Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor : Luz Magaly Turpo Mamani
Link github: https://github.com/luz897/ACTIVIDAD-02/

Trabajo Encargado - N° 002

Actividad N° 2 - Restricciones

Desarrolle los siguientes enunciados de acuerdo con el objetivo planteado en cada problema, ya sea maximización o minimización. Además, implemente una solución computacional utilizando el lenguaje de programación o framework de su preferencia, con el propósito de generar una solución genérica que permita la visualización gráfica de los resultados en todos los casos.

Ejercicio 01

Un algoritmo necesita procesar datos en lotes. Cada lote requiere x MB de memoria, pero la capacidad total de memoria disponible es de 1024 MB. El algoritmo puede procesar un máximo de 8 lotes. El objetivo es maximizar la cantidad de datos procesados, pero cada lote más allá del quinto reduce su eficiencia en un 20 %.

Formulación del problema

- Sea x el tamaño de los datos en MB que requiere cada lote.
- La capacidad total de memoria disponible es de 1024 MB.
- El algoritmo puede procesar un máximo de 8 lotes.
- Para los primeros 5 lotes, el tamaño total de los datos procesados es $5x$.
- Para los lotes 6 al 8, el tamaño de datos procesados es reducido en un 20 %, es decir, el tamaño procesado por cada uno de estos lotes es $0{,}8x$.

El objetivo es maximizar los datos procesados totales, que se expresan de la siguiente manera:

$$f(x) = 5x + 3(0{,}8x) = 5x + 2{,}4x = 7{,}4x$$

El total de memoria usada no puede exceder los 1024 MB, lo que se expresa como la siguiente restricción:

$$8x \leq 1024$$

De aquí, despejamos x:

$$x \leq \frac{1024}{8} = 128\,\text{MB}$$

El tamaño máximo de cada lote es $x = 128$ MB. Sustituyendo este valor en la función objetivo:

$$f(128) = 7{,}4 \times 128 = 947{,}2\,\text{MB}$$

Conclusión

La cantidad máxima de datos que puede procesar el algoritmo es 947,2 MB, teniendo en cuenta la reducción de eficiencia para los lotes a partir del sexto.

```python
Ejercicio01.py > ...
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

memoria_total = 1024

def calcular_datos_procesados(lotes, x):
    if lotes <= 5:
        return lotes * x
    else:

        return 5 * x + (lotes - 5) * 0.8 * x

st.title('Maximización de Datos Procesados por Lotes')

st.write('Este programa maximiza la cantidad de datos procesados considerando que a partir del lote 6 la eficiencia cae un 20%.')

lotes = st.slider('Seleccione el número de lotes (1 a 8)', min_value=1, max_value=8, value=5)

x = memoria_total / lotes

datos_procesados = calcular_datos_procesados(lotes, x)

st.write(f'Número de lotes: {lotes}')
st.write(f'Tamaño de cada lote: {x:.2f} MB')
st.write(f'Datos procesados: {datos_procesados:.2f} MB')

lotes_x = np.arange(1, 9)
datos_y = [calcular_datos_procesados(lote, memoria_total / lote) for lote in lotes_x]

fig, ax = plt.subplots()
ax.plot(lotes_x, datos_y, marker='o', linestyle='-', color='purple', label='Datos procesados')

for i, txt in enumerate(datos_y):
    ax.annotate(f'{txt:.2f}', (lotes_x[i], datos_y[i]), textcoords="offset points", xytext=(0,10), ha='center')

ax.set_xlabel('Número de lotes')
ax.set_ylabel('Datos procesados (MB)')
ax.set_title('Cantidad de datos procesados vs Número de lotes')
ax.grid(True)
ax.legend()

st.pyplot(fig)
```

