UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

Inteligencia Artificial

Ing. Juancarlos Santana Huamán
jsantana@ucss.edu.pe

Backpropagation en Inteligencia Artificial

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

¿Qué es Backpropagation o retropropagación en IA?

- Backpropagation es un algoritmo de aprendizaje supervisado en inteligencia artificial utilizado para **entrenar redes neuronales artificiales** (ANNs). El objetivo de Backpropagation es ajustar los pesos y sesgos de una red neuronal para minimizar la diferencia entre las salidas de la red y las salidas deseadas o esperadas durante el entrenamiento.
- El algoritmo utiliza una técnica de optimización matemática llamada gradiente descendente, que permite ajustar los pesos y sesgos de la red para minimizar el error. El proceso de ajuste se realiza en dos fases: en la **propagación hacia adelante** (forward propagation), se calcula la salida de la red para una entrada dada, y en la **propagación hacia atrás** (back propagation) se ajustan los pesos y sesgos de la red para minimizar el error.

¿Cómo funciona el algoritmo Backpropagation (retropropagación)?

- El algoritmo de retropropagación, también conocido como Backpropagation, es un método utilizado para entrenar redes neuronales supervisadas. Este algoritmo funciona en varias etapas:
  - **Primero**, se aplica un vector de entrada a la red y se calcula la salida correspondiente. Este proceso se conoce como "*forward pass*" o "*feed forward*" porque se introducen los inputs a la red, "de izquierda hacia la derecha", realizando todos los cálculos hasta obtener la primera estimación de las salidas de la red.
  - **Segundo**, se compara la salida obtenida con la salida deseada y se calcula el error. Este error es la diferencia entre las salidas de la red y las salidas deseadas o esperadas durante el entrenamiento.
  - **Tercero**, el algoritmo de retropropagación utiliza el error calculado para ajustar los pesos sinápticos en cada una de las capas intermedias. Este proceso se lleva a cabo desde la última capa hasta la primera capa, propagando el error hacia atrás y actualizando los pesos sinápticos a lo largo del camino.
  - **Finalmente**, para actualizar los pesos sinápticos de las neuronas, se utiliza un método conocido como descenso del gradiente (*gradient descent*). El objetivo es encontrar los valores óptimos para los pesos que minimizan el error en la red neuronal. Para lograr esto, se calcula la derivada parcial del error con respecto a cada peso y se actualiza en consecuencia.
- Es importante mencionar que a pesar de ser uno de los algoritmos más populares en el campo de las redes neuronales supervisadas, existen algunos **problemas asociados con el algoritmo de retropropagación**.
- El más comúnes es el **sobreajuste (overfitting)**, que ocurre cuando una red neuronal se ajusta demasiado bien a los datos específicos utilizados para entrenarla y no generaliza bien a nuevos datos.

Funciones de Backpropagation en el entrenamiento de redes neuronales

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- El algoritmo de retropropagación, también conocido como Backpropagation, tiene varias funciones clave en el entrenamiento de redes neuronales artificiales:
  - **Minimizar los errores**: El objetivo principal de Backpropagation es minimizar los errores en el proceso de aprendizaje automático de una máquina. Trata de imitar el funcionamiento de las redes de neuronas biológicas.
  - **Ajustar los parámetros**: Backpropagation ajusta los parámetros (pesos y sesgos) para reducir la tasa de error. Después de cada paso hacia adelante a través de una red, Backpropagation realiza un paso hacia atrás mientras ajusta los parámetros del modelo (pesos y sesgos).
  - **Detectar responsabilidades**: Lo que hace Backpropagation es detectar cuánta responsabilidad tiene cada uno de los nodos de la red en los posibles errores y corregir la configuración de los parámetros de esos nodos, que luego influirán en otros nodos y capas posteriores.
  - **Mejorar la precisión de las predicciones**: El objetivo del algoritmo Backpropagation es minimizar los errores en el proceso de aprendizaje y mejorar la precisión de las predicciones ajustando los pesos y sesgos de cada nodo de la red.
- En resumen, Backpropagation es un algoritmo que se emplea para entrenar redes neuronales artificiales con el objetivo de minimizar los errores en el proceso de aprendizaje automático de una máquina.

