Informe sobre Panel Conversatorio

“Transformación Digital y Ecosistema Digital”

Elias Sebastian Gill Quintana

Resumen completo (Boosting y AdaBoost) — todas las fórmulas incluidas

Perdón por la omisión anterior — ya preparé una versión completa que incluye **todas** las fórmulas y pseudocódigos que aparecen en el PDF, con explicaciones coloquiales y claras.

2.1 Procedimiento general de Boosting

Boosting entrena clasificadores secuencialmente y ajusta una distribución sobre los ejemplos para forzar a los siguientes clasificadores a corregir los errores de los anteriores.

```
Input: Sample distribution D;
       Base learning algorithm L;
       Number of learning rounds T.
Process:
  1. D1 = D.  % Initialize distribution
  2. for t = 1, ..., T :
      ht = L(Dt);  % Train a weak learner from distribution Dt
      t = P_{x ~ Dt}(ht(x) != f(x)); % error
      Dt+1 = Adjust Distribution(Dt, t)
  3. Output: H(x) = Combine Outputs({h1(x), ..., hT(x)})
```

Explicación simple: arrancás con pesos iguales; cada ronda entrenás un débil y aumentás el peso de lo que falló, así el siguiente clasificador lo intenta arreglar.

2.2 AdaBoost — algoritmo, fórmulas y derivaciones

Pseudocódigo (fiel al PDF):

```
Input: Data set D = {(x1,y1),...,(xm,ym)};
       Base learning algorithm L;
       Number of learning rounds T.
Process:
  1. D1(x) = 1/m. % Initialize the weight distribution
  2. for t = 1, ..., T :
       ht = L(D,Dt); % Train a classifier ht from D under distribution Dt
       t = P_{x~Dt}(ht(x) != f(x));
       if t > 0.5 then break
       t = (1/2) * ln((1-t)/t);
```

```
    Dt+1(x) = Dt(x)/Zt * exp(-t f(x) ht(x));
3. Output: H(x) = sign(_t t ht(x))
```

A continuación copio TODAS las fórmulas importantes y las explico en cristiano:

Pérdida exponencial (ecuación 2.1)

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

- **Qué significa:** si la predicción coincide con la etiqueta ($f(x)h(x) = 1$) el exponente es -1 y la pérdida baja; si se equivocan ($f(x)h(x) = -1$) la pérdida crece rápido (exponencialmente). Por eso AdaBoost castiga errores de forma acumulativa.

Modelo aditivo (ecuación 2.2)

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

- **Qué significa:** es una suma ponderada de clasificadores débiles. La salida final usa el signo: $\hat{y}(x) = \mathrm{sign}(H(x))$.

Relación con probabilidades y derivada (ecuaciones 2.3–2.5)

Derivando la pérdida exponencial para buscar el mínimo nos encontramos con (2.3):

$$\frac{\partial}{\partial H(x)} e^{-f(x)H(x)} = -f(x)e^{-f(x)H(x)} = -e^{-H(x)}P(f(x) = 1 \mid x) + e^{H(x)}P(f(x) = -1 \mid x).$$

Igualando a cero y resolviendo (2.4):

$$H(x) = 12 \ln \frac{P(f(x) = 1 \mid x)}{P(f(x) = -1 \mid x)}.$$

Y por eso el signo de $H(x)$ da la clase más probable (2.5):

$$\mathrm{sign}(H(x)) = \arg \max_{y\in\{-1,1\}} P(f(x) = y \mid x).$$

- **Explicación coloquial:** minimizar la pérdida exponencial equivale, en esencia, a aproximar log-proporciones de probabilidades y a elegir la clase con mayor probabilidad.

Pérdida de un solo término y cálculo de α_t (ecuaciones 2.6–2.8)

Bajo la distribución D_t, la pérdida del término $\alpha_t h_t$ es:

$$\mathcal{L}_{exp}(\alpha_t h_t \mid D_t) = E_{x\sim D_t}[e^{-\alpha_t f(x)h_t(x)}] = e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t,$$

donde $\epsilon_t = P_{x\sim D_t}(h_t(x) \neq f(x))$.