Figura 1: Código python

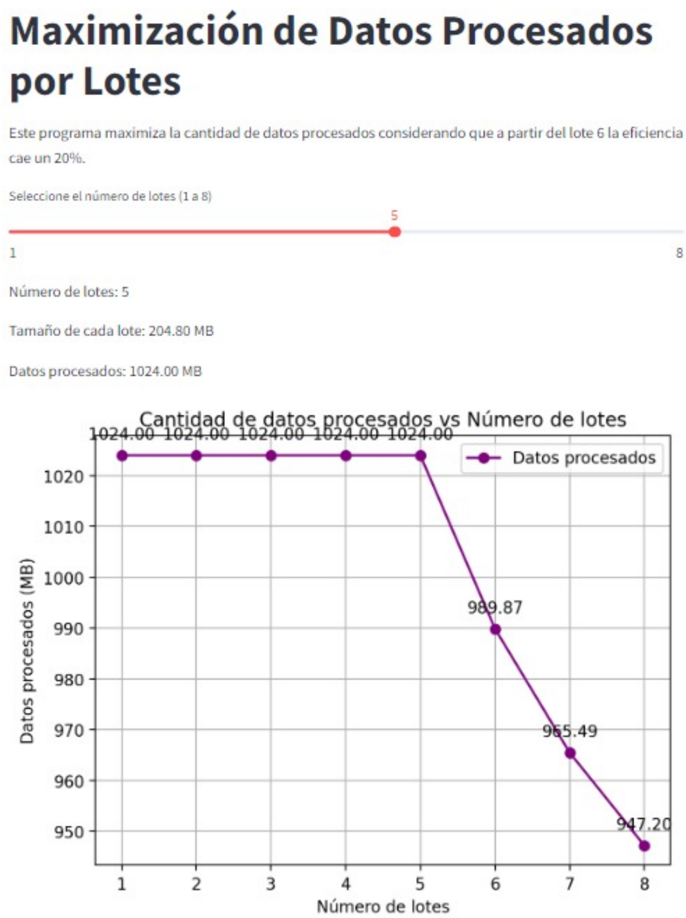

Figura 2: Gráfico

Ejercicio 02

Un sistema distribuido tiene 20 nodos. Cada nodo puede procesar x peticiones por segundo. El sistema en su conjunto no puede procesar más de 400 peticiones por segundo debido a limitaciones de red. Maximiza el número de peticiones procesadas sin exceder la capacidad de la red.

Formulación del problema

- Sea x el número de peticiones por segundo que puede procesar cada nodo.
- El sistema tiene un total de 20 nodos. Entonces, el número total de peticiones procesadas por el sistema se expresa como:

El sistema tiene un total de 20 nodos. Entonces, el número total de peticiones procesadas por el sistema se expresa como:

$$f(x) = 20x$$

El sistema no puede procesar más de 400 peticiones por segundo, por lo tanto, la restricción es:

$$20x \leq 400$$

Despejamos x:

$$x \leq \frac{400}{20} = 20$$

Sustituyendo $x = 20$ en la función objetivo:

$$f(20) = 20 \times 20 = 400 \text{ peticiones por segundo.}$$

Conclusión

La cantidad máxima de peticiones que puede procesar el sistema es de 400 peticiones por segundo, respetando la limitación impuesta por la red.

```
Ejercicio02.py > ...
import streamlit as st
import matplotlib.pyplot as plt

def calcular_peticiones(nodos, x):
    return min(nodos * x, 400)

st.title('Maximización de Peticiones Procesadas por un Sistema Distribuido')
st.write('Este programa calcula la cantidad máxima de peticiones procesadas por un sistema distribuido de nodos.')

nodos = st.slider('Seleccione el número de nodos (máx. 20)', min_value=1, max_value=20, value=10)
x = st.slider('Seleccione el número de peticiones por segundo que procesa cada nodo (máx. 20)', min_value=1, max_value=20, value=20)

peticiones_procesadas = calcular_peticiones(nodos, x)
st.write(f'Cantidad de peticiones procesadas: {peticiones_procesadas:.2f} peticiones por segundo')

fig, ax = plt.subplots()

nodos_x = list(range(1, 21))
peticiones_y = [calcular_peticiones(i, x) for i in nodos_x]

ax.plot(nodos_x, peticiones_y, marker='o', linestyle='--', color='purple', linewidth=2, markersize=8, label='Peticiones Procesadas')

for i, txt in enumerate(peticiones_y):
    ax.annotate(f'{txt:.2f}', (nodos_x[i], peticiones_y[i]), textcoords="offset points", xytext=(0,10), ha='center', fontsize=10, color='black')

ax.set_xlabel('Número de Nodos', fontsize=12, color='black')
ax.set_ylabel('Peticiones Procesadas', fontsize=12, color='black')
ax.set_title('Peticiones Procesadas en función del número de nodos', fontsize=14, color='black')
ax.grid(True, linestyle='-.', linewidth=0.7)

ax.set_xlim(1, 20)
ax.set_ylim(0, max(peticiones_y) + 50)
ax.legend()

st.pyplot(fig)
```

