Modelos de ensamble (parte II)

Clase sincrónica

{desafío} latam_

Implementar ensambles secuenciales en problemas complejos, ajustando diferentes factores para optimizar la predicción.

- **Unidad 1: Modelos de ensamble**
  (Parte I)

  (Parte II)

  (Parte III)
- **Unidad 2: Redes neuronales**
  (Parte I)

  (Parte II)
- **Unidad 3: Procesamiento y Redes recurrentes**
  (Parte I)

  (Parte II)

{desafío} latam_

¿Qué aprenderás en esta sesión?

Aprenderás algoritmos de Machine Learning por medio de métodos de ensamble secuenciales: Adaptive Boosting, Gradient Boosting y XGradient Boosting. Al finalizar sabrás cómo operan, para qué sirven y cómo implementarlos

{desafío}
latam_

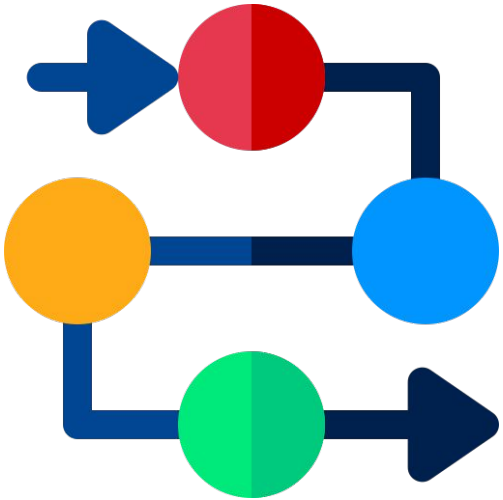
En general, ¿cómo funcionan los algoritmos de ensamble?

/* Boosting */

Boosting

Definición

Enfoque de ensamble secuencial, donde se usan aprendices débiles para conformar el ensamble, los cuales son potenciados secuencialmente para dar origen a un aprendiz fuerte.



Boosting

Ensambles paralelos y ensambles secuenciales

- En los ensambles paralelos se utilizan *aprendices fuertes*. En cambio, los ensambles secuenciales utilizan *aprendices débiles*.

- Para ambos tipos de ensamble el objetivo es ir dirigiendo el siguiente entrenamiento de un aprendiz de acuerdo a los errores del aprendiz actual.

- Los aprendices en los ensambles paralelos pueden ser entrenados en forma independiente, pudiendo aprovechar las actuales arquitecturas multicore. En los ensambles secuenciales secuenciales esto no ocurre, ya que precisamente se va utilizando un aprendiz después de otro.

/* Adaptive Boosting*/

Adaptive Boosting

Algoritmo de ensamble secuencial

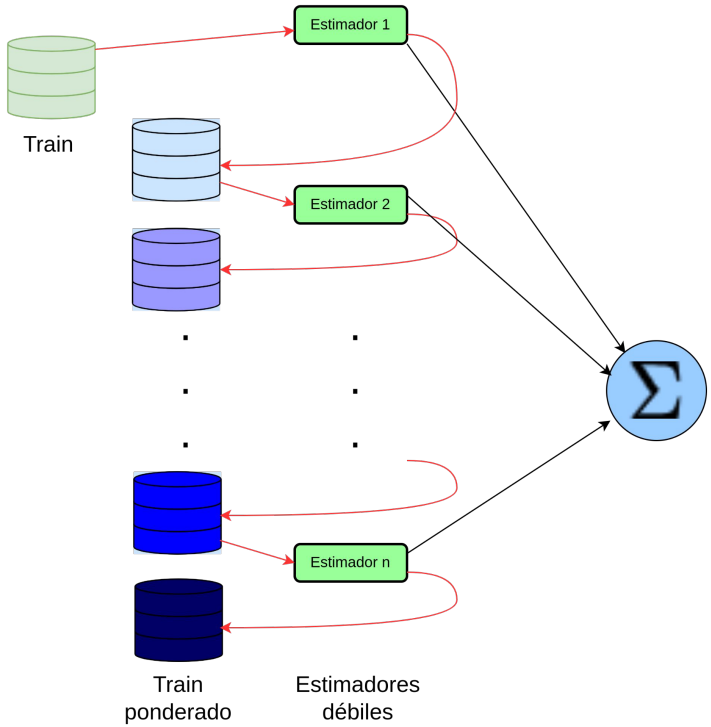
Fue propuesto por Freund y Schapire en 1996, siendo el primer algoritmo de ensamble secuencial.

