{desafío}
latam_

Boosting Ensembles _

Motivación

Preliminares

- **No Free Lunch Theorem**: No existe un algoritmo que presenta una solución satisfactoria para todos los problemas.
- Modelos previamente vistos:
  - Modelos de instancia única.
  - Modelos de ensambles paralelos.
- **Modelos de ensamble secuencial:** Buscan mejorar el desempeño de un **clasificador débil** en consideración a su desempeño anterior.
- Este principio se conoce como **Boosting** (Incremento en el desempeño predictivo).

Ventajas de los ensambles secuenciales

- Un ensamble secuencial permite coleccionar el resultado de un modelo en **múltiples instancias**.
- La principal ventaja de éstos es que permiten el ajuste iterado in situ de cada uno de los modelos.

Veremos dos variantes de boosting:

- Adaptive Boosting
- Gradient Boosting

Ambos se basan en el principio de **"automatizar"** el aprendizaje del clasificador débil, condicional a su tasa de error en la iteración previa.

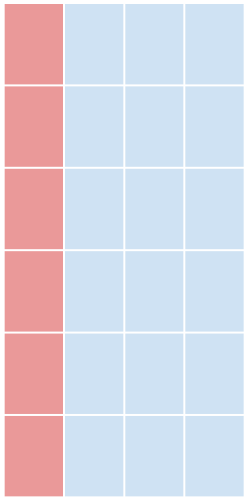
Adaptive Boosting

¿Qué es?

- Adaptive Boosting fue la primera materialización de un ensamble secuencial.
- **Caso canónico:** En base a un problema de clasificación binaria, evaluar la tasa de clasificaciones incorrectas en el modelo.
- Fue conceptualizado desde las ciencias de la computación, posteriormente incorporado en la estadística.
- Esto en consideración a sus múltiples ventajas:
  - Resistente al overfit
  - Produce clasificaciones correctas calibradas.
  - El error en validación tenderá a bajar condicional a la cantidad de iteraciones en el clasificador débil.

Mecanismo de Acción de AdaBoost

$w_i = 1/N$

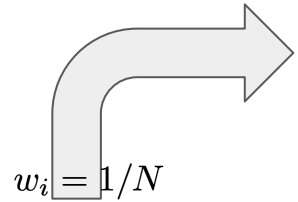


Comencemos con un conjunto de datos y una distribución de pesos **equiprobable.**

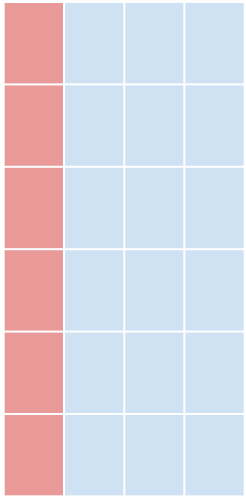
Al hacer equiprobables los pesos, asumimos un escenario donde todas las observaciones tienen la misma tasa de ser predichas correcta/incorrectamente.

Mecanismo de Acción de AdaBoost

$h_m(X_{\text{train}}) \quad \forall m \in M$



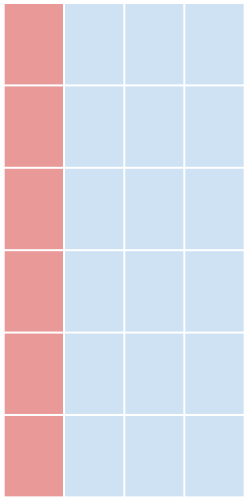
$w_i = 1/N$



Asumiendo un conjunto finito de modelos, entrenamos con su primera iteración.

Mecanismo de Acción de AdaBoost

$w_i = 1/N$



$h_m(X_{\text{train}}) \quad \forall m \in M$

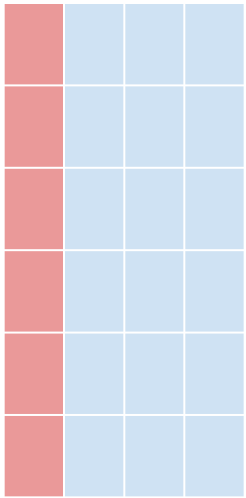
$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

Asumiendo un conjunto finito de modelos, entrenamos con su primera iteración.

En base a este primer modelo entrenado, evaluamos la tasa de error en las clasificaciones incorrectas.

Mecanismo de Acción de AdaBoost

$w_i = 1/N$



$h_m(X_{\text{train}}) \quad \forall m \in M$

$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

$$\alpha_m = \log\left((1 - \bar{\varepsilon_m})/\bar{\varepsilon_m}\right)$$

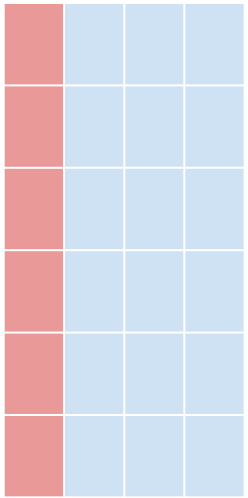
Asumiendo un conjunto finito de modelos, entrenamos con su primera iteración.

En base a este primer modelo entrenado, evaluamos la tasa de error en las clasificaciones incorrectas.

Con la tasa de clasificaciones incorrectas, modelamos el factor de ponderación, asignando mayor peso a las clasificaciones incorrectas.

Mecanismo de Acción de AdaBoost

$w_i = 1/N$



$h_m(X_{\text{train}}) \quad \forall m \in M$

$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

$$\alpha_m = \log\left((1 - \bar{\varepsilon_m})/\bar{\varepsilon_m}\right)$$

$$w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot \mathbb{I}(y_i \neq h_m(x_i))\right] \quad \forall i \in N$$

Asumiendo un conjunto finito de modelos, entrenamos con su primera iteración.

En base a este primer modelo entrenado, evaluamos la tasa de error en las clasificaciones incorrectas.

Con la tasa de clasificaciones incorrectas, modelamos el factor de ponderación, asignando mayor peso a las clasificaciones incorrectas.

Actualizamos los pesos **observacionales** en el conjunto de entrenamiento en función a las observaciones incorrectas y su peso.