Figura 3: Código python

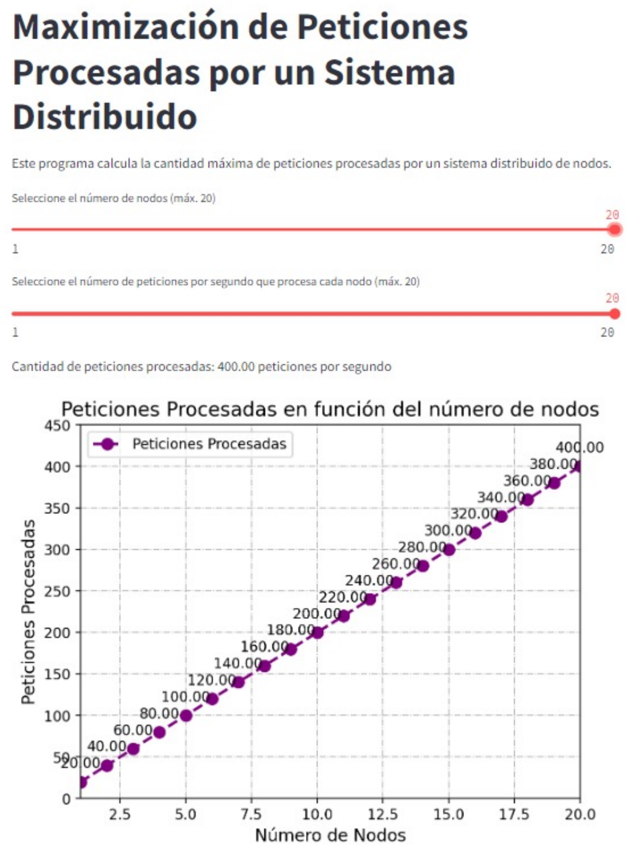

Figura 4: Gráfico

Ejercicio 03

Un script de Python tarda $5x+2$ segundos en procesar x datos. Por cada dato adicional, el tiempo de ejecución crece linealmente. Sin embargo, el sistema tiene un límite de tiempo de ejecución de 50 segundos. ¿Cuál es el número máximo de datos que puede procesar el script?

Formulación del Problema

Sea $t(x) = 5x + 2$, donde $t(x)$ es el tiempo total en segundos para procesar x datos.
La restricción es que el tiempo total no puede exceder los 50 segundos:

$$5x + 2 \leq 50$$

Despejando x:

$$5x \leq 48$$
$$x \leq \frac{48}{5} = 9{,}6$$

Como x debe ser un valor entero, el número máximo de datos que puede procesar el script es $x = 9$.

Conclusión

El número máximo de datos que puede procesar el script es 9, manteniendo el tiempo de ejecución dentro del límite de 50 segundos.

```python
import streamlit as st
import matplotlib.pyplot as plt

def tiempo_ejecucion(x):
    return 5 * x + 2

st.title('Maximización de Datos Procesados por el Script')
st.write('Este programa calcula el número máximo de datos que puede procesar el script sin exceder los 50 segundos.')

x = st.slider('Seleccione el número de datos (máx. 9)', min_value=1, max_value=9, value=5)

tiempo = tiempo_ejecucion(x)
st.write(f'Tiempo de ejecución: {tiempo:.2f} segundos')

fig, ax = plt.subplots()

datos_x = list(range(1, 10))
tiempo_y = [tiempo_ejecucion(i) for i in datos_x]

ax.plot(datos_x, tiempo_y, marker='o', linestyle='--', color='purple', linewidth=2, markersize=8, label='Tiempo de Ejecución')

for i, txt in enumerate(tiempo_y):
    ax.annotate(f'{txt:.2f}', (datos_x[i], tiempo_y[i]), textcoords="offset points", xytext=(0,10), ha='center', fontsize=10, color='black')

ax.set_xlabel('Número de Datos', fontsize=12, color='black')
ax.set_ylabel('Tiempo de Ejecución (segundos)', fontsize=12, color='black')
ax.set_title('Tiempo de Ejecución en función del número de datos', fontsize=14, color='black')
ax.grid(True, linestyle='-.', linewidth=0.7)

ax.set_xlim(1, 9)
ax.set_ylim(0, max(tiempo_y) + 10)
ax.legend()

st.pyplot(fig)
```

