Clasificadores combinados

Fernando Lozano

Universidad de los Andes

9 de noviembre de 2022

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 $h_1, h_2, \ldots, h_T$ minimizando error en diferentes versiones de los datos.

Clasificadores combinados

- Construir un clasificador combinando múltiples clasificadores.
- Clasificador combinado:
  - Obtener clasificadores $h_1, h_2, \ldots, h_T$ 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 $h_1, h_2, \ldots, h_T$ 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 $h_1, h_2, \ldots, h_T$ 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 (Boostrap Aggregating)

Bagging (Boostrap Aggregating)

- Bootstrapping (Efron, 1979):

Bagging (Boostrap Aggregating)

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

Bagging (Boostrap Aggregating)

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

Bagging (Boostrap Aggregating)

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

Bagging (Boostrap Aggregating)

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

Bagging (Boostrap Aggregating)

- Bootstrapping (Efron, 1979):
  - Técnica para mejorar precisión de parámetros/estimativos.
  - Muestreo con remplazo.
  - Muestra$\sim$ población.
- Clasificadores $h_1, h_2, \ldots$ 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

- Aprendibilidad PAC: encontrar h que satisfaga:

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

para cualquier distribución de los datos.

Boosting

- Aprendibilidad PAC: encontrar h que satisfaga:

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

para cualquier distribución de los datos.

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

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

para cualquier distribución de los datos.

Boosting

- Aprendibilidad PAC: encontrar h que satisfaga:

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

  para cualquier distribución de los datos.
- Suponga que podemos encontrar h_d, que satisface la condición de aprendibilidad débil

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

  para cualquier distribución de los datos.
- Es posible obtener aprendibilidad fuerte a partir de aprendibilidad débil?

Boosting

- Aprendibilidad PAC: encontrar h que satisfaga:

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

  para cualquier distribución de los datos.
- Suponga que podemos encontrar h_d, que satisface la condición de aprendibilidad débil

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

  para cualquier distribución de los datos.
- Es posible obtener aprendibilidad fuerte a partir de aprendibilidad 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 $\mathbf{P}_{\mathcal{D}_2}\left[h_1(x) \neq c(x)\right] = \frac{1}{2}$
    3. $h_3 \leftarrow A(\mathcal{D}_3)$ donde $\mathcal{D}_3$ es tal que $\mathbf{P}_{\mathcal{D}_3}\left[h_1(x) \neq h_2(x)\right] = 1$

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 $\mathbf{P}_{\mathcal{D}_2}\left[h_1(x) \neq c(x)\right] = \frac{1}{2}$
  3. $h_3 \leftarrow A(\mathcal{D}_3)$ donde $\mathcal{D}_3$ es tal que $\mathbf{P}_{\mathcal{D}_3}\left[h_1(x) \neq h_2(x)\right] = 1$

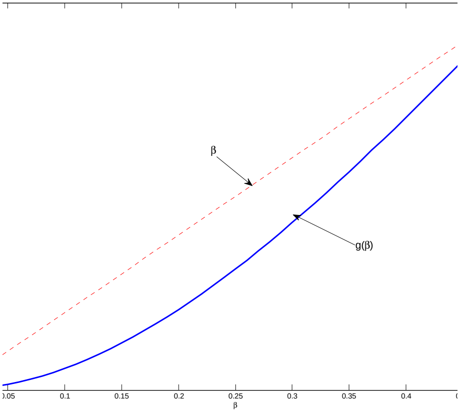
  Retorna $h(x) = \text{mayoría}\,(h_1(x), h_2(x), h_3(x))$

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 $\mathbf{P}_{\mathcal{D}_2}\left[h_1(x) \neq c(x)\right] = \frac{1}{2}$
  3. $h_3 \leftarrow A(\mathcal{D}_3)$ donde $\mathcal{D}_3$ es tal que $\mathbf{P}_{\mathcal{D}_3}\left[h_1(x) \neq h_2(x)\right] = 1$

