keep coding

Machine Learning y Ciberseguridad

Bagging y Random Forest

keep coding

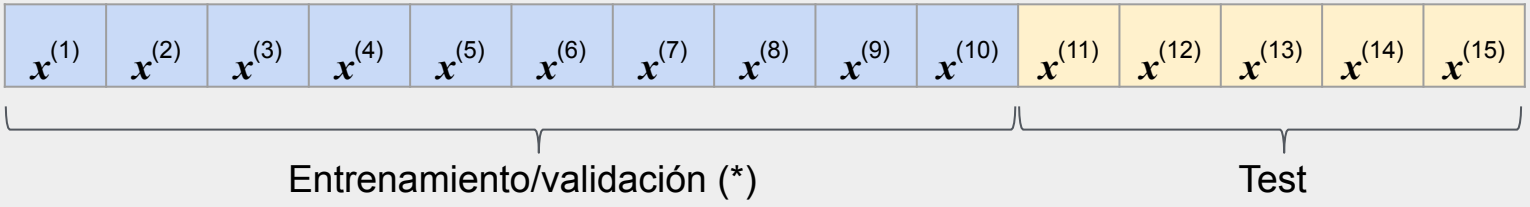
Motivación

- Combinar algoritmos, normalmente árboles, para mejorar sus prestaciones
- Proporcionan grandes prestaciones en problemas complejos

Índice

Remuestreo Bootstrap

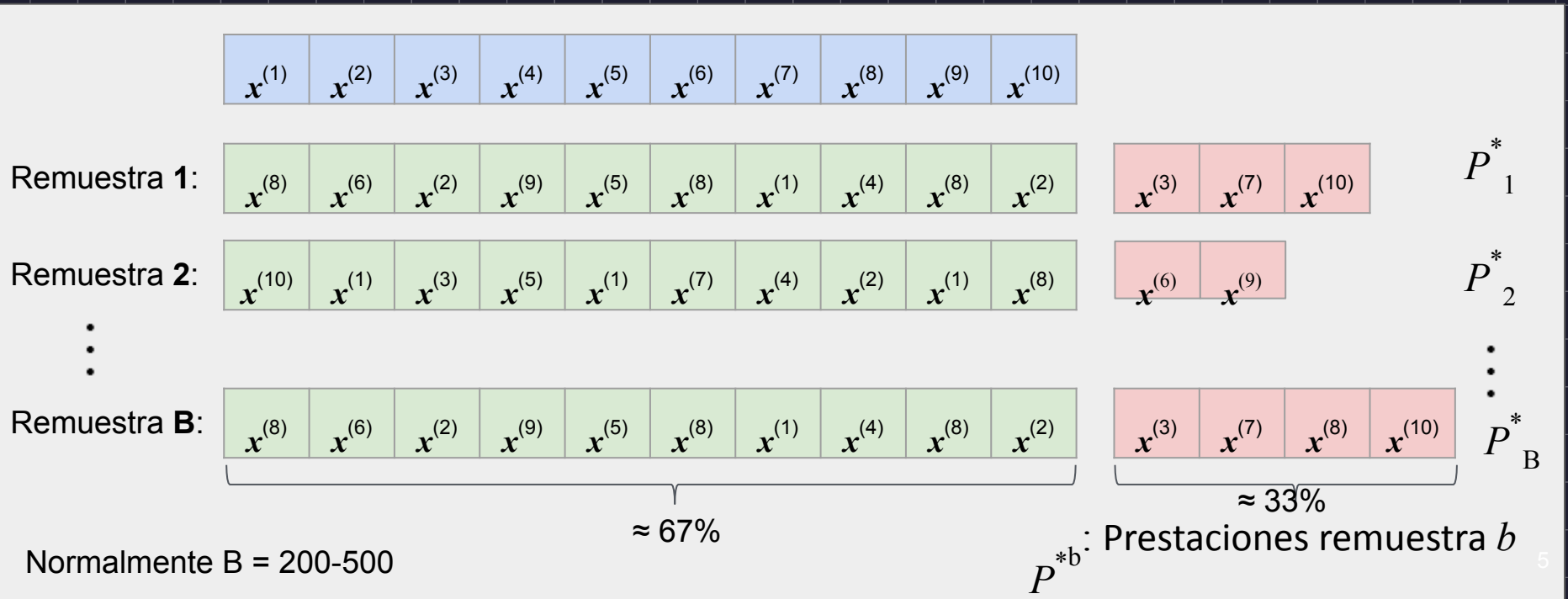
- Técnica estadística para cuantificar la incertidumbre de un estimador
  - En ML nos sirve para medir las prestaciones de un algoritmo
- Supongamos un problema de aprendizaje supervisado, donde disponemos de un conjunto de datos etiquetados {X, y}, con N = 15.



