Boosting

Carleos Artime, C.; Corral Blanco, N.

14 de noviembre de 2024

Boosting

El término *Boosting* se refiere a los procedimiento de aprendizaje supervisado que combinan clasificadores 'débiles' para conseguir uno 'eficiente'.

Se basa en las ideas desarrolladas por Kearns y Valiant (1988, 1989) acerca de si dado un conjunto de clasificadores débiles es posible crear un clasificador eficiente.

En el boosting se utiliza el mismo conjunto completo de datos, aunque en cada iteración se van variando su peso, ya que los casos que son mal clasificados ganan peso y los otros lo pierden.

AdaBoost

El Adaptative Boosting (AdaBoost) es un algoritmo fue desarrollado por Freund y Schapire en 1995.

El algoritmo trabaja con el conjunto de datos $(\vec{x}_i, y_i)$,i=1,...n donde $x_i \in X$ representa a las variables predictoras ; $y_i \in Y = -1, 1$ la variable dependiente.

El algoritmo AdaBoost es un método iterativo que en cada etapa t calcula un criterio de clasificación h_t y unos pesos D_t.

$$h_t : X \to Y$$

Además, en cada iteración se modifican las ponderaciones del conjunto de entrenamiento, aumentando la de los individuos mal clasificados y disminuyedo la del resto.

AdaBoost

Para una muestra de tamaño n, la distribución de pesos iniciales en el conjunto de entrenamiento es $D_1(i) = \frac{1}{n}$.

El algoritmo AdaBoost tiene los siguiente pasos:

- Calcular la función de clasificación h_t que haga mínimo el error

$$e_t = \sum_{h_t(x_i) \neq y_i} D_t(i)$$

- Calcular la ponderación α_t del clasificador h_t definida por

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

- Actualizar la distribución de pesos

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

Z_t es un factor para conseguir que D_{t+1} sea una probabilidad.

AdaBoost

El clasificador final del algoritmo se define mediante la función:

$$H(x) = \text{signo}\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right)$$

que es un promedio ponderado de todas los clasificadores obtenidos en las T iteraciones.

- ¿Cómo son las ponderaciones α_t de los clasificadores?
- ¿Cómo son las ponderaciones $D_t(i)$ de los individuos?

Ponderaciones de los clasificadores

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right)$$

$\alpha_t = \frac{1}{2} \ln(\frac{1-e_t}{e_t})$: función decreciente en e_t.

Cuanto mayor sea el error de clasificación e_t menor será la importancia de h_t en el clasificador final H.

¿Cómo actúa el coeficiente $\frac{1}{2}$ en el valor de α_t?

Ponderación de cada individuo

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

Acierto en la clasificación: $y_i = h_t(x_i)$

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

La ponderación disminuye exponencialmente en función de α_t

AdaBoost

Acierto en la clasificación: $y_i = h_t(x_i)$

- Cuando $e_t \simeq 0 \iff \alpha_t \simeq \infty \iff \exp(-\alpha_t) \simeq 0$.
  Ponderación *muy pequeña*
- Cuando $e_t \simeq 0{,}5 \iff \alpha_t \simeq 0 \iff \exp(-\alpha_t) \simeq 1$.
  La ponderación apenas cambia

AdaBoost

Fallo en la clasificación: $y_i \neq h_t(x_i)$

- Cuando $e_t \simeq 0 \iff \alpha_t \simeq \infty \iff \exp(\alpha_t) \simeq \infty$.
  Ponderación *muy alta*
- Cuando $e_t \simeq 0{,}5 \iff \alpha_t \simeq 0 \iff \exp(-\alpha_t) \simeq 1$.
  La ponderación apenas cambia

AdaBoost

Librerías de R para Boosting

- adaboost
- fastAdaboost
- mboost
- adabag
- glmboost
- ....

