

Clasificadores combinados

Fernando Lozano

Universidad de los Andes

3 de noviembre de 2017



Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:
 - ▶ Obtener clasificadores h_1, h_2, \dots, h_T minimizando error en diferentes **versiones** de los datos.

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:
 - ▶ Obtener clasificadores h_1, h_2, \dots, h_T minimizando error en diferentes **versiones** de los datos.
 - ▶ Formar combinación:

$$f(\mathbf{x}) = \sum_{i=1}^T \alpha_i h_i(\mathbf{x})$$

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:
 - ▶ Obtener clasificadores h_1, h_2, \dots, h_T minimizando error en diferentes **versiones** de los datos.
 - ▶ Formar combinación:

$$f(\mathbf{x}) = \sum_{i=1}^T \alpha_i h_i(\mathbf{x})$$

- ▶ Clasificar con el signo de f

$$h(\mathbf{x}) = \text{signo}(f(\mathbf{x})) = \begin{cases} 1 & \text{si } f(\mathbf{x}) \geq 0, \\ 0 & \text{si } f(\mathbf{x}) < 0 \end{cases}$$

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:
 - ▶ Obtener clasificadores h_1, h_2, \dots, h_T minimizando error en diferentes **versiones** de los datos.
 - ▶ Formar combinación:

$$f(\mathbf{x}) = \sum_{i=1}^T \alpha_i h_i(\mathbf{x})$$

- ▶ Clasificar con el signo de f

$$h(\mathbf{x}) = \text{signo}(f(\mathbf{x})) = \begin{cases} 1 & \text{si } f(\mathbf{x}) \geq 0, \\ 0 & \text{si } f(\mathbf{x}) < 0 \end{cases}$$

- Muy efectivos en la práctica.

Bagging (Bootstrap Aggregating)

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):
 - ▶ Técnica para mejorar precisión de parámetros/estimativos.

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):
 - ▶ Técnica para mejorar precisión de parámetros/estimativos.
 - ▶ Muestreo con remplazo.

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):
 - ▶ Técnica para mejorar precisión de parámetros/estimativos.
 - ▶ Muestreo con remplazo.
 - ▶ $Muestra \sim población$.

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):
 - ▶ Técnica para mejorar precisión de parámetros/estimativos.
 - ▶ Muestreo con remplazo.
 - ▶ $Muestra \sim$ población.
- Clasificadores h_1, h_2, \dots entrenados en muestras **bootstrap** de los datos de entrenamiento.

Bagging (Bootstrap Aggregating)

- Bootstrapping (Efron, 1979):
 - ▶ Técnica para mejorar precisión de parámetros/estimativos.
 - ▶ Muestreo con remplazo.
 - ▶ $Muestra \sim población$.
- Clasificadores h_1, h_2, \dots entrenados en muestras **bootstrap** de los datos de entrenamiento.
- Combinación por votación.

Algorithm 1 Bagging

for $t = 1$ to T **do**

 Obtenga \mathcal{S}_t de \mathcal{S} muestreando con reemplazo.

$h_t \leftarrow A(\mathcal{S}_t)$

end for

Retorne $f(x) = \text{votacion } \{h_t(x)\}$

Boosting

Boosting

- Aprendizabilidad PAC: encontrar h que satisfaga:

$$\mathbf{P}(e(h) \geq \epsilon) \leq \delta$$

para cualquier distribución de los datos.

Boosting

- Aprendizabilidad PAC: encontrar h que satisfaga:

$$\mathbf{P}(e(h) \geq \epsilon) \leq \delta$$

para cualquier distribución de los datos.

- Suponga que podemos encontrar h_d , que satisface la condición de **aprendizabilidad débil**

$$\mathbf{P}(e(h_d) \geq \epsilon_0) \leq \delta_0$$

para cualquier distribución de los datos.

Boosting

- Aprendizabilidad PAC: encontrar h que satisfaga:

$$\mathbf{P}(e(h) \geq \epsilon) \leq \delta$$

para cualquier distribución de los datos.

- Suponga que podemos encontrar h_d , que satisface la condición de **aprendizabilidad débil**

$$\mathbf{P}(e(h_d) \geq \epsilon_0) \leq \delta_0$$

para cualquier distribución de los datos.

- Es posible obtener **aprendizabilidad fuerte** a partir de **aprendizabilidad débil**?

Boosting

- Aprendizabilidad PAC: encontrar h que satisfaga:

$$\mathbf{P}(e(h) \geq \epsilon) \leq \delta$$

para cualquier distribución de los datos.

- Suponga que podemos encontrar h_d , que satisface la condición de **aprendizabilidad débil**

$$\mathbf{P}(e(h_d) \geq \epsilon_0) \leq \delta_0$$

para cualquier distribución de los datos.

- Es posible obtener **aprendizabilidad fuerte** a partir de **aprendizabilidad débil**?
- **Si!** (The strength of weak learnability, Schapire, 1996).

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo **$EX(c, \mathcal{D})$** .

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo $EX(c, \mathcal{D})$.
- Algoritmo **modesto**:

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo **$EX(c, \mathcal{D})$** .
- Algoritmo **modesto**:
 - 1 $h_1 \leftarrow A(\mathcal{D}_1)$ con $\mathcal{D}_1 = \mathcal{D}$

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo $EX(c, \mathcal{D})$.
- Algoritmo **modesto**:
 - 1 $h_1 \leftarrow A(\mathcal{D}_1)$ con $\mathcal{D}_1 = \mathcal{D}$
 - 2 $h_2 \leftarrow A(\mathcal{D}_2)$ donde \mathcal{D}_2 es tal que $\mathbf{P}_{\mathcal{D}_2} [h_1(x) \neq c(x)] = \frac{1}{2}$

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo $EX(c, \mathcal{D})$.
- Algoritmo **modesto**:
 - 1 $h_1 \leftarrow A(\mathcal{D}_1)$ con $\mathcal{D}_1 = \mathcal{D}$
 - 2 $h_2 \leftarrow A(\mathcal{D}_2)$ donde \mathcal{D}_2 es tal que $\mathbf{P}_{\mathcal{D}_2} [h_1(x) \neq c(x)] = \frac{1}{2}$
 - 3 $h_3 \leftarrow A(\mathcal{D}_3)$ donde \mathcal{D}_3 es tal que $\mathbf{P}_{\mathcal{D}_3} [h_1(x) \neq h_2(x)] = 1$