Derivando respecto de α_t y poniendo 0 (ecuación 2.7) obtenemos la solución cerrada (ecuación 2.8):

$$\alpha_t = 12 \ln \Big(\frac{1 - \epsilon_t}{\epsilon_t}\Big).$$

- **En palabras:** si h_t es mejor que azar ($\epsilon_t < 0{,}5$) entonces $\alpha_t > 0$. Si es muy bueno (ϵ_t pequeño), su peso crece.

Selección de h_t mediante reponderación (ecuaciones 2.9–2.15)

Tomando el combinado hasta la ronda $t-1$ como H_{t-1} y buscando el mejor h para añadir, la pérdida exponencial total es (2.9):

$$\mathcal{L}_{exp}(H_{t-1} + h \mid D) = E_{x\sim D}[e^{-f(x)(H_{t-1}(x)+h(x))}].$$

Aproximando $e^{-f(x)h(x)}$ por su expansión de Taylor (ecuación 2.10) y reorganizando se llega a definir una distribución D_t (ecuación 2.12):

$$D_t(x) = \frac{D(x)e^{-f(x)H_{t-1}(x)}}{E_{x\sim D}[e^{-f(x)H_{t-1}(x)}]}.$$

Bajo esa D_t la selección de h_t es equivalente a (ecuaciones 2.13–2.14):

$$h_t = \arg\max_h E_{x\sim D_t}[f(x)h(x)] = \arg\min_h E_{x\sim D_t}[I(f(x) \neq h(x))].$$

Es decir: elegir el clasificador que minimice el error bajo la distribución actualizada.

Actualización explícita de D_{t+1} (ecuaciones 2.15, 2.34)

La relación entre D y D_{t+1} es (2.15):

$$D_{t+1}(x) = \frac{D(x)e^{-f(x)H_t(x)}}{E_{x\sim D}[e^{-f(x)H_t(x)}]}.$$

En la implementación práctica queda como (línea 7 de la figura y 2.34):

$$D_{t+1}(x) = \frac{D_t(x)}{Z_t} e^{-\alpha_t f(x)h_t(x)}.$$

Expandiendo desde D_1: (ecuación 2.34 express)

$$D_{t+1}(x) = \frac{D_1(x)}{Z'_t} \prod_{i=1}^{t} e^{-\alpha_i h_i(x) f(x)}.$$

2.3 Ejemplos y comportamiento empírico

El capítulo muestra cómo AdaBoost combina clasificadores lineales débiles para resolver XOR y cómo en muchos conjuntos del UCI mejora al clasificador base (Figuras 2.3, 2.5, 2.6).

2.4 Teoría adicional: cotas y márgenes

Cota de Freund Schapire (ecuación 2.16) y número de rondas (2.17)

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}, \quad \gamma_t = 0{,}5 - \epsilon_t.$$

De ahí, bajo supuestos, se deduce (2.17):

$$T \leq \frac{1}{2\gamma^2} \ln \frac{1}{\epsilon}.$$

Cotas de generalización y margen (ecuaciones 2.18–2.20)

Generalización (2.18):

$$\epsilon_{gen} \leq \epsilon_{train} + \tilde{O}\Big(\sqrt{\frac{dT}{m}}\Big).$$

Definición de margen (2.19):

$$margen(x) = f(x)H(x), \qquad margennormalizada = \frac{\sum_t \alpha_t f(x) h_t(x)}{\sum_t \alpha_t}.$$

Cota basada en el porcentaje de ejemplos con margen menor que θ (2.20):

$$\epsilon_{gen} \leq P_{x\sim D}(f(x)H(x) \leq \theta) + \tilde{O}\Big(\sqrt{\frac{d}{m\theta^2}} + \sqrt{\frac{\ln(1/\delta)}{m}}\Big).$$

Breiman propone la mínima margen (2.21):

$$\rho = \min_{x\in D} f(x)H(x),$$

y su arc-gv modifica el peso con la fórmula (2.22):

$$\alpha_t = 12 \ln\Big(\frac{1+\gamma_t}{1-\gamma_t}\Big) - 12 \ln\Big(\frac{1+\rho_t}{1-\rho_t}\Big).$$

2.4.3 Vista estadística: LogitBoost (ecuaciones 2.23–2.24 y figura 2.9)

Estimación de probabilidad (2.23):

$$P(f(x) = 1 \mid x) = \frac{e^{H(x)}}{e^{H(x)} + e^{-H(x)}} = \frac{1}{1 + e^{-2H(x)}}.$$

