Introducción al aprendizaje por conjuntos

Introducción

Quisiéramos ofrecer una visión general básica del aprendizaje por conjuntos. Este aprendizaje implica combinar múltiples predicciones derivadas de diferentes técnicas para crear una predicción general más sólida.

Por ejemplo, las predicciones de un bosque aleatorio, una máquina de soporte vectorial y un modelo lineal simple pueden combinarse para crear un conjunto de predicciones final más sólido. La clave para crear un conjunto eficaz es la diversidad de modelos. Un conjunto con dos técnicas muy similares tendrá un rendimiento inferior al de un conjunto de modelos más diverso.

Algunas técnicas de aprendizaje por conjuntos, como la combinación y el apilamiento de modelos bayesianos, intentan ponderar los modelos antes de combinarlos. Dejaremos la discusión sobre cómo usar estas técnicas para otro día y, en su lugar, nos centraremos en la combinación de modelos simples.

Configuración inicial

Comenzaremos con variables definidas por el lector. En estas variables, x2 y x3 introducen tendencias no lineales distintivas y nos permitirá evaluar el rendimiento de las técnicas de aprendizaje no lineal.

```
> set.seed(10)
> y<-c(1:1000)
> x1<-c(1:1000)*runif(1000,min=0,max=2)
> x2<-(c(1:1000)*runif(1000,min=0,max=2))^2
> x3<-log(c(1:1000)*runif(1000,min=0,max=2))

> lm_fit<-lm(y~x1+x2+x3)
> summary(lm_fit)
```

Como puedes ver, el valor de R cuadrado es de 0.6658, lo que indica que los predictores tienen una correlación lineal menor con la variable dependiente.

```
Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max
-458.14 -128.49  -29.09   90.89  736.65

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.209e+01  2.138e+01  -2.904  0.00377 **
x1           2.322e-01  1.363e-02  17.039  < 2e-16 ***
x2           1.559e-04  8.673e-06  17.980  < 2e-16 ***
x3           6.672e+01  4.138e+00  16.124  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 167.2 on 996 degrees of freedom
Multiple R-squared:  0.6658,    Adjusted R-squared:  0.6648
F-statistic: 661.3 on 3 and 996 DF,  p-value: < 2.2e-16
```

```
> set.seed(10)
> all_data<-data.frame(y,x1,x2,x3)
> positions <- sample(nrow(all_data),size=floor((nrow(all_data)/4)*3))
> training<- all_data[positions,]
> testing<- all_data[-positions,]
> lm_fit<-lm(y~x1+x2+x3,data=training)
> predictions<-predict(lm_fit,newdata=testing)
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 173.3119
```

Dividir los datos en conjuntos de entrenamiento y prueba, y usar un modelo lineal simple para realizar predicciones sobre el conjunto de prueba, arroja un error cuadrático medio de 177.36.

```
> library(foreach)
> length_divisor<-6
> iterations<-5000
> predictions<-foreach(m=1:iterations,.combine=cbind) %do% {
+  training_positions <- sample(nrow(training), size=floor((nrow(training)/length_divisor)))
+  train_pos<-1:nrow(training) %in% training_positions
+  lm_fit<-lm(y~x1+x2+x3,data=training[train_pos,])
+  predict(lm_fit,newdata=testing)
+ }
> predictions<-rowMeans(predictions)
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 173.3296
```

Crear 5000 modelos lineales simples y promediar los resultados genera un error de predicción de 173.3296, ligeramente superior a los resultados iniciales.

Creación del primer conjunto

Para crear nuestro conjunto inicial, usaremos el modelo lineal y un bosque aleatorio. Un bosque aleatorio puede ser una potente técnica de aprendizaje. Crea múltiples árboles de decisión a partir de conjuntos aleatorios de predictores y observaciones. Será menos útil en este caso, ya que solo tenemos tres predictores, pero aun así ilustrará la idea general.

Ahora, probaremos la eficacia de un bosque aleatorio con nuestros datos:

```
> library(randomForest)
> rf_fit<-randomForest(y~x1+x2+x3,data=training,ntree=500)
> predictions<-predict(rf_fit,newdata=testing)
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 132.3325
```

Nuestro valor de error fue de 132.3325 con el código anterior. Como puedes ver, los bosques aleatorios pueden realizar predicciones mucho mejores que los modelos lineales. En este caso, el modelo lineal no pudo procesar los predictores no lineales, mientras que el bosque aleatorio sí. Sin embargo, hay que tener cuidado con la trampa del sobreajuste, ya que, si bien los bosques aleatorios son mucho menos propensos a este problema que un solo árbol de decisión, aún puede ocurrir con datos muy ruidosos.

Ten en cuenta que un bosque aleatorio ya incorpora el concepto de bagging en el algoritmo básico, por lo que no obtendremos ningún beneficio al ejecutar un bosque aleatorio a través de nuestra función de bagging. En su lugar, haremos lo siguiente:

```
> length_divisor<-6
> iterations<-5000
> predictions<-foreach(m=1:iterations,.combine=cbind) %do% {
+   training_positions <- sample(nrow(training), size=floor((nrow(training)/length_divisor)))
+   train_pos<-1:nrow(training) %in% training_positions
+   lm_fit<-lm(y~x1+x2+x3,data=training[train_pos,])
+   predict(lm_fit,newdata=testing)
+ }
> lm_predictions<-rowMeans(predictions)
> library(randomForest)
> rf_fit<-randomForest(y~x1+x2+x3,data=training,ntree=500)
> rf_predictions<-predict(rf_fit,newdata=testing)
```

```
> predictions<-(lm_predictions+rf_predictions)/2
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 145.0811
```

Esto combina los resultados del modelo lineal y del bosque aleatorio con pesos iguales. Como era de esperar, considerando que el modelo lineal es más débil que el bosque aleatorio, obtenemos un error de 145.0811. Si bien este no es un resultado fantástico, ilustra el aprendizaje conjunto.

Mejorando el rendimiento de nuestro conjunto

A partir de aquí, tenemos dos opciones. Podemos combinar las predicciones del modelo lineal y del bosque aleatorio en diferentes proporciones hasta obtener un mejor resultado (nosotros lo haríamos con precaución en el mundo real, utilizando un conjunto de referencia para encontrar los pesos óptimos y probando las ponderaciones en otro conjunto de referencia, ya que esto puede provocar sobreajuste si se realiza incorrectamente), o podemos reemplazar el modelo lineal por uno mejor. Probaremos ambos, pero primero combinaremos los resultados en diferentes proporciones. Nuestro conocimiento previo de las tasas de error nos indica que debemos utilizar una proporción pequeña para el modelo lineal.

```
> predictions<-(lm_predictions+rf_predictions*9)/10
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 133.0887
```

Esto resulta en una tasa de error de 133.0887, lo cual no representa una mejora con respecto al bosque aleatorio por sí solo. A continuación, reemplazamos el modelo lineal con una máquina de soporte vectorial (SVM) del paquete e1071, que proporciona una interfaz de R para libSVM. Una máquina de soporte vectorial se basa en matemáticas complejas, pero es básicamente una técnica de aprendizaje automático más robusta que puede detectar tendencias no lineales en los datos, dependiendo del tipo de kernel utilizado.

```
> library(e1071)
> svm_fit<-svm(y~x1+x2+x3,data=training)
> svm_predictions<-predict(svm_fit,newdata=testing)
> error<-sqrt((sum((testing$y-svm_predictions)^2))/nrow(testing))
> error
[1] 133.6113
```

El error al usar una SVM es de 133.6113, superior tanto al del bosque aleatorio como a los modelos lineales. A continuación, intentaremos usar la SVM con nuestra función de bagging.

```
> length_divisor<-6
> iterations<-5000
> predictions<-foreach(m=1:iterations,.combine=cbind) %do% {
+   training_positions <- sample(nrow(training), size=floor((nrow(training)/length_divisor)))
+   train_pos<-1:nrow(training) %in% training_positions
+   svm_fit<-svm(y~x1+x2+x3,data=training[train_pos,])
+   predict(svm_fit,newdata=testing)
+ }
> svm2_predictions<-rowMeans(predictions)
> error<-sqrt((sum((testing$y-svm2_predictions)^2))/nrow(testing))
> error
[1] 143.8163
```

El error con 5000 iteraciones de un modelo SVM es de 143.8163. En este caso, parece que la SVM funciona mejor sin técnicas de bagging. Es posible que haya muy pocos datos para que sea efectiva cuando se usa de esta manera. Sin embargo, el tiempo que tarda una SVM aumenta exponencialmente (creo) con más observaciones, por lo que a veces será necesario aplicar diversas técnicas, como la reducción del número de observaciones o características, para mejorar el rendimiento de la SVM a niveles tolerables. De ahora en adelante, utilizaremos los resultados de la SVM única para el resto de este documento.

```
> predictions<-(svm_predictions+rf_predictions)/2
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 128.7762
```

Al combinar equitativamente las predicciones del modelo svm del modelo único con las predicciones del bosque aleatorio, obtenemos una tasa de error de 128.7762, inferior tanto al modelo SVM solo como al modelo del bosque aleatorio solo.

```
> predictions<-(svm_predictions*2+rf_predictions)/3
> error<-sqrt((sum((testing$y-predictions)^2))/nrow(testing))
> error
[1] 129.5104
```

Si ajustamos las proporciones para enfatizar el modelo SVM más robusto, aumentamos nuestro error a 129.5104.

Conclusión

Como hemos visto, el aprendizaje por conjuntos puede superar a cualquier modelo individual si se utiliza correctamente. Un buen ejercicio para continuar sería ver si otros modelos pueden mejorar el rendimiento al añadirlos al conjunto.