Combinacion de clasificadores: Bagging y Boosting

Edgar Acuña

Department of Mathematics
University of Puerto Rico -Mayaguez

El error de Mala clasificacion

Sea C(x, L) , el clasificador construido usando la muestra de entrenamiento, L, y T una muestra de prueba, extraida de la misma poblacion de donde proviene L. Entonces, el error de mala clasificacion (ME) del clasificador C es la proporcion de casos en T que han sido mal clasificados por C.

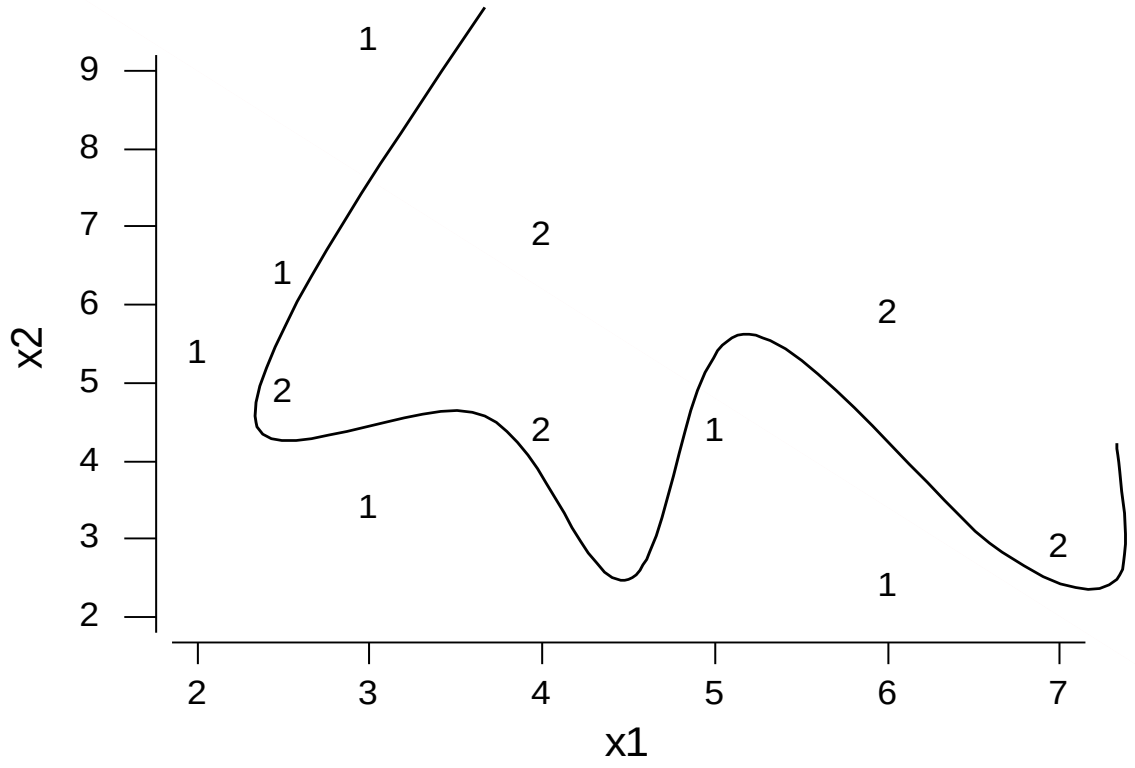
El ME puede ser descompuesto como

$$ME(C)=ME(C^*)+Sesgo^2(C)+Var(C)$$

donde $C^*(x)=argmax_j P(Y=j/X=x)$ (Bayes Classifier)

El clasificador puede sobreajustar los datos (Poco sesgo y varianza grande) o subajustar los datos (Sesgo grande y poca varianza).
Breiman (1996) heuristicamente define un clasificador como inestable si un pequeno cambio en los datos, L , puede producir grandes cambios en la clasificacion. Clasificadores inestables tienen sesgo bajo y varianza grande.
CART y redes neurales son clasificadores inestables.
Analisis discriminante lineal y clasificadores k-nn son estables.

Sobreajuste



Combinación de clasificadores

Combinando la prediccion de varios clasificadores se puede reducir la varianza y sesgo. Esta combinacion es llamada un **Ensemble** y en general es mas precisa que los clasificadores individuales. Entre los metodos para crear ensembles estan:

Bagging (Bootstrap aggregating by Breiman, 1996)

AdaBoosting (Adaptive Boosting by Freund and Schapire, 1996)

Arcing (Adaptively resampling and combining, by Breiman (1998).