  Retorna $h(x) = \text{mayoría}\,(h_1(x), h_2(x), h_3(x))$
- Se puede mostrar

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


β
g(β)
).05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
β

- 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

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

$h_1 \leftarrow$ 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

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

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

$h_2 \leftarrow$ 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
$\beta \leftarrow g^{-1}(\varepsilon)$
$h_1 \leftarrow$ BoostingFuerte$(\beta, \mathcal{D}'_1)$
$h_2 \leftarrow$ BoostingFuerte$(\beta, \mathcal{D}'_2)$
$h_3 \leftarrow$ 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

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

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

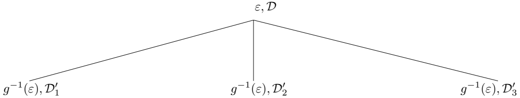
$h_2 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_2)$

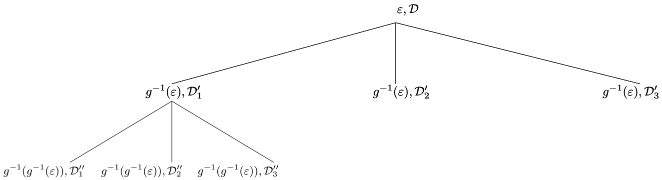
$h_3 \leftarrow \text{BoostingFuerte}(\beta, \mathcal{D}'_3)$

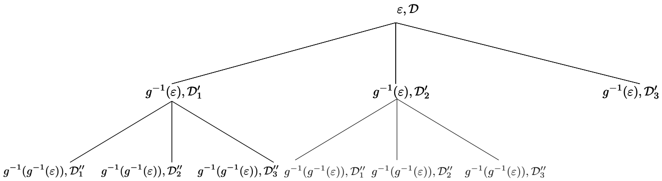
$h \leftarrow \text{mayoría}(h_1, h_2, h_3)$

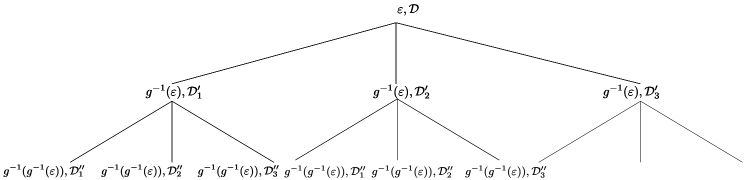
Algorithm 11 Algoritmo BoostingFuerte($\varepsilon, \mathcal{D}'$)

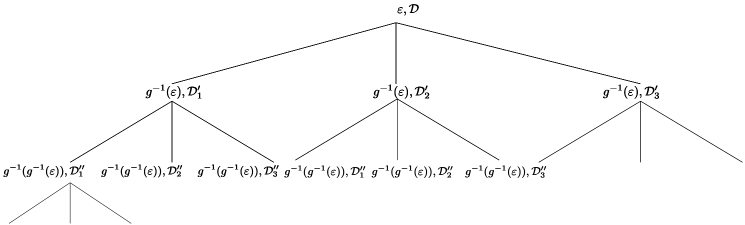
if $\varepsilon \geq \varepsilon_{WL}$ **then**
  Retorne $AD(\varepsilon, \mathcal{D}')$
end if
$\beta \leftarrow g^{-1}(\varepsilon)$
$h_1 \leftarrow$ BoostingFuerte($\beta, \mathcal{D}'_1$)
$h_2 \leftarrow$ BoostingFuerte($\beta, \mathcal{D}'_2$)
$h_3 \leftarrow$ BoostingFuerte($\beta, \mathcal{D}'_3$)
$h \leftarrow$ mayoría(h_1, h_2, h_3)
Retorne h

