Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3

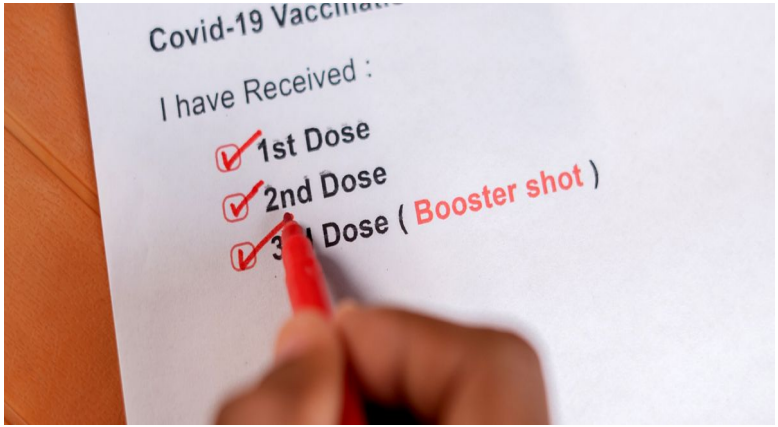
Aprendizaje automático

Aprendiendo mediante boosters

Boosters, boosting

Hay refuerzos en vacunas.

Hay refuerzos (boosters) para aprender!



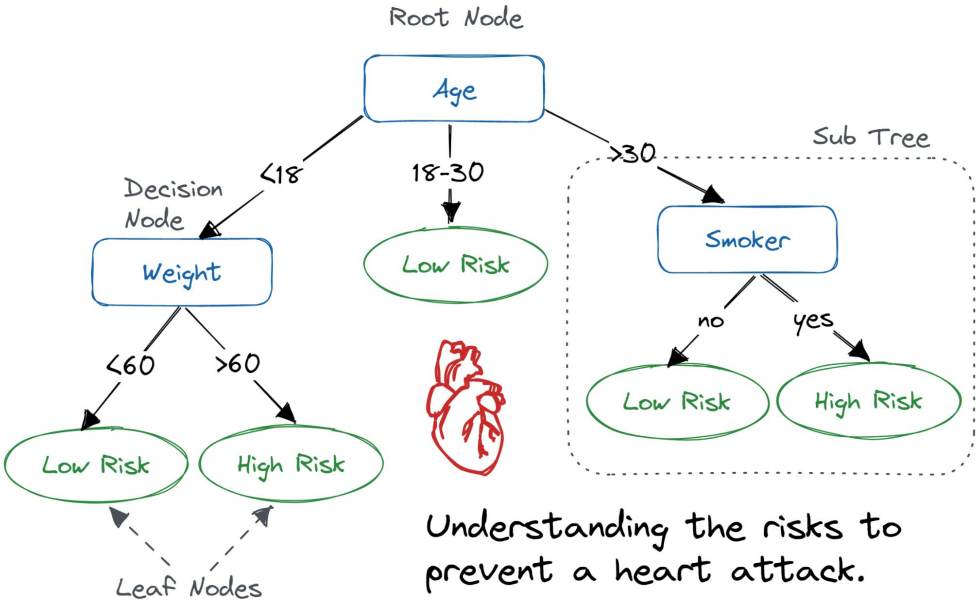
Primero un repaso: ensambles de árboles

Decision Trees:

- Unidad base
- Particiona variables
  - Busca bajar impurezas (GINI)
  - O bajar entropia (ganar información)

Ensambles de árboles:

- Random Forests
  - Bagging (Bootstrap aggregation)
- Extra Trees
  - Particiones aleatorias
- Boosted Trees
  - Esta clase!

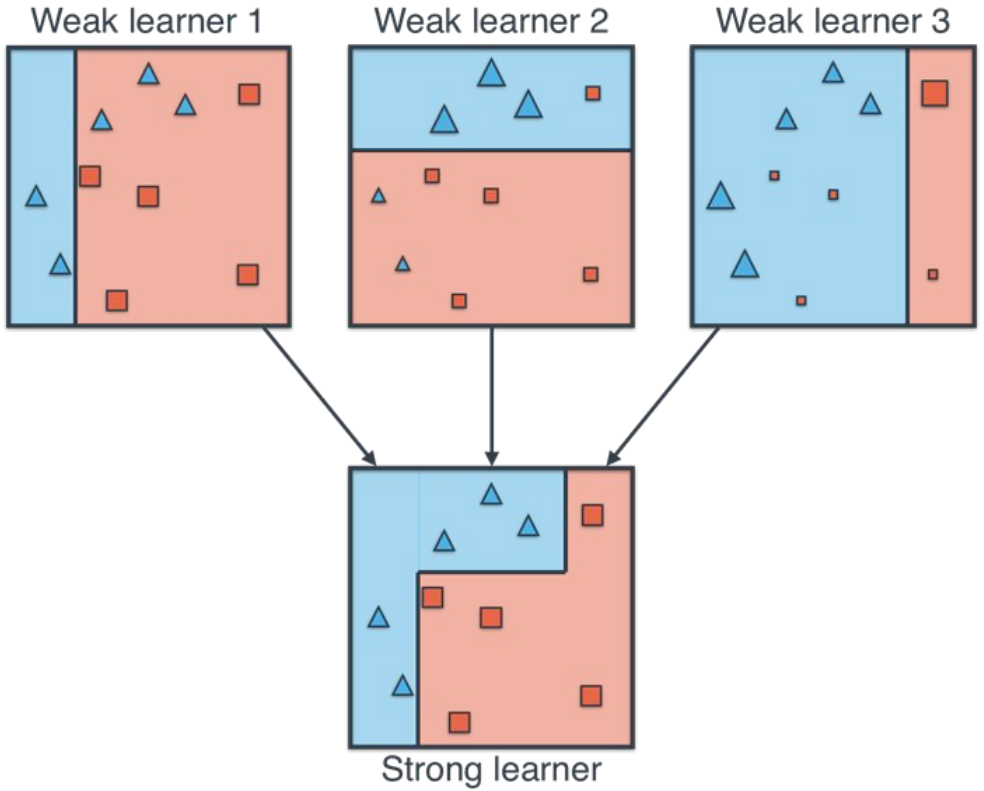


https://en.wikipedia.org/wiki/Decision_tree
https://en.wikipedia.org/wiki/Decision_tree_learning

El concepto de weak learners

Los modelos que aprenden algo

- Captan algo de la señal o estructura en los datos
- No son buenos modelos en forma global
- Combinándolos se logran mejores aprendizajes



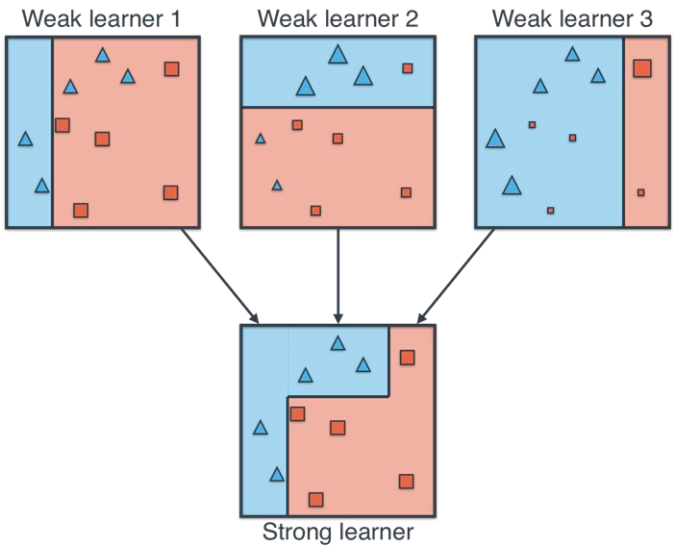
Concepto de ensambles

Combinar modelos

- **Weak learners** (aprendedores débiles → modelos subóptimos)
- **Strong learners** (mejores aprendedores)

↓

Robustez + Versatilidad

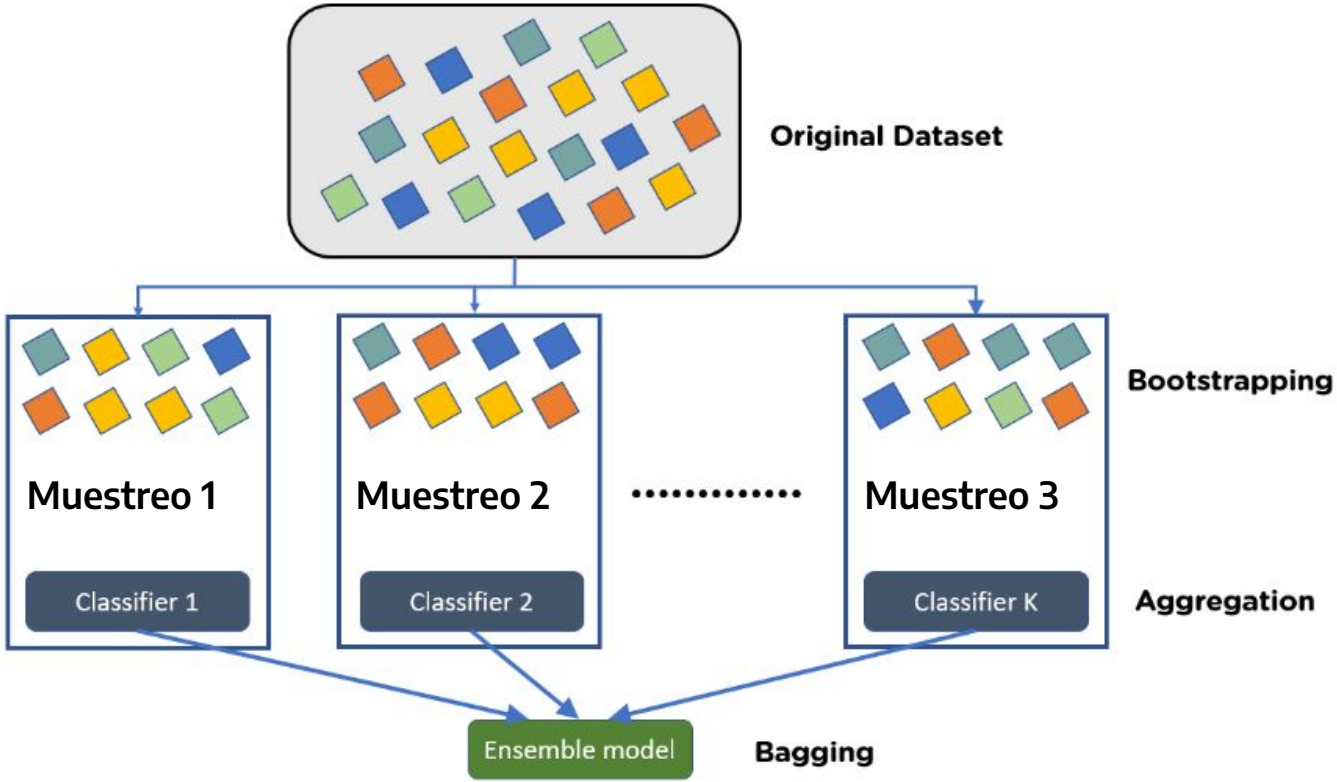


Aprendizaje paralelo: bagging

Random Forests

Cada modelo es un árbol entrenado sobre una muestra aleatoria de los datos originales

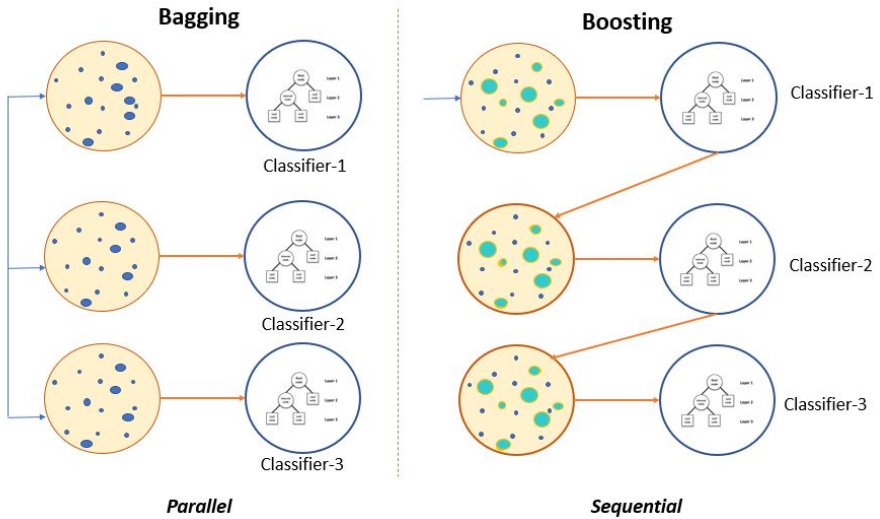
Aprendizaje en Paralelo!



Aprendizaje en serie: boosting

- Entrenar modelos ***secuencialmente***
- Enfocarse en cada paso en ***ejemplos de entrenamiento mal clasificados*** en el paso anterior
- Para enfocarse en ejemplos específicos boosting usa un ***set de entrenamiento pesado***

Peso = Weight (un coeficiente)



AdaBoost

AdaBoost = Adaptive Boosting (Freund & Schapire 1995)

Cómo funciona AdaBoost:

1. Entrenar un modelo
2. Usar el modelo
3. ***identificar casos mal clasificados***
4. Construir un modelo que clasifique mejor estos casos ← Como? **AdaBoost!**
5. Repetir los pasos 1 + 2

AdaBoost

Inicialización: todos los datos tienen el mismo ***peso***

- **w(i) = 1 / N (para i = 1, 2, ... N)**
- **D*t* = distribución de pesos para el modelo / clasificador *t***

Cálculo del error de clasificación (misclassification):

- **E_t = error del modelo / clasificador *t***
- **x(i) = elemento i (dato)**
- **w(i) = peso del elemento i**
- **y (i) = clase real**
- **h(i) = clase que predice el modelo**
- **I = clasificación (1 o -1)**

$$E_t = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i}$$

Cálculo de la performance:

- **α_t = el peso del modelo/clasificador *t***

$$\alpha_t = \frac{1}{2} ln \frac{(1 - TotalError)}{TotalError}$$

AdaBoost (continuado)

Actualización de los pesos:

- D_t = distribución de pesos

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Repetir:

- Hasta disminuir el error suficientemente
- O hasta un numero determinado de

AdaBoost como calcula los pesos

Ejemplo: supongamos que tenemos 4 datos (x), y que nuestro modelo t los clasifica así:

H_t (prediccion) = [1, 1, -1, -1]

Y_t (real) = [-1, 1, -1, 1]

Mal clasificado? = [1, 0, 0, 1]

Después de calcular el error y ajustar los pesos:

D_t = [0.5, 0.55, 0.7, 0.04]

E_t = (0.5 * 1) + (0.55 * 0) + (0.7 * 0) + (0.04 * 1) / (0.5 + 0.55 + 0.7 + 0.04)

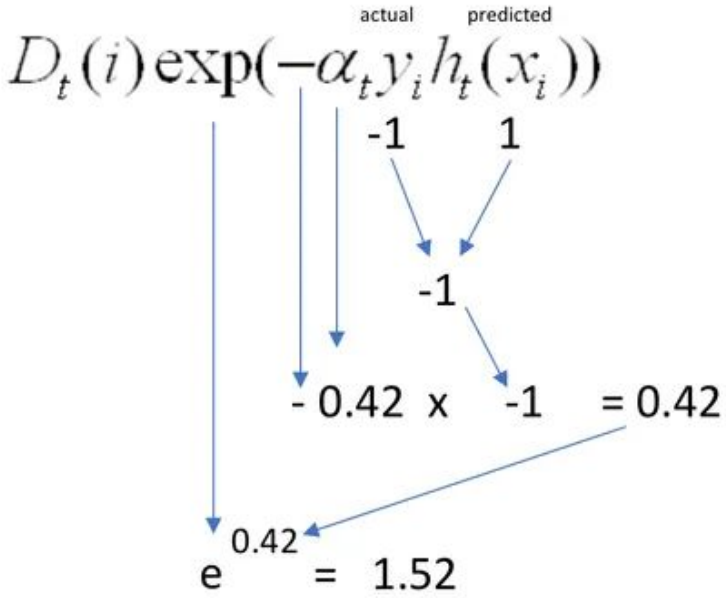
E_t = 0.3017...

Performance:

α_t = 1/2 * ln(1 - 0.3017 / 0.3017) = 0.42

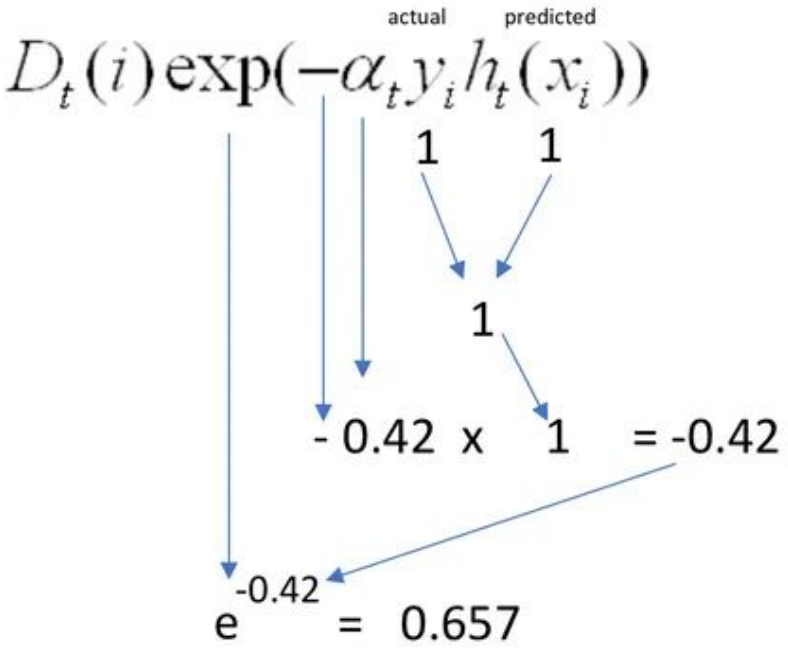
AdaBoost:

Qué pasa con los puntos mal clasificados?



AdaBoost

Y con los bien clasificados?



Gradient Boosting

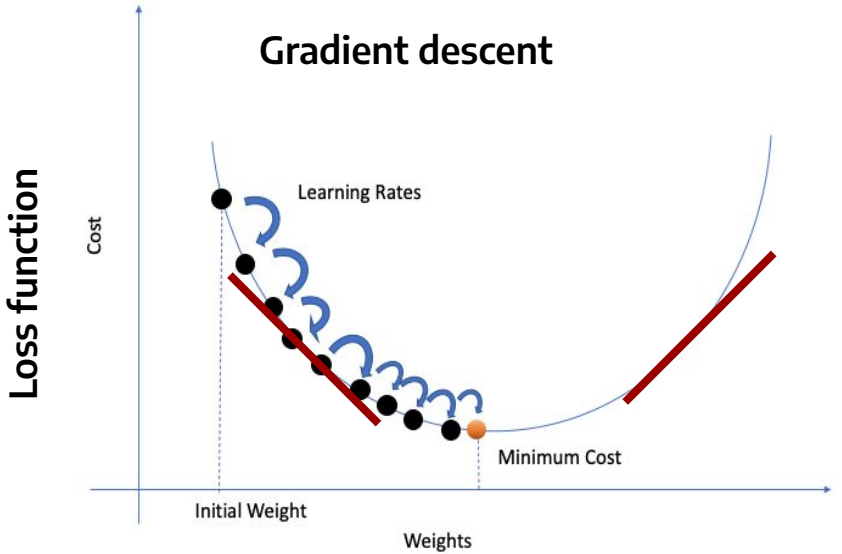
Gradient Boosting = Gradient descent + Boosting

En cada paso se mejora un “weak learner”

En AdaBoost se identifican problemas por su mayor peso

En GradientBoosting se identifican estos problemas por gradientes

- Residual error = diferencia entre el valor real y la predicción
- Gradient = residual error + signo (de la derivada o pendiente)



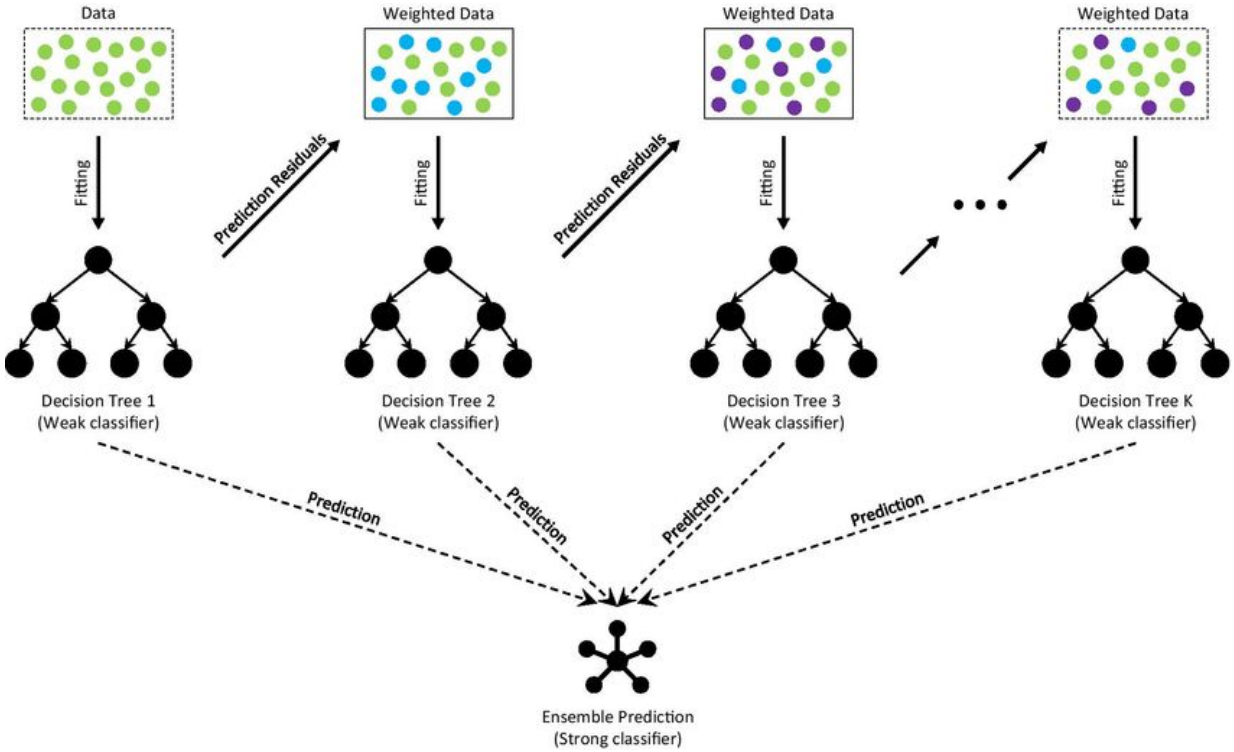
Gradient Boosting

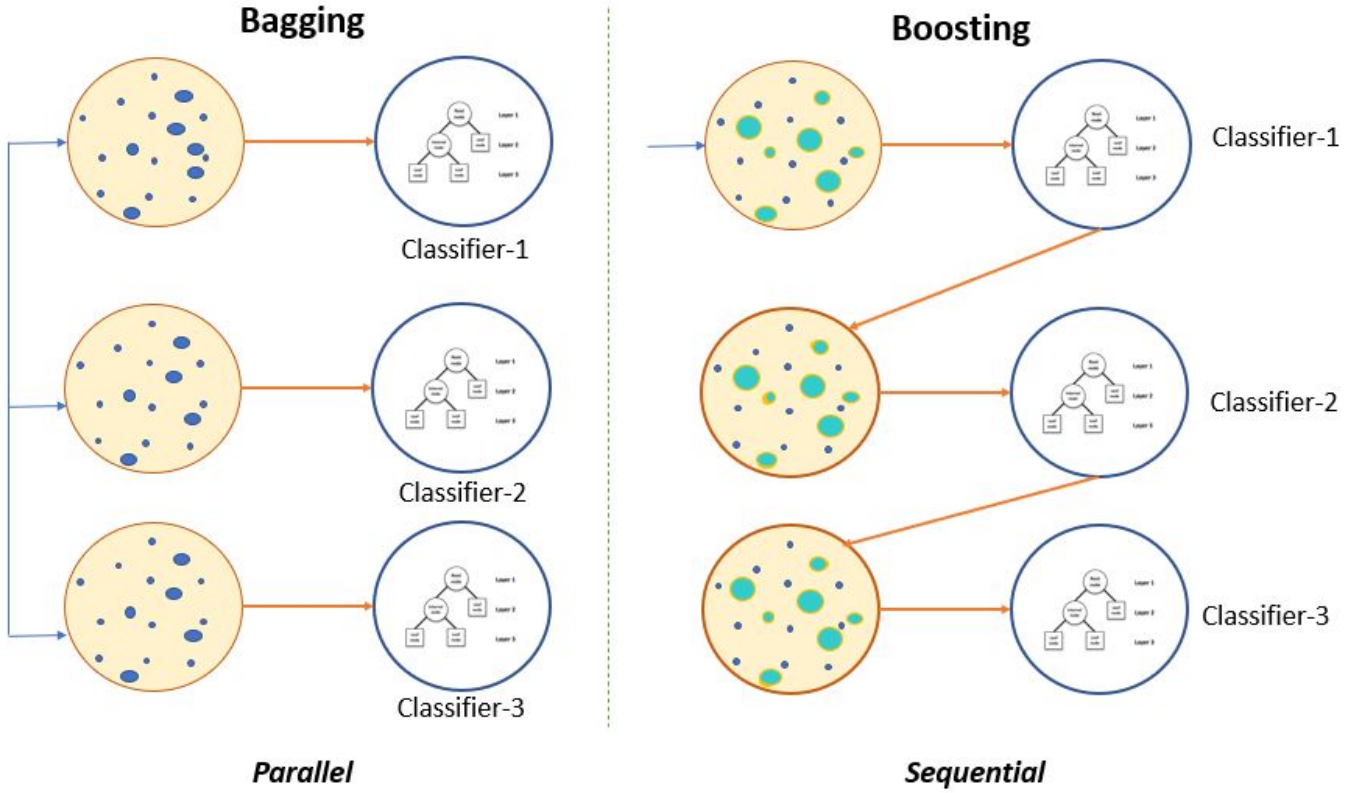
El uso de gradientes permite:

- **Intercambiar distintas funciones de pérdida (loss)**
  - Classifier: **loss*{'log_loss', 'exponential'}, default='log_loss'***
  - Regressor: **loss*{'squared_error', 'absolute_error', 'huber', 'quantile'}, default='squared_error'***

Gradient boosting (boosted trees?)

Ajusta una función derivable de pérdida (loss function) en cada paso.



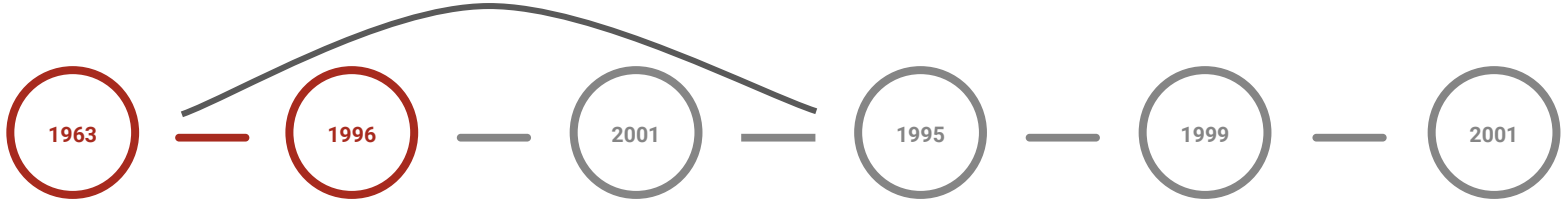

Bagging
Classifier-1
Classifier-2
Classifier-3
Parallel
Boosting
Classifier-1
Classifier-2
Classifier-3
Sequential

XGBoost

XGBoost = *Extreme Gradient Boosting*

- **Ensamble de modelos**
- **Empieza con un modelo base (base learner)**
  - Generalmente son decision trees (gbtree = gradient boosted tree)
  - Pero también pueden ser modelos lineales (gblinear = gradient boosted linear)
- **Funciona en forma similar a Gradient Boosting**
  - Agrega: uso de la segunda derivada de la función de pérdida
  - Optimizaciones computacionales, regularización para balancear sesgo vs desviación
    - `Estas optimizaciones corren en paralelo!`

Evolución de



Decision Trees

Lorem ipsum dolor sit amet, consectetur adipiscing.

Bagging

Método de ensamble de modelos en el cual se combinan los modelos mediante un sistema de votación (la mayoría gana).

Random Forests

Método de bagging mejorado en el cual en cada paso se usa un subconjunto aleatorio de features (variables). Es una combinación de data bagging + feature bagging.

Boosting

Modelos construidos secuencialmente. Se minimizan los errores de los modelos anteriores incrementando ('boosting') el peso de algunos modelos

Gradient Boosting

Boosting optimizado mediante el uso de 'Gradient Descent' para minimizar errores en modelos secuenciales

Extreme Gradient Boosting

Gradient boosting optimizado mediante procesamiento paralelo, podado de arboles, manejo de valores faltantes, y regularización

Boosting Tree