UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Informe - Introducción a Boosting (AdaBoost)

Integrantes:	Gustavo Aguilar Morita
	Sandra Hernández Yamunaqué
Curso:	Inteligencia Computacional
Programa:	Magíster en Ingeniería Informática
Profesor Cátedra:	Max Chacón Pacheco
Profesor Laboratorio:	Héctor Rojas Pescio

20 de Julio de 2021

1. Introducción

Boosting es un método que permite, elevar o mejorar la precisión de cualquier tipo de algoritmo de aprendizaje. De lo anterior, surge un concepto por el cual trasciende la metología en general, que es el **weak learner** (aprendiz débil).

Este último mediante una serie de lógicas de ensambles homogéneas, esto es, que se aplique el mismo algoritmo de aprendizaje para todos los modelos, logran generar un **strong learner** (aprendiz robusto o fuerte).

Algunas características importantes es que es un modelo que se construye de manera secuencial (representado en la figura 1), se conoce el error de cada modelo construído y disminuye significativamente el error de cada modelo anterior.

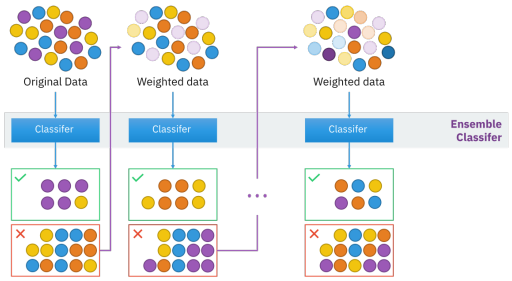

Figura 1: Ensamble secuencial del método de boosting. Cada iteración aporta significativamente al siguiente modelo.

Esta idea tiene sus cimientos en el modelo de PAC, propuesto por Kearns and Valiant (1994), donde en los próximos apartados, se verá que el rendimiento obtenido al mejorar levemente una probabilidad totalmente arbitraria, se logra construir un modelo de aprendizaje completamente robusto.

En este documento se verá uno de los algoritmos más populares del "Boosting", *AdaBoost* (**Ada**ptive **Boo**sting), donde el foco es iterar sobre la base de un aprendizaje sencillo y utilizar estrategias de mejora en los pesos de las distribuciones sobre el conjunto de entrenamiento.

En la figura 2 se tiene una representación de los clasificadores $h1$, $h2$ y $h3$, que van mejorando conforme se identifique aquellos puntos que han sido mal clasificados, y con pesos que

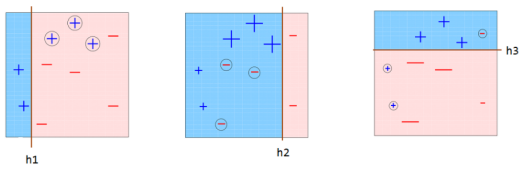

Figura 2: Mejora secuencial de clasificadores en *AdaBoost* para un conjunto de clases binarias

.

distribuyan la importancia requerida para obtener un mejor desempeño.

Uno de los aspectos fundamentales de AdaBoost, está en el análisis de los errores, en específico en la reducción del error de entrenamiento, donde Freund et al. (1999) demuestran que este puede ser acotado en términos de los errores singulares de las hipótesis más débiles. Por otra parte, se verá en este documento algunos detalles de la generalización del error, donde se puede observar que para un número grande de iteraciones de hipótesis débiles, el *boosting* no conduce a un overfit.

2. Metodología

2.1. AdaBoost

AdaBoost es un algoritmo propuesto por Freund y Schapire en 1995. El algoritmo se basa en el método Boosting por lo cual a partir de clasificadores sencillos o débiles se planea construir o formar un clasificador robusto o fuerte.
AdaBoost se presenta como un algoritmo que supera a los algoritmos anteriores basados en Boosting, pues este logra resolver las debilidades de la metodología.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

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

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Figure 1: The boosting algorithm AdaBoost.

Figura 3: Algoritmo de Boosting AdaBoost. Extraido del paper: A short introduction to boosting(Freund et al., 1999)