(*) numeración no es orden, los datos han sido ya aleatorizados

Remuestreo Bootstrap

- Bootstrap: remuestras con repetición



Índice

keep coding

Bagging: Bootstrap AGGregation

- Motivación: reducir varianza de los árboles de decisión (en función de la división los resultados pueden ser muy distintos)

- Utilizar bootstrap para combinar árboles de decisión:
  - Se construyen (entrenan) B árboles utilizando B remuestras
  - Se combina la salida para predecir una nueva muestra:
    - Regresión: average vote
    - Clasificación: majority vote

keep coding

Bagging: pros and cons

- OK
  - Mejoran las prestaciones sustancialmente
- KO
  - Si hay uno o varios predictores fuertes, puede que los B árboles generados sean bastante similares, por lo que no estamos reduciendo la varianza dado que los árboles están altamente correlacionados

Índice

keep coding

Random forest

- Motivación: decorrelacionar árboles remuestrados

- Utilizar bootstrap para combinar árboles de decisión:
  - Se construyen (entrenan) B árboles utilizando B remuestras
  - En la construcción de cada árbol, para cada split se fuerza a utilizar un subconjunto aleatorio de m < d predictores

- Normalmente m = $\sqrt{d}$
- Si m = d, entonces es Bagging
- Si el número de predictores relevantes es pequeño, y alta dimensionalidad, peligro de overfitting

Índice

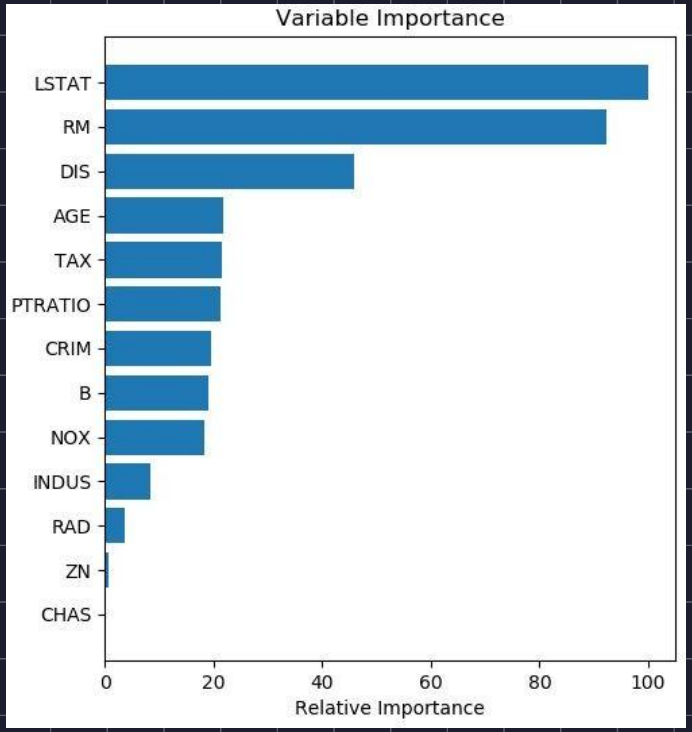
keep coding

Importancia de las variables

- Con la agregación de árboles se pierde interpretabilidad
- No obstante se puede extraer una **medida de la importancia** de cada variable
  - Cuánto mejoran las prestaciones en los splits asociados a dicha variable
  - En otras palabras: para cada split de cada árbol construido, se mide la mejora en prestaciones debido a la variable por la que se particiona el árbol.
- Medida relativa: se escala entre 0-100
- Puede aplicarse a un árbol individual, pero no es concluyente
- Se puede utilizar como ranking en selección de características, ¡pero hay que hacerlo bien! (recordad los métodos wrapper)

keep coding

Importancia de las variables



Referencias

- Introduction to Statistical Learning
  - Capítulo 5, sección 2
  - Capítulo 8, sección 2
- The Elements of Statistical Learning
  - Capítulo 10, sección 13
  - Capítulo 15
- Hands On Machine Learning.
  - Capítulo 7