Algoritmo modesto de boosting

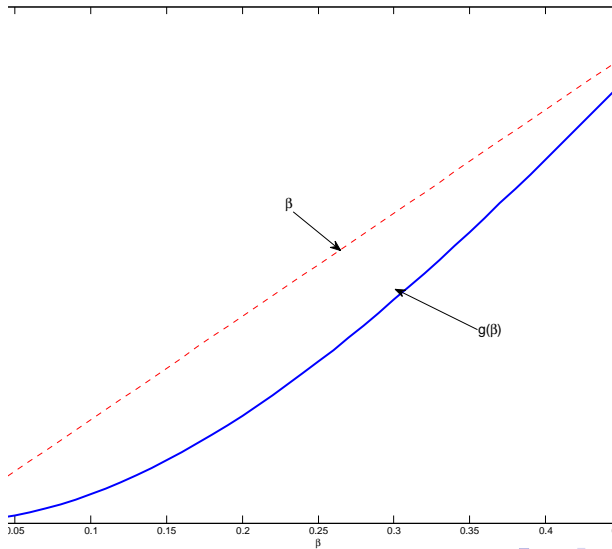
- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo $EX(c, \mathcal{D})$.
- Algoritmo **modesto**:
 - 1 $h_1 \leftarrow A(\mathcal{D}_1)$ con $\mathcal{D}_1 = \mathcal{D}$
 - 2 $h_2 \leftarrow A(\mathcal{D}_2)$ donde \mathcal{D}_2 es tal que $\mathbf{P}_{\mathcal{D}_2} [h_1(x) \neq c(x)] = \frac{1}{2}$
 - 3 $h_3 \leftarrow A(\mathcal{D}_3)$ donde \mathcal{D}_3 es tal que $\mathbf{P}_{\mathcal{D}_3} [h_1(x) \neq h_2(x)] = 1$Retorna $h(x) = \text{mayoría}(h_1(x), h_2(x), h_3(x))$

Algoritmo modesto de boosting

- Suponga que existe un algoritmo A , que toma datos $\{x_i, y_i\} \sim \mathcal{D}$, y retorna h , con $e(h) \leq \beta$, **para cualquier \mathcal{D}** .
- Podemos pedir datos (x_i, y_i) a un oráculo $EX(c, \mathcal{D})$.
- Algoritmo **modesto**:
 - 1 $h_1 \leftarrow A(\mathcal{D}_1)$ con $\mathcal{D}_1 = \mathcal{D}$
 - 2 $h_2 \leftarrow A(\mathcal{D}_2)$ donde \mathcal{D}_2 es tal que $\mathbf{P}_{\mathcal{D}_2} [h_1(x) \neq c(x)] = \frac{1}{2}$
 - 3 $h_3 \leftarrow A(\mathcal{D}_3)$ donde \mathcal{D}_3 es tal que $\mathbf{P}_{\mathcal{D}_3} [h_1(x) \neq h_2(x)] = 1$Retorna $h(x) = \text{mayoría}(h_1(x), h_2(x), h_3(x))$

- Se puede mostrar

$$\left. \begin{array}{l} \mathbf{P}_{\mathcal{D}} [h_1(x) \neq c(x)] \leq \beta \\ \mathbf{P}_{\mathcal{D}_2} [h_2(x) \neq c(x)] \leq \beta \\ \mathbf{P}_{\mathcal{D}_3} [h_3(x) \neq c(x)] \leq \beta \end{array} \right\} \Rightarrow \mathbf{P}_{\mathcal{D}} [h(x) \neq c(x)] \leq 3\beta^2 - 2\beta^3$$



- Asumimos que A retorna h con $e(h) \leq \beta < \frac{1}{2}$, para **cualquier distribución \mathcal{D}** .

- Asumimos que A retorna h con $e(h) \leq \beta < \frac{1}{2}$, para **cualquier distribución \mathcal{D}** .
- Queremos tener $e(h) \leq \varepsilon$

- Asumimos que A retorna h con $e(h) \leq \beta < \frac{1}{2}$, para **cualquier distribución \mathcal{D}** .
- Queremos tener $e(h) \leq \varepsilon$
- Si algoritmo débil garantiza $e(h) \leq g^{-1}(\varepsilon) > \varepsilon$

- Asumimos que A retorna h con $e(h) \leq \beta < \frac{1}{2}$, para **cualquier distribución \mathcal{D}** .
- Queremos tener $e(h) \leq \varepsilon$
- Si algoritmo débil garantiza $e(h) \leq g^{-1}(\varepsilon) > \varepsilon \Rightarrow$ **aplicamos algoritmo modesto**.

- Asumimos que A retorna h con $e(h) \leq \beta < \frac{1}{2}$, para **cualquier distribución \mathcal{D}** .
- Queremos tener $e(h) \leq \varepsilon$
- Si algoritmo débil garantiza $e(h) \leq g^{-1}(\varepsilon) > \varepsilon \Rightarrow$ **aplicamos algoritmo modesto**.
- Si no, podemos aplicar algoritmo modesto **recursivamente**.