Figura 5: Código python

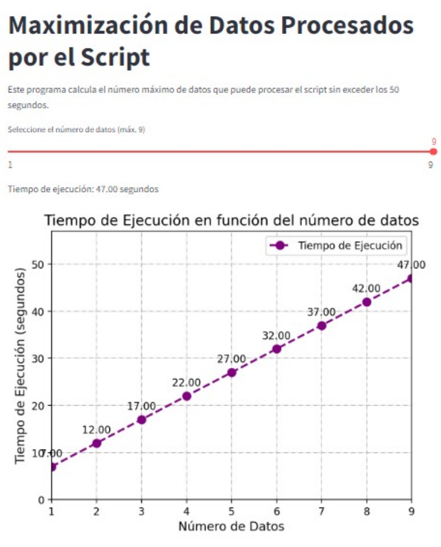

Figura 6: Gráfico

Ejercicio 04

Un servidor web procesa x peticiones por segundo, y el uso de la CPU sigue la fórmula:

$$u(x) = 2x^2 + 10x$$

donde $u(x)$ representa el porcentaje de uso de la CPU. La CPU no puede exceder el 80 % de uso. El objetivo es minimizar el uso de CPU sin caer por debajo del umbral de procesamiento de 10 peticiones por segundo.

Solución

Sabemos que la CPU no puede exceder el 80 % de uso, por lo tanto, planteamos la siguiente restricción:

$$u(x) \leq 80$$

Sustituyendo la expresión del uso de la CPU:

$$2x^2 + 10x \leq 80$$

Ahora, despejamos esta desigualdad. Primero restamos 80 de ambos lados de la ecuación:

$$2x^2 + 10x - 80 \leq 0$$

Esta es una ecuación cuadrática que podemos resolver aplicando la fórmula general para ecuaciones cuadráticas. La fórmula general es:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

donde $a = 2$, $b = 10$, y $c = -80$.

Primero, calculamos el discriminante:

$$\Delta = b^2 - 4ac = 10^2 - 4(2)(-80) = 100 + 640 = 740$$

Luego, calculamos las raíces de la ecuación cuadrática:

$$x_1 = \frac{-10 + \sqrt{740}}{2(2)} = \frac{-10 + 27{,}2}{4} = \frac{17{,}2}{4} = 4{,}3$$

$$x_2 = \frac{-10 - \sqrt{740}}{2(2)} = \frac{-10 - 27{,}2}{4} = \frac{-37{,}2}{4} = -9{,}3$$

Obtenemos dos soluciones: $x_1 = 4{,}3$ y $x_2 = -9{,}3$. Como el número de peticiones por segundo no puede ser negativo, descartamos x_2.

Además, dado que el umbral mínimo de procesamiento es de 10 peticiones por segundo, verificamos si con $x = 10$ se cumple la restricción. Sustituimos $x = 10$ en la fórmula para el uso de CPU:

$$u(10) = 2(10)^2 + 10(10) = 2(100) + 100 = 200 + 100 = 300$$

Esto significa que, al procesar 10 peticiones por segundo, el uso de la CPU sería del 300 %, lo cual excede el límite del 80 %. Por lo tanto, $x = 10$ no es una solución válida para cumplir con la restricción.

De esta manera, la cantidad máxima de peticiones que se puede procesar sin exceder el 80 % de uso de la CPU es aproximadamente 4.3 peticiones por segundo.

Conclusión

La solución es que el servidor puede procesar aproximadamente 4.3 peticiones por segundo sin exceder el 80 % de uso de la CPU. Este es el número máximo de peticiones permitido bajo las condiciones establecidas.

```
Ejercicio04.py > ...
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt

def uso_cpu(x):
    return 2 * x**2 + 10 * x

def resolver_ecuacion(a, b, c):
    discriminante = b**2 - 4*a*c
    if discriminante >= 0:
        raiz_discriminante = np.sqrt(discriminante)
        x1 = (-b + raiz_discriminante) / (2*a)
        x2 = (-b - raiz_discriminante) / (2*a)
        return x1, x2
    else:
        return None, None

st.title('Minimización del Uso de CPU en un Servidor Web')
st.write('Este programa calcula el número de peticiones por segundo que minimiza el uso de CPU sin exceder el 80% de capacidad.')

x = st.slider('Seleccione el número de peticiones por segundo (mín. 10)', min_value=10, max_value=20, value=10)

uso = uso_cpu(x)
st.write(f'Uso de CPU: {uso:.2f}%')

x1, x2 = resolver_ecuacion(2, 10, -80)
st.write(f'Los valores de x que cumplen con la restricción de CPU son: {x1:.2f} y {x2:.2f}')

fig, ax = plt.subplots()

datos_x = np.linspace(10, 20, 100)
cpu_y = [uso_cpu(i) for i in datos_x]

ax.plot(datos_x, cpu_y, marker='o', linestyle='--', color='purple', linewidth=2, markersize=4, label='Uso de CPU')

ax.set_xlabel('Peticiones por Segundo', fontsize=12, color='black')
ax.set_ylabel('Uso de CPU (%)', fontsize=12, color='black')
ax.set_title('Uso de CPU en función de las Peticiones por Segundo', fontsize=14, color='black')
ax.grid(True, linestyle='-.', linewidth=0.7)

ax.set_xlim(10, 20)
ax.set_ylim(0, max(cpu_y) + 10)
ax.legend()

st.pyplot(fig)
```

Figura 7: Código python

Figura 8: Gráfico

Ejercicio 05

Durante el entrenamiento de un modelo de machine learning, el batch size x afecta el tiempo de entrenamiento según la fórmula:

$$T(x) = \frac{1000}{x} + 0{,}1x$$

El tamaño del lote debe estar entre 16 y 128. Encuentra el tamaño de batch que minimiza el tiempo de entrenamiento.

Solución

Queremos minimizar el tiempo de entrenamiento $T(x)$. Para esto, debemos derivar la función y encontrar el valor crítico dentro del rango $16 \leq x \leq 128$.

La derivada de $T(x)$ con respecto a x es:

$$\frac{dT}{dx} = -\frac{1000}{x^2} + 0{,}1$$

Para encontrar los valores críticos, igualamos la derivada a cero:

$$-\frac{1000}{x^2} + 0{,}1 = 0$$

Despejamos x:

$$\frac{1000}{x^2} = 0{,}1$$

$$x^2 = \frac{1000}{0{,}1} = 10000$$

$$x = \sqrt{10000} = 100$$

El tamaño de batch que minimiza el tiempo de entrenamiento es $x = 100$.
Evaluamos el tiempo de entrenamiento en los extremos del rango $[16, 128]$:
- Para $x = 16$:

$$T(16) = \frac{1000}{16} + 0{,}1(16) = 64{,}1$$

- Para $x = 128$:

$$T(128) = \frac{1000}{128} + 0{,}1(128) = 20{,}61$$

- Para $x = 100$:

$$T(100) = \frac{1000}{100} + 0{,}1(100) = 20$$

Dado que $T(100) = 20$ es menor que los valores en los extremos, concluimos que $x = 100$ es el tamaño de batch que minimiza el tiempo de entrenamiento.

Conclusión

El tamaño de batch óptimo que minimiza el tiempo de entrenamiento es $x = 100$.

```python
Ejercicio05.py > ...
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

def tiempo_entrenamiento(x):
    return 1000 / x + 0.1 * x

st.title("Minimización del Tiempo de Entrenamiento en Función del Batch Size")
st.write(r'La función de tiempo de entrenamiento es $T(x) = \frac{1000}{x} + 0.1x$')
st.write("El tamaño del batch debe estar entre 16 y 128.")

limites = (16, 128)

resultado = minimize(lambda x: tiempo_entrenamiento(x[0]), x0=[64], bounds=[limites])

st.write(f"El tamaño de batch que minimiza el tiempo de entrenamiento es: {resultado.x[0]:.2f}")

x_vals = np.linspace(16, 128, 500)
y_vals = tiempo_entrenamiento(x_vals)

plt.figure(figsize=(8, 5))
plt.plot(x_vals, y_vals, label='T(x)', color='purple')
plt.axvline(resultado.x[0], color='red', linestyle='--', label=f'Batch size óptimo: {resultado.x[0]:.2f}')
plt.xlabel('Batch Size (x)')
plt.ylabel('Tiempo de Entrenamiento (T)')
plt.title('Minimización del Tiempo de Entrenamiento')
plt.legend()
plt.grid()

st.pyplot(plt)
```

Figura 9: Código python

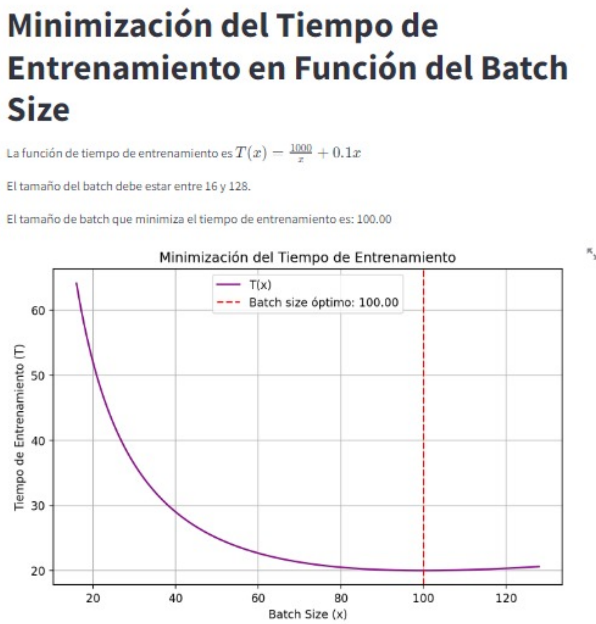

Figura 10: Gráfico

Ejercicio 06

Un sistema de transmisión de datos tiene un ancho de banda total de 1000 Mbps. Cada archivo que se transmite utiliza x Mbps. El sistema puede transmitir un máximo de 50 archivos a la vez, y cada archivo adicional más allá de 30 reduce el ancho de banda disponible en un 5 %. Maximiza el número de archivos transmitidos.

Solución

Datos del Problema

- Ancho de banda total: 1000 Mbps.
- Ancho de banda utilizado por archivo: x Mbps.
- Máximo de archivos que se pueden transmitir: 50.
- Reducción del ancho de banda: 5 % por cada archivo adicional más allá de 30.

Para n archivos, el ancho de banda disponible es:

$$A = \begin{cases} 1000 & \text{si } n \leq 30 \\ 1000 \times (1 - 0{,}05(n - 30)) & \text{si } n > 30 \end{cases}$$

Para $n > 30$:

$$A = 1000 - 50(n - 30) = 3000 - 50n$$

Para que el sistema pueda transmitir los archivos:

$$n \cdot x \leq A$$

Sustituyendo A:

$$n \cdot x \leq 3000 - 50n$$

Reorganizando:

$$n(x + 50) \leq 3000$$

Por lo tanto, el número máximo de archivos transmitidos está dado por:

$$n \leq \frac{3000}{x + 50}$$

Para maximizar el número de archivos, minimizamos x. Si consideramos que x es un valor pequeño, por ejemplo, $x = 1$:

$$n \leq \frac{3000}{1 + 50} = \frac{3000}{51} 58{,}82$$

Sin embargo, el máximo número de archivos que se pueden transmitir es $n = 50$.