AdaBoost

Código de R para fastAdaboost

```
load('YY.RData') summary(YY)
library(fastAdaboost, pos=16)
test.adaboost <- adaboost( G ~ y1+y2+y3+y4+y5+y6, data=YY,
nIter=10)
print(test.adaboost)
pred<- predict(test.adaboost, YY)
names(pred)
```

Real AdaBoost

La distribución de pesos iniciales es $D_1(i) = \frac{1}{n}$.

El algoritmo Real AdaBoost tiene los siguiente pasos:

- Usar un clasificador para estimar las probabilidades

$$p_t(\vec{x}) = \hat{P}_{D_t}(y = 1|\vec{x})$$

- Calcular

$$h_t(\vec{x}) = \frac{1}{2} \ln \frac{P_{D_t}(y = 1 \mid \vec{x})}{P_{D_t}(y = -1 \mid \vec{x})}$$

- Actualizar la distribución de pesos

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

Z_t es un factor para conseguir que D_{t+1} sea una probabilidad

Real AdaBoost

El clasificador final del algoritmo se define mediante la función:

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

Su mayor inconveniente es que $h_t(\vec{x})$ depende del cociente de probabilidades que puede ser muy inestable.

$$\frac{P_{D_t}(y = 1 \mid \vec{x})}{P_{D_t}(y = -1 \mid \vec{x})}$$

Logit AdaBoost

Se define $y^* = \frac{y+1}{2},\ y^* \in \{0,1\}$

Valores iniciales de la clasificación, los pesos y las probabilidades:

$$H(\vec{x}_i) = 0; \quad D_1(i) = \frac{1}{n}; \quad p_1(\vec{x}_i) = p_1(y = 1|\vec{x}_i) = \frac{1}{2}$$

El algoritmo Logit AdaBoost tiene los siguiente pasos:

- Calcular

$$z_i = \frac{y_i^* - p_t(\vec{x}_i)}{p_t(\vec{x}_i)(1 - p_t(\vec{x}_i))}$$

$$D_{t+1}(i) = p_t(\vec{x}_i)(1 - p_t(\vec{x}_i))$$

- Calcular $h_t(\vec{x})$ mediante una regresión mínimo-cuadrática D_t-ponderada de z_i sobre $\vec{x}_i$
- Actualizar $H(\vec{x}) \leftarrow H(\vec{x}) + \frac{1}{2}h_t(\vec{x})$ y $p_t(x) \leftarrow \frac{e^{H(\vec{x})}}{e^{H(x)}+e^{-H(x)}}$.

Logit AdaBoost

El clasificador final del algoritmo se define mediante la función:

$$H(x) = \text{signo}\left(\sum_{t=1}^{T} h_t(x)\right)$$

Cuando $p(\vec{x})$ está cerca de 0 o de 1 puede haber problemas de estabilidad en el cálculo de z_i.
En esos casos, se acota entre $-z_{\max}$ y $z_{\max}$
Valores recomendados: $2 < z_{\max} < 4$

Los pesos D_t no deben ser menores que $2 \times$ el cero de la máquina.

Gentle AdaBoost

Gentle AdaBoost usa un algoritmo tipo Newton por etapas.
En la etapa inicial $H(\vec{x}) = 0$

- Calcular $h_t(\vec{x})$ mediante el criterio de mínimos cuadrados ponderados de y sobre $\vec{x}$
- Actualizar la distribución de pesos

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

Z_t es un factor para conseguir que D_{t+1} sea una probabilidad

Gentle AdaBoost

El algoritmo Gentle AdaBoost usa una regresión de mínimos cuadrados ponderada para calcular $h_t(\vec{x})$.

Su principal diferencia con el Real AdaBoost es que evita calcular el log del cociente de probabilidades.

El clasificador final del algoritmo se define mediante la función:

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

AdaBoost para clasificación múltiple

Dado un conjunto de datos $(\vec{x}_i, y_i)$, donde $y_i \in \{1, \ldots, J\}$

El algoritmo AdaBoost.M1 tiene los siguiente pasos:

- Obtener la función de clasificación $h_t(\vec{x}) \in \{1, \ldots, J\}$
- Calcular el error

$$e_t = \sum_{h_t(\vec{x}_i) \neq y_i} D_t(i)$$

- Calcular la ponderación α_t del clasificador h_t definida por

$$\alpha_t = \frac{1}{2} ln(\frac{1 - e_t}{e_t})$$

- Actualizar la distribución de pesos

$$D_{t+1}(i) = \frac{D_t(i)\, exp(-\alpha_t\, I(h_t(\vec{x}_i) \neq y_i))}{Z_t}$$

Z_t es un factor para conseguir que D_{t+1} sea una probabilidad

AdaBoost para clasificación múltiple

El algoritmo AdaBoost.M1 modifica el cálculo de α_t y considera que la busqueda es adecuada cuando $e_t < \frac{1}{J}$

El clasificador final del algoritmo se define mediante la función:

$$H(x) = \arg\max_{j \in Y} \sum_{t=1}^{T} \alpha_t \, I(h_t(\vec{x}) = j)$$

Representa la clase 'más votada', para $\vec{x}$, en las T iteraciones.

AdaBoost para clasificación múltiple

Dado un conjunto de datos $(\vec{x}_i, y_i)$, donde $y_i \in \{1, \ldots, J\}$

El algoritmo SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) tiene los siguiente pasos:

- Obtener la función de clasificación $h_t(\vec{x}) \in \{1, \ldots, J\}$
- Calcular el error

$$e_t = \sum_{h_t(\vec{x}_i) \neq y_i} D_t(i)$$

- Calcular la ponderación α_t del clasificador h_t definida por

$$\alpha_t = \ln \frac{1 - e_t}{e_t} + \ln(J - 1)$$

- Actualizar la distribución de pesos; $\sum_i D_{t+1}(i) = 1$

$$D_{t+1}(i) \propto D_t(i) \exp(-\alpha_t I(h_t(\vec{x}_i) \neq y_i))$$

Logit AdaBoost para clasificación múltiple.

El algoritmo Logit AdaBoost aborda la clasificación de un individuo entre J clases como J problemas de clasificación binaria

Se consideran J variables de clasificación $y^*_{ij} = \frac{y_{ij}+1}{2}$

Las ponderaciones iniciales son: $D_1(i) = \frac{1}{n}$

Las probabilidades iniciales son

$$p_1(\vec{x}_i) = \hat{P}_{D_1}(y = 1|\vec{x}_i) = \frac{1}{J}$$

Los J clasificadores $H_j(\vec{x}) = 0$

Logit AdaBoost para clasificación múltiple.

Los etapas que sigue son las siguientes:

- Repetir para $t \in \{1, \ldots, T\}$
  - Repetir para $j \in \{1, \ldots, J\}$
    - Calcular

$$z_{ij} = \frac{y^*_{ij} - p_{j(t-1)}(\vec{x}_i)}{p_{j(t-1)}(\vec{x}_i)(1 - p_{j(t-1)}(\vec{x}_i))}$$

$$D_t(i,j) = p_{j(t-1)}(\vec{x}_i)(1 - p_{j(t-1)}(\vec{x}_i))$$

    - Calcular $h_j t(\vec{x})$ mediante una regresión mínimo-cuadrática D_t-ponderada de z_{ij} sobre $\vec{x}_i$
  - Actualizar $H_j(\vec{x}) \leftarrow H_j(\vec{x}) + \frac{1}{2} h_t(\vec{x})$ y $p_{jt}(x) \leftarrow \frac{e^{H_j(\vec{x})}}{e^{H_(x)} + e^{-H_(x)}}$ con la restrición $\sum_{j=1}^{J} H_j(\vec{x}) = 0$.

El clasificador final del algoritmo vale:

$$H(x) = \arg \max_j \left(H_j(\vec{x}) \right)$$