keep coding

LET'S CODE

Índice

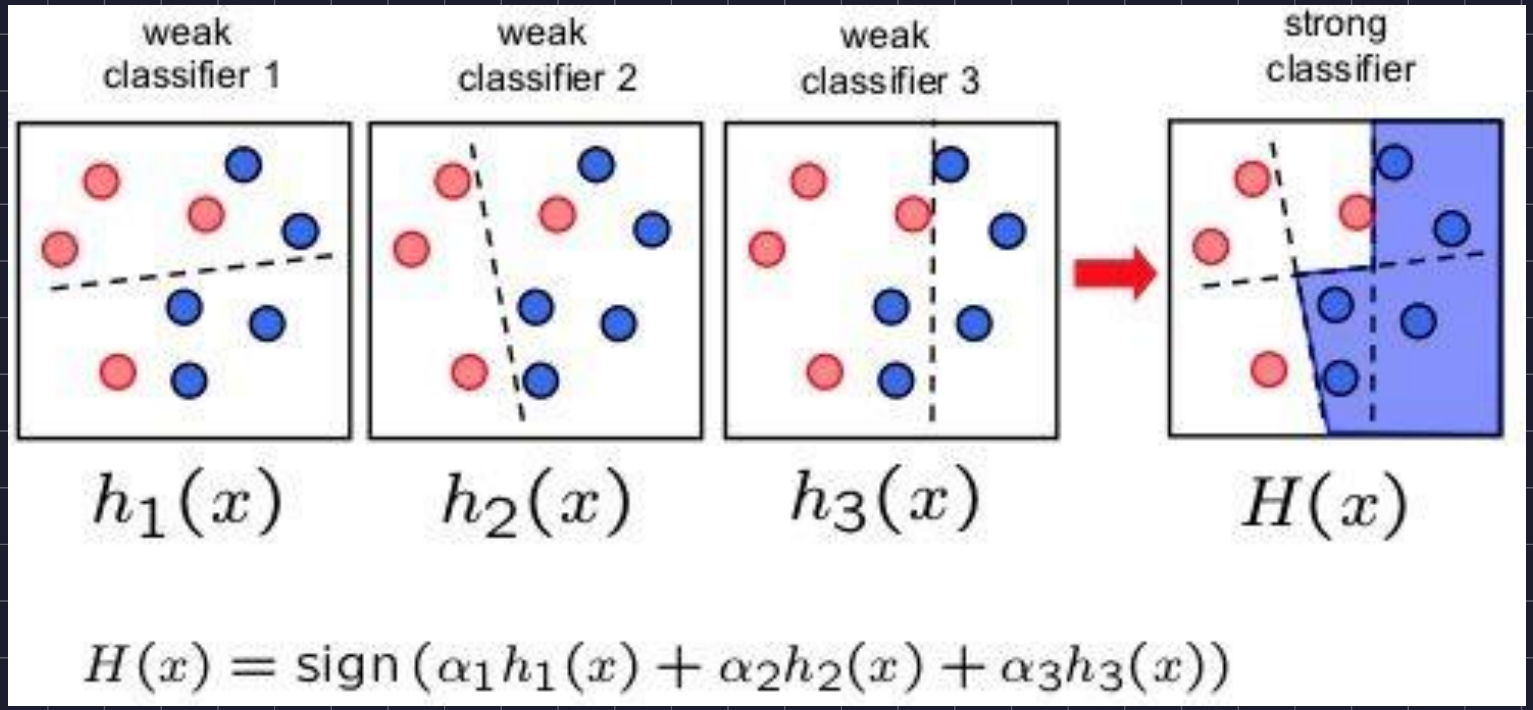
keep coding

Boosted Trees

- Una de las ideas más brillantes en machine learning tabular de los últimos tiempos
- Proceso iterativo basado en la combinación lineal de algoritmos sencillos (weak classifiers/regressors)
- Aplicable a cualquier algoritmo de ML. Normalmente árboles (Boosted Trees)
- Mucha gente lo usa como banco de pruebas para pruebas de concepto

