec{X}$ de la forma $\emptyset_i: \Re^N \rightarrow \{0,1\}$ y es definido completamente por los siguientes 3 parámetros: dos índices $k_{i,1}, k_{i,2}$ y un umbral θ_i. Para la forma 1 $\{k_{i,j}\}$ representa el índice de frecuencia y para la forma 2 representa el índice de los filtros.

El algoritmo Discrete Adaboost es utilizado para la selección de los rasgos binarios, el cual toma como datos de entrada los vectores de entrenamiento correspondientes a las etiquetas de las clases $y_j \in \{0, 1\}$ $(0: impostores, 1: clientes)$, el número de rasgos a seleccionar N_f,N_{tr}^* número de vectores de entrenamiento que se probaran aleatoriamente en cada iteración del algoritmo $(N_{tr}^* < N_{tr})$. Se inicializan los pesos al igual que como se menciona en epígrafes anteriores.

La salida del algoritmo es una secuencia de los mejores rasgos seleccionados $\{\emptyset_n^*\}_{n=1}^{N_f}$. Estos son linealmente combinados, para la creación del clasificador fuerte F de la forma:

$$F(\vec{X}) = \sum_{n=1}^{N_f} \alpha_n \emptyset_j(\vec{X}) \tag{69}$$

5.2.10 Mejoramiento del algoritmo Adaboost mediante el uso de VQMAP para la identificación de locutores

Teniendo en cuenta que el método Adaboosting puede mejorar el funcionamiento de los clasificadores base. Teniendo en cuenta que el estado del arte existente de los clasificadores para la identificación de locutores, *GMM-MAP* y *SVM*, son fuertes para los rasgos extraídos de la señal acústica y apoyándose en las características beneficiosas del Adaboosting, mencionadas anteriormente en el epígrafe 3.1; los atores plantean que tomando los resultados del método *SVM* como clasificador débil para el Adaboosting los resultados mejorarían notablemente. No obstante la utilización del estado del arte traería consigo un aumento del tiempo de reconocimiento es insatisfactorio. De ahí que los autores basándose en [61] donde realizan una adaptación máximo a posteriori *(MAP)* del modelo de cuantificación vectorial *(VQ)*, debido a que los modelos (VQ) por si solo sin la adaptación su resultado es insatisfactorio; teniendo en cuenta que el tiempo de entrenamiento es mucho menor y utilizándolo como clasificador débil del Adaboosting se lograría un mejor modelo que caracterice al locutor.

En [62] se basa en el aprendizaje de la extensión Adaboost.M1 del método Adaboosting utilizando *Maximum a Posteriori Vector Quantization Model (VQMAP)* como clasificador débil. Este se crea a partir de un nuevo conjunto de muestras generadas en función de la distribución discreta por muestra.

Pseudocódigo del método:

Datos de entrada:

- Conjunto de vectores de rasgos etiquetados de entrenamiento de la forma: $(X, Y) = \{(x_1, y_1), \dots, (x_T, y_T)\} | \; y_i = M_0 \in M = \{1, \dots, M\}$ donde x_i es el conjunto de rasgos por etiquetas, M es el número de clases y y_i es la etiqueta que la identifica.
- Modelo *UBM* utilizado para la creación del modelo *VQMAP* $\lambda_{UBM} = \{\mu_k, j = 1,2, \dots, K\}$.

1.Inicialización de los pesos de las muestras de entrenamiento $w_i = {}^{1}/_{N}, i = 1,2, \dots, N$