Mecanismo de Acción de AdaBoost

$h_m(X_{\text{train}}) \quad \forall m \in M$

w_i

$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

$$\alpha_m = \log\left((1-\bar{\varepsilon_m})/\bar{\varepsilon_m}\right)$$

$$w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot \mathbb{I}(y_i \neq h_m(x_i))\right] \quad \forall i \in N$$

En la segunda iteración del modelo, los ponderadores observacionales en el conjunto de entrenamiento son actualizados en consideración a la tasa de error del modelo en la instancia anterior.

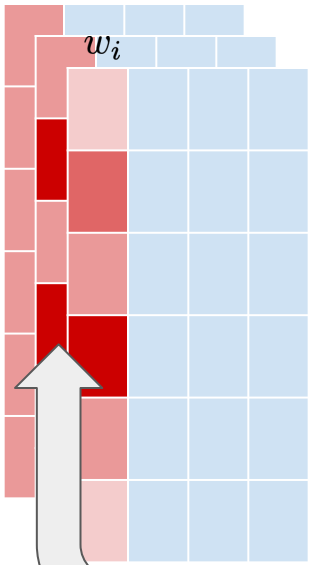
Mecanismo de Acción de AdaBoost

$h_m(X_{\text{train}}) \quad \forall m \in M$

$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

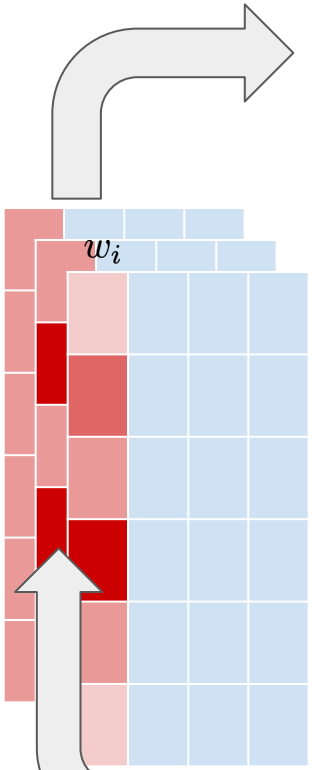
$$\alpha_m = \log\left((1 - \bar{\varepsilon_m})/\bar{\varepsilon_m}\right)$$

$$w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot \mathbb{I}(y_i \neq h_m(x_i))\right] \quad \forall i \in N$$



Iteramos el procedimiento por cada instancia del ensamble, actualizando los pesos observacionales en función del desempeño en su fase anterior.

Mecanismo de Acción de AdaBoost



w_i

$h_m(X_{\text{train}}) \quad \forall m \in M$

$$\bar{\varepsilon}_i = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

$$\alpha_m = \log\left((1 - \bar{\varepsilon_m})/\bar{\varepsilon_m}\right)$$

$$w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot \mathbb{I}(y_i \neq h_m(x_i))\right] \quad \forall i \in N$$

Iteramos el procedimiento por cada instancia del ensamble, actualizando los pesos observacionales en función del desempeño en su fase anterior.

El problema se resume en la sumatoria de la decisión actualizada de cada modelo.

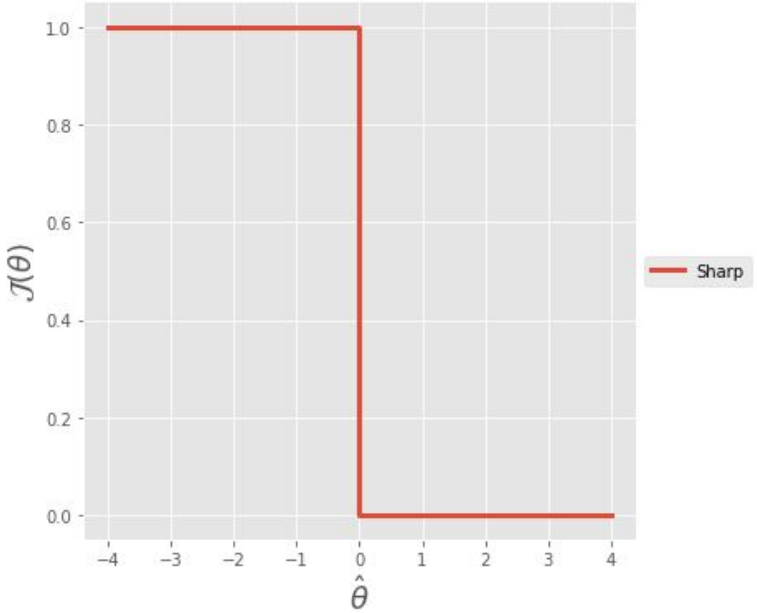
$$\mathcal{H}(x) = \text{sign}\left(\sum_{i=1}^{M} \alpha_m h_m(x)\right)$$

Gradient Boosting

¿Qué es?

- Adaptive Boosting entrena en función a la ponderación de observaciones incorrectamente clasificadas en el modelo anterior.
- Gradient Boosting entrena en función a los errores residuales del modelo anterior.
- Mientras que AdaBoost pondera a nivel de observaciones, Gradient Boosting penaliza en base al error residual.
- El principal elemento a considerar en Gradient Boosting es el hecho que para generar la penalización del error residual, debemos elegir una **función de pérdida**.

Funciones de Pérdida

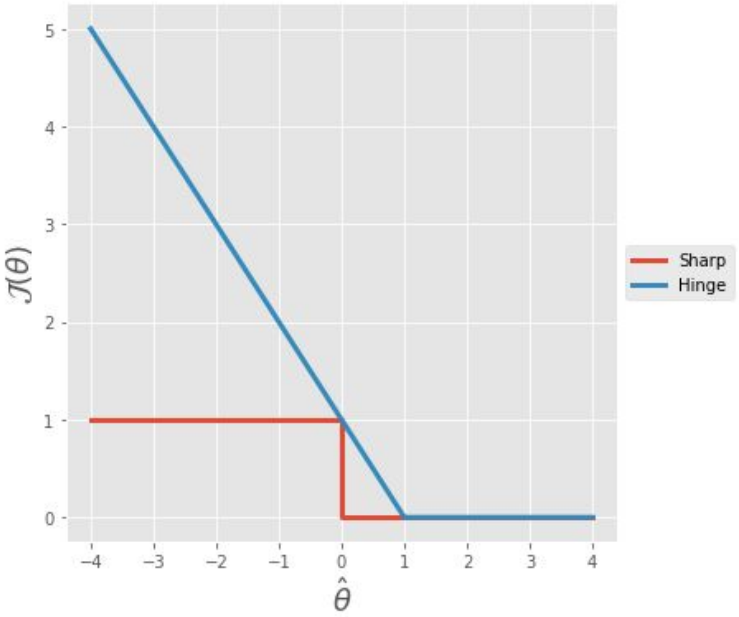


Definición: Mapea qué tan costosa es la elección de un parámetro en cuanto a pérdida de información.

