# Informe sobre Panel Conversatorio

"Transformación Digital y Ecosistema Digital"

Elias Sebastian Gill Quintana

## Resumen del documento: Boosting y AdaBoost

En este informe voy a explicar lo que dice el capítulo del PDF sobre el tema de **Boosting**, pero con un lenguaje mucho más sencillo y coloquial. No me voy a quedar en tecnicismos innecesarios, sino que voy a desarmar las fórmulas para que sean entendibles. Cada símbolo lo voy a ir comentando como si lo charláramos en una clase.

### 2.1 Procedimiento General de Boosting

La idea central de Boosting es muy simple: *si tenemos clasificadores flojitos (llamados débiles), que son apenas mejores que adivinar al azar, podemos combinarlos de manera inteligente para obtener un clasificador mucho más fuerte y confiable.*

El procedimiento general (Figura 2.1 del texto) funciona así:

```
Input: Distribución de datos D
       Algoritmo base de aprendizaje L
       Número de rondas T

Proceso:
  1. Empezar con D1 = D
  2. Para cada ronda t = 1,...,T:
       - Entrenar un clasificador ht con L usando Dt
       - Calcular el error t
       - Ajustar la distribución para la siguiente ronda Dt+1
  3. Combinar todos los clasificadores h1,...,hT
Output: Clasificador final H(x)
```

En pocas palabras: cada vez que entrenamos un nuevo clasificador, le damos más importancia a los ejemplos donde los anteriores se equivocaron.

### 2.2 El Algoritmo AdaBoost

AdaBoost es la versión más famosa de Boosting (Freund y Schapire, 1997). Funciona para clasificación binaria, donde las clases son $\{-1, +1\}$.

El algoritmo completo es este (Figura 2.2):

```
Input: Datos D = {(x1,y1),...,(xm,ym)}
       Algoritmo base L
       Número de rondas T
```

```
Proceso:
  1. Inicializar distribución D1(x) = 1/m
  2. Para t = 1,...,T:
        - Entrenar ht usando distribución Dt
        - Calcular error t = P(ht(x)  f(x)) bajo Dt
        - Si t > 0.5, detener
        - Calcular peso t = (1/2) * ln((1-t)/t)
        - Actualizar distribución:
          Dt+1(x) = Dt(x) * exp(-t * f(x) * ht(x)) / Zt
  3. Clasificador final:
     H(x) = sign(  t ht(x) )
```

**Explicación coloquial de cada símbolo:**

- $x$: un ejemplo de entrenamiento (una fila de los datos). - $y$: la etiqueta real del ejemplo ($+1$ o $-1$). - $ht(x)$: el clasificador débil en la ronda $t$. - $\epsilon_t$: el error del clasificador $ht$ (cuántas veces se equivoca, en proporción). - $\alpha_t$: el peso que le damos al clasificador $ht$. Si es muy bueno, se le da un peso alto, si es malo, se le da un peso bajo. - $D_t$: la distribución de pesos sobre los ejemplos en la ronda $t$. Básicamente, dice a qué ejemplos hay que prestar más atención. - $Z_t$: un número para normalizar y que $D_{t+1}$ siga siendo una distribución válida. - $H(x)$: el clasificador final, que combina a todos.

**Función de pérdida exponencial**

AdaBoost se puede entender como minimizar la siguiente función de pérdida:

$$\mathcal{L}_{exp}(h|D) = E_{x\sim D}\left[e^{-f(x)h(x)}\right]$$

- $f(x)$: la etiqueta real. - $h(x)$: la predicción del clasificador. - La idea es: si $h(x)$ acierta, la pérdida baja; si falla, la pérdida sube de forma exponencial.

De ahí sale que el clasificador combinado es:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Y la decisión final es tomar el signo:

$$sign(H(x)) = \{ +1 si la suma favorece a +1 - 1 si la suma favorece a -1$$

Esto corresponde a la probabilidad más alta (el clasificador final se acerca a la decisión Bayesiana).

## 2.3 Ejemplo Ilustrativo (XOR)

El texto explica cómo AdaBoost resuelve un problema clásico: el **XOR**. Con 4 puntitos en un plano, ningún clasificador lineal puede separarlos. Pero AdaBoost, combinando varios clasificadores lineales débiles, logra construir un clasificador no lineal que separa perfecto.

La clave es que, al equivocarse en un ejemplo, ese ejemplo recibe más peso en la siguiente ronda, y tarde o temprano otro clasificador lo corrige. Cuando combinamos todo, funciona.

## 2.4 Fundamentos Teóricos

AdaBoost tiene garantías teóricas de que el error baja rápido:

$$\epsilon \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}$$

Donde: - $\epsilon$: el error final. - $\gamma_t = 0{,}5 - \epsilon_t$: lo que llaman el "margen de ventaja" del clasificador $h_t$.

Esto significa que mientras cada clasificador sea un poquito mejor que azar, el error total se reduce exponencialmente al combinarlos.

Además, se estudian los márgenes (qué tan confiado está el clasificador en sus decisiones). AdaBoost tiende a aumentar los márgenes, lo cual explica por qué suele no sobreajustar incluso con muchos clasificadores.

## 2.5 Extensiones a Multiclase

Originalmente AdaBoost era para 2 clases, pero existen variantes como: - AdaBoost.M1: usa clasificadores multiclase directos. - SAMME: ajusta la fórmula del peso $\alpha_t$ agregando $\ln(|Y|-1)$, donde $|Y|$ es el número de clases.

## 2.6 Tolerancia al Ruido

AdaBoost puede ser muy sensible al ruido (ejemplos mal etiquetados), porque aumenta mucho el peso de ejemplos mal clasificados. Existen variantes más robustas como:

- **MadaBoost**: limita cuánto puede crecer el peso de un ejemplo. - **FilterBoost**: usa pérdida logarítmica en vez de exponencial. - **BrownBoost y RobustBoost**: se basan en procesos estocásticos (movimiento browniano, Ornstein-Uhlenbeck) y básicamente aprenden a "ignorar" ejemplos demasiado ruidosos.

## 2.7 Conclusión y Lecturas Adicionales

El capítulo cierra diciendo que Boosting fue un descubrimiento clave en aprendizaje automático. AdaBoost en particular mostró que es posible convertir clasificadores mediocres en clasificadores muy buenos. Además, abrió muchas preguntas teóricas, como por qué a veces parece inmune al sobreajuste.

Hay muchas variantes (LogitBoost, LPBoost, GentleBoost, etc.), cada una con sus matices, pero la idea general sigue siendo la misma: *aprender de los errores, reequilibrar los datos, y combinar muchos débiles para obtener un fuerte.*