

Semana 01

Que es una red Neuronal?



Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano. Esta formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida.

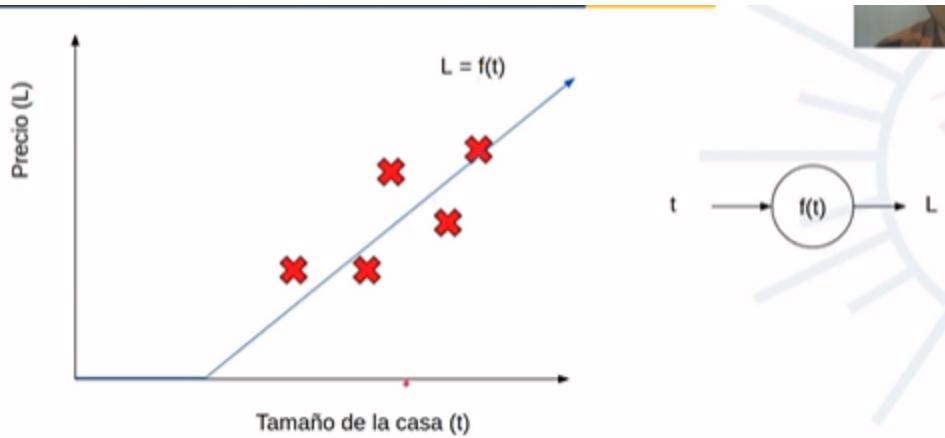
¿Cuál es su objetivo?

aprender modificándose automáticamente a si mismo de forma que puede llegar a realizar tareas complejas.

Data Set: un conjunto de datos tabulados en cualquier sistema de almacenamiento de datos estructurados. **El término hace referencia a una única base de datos de origen,**

la cual se puede relacionar con otras, cada columna del Dataset representa una variable y cada fila corresponde a cualquier dato que estemos tratando

Apartir de mas atributos, puedo lograr una predicción mas acertada.



Reconocimiento de caracteres, de imágenes, de voz,
Generación de texto, predicción de idioma

Se trata de una familia de algoritmos con los que podemos modelar comportamientos inteligentes.

En el caso de una red neuronal a cada una de las partes se le denomina neurona

La neurona tiene valores de entrada luego hace su cálculo interno y da un valor de salida.

Lo que hace la neurona es realizar una suma ponderada de los valores de entrada. La ponderación de cada entrada viene dada por el peso que se le asigna a cada una de las conexiones de entrada

Lo que hace una neurona internamente es una regresión lineal

Son modelos computacionales inspirados en el funcionamiento neuronal y su aplicación principal es el reconocimiento de patrones

Lo que hace una red neuronal es emular el funcionamiento del cerebro con distintos nodos(neuronas) y que se sitúan en diferentes niveles

Es un sistema intelectual que permite tomar decisiones de una manera similar a la que lo hacen los humanos.

Definición y Áreas de la IA

Es la parte de las ciencias de la computación relacionada con el diseño sistemas informáticos inteligentes, es decir sistemas que exhiben características que asociamos con inteligencia en el comportamiento humano:

Comprensión del lenguaje , aprendizaje, razonamiento , resolver problemas y así sucesivamente

Tiene características del ser humano,

Ejemplos: **Asistentes de voz, Smartphones, reconocimiento de rostro en facebook, los contenidos que recibes a través de las redes sociales los selecciona la inteligencia artificial.**

Los vendedores como Amazon utilizan la inteligencia artificial para recopilar información sobre tus hábitos y preferencias de compra, de modo que puedan personalizar tu experiencia online

Los bots utilizan la IA para ayudar a los clientes a buscar datos

Areas de la inteligencia Artificial

- **Computacion evolutiva:** desarrollo de algoritmos inspirados en el evolucion, series de individuos conforme avanzan surgen nuevas generaciones de individos, mutan las caracteristicas, problemas y refinacion de soluciones
- **Vision artificial:** reconocimiento de objetos
- **Procesamiento de lenguaje Natural:** ser capaz de escuchar la voz y convertirla en texto
- **Sistemas Expertos y Representación del conocimiento:** utilizar reglas de logica y despues utilizarlo como un sistema experto. Emular el comportamiento de un experto.
- **Planificación Automática y aprendizaje por reforzamiento:** definir una serie de pasos para resolver un problema
- **Aprendizaje Automatico (Machine Learnig)**
- **Robótica:** Es una area en la que convergen otras areas.

Aprendizaje Automático (Machine Learning)

Es una de las áreas de la inteligencia artificial

Estudio de los algoritmos que aprenden mediante la experiencia.

Un programa de computadora "aprende" de la experiencia E con respecto a alguna clase de tareas T y la medida de desempeño P si su desempeño en tareas en T, medido por P, mejora con la experiencia E. (Mitchell, 1997)

El aprendizaje automático (ML) se divide en:

Aprendizaje Supervisado

- Clasificación (etiquetas discretas)
 - Binaria (de dos etiquetas se selecciona una) → Clasificación de un correo electrónico, Bandeja de Spam o de Correos
 - Multiclasificación (de varias etiquetas se selecciona una) → Colores : varias etiquetas y seleccionamos una etiqueta para un elemento específico
 - Multietiqueta (de varias etiquetas se seleccionan varias) → Ese elemento puede tener varias etiquetas , ejemplo: noticias.
- Regresión (etiquetas continuas)

Aprendizaje No Supervisado (sin etiquetas)

- Agrupamiento (clustering), Ingeniería de Atributos, (crear atributos a partir de otros que tengo) etc.

Aprendizaje Semisupervisado

Se pueden crear redes neuronales para que hagan todo ese tipo de tareas.

Aprendizaje, multiclasificación, multietiqueta.

A veces no es fácil conseguir etiquetas, muestra positiva o negativa.

como asociar los parámetros que sí tienen etiqueta y los que no

Aprendizaje supervisado

Tiene dos tareas principales, regresión y clasificación cuya entrada es la misma y salida es etiqueta

Atributos (Entrada)	Etiqueta (Salida)	Aplicación
Características de una casa	Precio	Bienes raíces
Info. del anuncio y del usuario	Probabilidad de click	Mercadeo en línea
Audio	Texto	Reconocimiento de voz
Rostro en fotografía	Id de la persona	Etiquetado de foto
Imagen tomográfica	Si hay tumor o no	Medicina

Tipos de Datos

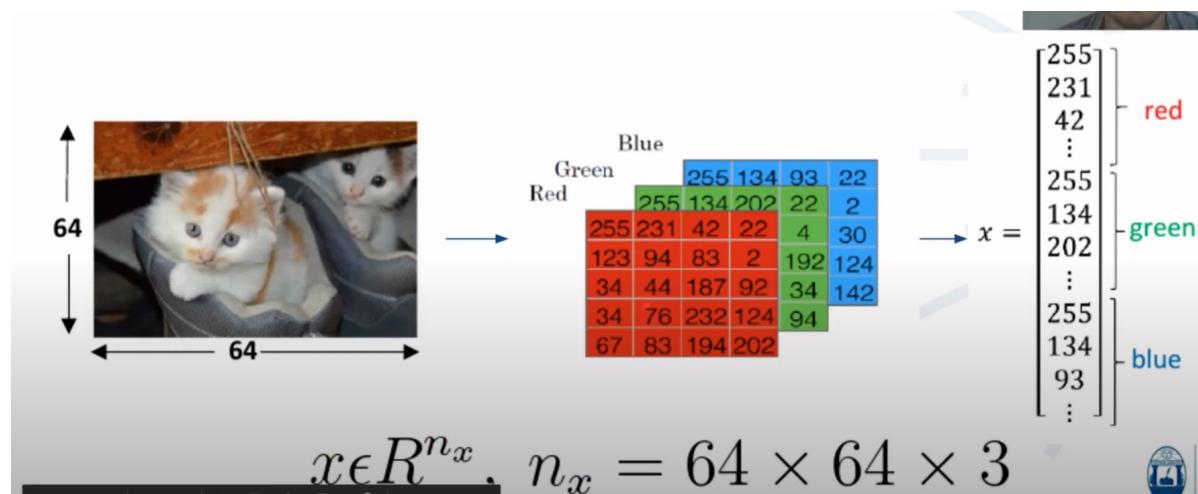
- **Datos Estructurados.**
 - Se pueden colocar como columnas de una base de datos.
- **Datos No Estructurados.**
 - Su formato es el que usamos los seres humanos: imágenes, audio, conversación escrita.