Gradient Boosting (Friedman, 1999)

Muestra Bootstrap

Una muestra bootstrap de una muestra de entrenamiento L es una muestra CON Reemplazo de L y de su mismo tamano.

```
> x=c(5,3,12,13,21,31,8,9,15,17,24,32) #muestra original
> boot1=sample(x,replace=T)
> boot1
 [1] 31  9 32 32 15 31 21 13 15 17 17  9
> unique(boot1)
[1] 31  9 32 15 21 13 17
> boot2=sample(x,replace=T)
> unique(boot2)
[1] 13 31 21  5 24 32  3 15
```

Aproximadamente 37% de las instancias de la muestra de entrenamiento L NO aparecen en una muestra boostrap cualquiera.

The Bagging Algorithm: Breiman (1996)

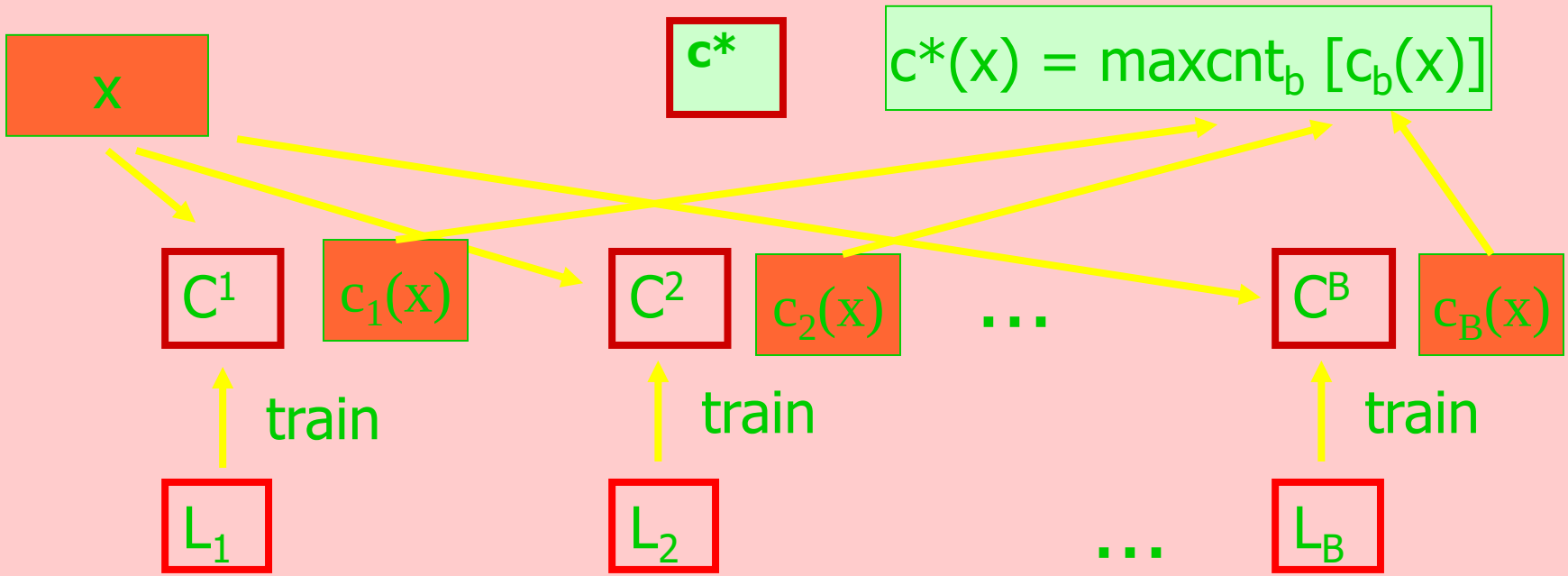
Input: learning set $\mathcal{L}$, classifier C, integer T (number of bootstrap samples), J clases.

1. For i=1 to T {
2. B_i= Muestra Bootstrap de $\mathcal{L}$ (muestra con reemplazo)
3. $C_i=C(B_i)$
4. }
5. $C_A(x)= \arg\max_{j\in\{1,.....J\}} \sum_{i:C_i(x)=j} 1$ (The class most voted)

Output: Ensemble C_A

Bagging con validacion cruzada

- From the training sample **L** select B random samples with replacement (bootstrap samples) obtaining **B** different training samples $L_1,...,L_B$ of size N.
- For each sample L_b a classifier C^b is built .
- Using 10-fold cross validation , each case x of L is assigned to the class c*(x)=j by voting.