2.Iteraciones hasta $t \leq S_{Boost}$

a. Se calcula la distribución discreta del conjunto de entrenamiento $p^t = w^t / \sum_{i=1}^{N} w_i^t$.
b. Se genera un subconjunto de entrenamiento (X^t, Y^t) basado en la distribución p^t.
c. Se entrena el clasificador $\Theta^{t}_{\mathrm{VQMAP}}: \{\lambda_1, \lambda_2, \dots, \lambda_{\mathrm{M}}\}, \lambda^{t}_{\mathrm{m}} = \left\{\mathrm{c}^{t}_{\mathrm{k}}, \mathrm{k} = 1,2, \dots, \mathrm{K}\right\}$ basado en el nuevo subconjunto de entrenamiento (X^t, Y^t).
d. Se realiza una clasificación $\Theta^{t}_{\mathrm{VQMAP}}$ sobre las muestras iniciales (X, Y) y se obtiene una sub hipótesis $h^t: X^t \rightarrow Y^t$
e. Se obtiene el error de entrenamiento de h^t $\varepsilon^t = \sum_{i=1}^{N} p_i^t [\![h^t(x_i^t) \neq y_i]\!]$.
   Condición de parada: si $\varepsilon^t > \frac{1}{2}$ o $S_{Boost} = t - 1$ se sale del ciclo.
f. Se obtiene el valor $\beta^t = \varepsilon^t / (1 - \varepsilon^t)$ y se actualiza los pesos de las muestras de la forma $w_i^{t+1} = w_i^t (\beta^t)^{1 - [\![h^t(x_i^t) \neq y_i]\!]}$

3.Salida combinación de clasificadores $h_f(x) = \arg\max_{1 \leq \mathrm{m} \leq \mathrm{M}} \sum_{\mathrm{t}=1}^{\mathrm{T_{Boost}}} \left(\log \frac{1}{\beta^{\mathrm{t}}}\right) [\![h^t(x) = y]\!]$.

Las experimentaciones se realizaron utilizando la base de datos Timi [63] basándose en su complejidad. Hicieron varios experimentos con diferentes cantidades de segmentos acústicos en el entrenamiento y tomaron en la fase de identificación diferentes números de locutores. Estas experimentaciones brotaron resultados beneficios en parte, principalmente enfocados en el tiempo de ejecución teniendo en cuenta que este es más rápido que el método *GMM-MAP*, no obstante no supera los resultados de efectividad del mismo, pero si los de *VQMAP.*

A continuación se observa una taxonomía de los sistemas de reconocimiento del locutor en función de los clasificadores débiles y las características principales presentes en estos sistemas:

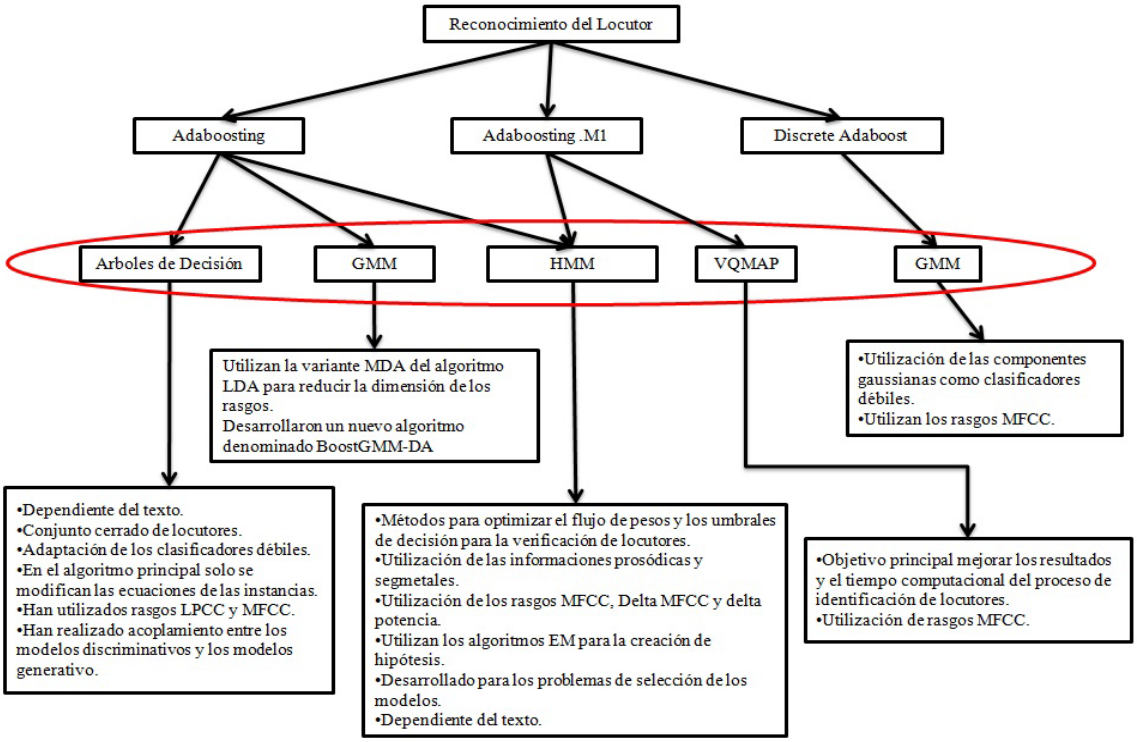

Fig. 10. Taxonomía de las variantes Boosting para el RAL en función de los clasificadores débiles