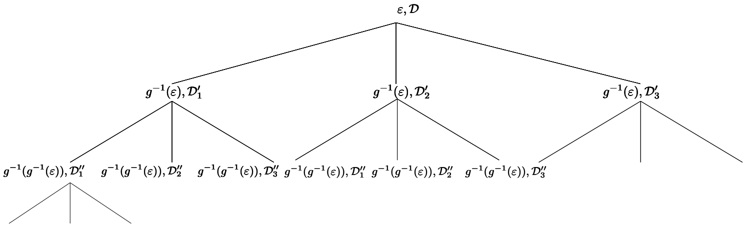

$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$


$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$

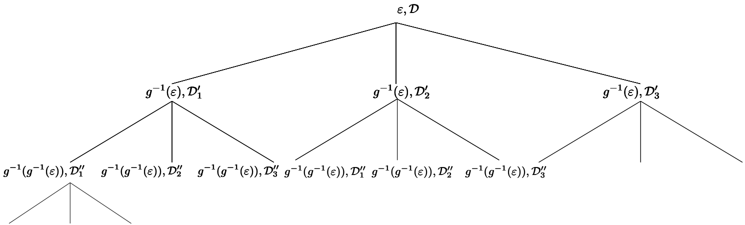

$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$


$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$

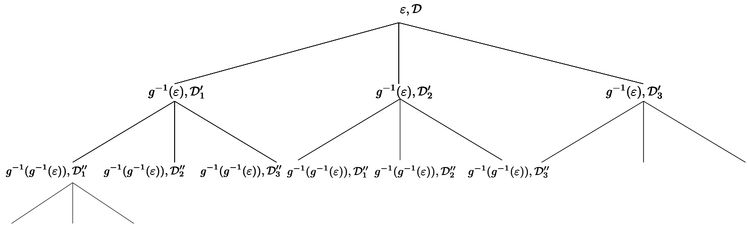

$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$


$\varepsilon, \mathcal{D}$
$g^{-1}(\varepsilon), \mathcal{D}'_1$
$g^{-1}(\varepsilon), \mathcal{D}'_2$
$g^{-1}(\varepsilon), \mathcal{D}'_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_1$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_2$
$g^{-1}(g^{-1}(\varepsilon)), \mathcal{D}''_3$

- 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

- Requierer oráculo.

Algoritmo de Boosting en el modelo PAC

- Requierer oráculo.
- Genera estructura no regular.

Algoritmo de Boosting en el modelo PAC

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

Algoritmo de Boosting en el modelo PAC

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

Algoritmo de Boosting en el modelo PAC

- Requierer 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 $D = \{D_1, D_2, \ldots, D_n\}$.

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 $D = \{D_1, D_2, \ldots, D_n\}$.
- $\{D_1, D_2, \ldots, D_n\}$ 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 $D = \{D_1, D_2, \ldots, D_n\}$.
- $\{D_1, D_2, \ldots, D_n\}$ 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 $D = \{D_1, D_2, \ldots, D_n\}$.
- $\{D_1, D_2, \ldots, D_n\}$ 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 $1, 2, \ldots,$ en las que obtiene hipótesis $h_1, h_2, \ldots$.

- 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 $1, 2, \ldots,$ en las que obtiene hipótesis $h_1, h_2, \ldots$.
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \ldots, n$.

- 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 $1, 2, \ldots,$ en las que obtiene hipótesis $h_1, h_2, \ldots$.
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \ldots, n$.
- 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 $1, 2, \ldots,$ en las que obtiene hipótesis $h_1, h_2, \ldots$.
- En la primera ronda se llama A con la distribución uniforme $D_i = \frac{1}{n}$, $i = 1, 2, \ldots, n$.
- 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

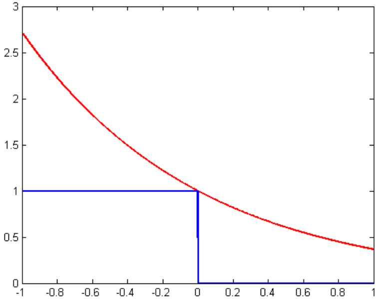
- Construir $f(\mathbf{x})$ para minimizar

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

AdaBoost

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

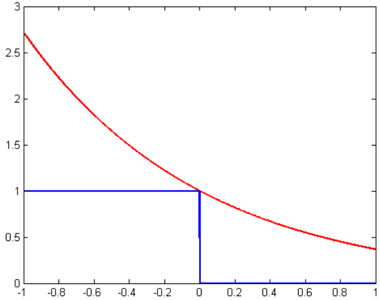
$$e(f) = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)} \geq \frac{1}{n}\sum_{i=1}^{n} I_{\{y_i f(\mathbf{x}_i)\leq 0\}}$$



AdaBoost

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

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



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

- Suponga que conocemos α_j, h_j para $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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 $j = 1, 2, \ldots, k-1$, 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:

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

- Queremos encontrar α, h que minimizan:

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

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

- Queremos encontrar α, h que minimizan:

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

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

- Queremos encontrar α, h que minimizan:

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

- $h = \arg\min_{g \in \mathcal{H}} e_D(g) \Longrightarrow$ Aprendiz 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 $i = 1 \ldots n$.