Objetivo: Solucionar un problema de optimización para encontrar un parámetro que minimice la pérdida.

Las variantes responde a la naturaleza del vector objetivo y método de estimación de los parámetros.

Funciones de Pérdida

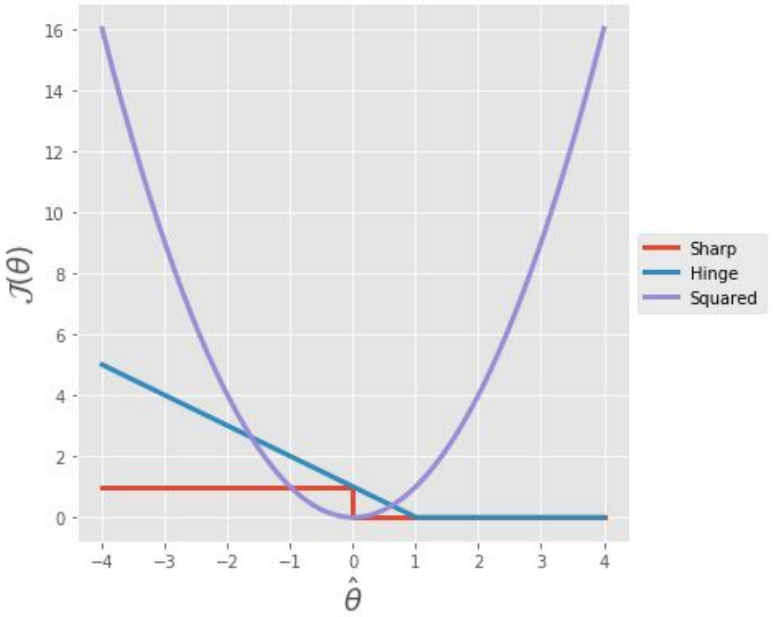


Definición: Mapea qué tan costosa es la elección de un parámetro en cuanto a pérdida de información.

Objetivo: Solucionar un problema de optimización para encontrar un parámetro que minimice la pérdida.

Las variantes responde a la naturaleza del vector objetivo y método de estimación de los parámetros.

Funciones de Pérdida

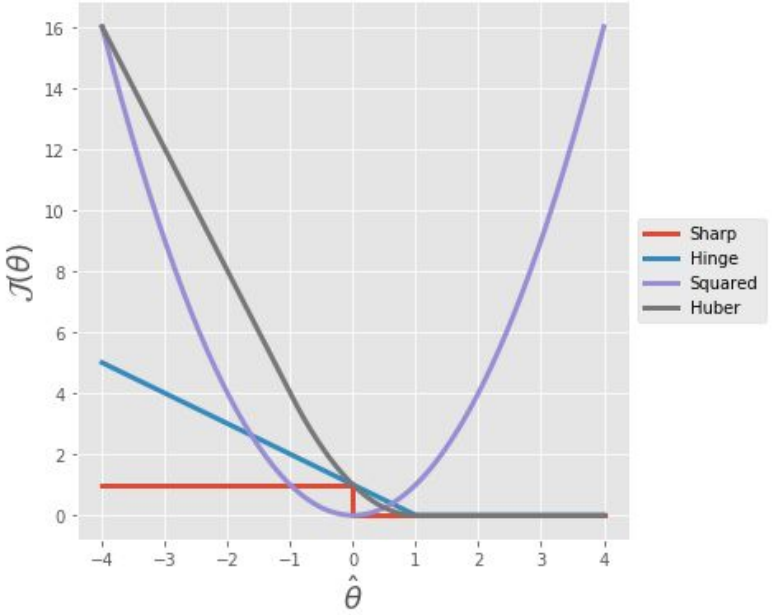


Definición: Mapea qué tan costosa es la elección de un parámetro en cuanto a pérdida de información.

Objetivo: Solucionar un problema de optimización para encontrar un parámetro que minimice la pérdida.

Las variantes responde a la naturaleza del vector objetivo y método de estimación de los parámetros.

Funciones de Pérdida



Definición: Mapea qué tan costosa es la elección de un parámetro en cuanto a pérdida de información.

Objetivo: Solucionar un problema de optimización para encontrar un parámetro que minimice la pérdida.

Las variantes responde a la naturaleza del vector objetivo y método de estimación de los parámetros.

Mecanismo de Acción de Gradient Boosting

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

Iniciamos nuestra función de pérdida en un punto cero.

Nuestro objetivo es generar una minimización de la pérdida.

Mecanismo de Acción de Gradient Boosting

$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

Dado un número fijo de iteraciones M, el objetivo es actualizar de manera secuencial la función de pérdida.

Mecanismo de Acción de Gradient Boosting

$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

Evaluamos el residual del gradiente entre dos puntos dados por la función.

Mecanismo de Acción de Gradient Boosting

$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

En base a este punto, optimizamos un gamma que nos permita reducir la distancia entre el residual del gradiente y la función entrenada.

Por ejemplos prácticos, asumimos que ésta función de pérdida a minimizar será cuadrática.

Mecanismo de Acción de Gradient Boosting

$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}_i, \gamma_m)$$

Generamos una actualización de la función en base a este nuevo parámetro gamma.

Mecanismo de Acción de Gradient Boosting

$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$f_0(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell\big(y_i, \phi(\mathbf{x}_i, \gamma)\big)$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}_i, \gamma_m)$$

Iteramos el procedimiento por cada instancia del ensamble, actualizando la minimización del residual del gradiente.