Usos de Backpropagation en la actualidad

- El algoritmo de retropropagación, o Backpropagation, tiene una amplia gama de aplicaciones en diversos campos. Se utiliza en el reconocimiento óptico de caracteres, la **conversión de texto escrito a voz**, el procesado de imágenes médicas y la inspección automática de defectos.
- Además, Backpropagation puede ser una herramienta útil en la i**dentificación de patrones y tendencias en grandes conjuntos de datos de tráfico web y consumo de contenido**, y en la optimización de la experiencia de usuario y la generación de contenido innovador y relevante para una audiencia específica.
- En general, el algoritmo de retropropagación se emplea para **entrenar redes neuronales artificiales** con el objetivo de minimizar los errores en el proceso de aprendizaje automático de una máquina. Por lo tanto, cualquier aplicación que implique el uso de redes neuronales puede beneficiarse del uso de Backpropagation.

Aplicaciones del algoritmo Backpropagation en Content marketing y SEO

- Backpropagation tiene aplicaciones en Content marketing y SEO, algunas de ellas pueden ser:
  - **Análisis de palabras clave**: Al entrenar una red neuronal utilizando Backpropagation con grandes cantidades de datos de palabras clave, se puede identificar patrones y tendencias en el comportamiento del usuario y las preferencias de búsqueda relacionadas con ciertos temas o palabras clave. Esto puede ayudar a las empresas de SEO a diseñar estrategias de optimización de búsqueda más efectivas.
  - **Personalización de publicidad digital**: Utilizando la propagación hacia adelante, se puede predecir el comportamiento del usuario y personalizar anuncios digitales en función de las necesidades y preferencias de cada usuario. Esto puede mejorar la eficacia de la publicidad digital y aumentar el retorno de inversión (ROI) de las campañas de publicidad.
  - **Análisis y predicción de tendencias**: Utilizando Backpropagation para entrenar una red neuronal con grandes cantidades de datos de tráfico web y consumo de contenido, se pueden identificar patrones y tendencias en el comportamiento del usuario. Esto puede ayudar a las empresas de Content marketing a diseñar estrategias de contenido más efectivas y a adaptar su contenido a las necesidades e intereses de su audiencia.
- En general, Backpropagation puede ser una herramienta útil en la identificación de patrones y tendencias en grandes conjuntos de datos de tráfico web y consumo de contenido, y en la optimización de la experiencia de usuario y la generación de contenido innovador y relevante para una audiencia específica.

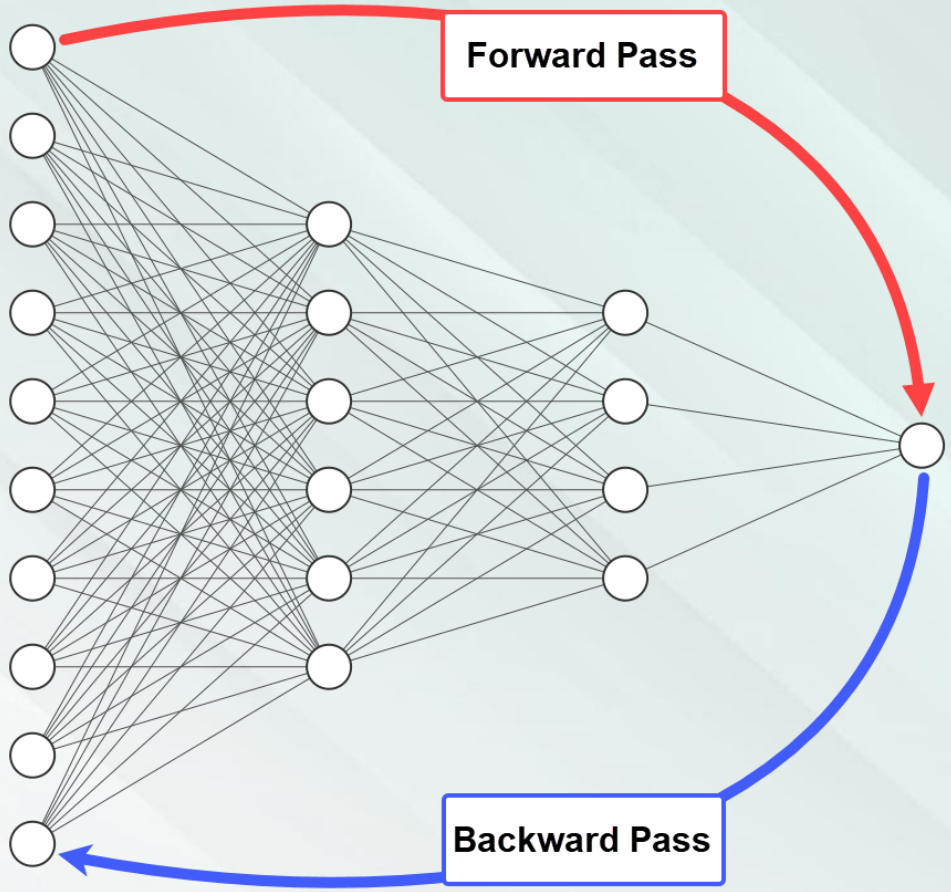
Pases hacia adelante y hacia atrás en redes neuronales