Algorithm 13 AdaBoost

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

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

Algorithm 14 AdaBoost

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

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

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

Algorithm 15 AdaBoost

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

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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

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

- 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

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

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

Consecuencias

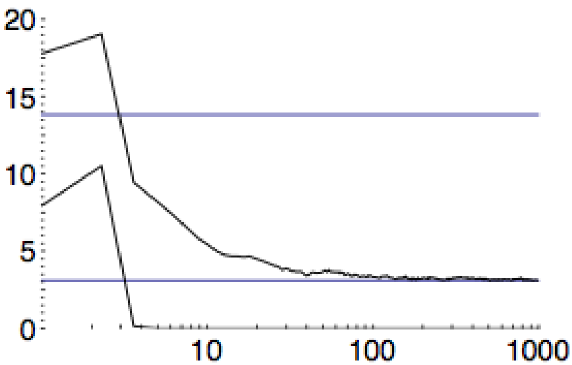
- Sea $\mathcal{F} = \text{conv}(\mathcal{H}) = \left\{ f = \sum_{i=1}^{T} \alpha_t h_t \ : \ h \in \mathcal{H},\ \alpha_t \geq 0, \sum_t \alpha_t = 1 \right\}$
- $VC(\mathcal{F}) \leq 2(VC(\mathcal{H}) + 1)(T + 1) log_2(e(T + 1))$

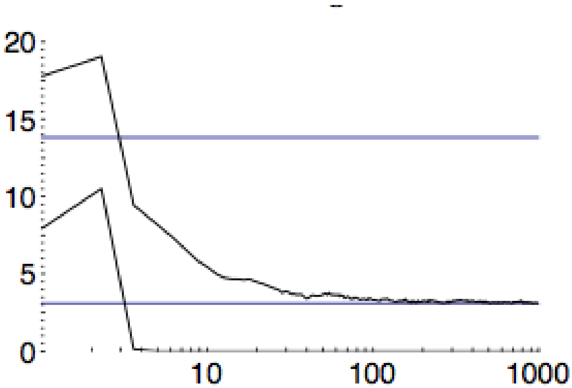
Consecuencias

- 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.

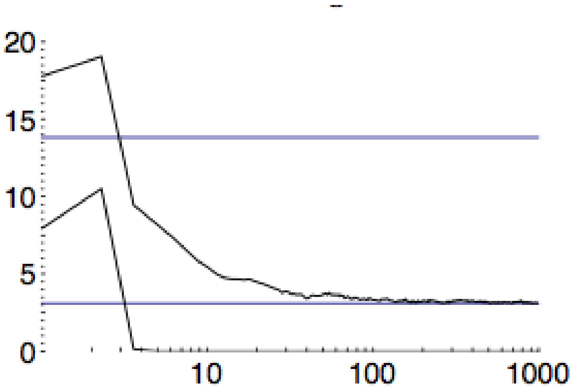
Consecuencias

- 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?

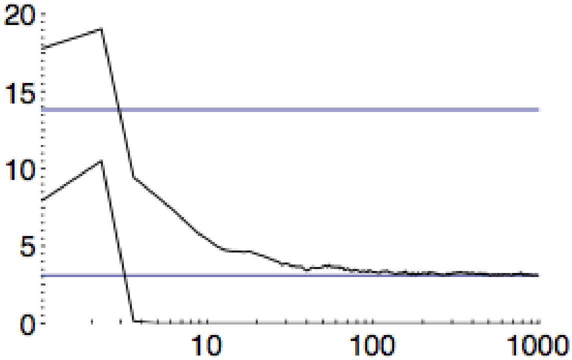

20
15
10
5
0
10
100
1000



- 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 $T \ggg$.