La Figura 3 muestra el pseudocódigo de AdaBoost para un clasificador de un sola clase, el cual a partir de un conjunto de entrenamiento (x_m, y_m) como entrada y siendo x_i un dato perteneciente al dominio de X e Y_i la etiqueta del dato que indica si el dato pertenece (1) o no a la clase (-1), se procede asignar pesos de igual valor para todo el conjunto de entrenamiento para luego empezar el entrenamiento o las rondas que buscan minimizar el error generado por el clasificador débil usado en la ronda t hasta completar las T rondas. En cada una de las rondas, cada clasificador débil tiene el objetivo de encontrar una hipótesis débil apropiada para la distribución D_t, luego para evaluar que tan bien funciona la hipótesis generada, se calcula el error de la misma de acuerdo a la distribución de la ronda t correspondiente (D_t). A partir del error obtenido en la ronda (ϵ_t) se calcula el alpha

t. Mientras menor sea el error mayor será el valor del α_t y viceversa, con este valor se actualizan los pesos de los datos y ya que se está bajo el concepto de boosting la fórmula utilizada para actualizar los pesos utiliza el alpha para aumentar el peso de aquellos datos que fueron clasificados erróneamente por el clasificador débil y su vez disminuir el peso de los datos bien clasificados. El objetivo de lo anterior es que para la siguiente ronda t los datos con mayor error sean considerados más importantes que el resto y ser si o si cubiertos por el clasificador débil correspondiente. Cabe señalar que la fórmula que se utiliza para actualizar no entrega valores normalizados por lo que se utiliza un función Z_t para llevar a cabo la tarea de normalizar los datos. Finalmente luego de terminar las rondas el algoritmo entrega la hipótesis robusta o fuerte ($H(x)$) construida a partir del valor ponderado de los pesos asignados a las hipótesis débiles, cuyo valor corresponde al α_t obtenido en cada ronda.

2.2. Error de entrenamiento

En *AdaBoost* su idea fundamental es la reducción del error de entrenamiento, y para ello se propone considerar una mejora significativa respecto a un problema totalmente arbitrario, tal que $\frac{1}{2} - \gamma_t$, donde γ_t corresponde a un valor > 0 y acotado superiormente. Es evidente que si $\gamma_t \sim 0$, entonces la mejora en la hipotesis individual no es significativa. Al considerar ϵ_t como la tasa de error de cada hipótesis débil, Freund and Schapire. (1997) demuestran que el error de hipótesis final puede ser expresado en términos de este parámetro de mejora (γ_t), acotadas por una expresión que converge a medida que se tienen mejoras relevantes ($\gamma_t > 0$), como se muestra en la ecuación 2.

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

Un ejemplo de esta expresión, suponer que todos las mejoras γ_t son al menos del 8 %, esto quiere decir que ningún error de hipótesis débil h_t tendrá un valor superior a 42 %, entonces tendremos que:

$$(\sqrt{1-4(0,08)^2})^T \approx (0,99)^T \tag{2}$$

De la figura 4, se puede ver la rapidez de decaemiento al incrementar el "número de rondas de mejora" T. Ahora bien, de esto surge el cuestionamiento, hasta qué punto se debe permitir el entrenamiento, con el fin de evitar el *overfitting*. En el apartado del Error Generalizado, permitirá comprender de mejor manera el comportamiento entorno a esta conjetura preliminar.

De acuerdo a una experimentación realizada por Freund and Schapire. (1996), se logra observar cómo el boosting mejora significativamente la tasa de error del modelo, al ver la figura 5, si se considera por ejemplo $\frac{y}{x} \sim 1$ tal que no hay diferencias, entonces aquellos puntos que se encuentra sobre la curva ($y > x$), son evidencia de la mejora aplicada.

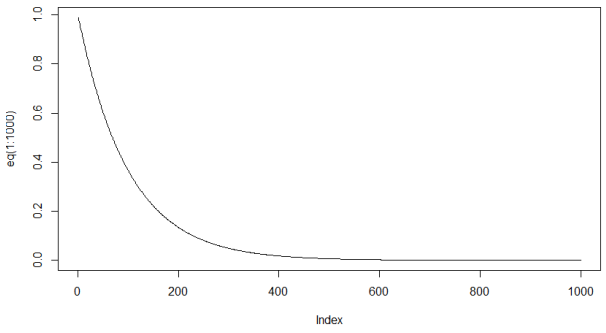

Figura 4: Descenso rápido del error de hipótesis total, para mejoras entorno al 8 % de cada hipótesis débil para un $T = 1000$.

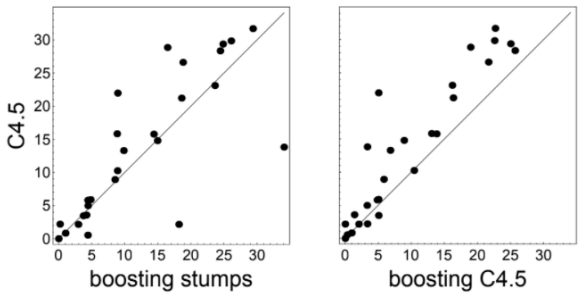

Figura 5: Ejemplo de dos situaciones de aplicación de boosting, tanto stumps como en C4.5. Donde el eje y corresponde a la tasa de error porcentual de pruebas y el eje x corresponde a la tasa de error de los boost.