Algorithm 2 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

Algorithm 3 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

Algorithm 4 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**
 Retorne $AD(\varepsilon, \mathcal{D}')$

Algorithm 5 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**
 Retorne $AD(\varepsilon, \mathcal{D}')$
end if

Algorithm 6 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

Algorithm 7 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

$h_1 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_1)$

Algorithm 8 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

$h_1 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_1)$

$h_2 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_2)$

Algorithm 9 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

$h_1 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_1)$

$h_2 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_2)$

$h_3 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_3)$

Algorithm 10 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

$h_1 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_1)$

$h_2 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_2)$

$h_3 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_3)$

$h \leftarrow \text{mayoría}(h_1, h_2, h_3)$

Algorithm 11 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

if $\varepsilon \geq \varepsilon_{WL}$ **then**

 Retorne $AD(\varepsilon, \mathcal{D}')$

end if

$\beta \leftarrow g^{-1}(\varepsilon)$

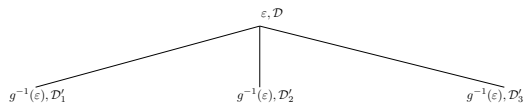
$h_1 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_1)$

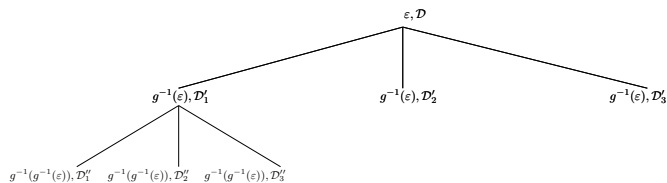
$h_2 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_2)$

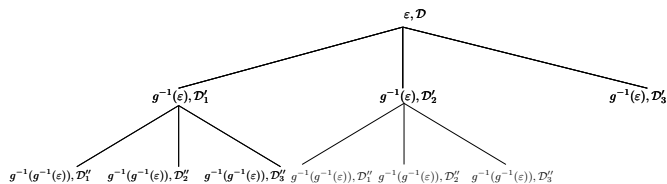
$h_3 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_3)$

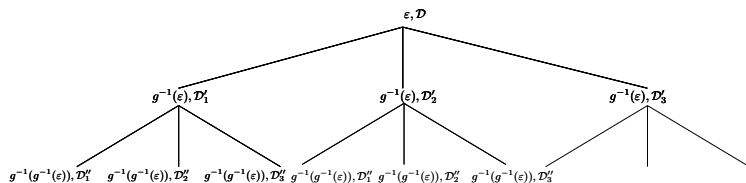
$h \leftarrow \text{mayoría}(h_1, h_2, h_3)$

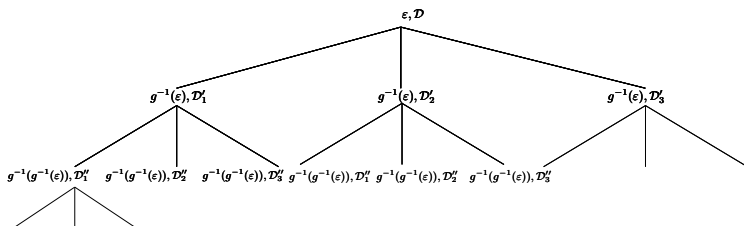
Retorne h

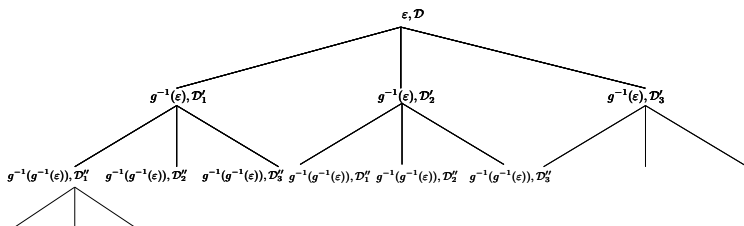




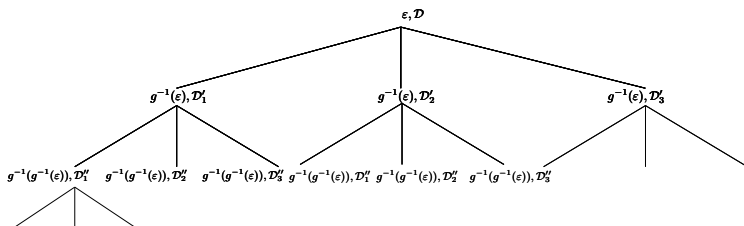




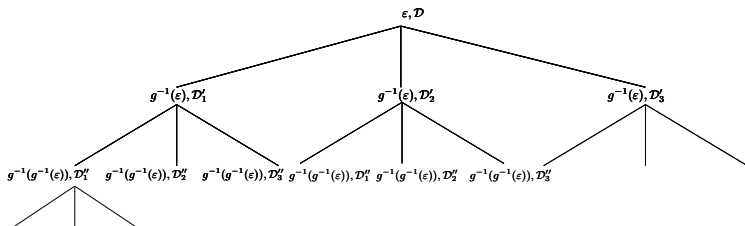




- Eficiencia?



- Eficiencia?
 - Profundidad del árbol de recursión?



- Eficiencia?

- ▶ Profundidad del árbol de recursión?
- ▶ Número de llamadas a $EX(c, \mathcal{D})$?

Algoritmo de Boosting en el modelo PAC

- Requiere oráculo.

Algoritmo de Boosting en el modelo PAC

- Requiere oráculo.
- Genera estructura no regular.

Algoritmo de Boosting en el modelo PAC

- Requiere oráculo.
- Genera estructura no regular.
- Requiere conocer **garantía de error del algoritmo débil**.

Algoritmo de Boosting en el modelo PAC

- Requiere oráculo.
- Genera estructura no regular.
- Requiere conocer **garantía de error del algoritmo débil**.
- No es práctico.

Algoritmo de Boosting en el modelo PAC

- Requiere oráculo.
- Genera estructura no regular.
- Requiere conocer **garantía de error del algoritmo débil**.
- No es práctico.

Boosting Adaptivo

- Datos $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathcal{X}$ y $y_i \in \{-1, 1\}$

Boosting Adaptivo

- Datos $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathcal{X}$ y $y_i \in \{-1, 1\}$
- Asociamos a los datos un **vector de pesos** $D = \{D_1, D_2, \dots, D_n\}$.