- Incluso cuando el error empírico es cero, error real disminuye.
- Eventualmente puede producir sobreajuste para $T \ggg$.
- 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 interprentar $|m_i|$ 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 interprentar $|m_i|$ como una medida de confianza.
- AdaBoost intenta minimizar una función de costo del márgen:

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

Generalización de AdaBoost

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

$\forall \alpha \in (0,1)$ *with probability at least* $1-\alpha$ *for all* $f \in \mathrm{conv}(\mathcal{H})$ *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

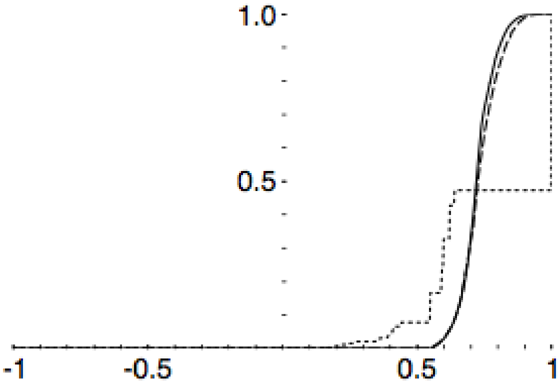
Theorem (Schapire, Freund, Bartlett y Lee, 1998)

$\forall \alpha \in (0,1)$ *with probability at least* $1-\alpha$ *for all* $f \in \mathrm{conv}(\mathcal{H})$ *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 $\mathbf{P}_{\mathcal{D}}\left[h(x) \neq y\right]$.

Problemas multiclase y multietiqueta

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

Distancia de Hamming

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

Distancia de Hamming

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

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

Distancia de Hamming

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

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

- Promedio del error en k problemas binarios.

Distancia de Hamming

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

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

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

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

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

Distancia de Hamming

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

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

- 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 $h : \mathcal{X} \to 2^{\mathcal{Y}}$ con $h : \mathcal{X} \times \mathcal{Y} \to \{-1, +1\}$, 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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $i = 1 \ldots n$.

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 $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Clasificación multiclase usando ranking

- Hipótesis asigna ranking a las etiquetas.
- Queremos que etiquetas correctas reciban ranking más alto.
- Hipótesis $f\ :\ \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- 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 $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- 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 $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- 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 $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- 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 $f\ :\ \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- 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: $D_{t+1}(i,l_1,l_2) = \frac{D_t(i,l_1,l_2)\exp\left(\frac{1}{2}\alpha_t(h_t(x_i,l_1)-h_t(x_i,l_2))\right)}{Z_t}$

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

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

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

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

Donde Z_t normaliza D de manera que sea una distribución.

end for

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

Gradient Boosting

- Clase de funciones $\mathcal{F}$.

Gradient Boosting

- Clase de funciones $\mathcal{F}$.
- Espacio de combinaciones lineales $\mathrm{span}(\mathcal{F})$

Gradient Boosting

- Clase de funciones $\mathcal{F}$.
- Espacio de combinaciones lineales $\mathrm{span}(\mathcal{F})$
- Producto punto en $\mathrm{span}(\mathcal{F})$.

Gradient Boosting

- Clase de funciones $\mathcal{F}$.
- Espacio de combinaciones lineales $\text{span}(\mathcal{F})$
- Producto punto en $\text{span}(\mathcal{F})$.
- Funcional de costo $C(F) : \text{span}(\mathcal{F}) \to \mathbb{R}$ a minimizar.

Gradient Boosting

- Clase de funciones $\mathcal{F}$.
- Espacio de combinaciones lineales $\mathrm{span}(\mathcal{F})$
- Producto punto en $\mathrm{span}(\mathcal{F})$.
- Funcional de costo $C(F) : \mathrm{span}(\mathcal{F}) \rightarrow \mathbb{R}$ a minimizar.
- Dada $F \in \mathrm{span}(\mathcal{F})$, encontrar $f \in \mathcal{F}$ de tal manera que $C(F + \alpha f) < C(F)$.

