Apuntes Semana 4 Clase #2

28/08/2025

Alex Steven Naranjo Masis
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
Email: alnaranjo@estudiantec.cr

***Resumen*—Para esta clase se repasaron temas de la clase anterior como lo son KNN, regresión lineal, Mean Square Error, Descenso del gradiente y un repaso general de derivadas. Y luego del repaso continuamos viendo temas como lo son: Derivadas Parciales con respecto a w y b en la función de pérdida con el fin de actualizarlos y ajustar la función, y por último vimos Epoch y Batch.**

***Index Terms*—KNN, Regresión Lineal, Mean Square Error, MAE, Descenso del Gradiente, Epoch y Batch**

I. Noticas de la Semana

A. *Small Language Models are the Future of Agentic AI*

En el artículo se dice que los modelos de lenguaje pequeños (SLMs) son más adecuados que los grandes (LLMs) para ciertos sistemas inteligentes autónomos (agentic AI), especialmente en tareas especializadas y repetitivas. [1]

B. *Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence*

El estudio analiza cómo la adopción de la inteligencia artificial generativa ha afectado al mercado laboral en EE.UU., utilizando datos administrativos mensuales de nóminas de ADP, el mayor procesador de nóminas del país, el cual abarca millones de trabajadores en decenas de miles de empresas. [2]

II. Repaso Clase Anterior

A. *K Nearest Neighbor (kNN)*

En resumen, cuando obtenemos una nueva instancia, medimos contra todos los elementos del dataset, y tomamos las distancias más cercanas, y en base a eso determinábamos la clase de la nueva instancia.

Contamos con el hiperparámetro K.

Es un algoritmo de lazy learning, porque realmente no se aprende de los datos.

A1. Ventajas:

- Sencillo de implementar.
- Es flexible: Aplica tanto para regresion como clasificación.

A2. Desventajas:

- Las características irrelevantes pueden distorsionar las distancias
- Es computacionalmente costoso.
- Poco eficiente en grandes volúmenes de datos.

B. *Regresión Lineal*

Lo que queremos hacer es encontrar la línea que mejor se ajuste a los datos, para poder realizar una predicción de un valor.

B1. Variables:

- Variables independientes: Son las características de la muestra.
- Variables dependientes: Es el valor a predecir y es afectada por las varibales independientes

Con esto lo que queremos hacer es encontrar un modelo estadístico lineal: $f_{w,b}(X) = wX + b$

Donde:

- X es un vector D-dimensional.
- w es un vector D-dimensional.
- b un número real.
- wX es un producto punto, dándonos como resultado un escalar.

El modelo está parametrizado por **w** y **b**, por lo que debemos encontrar los valores óptimos de **w** y **b** que harán que la función realice las predicciones más precisas. Pero ojo, $Optimo \neq Perfecto$

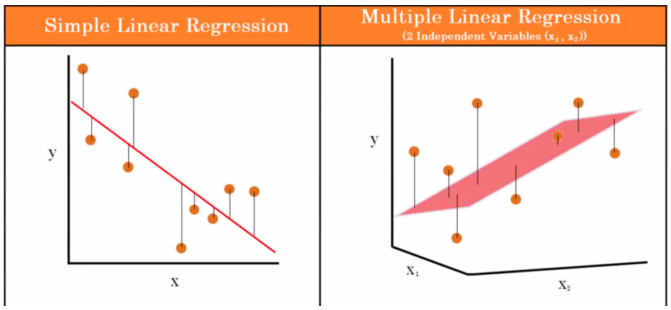

Figura 1. Tipos de Regresión

C. *Función de Pérdida*

Necesitamos de un método qué nos permita cuantificar qué tan bien se ajusta nuestro modelo a los datos. Función de Pérdida = Medida del error del modelo

D. *Error Cuadrático Medio (MSE)*

Es el resultado del modelo contra la etiqueta. Sumamos todos los errores de los samples y lo promediamos.

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(f_{w,b}(x_i) - y_i\right)^2$$

D1. Conceptos Clave):

- **Loss Function:** $\left(f_{w,b}(x_i) - y_i\right)^2$ es la medida de penalidad que cuantifica el error de cada ejemplo.
- **Error Cuadrático:** Penaliza los errores grandes.
- **Cost Function:** Es el promedio de la Loss Function sobre todo el dataset.
- **Objetivo:** Minimizar L para ajustar los parametros **w,b**.

El motivo por el cual queremos minimizar L, es porque entre menor sea L, significa que tenemos un mejor modelo, y entre más grande significa que tenemos un peor modelo.

E. *¿Por qué MSE y no MAE?*

Es debido a qué es cuadrática, y esto nos asegura que vamos a tener un punto mínimo. Y támbien es porque la función no MAE no es smooth, por lo que no nos va a permitir obtener las derivadas en todos los puntos, lo que induce a errores de cálculo

F. *Derivadas Generales*

Regla	**Función** $f(x)$	**Derivada** $f'(x)$
Constante	k	0
Identidad	x	1
Constante multiplicativa	kx	k
Potencia	x^n	nx^{n-1}
Suma	$u(x) + v(x)$	$u'(x) + v'(x)$
Producto	$u(x)v(x)$	$u'(x)v(x) + u(x)v'(x)$
Constante sumada	$u(x) + z$	$u'(x)$
Derivadas parciales	$f(x,y) = 2x + 3y$	$\frac{\partial f}{\partial x} = 2, \frac{\partial f}{\partial y} = 3$

Cuadro I
REPASO DE DERIVADAS BÁSICAS

G. *Descenso del gradiente*

El descenso del gradiente es un algoritmo iterativo de optimización para encontrar el mínimo de una función. Funciona actualizando repetidamente los parámetros en la dirección opuesta al gradiente de la función de costo.