Boosting Adaptivo

- Datos $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathcal{X}$ y $y_i \in \{-1, 1\}$
- Asociamos a los datos un **vector de pesos** $D = \{D_1, D_2, \dots, D_n\}$.
- $\{D_1, D_2, \dots, D_n\}$ es una distribución, es decir

$$D_i \geq 0 \quad \text{y} \quad \sum_{i=1}^n D_i = 1$$

Boosting Adaptivo

- Datos $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathcal{X}$ y $y_i \in \{-1, 1\}$
- Asociamos a los datos un **vector de pesos** $D = \{D_1, D_2, \dots, D_n\}$.
- $\{D_1, D_2, \dots, D_n\}$ es una distribución, es decir

$$D_i \geq 0 \quad \text{y} \quad \sum_{i=1}^n D_i = 1$$

- Clase de hipótesis base: $h \in \mathcal{H}$, y $h : \mathcal{X} \longrightarrow \{-1, 1\}$

Boosting Adaptivo

- Datos $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, con $\mathbf{x}_i \in \mathcal{X}$ y $y_i \in \{-1, 1\}$
- Asociamos a los datos un **vector de pesos** $D = \{D_1, D_2, \dots, D_n\}$.
- $\{D_1, D_2, \dots, D_n\}$ es una distribución, es decir

$$D_i \geq 0 \quad \text{y} \quad \sum_{i=1}^n D_i = 1$$

- Clase de hipótesis base: $h \in \mathcal{H}$, y $h : \mathcal{X} \longrightarrow \{-1, 1\}$
- Error pesado de una hipótesis h de acuerdo a D :

$$e_D(h) = \sum_{i=1}^n D_i I_{\{y_i f(\mathbf{x}_i) \leq 0\}} = \sum_{i : h(\mathbf{x}_i) \neq y_i} D_i$$

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- AdaBoost procede en una serie de **rondas** $1, 2, \dots$, en las que obtiene hipótesis h_1, h_2, \dots .

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- AdaBoost procede en una serie de **rondas** $1, 2, \dots$, en las que obtiene hipótesis h_1, h_2, \dots .
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \dots, n$.

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- AdaBoost procede en una serie de **rondas** $1, 2, \dots$, en las que obtiene hipótesis h_1, h_2, \dots .
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \dots, n$.
- En la siguiente ronda se modifica D :

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- AdaBoost procede en una serie de **rondas** $1, 2, \dots$, en las que obtiene hipótesis h_1, h_2, \dots .
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \dots, n$.
- En la siguiente ronda se modifica D :

$$D_i \begin{cases} \text{aumenta} & \text{si } h_1(\mathbf{x}) \neq y_i, \\ \text{disminuye} & \text{si } h_1(\mathbf{x}) = y_i. \end{cases}$$

- Asumimos acceso a un **aprendiz débil** A , que recibe S y D y retorna $h \in \mathcal{H}$ con

$$e_D(h) < \frac{1}{2}$$

- AdaBoost procede en una serie de **rondas** $1, 2, \dots$, en las que obtiene hipótesis h_1, h_2, \dots .
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \dots, n$.
- En la siguiente ronda se modifica D :

$$D_i \begin{cases} \text{aumenta} & \text{si } h_1(\mathbf{x}) \neq y_i, \\ \text{disminuye} & \text{si } h_1(\mathbf{x}) = y_i. \end{cases}$$

- Se itera este procedimiento, modificando los pesos en cada ronda de acuerdo a la hipótesis de la ronda anterior.

AdaBoost

- Construir $f(\mathbf{x})$ para minimizar

$$e(f) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

AdaBoost

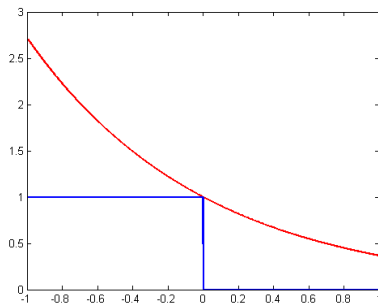
- Construir $f(\mathbf{x})$ para minimizar

$$e(f) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \geq \frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}}$$

AdaBoost

- Construir $f(\mathbf{x})$ para minimizar

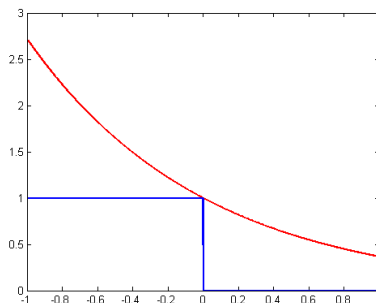
$$e(f) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \geq \frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}}$$



AdaBoost

- Construir $f(\mathbf{x})$ para minimizar

$$e(f) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \geq \frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}}$$



- Maximizar función de costo de los márgenes en los datos.

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k - 1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k - 1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k - 1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

$$e(f_k) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k - 1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

$$\begin{aligned} e(f_k) &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \\ &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i) - y_i \alpha_k h_k(\mathbf{x}_i)} \end{aligned}$$

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k-1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

$$\begin{aligned} e(f_k) &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \\ &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i) - y_i \alpha_k h_k(\mathbf{x}_i)} \\ &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} e^{-y_i \alpha_k h_k(\mathbf{x}_i)} \end{aligned}$$

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k-1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

$$\begin{aligned} e(f_k) &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \\ &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i) - y_i \alpha_k h_k(\mathbf{x}_i)} \\ &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} e^{-y_i \alpha_k h_k(\mathbf{x}_i)} \\ &= \frac{1}{n} \left(\sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} \right) \sum_{i=1}^n \left(\frac{e^{-y_i f_{k-1}(\mathbf{x}_i)}}{\sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)}} \right) e^{-y_i \alpha_k h_k(\mathbf{x}_i)} \end{aligned}$$

- Suponga que conocemos α_j, h_j para $j = 1, 2, \dots, k-1$, y queremos hallar α_k y h_k . Denote $f_k = \sum_{j=1}^k \alpha_j h_j$.

$$\begin{aligned}
 e(f_k) &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} \\
 &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i) - y_i \alpha_k h_k(\mathbf{x}_i)} \\
 &= \frac{1}{n} \sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} e^{-y_i \alpha_k h_k(\mathbf{x}_i)} \\
 &= \frac{1}{n} \left(\sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} \right) \sum_{i=1}^n \left(\frac{e^{-y_i f_{k-1}(\mathbf{x}_i)}}{\sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)}} \right) e^{-y_i \alpha_k h_k(\mathbf{x}_i)} \\
 &= \frac{1}{n} \left(\sum_{i=1}^n e^{-y_i f_{k-1}(\mathbf{x}_i)} \right) \sum_{i=1}^n D_i e^{-y_i \alpha_k h_k(\mathbf{x}_i)}
 \end{aligned}$$