- Para entrenar una red neuronal, hay 2 pasos (fases):
  - Adelante
  - Hacia atrás
- En el paso hacia adelante, comenzamos propagando las entradas de datos a la capa de entrada, pasamos por las capas ocultas, medimos las predicciones de la red desde la capa de salida y, finalmente, calculamos el error de la red en función de las predicciones que hizo la red.
- Este error de red mide la distancia a la red para realizar la predicción correcta. Por ejemplo, si la salida correcta es 4 y la predicción de la red es 1,3, el error absoluto de la red es 4 - 1,3 = 2,7. Cabe destacar que el proceso de propagación de las entradas desde la capa de entrada a la capa de salida se denomina **propagación hacia adelante** . Una vez calculado el error de red, la fase de propagación hacia adelante finaliza y comienza la propagación hacia atrás.

Pases hacia adelante y hacia atrás en redes neuronales

- La siguiente figura muestra una flecha roja que apunta en la dirección de la propagación hacia adelante.
- En el paso hacia atrás, el flujo se invierte, de modo que comenzamos propagando el error a la capa de salida hasta llegar a la capa de entrada, pasando por las capas ocultas. El proceso de propagación del error de red desde la capa de salida a la capa de entrada se denomina **propagación hacia atrás** o **retropropagación** simple . El algoritmo de retropropagación es el conjunto de pasos que se utilizan para actualizar los pesos de la red y reducir el error de red.
- En la siguiente figura, la flecha azul apunta en la dirección de propagación hacia atrás.
- Las fases de avance y retroceso se repiten a partir de algunas épocas. En cada época, ocurre lo siguiente:
- Las entradas se propagan desde la capa de entrada a la de salida.
- Se calcula el error de red.
- El error se propaga desde la capa de salida a la capa de entrada.

Pases hacia adelante y hacia atrás en redes neuronales



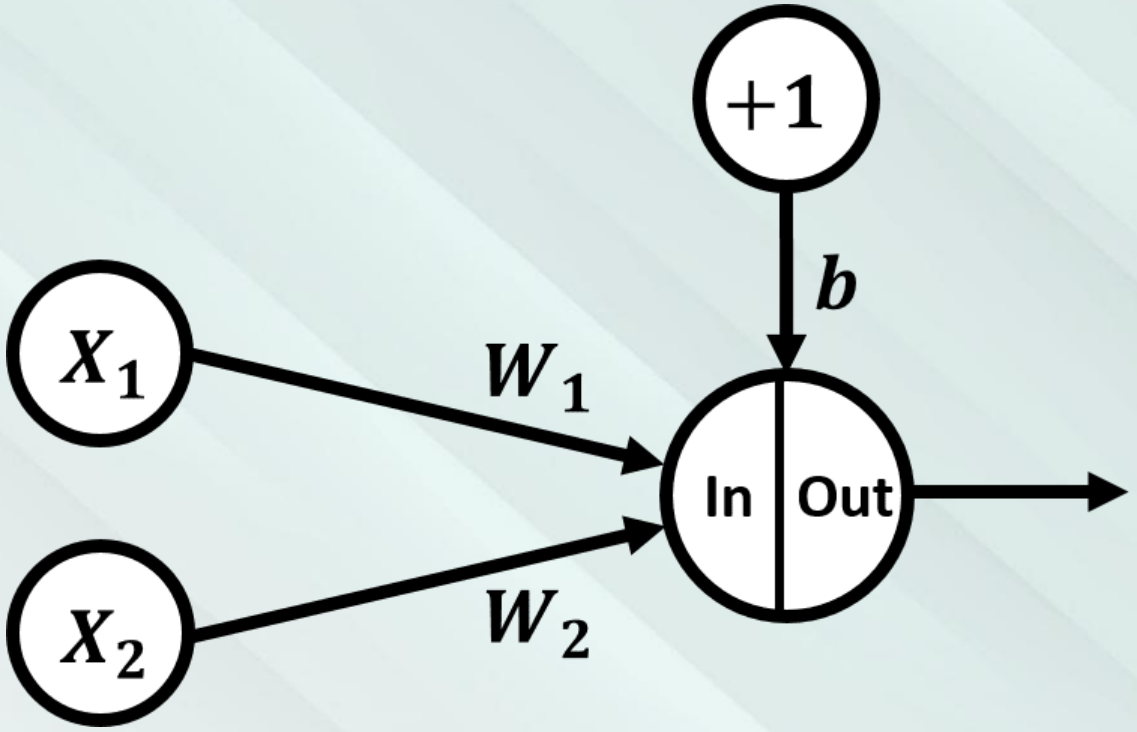
¿Por qué utilizar el algoritmo de retropropagación?