Datos Estructurados

separar en String, fecha etc

No estructurados

▼ Clasificación



Para realizar el entrenamiento necesitamos multiples instancias

$$X = \begin{bmatrix} | & | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & \dots & | \end{bmatrix} \quad Y = [y^{(1)} \ y^{(2)} \ \dots \ y^{(m)}]$$

$X \in R^{n_x \times m}$ $Y \in R^{1 \times m}$

X: Atributos

Y: Etiquetas

Regresion logistica

Nos permite modelar una variable binaria, dos valores

La modela, modelando una probabilidad

Es un modelo matemático que es su forma básica usa una función logística para modelar una variable binaria dependiente.

$$\hat{y} = P(y = 1|x)$$

$$x \in \mathbb{R}^{n_x}, 0 \leq \hat{y} \leq 1$$

La regresión nos permite modelar una probabilidad

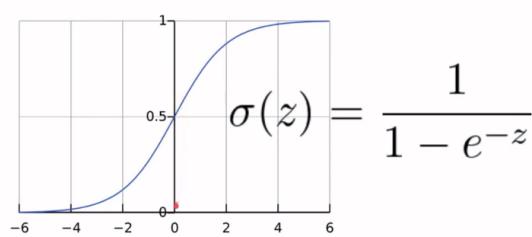
Para realizar mi clasificación

P= probabilidad

y= la probabilidad que la etiqueta sea 1 dado los atributos

Nx= numero de atributos

Función logística:



$$\hat{y} = \sigma(z), z = w^T x + b$$

Parámetros:

$$w \in \mathbb{R}^{n_x}$$

$$b \in \mathbb{R}$$



W^T = W traspuesta de X

W va a ser igual al mismo tamaño de X o de atributos

Producto Punto

$$w = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad w^T = [4 \quad 5 \quad 6] \quad x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$w^T x = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 4 + 10 + 18 = 32$$

Se establece un umbral, podria ser 0.5

Semana 02

Función de Costo

Esta FUNCION nos permite medir o desarrollar el desempeño

El **Aprendizaje automático** tiene que tener una medida de desempeño.

Función de Perdida: que es parte esencial de la función de costo. Nos sirve para medir el error de nuestro modelo,

cuando estamos realizando el aprendizaje(A este proceso se le suele llamar el entrenamiento) utilizando los datos para ajustar o **para entrenar el modelo**.

Tenemos acceso a las etiquetas reales de las instancias que estamos leyendo, que estamos utilizando , entonces tenemos la posibilidad de medir la predicción que da nuestro modelo y compararlo con la etiqueta real que tenemos.

La diferencia que haya entre nuestra predicción y la etiqueta real va a ser un error si nuestra etiqueta es igual a la etiqueta real (la etiqueta pre decida que predice nuestro modelo igual la etiqueta)entonces el error va a ser tiene que ser mínimo o debería de ser cero.

Entonces la **función de pérdida** nos sirve para medir esa diferencia ese error y este error pues se le llama pérdida.

Existen múltiples funciones de pérdida que pueden utilizarse para para hacer esta comparación, sin embargo una de las más utilizadas es esta que tenemos acá

$$L = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

que está basada en el cálculo de logaritmos.

Esta función de pérdida pareciera un poquito larga pero es muy sencilla al considerar que las etiquetas reales en clasificación binaria sólo tienen dos valores que serían 0 y 1 entonces veamos que tenemos en esta ecuación tenemos **y** tenemos **y** gorrito.

y en esta ecuación significa: la etiqueta real , en esta ecuación y el resto que hemos hablado en la anotación de la etiqueta real

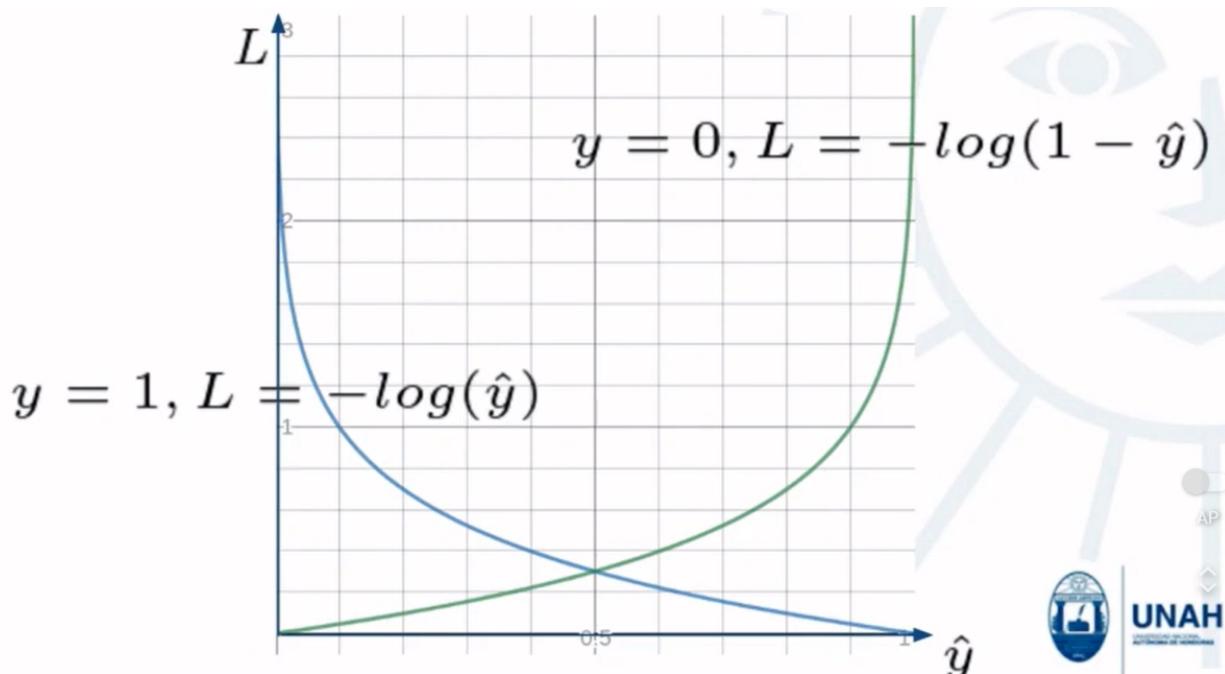
y gorrito es la predicción entonces la etiqueta real puede ser cero o puede ser 1. Si la etiqueta real es cero vemos que pasa ese término de aquí se

$$-(y \log(\hat{y}))$$

vuelve 0 acá tengo un número cero.

cuando $y=0$ la pérdida es igual a - logaritmo de 1- y gorrito ahora sí

$y=1$ entonces qué



Grafica azul m es de la perdida con respecto a y gorrito.

y gorrito es nuestra predicción y vean qué es lo que ocurre si la etiqueta real era uno entonces si yo predigo un valor de 1 significa que no hay error no hay pérdida ósea lo hicimos se hizo bien la predicción por tanto la pérdida en este punto de cero ahora si yo me empiezo a alejar de uno y empiezo a hacer una predicción distinta de uno por ejemplo siempre digo 0.5 recordando que que se puede predecir 0.5 porque los valores que tenemos son valores de probabilidad el sí predijo punto 0.5 el error es más alto ahora si yo predigo 0

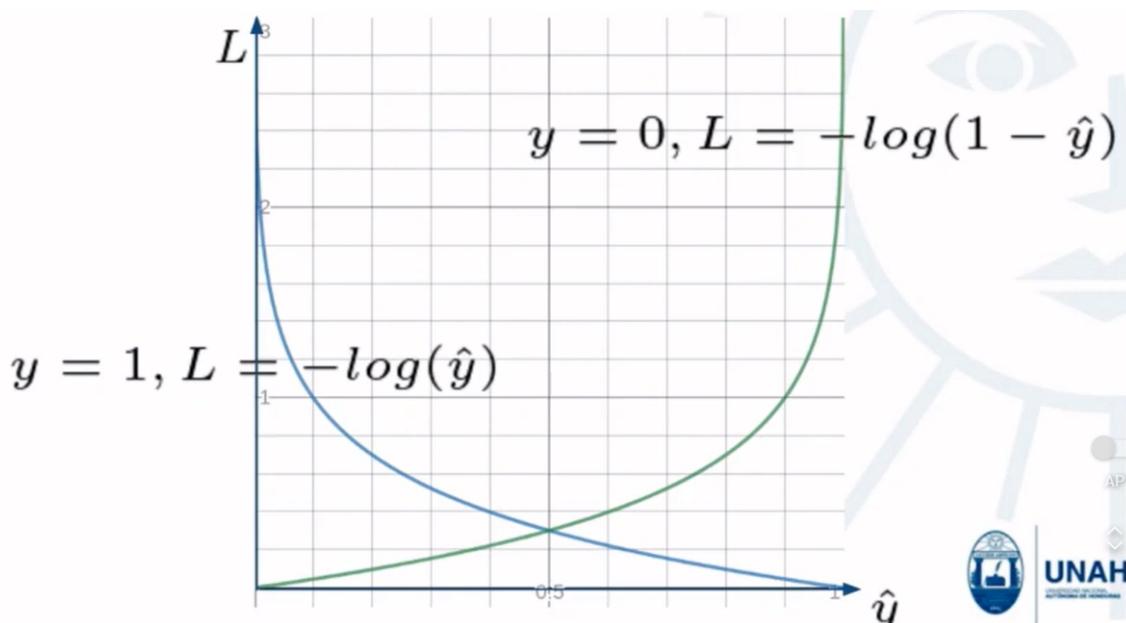
o sea que tengo un error estoy **completamente separado de lo que** realmente debió haber sido yo predijo 0 **pero la etiqueta real es 1.**

Entonces al predecir cero **la pérdida va atendiendo a infinito** lo mismo con el otro lado si la etiqueta **real es 0 y la etiqueta predecida** yo predigo un 0 entonces el error de **la partida a 0.**

pero mientras yo más **me vaya separando de esa predicción o sea yo voy a acercándome a 1** es decir alejándome de la etiqueta real mi pérdida va a ir aumentando **hasta infinito**

vemos entonces que la función de pérdidas nos sirve para simplemente **medir la diferencia o el error que yo tengo entre entre la etiqueta real y la predicción hecha por el modelo.**

El hecho de que tenga esta forma también es muy útil a la hora de optimizar esta función vamos a ver qué va a ser necesario optimizarla.



función de costo

Permite tener una vision global del rendimiento del modelo en todo el conjunto de datos.