- Queremos encontrar α, h que minimizan:

$$\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)}$$

- Queremos encontrar α, h que minimizan:

$$\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)}$$

- Queremos encontrar α, h que minimizan:

$$\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)} = \sum_{i: y_i \neq h(\mathbf{x}_i)} D_i e^{\alpha} + \sum_{i: y_i = h(\mathbf{x}_i)} D_i e^{-\alpha}$$

- Queremos encontrar α, h que minimizan:

$$\begin{aligned}\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)} &= \sum_{i: y_i \neq h(\mathbf{x}_i)} D_i e^{\alpha} + \sum_{i: y_i = h(\mathbf{x}_i)} D_i e^{-\alpha} \\ &= e_D(h) e^{\alpha} + (1 - e_D(h)) e^{-\alpha}\end{aligned}$$

- Queremos encontrar α, h que minimizan:

$$\begin{aligned}\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)} &= \sum_{i: y_i \neq h(\mathbf{x}_i)} D_i e^{\alpha} + \sum_{i: y_i = h(\mathbf{x}_i)} D_i e^{-\alpha} \\ &= e_D(h) e^{\alpha} + (1 - e_D(h)) e^{-\alpha}\end{aligned}$$

- $h = \arg \min_{g \in \mathcal{H}} e_D(g)$

- Queremos encontrar α, h que minimizan:

$$\begin{aligned}\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)} &= \sum_{i: y_i \neq h(\mathbf{x}_i)} D_i e^{\alpha} + \sum_{i: y_i = h(\mathbf{x}_i)} D_i e^{-\alpha} \\ &= e_D(h) e^{\alpha} + (1 - e_D(h)) e^{-\alpha}\end{aligned}$$

- $h = \arg \min_{g \in \mathcal{H}} e_D(g) \implies$ Aprendiziz débil

- Queremos encontrar α, h que minimizan:

$$\begin{aligned}\sum_{i=1}^n D_i e^{-y_i \alpha h(\mathbf{x}_i)} &= \sum_{i: y_i \neq h(\mathbf{x}_i)} D_i e^{\alpha} + \sum_{i: y_i = h(\mathbf{x}_i)} D_i e^{-\alpha} \\ &= e_D(h) e^{\alpha} + (1 - e_D(h)) e^{-\alpha}\end{aligned}$$

- $h = \arg \min_{g \in \mathcal{H}} e_D(g) \implies$ Aprendiziz débil
- Con h fija, encontramos α derivando e igualando a cero:

$$\alpha = \frac{1}{2} \ln \left(\frac{1 - e_D}{e_D} \right)$$

Algorithm 12 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

Algorithm 13 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ **to** T **do**

Algorithm 14 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Algorithm 15 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i:h_t(X_i) \neq y_i} D_t(i)$.

Algorithm 16 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i:h_t(X_i) \neq y_i} D_t(i)$.

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

Algorithm 17 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i: h_t(X_i) \neq y_i} D_t(i)$.

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

 Actualice D: $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

Algorithm 18 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i: h_t(X_i) \neq y_i} D_t(i)$.

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

 Actualice D : $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

 Donde Z_t normaliza D de manera que $\sum_{i=1}^t D_{t+1}(i) = 1$.

Algorithm 19 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i: h_t(X_i) \neq y_i} D_t(i)$.

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

 Actualice D : $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

 Donde Z_t normaliza D de manera que $\sum_{i=1}^t D_{t+1}(i) = 1$.

end for

Algorithm 20 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i: h_t(X_i) \neq y_i} D_t(i)$.

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

Actualice D : $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

Donde Z_t normaliza D de manera que $\sum_{i=1}^t D_{t+1}(i) = 1$.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x)$

Algorithm 21 AdaBoost

$D_1(i) = 1/n$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

$\epsilon_t = \sum_{i: h_t(X_i) \neq y_i} D_t(i)$.

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

Actualice D : $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

Donde Z_t normaliza D de manera que $\sum_{i=1}^t D_{t+1}(i) = 1$.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x)$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

$$e_{D_{t+1}}(h) = \sum_{i: h(\mathbf{x}_i) \neq y_i} D_{t+1}(i)$$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

$$\begin{aligned} e_{D_{t+1}}(h) &= \sum_{i: h(\mathbf{x}_i) \neq y_i} D_{t+1}(i) \\ &= \sum_{i: h(\mathbf{x}_i) \neq y_i} \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} \end{aligned}$$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

$$\begin{aligned} e_{D_{t+1}}(h) &= \sum_{i: h(\mathbf{x}_i) \neq y_i} D_{t+1}(i) \\ &= \sum_{i: h(\mathbf{x}_i) \neq y_i} \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} \\ &= \frac{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}} \end{aligned}$$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

$$\begin{aligned} e_{D_{t+1}}(h) &= \sum_{i: h(\mathbf{x}_i) \neq y_i} D_{t+1}(i) \\ &= \sum_{i: h(\mathbf{x}_i) \neq y_i} \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} \\ &= \frac{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}} \\ &= \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{\sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{\epsilon_t(1-\epsilon_t)}} \end{aligned}$$

- Note que

$$D_{t+1}(i) = \begin{cases} \frac{D_t(i) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}}{Z_t} & \text{si } y_i = h_t(\mathbf{x}_i) , \\ \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} & \text{si } y_i \neq h_t(\mathbf{x}_i) . \end{cases}$$

- El error pesado de h_t con respecto a D_{t+1} :

$$\begin{aligned} e_{D_{t+1}}(h) &= \sum_{i: h(\mathbf{x}_i) \neq y_i} D_{t+1}(i) \\ &= \sum_{i: h(\mathbf{x}_i) \neq y_i} \frac{D_t(i) \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{Z_t} \\ &= \frac{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{\epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t) \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}} \\ &= \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{\sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{\epsilon_t(1-\epsilon_t)}} = \frac{1}{2} \end{aligned}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t} = \frac{e^{-y_i f(\mathbf{x}_i)}}{n \prod_t Z_t}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t} = \frac{e^{-y_i f(\mathbf{x}_i)}}{n \prod_t Z_t}$$

luego:

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \prod_t Z_t$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t} = \frac{e^{-y_i f(\mathbf{x}_i)}}{n \prod_t Z_t}$$

luego:

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t} = \frac{e^{-y_i f(\mathbf{x}_i)}}{n \prod_t Z_t}$$

luego:

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

- Si $\epsilon_t < \frac{1}{2}$ el error empírico **decrece exponencialmente!**

Error empírico

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)}$$

y

$$D_{t+1}(i) = \frac{e^{-\sum_t \alpha_t y_i h_t(\mathbf{x}_i)}}{n \prod_t Z_t} = \frac{e^{-y_i f(\mathbf{x}_i)}}{n \prod_t Z_t}$$

luego:

$$\frac{1}{n} \sum_{i=1}^n I_{\{y_i f(\mathbf{x}_i) \leq 0\}} \leq \prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1 - \epsilon_t)}$$

- Si $\epsilon_t < \frac{1}{2}$ el error empírico **decrece exponencialmente!**
- Si $\epsilon_t < \frac{1}{2}$ el **error empírico llega a cero en un número finito de pasos.**

Consecuencias

- Sea $\mathcal{F} = \text{conv}(\mathcal{H}) = \left\{ f = \sum_{i=1}^T \alpha_i h_i : h_i \in \mathcal{H}, \alpha_i \geq 0, \sum_i \alpha_i = 1 \right\}$

Consecuencias

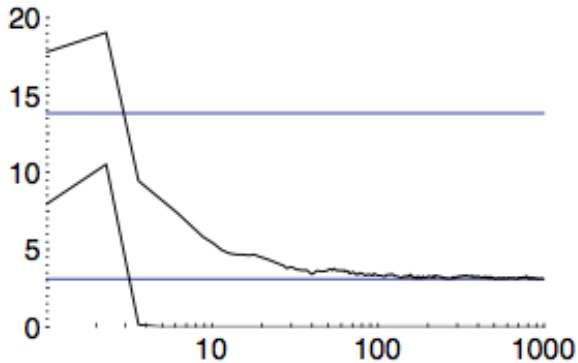
- Sea $\mathcal{F} = \text{conv}(\mathcal{H}) = \left\{ f = \sum_{i=1}^T \alpha_t h_t : h \in \mathcal{H}, \alpha_t \geq 0, \sum_t \alpha_t = 1 \right\}$
- $VC(\mathcal{F}) \leq 2(VC(\mathcal{H}) + 1)(T + 1)\log_2(e(T + 1))$

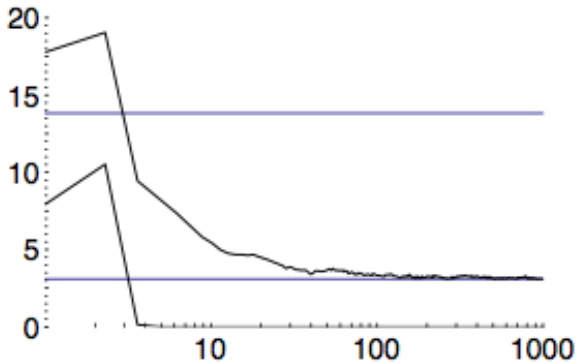
Consecuencias

- Sea $\mathcal{F} = \text{conv}(\mathcal{H}) = \left\{ f = \sum_{i=1}^T \alpha_t h_t : h \in \mathcal{H}, \alpha_t \geq 0, \sum_t \alpha_t = 1 \right\}$
- $VC(\mathcal{F}) \leq 2(VC(\mathcal{H}) + 1)(T + 1)\log_2(e(T + 1))$
- Si $VC(\mathcal{H}) < \infty$ entonces AdaBoost hace boosting en el modelo PAC.

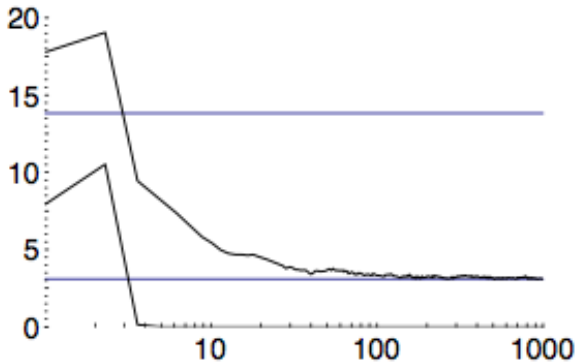
Consecuencias

- Sea $\mathcal{F} = \text{conv}(\mathcal{H}) = \left\{ f = \sum_{i=1}^T \alpha_t h_t : h \in \mathcal{H}, \alpha_t \geq 0, \sum_t \alpha_t = 1 \right\}$
- $VC(\mathcal{F}) \leq 2(VC(\mathcal{H}) + 1)(T + 1)\log_2(e(T + 1))$
- Si $VC(\mathcal{H}) < \infty$ entonces AdaBoost hace boosting en el modelo PAC.
- Sobreajuste de los datos de entrenamiento?

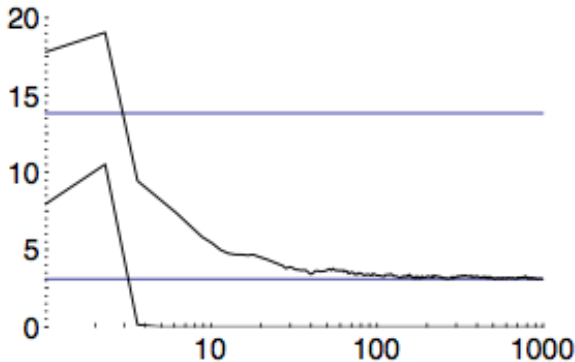




- Incluso cuando el error empírico es cero, error real disminuye.