Gradient Boosting

- Clase de funciones $\mathcal{F}$.
- Espacio de combinaciones lineales $\text{span}(\mathcal{F})$
- Producto punto en $\text{span}(\mathcal{F})$.
- Funcional de costo $C(F) : \text{span}(\mathcal{F}) \to \mathbb{R}$ a minimizar.
- Dada $F \in \text{span}(\mathcal{F})$, encontrar $f \in \mathcal{F}$ de tal manera que $C(F + \alpha f) < C(F)$.
- Buscar en dirección f que maximiza $-\langle \nabla C(F), f \rangle$, donde

$$\nabla C(F)(\mathbf{x}) = \left. \frac{\partial C(F + \delta \mathbf{1_x})}{\partial \delta} \right|_{\delta=0}$$

Algorithm 1 : AnyBoost

Require :
- An inner product space $(\mathcal{X}, \langle , \rangle)$ containing functions mapping from X to some set Y.
- A class of base classifiers $\mathcal{F} \subseteq \mathcal{X}$.
- A differentiable cost functional $C\colon \operatorname{lin}(\mathcal{F}) \to \mathbb{R}$.
- A weak learner $\mathcal{L}(F)$ that accepts $F \in \operatorname{lin}(\mathcal{F})$ and returns $f \in \mathcal{F}$ with a large value of $-\langle \nabla C(F), f \rangle$.

Let $F_0(x) := 0$.
for $t := 0$ to T **do**
  Let $f_{t+1} := \mathcal{L}(F_t)$.
  if $-\langle \nabla C(F_t), f_{t+1} \rangle \leq 0$ **then**
    return F_t.
  end if
  Choose w_{t+1}.
  Let $F_{t+1} := F_t + w_{t+1} f_{t+1}$
end for
return F_{T+1}.

Clasificación

- Clasificar con $\text{sign}(F)$.

Clasificación

- Clasificar con $\text{sign}(F)$.
- Producto punto:

$$\langle F, G \rangle = \frac{1}{n} \sum_{i=1}^{n} F(\mathbf{x}_i) G(\mathbf{x}_i)$$

Clasificación

- Clasificar con $\mathrm{sign}(F)$.
- Producto punto:

$$\langle F, G \rangle = \frac{1}{n} \sum_{i=1}^{n} F(\mathbf{x}_i) G(\mathbf{x}_i)$$

- Función de costo:

$$C(F) = \frac{1}{n} \sum_{i=1}^{n} c(y_i F(\mathbf{x}_i))$$

Clasificación

- Clasificar con $\text{sign}(F)$.
- Producto punto:

$$\langle F, G \rangle = \frac{1}{n} \sum_{i=1}^{n} F(\mathbf{x}_i) G(\mathbf{x}_i)$$

- Función de costo:

$$C(F) = \frac{1}{n} \sum_{i=1}^{n} c(y_i F(\mathbf{x}_i))$$

donde $c : \mathbb{R} \to \mathbb{R}$ es función (diferenciadle, decreciente) de costo del márgen $yF(\mathbf{x})$

Clasificación

- Clasificar con $\text{sign}(F)$.
- Producto punto:

$$\langle F, G \rangle = \frac{1}{n} \sum_{i=1}^{n} F(\mathbf{x}_i) G(\mathbf{x}_i)$$

- Función de costo:

$$C(F) = \frac{1}{n} \sum_{i=1}^{n} c(y_i F(\mathbf{x}_i))$$

  donde $c : \mathbb{R} \to \mathbb{R}$ es función (diferenciadle, decreciente) de costo del márgen $yF(\mathbf{x})$
- Tenemos:

Clasificación

- Clasificar con $\operatorname{sign}(F)$.
- Producto punto:

$$\langle F, G\rangle = \frac{1}{n}\sum_{i=1}^{n} F(\mathbf{x}_i)G(\mathbf{x}_i)$$

- Función de costo:

$$C(F) = \frac{1}{n}\sum_{i=1}^{n} c(y_i F(\mathbf{x}_i))$$

  donde $c : \mathbb{R} \to \mathbb{R}$ es función (diferenciadle, decreciente) de costo del márgen $yF(\mathbf{x})$
- Tenemos:

$$-\langle \nabla C(F), f\rangle = -\frac{1}{n^2}\sum_{i=1}^{n} y_i f(\mathbf{x}_i) c'(y_i F(\mathbf{x}_i))$$

- Escoger f que minimice ell error pesado:

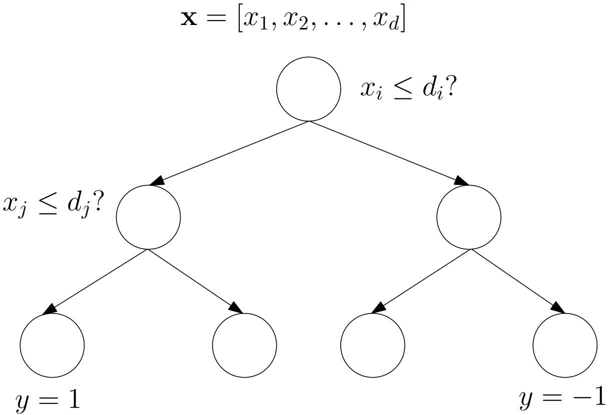
$$\sum_{i\,:\,y_i f(\mathbf{x}_i)<0} D_i, \quad \text{donde} \quad D_i = \frac{c'(y_i F(x_i))}{\sum_{i=1}^{n} c'(y_i F(x_i))}, \;\; i = 1, \ldots, n$$

- Escoger f que minimice ell error pesado:

$$\sum_{i\,:\,y_i f(\mathbf{x}_i)<0} D_i, \quad \text{donde} \quad D_i = \frac{c'(y_i F(x_i))}{\sum_{i=1}^{n} c'(y_i F(x_i))}, \quad i = 1, \ldots, n$$

Algorithm	Cost function	Step size
AdaBoost [9]	$e^{-yF(x)}$	Line search
ARC-X4 [2]	$(1 - yF(x))^5$	$1/t$
ConfidenceBoost [19]	$e^{-yF(x)}$	Line search
LogitBoost [12]	$\ln(1 + e^{-yF(x)})$	Newton-Raphson

Arboles de decisión



Entrenamiento

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^n$.

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
- Seleccionar x_j, d_j de manera greedy.

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
- Seleccionar x_j, d_j de manera greedy.
- Maximizar Ganancia de información:
- Minimizar impureza de los nodos resultantes:

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
- Seleccionar x_j, d_j de manera greedy.
- Maximizar Ganancia de información:
- Minimizar impureza de los nodos resultantes:

$$P_N = p_1(1 - p_1) + p_2(1 - p_2)$$

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
- Seleccionar x_j, d_j de manera greedy.
- Maximizar Ganancia de información:
- Minimizar impureza de los nodos resultantes:

$$P_N = p_1(1-p_1) + p_2(1-p_2) = 1 - p_1^2 - p_2^2$$

Entrenamiento

- Datos $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$.
- Seleccionar x_j, d_j de manera greedy.
- Maximizar Ganancia de información:
- Minimizar impureza de los nodos resultantes:

$$P_N = p_1(1-p_1) + p_2(1-p_2) = 1 - p_1^2 - p_2^2$$

- Indice de Gini:

$$G = P_{N_1} \times \frac{N_1}{N} + P_{N_2} \times \frac{N_2}{N}$$

XgBoost

- Arboles (clasificación regresión, ranking).

XgBoost

- Arboles (clasificación regresión, ranking).
- Dirección de Newton (aproximación cuadrática de $C(F)$).

XgBoost

- Arboles (clasificación regresión, ranking).
- Dirección de Newton (aproximación cuadrática de $C(F)$.
- Introduce regularización (penaliza árboles con hojas muy puras).

XgBoost

- Arboles (clasificación regresión, ranking).
- Dirección de Newton (aproximación cuadrática de $C(F)$.
- Introduce regularización (penaliza árboles con hojas muy puras).
- Algoritmo eficiente para encontrar mejor split (aproximado).

XgBoost

- Arboles (clasificación regresión, ranking).
- Dirección de Newton (aproximación cuadrática de $C(F)$).
- Introduce regularización (penaliza árboles con hojas muy puras).
- Algoritmo eficiente para encontrar mejor split (aproximado).
- Implementación eficiente.