Mecanismo de Acción de Gradient Boosting

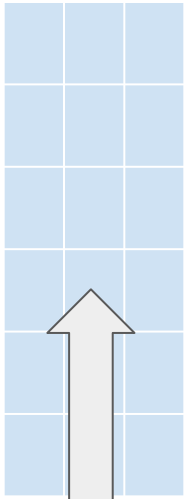
$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}_i, \gamma_m)$$

$$f_1(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell(y_i, \phi(\mathbf{x}_i, \gamma))$$



Iteramos el procedimiento por cada instancia del ensamble, actualizando la minimización del residual del gradiente.

Mecanismo de Acción de Gradient Boosting

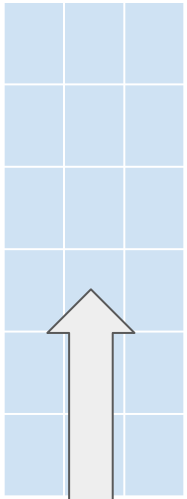
$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}_i, \gamma_m)$$

$$f_2(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell(y_i, \phi(\mathbf{x}_i, \gamma))$$



Iteramos el procedimiento por cada instancia del ensamble, actualizando la minimización del residual del gradiente.

Mecanismo de Acción de Gradient Boosting

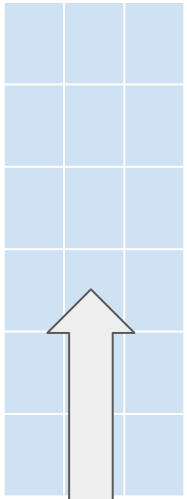
$$h_m(X_{\text{train}}) \quad \forall m \in M$$

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(\mathbf{x}_i))}{\partial f \mathbf{x}_i}\right]_{f(\mathbf{x}_i = f_{m-1}(\mathbf{x}_i))}$$

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i, \gamma_m))^2$$

$$f_m(\mathbf{x}_i) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}_i, \gamma_m)$$

$$f_3(\mathbf{x}) = \underset{\gamma \in \Gamma}{\text{argmin}} \sum_{i=1}^{N} \ell(y_i, \phi(\mathbf{x}_i, \gamma))$$



Iteramos el procedimiento por cada instancia del ensamble, actualizando la minimización del residual del gradiente.

Optimización del Gradiente 101

Algunos elementos a tomar en cuenta

- Como el nombre lo indica, Gradient Boosting busca fortificar el comportamiento del algoritmo en el gradiente (o dirección principal) del error residual.
- Uno de los principales hiperparámetros en Gradient Boosting es `learning_rate`.
- Por learning rate, generamos una norma sobre cómo recorrer la función de pérdida para identificar un parámetro que la minimice.
- Dado que nuestro objetivo es encontrar un mínimo en una función de pérdida, generamos un proceso que busca bajar por ésta.
- Por el momento ignoremos los detalles técnicos asociados a descenso del gradiente.

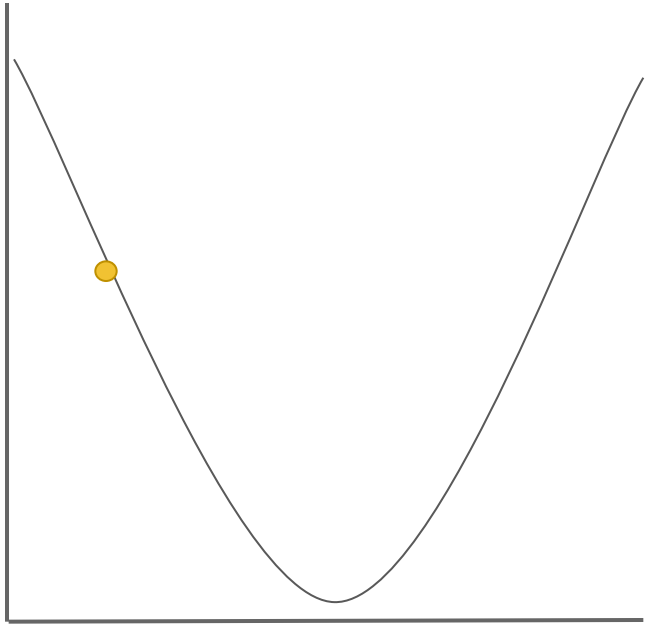
La tasa de aprendizaje

Con la tasa de aprendizaje, definimos los pasos del algoritmo para evaluar el mínimo de la función de pérdida.

Nos encontraremos con dos escenarios yuxtapuestos:

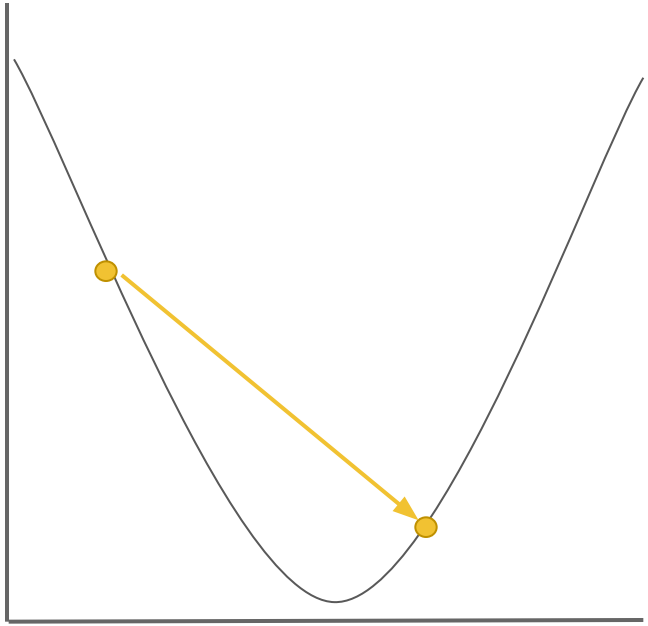
- Pasos grandes en la tasa de aprendizaje.
- Pasos pequeños en la tasa de aprendizaje.

Tasa de Aprendizaje Alta



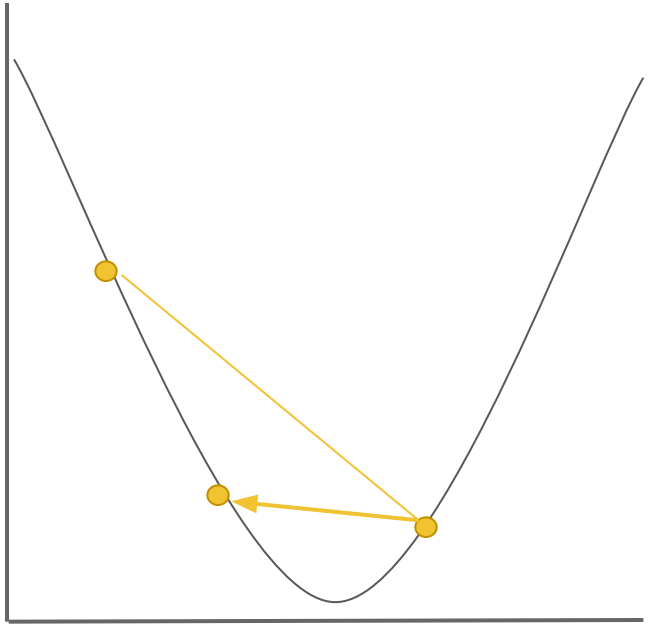
- Tomamos un punto de partida (generalmente aleatorio).

Tasa de Aprendizaje Alta



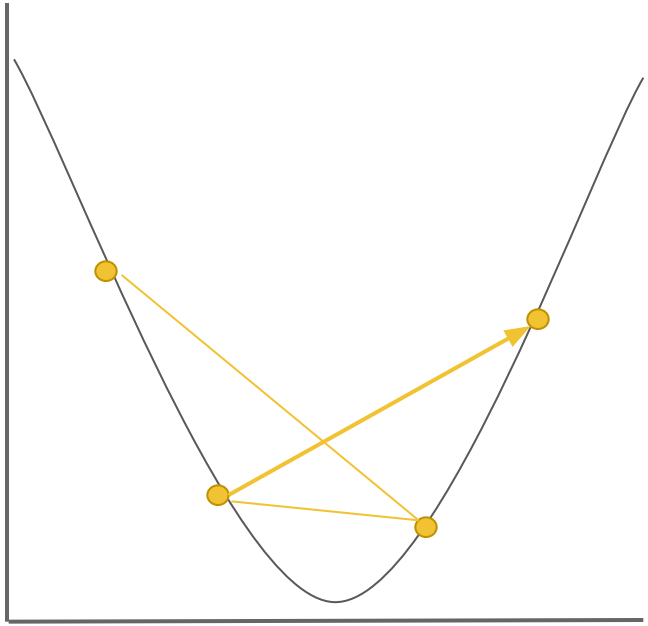
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es alta, podremos recorrer la superficie de la función de pérdida de manera mucho más rápida.

Tasa de Aprendizaje Alta



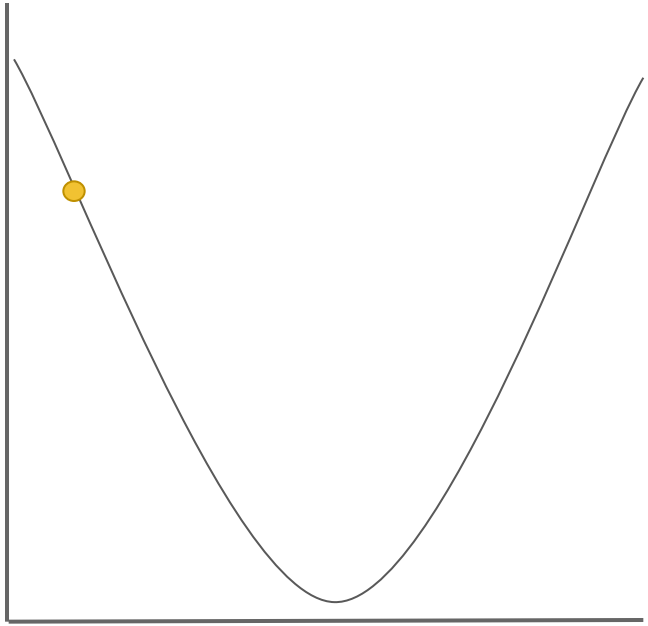
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es alta, podremos recorrer la superficie de la función de pérdida de manera mucho más rápida.
- El problema es que podemos ignorar un óptimo mediante estos pasos grandes.

Tasa de Aprendizaje Alta



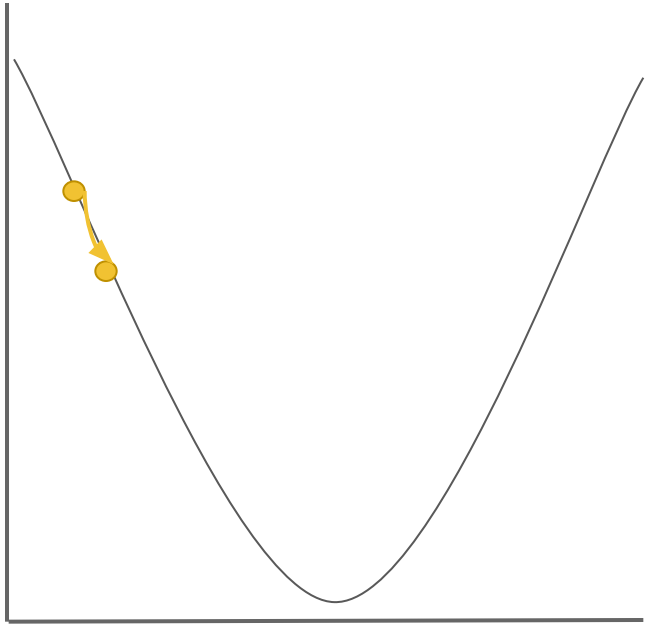
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es alta, podremos recorrer la superficie de la función de pérdida de manera mucho más rápida.
- El problema es que podemos ignorar un óptimo mediante estos pasos grandes.
- También, podemos extrapolar la búsqueda del óptimo en una región de la pérdida que no sea informativa. Esto genera problemas de estabilidad y falta de convergencia.

Tasa de Aprendizaje Baja



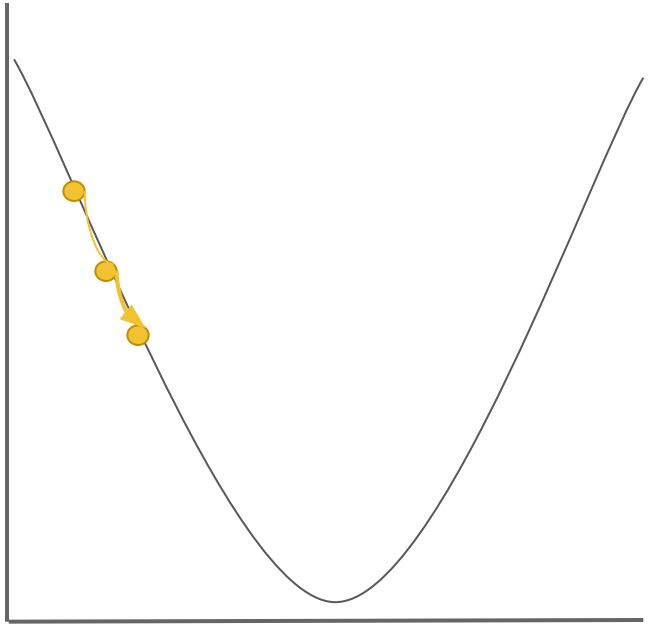
- Tomamos un punto de partida (generalmente aleatorio).

Tasa de Aprendizaje Baja



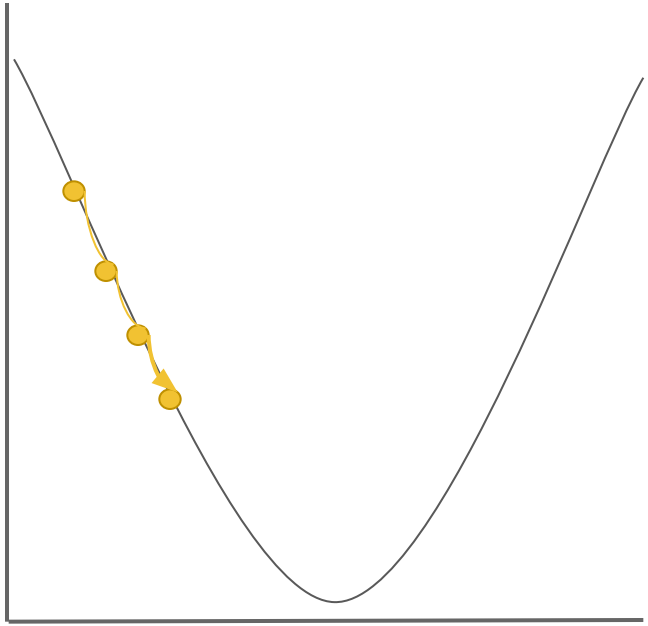
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.

Tasa de Aprendizaje Baja



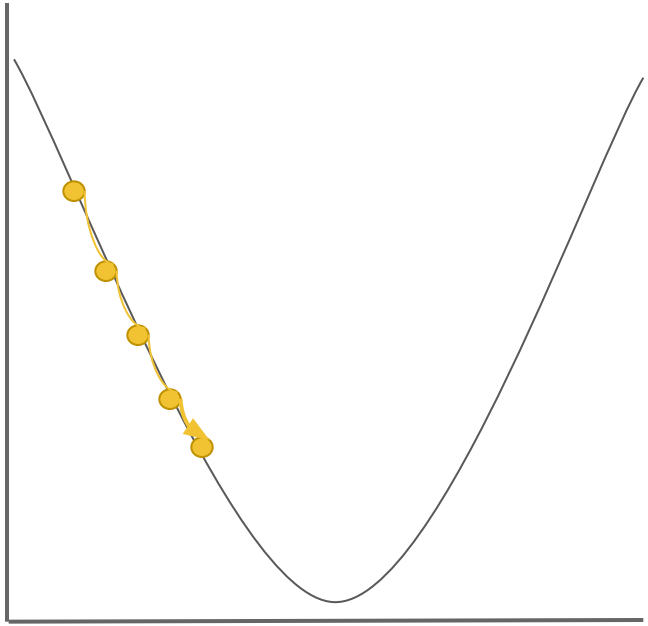
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.

Tasa de Aprendizaje Baja



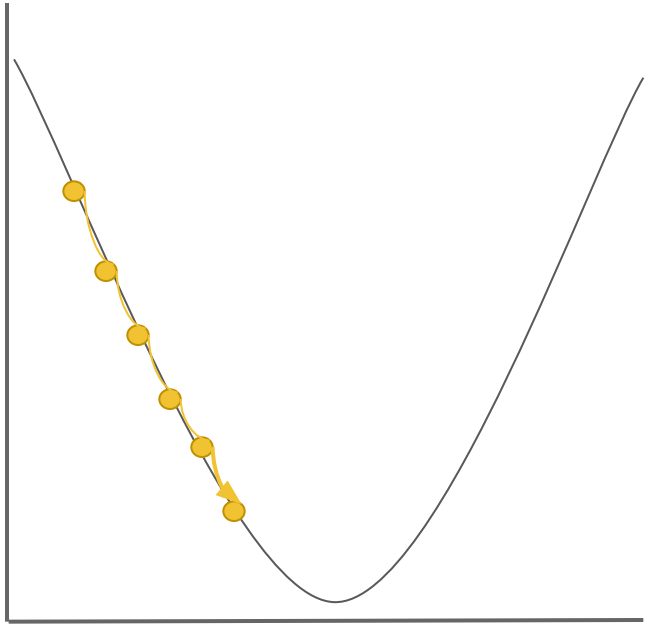
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.

Tasa de Aprendizaje Baja



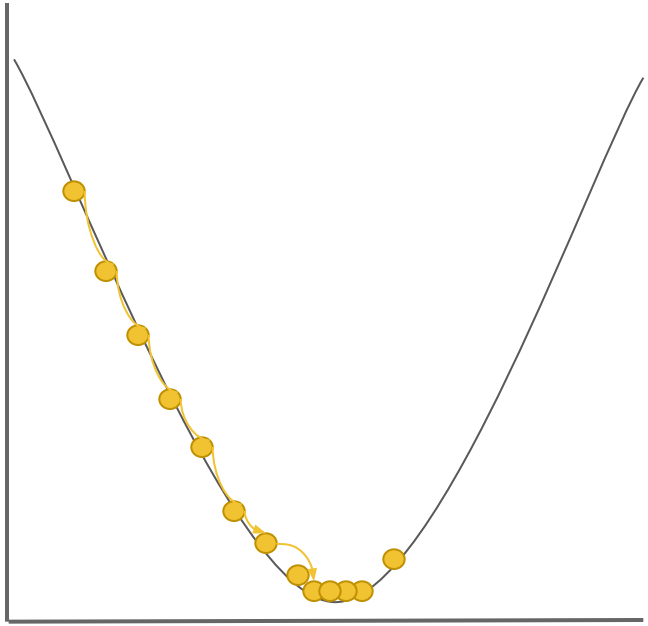
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.

Tasa de Aprendizaje Baja



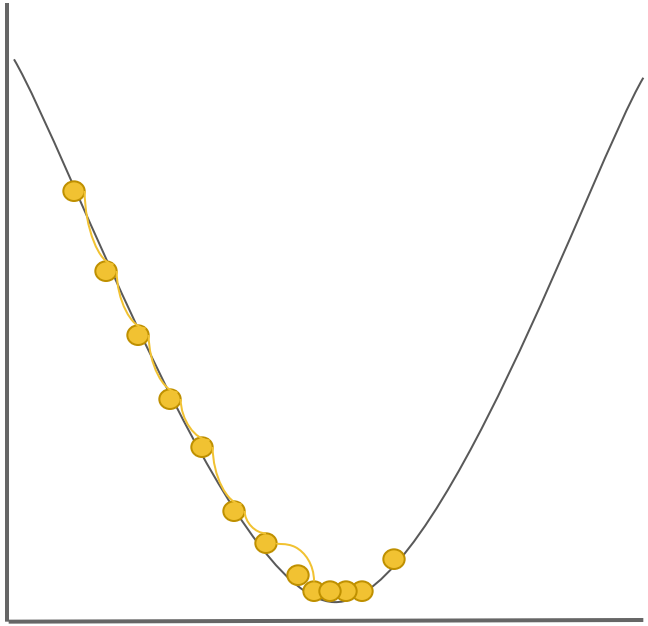
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.

Tasa de Aprendizaje Baja



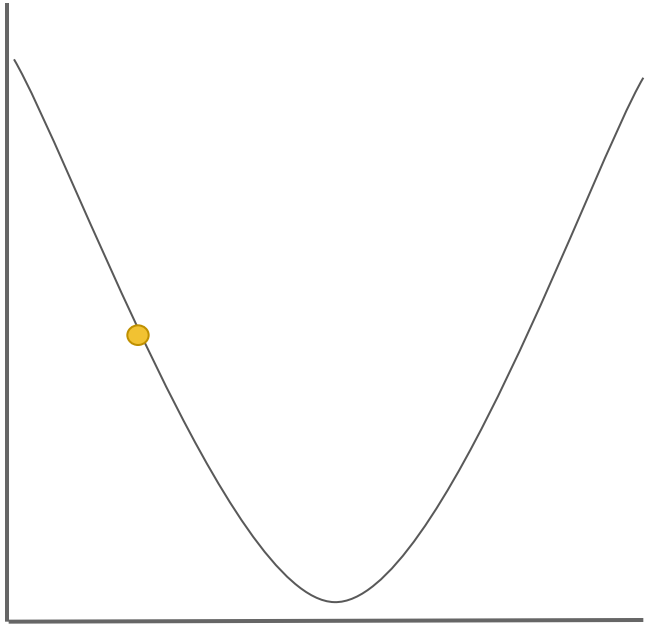
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.
- La ventaja de este punto es que nos asegurará un parámetro óptimo que minimice la pérdida.

Tasa de Aprendizaje Baja



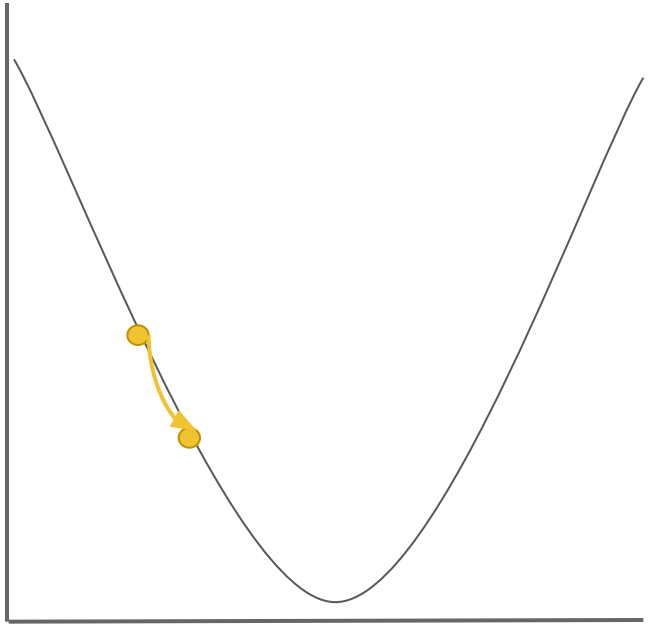
- Tomamos un punto de partida (generalmente aleatorio).
- Cuando la tasa de aprendizaje es baja, recorremos la superficie de la función de pérdida de una manera mucho más lenta.
- La ventaja de este punto es que nos asegurará un parámetro óptimo que minimice la pérdida.
- El problema es que se puede estancar en mínimos locales, y también tiene un tiempo de entrenamiento mucho mayor.

“Goldilocks” Region



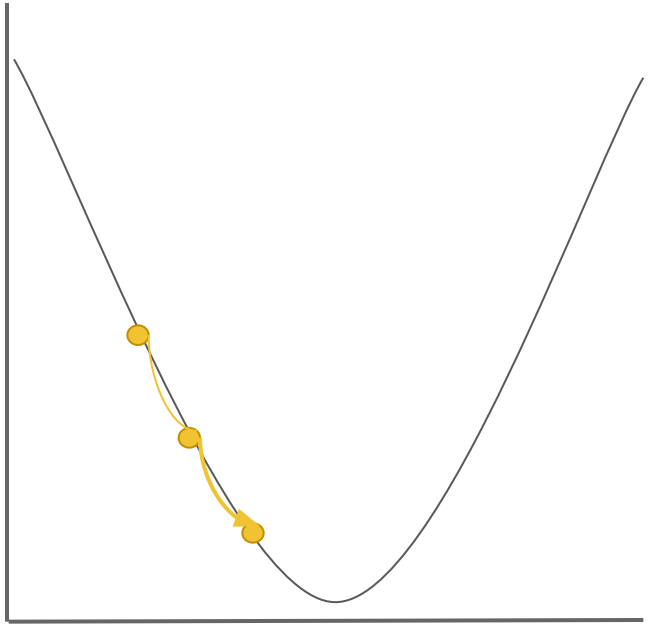
- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.

“Goldilocks” Region



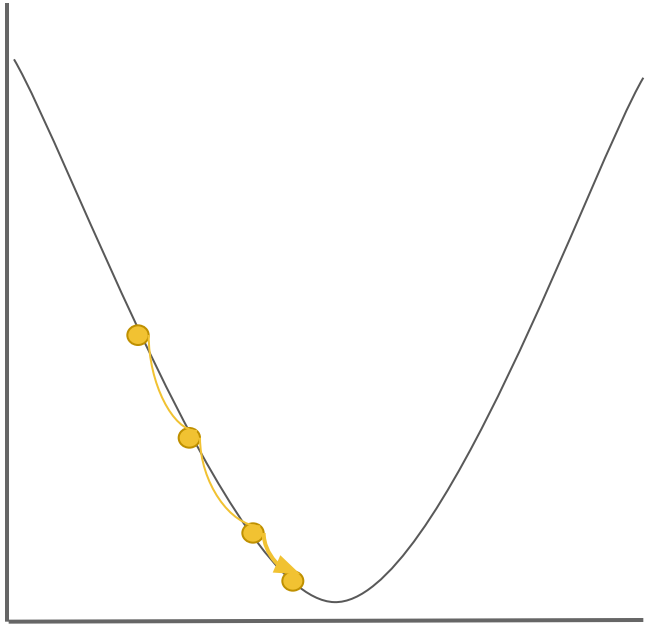
- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.
- Así, lamentablemente no existe una receta única para calibrar la tasa de aprendizaje en un algoritmo secuencial.

“Goldilocks” Region



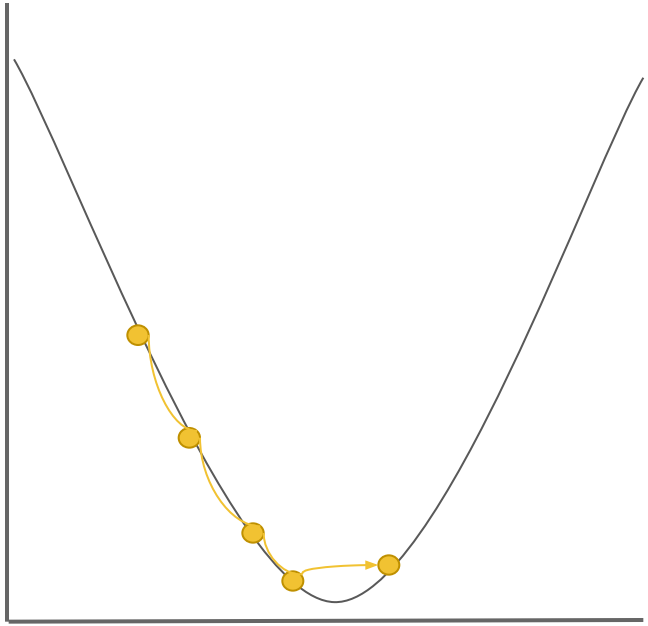
- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.
- Así, lamentablemente no existe una receta única para calibrar la tasa de aprendizaje en un algoritmo secuencial.

“Goldilocks” Region



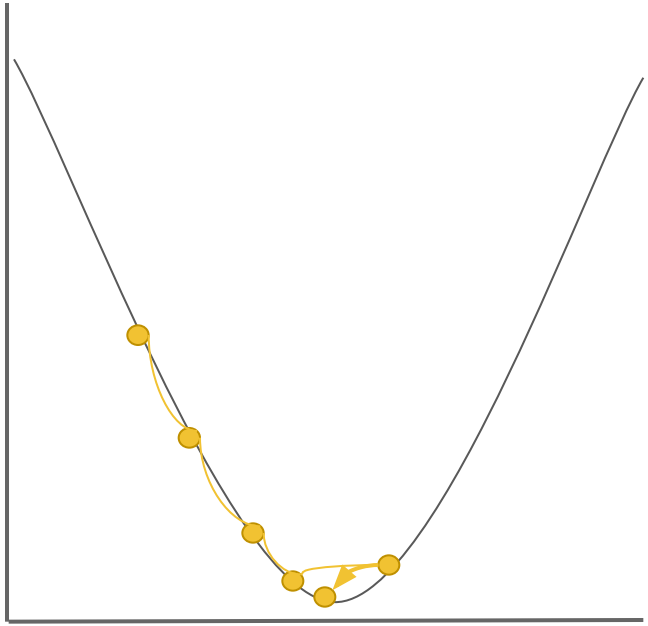
- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.
- Así, lamentablemente no existe una receta única para calibrar la tasa de aprendizaje en un algoritmo secuencial.

“Goldilocks” Region



- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.
- Así, lamentablemente no existe una receta única para calibrar la tasa de aprendizaje en un algoritmo secuencial.

“Goldilocks” Region



- De manera similar a muchos problemas de Machine Learning, nuestro objetivo en la tasa de aprendizaje es encontrar un intermedio que optimice la pérdida con el menor tiempo de ejecución posible.
- Así, lamentablemente no existe una receta única para calibrar la tasa de aprendizaje en un algoritmo secuencial.

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com