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Anteriormente, explicamos que la red se entrena mediante dos pasadas: una hacia adelante y otra hacia atrás. Al final de la pasada hacia adelante, se calcula el error de la red, que debe ser lo más bajo posible.
- Si el error actual es alto, la red no aprendió correctamente de los datos. ¿Qué significa esto? Significa que el conjunto actual de ponderaciones no es lo suficientemente preciso como para reducir el error de red y realizar predicciones precisas. Por lo tanto, debemos actualizar las ponderaciones de la red para reducir el error.
- El algoritmo de retropropagación es uno de los encargados de actualizar los pesos de la red con el objetivo de reducir el error de red. Es muy importante.
- Estas son algunas de las ventajas del algoritmo de retropropagación:
  - Es eficiente en el uso de memoria al calcular las derivadas, ya que utiliza menos memoria que otros algoritmos de optimización, como el algoritmo genético. Esta es una característica muy importante, especialmente en redes grandes.
  - El algoritmo de retropropagación es rápido, especialmente para redes pequeñas y medianas. A medida que se añaden más capas y neuronas, se vuelve más lento a medida que se calculan más derivadas.
  - Este algoritmo es lo suficientemente genérico para funcionar con diferentes arquitecturas de red, como redes neuronales convolucionales, redes generativas antagónicas, redes totalmente conectadas y más.
  - No hay parámetros para ajustar el algoritmo de retropropagación, por lo que la sobrecarga es menor. Los únicos parámetros del proceso están relacionados con el algoritmo de descenso de gradiente, como la tasa de aprendizaje.

Cómo funciona el algoritmo de retropropagación

- El funcionamiento del algoritmo se explica mejor con una red simple, como la que se muestra en la siguiente figura. Solo tiene una capa de entrada con dos entradas (X^- y X^-) y una capa de salida con una salida. No hay capas ocultas.
- Los pesos de las entradas son W_1 y W_2 , respectivamente. El sesgo se considera una nueva neurona de entrada para la neurona de salida, la cual tiene un valor fijo +1 y un peso b. Tanto los pesos como los sesgos podrían denominarse **parámetros**.



Cómo funciona el algoritmo de retropropagación

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Supongamos que la capa de salida utiliza la función de activación sigmoidea definida por la siguiente ecuación:

$$f(s) = \frac{1}{1 + e^{-s}}$$

- Donde s es la suma de productos (SOP) entre cada entrada y su peso correspondiente:

$$s = X_1 \times W_1 + X_2 \times W_2 + b$$

- Para simplificar, en este ejemplo se utiliza una sola muestra de entrenamiento. La siguiente tabla muestra la muestra de entrenamiento con la entrada y su correspondiente salida deseada (es decir, correcta). En la práctica, se utilizan más ejemplos de entrenamiento.

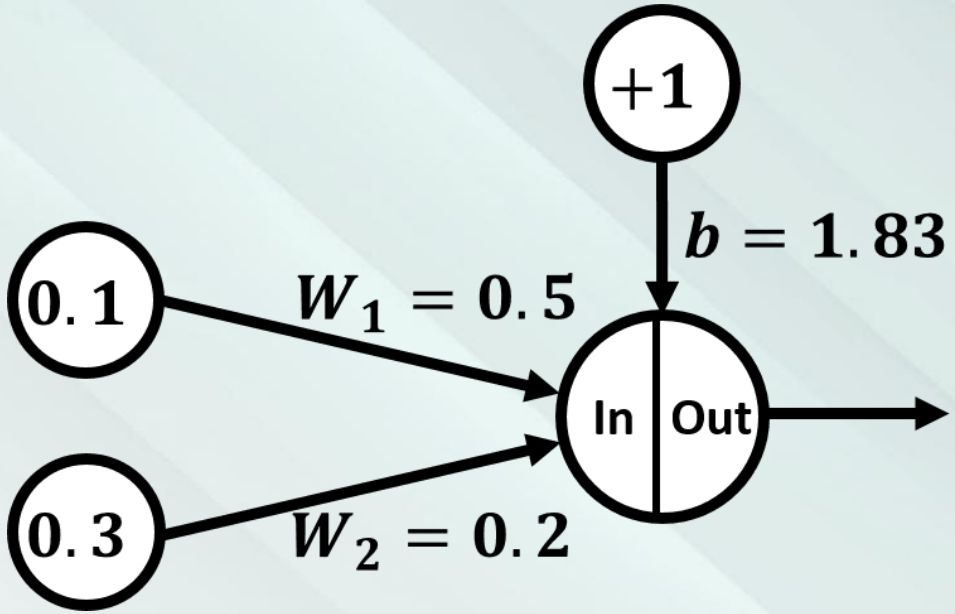
X_1	X_2	Salida deseada
0.1	0.3	0.03

Cómo funciona el algoritmo de retropropagación

- Supongamos que los valores iniciales tanto para los pesos como para el sesgo son como en la siguiente tabla.

W_1	W_2	b
0.5	0.2	1.83

- Para simplificar, los valores de todas las entradas, pesos y sesgos se agregarán al diagrama de red.
- Ahora, entrenemos la red y veamos cómo la red predecirá la salida de la muestra en función de los parámetros actuales.
- Como comentamos anteriormente el proceso de entrenamiento tiene 2 fases, avance y retroceso.



Pase hacia adelante

- La entrada de la función de activación será el SOP entre cada entrada y su peso. El SOP se suma al sesgo para obtener la salida de la neurona:

$$s=X_1 * W_1 + X_2 * W_2 + b$$

$$s=0,1*0,5+0,3*0,2+1,83$$

$$s=1,94$$

- Luego se aplica el valor 1,94 a la función de activación (sigmoidea), lo que da como resultado el valor 0,874352143.

$$f(s) = \frac{1}{1+e^{-s}}$$

$$f(s) = \frac{1}{1+e^{-1.94}}$$

$$f(s) = \frac{1}{1+0.143703949} \qquad f(s) = 0.874352143$$

Pase hacia adelante

- La salida de la función de activación de la neurona de salida refleja la salida predicha de la muestra. Es obvio que existe una diferencia entre la salida deseada y la esperada. Pero ¿por qué? ¿Cómo podemos acercar la salida predicha a la deseada? Responderemos estas preguntas más adelante. Por ahora, veamos el error de nuestra red basado en una función de error.
- Las funciones de error indican la proximidad entre los resultados previstos y los deseados. El valor óptimo de error es **cero** , lo que significa que no hay error alguno y que tanto los resultados deseados como los previstos son idénticos. Una de estas funciones es la **función de error al cuadrado** , definida en la siguiente ecuación:
- Tenga en cuenta que el valor 12 multiplicado por la ecuación es para simplificar el cálculo de las derivadas utilizando el algoritmo de retropropagación.

Pase hacia adelante

- Basándonos en la función de error, podemos medir el error de nuestra red de la siguiente manera:

$$E = \frac{1}{2}(desired - predicted)^2$$

$$E = \frac{1}{2}(0.03 - 0.874352143)^2$$

$$E = \frac{1}{2}(-0.844352143)^2$$

$$E = \frac{1}{2}(0.712930542)$$

$$E = 0.356465271$$

- El resultado muestra que hay un error, y uno grande: (**~0.357**). Este error simplemente nos da una indicación de la distancia entre los resultados previstos y los deseados.
- Sabiendo que hay un error, ¿qué debemos hacer? Debemos minimizarlo. Para minimizar el error de red, debemos cambiar algo en ella. Recuerde que los únicos parámetros que podemos cambiar son los pesos y los sesgos. Podemos probar diferentes pesos y sesgos y luego probar nuestra red.
- Calculamos el error, luego finaliza el pase hacia adelante y debemos comenzar el **pase hacia atrás** para calcular las derivadas y actualizar los parámetros.
- Para sentir prácticamente la importancia del algoritmo de retropropagación, intentemos actualizar los parámetros directamente sin usar este algoritmo.