Entre sus principales características se encuentran la reducción del overfit, y la mejora del desempeño del modelo por medio de la reducción del error en cada iteración, gracias al uso de aprendices débiles.

Adaptive Boosting

Algoritmo de ensamble secuencial

- Inicialización
- Iterar para cada aprendiz débil
  - Entrenamiento del aprendiz débil
  - Calcular el error ponderado
  - Calcular coeficiente alpha
  - Actualizar los pesos de la instancia y su normalización
- Salida construcción del modelo final.



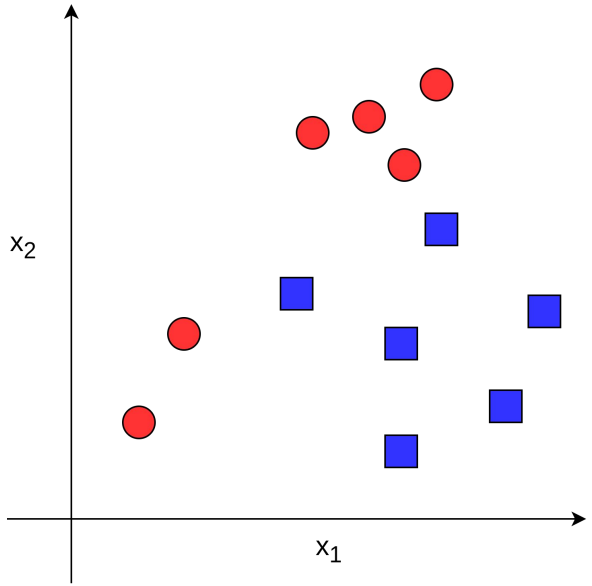
Adaptive Boosting

Inicialización

Se toma el conjunto de entrenamiento y aplicando una ponderación uniforme a cada observación se procede a iniciar el proceso.