2.3. Error generalizado

Considerar a continuación, el tamaño de la muestra m, VC-dimension d y el número de rondas de boosting T, se puede expresar el error generalizado de la hipótesis final como:

$$err(H) \leq \widehat{err(H)} + O(\sqrt{\frac{Tln|H| + Tln\frac{m}{T} + ln\frac{1}{\sigma}}{m}}) \tag{3}$$

De la ecuación 3 se desprenden algunas cosas interesantes, como por ejemplo, si T fuese muy pequeño, el primer término del lado derecho de la inecuación es dominante, por lo que se observa la caída rápida (ilustrada en la figura 4) y para T grandes, el segundo término domina y por lo tanto se observa un incremento sustancial, por lo que podría causar

preliminarmente algún tipo de $overfit$, como se observa en la figura 6. Sin embargo, en

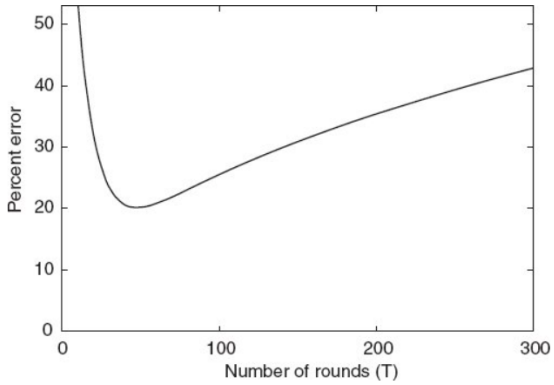

Figura 6: Gráfico que representa la ecuación 3, con $\gamma = 0,2$, $m = 10^6$, $ln|H| = 10$ y $\sigma = 0,05$.

diversas experimentaciones se logra evidenciar que esto no ocurre, de hecho el perfil típico que se obtiene es representada en la figura 7, donde el error de test sigue decreciendo, aún cuando el error de entrenamiento está entorno al 0.

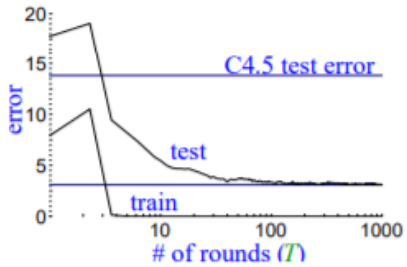

Figura 7: Perfil típico de boosting, de los errores de test y de entrenamiento, para un problema de C4.5.

3. Conclusión

El paper revisado para el desarrollo del presente informe y que lleva el título de "A short introduction to boosting " (Freund et al., 1999) nos habla de la metodología Boosting a través del algoritmo AdaBoost, el método como el algoritmo destacan por su simplicidad y cómo a partir de ella se pueden obtener resultados de muy buena calidad. AdaBoost presenta una nueva manera de abordar tipos de problemas relacionados al aprendizaje como lo son los clasificadores por ejemplo, la cual consiste cambiar la forma de abordar el problema, la tendencia actual es centrar los esfuerzos en crear un algoritmo que obtenga el resultado deseado al ejecutarlo, en cambio AdaBoost busca encontrar algoritmos fáciles de implementar y de resultados mejores a los que se obtendría con una implementación basada en la aleatoriedad de manera que el conjunto de algoritmos cubran las debilidades de se su predecesor (recordar que AdaBoost se ejecuta de manera secuencial) y de esta manera lograr obtener un buen resultado final. Una de las ventajas destacable del algoritmo es la modularidad del código, el cual lo vuelve fácil de programar y de probar con diversas funciones para calcular las ponderaciones o usar variadas distribuciones, pues cada uno representa un modulo del código, es decir, AdaBoost es un algoritmo flexible que presenta varias ventajas con respecto a sus predecesores y es por ello que ha sido probado en diversos estudios ya sea para compararlo con otros métodos y/o algoritmos o para evaluar la eficiencia del algoritmo, en ambos casos los resultados son favorecedores para AdaBoost al ser comparado e incluso resulta ser un algoritmo bastante eficiente. AdaBoost a pesar de se tener varias puntos a favor, no es perfecto, entre las debilidades de AdaBoost nos encontramos que al destacar aquellos datos mal clasificados es sensible a los datos atípicos, por lo que el rendimiento del algoritmo depende de los datos y de qué tan débiles sean los algoritmos que constituirán al clasificador final.

Bibliografía

Freund, Y., Schapire, R., and Abe, N. (1999). *A short introduction to boosting*, volume 14. JAPANESE SOC ARTIFICIAL INTELL.

Freund, Y. and Schapire., R. E. (1996). Experiments with a new boosting algorithm.

Freund, Y. and Schapire., R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting.

Kearns, M. J. and Valiant, L. G. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*.