6 Conclusiones

En el reconocimiento automático del locutor *(RAL)* se ha explorado muy poco la utilización del Adaboosting. Los primeros resultados fueron obtenidos en el año 2000 y en lo adelante se ha venido trabajando en base a mejorar el rendimiento (en función al tiempo de ejecución y el costo computacional) y a su vez alcanzar mejores resultados en la efectividad de los sistemas de reconocimiento. Existiendo pocos detalles sobre la introducción de nuevos algoritmos para el reconocimiento del locutor mediante el uso de cualquier variante del Boosting.

Tabla 1. Comparación de los métodos más relevantes en la bibliografía.

Métodos	Variante	Aplicado a	Rasgos Utilizados	Resultados ERR	Línea base	Costo Computacional
Boosted-GMM	Discrete Adaboost	Creación de un nuevo modelo del locutor.	• Vectores de rasgos de 10 MFCC y sus deltas, mas la energía.	Mejora en un 10%	GMM-MAP	----------
HMM	Adaboost. M1	Optimización en la selección de los modelos	• Vectores de rasgos de 12 MFCC y sus deltas, mas la energía. • Rasgos Prosódicos $\log F_0$ y $\Delta \log F_0$.	10%	HMM	----------
Vad-Boost	Adaboost	Selección de tramas acústicas en la señal de audio.	• Vector de rasgos 12 MFCC y sus deltas.	7.8% con 10 db	GMM-UBM	Mejora el costo computacional.
BoostGMM-DA	Adaboost con varios ajustes	Creación de un nuevo modelo del locutor.	Vectores de rasgos de 42 elementos: 12 MFCC, el logaritmo de la energía y pitch (f_0).	En función del rango de reconocimiento obtuvo una mejora de 6.39%.	GMM-MER	----------
VQMB	Adaboost. M1	Creación de un nuevo modelo del locutor.	Vectores de rasgos MFCC	Con respecto a: • VQ-MAP mejora un 1.1% • GMM-MAP empeora un 4.9%	VQ-MAP y GMM-MAP	Es 9 veces más rápido el reconocimiento que el GMM-MAP.

Hasta el momento se ha experimentado la utilización de algunas variantes o acoples de algoritmos de clasificación con los métodos de la familia Boosting, principalmente con el método Adaboosting, las extensiones Adaboosting.M1y Discrete Adaboost, enfocados la mayoría de mejorar los resultados de los sistemas RAL apoyados principalmente en la selección y obtención de nuevos y mejores rasgos, en la optimización de las predicciones de los umbrales de decisión y la minoría enfocada a la creación de un nuevo clasificador que sea capaz de identificar de forma más eficiente que el estado del arte *GMM-MAP-UBM*.

Dentro de los trabajos basados en la creación de un nuevo clasificador se encuentra el trabajo desarrollado en el 2003, donde plantean que la combinación de las medias de una serie de modelos de mezclas gaussianas *(GMM)* robustece el modelo que representa al locutor. Y obtuvieron una mejora de un 10% comparados con las GMM adaptadas y utilizando la base de datos Nist96. Otros de los trabajos más enfocados a nuestro objetivo se observa en [59], en el cual crean un modelo *GMM* a partir de una proyección de los rasgos acústicos en un nuevo espacio utilizando *MDA* y utilizando la metodología del Adaboosting para obtener un clasificador fuerte que sea más discriminante. Este resultado se aplicó al reconocimiento de emociones mediante el habla obteniendo una pequeña mejora en el porciento de reconocimiento comparado con las *GMM*. Aquí se utilizan vectores de rasgos *MFCC* de 42 dimensiones lo cual provocaría un aumento en el tiempo de ejecución del sistema y con esto la complejidad del mismo. El trabajo más reciente enfocado a nuestro objetivo es el representado en [62], donde parados sobre las teorías de Boosting nos plantean que las *GMM-MAP* y *SVM* son muy fuerte ante datos acústicos por lo que no representan un clasificador débil, y por tanto no es adecuada su utilización por cuestiones de sobre ajuste del modelo, de ahí que se plantearon la utilización del aprendizaje del Adaboosting y con clasificadores débiles el modelo *VQ*-MAP teniendo en cuenta que es una forma simple de representar las *GMM-MAP*. Las experimentaciones brotaron resultados donde superan los obtenidos con *VQ-MAP* pero no los plasmados por las *GMM-MAP*, no obstante aumenta el tiempo de ejecución de los sistema de reconocimiento de locutores.