$$w_i^m = \frac{1}{n}$$

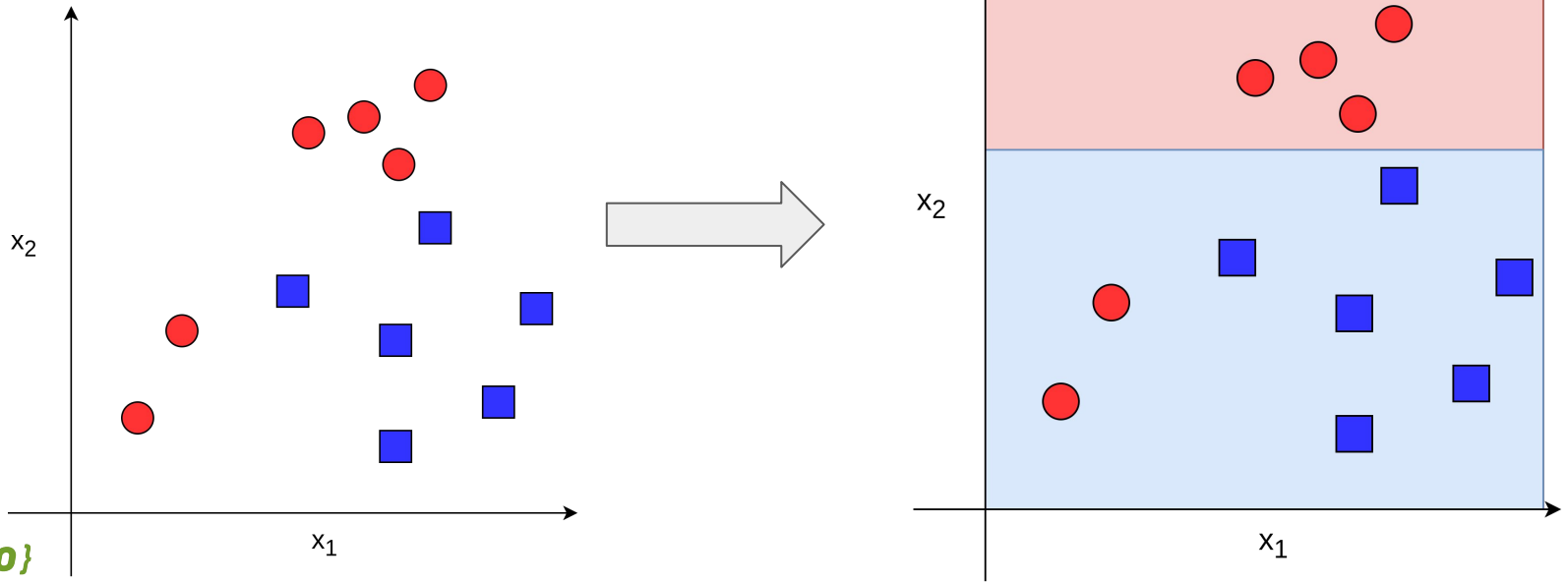
$$\sum_{i=1}^{n} w_i^m = 1$$



Adaptive Boosting

Iterar para cada aprendiz débil $h_m(X_{train})$

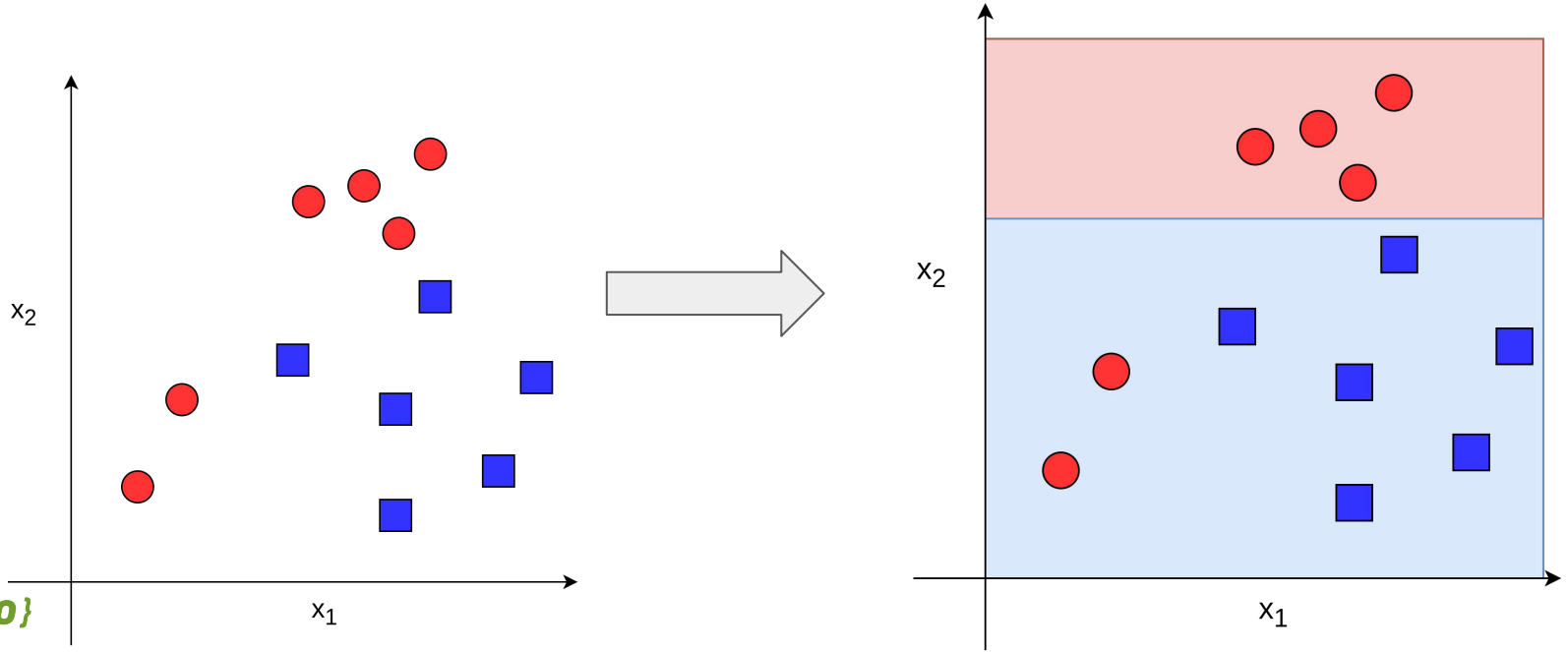
Se entrena el aprendiz débil y se calculan las predicciones.