- Incluso cuando el error empírico es cero, error real disminuye.
- Eventualmente puede producir sobreajuste para $T \gg \gg$.



- Incluso cuando el error empírico es cero, error real disminuye.
- Eventualmente puede producir sobreajuste para $T \gg \gg$.
- Sobreajuste con datos ruidosos.

Márgenes

Márgenes

- El **márgen** de f en (\mathbf{x}_i, y_i) es $m_i = y_i f(\mathbf{x}_i)$.

Márgenes

- El **márgen** de f en (\mathbf{x}_i, y_i) es $m_i = y_i f(\mathbf{x}_i)$.
- (\mathbf{x}_i, y_i) es incorrectamente clasificado si $m_i < 0$.

Márgenes

- El **márgen** de f en (\mathbf{x}_i, y_i) es $m_i = y_i f(\mathbf{x}_i)$.
- (\mathbf{x}_i, y_i) es incorrectamente clasificado si $m_i < 0$.
- Si $m_i > 0$, podemos interpretar $|m_i|$ como una **medida de confianza**.

Márgenes

- El **márgen** de f en (\mathbf{x}_i, y_i) es $m_i = y_i f(\mathbf{x}_i)$.
- (\mathbf{x}_i, y_i) es incorrectamente clasificado si $m_i < 0$.
- Si $m_i > 0$, podemos interpretar $|m_i|$ como una **medida de confianza**.
- AdaBoost intenta minimizar una **función de costo del márgen**:

$$\phi(m_1, \dots, m_n) = \frac{1}{n} \sum_{i=1}^n e^{-y_i f(\mathbf{x}_i)} = \frac{1}{n} \sum_{i=1}^n e^{-m_i}$$

Generalización de AdaBoost

Theorem (Schapire, Freund, Bartlett y Lee, 1998)

$\forall \alpha \in (0, 1)$ with probability at least $1 - \alpha$ for all $f \in \text{conv}(\mathcal{H})$ the following inequality holds:

$$P\{(x, y) : yf(x) \leq 0\} \leq \inf_{\delta} \left[P_n\{(x, y) : yf(x) \leq \delta\} + \frac{C}{\sqrt{n}} \left(\frac{V(\mathcal{H}) \log^2(\frac{n}{V(\mathcal{H})})}{\delta^2} + \log(1/\alpha) \right)^{1/2} \right].$$

Generalización de AdaBoost

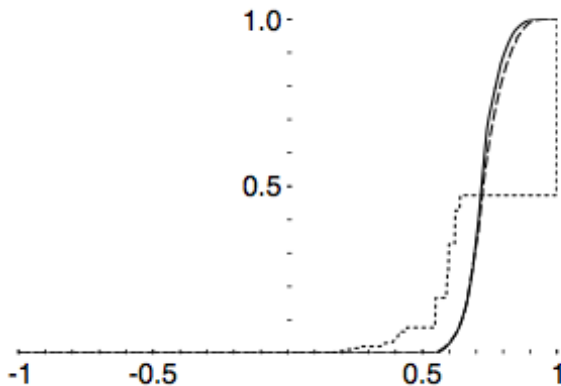
Theorem (Schapire, Freund, Bartlett y Lee, 1998)

$\forall \alpha \in (0, 1)$ with probability at least $1 - \alpha$ for all $f \in \text{conv}(\mathcal{H})$ the following inequality holds:

$$P\{(x, y) : yf(x) \leq 0\} \leq \inf_{\delta} \left[P_n\{(x, y) : yf(x) \leq \delta\} + \frac{C}{\sqrt{n}} \left(\frac{V(\mathcal{H}) \log^2(\frac{n}{V(\mathcal{H})})}{\delta^2} + \log(1/\alpha) \right)^{1/2} \right].$$

- Un clasificador combinado con **márgenes grandes** puede tener **probabilidad de error** pequeña.

Efecto de Adaboost en los márgenes



Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.
 - ▶ Meta de aprendizaje: minimizar $\mathbf{P}_{\mathcal{D}} [h(x) \neq y]$.

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.
 - ▶ Meta de aprendizaje: minimizar $\mathbf{P}_{\mathcal{D}} [h(x) \neq y]$.
- Multiclase y multietiqueta:

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.
 - ▶ Meta de aprendizaje: minimizar $\mathbf{P}_{\mathcal{D}} [h(x) \neq y]$.
- Multiclase y multietiqueta:
 - ▶ $(x, Y), Y \subseteq \mathcal{Y}$.

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.
 - ▶ Meta de aprendizaje: minimizar $\mathbf{P}_{\mathcal{D}}[h(x) \neq y]$.
- Multiclase y multietiqueta:
 - ▶ $(x, Y), Y \subseteq \mathcal{Y}$.
 - ▶ Meta?

Problemas multiclase y multietiqueta

- \mathcal{Y} : posibles etiquetas.
- $|\mathcal{Y}| = k \geq 2$.
- Multiclase:
 - ▶ $(x, y), y \in \mathcal{Y}$.
 - ▶ Meta de aprendizaje: minimizar $\mathbf{P}_{\mathcal{D}} [h(x) \neq y]$.
- Multiclase y multietiqueta:
 - ▶ $(x, Y), Y \subseteq \mathcal{Y}$.
 - ▶ Meta? depende del problema.

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$
- Función de pérdida de Hamming:

$$\text{hloss}_{\mathcal{D}}(h) = \frac{1}{k} \mathbf{E}_{\mathcal{D}} [|h(x) \triangle Y|]$$

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$
- Función de pérdida de Hamming:

$$\text{hloss}_{\mathcal{D}}(h) = \frac{1}{k} \mathbf{E}_{\mathcal{D}} [|h(x) \triangle Y|]$$

- Promedio del error en k problemas binarios.

