Boosting

Carleos Artime, C.; Corral Blanco, N.

21 de noviembre de 2018

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 desarrolado 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 una distribución de 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 inidiales 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

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

- Calcular la ponderación α_t del clasificador h_t definida por

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

- 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 algoritmos se define mediante la función:

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

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

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 algoritmos se define mediante la función:

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

Su mayor inconveniente es que $h_t(\vec{x})$ depende del cociente de probabilidades

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

Logit AdaBoost

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

$$p_1(\vec{x}_i) = \hat{P}_{D_1}(y = 1|\vec{x}_i) = \frac{1}{2}$$

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

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 algoritmos 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 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 lugar de calcular el óptimo en cada una de ellas
En la etapa inicial $H(\vec{x}) = 0$

- Calcular $h_t(\vec{x})$ mediante una regresión ponderada de mínimos cuadrados 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 probabilidad
El clasificador final del algoritmos 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 algoritmos se define mediante la función:

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

que representa la categoría 'más votada', para la muestra $\vec{x}$, 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 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

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

- 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

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 $p_{jt}(x) \leftarrow \frac{e^{F_j(\vec{x})}}{e^{F(x)} + e^{-F(x)}}$ con la restrición $\sum_{j=1}^{J} H_j(\vec{x}) = 0$.

El clasificador final del algoritmos vale:

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

Coeficiente kappa de Cohen

- Mide el grado de concordancia entre dos observadores.
- Se clasifican N individuos en k categorias excluyentes.
- Se elimina el efecto de coincidencias por azar.
- Inconvenientes:
  - Las frecuencias marginales se consideran fijas
  - Esa corrección tiene sentido cuando los observadores responden al azar cuando no tienen una respuesta clara.

Coeficiente kappa de Cohen

$$\kappa = \frac{\Pr(C) - \Pr(Ca)}{1 - \Pr(Ca)}$$

P(C) = Probabilidad de Concordancia entre los observadores
P(Ca) = Probabilidad de Concordancia al azar entre los observadores
Si $\kappa = 1$, acuerdo total.
Si no hay acuerdo entre los calificadores distinto al que cabría esperar por azar, $\kappa = 0$.
Estadísticos similares:

pi de Scott (1955) Difieren en cuanto a la forma de cálculo de Pr(Ca).

kappa de Fleiss (1971) Se usa cuando hay más de dos observadores. Generaliza la pi de Scott.

Coeficiente kappa de Cohen: Ejemplo

Se tiene un grupo de 50 personas que presentan una solicitud de subvención.
Cada propuesta de subvención es analizada por dos evaluadores (A y B) que anotan un "Sí.º un "No".

A: \ B:	Sí	No
Sí	20	5
No	10	15

Diagonal $\Rightarrow$ concordancia.
Antidiagonal $\Rightarrow$ discordancia.

$$\Pr(C) = \frac{20+15}{50} = 0{,}7$$

$$\Pr(Ca) = \frac{25}{50}\frac{30}{50} + \frac{25}{50}\frac{20}{50} = 0{,}5$$

$$\kappa = \frac{0{,}7 - 0{,}5}{1 - 0{,}5} = 0{,}4$$