Lic. en Sistemas de Información

Inteligencia Artificial

TP 1 - Introducción a Python para IA

Consigna: Realizar los siguientes ejercicios y entregar el código en formato notebook.

EJERCICIO 1 - Minimización de una funcion utilizando la dirección negativa que indica el vector gradiente . ESCRIBIR CÓDIGO EN PYTHON COLAB

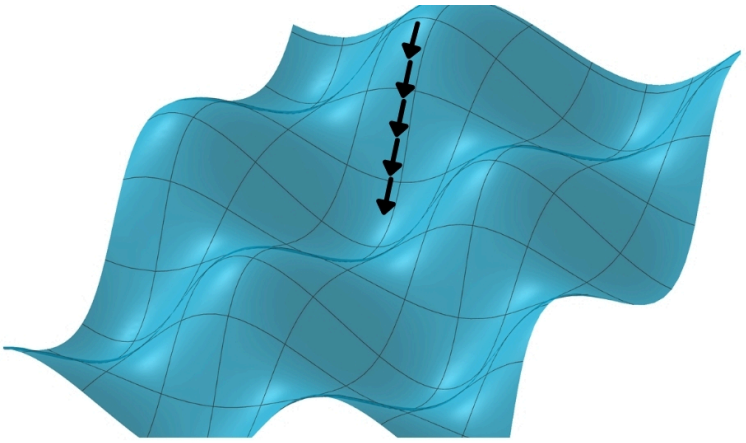


Fig 1. Gradiente descendente, proceso iterativo de búsqueda del mínimo

Se llama gradiente en un punto de una función real de varias variables reales al conjunto ordenado de las derivadas parciales de esa función en ese punto. El vector gradiente es utilizado en IA para buscar extremos en funciones de varias variables. Lo interesante del gradiente es que indica la dirección de máximo cambio de la función, por lo cual es útil para buscar extremos como máximos y mínimos.

El gradiente de una función $f(x, y, z)$ en el punto x_0, y_0, z_0 es

$\frac{\partial f}{\partial x}(x_0, y_0, z_0), \frac{\partial f}{\partial y}(x_0, y_0, z_0), \frac{\partial f}{\partial z}(x_0, y_0, z_0)$ para f definida en $\Re^3$. Este concepto se puede generalizar para espacios de mayores dimensiones. Considere la función f definida en la Fig 3.

BioIng. Hadad, A.
Lic. Díaz, Santiago R.

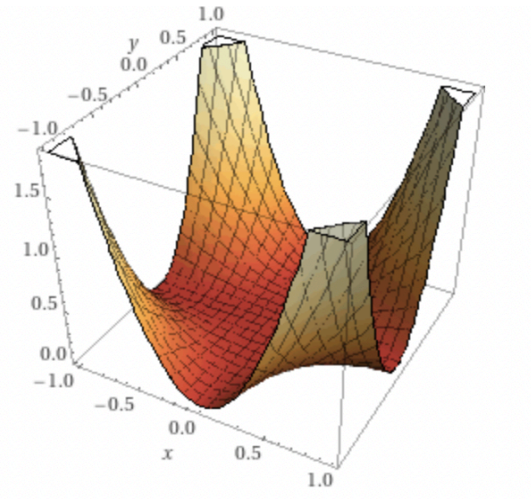


Fig 2. Funcion $f(x,y) = 3x^2y^2$

Realizar un proceso iterativo en python acotado al rango de la función de la Fig 3, que busque el mínimo desplazándose en la dirección del gradiente. Considere un desplazamiento de $dx = dy = 0.05$. Inicie en un valor aleatorio entre 0 y 1 para el par (x, y) muestre los valores de la función f con el siguiente formato en pantalla:

$Iteración\ 0$ ....

…

$Iteración\ i\colon\ par\ (x_i, y_i)\ ,\ valor\ función\ f\ \ (x_i, y_i)$

…

$Iteración\ 100$ ....

NOTA: Establecer un número máximo de iteraciones de $n = 100$.

Responder:

1. Explicar cómo funciona el algoritmo de descenso de gradiente
2. ¿Qué sucede con los factores multiplicativos?
3. ¿Puede realizar el ejercicio con una función que no es diferenciable en el rango en cuestión? explicar con código.