Adaptive Boosting

Calcular la tasa de error ponderada

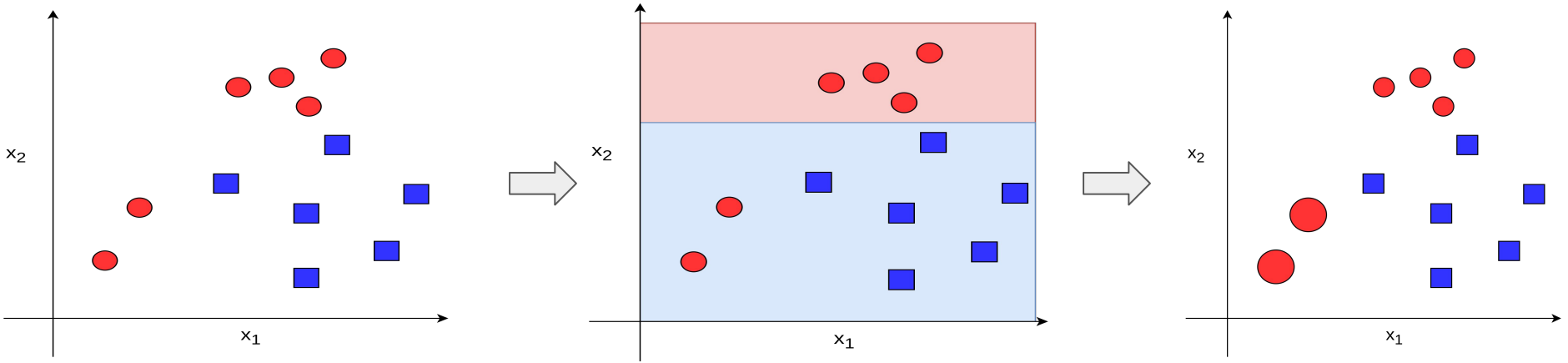
Se realizan las predicciones usando el modelo entrenado, y con esto calculamos los errores



Adaptive Boosting

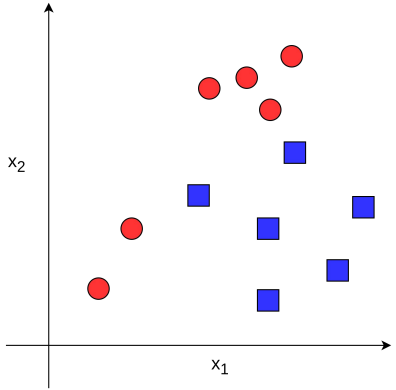
Iterar

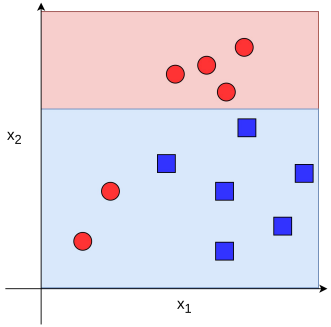
Una vez terminada la iteración, el conjunto de entrenamiento original queda con las observación mal clasificadas ponderadas, para ser usadas en el siguiente aprendiz débil.