Bagging usando R

```
>crossval(ionosphere,method="rpart",repet=10)
[1] 0.1287749
>library(adabag) #hace solo bagging para arboles y estima el error con validacion cruzada
>library(mlbench)
>data(Ionosphere)
>Ionosphere$V2<-NULL #Todos los valores de V2 son CEROS
>ionos.cv=bagging.cv(Class ~ ., data=Ionosphere)
> ionos.cv$error
[1] 0.09401
> ionos.cv$confusion
                Observed Class
Predicted Class bad good
           bad  108   15
           good  18  210
```

Estimacion del error por Out-of-Bag (OOB)

Asumiendo que se toman K=100 muestras bootstrap de la muestra original L, entonces una instancia de L no sera elegida en aproximadamente 37 de estas muestras y puede ser considerada como parte de una muestra de prueba T.
Asi, una instancia cualquiera tendria como 37 predicciones entre las que se votaria para tener una prediccion final
El estimado del error de clasificacion por Out-of-bag es el promedio del numero de errores de dichas predicciones

Bagging usando R[1]

```
>library(ipred)
>a=bagging(Class ~ ., data=Ionosphere, coob=TRUE)
>#Out-of-Bag error estimation
> mean(predict(a,Ionosphere[,-34])$class!=Ionosphere$Class)
[1] 0.04273504
 bagging(as.factor(V33) ~ ., data=ionosphere, coob=TRUE)

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = as.factor(V33) ~ ., data =
ionosphere,
    coob = TRUE)

Out-of-bag estimate of misclassification error:  0.094
```

Bagging usando R [2]

```
>library(adabag) #hace solo bagging para arboles pero
estimando el error con validacion cruzada
>library(mlbench)
>data(PimaIndiansDiabetes)
>Diad.cv=bagging.cv(diabetes ~ ., data=PimaIndiansDiabetes
> diab.cv$error
[1] 0.2421875
> diab.cv$confusion
                Observed Class
Predicted Class neg pos
            neg 426 112
            pos  74 156
```

Bagging usando R[3]

> crossval(diabetes,method="rpart",repet=10)
[1] 0.2610677
>library(ipred) #hace solo bagging para arboles pero estimando el error con out-of-bag
> bagging(as.factor(V9) ~ ., data=diabetes, coob=TRUE)

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = as.factor(V9) ~ ., data = diabetes,
    coob = TRUE)

Out-of-bag estimate of misclassification error: 0.2461
coob = TRUE)

Boosting

Aqui tambien se eligen muestras con reemplazo, pero se dan peso a cada observacion. En la primera muestra bootstrap cada observacion tiene el mismo peso 1/N, pero despues el peso de cada observacion va cambiando de tal manera que las observaciones incorrectamente clasificadas en la muestra actual tengan mayor peso de ser elegidas en la siguiente muestra.

Adaboosting Algorithm:Freund & Schapire[1996]

Input: Learning set L , classifier C, integers N, T

1. B= L with weights $w_1(x_j)=1/N$. For j=1,…..N

2. For i=1 to T { $C_i=C(B)$

3. Set $e_i = \sum_{x_j \in B: C_i(x_j) \neq y_j} w_i(\mathbf{x}_j)$. If either $e_i>1/2$ or $e_i=0$ then restart assigning equal weights.

4. Set $\beta_i=e_i/(1-e_i)$

5. Udpate the weights: for each $x_j \in B$, if $C_i(x_j)\neq y_j$ then $w_{i+1}(x_j)=w_i(x_j)/2e_i$, else $w_{i+1}(x_j)=w_i(x_j)/2(1-e_i)$}

6.$C^*(x)=\arg\max_{j\in\{1,\ldots.. J\}} \sum_{i:C_i(x)=j} \log \frac{1}{\beta_i}$

Output: Ensemble C*

Boosting usando R

```
>library(adabag) #hace solo boosting para arboles pero
estimando el error con validacion cruzada
>library(mlbench)
> diabetes.adaboostcv <- boosting.cv(diabetes
~.,data=PimaIndiansDiabetes,mfinal=mfinal,
coeflearn="Zhu",
+ control=rpart.control(maxdepth=maxdepth))
>data(PimaIndiansDiabetes)
> diabetes.adaboostcv[-1]
$confusion
                Observed Class
Predicted Class neg pos
            neg 395 116
            pos 105 152
$error
[1] 0.2877604
```

- Bagging reduce la varianza. AdaBoosting reduce tanto el sesgo como la varianza.
- Bagging puede ser facilmente paralelizado, pero Boosting es esencialmente secuencial y solo alguna parte del algoritmo puede ser paralelizado.
- Boosting es solo util para datos de gran tamano y para clasificadores con pobre rendimiento.