Basándonos en lo explicado anteriormente de los métodos pertenecientes a la familia Boosting y sus aplicaciones en la rama del reconocimiento del locutor y del habla podemos llegar a varias conclusiones:

6.1 Funcionamiento y eficiencia de los algoritmos Boosting

Los métodos existentes en la actualidad, incluidos dentro de la familia de algoritmos Boosting, se han venido desarrollando en el transcurso del tiempo con el objetivo de mejorar el rendimiento ante las tareas de clasificación. Los principales pasos de avances han sido dirigidos a las formas de obtención de los pesos y la actualización de los mismos, para esto se ha utilizado diferentes estructuras (funciones de pérdidas, funciones de regresión e integraciones con otros algoritmos). No obstante, los resultados obtenidos en estos algoritmos presentan una fuerte relación con el área donde son utilizados, las adaptaciones de los algoritmos han estado particularizadas en función de los datos a utilizar o sobre qué se enfoca el algoritmo. Estas adaptaciones traen consigo diferentes restricciones, como en el caso de los clasificadores débiles que trabajan solamente sobre determinados datos o reajustes en las hipótesis de salida. También podemos decir que todas las variantes creadas a partir del Boosting dependen de que los clasificadores débiles presenten menor error de clasificación que una selección aleatoria. La variante más adecuada a utilizar es Adaboosting.M1 o Adaboosting.M2 dado que es una generalización del algoritmo Adaboosting modificado para la clasificación de múltiples clases.

6.2 Rasgos utilizados en la rama del reconocimiento del locutor y del habla

La mayoría de los trabajos realizados aplicados al reconocimiento del locutor y del habla utilizan los rasgos *MFCC* (Coeficientes Cepstrales en Frecuencia Mel) y en unos pocos la información prosódica y todos proponen como trabajo futuro la búsqueda o utilización de otros tipos de rasgos más robustos al ruido y discriminativos.

6.3 Clasificadores débiles utilizados en aplicaciones del reconocimiento del locutor y el habla

Se ha utilizado una amplia gama de algoritmos de clasificación ya existentes, no obstante se ha trabajado muy poco en el espacio de las Mezclas Gaussianas, las cuales son obtenidas con el algoritmo que representa el estado del arte en el reconocimiento del locutor GMM-UBM; en [51] particularmente solo se utilizan 2 mezclas Gaussianas como clasificador. Teniendo en cuenta la información existente en la señal de voz podemos decir que solo dos vectores de medias y varianzas respectivamente no contienen toda la información necesaria y en ese caso se perdería información que pudiera ser de gran utilidad.

Partiendo de las conclusiones realizadas y que existe en los modelos *GMM-MAP-UBM* componentes gaussianas que son exactamente iguales a las componentes del *UBM* y teniendo en cuenta que esto provoca redundancia en la información contenida en el modelo, podemos decir que la utilización de las componentes gaussianas contenidas dentro del modelo *GMM-MAP-UBM* de forma independiente se puede ver como un clasificador débil. Los cuales pueden ser combinados para obtener un clasificador fuerte y además eliminar la redundancia y con esto un sistema de reconocimiento automático del locutor más rápido.

7 Líneas de desarrollo

El estado del arte de los métodos de reconocimiento del locutor se encuentra apoyado en la utilización de los súper vectores de los modelos GMM-UBM-MAP [4] como rasgos para el clasificador SVM. Teniendo en cuenta que dichos modelos adaptados del UBM contienen todas las componentes Gaussianas del UBM, se provoca una redundancia apreciable en la información contenida en el modelo de cada locutor, la cual afecta en ocasiones la efectividad y robustez de los clasificadores. Si se lleva a cabo una inteligente selección de aquellas componentes Gaussianas que presenten un mayor carácter discriminativo sobre los rasgos del locutor desconocido y se lleva a cabo su posterior combinación, se

pudiera obtener una mejora en la efectividad, robustez y en la eficiencia de los resultados del reconocimiento del locutor.

Para esto las técnicas Boosting aplicadas al espacio de las Mezclas Gaussianas constituyen una interesante y no explorada línea de trabajo, para encontrar una nueva combinación de componentes Gaussianos, capaz de mejorar los resultados en el reconocimiento del locutor.

Teniendo en cuenta el estudio realizado, se pueden definir un grupo de tareas posibles a desarrollar que posibilitarían el mejoramiento de los resultados en reconocimiento del locutor:

1. Seguir profundizando en la línea del Boosting para la clasificación ante la presencia de múltiples clases.
2. Encontrar nuevas estructuras matemáticas para optimizar la estimación y actualización de los pesos.
3. Utilización o desarrollo de nuevas formas de combinación de los clasificadores débiles para robustecer el clasificador final.
4. Utilización de nueva información proveniente de la señal de voz que caracteriza al locutor, para mejorar los resultados ante la variabilidad de canal y el estado del locutor.
5. Búsqueda u obtención de un conjunto de clasificadores débiles genérico que sean capaces de adaptarse a cualquier locutor deseado.
6. Búsqueda o selección de rasgos más eficientes y robustos para la clasificación de locutores ante la variabilidad del canal, la presencia de ruido y el estado del locutor.

Referencias bibliográficas

1. Sadaoki Furui, “50 years of progress in speech and speaker recognition”.Department of Computer Science Tokyo Institute of Technology. 2005.
2. Campbell J.P.: Speaker Recognition: A Tutorial. Proceedings of the IEEE, Vol. 85, No. 9, pp. 1437-1462, 1997.
3. Douglas A. Reynolds, “Automatic Speaker Recognition: Current Approaches and Future Trends”. MIT Lincoln Laboratory, Lexington, MA USA. 2001.
4. Tomi Kinnunen, Haizhou Li. “An overview of text-independent speaker recognition:From features to super vectors”. 2010.
5. Frederic Bimbot, Jean-Francois Bonastre, Corinne Fredouille, Guillaume Gravier, Ivan Magrin-Chagnolleau, SylvainMeignier, Teva Merlin, Javier Ortega-Garcıa, Dijana Petrovska-Delacretaz, and Douglas A. Reynolds. “A Tutorial on Text-Independent Speaker Verification”. EURASIP Journal on Applied Signal Processing 430–451. 2004.
6. RT.: Rafael Fernández, José R. Calvo y Gabriel Hernández, “Métodos de Extracción, Selección y clasificación de Rasgos Acústicos para el Reconocimiento del Locutor”. Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/ 218 y 222, Siboney, Playa, Habana, Cuba.
7. Hanzen, L. Salomon, P.: “Neural Networks Ensembles”. IEEE Trans. Pattern Analysis and Machine Intelligence, 12, p. 993-1001, 1990.
8. Krogh, A. y Vedelsby, J.: “Neural Networks Ensembles, Cross Validation and Sakoe H., 1995.
9. Dietterich, T. G. Ensemble Methods in Machine Learning. In Proceedings of the 1st International Workshop on Multiple Classifier Systems, pages 1–15. Lectures Notes in Computer Science, 2000.
10. Breiman, L. Bagging predictors. Machine Learning, 24, 123–140, 1996.
11. Antonio Solanas y Vicenta Sierra, “Bootstrap: fundamentos e introducción a sus aplicaciones,” Universidad de Barcelona, 1992.
12. Dietterich, T.G.: “Ensemble methods in machine learning”. En Multiple Classifier Systems, Cagliari, Italia. 2000.
13. Skurichina, M.: “Stabilizing Weak Classifiers”. Tesis Doctoral, Delft University of Technolog, Delft, Holanda. 2000.
14. Schapire, R., “The Strength of Weak Learnability. Machine Learning,” 1990.
15. Freund, Y., Schapire, R., Boosting a weak learning algorithm by majority. Information and Computation, 1995.
16. David H. Wolpert. Stacked generalization. Neural Networks, 5:241-259, 1992.

17. Bauer, E. and R. Kohavi. An empirical comparison of voting classification, 1999.
18. Freund, Y., Schapire, R. Experiments with a New Boosting Algorithm. in Machine Learning: Proceedings of the Thirteenth International Conference. 1996.
19. Schapire, R., Freund, Y., "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. journal of computer and system sciences". 55: p. 119-139. 1997.
20. Morgan D.B. y Scofield C.L.: "Neural Networks and Speech Processing", Kluwer Academic Publishers. 1991.
21. Quinlan, J. R. Simplifying Decision Trees. Simplifying Decision Trees. International Journal of Man-Machine Studies 27(3): 221-234, 1987.
22. J. Friedman, T. Hastie and R. Tibshirani, Additive Logistic Regression: a Statistical View of Boosting, The Annals of Statistics, 28: 337-374, 2000.
23. Bagging and Boosting with Dynamic Integration of Classifiers, Alexey Tsymbal, Seppo Puuronen, Department of Computer Science and Information Systems, University of Jyväskylä, Finland. 2000
24. G. I. Webb. MultiBoosting: A technique for combining boosting and wagging. Machine Learning, 2000.
25. Y. Freund. An adaptive version of the boost by majority algorithm. Machine Learning 43, 2001.
26. McDonald, R., Eckley, I. y Hand, D. A Multi-Class Extension to the BrownBoost Algorithm, 2003.
27. P. Buhlmann and B. Yu. "Invited discussion on 'Additive logistic regression: a statistical view of boosting". The Annals of Statistics, 28(2):377–386, April 2000.
28. Stan Z. Li, ZhenQiu Zhang, Heung-Yeung Shum, HongJiang Zhang FloatBoost Learning for Classification, 2003.
29. P. Pudil, J. Novovicova, and J. Kittler. "Floating search methods in feature selection". Pattern Recognition Letters,(11):1119–1125, 1994.
30. Nikunj C. Oza, Boosting with Averaged Weight Vector,Computational Sciences Division NASA Ames Research Center. 2003.
31. Jyrki Kivinen y Manfred K.Warmuth, "Boosting as Entropy Projection". In Proceedings of the Twelfth Annual Conference of Computational Learning Theory, 134-144, 1999.
32. Vezhnevets, A., Vezhnevets, V., "Modest AdaBoost" Teaching AdaBoost to Generalize Better. Moscow State University, 2005.
33. Ji Zhu, Hui Zou, Saharon Rosset and Trevor Hastie Multi-class AdaBoost, Statistics and Its Interface Volume 2, 349–360, 2009.
34. Morgan Kaufmann y Quinlan, J. R. C4.5: Programs for Machine Learning. 1993.
35. Paul Viola, Michael J. Jones Rapid. Object Detection Using a Boosted Cascade of Simple Features. IEEE COMPUT SOC CONF COMPUT VISION PATTERN RECOGNIT, 2001.
36. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. Lienhart et al. 2003.
37. A Detector of Tree of Boosted Classifiers for Real-Time Object Detection and Tracking. Rainer Lienhart et al. Microcomputer Research Labs, Intel Corporation, Santa Clara. 2003.
38. Robust Real-Time Face Detection. Paul Viola, Michael J. Jones. International Journal of Computer Vision. 2004.
39. J. Rodríguez Carlos J. Alonso, Q. Isaac Moro. Clasificación de Patrones Temporales en Sistemas Dinámicos mediante Boosting y Alineamiento Dinámico Temporal. Juan, Lenguajes y Sistemas Informáticos Grupo de Sistemas Inteligentes, Depto. Informática Universidad de Burgos Universidad de Valladolid.
40. Introducción a la Minería de Datos. Colectivo de Autores. Editorial Pearson. 2004.
41. Gary Cook y Tony Robinson, "Boosting the performance of connectionist large vocabulary speech recognition". ICSLP 1996.
42. Geoffrey Zweig y Mukund Padmanabhan, "Boosting Gaussian Mixture in an LVCSR System". IEEE, 2000.
43. Rong Zhang y Alexander I. Rudnicky, "Improving the performance of an LVCSR system through ensembles of Acoustic Models". IEEE 2003.
44. Christos Dimitrakakis y Samy Bengio, "Boosting HMMs with an application to speech recognition". International Conference on Acoustic, Speech. 2004.
45. Oh-Wook Kwon and Te-Won Lee. Optimizing Speech/Non-Speech Classifier design using Adaboost. IEEE, 2003.
46. Benyassine, E. Shlomot, and H.-Y. Su, "ITU-T Recommendation G.729 Annex B: A silent compression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data application," IEEE Comunication Magazine, pp.64-73, 1997.

47. Pei Yin, Irfan Essa, Thad Starner y James M. Rehg, "Discriminative Feature Selection for Hidden Markov Models using segmental Boosting". ICASSP, 2008.
48. T. Takiguchi, N. Miyake, H. Matsuda y Y. Ariki, "Voice and Noise Detection with Adaboosting". Book Robust Speech Recognition and Understanding, epig. 4, 77-84. 2007.
49. E. Alpaydin. "Introduction to Machine Learning," The MIT Press, ISBN-10: 0262012111. 2004.
50. Say Wei FOO, Eng Guan LIM, "Speaker Recognition using adaptively Boosted Decision Tree Classifier". ICASSP, 2002.
51. Stan Z. Li, Dong Zhang, Chengyuan Ma, Heung-Yeung Shum y Eric Chang, "Learning to Boost GMM Based Speaker Verification". 8^{th}European Conference of …, 2003.
52. Ke Chen, "Boosting Imput/Output Hidden Markov Models for Sequence Classification". Advances in Natural Computation - Springer, 2005.
53. Taichi Asami, Koji Iwano y Sadaoki Furui, "Stream-Weight Optimization by LDA and Adaboost for Multi-Stream Speaker Verification". Interspeech, 2005.
54. K. Iwano, T. Seki and S. Furui, "Noise robust speech recognition using F0 contour information," IEICE Trans.on Information and Systems, vol.E87-D, no.5, pp.1102-1109, 2004.
55. Taichi Asami, Koji Iwano y Sadaoki Furui, "A Stream-Weight and Threshold estimation method using Adaboosting for multi-stream speaker verification". Interspeech, 2006.
56. Gabriel Hernandez, Dr. José Ramón Calvo, Rafael Fernandez Ivis Rodes y Rafael Martínez, "Noise Robust Voice Detector for Speaker Recognition". ICPR, 2006.
57. F. Silva Mata, E. Garea, E. M. Alvarez and J. L. Gil. "A Fast Adaboosting based Method for Iris and Pupil Contour Detection". *LNCS 4225*, ISBN 978-3-540-46556-0, 2006.
58. Amarnag Subramanyal,Zhengyou Zhang, Arun C. Surendran, Patrick Nguyen, Mukund Narasimhan y Alex Acero, "A Generative-Discriminative framework using Ensemble Methods for text-dependent Speaker Verification". ICASSP, 2007.
59. Hao Tang y Thomas S. Huang, "Boosting Gaussian Mixture Models via Discriminate Analysis". ICPR, 2008.
60. Anindya Roy, Mathew Magimai.-Doss y Sebastien Marcel, "Boosted Binary Features for Noise-Robust Speaker Verification". ICPR, 2010.
61. V. Hautamaki, T. Kinnunen, I. Karkkainen, J. Saastamoinen, M. Tuononen, and P. Franti, "Maximum a Posteriori adaptation of the centroid model for speaker verification," IEEE Signal Processing Letters, vol. 15, pp. 162-165, 2008.
62. Haiyang Wu, Yong Lü, and Zhenyang Wu, "Improved AdaBoost Algorithm Using VQMAP for Speaker Identification", 2010 International Conference on Electrical and Control Engineering, pp. 1176 – 1179, 2010.
63. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1

RT_041, febrero 2011
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: Di. Alejandro Pérez Abraham
RNPS No. 2142
ISSN 2072-6287
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
La Habana. Cuba. C.P. 12200
Impreso en Cuba