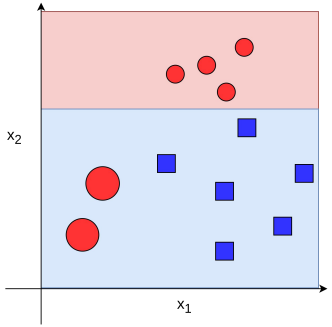


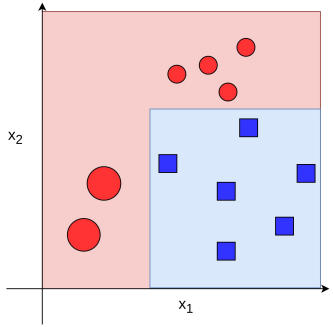
Adaptive Boosting

Construcción del modelo final para predicciones









/* Gradient Boosting */

Gradient Boosting

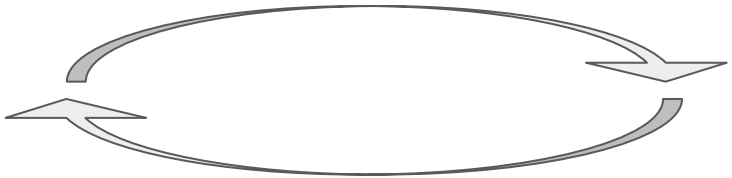
Descripción

- Gradient Boosting entrena cada aprendiz débil de acuerdo a los errores residuales del aprendiz anterior.
- Adaptive Boosting pondera las observaciones mal clasificadas, en cambio Gradient Boosting penaliza en base al error residual
- La penalización realizada en Gradient Boosting se hace por medio de una **función de pérdida** o **función de costo**. función de pérdida o costo, que mide la discrepancia entre los verdaderos valores y los valores predichos. Usando esta función el problema de encontrar el “mejor” modelo, se traduce en un problema de optimización.