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$
- Función de pérdida de Hamming:

$$\text{hloss}_{\mathcal{D}}(h) = \frac{1}{k} \mathbf{E}_{\mathcal{D}} [|h(x) \triangle Y|]$$

- Promedio del error en k problemas binarios.
- Para $Y \subseteq \mathcal{Y}$ definimos:

$$Y[l] = \begin{cases} +1 & \text{si } l \in Y \\ -1 & \text{si } l \notin Y \end{cases}$$

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$
- Función de pérdida de Hamming:

$$\text{hloss}_{\mathcal{D}}(h) = \frac{1}{k} \mathbf{E}_{\mathcal{D}} [|h(x) \triangle Y|]$$

- Promedio del error en k problemas binarios.
- Para $Y \subseteq \mathcal{Y}$ definimos:

$$Y[l] = \begin{cases} +1 & \text{si } l \in Y \\ -1 & \text{si } l \notin Y \end{cases}$$

- Identificamos función $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ con $h : \mathcal{X} \times \mathcal{Y} \rightarrow \{-1, +1\}$, con $h(x, l) = h(x)[l]$.

Distancia de Hamming

- Hipótesis $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$
- Función de pérdida de Hamming:

$$\text{hloss}_{\mathcal{D}}(h) = \frac{1}{k} \mathbf{E}_{\mathcal{D}} [|h(x) \triangle Y|]$$

- Promedio del error en k problemas binarios.
- Para $Y \subseteq \mathcal{Y}$ definimos:

$$Y[l] = \begin{cases} +1 & \text{si } l \in Y \\ -1 & \text{si } l \notin Y \end{cases}$$

- Identificamos función $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ con $h : \mathcal{X} \times \mathcal{Y} \rightarrow \{-1, +1\}$, con $h(x, l) = h(x)[l]$.
- Dato $(x_i, Y_i) \longrightarrow k$ datos $((x_i, l), Y_i[l])$

Algorithm 22 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

Algorithm 23 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ **to** T **do**

Algorithm 24 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Algorithm 25 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Algorithm 26 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Actualice D : $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$

Algorithm 27 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Actualice D : $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

Algorithm 28 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Actualice D : $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Algorithm 29 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Actualice D : $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x, l)$

Algorithm 30 AdaBoost.MH

$D_1(i, l) = 1/(nk)$ para $i = 1 \dots n$.

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t)$.

Escoja α_t

Actualice D : $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x, l)$

Clasificación multiclase usando ranking

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.
- Para un dato (x, Y) consideramos pares de etiquetas **cruciales**:
 $l_1, l_2 : l_1 \notin Y, l_2 \in Y$

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.
- Para un dato (x, Y) consideramos pares de etiquetas **cruciales**:
 $l_1, l_2 : l_1 \notin Y, l_2 \in Y$
- f **desordena** (l_1, l_2) si $f(x, l_1) \geq f(x, l_2)$.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.
- Para un dato (x, Y) consideramos pares de etiquetas **cruciales**:
 $l_1, l_2 : l_1 \notin Y, l_2 \in Y$
- f **desordena** (l_1, l_2) si $f(x, l_1) \geq f(x, l_2)$.
- Meta:

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.
- Para un dato (x, Y) consideramos pares de etiquetas **cruciales**:
 $l_1, l_2 : l_1 \notin Y, l_2 \in Y$
- f **desordena** (l_1, l_2) si $f(x, l_1) \geq f(x, l_2)$.
- Meta: Encontrar f con **pocos pares cruciales desordenados**.

Clasificación multiclase usando ranking

- Hipótesis asigna **ranking** a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.
- Para un x dado la etiqueta l_1 tiene un ranking más alto que la etiqueta l_2 si $f(x, l_1) > f(x, l_2)$.
- Para un dato (x, Y) consideramos pares de etiquetas **cruciales**:
 $l_1, l_2 : l_1 \notin Y, l_2 \in Y$
- f **desordena** (l_1, l_2) si $f(x, l_1) \geq f(x, l_2)$.
- Meta: Encontrar f con **pocos pares cruciales desordenados**.

$$\text{rloss}_{\mathcal{D}}(f) = \mathbf{E}_{\mathcal{D}} \left[\frac{|\{(l_1, l_2) \in (\mathcal{Y} - Y) \times Y : f(x, l_1) \geq f(x, l_2)\}|}{|Y| |\mathcal{Y} - Y|} \right]$$

Algorithm 31 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

Algorithm 32 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ **to** T **do**

Algorithm 33 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$$h_t \leftarrow A(S, D_t).$$

Algorithm 34 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Algorithm 35 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Actualice D : $D_{t+1}(i, l_1, l_2) = \frac{D_t(i, l_1, l_2) \exp(\frac{1}{2}\alpha_t(h_t(x_i, l_1) - h_t(x_i, l_2)))}{Z_t}$

Algorithm 36 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Actualice D : $D_{t+1}(i, l_1, l_2) = \frac{D_t(i, l_1, l_2) \exp(\frac{1}{2}\alpha_t(h_t(x_i, l_1) - h_t(x_i, l_2)))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

Algorithm 37 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Actualice D : $D_{t+1}(i, l_1, l_2) = \frac{D_t(i, l_1, l_2) \exp(\frac{1}{2}\alpha_t(h_t(x_i, l_1) - h_t(x_i, l_2)))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Algorithm 38 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Actualice D : $D_{t+1}(i, l_1, l_2) = \frac{D_t(i, l_1, l_2) \exp(\frac{1}{2}\alpha_t(h_t(x_i, l_1) - h_t(x_i, l_2)))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x, l)$

Algorithm 39 AdaBoost.MR

$$D_1(i, l_1, l_2) = \begin{cases} \frac{1}{n|Y_i||\mathcal{Y}-Y_i|} & \text{si } l_1 \notin Y_i, l_2 \in Y_i \\ 0 & \text{en otro caso} \end{cases}$$

for $t = 1$ to T **do**

$h_t \leftarrow A(S, D_t).$

Escoja α_t

Actualice D : $D_{t+1}(i, l_1, l_2) = \frac{D_t(i, l_1, l_2) \exp(\frac{1}{2}\alpha_t(h_t(x_i, l_1) - h_t(x_i, l_2)))}{Z_t}$

Donde Z_t normaliza D de manera que sea una distribución.

end for

Retorne $f(x) = \sum_{i=1}^T \alpha_t h_t(x, l)$
