Boosting

Boosting es un "Ensemble Method" que combina varios "week learners" para crear un "strong learner". En general, todas las técnicas de boosting entrenan predictores de forma secuencial, y cada nuevo predictor "aprende" de los errores de su predecesor.

Primero tenemos que entender qué es un "weak learner". Un "weak learner" es un predictor que se comporta solo levemente mejor que un predictor aleatorio. Luego, un "strong learner" es un buen predictor. Ahora, ¿por qué un ensemble de "weak learners" puede ser un "strong learner"?

Pensemos en el siguiente ejemplo: supongamos una moneda sesgada con 51% de probabilidad de ser sello y 49% de ser cara. Si lanzamos la moneda 1000 veces, hay 75% de probabilidades de que la clase ganadora sea sello, y con 10000 lanzamientos, la probabilidad sube a 97%. Este mismo fenómeno se da al hacer boosting.

De todas formas, en el caso de los predictores hay correlación entre los errores de los predictores, principalmente porque todos se entrenan sobre los mismos datos, que hace que no sea verdad decir que 1000 clasificadores tengan 75% de accuracy.

Ahora vamos a estudiar uno de los métodos de Boosting más famosos.

AdaBoost

La idea de este algoritmo es que cada nuevo predictor le de más atención a las observaciones que el predictor anterior no supo clasificar.
En general, el "weak learner" que se use es un "decision stump".

Stump ⟶ 

Que corresponde a un decision tree de profundidad 1. Como podemos suponer, sus predicciones no son muy buenas.
Una diferencia respecto a hacer bagging es que al final no todos los votos pesan lo mismo.

Consideremos un dataset inicial con n observaciones. Tenemos que asignarle un peso a cada observación que inicialmente es $\frac{1}{n}$ para cada observación. Ahora entrenamos un primer predictor y hacemos predicciones sobre todo el dataset y calculamos el error del predictor

$$r = \frac{\sum_{\substack{i=1 \\ \hat{y}_i \neq y_i}}^{n} w_i}{\sum_{i=1}^{n} w_i}$$

Con w_i peso actual de la instancia i

Aquí estamos sumando los pesos de todas las instancias que clasificamos mal. Luego calculamos el peso del predictor como

$$\alpha = \eta \log\left(\frac{1-r}{r}\right)$$

Con η una tasa de aprendizaje

Es decir, pesa más mientras menos se equivoque. Ahora actualizamos los pesos:

$$w_i = \begin{cases} w_i & \text{si } \hat{y}_i = y_i \\ w_i e^{\alpha} & \text{si } \hat{y}_i \neq y_i \end{cases}$$

Y normalizamos los pesos (dividimos cada peso

por $\sum_{i=1}^{n} w_i$).

¡Y estamos listos con el primer predictor! Ahora entrenamos el segundo considerando los nuevos pesos.

¿Cómo entrenar con pesos? Hay dos opciones:

1) Calculamos la impureza con una función que admite pesos.
2) Samplemos el dataset considerando los pesos como una "distribución" solo para entrenar el predictor.

Luego el segundo predictor calcula r, α y actualiza los pesos. Luego repetimos con todos los predictores que queramos.

Ahora al momento de predecir una instancia nueva, cada predictor vota por la clase que predice, y el voto vale α. Gana la clase con la suma más grande. Esto es:

$$\hat{y}(x) = \underset{k}{\operatorname{argmax}} \sum_{\substack{j=1 \\ \hat{y}_j(x)=k}}^{m} \alpha_j$$

Con m número de predictores