EJERCICIO 2 - CÁLCULOS MATRICIALES. ESCRIBIR CÓDIGO EN PYTHON COLAB.

El cálculo matricial es fundamental en la implementación de redes neuronales debido a su eficiencia para realizar operaciones lineales y transformaciones. Aquí tienes un ejemplo simple de cómo se utiliza el cálculo matricial en las redes neuronales, específicamente en la propagación hacia adelante (forward propagation) de una red neuronal básica con una sola capa oculta.

Imagina que tienes una red neuronal con:

- Una capa de entrada de 3 neuronas.
- Una capa oculta de 2 neuronas.
- Una capa de salida de 1 neurona.

Paso 1: Definir los datos de entrada

Supongamos que tienes un único vector de entrada X que representa tres características de un elemento: $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

Paso 2: Definir los pesos

Los pesos entre la capa de entrada y la capa oculta se representan con la matriz $W^{(1)}$ y los pesos entre la capa oculta y la capa de salida con $W^{(2)}$: $W^{(1)} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$ $W^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{bmatrix}$

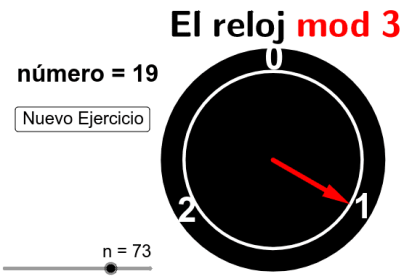
Paso 3: Cálculo de la activación de la capa oculta

La entrada a las neuronas de la capa oculta se calcula como $Z^{(1)} = W^{(1)} \cdot X$: $Z^{(1)} = \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \end{bmatrix}$ Luego aplicamos una función de activación, por ejemplo, la función ReLU, a cada elemento de $Z^{(1)}$: $A^{(1)} = \mathrm{ReLU}(Z^{(1)})$

Paso 4: Cálculo de la salida de la red

Finalmente, la entrada a la neurona de salida se calcula como $Z^{(2)} = W^{(2)} \cdot A^{(1)}$ y se aplica otra función de activación (por ejemplo, sigmoide para problemas de clasificación binaria): $Z^{(2)} = w_{11}^{(2)}a_{11} + w_{21}^{(2)}a_{21}$ $A^{(2)} = \sigma(Z^{(2)})$

EJERCICIO 3 - ANILLO. ESCRIBIR CÓDIGO EN PYTHON COLAB



Sea el reloj de la figura, escribir una función para encontrar el valor en el anillo Z_k para cualquier $n \geq 0$. pruebe el código con $n = 73$ y $k = 3$. Cree un array aleatorio de números enteros aleatorios

de 20.000 elementos, computar el anillo y medir el tiempo que tarda la operación sobre todos los elementos del array. Grafique el tiempo acumulado en cada iteración, muestre gráficamente que el costo de computación es lineal en un gráfico donde x = n de interacción, y $t = TA$ (tiempo acumulado), donde $t = 0$ en $x = 0$.

Nota: para graficar utilizar sanborns o matplotlib, investigar las librerías e implementar el gráfico en el código.

EJERCICIO 4 - HIPERPLANO. ESCRIBIR CÓDIGO EN PYTHON COLAB

Ejercicio: Determinar el hiperplano que separa dos conjuntos de puntos en $\mathbb{R}^3$

Dados los siguientes conjuntos de puntos en el espacio tridimensional $\mathbb{R}^3$:

- Conjunto A: $(1, 2, 3)$, $(2, 3, 5)$, $(3, 4, 7)$
- Conjunto B: $(-1, -2, -1)$, $(-2, -4, -2)$, $(-3, -6, -3)$

Determina la ecuación de un hiperplano que pueda separar efectivamente el Conjunto A del Conjunto B. Asegúrate de verificar que todos los puntos de un conjunto estén de un lado del hiperplano y todos los puntos del otro conjunto estén del lado opuesto.

Puede repasar conceptos en

- https://docs.python.org/es/3/tutorial/