Gradient Boosting

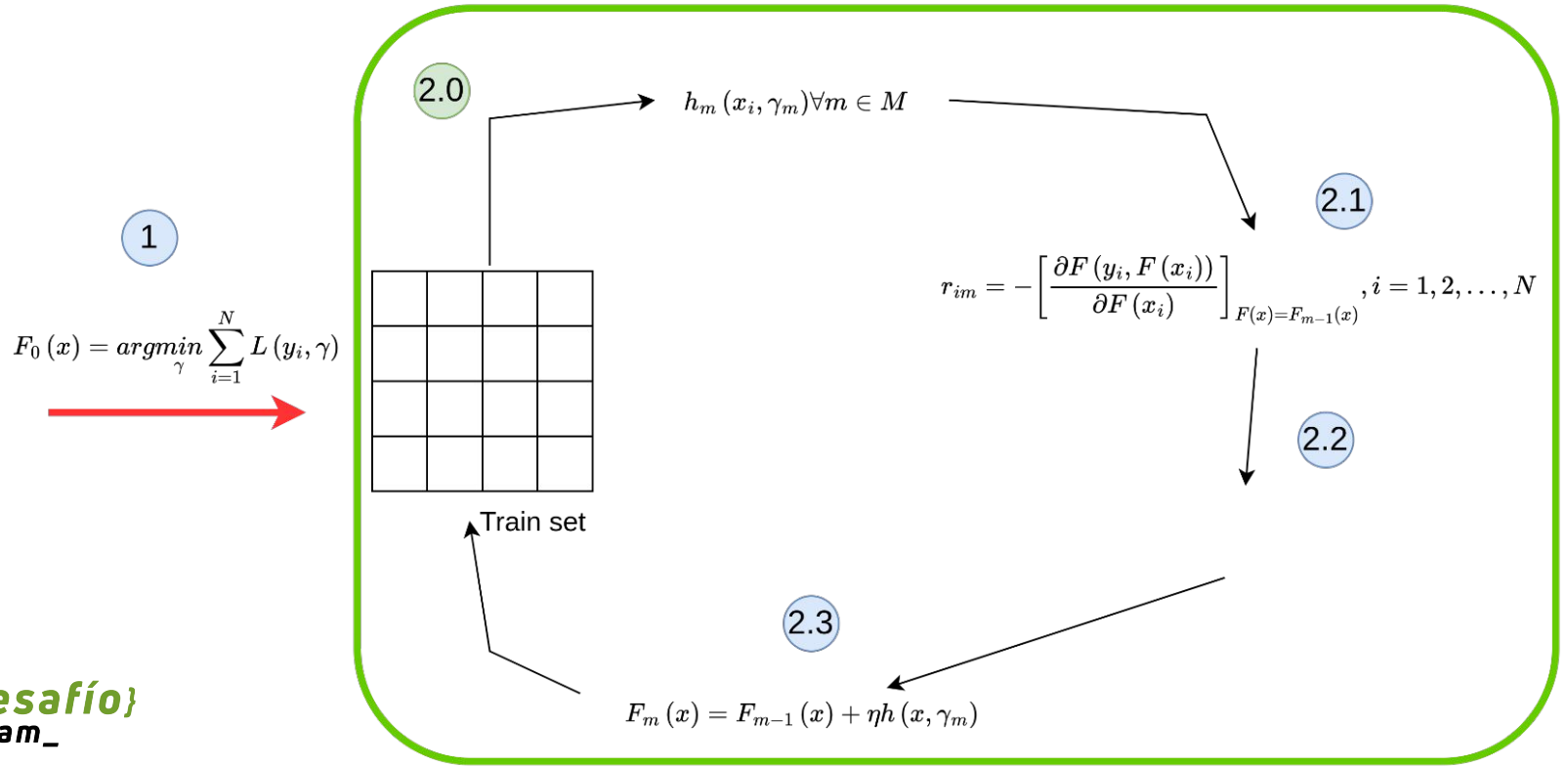
Algoritmo paso a paso

Paso 1	Paso 2.1	Paso 2.2	Paso 2.3
Se inicializa el modelo para devolver un valor predicho constante, que minimiza la función de pérdida.	Se calcula el pseudo residuo, por medio del gradiente negativo de la función de pérdida con respecto a los valores predichos.	Se entrena un aprendiz débil con los datos de entrenamiento asociados al residuo en la iteración anterior.	Se actualiza el modelo añadiendo los valores del aprendiz entrenado.



Gradient Boosting

Algoritmo paso a paso



{desafío} latam_

Gradient Boosting

Hiper parámetro learning_rate

- ***learning_rate*** define el “paso” que se dará en la función de pérdida en cada iteración, con el objetivo de alcanzar un mínimo.

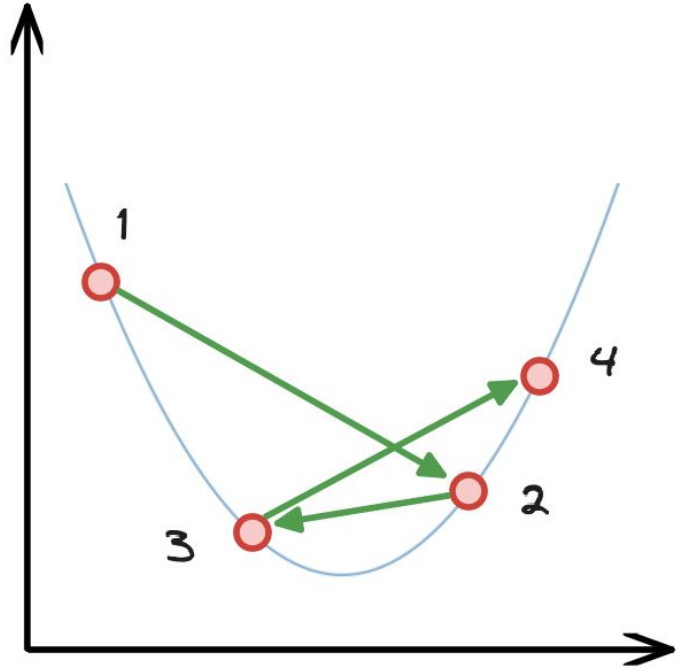
- Recordemos que en el descenso del gradiente nos ubicamos en un punto inicial y calculamos la dirección en que la función desciende más rápidamente. Esto se hace calculando el gradiente; junto con ello debemos definir cuánto nos moveremos en la dirección encontrada.