keep coding

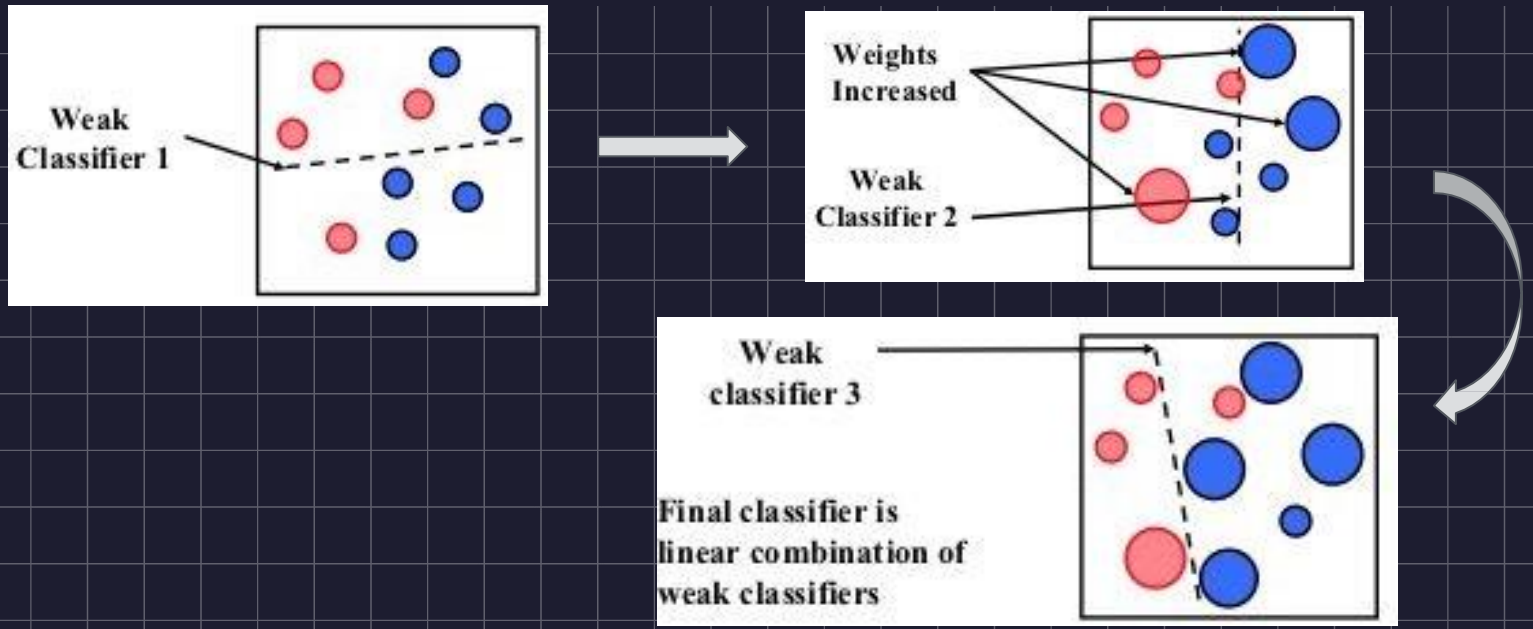
Boosted Trees: Intuición



keep coding

Proceso Iterativo

- En cada iteración le damos más peso a los errores



keep coding

Hiperparámetros

- **Número de árboles** (iteraciones). Si es muy alto peligro de overfitting. Seleccionamos con validación cruzada

- **Tasa de aprendizaje** (Learning Rate, alpha). Número positivo de valor pequeño, normalmente 0.01, 0.001. Está relacionado con el número de iteraciones. Si alpha es pequeño, se necesitarán más iteraciones para que las prestaciones converjan

- **Profundidad del árbol**. Controlamos la complejidad del árbol. Idealmente interesa que sea pequeña, así la capacidad de generalización es mayor. Se recomienda comenzar con valores pequeños.

keep coding

Implementaciones

- **Sklearn:**
  https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting
- **XGBoost (el más potente):**
  https://github.com/dmlc/xgboost
- **LightGBM (algo intermedio):**
  https://github.com/Microsoft/LightGBM

keep coding

In a nutshell

- Relaciones no lineales
- Sin necesidad de escalar variables
- Árbol aislado, muy interpretable (si no es muy profundo)
- Random Forest, algoritmo muy robusto, es un buen benchmark
- Mejores prestaciones, si se eligen con cuidado los parámetros libres
  - La potencia sin control no sirve de nada

Referencias

- The Elements of Statistical Learning.
  - Capítulo 10

- Introduction to Statistical Learning.
  - Capítulo 8, sección 3

keep coding

LET'S CODE

www.keepcoding.io cursos@keepcoding.io (+34) 916 33 1779