Ecuación de actualización de parámetros

- Los parámetros se pueden cambiar según la siguiente ecuación:
- $W_{(n+1)} = W(n) + \eta[d(n) - Y(n)]X(n)$
- Dónde:
- n: Paso de entrenamiento (0, 1, 2, …).
- W(n): Parámetros del paso de entrenamiento actual. Wn=[bn,W1(n),W2(n),W3(n),…, Wm(n)]
- η : Tasa de aprendizaje con un valor entre 0,0 y 1,0.
- d(n): Salida deseada.
- Y(n): Salida prevista.
- X(n): Entrada actual en la que la red realizó una predicción falsa.

- Para nuestra red, estos parámetros tienen los siguientes valores:
  - n: 0
  - W(n): [1,83, 0,5, 0,2]
  - η : Debido a que es un hiperparámetro, podemos elegirlo 0,01, por ejemplo.
  - d(n): [0,03].
  - Y(n): [0,874352143].
  - X(n): [+1, 0.1, 0.3]. El primer valor (+1) es para el sesgo.

Ecuación de actualización de parámetros

- Podemos actualizar nuestros parámetros de red de la siguiente manera:
  - $W_{(n+1)} = W(n) + \eta[d(n) - Y(n)]X(n)$
  - *=[1,83, 0,5, 0,2]+0,01[0,03-0,874352143][+1, 0,1, 0,3]*
  - *=[1,83, 0,5, 0,2]+0,01[-0,844352143][+1, 0,1, 0,3]*
  - *=[1,83, 0,5, 0,2]+-0,00844352143[+1, 0,1, 0,3]*
  - *=[1,83, 0,5, 0,2]+[-0,008443521, -0,000844352, -0,002533056]*
  - *=[1.821556479, 0.499155648, 0.197466943]*
- Los nuevos parámetros se enumeran en la siguiente tabla:

W1nuevo	W2new	nuevo
0,197466943	0,499155648	1.821556479

Ecuación de actualización de parámetros

- Con base en los nuevos parámetros, recalcularemos la salida predicha. Esta nueva salida predicha se utiliza para calcular el nuevo error de red. Los parámetros de red se actualizan según el error calculado. El proceso continúa actualizando los parámetros y recalculando la salida predicha hasta alcanzar un valor aceptable para el error.
- Aquí, actualizamos correctamente los parámetros sin usar el algoritmo de retropropagación. ¿Aún necesitamos ese algoritmo? Sí. Ya verás por qué.
- La ecuación de actualización de parámetros depende únicamente de la tasa de aprendizaje para actualizar los parámetros. Cambia todos los parámetros en dirección opuesta al error.
- Pero, usando el algoritmo de retropropagación, podemos saber cómo se correlaciona cada peso con el error. Esto nos indica el efecto de cada peso en el error de predicción. Es decir, ¿qué parámetros aumentamos y cuáles disminuimos para obtener el menor error de predicción?
- Por ejemplo, el algoritmo de retropropagación podría brindarnos información útil, como que aumentar el valor actual de W1 en 1,0 aumenta el error de red en 0,07. Esto nos indica que un valor menor de W1 es mejor para minimizar el error.

Derivada parcial

- Una operación importante en el paso hacia atrás es el cálculo de derivadas. Antes de analizar el cálculo de derivadas en el paso hacia atrás, podemos empezar con un ejemplo sencillo para simplificar las cosas.
- Para una función multivariante como $Y=X_2Z+H$, ¿cuál es el efecto en la salida Y dado un cambio en la variable X? Podemos responder esto mediante **derivadas parciales** , como se indica a continuación:

$$\frac{\partial Y}{\partial X} = \frac{\partial}{\partial X}(X^2 Z + H) \qquad \frac{\partial Y}{\partial X} = 2XZ + 0 \qquad \frac{\partial Y}{\partial X} = 2XZ$$

- Tenga en cuenta que todo excepto X se considera constante. Por lo tanto, H se reemplaza por 0 después de calcular una derivada parcial. Aquí, ∂X significa un cambio mínimo en la variable X. De igual manera, ∂Y significa un cambio mínimo en Y. El cambio en Y es el resultado de cambiar X. Al hacer un cambio muy pequeño en X, ¿cuál es el efecto en Y?
- El pequeño cambio puede ser un aumento o una disminución de un valor minúsculo. Sustituyendo los diferentes valores de X, podemos determinar cómo cambia Y con respecto a X.
- Se puede seguir el mismo procedimiento para comprender cómo cambia el error de predicción de la NN con respecto a los cambios en los pesos de la red. Por lo tanto, nuestro objetivo es calcular $\partial E/W_1$ y $\partial E/W_2$, ya que solo tenemos dos pesos W_1 y W_2.

UNIVERSIDAD
CATÓLICA
SEDES SAPIENTIAE