Gradient Boosting

Hiper parámetro learning_rate

***learning_rate* muy alto**

- Provoca que nos movamos más rápido en la función de pérdida
- Puede provocar falta de convergencia, en cuanto a que podríamos estar cerca del óptimo pero ya que el paso es alto nos haría volver a alejarnos una y otra vez.

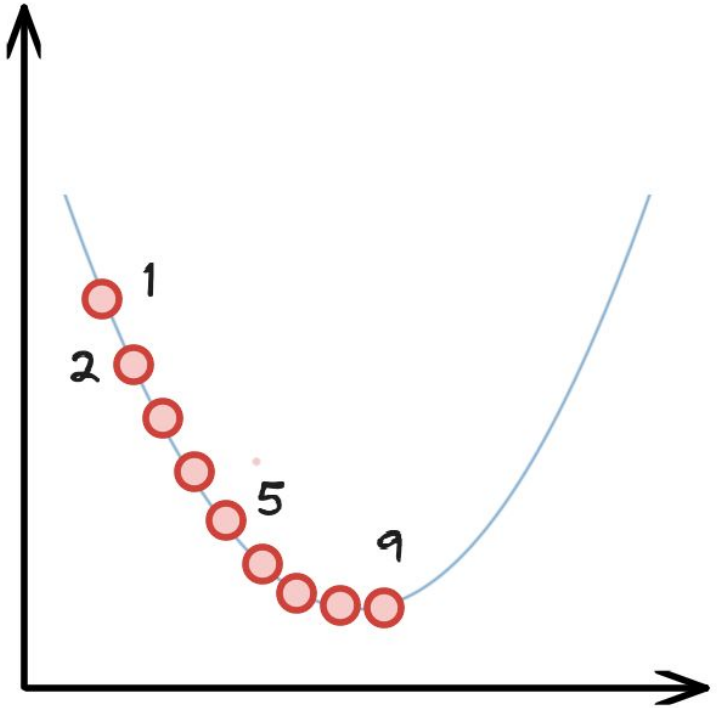


Gradient Boosting

Hiper parámetro learning_rate

***learning_rate* muy bajo**

- Provoca que nos movamos lentamente en la función de pérdida, provocando un tiempo excesivo en el entrenamiento.
- De esta forma podremos alcanzar un mínimo. sin embargo, podemos caer fácilmente en mínimos locales.



/* XGradient Boosting*/

XGradient Boosting

Descripción

- Implementación de Gradient Boosting que acelera el entrenamiento, presenta un alto rendimiento.

- Ha ganado popularidad ya que ha ganado varios concursos de Kaggle

Se recomienda para:

- Conjuntos de alta dimensión, para grandes volúmenes de datos, en que se requiera alta precisión.

¡Manos a la obra!
Aplicando modelos boosting

¡Manos a la obra!

Aplicando modelos boosting

Veremos a continuación cómo implementar modelos Boosting con Python. Para ello, utiliza el archivo **01 - Boosting** y sigue los pasos que te indicará tu profesor.

{desafío} latam_

Desafío
“Detección de cardiopatía”

{desafío}
latam_

Desafío

"Detección de cardiopatía"

- Descarga el archivo "Desafío".
- Tiempo de desarrollo asincrónico: desde 2 horas.
- Tipo de desafío: individual.

¡AHORA TE TOCA A TI! 💪

{desafío} latam_

Ideas fuerza

Ensamble secuencial ***Adaptive Boosting*** *utiliza* ***ponderación de observaciones mal clasificadas*** *para* ***ir entrenando un aprendiz fuerte***

Ensamble secuencial ***Gradient Boosting penaliza los residuos*** *en la función de* ***pérdida*** *potenciando un aprendiz fuerte*

XGradient Boosting *implementación de Gradient Boosting de alto rendimiento computacional, y buen desempeño. Determinar un buen* ***Learning rate*** *es importante para lograr un buen rendimiento.*

Cada aprendiz débil es un paso a la perfección, corrigiendo errores y refinando la predicción.

Próxima sesión...

Entender las problemáticas al entrenar modelos calibrando sobreajuste y subajuste.

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales