Tarea 7, Reconocimiento de Patrones

Francisco Javier Peralta Ramírez

1. En clasificación binaria, $y \in \{-1, 1\}$, verifica que si $\hat{f} = \min_f E \exp(-Yf(X))$, $\hat{y}(x) = sgn(\hat{f}(x))$ asigna x a la clase más probable.

   Empezamos tomando

$$\min_f E \exp(-Yf(X) = E_X E_{Y|X=x} \exp(-Yf(x)$$

   Es suficiente con minimizar para cada x

$$\begin{aligned}&\min_f E_{Y|X=x} \exp(-Yf(x))\\ &= P(Y=-1|X=x)\exp(f(x)) + P(Y=1|X=x)\exp(-f(x))\end{aligned}$$

   Encontramos el mínimo derivando e igualando a 0

$$\begin{aligned}\frac{\partial}{\partial f(x)} &= P(Y=-1|X=x)\exp(f(x)) - P(Y=1|X=x)\exp(-f(x)) = 0\\ \to P(Y=-1|X=x)\exp(f(x)) &= P(Y=1|X=x)\exp(-f(x))\\ \frac{P(Y=1|X=x)}{P(Y=-1|X=x)} &= \frac{\exp(f(x))}{\exp(-f(x))}\\ \frac{P(Y=1|X=x)}{P(Y=-1|X=x)} &= \exp(2f(x))\\ f(x) &= \frac{1}{2}\log\frac{P(Y=1|X=x)}{P(Y=-1|X=x)}\end{aligned}$$

   Notamos que si $P(Y=1|X=x) > P(Y=-1|X=x)$ entonces $\frac{P(Y=1|X=x)}{P(Y=-1|X=x)} > 1$ entonces $f(x) > 0$ y $\hat{y} = 1$, y si $P(Y=1|X=x) < P(Y=-1|X=x)$ entonces $\frac{P(Y=1|X=x)}{P(Y=-1|X=x)} < 1$ entonces $f(x) < 0$ y $\hat{y} = -1$. Esto corresponde al clasificador Bayesiano óptimo, es decir, se asigna x a la clase más probable.

2. Si usamos $y \in \{-1, 1\}$, verifica que la logverosimilitud de la regresión logística será de la forma

$$\sum_i \log\frac{1}{1+exp(-y_i f(x_i)}, \quad f(x) = \alpha + \beta^{\intercal} x$$

   Es decir, con el método de máxima verosimilitud para la estimación de los parámetros se usa como función de costo:

$$\log(1+\exp[-yf(x)])$$

   Tenemos que $P(Y=1|X=x) = 1/(1+\exp[-f(x)]) = \pi(x)$ por lo tanto $P(Y=-1|X=x) = 1/(1+\exp[f(x)]) = 1-\pi(x)$ En clase definimos la función de probabilidad como $p(x_i, y_i) = \pi x_i{}_i^y(1-\pi x_i)^{1-y_i}$, pero ahora con $y \in \{-1, 1\}$ tenemos que cambiar los exponentes de tal forma que el primer termino se eleve a 1 y segundo termino se eleve a 0 cuando $y = 1$ al reves cuando $y = -1$.

   Podemos usar

$$p(x_i, y_i) = \pi x_i{}^{(y_i+1)/2}(1-\pi x_i)^{(1-y_i)/2}$$

Con esto tenemos la función de verosimilitud (L) y log-verosimilitud (l)

$$
\begin{aligned}
L &= \prod_{i=1}^{n} \pi x_i^{(y_i+1)/2}(1-\pi x_i)^{(1-y_i)/2} \\
l &= \log(\prod_{i=1}^{n} \pi x_i^{(y_i+1)/2}(1-\pi x_i)^{(1-y_i)/2}) \\
l &= \sum_{i=1}^{n} \log(\pi x_i^{(y_i+1)/2}(1-\pi x_i)^{(1-y_i)/2}) \\
l &= \sum_{i=1}^{n} \frac{y_i+1}{2}\log(\pi x_i) + \frac{1-y_i}{2}\log(1-\pi x_i) \\
l &= \frac{1}{2}\sum_{i=1}^{n}(y_i+1)\log\left(\frac{1}{1+\exp[-f(x_i)]}\right) + (1-y_i)\log\left(\frac{1}{1+\exp[f(x_i)]}\right)
\end{aligned}
$$

Cuando $y_i = 1$

$$l = \sum_{i=1}^{n} \log \frac{1}{1+\exp[-f(x_i)]}$$

Cuando $y_i = -1$

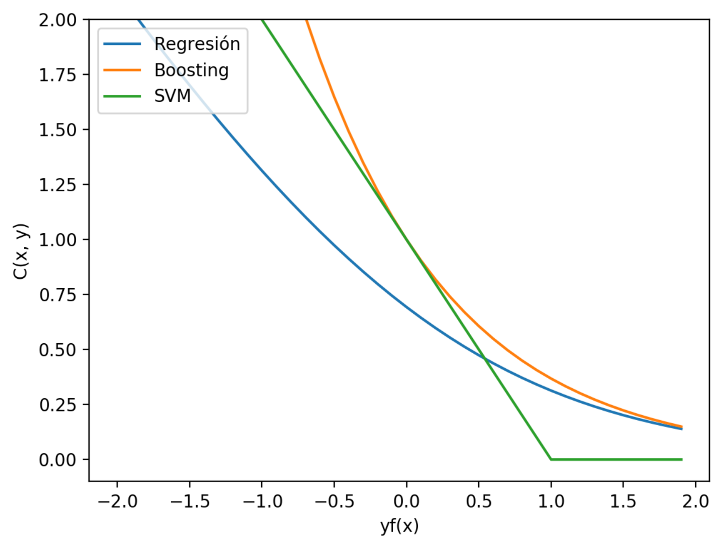
$$l = \sum_{i=1}^{n} \log \frac{1}{1+\exp[f(x_i)]}$$

Por lo que podemos simplificar a

$$l = \sum_{i=1}^{n} \log \frac{1}{1+\exp[-y_i f(x_i))}$$

Podemos graficar nuestra función de costo y comparar con *Boosting* y *SVM*. Las funciones son:

- **Regresión:** $C(x, y) = \log(1 + \exp[-yf(x)])$
- **Boosting:** $C(x, y) = \exp[-yf(x)]$
- **SVM:** $C(x, y) = \text{máx}(0, 1 - yf(x))$



Notamos que todos los métodos penaliza cuando se está clasificando bien, con SVM siendo el que menos penaliza y no penaliza despues de cierto punto. Por otra parte notamos que regresión es la función que menos incrementa con las malas penalizaciones.

3. Implementa en R la idea de random forests a partir de la libreria rpart para árboles de decisión. Usalo para construir clasificadores para los datos SPAM. Compáralo con Boosting.

   Iniciamos leyendo los datos de un archivo csv, y cambiamos el nombre de la última columna a “Y”

```
data <- read.csv("datos.data")
names(data)[length(names(data))] <- "Y" #cambia nombre del ultimo valor
data$Y <- factor(data$Y)
```

   Separamos los datos en un conjunto de entrenamiento y uno de prueba.

```
train_pct <- 0.75
n_train <- floor(train_pct * nrow(data))
rand_ind <- sample(nrow(data))

train <- data[head(rand_ind, n=n_train),]
test <- data[tail(rand_ind, n=nrow(data) - n_train),]
```

   Generamos multiples predictores con arboles usando modelos con un subconjunto de las variables disponibles

```
n_trees <- 51
n_pred <- trunc(sqrt(ncol(data)))

forest <- vector(mode="list", length=n_trees)
for (i in 1:n_trees) {
  p <- names(train)[sample(ncol(data) - 1, n_pred)]
  f <- paste("Y ~ ", paste(p, collapse = "+"))
  train_set <- train[sample(nrow(train), replace = T), ]
  fit <- rpart(as.formula(f), method='class', data=train_set)
  forest[[i]] <- fit
}
```

   Para obtener la predicción checamos si la mitad o más de los arboles predijeron una clase

```
res <- rep(0, nrow(test))
for (i in 1:nrow(test)) {
  val <- 0
  for (j in 1:n_trees) {
      if(predict(forest[[j]], newdata = test[i,], type = "class") == 1)
        val <- val +1
  }
  if(val >= n_trees/2) res[i] = 1
}
table(res, test$Y)
```

		classs	
		0	1
pred	0	690	146
	1	11	303

   Esto nos da un *Accuracy* de 0.86.

   Para Boosting

```
control <- rpart.control(cp = -1, maxdepth = 10, maxcompete = 1, xval = 0)
boost <- ada(Y ~ ., data = train, type = "discrete", control = control, iter = 50)
res_boost <- predict(boost, newdata =test)
table(res_boost, test$Y)
```

		classs	
		0	1
pred	0	681	24
	1	20	425

   Lo que nos da un *Accuracy* de 0.96