Pérdida logarítmica (2.24):

$$\mathcal{L}_{\log}(h \mid D) = E_{x\sim D}[\ln(1 + e^{-2f(x)h(x)})].$$

LogitBoost (fig. 2.9): pseudocódigo ya incluido arriba — utiliza actualizaciones tipo Newton:

```
pt(x) = 1/(1+e^{-2 H_{t-1}(x)})
yt(x) = (y_{t-1}(x) - pt(x)) / (pt(x)(1-pt(x)))
Dt(x) = pt(x)(1-pt(x))
ht = L(D, yt, Dt)
Ht = H_{t-1} + 1/2 ht
```

2.5 Multiclase: SAMME y descomposiciones

SAMME (ecuación 2.31):

$$\alpha_t = 12 \ln\Big(\frac{1-\epsilon_t}{\epsilon_t}\Big) + \ln(|Y| - 1).$$

Decisión multiclasica a partir de salidas reales (2.33):

$$\hat{y}(x) = \arg\max_{y\in Y} H_y(x), \quad H_y(x) = \sum_t \alpha_t^{(y)} h_t^{(y)}(x).$$

2.6 Robustez y variantes (MadaBoost, FilterBoost, BrownBoost, RobustBoost)

MadaBoost (regla 2.35):

$$D_{t+1}(x) = \frac{D_1(x)}{Z'_t} \min\Big\{1, \prod_{i=1}^{t} e^{-\alpha_i h_i(x) f(x)}\Big\}.$$

FilterBoost / log-loss update (2.38):

$$D_t(x) = \frac{D(x)}{Z_t} \cdot \frac{1}{1 + e^{f(x)H_{t-1}(x)}}.$$

BrownBoost (2.39–2.43):

$$\mathcal{L}_{bmp}(H_{t-1} + h) = E_{x\sim D}\Big[1 - \mathrm{erf}\Big(\frac{f(x)H_{t-1}(x) + f(x)h(x) + c - t}{\sqrt{c}}\Big)\Big],$$

$$\mathrm{erf}(a) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{a} e^{-x^2}\, dx.$$

Aproximando se llega a (2.41) y a la regla (2.42):

$$h_t = \arg\max_h E_{x\sim D}\Big[e^{-(f(x)H_{t-1}(x)+c-t)^2/c} f(x)h(x)\Big],$$

y pesos (2.43):

$$D_t(x) = \frac{D(x)}{Z_t} e^{-(f(x)H_{t-1}(x)+c-t)^2/c}.$$

RobustBoost (2.44–2.46): objetivo en márgenes (2.44) y pérdida Ornstein-Uhlenbeck (2.45) que lleva a:

$$D_t(x) = \frac{D(x)}{Z_t} \exp\Big(-\frac{(f(x)H_{t-1}(x) - \mu(t_c))^2}{2\sigma(t_c)^2}\Big).$$

2.7 LPBoost y programación (ecuaciones 2.25–2.30)

Planteo de complejidad (2.25):

$$\sum_{h\in H} \alpha_h + C \sum_{i=1}^{m} \xi_i \le B.$$

Primal (2.27):

$$\min_{\alpha,\xi} \sum_{j=1}^{T} \alpha_j + C \sum_{i=1}^{m} \xi_i \quad s.t. y_i \sum_{j=1}^{T} H_{i,j}\alpha_j + \xi_i \ge 1.$$

Dual (2.29):

$$\min_{w,\beta} \beta \quad s.t. \sum_{i=1}^{m} w_i y_i H_{i,j} \le \beta,\ \sum_i w_i = 1,\ w_i \in [0, C'].$$

El algoritmo LPBoost (fig.2.10) implementa column generation para resolver esto.

Aclaración final de símbolos y atajos

x: ejemplo. $f(x)$: etiqueta real (± 1). $h_t(x)$: hipótesis t-ésima (± 1). ϵ_t: error de h_t bajo D_t. α_t: peso de h_t. $D_t(x)$: peso del ejemplo en la ronda t. $H(x)$: suma ponderada final.

Si querés que compile este `.tex` y te suba el PDF listo, lo hago en seguida. Abrazo — y gracias por avisar, lo corregí y ahora está todo incluido.