Boosting para Clasificación

Curso de Machine Learning

Motivación

- Un solo modelo puede no ser suficiente para generalizar bien.
- Modelos simples suelen tener alto sesgo o cometer errores sistemáticos.
- Idea: combinar múltiples modelos débiles para formar uno fuerte.

Ensemble Learning

- Técnicas que combinan varios modelos.
- Buscan mejorar estabilidad y generalización.
- Familias: Bagging, Boosting, Stacking.

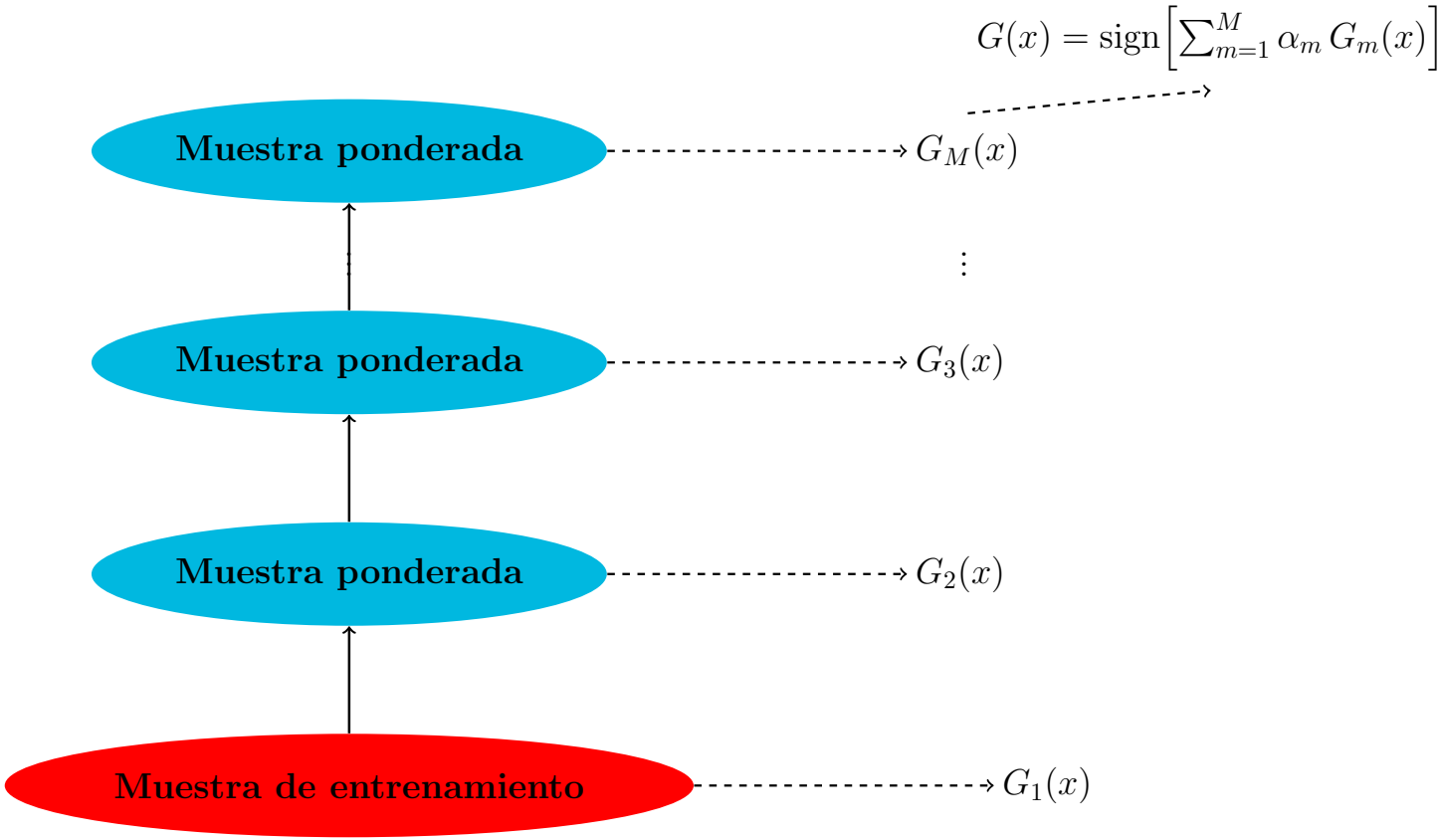
Weak learner vs Strong learner

- **Weak learner:** desempeño apenas mejor que el azar.
- **Strong learner:** alto desempeño predictivo.
- Boosting convierte una colección de weak learners en un strong learner.

Idea central de Boosting

- Entrena modelos **secuencialmente**.
- Cada modelo intenta **corregir** los errores del anterior.
- Las muestras difíciles ganan más influencia en iteraciones posteriores.

Esquema conceptual del Boosting



¿Qué es AdaBoost?

- AdaBoost = *Adaptive Boosting*.
- Ajusta un modelo débil por iteración usando datos **ponderados**.
- Combina los modelos por **votación ponderada**.

AdaBoost.M1: Supuestos

- Clasificación binaria.
- Etiquetas: $y \in \{-1, +1\}$.
- El aprendiz base suele ser simple (p.ej. *decision stump*).

Notación

- Datos: $\{(x_i, y_i)\}_{i=1}^{N}$
- Pesos por muestra: w_i con $\sum_i w_i = 1$
- Iteraciones: $m = 1, \ldots, M$
- Clasificadores débiles: $G_m(x)$

Inicialización de pesos

$$w_i \leftarrow \frac{1}{N}, \quad i = 1, \ldots, N$$

- Al inicio, todas las muestras importan por igual.

Iteración m: visión general

1. Entrenar G_m con pesos w_i.
2. Medir el error ponderado err_m.
3. Calcular la importancia del modelo α_m.
4. Reponderar las muestras (suben las mal clasificadas).

Aprendiz base típico: Decision stump

- Árbol de decisión de profundidad 1.
- Regla con una sola variable y un umbral:

$$G(x) = \begin{cases} +1 & \text{si } x_j \geq t \\ -1 & \text{si } x_j < t \end{cases}$$

¿Qué "optimiza" el stump?

- Dado w_i, el stump busca (j, t) que minimiza:

$$\sum_{i=1}^{N} w_i \, \mathbb{I}\big(y_i \neq G(x_i)\big)$$

- AdaBoost no fija t; lo encuentra el aprendiz base para cada iteración.

Error ponderado

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i \, \mathbb{I}\big(y_i \neq G_m(x_i)\big)}{\sum_{i=1}^{N} w_i}$$

- Si $\sum_i w_i = 1$, entonces $\mathrm{err}_m = \sum_i w_i \, \mathbb{I}(\cdot)$.

Peso del clasificador

$$\alpha_m = \log\left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m}\right)$$

- Si err_m es pequeño, α_m es grande (modelo más influyente).
- Si $\mathrm{err}_m \approx 0{,}5$, entonces $\alpha_m \approx 0$ (no aporta).

Actualización de pesos

$$w_i \leftarrow w_i \cdot \exp\Big(\alpha_m \, \mathbb{I}(y_i \neq G_m(x_i))\Big)$$

- Si $y_i \neq G_m(x_i)$ entonces w_i se multiplica por e^{α_m}.
- Si acierta, w_i queda igual.

Normalización

- Tras actualizar, se renormaliza para que $\sum_i w_i = 1$.
- Esto estabiliza el entrenamiento y permite interpretar w_i como distribución.

Clasificador final

$$G(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m \, G_m(x)\right)$$

- Es una votación ponderada por las importancias α_m.

Algoritmo AdaBoost.M1

Algorithm 1: AdaBoost.M1

Input: Datos de entrenamiento $\{(x_i, y_i)\}_{i=1}^N$ con $y_i \in \{-1, +1\}$; número de iteraciones M; aprendiz base

Output: Clasificador final $G(x)$

1 Inicializar pesos de observación $w_i \leftarrow \frac{1}{N}$, para $i = 1, 2, \ldots, N$

2 **for** $m \leftarrow 1$ **to** M **do**

3 **(a)** Ajustar un clasificador $G_m(x)$ sobre los datos usando pesos w_i

4 **(b)** Calcular

$$\text{err}_m = \frac{\sum_{i=1}^N w_i \, \mathbb{I}\big(y_i \neq G_m(x_i)\big)}{\sum_{i=1}^N w_i}.$$

(c) Calcular

$$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right).$$

(d) Actualizar pesos:

$$w_i \leftarrow w_i \cdot \exp\Big(\alpha_m \, \mathbb{I}\big(y_i \neq G_m(x_i)\big)\Big), \qquad i = 1, 2, \ldots, N.$$

5 **Salida:** $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m \, G_m(x)\right]$

Lectura práctica del algoritmo

- AdaBoost define una **distribución** sobre las muestras (los w_i).
- El aprendiz base se entrena con esos pesos.
- El error y α_m determinan cuánto confiar en G_m.
- La reponderación fuerza a atender los errores.

¿Qué "aprende" realmente AdaBoost?

- Aprende una combinación lineal de predictores base:

$$F(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

- La predicción final es el signo de $F(x)$.

Interpretación geométrica

- Cada stump agrega una frontera simple.
- La suma ponderada puede aproximar fronteras complejas.
- Se obtiene flexibilidad sin un solo modelo muy complejo.

Ejemplo mental: por qué los pesos importan

- Si una muestra es difícil y se clasifica mal, su peso sube.
- En la siguiente iteración, ese error “cuesta más”.
- Por tanto, el nuevo clasificador cambia para reducir ese costo.

Relación con la pérdida exponencial (idea)

- AdaBoost está relacionado con minimizar una pérdida tipo exponencial.
- La reponderación puede verse como una forma de aumentar penalización a errores.

Ventajas

- Suele funcionar muy bien con modelos base simples.
- Mejora progresiva al enfocarse en errores.
- Conceptualmente claro para introducir Boosting.

Limitaciones

- Sensible a ruido y outliers (pueden recibir peso excesivo).
- Si el aprendiz base no supera el azar, se estanca.
- No entrega probabilidades de manera natural (sin calibración).

Buenas prácticas

- Controlar M y/o usar *early stopping*.
- Usar stumps o árboles pequeños para mantener interpretabilidad.
- Validar con partición train/validation y métricas apropiadas.

AdaBoost como base conceptual de métodos modernos

- AdaBoost ayuda a entender la idea de ensamble secuencial.
- Gradient Boosting y XGBoost agregan optimización explícita y regularización.

Conclusiones

- Boosting: combinación secuencial de modelos débiles.
- AdaBoost.M1 adapta la atención mediante pesos w_i.
- El ensamble final es una votación ponderada por α_m.
- Es un puente excelente entre teoría e implementación moderna.