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), 1 \leq i \leq m$$

$$J(w, b) = \frac{1}{m} \cdot \sum_{i=1}^m L(y^{(i)}, \hat{y}^{(i)})$$

el entrenamiento no se hace como una **sola instancia la función de pérdida** mide el error entre una etiqueta real y una etiqueta predecida , nuestro conjunto de entrenamiento va a tener más de una sola instancia o sea va a haber múltiples etiquetas y múltiples etiquetas predecidas y múltiples etiquetas reales por tanto entonces necesito una función que me dé una visión global del rendimiento de todo el modelo en todos los datos

entonces esto es la función de costo

la función de costos se representa por la letra J y es un promedio , la sumatoria de las pérdidas en todas las intancias de entrenamiento

y tenemos a esta y que representa y va desde 1 hasta **m que recuerdo recordando m indica todas las instancias de entrenamiento**

o sea que yo sumo todas las las pérdidas en todas las instancias de entrenamiento la dividido entre m y obtengo un promedio el promedio de las pérdidas es igual al costo j

si venga aquí tenemos la misma ecuación que teníamos antes acerca de cómo se calcula él el valor de la etiqueta pre decida utilizando la regresión logística este

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), 1 \leq i \leq m$$

valor de la etiqueta predecida es igual a la función sin molde de w transpuesta de x el producto punto de v w x x + b

sin embargo acá tenemos este super índice con él con él entre paréntesis con él y que ya lo habíamos mencionado anteriormente este súper índice indica la instancia va a ser

va a haber una etiqueta predecida para la instancia del 1, para la instancia otra para la instancia 2 y así hasta la instancia m vamos a tener m etiquetas predecidas y esas etiquetas predecidas que son estas que son estas que se ven acá

$$\sum_{i=1}^m L(\hat{y}^{(i)}, \hat{y}^{(i)})$$

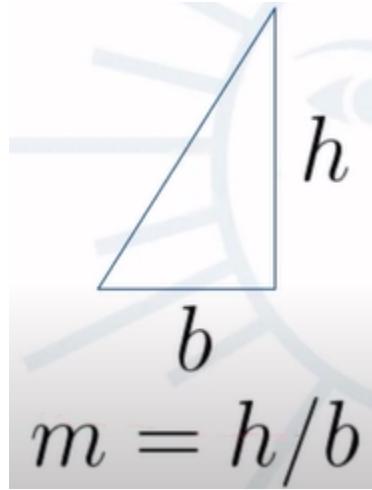
van a compararse con m etiquetas reales que yo voy a tener en el vector y mayúscula que es lo que tengo aquí, entonces esta pérdida va a haber m perdidas distintas L y de gorrito y \hat{y} representa la pérdida en cada una de las instancias sumó estas pérdidas vean que estoy explicando esto solamente sumó estas pérdidas todas las dividida entre m y obtengo el costo

El costo y aparentemente o se muestra se suele mostrar como que depende del w y b porque depende de w en realidad porque las etiquetas reales ya están definidas o sea no es algo que va a cambiar lo que va a cambiar son las etiquetas predecidas y las etiquetas predecidas no dependen de x porque aquí ya está definida los x son los atributos que están en el conjunto de entrenamiento sino que dependen de w y de b que son los verdaderos parámetros del modelo por eso decimos entonces que el costo depende de w y de b.

Derivadas: un repaso

La derivada de una función es la pendiente de la **recta tangente a la curva de dicha función** en un punto determinado.

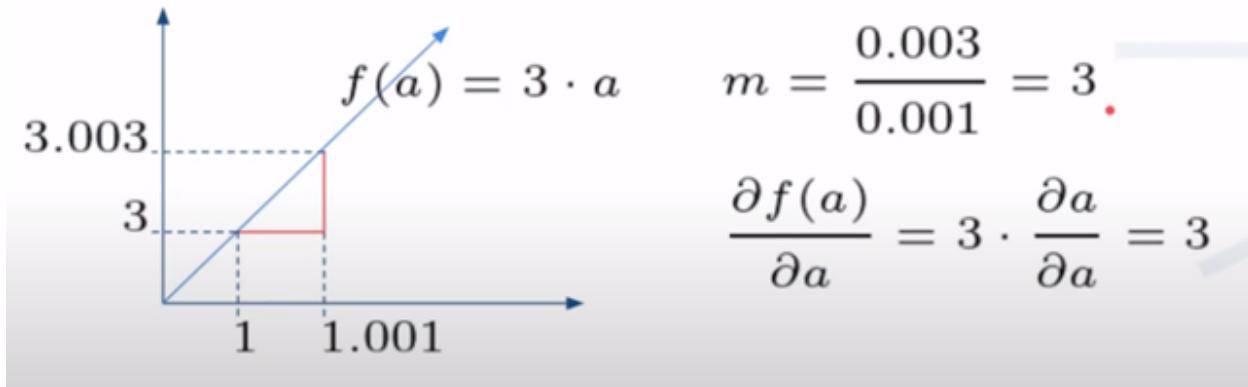
De acuerdo con esta definición entonces la derivada es un valor numérico que me da la inclinación de una recta tangente a una función en un punto específico. Por ejemplo si la hipotenusa este triángulo rectángulo fuese la recta tangente entonces la derivada sería la pendiente de esta inclinación de esta recta, la inclinación lo tendríamos dividiendo la altura entre la base en la derivada. entonces será igual h entre b $m=h/b$



está recto este sería tangente a una curva de una función que tiene un punto preciso y esa sería la derivada h entre b

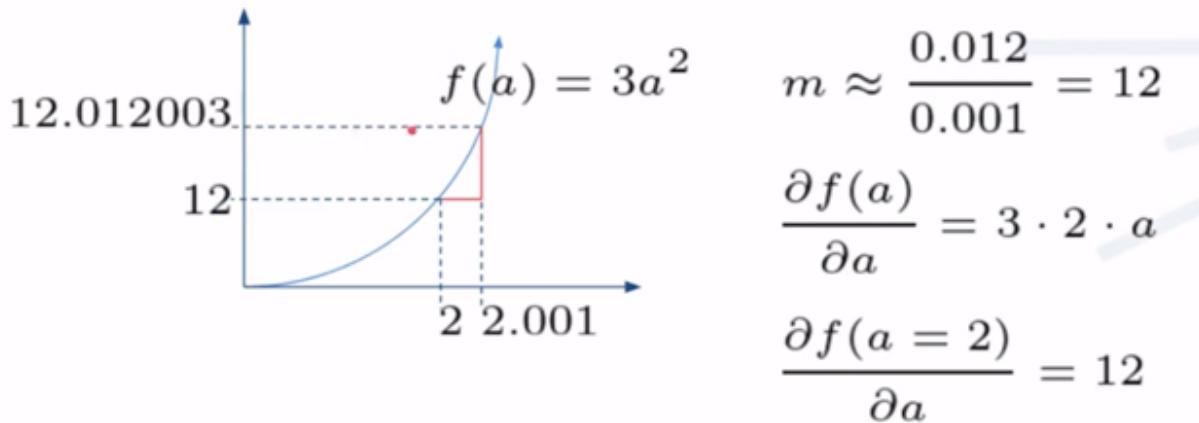
extendiendo este ejemplo a uno un poquito más aplicado, podemos ver los siguientes digamos que podemos encontrar la derivada de la función $f(a) = 3a$ para hacerlo utilizando este concepto verdad el triángulo podríamos evaluar la derivada en un punto ahora cuando es igual a 1 observamos que esta gráfica no está hecha digamos a escala correcta pero el templo parece un poco grande para que se vea claro entonces empezamos en un punto uno quiere evaluar la derivada aquí para evaluar la derivada en ese punto yo podría entonces crear un pequeño triangulito para hacerlo e incremento un poquito verdad uno digamos el 0.01 y evaluó la altura de ese triángulo de acuerdo a la función en este punto ahora en 0 a 1.001 la función vale 3.003 entonces tengo este pequeño triángulo y para calcular la inclinación del de su poder no usar que sería la inclinación de la recta tangente a la curva en este punto observamos que en este caso esta

función lineal la la recta tangente a la curva es es la misma que la recta que la función porque una función lineal entonces la inclinación sería igual a la altura entre la base que sería igual a $m = 0.003 / 0.001$ y eso me va a dar igual a 3 si yo calculo la derivada utilizando la fórmula de derivación analítica de manera analítica entonces yo puedo voy a llegar a la misma conclusión para la derivada de $f(a)$ con respecto a ser igual a 3 por la derivada a con respecto a a que es 1, entonces sería igual a 3 obtengo entonces el mismo resultado de modo que otra vez la derivada simplemente en la inclinación de esta recta en este punto en este caso la inclinación vale 3



ahora miremos otro ejemplo que nos termina de ilustrar esta esta situación digamos que la función favor a la escuadra tica la función especial igual a $3x$ al cuadrado en este caso yo pues quiero evaluar nuevamente la derivada la función en este caso cuando es igual a 2 puedo utilizar la misma idea incremento da un poquito de 0.001 y evaluo la función en este punto en este caso la función al evaluar la lo pueden comprobar con su calculadora 3 por 2.001 al cuadrado me da igual a esto a 12.012003 en el caso que fuera de 2 verdad sería igual 3 por 4 12 tengo este pequeño triángulo y calculo la deriva calculó la inclinación ahora esta inclinación sería la inclinación de la hipotenusa de este triángulo pero ojo aquí vemos una cuestión en aproximadamente en la diapositiva

anterior era igual ahora yo tengo un aproximado porque porque no estoy **incluyendo este de 0.03** estoy quedándome con la cifra más importante 0.002 entre 0.001 me da igual a 12



Debemos hacer un salto infinitesimal, infinitamente pequeño para obtener el valor correcto.

Algoritmo de descenso de gradiente

un algoritmo de optimización que **utilizamos para minimizar una función** específica.

en nuestro caso **la función que queremos minimizar es la** función de costos

la función de costos $j(w,b)$ nos servía para determinar o evaluar el desempeño de nuestro algoritmo con respecto a las etiquetas que contamos en nuestro conjunto de datos

sin embargo decíamos también que la función de costo depende de **w y b donde son los parámetros del modelo.** y gorrito en realidad sigma de w traspuesta de $x(i) + b$ entonces estos parámetros necesitamos encontrarlos pero necesitamos encontrarlos de forma tal de que el costo sea mínimo, o sea yo quiero **parámetros que minimicen el costo**

$$J(w, b) = \frac{1}{m} \cdot \sum_{i=1}^m L(y^{(i)}, \sigma(w^T x^{(i)} + b))$$

$$J(w, b) = \frac{1}{m} \cdot \sum_{i=1}^m L(y^{(i)}, \hat{y}^{(i)})$$

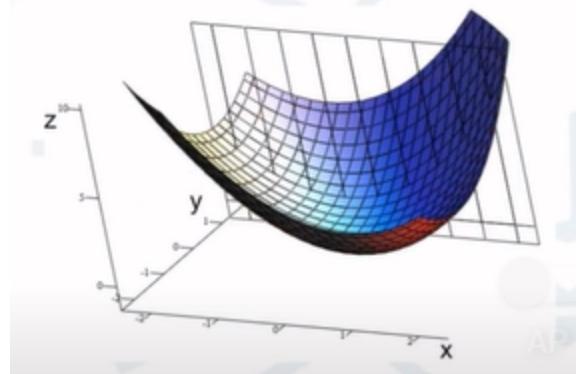
como encontrarlos **parámetros que minimicen el costo** ? la respuesta es mediante el algoritmo de **descenso de gradiente**.

en la gráfica que aparece abajo podemos ver la imagen de una función de costo así como la **tenemos definida** con la pérdida igual a aquella función que utilizan

logaritmos vista anteriormente tiene una de las ventajas y es que a la hora de considerar múltiples variables en este caso dos variables w y b , su forma es convexa.

El algoritmo de descenso de gradiente es un algoritmo iterativo que funciona paso a paso empieza en un punto cualquiera de la gráfica, intenta paso a paso ir avanzando moviendo en este caso si yo fuera w y x fuera b moviendo cambiando los valores de w y b para que poco a poco vayamos buscando el valor donde el costo es mínimo

el Algoritmo de descenso de gradiente no nos garantiza a encontrar el valor exacto o el punto exacto sino un punto aproximado **pero generalmente ese punto aproximado** es lo que es suficiente para nosotros



w como dijimos antes va a tener **el mismo tamaño que x y es donde x es el número y el tamaño de atributos**

por ejemplo en el caso de las imágenes de datos de $64 \times 64 \times 3$ píxeles vamos a tener más de 12.000 dimensiones **esas 12 mil dimensiones son intratables** de forma analítica a resolver el problema así por tanto en el algoritmo de descenso de gradiente nos proporciona una solución computacionalmente a tratar tratable

también el Algoritmo de descenso de gradiente es útil cuando ya tenemos funciones que no sean **convexas sino que de funciones que** tengan múltiples puntos mínimos podríamos pensar como un valle con **montañas o mejor dicho un terreno que** tenga multiples partes que se undan verdad entonces ahí el algoritmo de descenso es mas útil.

que buscar la derivada de igual 0 **porque la derivada de igual a 0 puede ser igual a 0** en muchos puntos dependiendo de la complejidad de la función

EJEMPLO DE FUNCIONAMIENTO

repita {

$$w_1 := w_1 - \alpha * \delta J / \delta w_1$$

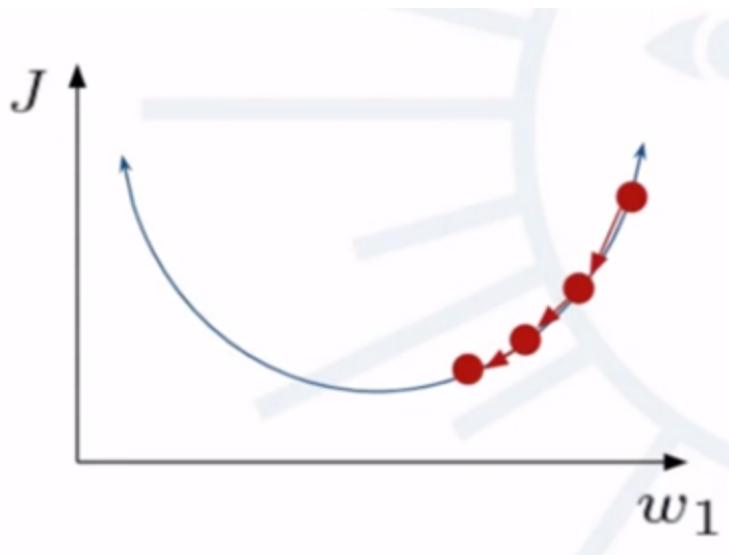
$$w_2 := w_2 - \alpha * \delta J / \delta w_2$$

...

$$w_{nx} := w_{nx} - \alpha * \delta J / \delta w_{nx}$$

$$b := b - \alpha * \delta J / \delta b$$

}



el funcionamiento del algoritmo de descenso de gradiente se puede resumir en una sola ecuación muy **sencilla que funciona optimizando las** variables de forma independiente

w tiene nx elementos en nx componente w 1 w 2 hasta w en nx y tenemos **b que es un valor real**, sin embargo en todos la ecuación es la misma. El algoritmo de descenso de gradiente itera, repita una cantidad de veces buscando el punto mínimo.

entonces qué es lo que hace? el valor de un parámetro por ejemplo w_1 , lo actualiza de la le resta una constante alfa multiplicada por la derivada de la función en este caso la función del costo con respecto a ese parámetro.

veamos el valor de la derivada, el valor de la derivada de una función con respecto a ese parámetro me da la pendiente de la recta tangente **a la función en un punto específico**

yo parto de un punto por ejemplo **el primero y yo encuentro la pendiente** de la recta tangente a la función en ese punto entonces , considerando solamente una una variable, entonces la pendiente de la recta tangente ese punto va a tener un valor positivo. este valor positivo se va a multiplicar por alfa que es un valor positivo y por un menos y lo que va a ocasionar es que w_1 se le reste un valor entonces a ese valor que teníamos antes se le va a restar un valor que nos va a llevar al siguiente punto .

yo solo puedo hacer dos cosas con w_1 sumarle valores o restarle aquí en el ejemplo se le restó porque la derivada me está saliendo positiva una inclinación positiva

sin embargo si hubiesemos partido desde este punto la derivada en este punto la recta sería negativa y por tanto esto sería menos por menos sería un más entonces si yo parto desde acá le estaría sumando a la el valor de w_1

la derivada la inclinación de la derivada me sirve para saber en qué dirección moverme si le tengo que sumar o si le tengo que restar a la a la variable al parámetro para acercarme aquí donde quiero llegar el punto mínimo.

En la siguiente iteración w_1 ya valdría (el siguiente punto) sin embargo aquí la derivada sigue siendo positiva entonces le voy a restar otro valor, esta derivada positiva por este menos me va a ocasionar una resta y poco a poco me voy acercando.

otra cosa que observamos es que aquí el salto fue más grande sin embargo aquí el salto fue más pequeño y aquí el salto es más pequeño

por qué porque aquí la derivada de la más grande era más más inclinada aquí la derivada se va haciendo menos inclinada aquí se va haciendo menos inclinada y poco a poco se va acercando a cero

cuando más pequeña derivada más pequeño es el salto y esto es una ventaja porque cuando yo estoy cerca del punto mínimo quiero hacer saltos pequeñitos porque si hago un salto muy grande me puedo pasar al otro lado y entonces perdería el punto mínimo que desea buscar aquí viene entonces el juego entra en juego también el parámetro alfa,

el parámetro alfa que es constante me sirve para aumentar o reducir el tamaño del salto que yo estoy haciendo a este parámetro alfa se le llama el ritmo de aprendizaje o learning rate en inglés el learning rate o ritmo de aprendizaje y entre más alto es lo que va a ocasionar es que se hagan saltos más largos y entre más pequeños se hagan saltos más pequeños es bueno tener un learning rate pequeño porque así yo me aseguro de llegar al punto mínimo.

sin embargo si el learning rate es muy pequeño que pasa que si yo parto tengo la suerte de partir de un punto un poco lejano al punto mínimo entonces me va a tomar muchísimas iteraciones puede llegar a tomar muchísimas iteraciones llegar hasta el punto mínimo tantas que se consuman demasiado tiempo de procesamiento

bien entonces que el algoritmo descenso gradiente hacen lo mismo con w_2 doble 3 hasta w_{nx} y hace lo mismo con b .

la idea es esencialmente la misma ir moviendo cambiando el ajustando el parámetro paso a paso acercándose hacia el punto mínimo y este punto mínimo el tamaño del salto está en mitad determinado por el learning rate y **está determinado por la magnitud de la derivada y la dirección del salto está determinada por la x en la dirección de la derivada si es positiva o si es negativa**

cuando termine el algoritmo de descenso de gradiente?

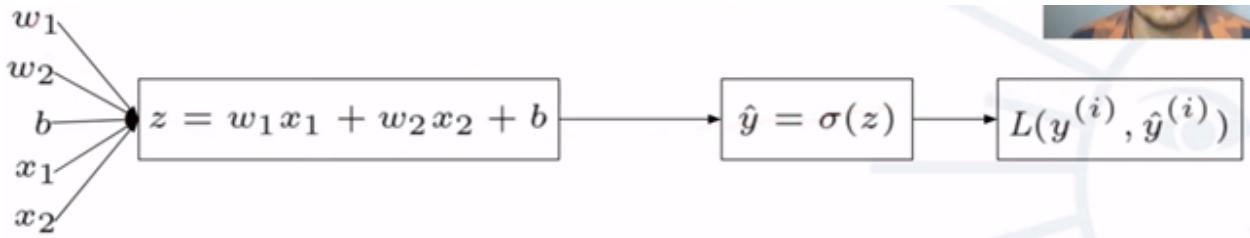
pues va a terminar cuando nosotros estemos satisfechos y ahí empecemos por ejemplo a considerar que ya estamos llegando a un punto de costo muy pequeño también va a terminar se suele definir un número de iteraciones fijos por ejemplo 1000 o 2000 y observar cuál es el comportamiento algoritmo en esas 2 mil iteraciones

Grafo de Calculo de la Regresión Logistica: Propagación hacia delante y hacia atrás., retropropagacion

Grafo de cálculo

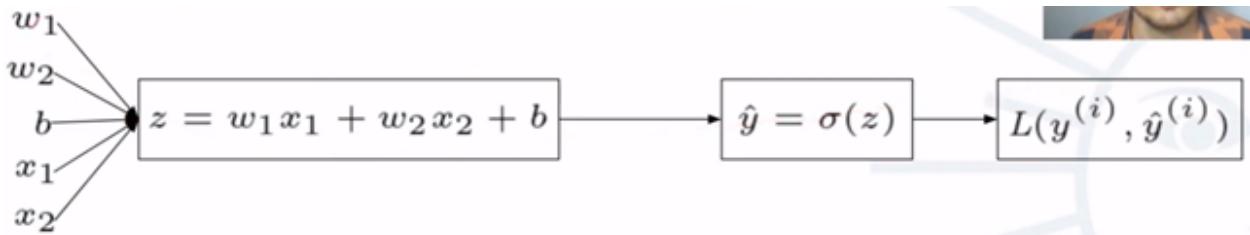
Herramienta gráfica que muestra los pasos necesarios para obtener un cálculo o un valor final buscado, las dependencias de dichos pasos. En el caso de la regresión logística sería

Ejemplo $n_x = 2$ (sólo 2 atributos)



con n que igual 2 w va a tener dos dimensiones y x **va a tener 2 dimensión que son las que** están presentadas aca partimos entonces de w 1 y 2 de b , x_1 y x_2 sabemos que estos son parámetros del modelo todo esto sirve como entrada para el siguiente paso el cálculo de z el cálculo de z :

como ya sabemos la fórmula del $z = w$ traspuesta por $x + b$. w traspuesta por x ya multiplicado ha hecho la expansión será igual a esto



después de haber calculado z pues lo que me queda es aplicarle la función sigmoide a z y obtener un valor de probabilidad y gorrito,

luego obteniendo este valor de probabilidad esta etiqueta presida yo puedo calcular la perdida

vemos este cálculo es para una sola instancia del conjunto entrenamiento sin embargo para obtener el valor final definitivo que sería el costo necesitaría hacer todo este proceso para las m instancias de modo que habiendo calculado la pérdida de las instancias podría yo calcular el costo.

Ahora la propagación hacia delante simplemente consiste en en el cálculo de estos valores hasta llegar al valor final.

sin embargo existe lo que se conoce como la **propagación hacia atrás** como vimos recientemente en el algoritmo de descenso de gradiente para poder actualizar los valores de los parámetros **dicho algoritmo necesita obtener los** valores de las

derivadas de la del costo con respecto a esos parámetros. para poder obtener esos valores de las derivadas nos sirve la propagación hacia atrás partiendo de la pérdida en el primer paso en la preparación hacia atrás sería :

1. calcular la derivada de la pérdida con respecto a y gorrito .(no con respecto a y porque no es un valor que nos interesa y es un valor que ya está el conjunto entrenamiento sin embargo y gorrito sí porque el que depende de nuestros parámetros. Por tanto la derivada de L con respecto al y gorrito conociendo la fórmula con logaritmos dada anteriormente la derivada sería ésta

$$\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

ahora lo que me interesa a mí es la derivada de L con respecto a a estos parámetros a los parámetros w y b para hacer eso entonces tengo que dar un paso más atrás lo primero que podría hacer en este siguiente paso sería

2. obtener la derivada de y gorrito con respecto a z que es la función que tengo aquí esta derivada y gorrito con respecto a z = a sigma de zeta por uno menos sigma de z.

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

y como sigma de z= y gorrito la puedo también escribir si y gorrito por 1 - y gorrito

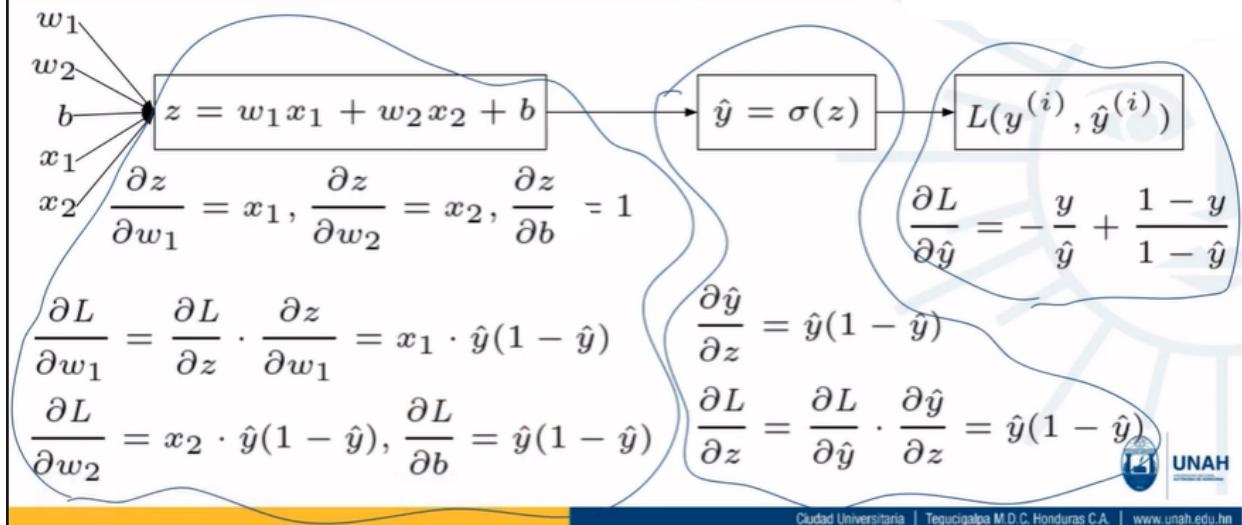
ahora siempre estoy interesada en obtener la derivada de la pérdida de este la que yo puedo calcular en este momento pero me interesa la derivada de la pérdida en este caso ahora con una variable más atrás con respecto a z , entonces la derivada de la pérdida con respecto acepta para obtenerlo pues puedo aplicar la famosa regla de la cadena la derivada de la pérdida con respecto a z va a ser igual la derivada de la pérdida con respecto a la gorrito por la derivada de diego rito con respecto a Z

la derivada de la pérdida con respecto
al ego rito la calcula en el paso
anterior es ésta y de llegó recto con
respecto acepta a la acabo de calcular
es este entonces al multiplicar llegó
ritos por 1 - llegó rito por todo esto
que tenemos acá y simplificar nos queda
este valor llegó ritos por un número
sin embargo todavía no he llegado hasta
acá tengo que dar un paso más en el

3. siguiente paso entonces lo primero que voy a poder calcular es la derivada de z con respecto a estas variables x_1 y x_2 no me interesan porque están dadas en conjunto entrenamiento no interesan w_1 w_2 y b calculó estas derivadas de z con respecto a w_1 que con una función lineal bastante sencilla simplemente x_1 receta con respecto a w_2 es x_2 y la derivada de zeta con respecto a b es 1 con estas ya puedo aplicar la regla de la cadena nuevamente la derivada de I con respecto a w_1 va a ser simplemente la derivada de I con respecto a z que la calcula en el paso anterior por la derivada de z con respecto a w_1 que es simplemente x_1 por tanto el

resultado es simplemente x_1 por diego
ritó menos 1 llevó ritmo de forma
similar va a ocurrir con la derivada de
 I con respecto a w_2 va a ser igual a x_2
 x_1 menos llegó por x llegó ritó por 1 -
diego ruiz y finalmente la deriva
con respecto a ello va a ser igual 1 por
llegó rito por 1 - hierba
con esto entonces este valor
este valor y este valor son los valores
de las derivadas parciales que yo
necesito casi casi porque porque son
derivadas de I con respecto a los
valores a los parámetros sin embargo lo
que nos interesaba era la derivada lo
que nos interesa al final es la derivada
de a_j con respecto a estos parámetros
sin embargo para eso necesitamos
considerar las m instancias esas m
instancias las vamos a considerar despues.

Ejemplo con $n_x = 2$



Regresión Logística con m instancias

$$\frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial w_1}, \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial w_2}, \dots, \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial w_{n_x}}, \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial b}$$

$$J(w, b) = \frac{1}{m} \cdot \sum_{i=1}^m L(y^{(i)}, \hat{y}^{(i)})$$

$$\frac{\partial J(w, b)}{\partial w_1} = \frac{\partial}{\partial w_1} \left[\frac{1}{m} \cdot \sum_{i=1}^m L(y^{(i)}, \hat{y}^{(i)}) \right]$$

$$\frac{\partial J(w, b)}{\partial w_1} = \frac{1}{m} \cdot \sum_{i=1}^m \frac{\partial L(y^{(i)}, \hat{y}^{(i)})}{\partial w_1}$$

Lo que necesitamos saber es el costo, por esa razon vamos a obtener la derivada del costo

Para conocer el costo simplemente debemos calcular el promedio de las perdidas.

Veamos que pasa con las derivadas

Derivada de la perdida con respecto a w1

derivada del costo con respecto a los parámetros es igual a simplemente la la sumatoria o el promedio de la **derivada de la pérdida con respecto a ese parametro, en los diferentes instancias.**

Para calcular el costo calculo el promedio de las perdidas

Para calcular la derivada del costo con respecto a un parametro calculo el promedio de las derivadas de las perdidas con respecto a este parametro

Pseudocódigo con m instancias

```
logistic_regression(x1[m], x2[m], y[m]):  
    J, dw1, dw2, db := 0;  
    z, y_hat, dz := array[m];  
    for i = 1 to m  
        z[i] := w1*x1[i] + w2*x2[i] + b;  
        y_hat[i] := sigmoid(z[i]);  
        J += L(y[i], y_hat[i]);  
        dz[i] := y_hat[i] - y[i];  
        dw1 += x1[i]*dz[i]; dw2 += x2[i]*dz[i];  
        db += dz[i];  
    J /= m, dw1 /=m, dw2 /=m, db /=m;
```

Regresion logistica, recibe 3 arreglos empiezan en uno y terminan en m para que coincidan con el esquema matematico que tenemos.

Guarda todos los atributos 1 de las instancias desde la 1 hasta m

En y estan las etiquetas.

2. variables iniciadas en 0

3. areglos para guardar las instancias, inicializados como arreglos de tamaño m, se aume que empiezan en 0

4. ciclo para considerar las m intancias, tenemos dos pasos la propagacion hacia delante (Se resumia en las primeras tres lineas)

El calculo de z para la i esima instancia, w1, w2, y b seran aleatorios al principio y despues se ajustaran

Luego se calcula y gorrito, funcion sigmoide de z.

luego J ira acumulando la suma de las perdidas

1. W1, w2, b . Inicialmente vana a ser aleatorios y se van a ir ajustando

2. j acumula la suma de las perdidas

3. Propiamente hacia atras no necesita calcular derivadas

4. dw1 y db quieren obtener el costo con respecto a los parametros

y hacia atras acumulo los calores de las sumas dentro del ciclo.

Fuera del ciclo:

j/m para obtener el costo, dw1/m para obtener la derivada del costo con respecto a w1, w2 y

Costo y derivadas del costo con respecto a los parametros considerando las m intancias de entrenamiento

```

logistic_regression(x1[m], x2[m], y[m]):  

    J, dw1, dw2, db := 0;  

    z, y_hat, dz := array[m];  

    for i = 1 to m  

        z[i] := w1*x1[i] + w2*x2[i] + b;  

        y_hat[i] := sigmoid(z[i]);  

        J += L(y[i], y_hat[i]);  

        dz[i] := y_hat[i] - y[i];  

        dw1 += x1[i]*dz[i]; dw2 += x1[i]*dz[i];  

        db += dz[i];  

    J /= m, dw1 /=m, dw2 /=m, db /=m;

```

Prop. hacia adelante

Prop. hacia atrás

Semana 03

Instalación de Python en Ubuntu

```
Fine, Esc, View, Print, Terminal Help  
raul@nuc-unah:~$ python --version  
Python 2.7.17  
raul@nuc-unah:~$ sudo apt install python3  
[sudo] password for raul:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python3 is already the newest version (3.6.7-1~18.04).  
The following packages were automatically installed and are no longer required:  
  libhawtjni-runtime-java libjansi-java libjansi-native-java libjline2-java libllvm7 libllvm8  
  libpango1.0-0 linux-headers-4.15.0-108 linux-headers-4.15.0-108-generic  
  linux-image-4.15.0-108-generic linux-modules-4.15.0-108-generic  
  linux-modules-extra-4.15.0-108-generic scala-library scala-parser-combinators scala-xml  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 34 not upgraded.  
raul@nuc-unah:~$ █
```

Ejemplo:

Nuestro primer programa en python lo que hace es que lee una serie de notas que un estudiante obtuvo y imprime el **promedio**.

Entonces, para realizar este proceso primero se crea una variable puntos. Esta variable puntos o este objeto puntos mejor dicho es un objeto de tipo diccionario en python. Los objetos diccionarios son conjuntos de pares clave valor.

En este caso a b y c son las claves y 10, 8.0 y 5.0 son los valores. Los diccionarios lo que permiten es utilizar una clave para obtener el valor. De la clave a me devolvería 10, la clave b devuelve 8 y la clave c 5.

Luego tenemos dos objetos tipo enteros inicializados en cero y uno de tipo Booleano que tiene un valor de falso. Vemos que los tipos no se declaran en python simplemente el tipo se infiere o se obtiene a partir de el objeto que está a la derecha del igual.

Vemos que en python no es necesario crear una clase , crear un método, una función main para poder empezar a programar simplemente empezamos a escribir el código.

Ahora miramos acá que un ciclo while **la negación en python se hace utilizando la palabra reservada a not**. Not fin osea mientras no sea fin continúa haciendo lo siguiente, empezamos el ciclo con dos puntos

Si no se identa el código entonces no se considera que es forma parte del ciclo while.

input():

función que me sirve para leer texto desde la consola, leo una nota de la consola en este caso la nota se espera que sea a b o c.

Si la nota es algo vacío o sea no escribió nada solamente se escribió un enter entonces asignamos como fin = true. O sea ya no vamos a continuar el ciclo después. Sin embargo, si no es vacío entonces se busca si la nota está en puntos . **Not in** es un operador del diccionario que me permite verificar si está este texto que escribió el usuario pertenece o no pertenece mejor dicho del diccionario si no pertenece al diccionario entonces imprimo en la consola nota desconocida. Sin embargo si pertenece al diccionario entonces iría el else, al número de cursos le aumentaría en 1 y a total puntos y le aumentaría la cantidad de puntos que corresponde la nota que el usuario ingreso y que eso sí lean si ingresó a entonces se le sumaría 10 si ingreso b se le sumaría a 8 y se ingresó 5 se se le sumaría.

Este ciclo se va a repetir indefinidamente hasta que el usuario no ingrese nada, hasta que nota sea igual igual a vacío.

Al final entonces si el número de curso del mayor que 0 entonces se va a imprimir el promedio.

El promedio se imprime utilizando la función print y aquí pues interesante es la forma que sea imprime. Tenemos un string es indicado por comillas simples tiene un método que es el método format. El método format recibe un argumento en este caso lo que está imprimiendo en el promedio total puntos entre el número de cursos me imprimiría y el promediome calcularia un número real y este valor que se produce dentro del format va a ser mostrado en vez de la llave cero, es decir el string colocó este esta simbología llave cero para indicar que el primer argumento del format se va a poner dentro del las llaves, yo puedo utilizar llave 1 y mandar un segundo argumento a format para que se muestre otro valor y llave 2 y así sucesivamente para mostrar más valores dentro de un string la ventaja de esto es que de esta forma no tenemos que estar concatenando strings que es algo una práctica bastante incómoda y que también puede ser muy común.

```

puntos = {'A':10.0, 'B':8.0, 'C':5.0}
numeroCursos =0
totalPuntos =0
fin = False

print('Bienvenido, Ingrese sus notas')
while not fin:
    nota = input()
    if nota == '':
        fin = True
    elif nota not in puntos:
        print('Nota desconocida')
    else:
        numeroCursos += 1
        totalPuntos += puntos[nota]

if numeroCursos>0:
    print('Su promedio es {}'.format(totalPuntos/numeroCursos))

```

```

puntos = {'A':10.0, 'B':8.0, 'C':5.0}
numeroCursos =0
totalPuntos =0
fin = False

print('Bienvenido, Ingrese sus notas')
while not fin:
    nota = input()
    if nota == '':
        fin = True
    elif nota not in puntos:
        print('Nota desconocida')
    else:
        numeroCursos += 1
        totalPuntos += puntos[nota]

if numeroCursos>0:
    print('Su promedio es {}'.format(totalPuntos/numeroCursos))

```

Bienvenido, Ingrese sus notas

A
B
C

Su promedio es 7.666666666666667

Objetos e Identificadores en Python

¿ Que son los objetos?

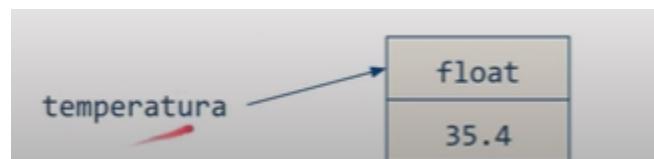
son la **forma en que python representa los** valores.

En python todos los valores son objetos. El comando más importante de python o el enunciado mas importante sería el de asignación. El enunciado de asignación pues tiene tres partes:

1. Un lado izquierdo
2. un igual y
3. El lado derecho

Ejemplo: temperatura = 35.4

El lado izquierdo vamos a tener un nombre o un identificador que se va a asociar mediante el igual a un objeto que produce el lado derecho, sabemos que el lado derecho puede haber una función una expresión que al final se evalúa a un objeto. En este caso tenemos un objeto literal el cual se asocia a al nombre temperatura, esto gráficamente conceptualmente podemos verlo asi:



veremos que tenemos el nombre asociado a un objeto en este caso este es un objeto de tipo float.

Los objetos en python tienen tipo lo que pasa que este tipo no está declarado de forma evidente sino que se conoce a partir en este caso del literal del objeto, en este caso literal porque tiene un punto entonces es un float .

Identificadores

Los identificadores son sensitivos a mayúsculas y minúsculas. Es decir no es lo mismo HOLA con mayúscula a hola con minúscula son identificadores diferentes.

También en python se suele utilizar el guión bajo para unir las palabras distintas digamos si tengo un identificador como producto punto se suele escribir producto que un bajo punto no producto mayúsculas p punto sino que usa para unir las palabras. También se pueden usar números y la restricción es que no se pueden utilizar o no se puede utilizar un número para iniciar el nombre de un identificador sino que tiene que empezar con una letra

Existe un grupo de palabra reservada que no se pueden utilizar son palabras que utiliza el lenguaje para fines especiales. Se puede esperar de que son utilizadas en otros lenguajes de programación también y por tanto estas palabras no se pueden usar como identificadores

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Tipado Dinámico

Python es un lenguaje de tipo dinámico qué significa esto que los identificadores pueden estar asociados a un objeto de un tipo en un momento y pueden estar asociado a un objeto de un tipo diferente en un momento posterior

vemos acá por ejemplo que a este identificador está asociado a un objeto de tipo string, yo imprimo a y pues imprimiría hola, luego yo puedo asociar a un objeto de tipo entero y se imprimiría el 2 luego yo puedo asociar a un tipo a un objeto de tipo booleano y pues podría usarlo dentro de un if para imprimir verdadero

```
a = 'hola'  
print(a)  
a = 2  
print(2)  
a = True  
if (a):  
    print('Verdadero')
```

```
a = 'hola'  
print(a)  
a = 2  
print(2)  
a = True  
if (a):  
    print('Verdadero')
```

```
hola  
2  
Verdadero
```

A pesar de que python es un lenguaje de tipo dinámico y que los objetos identificadores pueden asociarse a objetos de diferentes tipos esta no es una práctica conveniente cuando se está desarrollando un programa.

El único fin de esto es que podemos utilizar esto debido a que python es un lenguaje también interactivo, es posible abrir la consola de python y utilizar y escribir instrucciones de manera interactiva y obtener respuesta inmediata.

Sin embargo cuando crea un programa en python desde el editor de texto lo conveniente es que cada identificador esté asociado a un objeto del mismo tipo siempre, sin embargo es el programador el que tiene la responsabilidad de hacer esto

¿Qué son los Aliases?

Los alias es una forma de tener varios nombres o varios identificadores para un mismo objeto

Es posible crear un alias asignando un identificador a un objeto existente por ejemplo:

yo tengo el objeto temperatura

temperatura = a 3.54 sin embargo yo puedo hacer original temperatura que es lo que hace esto? crear un alias, otro identificador para el mismo objeto.

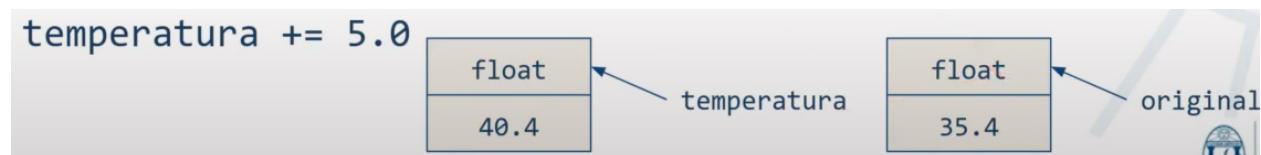
Entonces esto conceptualmente gráficamente lo podemos ver así :



dos identificadores para el mismo objeto.

Los alias también se pueden romper. si yo a uno de estos objetos en este caso temperatura le sumo cinco por ejemplo entonces estoy rompiendo el alias temperatura vale ahora 40.4 cree un nuevo objeto y original vale 35.4

vemos acá que este comportamiento parece un poco extraño no pareciera que fuera lo común en lenguajes de programación sin embargo este una práctica común en otros lenguaje de programación vamos a volver o vamos a aclarar este concepto un poquito más cuando hablemos acerca de la inmutabilidad de los objetos en este caso los objetos de tipo flow son inmutables por esta razón este flow 35.4 no se puede cambiar sino que al modificarlo se crea un objeto nuevo y se rompe la ley.



Clases Predefinidas Numéricas

Clases Predefinidas

Predefinidas: Que se pueden utilizar sin importar nada adicional

Algunas de las clases predefinidas mas usadas en Python son:

Class	Description	Immutable?
<code>bool</code>	Boolean value	✓
<code>int</code>	integer (arbitrary magnitude)	✓
<code>float</code>	floating-point number	✓
<code>list</code>	mutable sequence of objects	
<code>tuple</code>	immutable sequence of objects	✓
<code>str</code>	character string	✓
<code>set</code>	unordered set of distinct objects	
<code>frozenset</code>	immutable form of set class	✓
<code>dict</code>	associative mapping (aka dictionary)	

Clases numéricas : Bool, int, float

Secuencias de elementos : list, tuple, str

Representar conjuntos: set, Frozenset

Representar diccionarios: dict.

Las clases que están marcadas con un check **son clases inmutables y las clases que** no tienen el check son clases mutables

¿Cuál es la diferencia entre las clases mutables y las inmutables?

Las clases mutables una vez que se ha creado un objeto de dicha clase este objeto puede cambiar su valor. Por el contrario en la clase inmutable es un objeto de esta clase una vez que ha sido creado ya no puede cambiar su valor tiene que ser sustituido por un objeto diferente.

Vemos entonces que las clases la mayoría son inmutables sin embargo tenemos las listas, los sets y los diccionarios como las tres excepciones a esta regla dentro de la clase predefinida mas comunes. Las listas, los diccionarios y los sets.

Clases numéricas

- Clase Bool
 - Nos sirve para representar valores Booleanos y sabemos que solo hay dos posibles instancias o valores que pueden haber True y False. Recordando que en python tenemos diferencia entre mayúsculas minúsculas, hay sensibilidad entonces true va con T mayúscula F mayuscula. I

- Las clases tienen un constructor con el mismo nombre de la clase. El constructor permite crear objetos de tipo bool a partir de otros objetos.
 - En el caso de que le enviemos números ya sea flotantes o números enteros se evaluarán a false se convertirán a false si son cero o si son diferentes de cero se convertirán en True.
 - En el caso de decir que le mandemos una secuencia el constructor de bool en la que se retornará un valor false si la secuencia está vacía de lo contrario se retornará un valor True.
- Clase Int
 - Con respecto a la clase y algo muy interesante es que no hay diferentes tipos de int, no tenemos int cortos ni largos. Simplemente tenemos una sola clase int que tiene magnitud arbitraria es decir el lenguaje que se cargará el motor del lenguaje se encargará de darle una capacidad almacenamiento suficiente para almacenar o para soportar el número que se esté manejando el número entero y éste solamente va a estar limitado por la memoria del computador. Por tanto podemos manejar entero muy grandes con python sin tener que hacer cambios. Obviamente esto tiene un impacto en la eficiencia del lenguaje, en python pues se prefiere o se da favor a la facilidad de programación y a la conveniencia más que a la eficiencia.
 - El constructor de como en la clase bool también tenemos un constructor de enteros. Este constructor nos permite construir un entero a partir de un número flotante truncando los por ejemplo 2.6 al mandarse el constructor de que retornaría un 2 lo trunca los corta no lo apróxima queda la parte entera,
 - También puede recibir cadenas válidas por ejemplo un string que contenga un número entero y si se hace la conversión automáticamente.
 - Clase Float
 - Es la única clase que nos permite manejar números flotantes con el punto decimal
 - En el caso de Float a diferencia del int tenemos una apreciación fija es decir que los Floats tienen una limitación, esa limitación está dada por la por la

implementación de python y habría que validarla en cada uno de los motores, un motor específico que se esté trabajando.

- En el caso de float también hay también
- Hay un constructor como en el resto, puedo construir floats a partir de enteros y puedo construir floats a partir de cadenas
- Por otro lado también puedo hacerlo con un booleano, float(True) retorno 1.0 y float(False) me retorna 0.0

Algo similar ocurrirá con el constructor de int int(True) me va a devolver 1 e int(False) me va a devolver 0

Ejemplo

Para ejecutar el intérprete de python simplemente tenemos que escribir el comando **python3** Mediante el intérprete de python podemos ejecutar código en python y obtener una retroalimentación inmediata del resultado de ese código.

Por ejemplo podemos crear un entero, asignarle 1 y ya fue asignado puedo ver el valor de las variables del objeto que yo creo simplemente escribiendo el nombre del objeto y presionando enter.

```
raul@nuc-unah:~$ python3
Python 3.6.9 (default, Apr 18 2020, 01:56:04)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> i = 1
>>> i
1
>>>
```

Por ejemplo el valor de i es 1. i es un número entero porque éste era un literal objeto entero por tanto y el tipo de i es entero. Podemos utilizar la función type para confirmar la clase

```
type(i)
##Clase int
```

ahora recordando los enteros son inmutables por esta razón si yo creo

un alias tenemos dos nombres para el mismo método pero si yo luego hago que i sea igual a cuatro por ejemplo, no estoy modificando el objeto uno simplemente estoy haciendo que el nombre i haga referencia esté haciendo referencia a un objeto nuevo al objeto j entonces si imprimo j muestra 1 porque el uno no fue modificado simplemente y se hizo referencia a un momento nuevo pero j seguía haciendo referencia al mismo objeto que inmutable

```
>>> type(i)
<class 'int'>
>>> j = i
>>> i = 4
>>> j
1
>>> type(j)
<class 'int'>
```

En el constructor de int tiene el mismo nombre de la clase y me permite construir enteros a partir de otros objetos por ejemplo a partir de un flotante le quitó la parte decimal

```
int(2.7)
```

2

Obtener a partir de un booleano no como True puedo obtener el número 1

```
int(True)
```

1

y a partir de una cadena por ejemplo puede obtener un número entero 234

```
int("234")
```

234

y obviamente si la cadena está más formada voy a obtener una excepción por ejemplo

```
>>> int("3f4r")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '3f4r'
>>> █
```

también tenemos el constructor de bool que acabamos de ver por ejemplo bool si recibe el entero uno construye un True esto yo obviamente lo puedo tener una variable y tener allí el valor de True.

```
>>> bool(1)
True
>>> b = bool(1)
>>> b
True
```

por ejemplo puedo construir un float a partir de un entero entero 45 yo tengo que ser igual 45 punto 0 entonces como podemos ver el intérprete de payton nos permite ejecutar código en python y dividió tener una retroalimentación inmediata de ese código esa es una de las ventajas de que python sea un lenguaje interpretado

```
>>> f = float(45)
>>> f
45.0
```

un lenguaje interpretado pues usualmente va a tener un intérprete significa que su código es traducido a lenguaje de máquina línea a línea cada una de estas líneas se traduce a diferencia del lenguaje compilado que se toma todo el código se traduce al lenguaje de máquina python no esto le da a python versatilidad, flexibilidad pero también esto tiene un impacto en la eficiencia sin embargo hay librería de payton que nos permiten ayudar a mejorar eso.

Clases Predefinidas Secuencias

Qué son las secuencias?

las secuencias son clases que sirven como contenedores de otros objetos y que a esos objetos que contienen les ponen un orden definido y por eso se llaman secuencias porque

existen elementos existe elemento uno el dos y así sucesivamente hasta la cantidad de elementos que tengan.

Las tres clases secuencias mas importantes en Python, son:

- list
- tuple
- str

Clase List

La clase list es una estructura referencial, qué significa esto?, significa que lo que almacena en realidad **la secuencia** son las referencias o los apuntadores o las direcciones de memoria de los objetos reales o de los objetos que pertenecen a la secuencia. Entonces las listas podemos considerarlas como arreglos porque almacenan en forma continua en memoria pero no los objetos en sí sino las referencias o los apuntadores a los objetos que pertenecen a las listas.

Que cosas pueden pertenecer a un list?

cualquier cosa cualquier objeto de python puede pertenecer a un list .

cualquier cosa se refiere a que pueden haber cosas mezcladas puede haber un entero puede haber un número un string etcétera. Ejemplo:



De manera continua en memorias estan guardado los apuntadores, la referencia pero el primer elemento es un entero el segundo en un string el tercero es un booleano

- las listas utilizan índices basados en cero. usan índice pasado en 0 eso quiere decir? que el primer elemento se tiene un índice 0 el segundo un segundo y así sucesivamente
- En python por defecto no existe el tipo array por lo que las listas son lo que utilice utilizaríamos en vez de un array que se utilizaría en otro lenguaje. Ahora sí pueden haber arrays pero esto tiene que ser con una clase o con una librería que importe el tipo de array.
- otra cosa importante de las listas que pueden aumentar su tamaño y disminuir su tamaño las listas son mutables recordando y pueden aumentar en su número de elementos y disminuir el tamaño máximo de una lista está definido o está limitado mejor dicho por el tamaño máximo la capacidad máxima de memoria en el computador.

Ejemplo de cómo definir una lista yo puedo hacerlo con el constructor de list pero esto es poco común le mandó al constructor un elemento. Lo más común utilizar los corchetes

```
a = list(1, 'hola', True)
a = [1, 'hola', True]
```

Clase tuple

- **Secuencias inmutables.** Esto permite que sus operaciones sean más eficientes.
- Igual que las listas están basadas en referencias.
- Utilizan los paréntesis en vez de los corchetes:
tuple = (1,2,3)
- Una tupla de un sólo elemento debe representarse usando una coma: tuple = (x,)
- ¿Se puede modificar un lista dentro de una tupla?

LA respuesta es si, porque la tupla lo que contienen son referencias, entonces al modificar una lista, puedo aumentarla de tamaño, pero su referencia sigue siendo la misma por tanto la tupla queda sin modificar pero la lista si puede modificarse.

Clase str

Clase str

- Secuencia inmutable de caracteres Unicode.
- Los caracteres pueden estar encerrados entre comillas dobles o simples.
- Es posible usar la contra-pleca como carácter de escape, pero para que forme parte de la cadena se debe escapar.

```
cadena = "Don't worry"  
cadena = 'Don\'t worry'  
ubicacion = 'C:\\user\\documents'
```

Unicode:

caracteres unicode lo que nos permite utilizar caracteres de distintos alfabetos para no solo el alfabeto latino ahora sino que el alfabeto **griego por ejemplo, chino.**

\sirve para \n, tabulador

dos contraplecas en un string implica una contrapleca en la realidad

Clase str

- Además es posible usar la triple comilla doble """ o simple "" para evitar escapar los saltos de línea.

```
cadena = """Este es un ejemplo de una cadena  
con múltiples saltos de línea que  
no es necesario escapar con contra-pleca.
```

```
Fin de la cadena."""
```

```
>>> l = [1,2,3]  
>>> l[0]  
'  
>>> l[0] = 4  
>>> l  
[4, 2, 3]  
>>> t = (1,2,3)  
>>> t[0]  
'  
>>> t[0] = 4  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'tuple' object does not support item assignment
```

```
>>> t = (1,2,l)  
>>> t[2]  
[4, 2, 3]  
>>> t[2][0] = 1  
>>> t  
(1, 2, [1, 2, 3])  
>>> s = "hola mundo"  
>>> s = '''Esta es una string con  
... multiples lineas  
...  
...'''  
>>> s  
'Esta es una string con\nmultiples lineas\n\n'  
>>> print(s)  
Esta es una string con  
multiples lineas
```

Clases Predefinidas Conjuntos y Diccionarios

Clases para representar conjuntos

Clases set y frozenset

- Representan una colección de elementos únicos y sin un orden inherente.
- Están optimizados para consultar si un elemento pertenece o no al conjunto.
- Sólo tipos inmutables pueden ser parte de un set o frozenset.
- Dado que los frozenset son inmutables, un set puede contener frozensets.

cuál es la función más importante

de los conjuntos su función más
importante es saber si un elemento
pertenece o no pertenece al conjunto

- Son elementos se delimitan utilizando las llaves {}. Aunque las llaves vacías {} no representan un conjunto vacío, sino un diccionario vacío.
- Es posible construir un conjunto a partir de una secuencia utilizando el constructor set().
- Si el constructor no recibe argumentos retorna un conjunto vacío.

```
conjunto = { 1, 2, 3, "hola mundo" }
```

```
conjunto = set([1, 1, 2, 2, 3, "hola mundo"])
```

Los conjuntos no repiten los elementos

Clase Dict

Claves, valor.

Clase dict

- Representan una colección de pares clave - valor.
- Las claves de un diccionario deber ser valores inmutables. Los valores no tienen este requisito. Las claves no se pueden repetir.
- Los elementos se delimitan usando las llaves {} y los dos puntos: diccionario = {"a":1, "b":2}
- El constructor de un diccionario puede recibir una lista de pares para crearlo:

```
diccionario = dict([('a', 1), ('b', 2)])
```