G1. Regla de actualización:

$$x_{\text{nuevo}} = x_{\text{antiguo}} - \alpha \cdot (2x)$$

G2. Importancia del α: Es el learning rate, debe ser pequeño para no pasarnos del punto mínimo. Este es un hiperparámetro

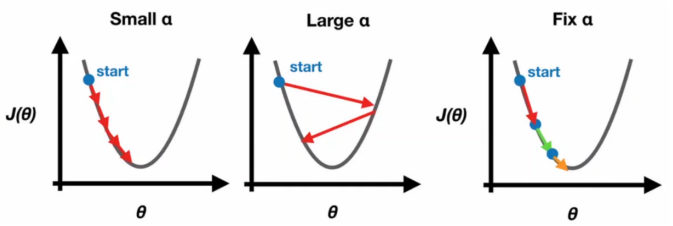

Figura 2. Comparación de distintos valores para alpha

III. CONTENIDO DE LA CLASE

A. *Función de Pérdida y sus Derivadas Parciales*

Para optimizar los parámetros w y b de nuestro modelo, necesitamos actualizar sus valores de manera que la función de pérdida se minimice. Para esto, evaluamos cómo cada parámetro afecta la pérdida utilizando derivadas parciales con respecto a w y b.

Considerando la función de pérdida basada en el error cuadrático medio (MSE) para nuestro modelo lineal $f_{w,b}(x) = wx + b$, tenemos:

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N}\left((wx_i + b) - y_i\right)^2$$

Las derivadas parciales de L con respecto a w y b se calculan como:

$$\frac{\partial L}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}\left((wx_i + b) - y_i\right)x_i$$

$$\frac{\partial L}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}\left((wx_i + b) - y_i\right)$$

Estas derivadas nos indican la dirección y magnitud del ajuste necesario para cada parámetro, permitiendo aplicar algoritmos de optimización como el *gradient descent* para actualizar w y b.

B. *Epoch*

Una **epoch** es una iteración completa sobre todo el conjunto de entrenamiento. Es un hiperparámetro que define cuántas veces se recorrerá el dataset completo durante el entrenamiento, por ejemplo, `epochs = 5`.

Si tenemos 10 000 muestras y ejecutamos 5 epochs, significa que se procesarán todas las muestras 5 veces en total. La actualización de los parámetros puede realizarse al finalizar cada epoch o de manera más frecuente utilizando batches.

C. *Batch*

Un **batch** es un subconjunto del conjunto de entrenamiento que se utiliza para calcular la gradiente y actualizar los parámetros del modelo.

Por ejemplo, si tenemos 10 000 muestras y un `batch size = 1 000`, necesitaremos 10 batches para completar

una epoch. Cada batch permite calcular la gradiente y actualizar los parámetros sin esperar a procesar todo el dataset. Dependiendo de la estrategia, se puede actualizar los parámetros después de cada batch o acumular gradientes antes de la actualización.

C1. Batch Gradient Descent (Vanilla): El **Batch Gradient Descent** calcula la gradiente utilizando todo el dataset:

$$\nabla L = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L}{\partial \theta_i}$$

y actualiza los parámetros solo después de procesar el conjunto completo.

Ventajas:

- Gradiente estable y pasos consistentes.
- Ayuda a evitar mínimos locales y aporta robustez en la optimización.

Desventajas:

- Requiere todo el dataset en memoria.
- Las actualizaciones son lentas para datasets grandes.
- La gradiente muy estable puede ocultar señales útiles.

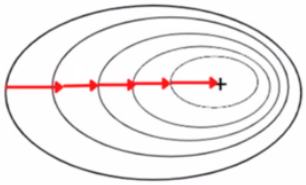

Figura 3. Batch Gradient Descent

C2. Stochastic Gradient Descent (SGD): El **Stochastic Gradient Descent** actualiza los parámetros después de cada muestra del dataset (o un pequeño conjunto aleatorio de muestras).

Ventajas:

- Detecta rápidamente si el algoritmo puede converger.
- Útil para datasets muy grandes.

Desventajas:

- Las actualizaciones pueden ser muy ruidosas.
- La trayectoria de los parámetros es oscilatoria.
- Muchas actualizaciones pueden ser costosas computacionalmente.

$$w \leftarrow w - \alpha \frac{\partial L}{\partial w}$$

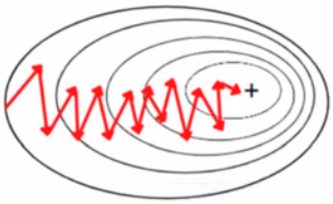

Figura 4. Stochastic Gradient Descent

C3. Mini-batch Gradient Descent: El **Mini-batch Gradient Descent** combina las estrategias anteriores: calcula la gradiente sobre batches de tamaño intermedio.

Ventajas:

- Reduce el ruido respecto a SGD y es más estable.
- Más eficiente que Batch GD.
- Mejora la explotación de hardware (vectorización, GPUs).

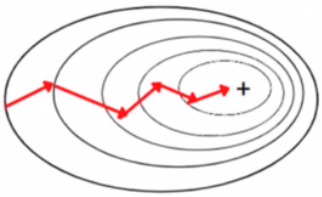

Figura 5. Mini-batch Gradient Descent

REFERENCIAS

[1] Belcak, P., et al, "Small Language Models are the Future of Agentic AI"2025.

[2] E. Brynjolfsson et al., "Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence" 2025.