Métodos de agregación de modelos y aplicaciones

Model aggregation methods and applications

Mathias Bourel[1]

Recibido: Mayo 2012 Aprobado: Agosto 2012

Resumen.- Los métodos de agregación de modelos en aprendizaje automático combinan varias hipótesis hechas sobre un mismo conjunto de datos con el fin de obtener un modelo predictivo con una mejor performance. Los mismos han sido ampliamente estudiados y han dado lugar a numerosos trabajos tanto experimentales como teóricos en diversos contextos: clasificación, regresión, aprendizaje no supervisado, etc. El objetivo de este trabajo es en un primer momento repasar varios métodos conocidos de agregación de modelos y luego realizar dos aplicaciones para comparar sus performances. La primera consiste en estudiar sus predicciones sobre distintas bases de datos para la clasificación, en particular en problemas de varias clases, y la segunda en utilizarlos en el contexto de la estimación de la densidad de una variable aleatoria.

Palabras clave: Agregación de modelos; Boosting; Bagging; Random Forest; Stacking.

Summary.- *Aggregation methods in machine learning models combine several assumptions made on the same dataset in order to obtain a predictive model with higher accuracy. They have been extensively studied and have led to numerous experimental and theoretical works in various contexts: classification, regression, unsupervised learning, etc. The aim of this work is in a first time reviewing several known models of aggregation and then compares their performances over two applications. The first is for making predictions on different databases, particularly in multiclass problems, and the second to use them in the context of estimating the density of a random variable.*

Keywords: *Model Aggregation; Boosting; Bagging; Random Forest; Stacking.*

1. Introducción.- El principio del Aprendizaje Automático o Machine Learning consiste, en el contexto supervisado, a partir de una muestra de aprendizaje $\mathcal{L} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ constituida por n realizaciones de un par de variables aleatorias (X,Y), construir una función $f: \mathcal{X} \longrightarrow \mathcal{Y}$ con la cual, dada una nuevo vector de entrada X, se pueda predecir con cierto grado de certeza la variable $Y = f(X)$. Para cada observación (X_i, Y_i) de $\mathcal{L}$, a la variable $X_i \in \mathcal{X}$ se le llama variable de entrada o explicativa o input (puede ser un vector) y a $Y_i \in \mathcal{Y}$ variable dependiente u output. Cuando la variable dependiente es discreta se habla de un problema de clasificación y cuando es continua de un problema de regresión. Por ejemplo si disponemos de datos ambientales un día d, un posible vector de entrada podría ser X = (temperatura, viento, presión atmosférica, humedad) y la salida Y el nivel de contaminación al día siguiente. Si relevamos durante varios días en un mismo periodo estas características se dispondrá de una base de datos cuyo objetivo será, en similar periodo, dado un nuevo vector X con los atributos (temperatura, viento, presión atmosférica, humedad), predecir el nivel de contaminación Y al día siguiente.

Matemáticamente, para lograr este cometido, se supone que f pertenece a cierta clase de funciones $\mathcal{H}$ y se busca f^* que minimiza el riesgo R_L sobre una función de pérdida L, es decir:

[1] Magister en Matemática, Universidad de la República, Docente Área Matemática Universidad de Montevideo, mbourel@um.edu.uy

ISSN 1510-7450 • ISSN (en línea) 1688-9584

$$f^* = Arg\min_{f\in\mathcal{H}} R_L(f) = Arg\min_{f\in\mathcal{H}} E_{(X,Y)}\left(L(f,X,Y)\right)$$

En un problema de clasificación, si $Y \in \{1,\dots,K\}$, la función de perdida es $L(f,X,Y) = 1_{\{Y\neq f(X)\}}$, el riesgo es $R_L(f) = P\big(Y \neq f(X)\big)$ y $f^*(x) = \text{Arg}\max_{k\in\{1,\dots,K\}} P(Y = k|X = x)$. Es decir, f^* predice la clase k que hace máxima la probabilidad a posteriori de Y conociendo X. Este clasificador se conoce como clasificador de Bayes.

En un problema de regresión se suele usar como función de perdida a $L(f,X,Y) = (Y - f(X))^2$, el riesgo es $R_L(f) = E_{(X,Y)}((Y - f(X))^2)$ y $f^*(x) = E(Y|X = x)$.

En la práctica, al desconocer la distribución conjunta de (X,Y) (¡y también las marginales!), se busca encontrar una función f_n^*, basada en la distribución empírica, que minimiza el riesgo empírico $R_{n,L}(f)$ sobre la muestra $\mathcal{L}$, es decir:

$$f_n^* = Arg\min_{f\in\mathcal{H}} R_{n,L}(f) = Arg\min_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} L(f(X_i),X_i,Y_i)$$

Es posible adaptar el razonamiento anterior al contexto no supervisado, es decir cuando la muestra $\mathcal{L}$ está formada únicamente de variables de entradas y se desconocen los outputs. Por ejemplo, es el caso de muestras provenientes de la realización de una variable aleatoria a la cual por ejemplo se quiere luego estimar la densidad, o de muestras que contienen distintas características de individuos y se los quiere agrupar por grupos de afinidades (clustering).

El enfoque anterior está a la base de la teoría del aprendizaje automático que se puede encontrar desarrollada por ejemplo en los libros [1] y [2] así como de las numerosas aplicaciones que han ido apareciendo.

Un método de aplicación muy utilizado en aprendizaje automático es Classification And Regression Trees (CART), algoritmo desarrollado por Breiman en 1984 en [3] que permite obtener, como su nombre lo indica, arboles de clasificación o de regresión mediante un método jerárquico divisivo. Para construir un árbol mediante este método, se realizan una serie de divisiones binarias de los datos en subconjuntos los más homogéneos posibles (a partir de algún criterio), según diversas reglas de decisiones, hasta llegar a un árbol maximal donde se reparten todas las observaciones y que contiene en cada hoja (nodo terminal) una muy poca cantidad de datos (típicamente 5). Como las reglas de división son del tipo "si x>1 entonces… y si $x\leq 1$ entonces…", un árbol de decisión divide recursivamente el espacio de observaciones con planos perpendiculares a los ejes coordenadas. A los efectos de evitar el sobreajuste sobre los datos con los cuales se entrena el modelo, se utiliza un algoritmo aglomerativo (pruning) que reúne varias hojas y ramas del árbol, de alguna manera inadecuadas, obteniéndose un árbol que tenga mejor poder de predicción. En cada hoja, se elige la clase que está más representada mediante un voto mayoritario en el caso de la clasificación o se hace el promedio de la variable dependiente en el caso de la regresión. Una de las mayores ventajas de CART es su fácil interpretabilidad y, por lo tanto, es muy fácil dada una nueva observación, predecir la salida que da el algoritmo. Sin embargo este algoritmo es muy inestable: una pequeña variación en el conjunto de los datos conlleva varias veces a un árbol totalmente distinto. Sugerimos al lector interesado en profundizar sobre esta técnica recurrir a [3] o a [1].

Una manera de remediar el problema anterior es hacer una agregación de modelos. Los métodos de agregación de modelos consisten en, a partir de un mismo conjunto de datos $\mathcal{L}$, construir varios predictores que llamaremos hipótesis intermediarias y combinarlos de alguna forma para poder obtener un predictor más estable, con mayor performance y disminuyendo la varianza. Además de tomar en cuenta qué tipo de modelos intermediarios se consideran, existen varias maneras de combinarlos.

ISSN 1510-7450 • ISSN (en línea) 1688-9584

Este trabajo se desarrolla de la manera siguiente. En la sección 2 exponemos generalidades sobre los métodos de agregación de modelos y algunos enfoques. La sección 3 explicita cuatro algoritmos de agregación de modelos conocidos: Bagging, Random Forest, Boosting y Stacking. Finalmente en la sección 4 presentamos dos aplicaciones en las cuales juegan un papel importante este tipo de métodos: la clasificación de datos y la estimación de densidad.

2. Métodos de agregación.- En un problema de clasificación de varias clases, si disponemos de un conjunto de datos $\mathcal{L}$, a los efectos de obtener modelos de predicción más robusto, podemos construir M clasificadores $g_1, g_2, \dots, g_M$ para predecir la variable dependiente Y y combinarlos. Se puede definir un "clasificador agregado" de manera natural, como el voto por mayoría de los M clasificadores intermediarios, es decir:

$$f(x) = Arg \max_{k\in\{1,\dots,K\}} (\#\{m: g_m(x) = k\}) = Arg \max_{k\in\{1,\dots,K\}} \left(\sum_{m=1}^{M} 1_{\{g_m(x)=k\}} \right)$$

Otra manera de combinar estos clasificadores intermediarios puede ser mediante un voto ponderado. Si $\alpha_1, \dots, \alpha_M$ son números reales, podemos definir el "clasificador agregado" como el que predice la clase resultante de la mayor suma ponderada de los clasificadores $g_1, g_2, \dots, g_M$ que la predicen:

$$f(x) = Arg \max_{k\in\{1,\dots,K\}} \left(\sum_{m=1}^{M} \alpha_m 1_{\{g_m(x)=k\}} \right)$$

Análogamente para el caso de la regresión, un predictor agregado podría definirse como el promedio de las predicciones hechas por $g_1, g_2, \dots, g_M$ sobre la muestra $\mathcal{L}$ donde también se puede considerar una ponderación. En este caso el predictor agregado sería por lo tanto una combinación lineal de la forma

$$f(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$$

donde $\alpha_1, \dots, \alpha_M$ son coeficientes reales.

Los métodos descriptos anteriormente son ejemplos de métodos de agregación de modelos y buscan a partir de un conjunto de hipótesis de base $g_1, g_2, \dots, g_M$ combinarlos a los efectos de obtener un nuevo predictor más estable, que disminuya el error cometido.

Estos métodos han generado numerosos trabajos científicos donde se abordan las propiedades teóricas y las performances de los mismos, tomando en cuenta dos factores importantes: la manera de obtener el conjunto de las hipótesis intermediarias y la manera de combinarlas. Se pueden dividir estos métodos en dos grandes familias: los métodos de agregación de modelos homogéneos y los métodos de agregación de modelos heterogéneos. La primera familia esta formada por aquellos métodos que combinan hipótesis de la misma naturaleza: las distintas hipótesis provienen de un mismo tipo de algoritmo de aprendizaje. Por ejemplo, se combinan varios arboles de clasificación o varias redes neuronales. En esta categoría los métodos más conocidos son Bagging, Random Forest y Boosting. Aquellos métodos de agregación que combinan hipótesis proveniente de distintos algoritmos de aprendizaje, por ejemplo combinar tres arboles de clasificación, una máquina de soporte vectorial y una red neuronal, constituyen la familia de los métodos de agregación heterogéneos. El procedimiento Stacking es un ejemplo de métodos de agregación de modelos heterogéneos.

2.1. Bagging.- La técnica Bagging, por Booststrap aggregating, fue introducido por Breiman en 1996 en [4] y es un método de agregación de modelos homogéneos que se basa sobre el voto

ISSN 1510-7450 • ISSN (en línea) 1688-9584

mayoritario o el promedio según el caso. El método consiste en hacer varias remuestras del conjunto de datos iniciales y promediar las predicciones hechas por los distintos clasificadores. Más precisamente, a partir de un conjunto de datos $\mathcal{L} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ se realiza, mediante el procedimiento bootstrap [5], un tiraje con reposición de n observaciones de $\mathcal{L}$ obteniéndose una nueva muestra $\mathcal{L}^*$. Cada elemento de $\mathcal{L}$ tiene una probabilidad de $1 - \left(1 - \frac{1}{n}\right)^n \approx 0{,}63$ (si n es grande) de salir sorteado y por lo tanto de aparecer en $\mathcal{L}^*$. A esta nueva muestra se le aplica el algoritmo deseado. Se repite este procedimiento M veces obteniendo de esta manera M modelos $g_1, \dots, g_M$ y el predictor final es el voto mayoritario de los M estimadores en clasificación o el promedio en el caso de la regresión (Figura I), es decir:

$$f_{bag}(x) = Arg\max_y\left(\#\{m: g_m(x) = y\}\right) \quad \text{(clasificación)} \quad o \quad f_{bag}(x) = \frac{1}{M}\sum_{m=1}^{M} g_m(x) \quad \text{(regresión)}$$

Este método reduce el error al combinar varios clasificadores con alta varianza, como por ejemplo, los árboles de clasificación, y se han mostrado en varios trabajos las mejoras obtenidas en las performances en problemas de clasificación y de regresión. El fundamento teórico se basa en que el error promedio que se obtiene sobre la muestra de entrenamiento $\mathcal{L}$ es mayor o igual al error obtenido por el estimador Bagging. En efecto se puede probar (ver [4]) que si f es un predictor obtenido sobre $\mathcal{L}$ y f_{bag} el obtenido por Bagging entonces:

$$E_{\mathcal{L}}(E_{(X,Y)}(Y - f(X))^2) \geq E_{(X,Y)}(Y - f_{bag}(X))^2$$

A continuación se resume el algoritmo Bagging:

Sea $\mathcal{L} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ con $X_i \in \mathcal{X}$, $Y_i \in \{1, \dots, K\}$ o $Y_i \in \mathbb{R}$

1) Se construyen M muestras bootstrap $\mathcal{L}_1^*, \mathcal{L}_2^*, \dots, \mathcal{L}_M^*$ de tamaño n.
2) Para cada una de ellas se calculan estimadores $g_1^*, g_2^*, \dots, g_M^*$.
3) El estimador Bagging es:

$$f_{bag}(x) = Arg\max_y(\#\{m: g_m^*(x) = y\}) \quad \text{(clasificación)} \quad o \quad f_{bag}(x) = \frac{1}{M}\sum_{m=1}^{M} g_m^*(x) \quad \text{(regresión)}$$

Este algoritmo es muy sencillo de utilizar y es fácil adaptarlo a cualquier método de aprendizaje con que se trabaja.

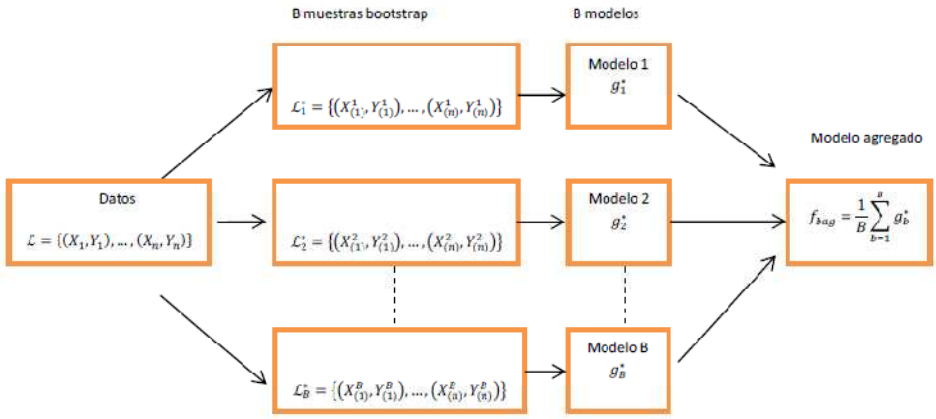

Figura I.- El método Bagging en regresión.

2.2. Random Forest.- Breiman en 2001 propone el algoritmo Random Forest (arboles aleatorios) en [6] que combina las técnicas de CART y Bagging. El mismo tiene como propósito

ISSN 1510-7450 • ISSN (en línea) 1688-9584

incorporar la aleatoriedad en las distintas etapas de la construcción de un árbol obtenido por CART. Al igual que en Bagging, se sortean M muestras bootstrap del conjunto de datos originales $\mathcal{L}$. Sobre las mismas se construyen M arboles para los cuales, en cada nodo, se elige la mejor subdivisión hecha por un subconjunto de variables explicativas seleccionadas aleatoriamente. Los arboles que se obtienen son maximales, es decir que en este caso no se podan. En el caso de un problema de clasificación, la predicción de una observación hecha por Random Forest es la clase más votada entre las predicciones hechas por los M arboles y en regresión se hace un promedio de los valores asignados. Se ha mostrado que Random Forests es uno de los algoritmos con mejores performances en los problemas de aprendizaje, en particular en aquellos que cuentan con una cantidad importante de variables explicativas. Sin embargo si sólo algunas pocas son de relevancia, el Boosting suele ser más eficaz que Random Forest. Finalmente mencionamos que se cuenta con varios criterios que permiten obtener la importancia de cada una de las variables explicativas en el modelo agregado por Random Forest.

2.3. Boosting.- Boosting busca combinar a través de varias iteraciones de un mismo algoritmo de base las predicciones hechas de manera adaptativa. Más precisamente la performance del predictor de la etapa m influye sobre la manera en que se considera la muestra en la etapa m+1. Realizando estas variaciones adaptativas de la muestra y asignando un peso a cada predictor intermediario se obtiene un clasificador final más eficiente, que disminuye la varianza y también el sesgo. El algoritmo Adaboost (Figura II) desarrollado por Freund et Schapire en [7] es sin duda el algoritmo del tipo Boosting más conocido y estudiado. Este utiliza por lo general los arboles de decisión como modelos de base, por lo cual forma parte de los métodos de agregación de modelos homogéneos, y fue pensado originalmente para la clasificación binaria. A la iteración m del algoritmo, siguiendo cierta distribución, se construye un árbol y se observa las previsiones de éste sobre todos los datos de la muestra. A aquellos datos mal clasificados, se les asignan un peso mayor, influyendo de esta manera sobre la distribución de los datos de la etapa m+1 y forzando al predictor correspondiente en tomarlas "más en cuenta". El predictor que se obtiene al final resulta de un voto mayoritario ponderado o de un promedio ponderado de los predictores de las distintas etapas. Al focalizarse sobre los ejemplos mal clasificados, el error empírico disminuye rápidamente. Sin embargo, en vez de contribuir a un sobreaprendizaje sobre los datos utilizados, se prueba que el error de generalización disminuye también, lo cual hace de Adaboost un algoritmo con un muy buen rendimiento.

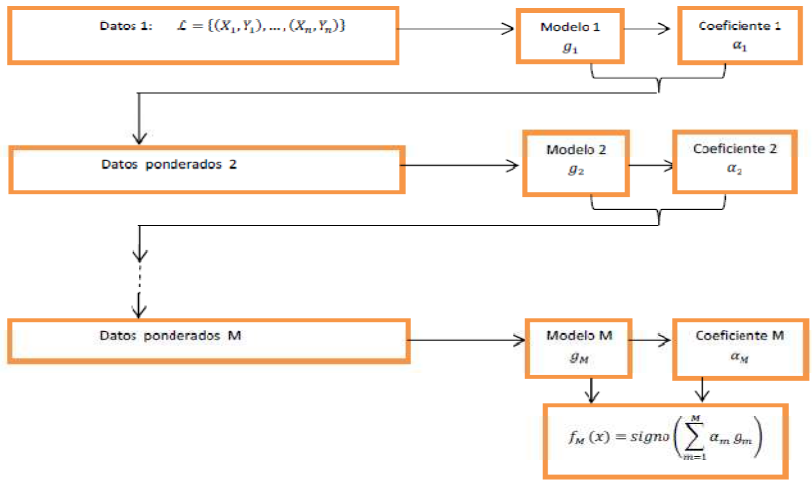

Figura I.- El método Adaboost para la clasificación binaria

A continuación damos una descripción explicita de Adaboost para la clasificación binaria:

1) Si $\mathcal{L} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ con $X_i \in \mathcal{X}$, $Y_i \in \{-1,1\}$, se inicializan los pesos de las observaciones con $w_1(i) = \frac{1}{n}$ para todo $i = 1, \dots, n$.
2) Para $m = 1, \dots, M$:
   a. Se construye a partir de $\mathcal{L}$ y de los pesos $w_m(i)$ un clasificador $g_m: \mathcal{X} \to \{-1,1\}$ que minimiza el error global $\epsilon_m = \sum_{i=1}^{n} w_m(i)\, 1_{\{g_m(x_i) \neq y_i\}}$.
   b. Calculamos $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
   c. Actualizamos los pesos: $w_{m+1}(i) = w_m(i) e^{\alpha_m 1_{\{g_m(x_i) \neq y_i\}}}$ para $i = 1, \dots, n$ y los normalizamos.
3) El clasificador final es:

$$f(x) = Argmax_y \sum_{m=1}^{M} \alpha_m 1_{\{g_m(x)=y\}} = signo\left(\sum_{m=1}^{M} \alpha_m g_m(x)\right)$$

Observar que a cada etapa m del algoritmo, se le asigna un peso mayor a los datos mal clasificados por g_m (estos son los ejemplos para los cuales $y_i g_m(x_i) = -1$). Si el error ϵ_m, la suma de los pesos de las observaciones mal clasificadas por el clasificador de la etapa m es mayor que $\frac{1}{2}$ (esto significa que más de la mitad de las observaciones están mal clasificadas) se detiene el algoritmo. Esta condición implica que el clasificador debe predecir mejor que una clasificación aleatoria.

En [8], se puede encontrar de manera muy clara una prueba de que el clasificador f construido por Adaboost minimiza el riesgo de la función de pérdida exponencial $L(y, f) = e^{-yf(x)}$ y que coincide con la regla de clasificación de Bayes $F(x) = \underset{y}{\text{Argmax}}\, P(Y = y|X = x)$, minimizando por lo tanto el error de clasificación.

En el caso de la regresión, donde la variable dependiente es continua, se puede adaptar el clasificador anterior definiendo f(x) como:

$$f(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$$

que resulta ser un promedio ponderado de los distintos resultados obtenidos por $g_1, g_2, \dots, g_M$.

Una variante de Adaboost, Arcing, introducido por Breiman en [9], consiste en utilizar en cada etapa los pesos de las observaciones para extraer una muestra bootstrap con la cual se construye el clasificador intermediario.

Adaboost ha sido muy estudiado y sus propiedades teóricas han inspirado muchas generalizaciones, enfoques y extensiones al caso en que la variable a predecir pertenezca a un conjunto con más de dos elementos o sea continua. En el caso de problemas de varias clases (o multiclases), se conocen varias extensiones de Adaboost. La más inmediata es Adaboost.M1 que es el mismo algoritmo que Adaboost pero considerando que la variable dependiente pueda pertenecer a más de dos clases, es decir $y_i \in \{1, \dots, K\}$ (si K=2, Adaboost.M1 coincide con Adaboost). Otros métodos reducen y transforman el problema en aplicar reiteradas veces Adaboost a múltiples problemas binarios (por ejemplo Adaboost.M2, Adaboost.MH) que no conducen necesariamente a predictores optimales y conllevan naturalmente a aumentar la complejidad. Los argumentos teóricos justificando las implementaciones de las adaptaciones de

ISSN 1510-7450 • ISSN (en línea) 1688-9584

Adaboost a estos contextos no están tan desarrollados como para el caso binario y son objetos de estudio de varios trabajos recientes, donde se busca obtener directamente un clasificador multiclases (ver [10],[11],[12]). En [10] los autores proponen el método SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function). En este caso, el algoritmo es el mismo que Adaboost.M1 con la diferencia que en el paso 2.b) se le suma a α_m el termino $\log(K-1)$ (observe que si el problema es binario SAMME coincide con Adaboost). El ajuste anterior se debe al tipo de función de perdida que se busca minimizar, en este caso la función de perdida multiclases exponencial $L(y,f) = e^{-\frac{1}{K}y^t f}$, que extiende de manera natural la función de perdida exponencial que se utiliza para el caso binario.

2.4. Stacking.- Este algoritmo pertenece al grupo de los métodos de agregación heterogéneos, y por lo tanto tiene la particularidad de poder combinar varios estimadores provenientes de distintos métodos de aprendizaje. Fue introducido por Wolpert en 1992 en [13]. Se elige al principio una cantidad M de algoritmos que se quiere combinar. En una primera etapa, llamada nivel-0, se divide aleatoriamente la muestra de entrenamiento $\mathcal{L} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\}$ en J subconjuntos $L_1, L_2, \dots, L_J$. Luego, en un procedimiento de validación cruzada, se consideran para cada $j = 1, \dots, J$ los conjuntos $L^{(-j)} = L \backslash L_j$ con los cuales se entrenan los M algoritmos de aprendizaje y denotamos por $g_m^{(-j)}$ el modelo obtenido por el algoritmo m sobre el conjunto de entrenamiento $L^{(-j)}$. Estos modelos obtenidos se llaman modelos de nivel-0. A partir de ellos, y para cada observación X_i de L_j, denotamos por z_{mi} la predicción hecha por el modelo $g_m^{(-j)}$, es decir $z_{mi} = g_m^{(-j)}(X_i)$. De esta manera, obtenemos para cada observación i de L_j un nuevo vector de predicciones $(Y_i, z_{1i}, z_{2i}, \dots, z_{Mi})$. Se dispone entonces de un nuevo conjunto de entrenamiento $\mathcal{L}'$ de n vectores con M+1 coordenadas que llamamos nivel-1 de datos. Éste último permite entrenar g, el modelo de nivel-1. Finalmente a partir de toda la muestra de entrenamiento $\mathcal{L}$ se obtienen los M modelos $g_1, \dots, g_M$ proveniente de los M algoritmos de aprendizaje considerados en el nivel-0. Para clasificar un nuevo dato, se utilizan los modelos $g_1, \dots, g_M$ y g. La predicción final sobre un nuevo dato x es:

$$f(x) = g(x, g_1(x), \dots, g_M(x)).$$

Una variante del procedimiento de validación cruzada utilizado para formar los conjuntos de entrenamiento de nivel-0 es aplicar la técnica del "leave-one-out" a la muestra $\mathcal{L}$: para cada i, se considera el conjunto $\mathcal{L}^{(-i)} = \mathcal{L} \backslash \{(X_i, Y_i)\}$ y se consideran $g_1^{(-i)}, g_2^{(-i)}, \dots, g_M^{(-i)}$ los M modelos del nivel-0 entrenados sobre $\mathcal{L}^{(-i)}$ y el vector de predicciones $Z_i = (g_1^{(-i)}(X_i), g_2^{(-i)}(X_i), \dots, g_M^{(-i)}(X_i))$. Entonces el conjunto de entrenamiento para el nivel-1 es $\mathcal{L}' = \{(Y_1, Z_1), (Y_2, Z_2) \dots, (Y_n, Z_n)\}$. Esta idea es la que utiliza Breiman en [14] para aplicar el método Stacking al caso de la regresión y donde el predictor final del nivel-1es:

$$f(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$$

siendo $\alpha_1, \dots, \alpha_M$ coeficientes que se obtienen mediante una regresión ridge de Z sobre Y y $g_1, \dots, g_M$ son los M modelos obtenidos a partir del conjunto$\mathcal{L}$.

Como dijimos anteriormente la cantidad M de modelos que se utilizan es fija antes de comenzar y es una componente esencial en el diseño del algoritmo. Ésta es una diferencia significativa con el Bagging o el Boosting, donde M influye únicamente en la cantidad final de sumandos, no en como se conforma la combinación lineal. En [15], los autores hacen un estudio empírico muy interesante sobre problemas de clasificación donde muestran que este método obtiene mejores resultados si en vez de utilizar la asignación realizada por cada hipótesis se consideran las

probabilidades que cada hipótesis asigna a cada clase. Por otro lado los mejores resultados se obtienen cuando se usan modelos de regresión lineal en el nivel-1 y que los resultados obtenidos por esta técnica son mejores que las obtenidas por voto mayoritario.

En [16], Smyth et Wolpert adaptan la teoría del Stacking a un contexto de aprendizaje no supervisado: la estimación de la densidad de una variable aleatoria. Para ello se combinan M estimadores de densidad de manera análoga al caso supervisado visto anteriormente. Recordamos que si se dispone de n realizaciones $\{X_1, X_2, \dots, X_n\}$ de una variable aleatoria X con densidad desconocida f, un estimador por núcleo de f en $x \in \mathbb{R}$ es:

$$g(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

donde K es una función núcleo (K es una función no negativa, simétrica y tal que $\int K(t)dt = 1$) y h el parámetro de ancho de ventana. Se empieza realizando una partición mediante un procedimiento de validación cruzada del conjunto de datos $\mathcal{L} = \{X_1, X_2, \dots, X_n\}$ en J subconjuntos $L_1, L_2, \dots, L_J$ y se consideran para cada $j = 1, \dots, J$ los conjuntos $L^{(-j)} = L \backslash L_j$. Se ajustan M estimadores de densidad por núcleo $g_1, \dots, g_M$ sobre cada una de las muestras $L^{(-1)}, \dots, L^{(-J)}$ obteniéndose de esta manera para cada $m \in \{1, \dots, M\}$ los estimadores $g_m^{(-1)}, \dots, g_m^{(-J)}$. Éstos se evalúan sobre $L_1, L_2, \dots, L_J$ y se obtiene la matriz de tamaño $n \times M$:

$$A = \begin{pmatrix} g_1^{(-1)}(L_1) & \dots & g_m^{(-1)}(L_1) \dots & g_M^{(-1)}(L_1) \\ \vdots & & \vdots & \vdots \\ g_1^{(-J)}(L_J) & \dots & g_m^{(-J)}(L_J) \dots & g_M^{(-J)}(L_J) \end{pmatrix} = \begin{pmatrix} | & | & \dots & | \\ g_1^*(\mathcal{L}) & g_2^*(\mathcal{L}) & & g_M^*(\mathcal{L}) \\ | & | & \dots & | \end{pmatrix}.$$

La matriz A se utiliza para obtener, mediante el algoritmo Expectation-Maximization - EM, los coeficientes $\alpha_1, \dots, \alpha_M$ que maximizan la log verosimilitud del modelo $f_s(x) = \sum_{m=1}^{M} \alpha_m g_m^*(x)$. Por último se re-estiman $g_1, \dots, g_M$ utilizando todo el conjunto de datos $\mathcal{L}$ y el "estimador Stacking" es la combinación lineal de estos modelos con los coeficientes $\alpha_1, \dots, \alpha_M$ encontrados anteriormente, es decir:

$$f_s(x) = \sum_{m=1}^{M} \alpha_m g_m(x)$$

Esta técnica de estimación de la densidad nos servirá en las dos aplicaciones que desarrollamos a continuación.

3. Aplicaciones.- En esta sección presentaremos dos aplicaciones: la primera evaluar las performances de los algoritmos de agregación sobre base de datos conocidas comparando con los resultados de la propia base de datos (aprendizaje supervisado) y la segunda utilizar las técnicas de agregación de modelos al problema de de la estimación de la densidad de una variable aleatoria (aprendizaje no supervisado).

3.1. Problema de clasificación.- Con el fin de comparar las performances de los métodos de agregación vistos anteriormente, elegimos cuatro bases de datos disponibles en el UCI Machine Learning Repository (http://mlearn.ics.uci.edu/). No nos limitaremos únicamente a problemas binarios sino que también evaluaremos sus comportamientos sobre conjuntos cuya variable

ISSN 1510-7450 • ISSN (en línea) 1688-9584

dependiente pueda pertenecer a más de dos clases. También elegimos estas bases de datos teniendo en cuenta la cantidad de variables explicativas involucradas.

Las bases de datos utilizadas (Tabla I) son:

- Breast (2 clases). Esta base de datos contiene los datos de cáncer de mamá de 699 individuos con 9 variables observadas vinculadas a esta patología y se clasifica el tipo de cáncer en dos tipos: benigno o maligno.
- Iris (3 clases). Esta base de datos contiene los datos de 150 flores de iris a las cuales se les midió 4 variables el ancho y el largo del sépalo, el ancho y el largo del pétalo y se las clasifica en tres especies: setosa, virginica o versicolor.
- Wine (3 clases). Esta base de datos contiene los datos de 178 vinos provenientes de la misma región de Italia a los cuales se les mide 13 componentes etílicos y se los clasifica en tres tipos.
- Glass (6 clases). Esta base de datos contiene los datos de 214 cristales a los cuales se les midió 10 variables y se los clasifica en 6 clases.

Cantidad:	Breast	Iris	Wine	Glass
Clases	2	3	3	6
Observaciones	699	150	178	214
Variables explicativas	9	4	13	10

Tabla I.- Descripción de los datos utilizados

Los algoritmos comparados son Adaboost.M1, SAMME, Bagging, Random Forest y Stacking para la estimación de densidad aplicado a los problemas de clasificación. En efecto, la formula de Bayes nos da un procedimiento para lograr este propósito. Supongamos que queremos clasificar un individuo *x* proveniente de una variable aleatoria *X* en un problema de *K* clases. Si conocemos f_k la función de densidad de la clase k y $p_k = P(Y = k)$ la probabilidad a priori de la clase k entonces, utilizando la regla de decisión de Bayes para maximizar la probabilidad a posteriori $P(Y = k|X = x) = \frac{p_k f_k(x)}{\sum_{k=1}^{K} p_k f_k(x)}$, el clasificador resulta ser

$$h(x) = Arg \max_{k\in\{1,\dots,K\}} p_k f_k(x).$$

En la práctica se estima p_k por la proporción $\frac{n_k}{n}$ de la clase k dentro de la muestra (n_k es la cantidad de veces que se observa la clase k en la muestra y n la cantidad total de datos). La estimación de la densidad f_k se obtiene por algún método de estimación, en general por el método de estimación por núcleo. Si la variable $X \in \mathbb{R}^d$, una manera de estimar la densidad de X es considerar la estimación por núcleo multivariada:

$$\hat{f}(x,h) = \frac{1}{nh^d} \sum_{i=1}^{n} \left(\prod_{j=1}^{d} K\left(\frac{x_j - x_j^i}{h} \right) \right)$$

siendo $x^1, \dots, x^n \in \mathbb{R}^d$ las observaciones, K una función núcleo y $x = (x_1, x_2, \dots, x_j) \in \mathbb{R}^d$.

Cada conjunto de datos se divide aleatoriamente en una muestra base (2/3 de los datos) para el aprendizaje a los efectos de entrenar el modelo y una muestra test (1/3 de los datos) para evaluar la performance del modelo.

Se realiza una primera simulación para estudiar la evolución de la tasa de error en la muestra test en función de la cantidad de modelos que se agregan al estimador final. En la Figura III presentamos las graficas de esta evolución, promediando los resultados obtenidos tras haber corrido 5 veces los algoritmos Bagging (Bag), Random Forest (RF), Adaboost (Ada) y Samme (Sam) -cuando corresponde- en cada uno de los conjuntos de datos y utilizando como hipótesis de base arboles de clasificación obtenidos por CART. En las gráficas obtenidas podemos ver que Random Forest es el algoritmo que produce la menor tasa de error sobre todos los conjuntos de datos utilizados. Por otro lado observamos que Samme tiene buenas performances sobre los conjuntos de tres clases, mejorando los resultados obtenidos por Adaboost.M1, pero que su desempeño no es bueno en el caso de Glass (6 clases). Podemos ver también que, en general, la tasa de error va disminuyendo a medida que se agregan estimadores en las técnicas que usamos.

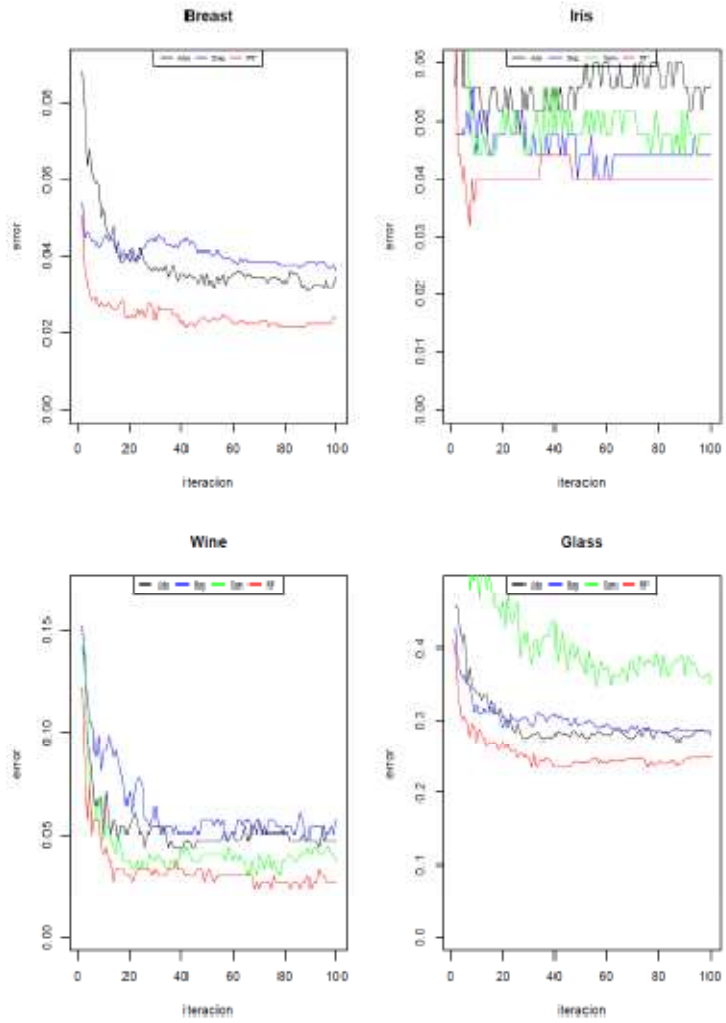

Figura III.- Evolución de la tasa de error en función de la cantidad de iteraciones sobre cuatro conjuntos de datos del UCI

Por otro lado, en la Tabla II, presentamos las tasas de error obtenidas en la muestra test sobre los mismos conjuntos de datos a los efectos de comparar los métodos de agregación Bagging, Random Forest, Adaboost.M1, SAMME (cuando corresponde) y Stacking aplicado a la

estimación de la densidad. Para mostrar como la agregación de modelos favorece buenos resultados incluimos en esta tabla los resultados obtenidos por un árbol de clasificación construido con CART. Para Bagging, Random Forest, Adaboost.M1, SAMME utilizamos 100 hipótesis intermediarias y promediamos los resultados luego de 20 corridas. Para Stacking, agregamos cuatro estimadores por núcleos gaussianos $K(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$, con anchos de ventana $h \in \{0.1, 0.2, 0.3, 0.4\}$ y la matriz *A* se obtiene a partir de una validación cruzada de *J=10*.

Observamos que Random Forest es el algoritmo que tiene mejores desempeños globales sobre estas bases de datos. Mismo en el caso de Iris, la tasa de error obtenida por Random Forest es muy parecida a la de Stacking. Sobre los conjuntos de datos con 3 clases (Iris y Wine), SAMME mejora los resultados obtenidos por Adaboost.M1 pero no es el caso para Glass. Finalmente parecería ser que Stacking aplicado a la estimación de la densidad no obtiene buenos resultados (por lo menos competitivos) en el caso que la cantidad de variables explicativas es importante.

	CART	Adaboost.M1	Bagging	R. Forest	SAMME	Stacking
Breast	0.075	0.042	0.046	**0.032**	--	0.088
Iris	0.059	0.053	0.058	0.047	0.051	**0.042**
Wine	0.119	0.036	0.077	**0.027**	0.035	0.235
Glass	0.401	0.272	0.266	**0.238**	0.33	0.41

Tabla II.- Comparación de la performance de CART, Adaboost.M1, Bagging, Random Forest, SAMME y Stacking

3.2. Estimación de la densidad.- En esta parte utilizaremos algunos algoritmos de agregación de modelos adaptándolos al contexto de la estimación de densidad. Existen relativamente pocos trabajos sobre este tema, y a la vez, son muy pocas las comparaciones sobre los desempeños entre estos tipos de algoritmos. Esta aplicación es parte de un trabajo de investigación realizado en conjunto con B. Ghattas en [17] donde comparamos las performances sobre distintos modelos, con distintos grados de dificultad, de varios estimadores de densidad obtenidos por agregación. Proponemos dos tipos de agregación a partir hipótesis intermediarias sencillas como los histogramas. Llamamos estos dos métodos AggHist y BagHist y pertenecen a la familia de métodos de agregación homogéneos.

En AggHist empezamos construyendo un histograma *h*. A cada iteración *m* del algoritmo se le suma un ruido gaussiano a los puntos de corte de los intervalos de *h* y se construye un nuevo histograma h_m basado en estos nuevos intervalos. El predictor final es el promedio de estos histogramas, es decir:

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} h_m(x).$$

En BagHist, a cada iteración *m* del algoritmo se sortea una muestra bootstrap de los datos y se construye un histograma h_m sobre la misma. El predictor final resulta del promedio de estos histogramas:

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} h_m(x).$$

Para cada uno de estos métodos realizamos varias simulaciones a los efectos de determinar por máxima verosimilitud la cantidad de puntos de corte optimal y la amplitud de la perturbación aleatoria que se efectúa en AggHist.

La primera simulación que presentamos resulta de comparar las curvas obtenidas por AggHist, BagHist, agregando para cada uno de estos métodos M=300 histogramas, a los efectos de ver como se ajustan a la verdadera densidad. En la Figura IV mostramos estas gráficas para el caso de la ley Gaussiana estándar y de la Claw Density. Sobre las gráficas obtenidas podemos ver que las curvas que se obtienen parecen ajustarse razonablemente bien a la curva de la verdadera densidad, en particular en los ajustes de las distintas modas.

Sobre estos mismos modelos, en la Figura V mostramos como va decreciendo el error cuadrático medio de las predicciones en función de la cantidad de histogramas que se agregan al correr 100 veces cada algoritmo y promediando.

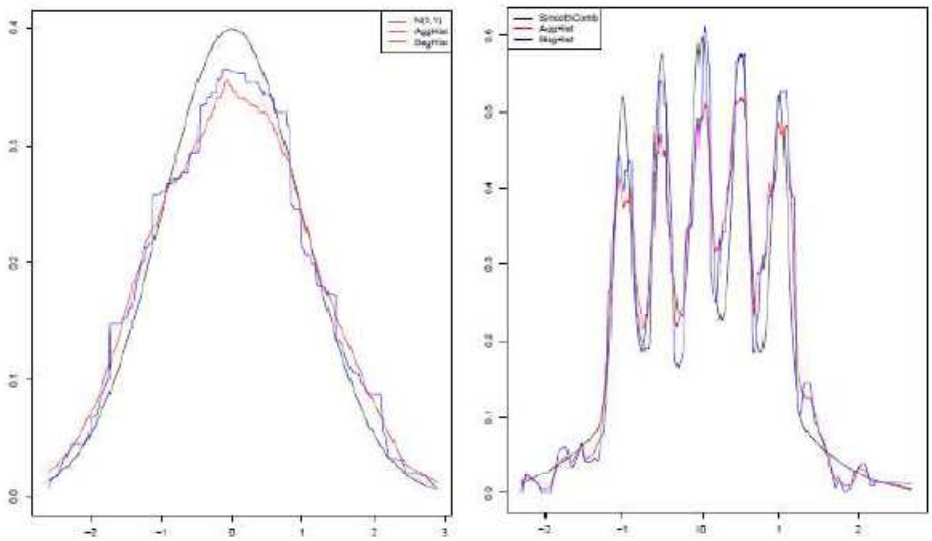

Figura IV.- Comparaciones de los gráficos de las densidades obtenidos por AggHist, BagHist con la densidad simulada: a la izquierda la ley Gaussiana $N(0,1)$ y a la derecha la Claw Density.

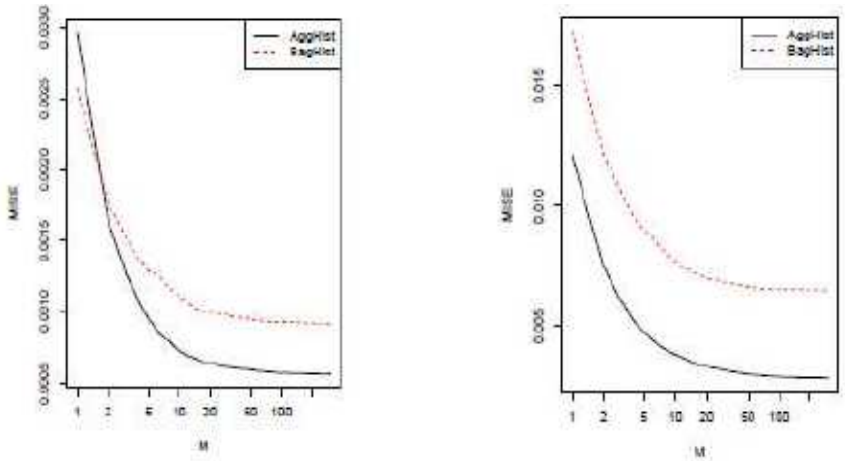

Figura V.- Curvas de evolución del MISE en función de la cantidad de modelos agregados: a la izquierda para la ley Gaussiana $N(0,1)$ y a la derecha para la Claw Density

Finalmente comparamos la performance de nuestros algoritmos con dos técnicas de agregación de estimadores de densidad conocidas, utilizando los mismos estimadores que en la literatura consultada:

- El *Stacking* de 3 núcleos gaussianos y 3 núcleos triangulares con ancho de ventana $h \in \{0.1, 0.2, 0.3\}$ y una validación cruzada de *J=10* como en [16].

ISSN 1510-7450 • ISSN (en línea) 1688-9584

- *AggPure*, algoritmo desarrollado por Rigollet y Tsybakov en [18]. Para este método, al igual que en Stacking, se elige al principio una cantidad fija *M* de estimadores de la densidad por núcleo con anchos de ventana distintos. En cada iteración del algoritmo se divide el conjunto de los datos en dos partes: la primera parte sirve para obtener los estimadores de densidad y con la segunda parte se optimizan los coeficientes que serán utilizados para realizar la combinación lineal, obteniéndose un estimador del tipo $f_s(x) = \sum_{m=1}^{M} \alpha_m f_m(x)$. Se reitera este proceso tantas veces se desea y se computa un estimador final f, que resulta ser el promedio de las estimaciones realizadas, es decir si S es conjunto de todas las divisiones hechas entonces

$$f_{AggPure}(x) = \frac{1}{\#S} \sum_{s \in S} f_s(x)$$

  En nuestro experimento realizamos 10 divisiones y utilizamos 4 estimadores por núcleos gaussianos con ancho de ventana $h \in \{0.001, 0{,}005, 0.01, 0.05, 0.1, 0.5\}$ como en [18].

A los efectos de mostrar como la agregación de estimadores de densidades tiene ventajas sobre otros métodos clásicos, incluimos también en nuestras comparaciones los errores cometidos por las predicciones hechas por un histograma (Hist) y las predicciones hechas por un estimador por núcleo Gaussiano simple (KDE) donde el ancho de ventana es optimizada por el método de Silvermann.

En el experimento comparamos y promediamos los errores cuadráticos medios (Mean Integrated Square Error, MISE) sobre un conjunto de 500 datos realizando 100 corridas de cada algoritmo.

Las funciones de densidad utilizadas son:

- (M1) la ley Gaussiana estándar.
- (M2) la ley exponencial
- (M3) una mezcla de dos Gaussianas
- (M4) una mezcla de tres Gaussianas.
- (M5) la Claw Density.

Estas simulaciones son partes de las que se realizan en [17].

	Hist	AggHist	BagHist	KDE	Stacking	AggPure
M1	0.171	0.056	0.085	0.085	**0.054**	0.076
M2	0.739	0.720	**0.701**	5.98	1.12	1.28
M3	0.061	**0.024**	0.040	0.094	**0.024**	0.036
M4	0.08	**0.034**	0.050	0.206	0.035	0.036
M5	0.859	**0.531**	0.547	2.12	0.538	0.768

Tabla III: Comparación de los errores obtenidos en los distintos modelos.
Los valores indicados en esta tabla corresponden a 100 veces el error MISE.

En la Tabla III presentamos los resultados obtenidos. Podemos ver que sobre los modelos considerados las técnicas de agregación de estimadores de densidad tienen mejores resultados que los estimadores clásicos. Nuestros algoritmos son más eficientes en 4 de estos 5 casos y AggHist obtiene un resultado muy cercano al de Stacking para el modelo M1. Observamos que Stacking tiene muy buenas performances en los modelos de mezclas de Gaussianas. Comprobamos al correr los algoritmos que los tiempos de respuestas de AggHist y de BagHist son menores que los de Stacking y de AggPure: en efecto sus complejidades son proporcionales a la de un histograma.

4. Conclusiones.- En este artículo repasamos los principales métodos de agregación de modelos conocidos en la literatura del aprendizaje automático y sus desempeños en aplicaciones provenientes de enfoques distintos: la clasificación y la estimación de densidad.

Estos métodos intentan mejorar bajo distintos puntos de vista los modelos predictivos obtenidos por uno o varios métodos determinados, con distintos grados de avances teóricos y experimentales. Si bien en el contexto del aprendizaje supervisado y en problemas binarios los métodos de agregación de modelos han dado muy buenos resultados y tienen un respaldo teórico importante, no se cuenta con el mismo grado de certeza en problemas de varias clases ni mucho menos en contexto no supervisado. La tesis doctoral [19] en curso se centra en esta problemática, encontrando y profundizando resultados tanto empíricos como teóricos sobre el problema de la estimación de la densidad uni y multidimensional a partir de las técnicas de agregación y sobre la extensión del Boosting para los problemas de varias clases.

5. Referencias

[1] Hastie, T., Tibshirani, R. y Friedman, J; *The Elements of Statistical Learning*, Springer-Verlag 2001, New York.

[2] Vapnik, V; *Statistical Learning Theory*, Wiley 1998, New York.

[3] Breiman, L., Friedman, J.,Olshen, R., y Stone, C; *Classification and Regression Trees.* Belmont, CA 1984: Chapman & Hall.

[4] Breiman, L; *Bagging predictors*, Machine Learning, 1996. 24(2): p. 123–140.

[5] Bradley, E y Tibshirani, R; *An Introduction to the Bootstrap*, Chapman & Hall/CRC 1993.

[6] Breiman, L; *Random forests*, Machine Learning, 2001. 45(1): p. 5–32.

[7] Freund, Y. y Schapire, E; *A decision-theoretic generalization of on-line learning and application to boosting*, Journal of Computer and System Sciences, 1997. 55(1): p 119-13.

[8] Friedman, J., Hastie, T. y Tibshirani, R; *Additive logistic regression: a statistical view of boosting*, Annals of Statistics, 2000. 28: p. 337–407.

[9] Breiman, L; *Arcing Classifiers*, The Annals of Statistics, 1998. 26(3): p. 801 – 849.

[10] Zhu, J., Zou, H. y Rosset, S., Hastie, T; *Multi-class Adaboost*, Statistics and its Interface, 2009. 2: p. 349 – 360.

[11] Saberian, M. J y Vasconcelos, N; *Multiclass Boosting: Theory and Algorithms*, Neural information Processing Systems, 2011.

[12] Mukherjee, I. y Schapire, R.E; *A theory of multiclass boosting*, Neural information Processing Systems, 2010.

[13] Wolpert, D.H; *Stacked Generalization*, Neural Networks, 1992. 5: p. 241–259.

[14] Breiman, L; *Stacked regression*, Machine Learning, 1996. 24(1): p. 49–64.

[15] Ting, K.M y Witten, I.H; *Issues in Stacked Generalization*, Journal of Artificial Intelligence Research, 1999. 10: p. 271 – 289.

[16] Smyth,P. y Wolpert, D.H; *Linearly combining density estimators via stacking*, Machine Learning, 1999. 36(1-2): p. 59 – 83.

[17] Bourel, M. y Ghattas, B; *Aggregating density estimators: an empirical study*, Enviado a publicación. 2012.

[18] Rigollet, P. y Tsybakov, A.B; *Linear and convex aggregation of density estimators*, Math. Methods Statist., 2007. 16(3): p. 260 – 280.

[19] Bourel, M; *Apprentissage statistique par agrégation de modèles*, tesis doctoral, Universidad de la República, Uruguay-Université d'Aix-Marseille, France. En curso.