Conclusión

El número máximo de archivos que se puede transmitir en el sistema es $n = 50$.

```python
Ejercicio06.py > ...
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt

st.title("Maximización de Archivos Transmitidos")

ancho_banda_total = 1000
max_archivos = 50
reduction_start = 30
reduction_factor = 0.05

x = st.number_input("Ancho de banda utilizado por cada archivo (Mbps)", min_value=1, max_value=100, value=20)

def calcular_archivos_maximos(x):
    for n_archivos in range(1, max_archivos + 1):
        if n_archivos <= reduction_start:
            ancho_banda_utilizado = n_archivos * x
        else:
            ancho_banda_utilizado = n_archivos * x * (1 - reduction_factor)

        if ancho_banda_utilizado > ancho_banda_total:
            return n_archivos - 1

    return max_archivos

max_archivos_transmitidos = calcular_archivos_maximos(x)

st.write(f"El número máximo de archivos que se pueden transmitir es: {max_archivos_transmitidos}")

n_archivos_range = np.arange(1, max_archivos + 1)
ancho_banda_utilizado = []

for n_archivos in n_archivos_range:
    if n_archivos <= reduction_start:
        ancho_banda_utilizado.append(n_archivos * x)
    else:
        ancho_banda_utilizado.append(n_archivos * x * (1 - reduction_factor))

plt.figure(figsize=(8, 5))
plt.plot(n_archivos_range, ancho_banda_utilizado, label='Ancho de banda utilizado', color='purple')
plt.axhline(ancho_banda_total, color='red', linestyle='--', label='Ancho de banda total (1000 Mbps)')
plt.xlabel('Número de archivos transmitidos')
plt.ylabel('Ancho de banda utilizado (Mbps)')
plt.title('Uso del Ancho de Banda en Función del Número de Archivos')
plt.legend()
plt.grid()

st.pyplot(plt)
```

Figura 11: Código python

Figura 12: Gráfico

Ejercicio 07

Un sistema de colas procesa x trabajos por segundo. La función del tiempo de respuesta está dada por:

$$T(x) = \frac{100}{x} + 2x$$

Minimiza el tiempo de respuesta del sistema, considerando que el sistema debe procesar al menos 5 trabajos por segundo.

Solución

La función que describe el tiempo de respuesta es:

$$T(x) = \frac{100}{x} + 2x$$

Dado que el sistema debe procesar al menos 5 trabajos por segundo:

$$x \geq 5$$

Para minimizar $T(x)$, calculamos la derivada:

$$T'(x) = -\frac{100}{x^2} + 2$$

Igualando a cero:

$$-\frac{100}{x^2} + 2 = 0$$

$$2 = \frac{100}{x^2}$$

$$2x^2 = 100$$

$$x^2 = 50$$

$$x = \sqrt{50} = 7{,}07$$

Evaluamos $T(x)$ en $x = 5$ y $x = 7{,}07$:

$$T(5) = \frac{100}{5} + 2 \times 5 = 20 + 10 = 30$$

$$T(7{,}07) = \frac{100}{7{,}07} + 2 \times 7{,}07 = 14{,}14 + 14{,}14 = 28{,}28$$

Comparando los resultados:

- $T(5) = 30$
- $T(7{,}07) = 28{,}28$

Conclusión

El valor óptimo para minimizar el tiempo de respuesta es $x = 7{,}07$ trabajos por segundo.

```python
Ejercicio07.py > ...
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt

def tiempo_respuesta(x):
    return (100 / x) + 2 * x

st.title("Minimización del Tiempo de Respuesta en un Sistema de Colas")
st.write("Este programa calcula el tiempo de respuesta para diferentes tasas de procesamiento.")

x = st.slider("Selecciona la tasa de procesamiento (trabajos por segundo):", min_value=5, max_value=20, value=5)

T_x = tiempo_respuesta(x)
st.write(f"El tiempo de respuesta para {x} trabajos por segundo es: {T_x:.2f} segundos.")

fig, ax = plt.subplots()

datos_x = np.linspace(5, 20, 100)
tiempo_y = [tiempo_respuesta(i) for i in datos_x]

ax.plot(datos_x, tiempo_y, marker='o', linestyle='--', color='purple', linewidth=2, markersize=4, label='Tiempo de Respuesta')

ax.set_xlabel('Tasa de Procesamiento (trabajos por segundo)', fontsize=12, color='black')
ax.set_ylabel('Tiempo de Respuesta (segundos)', fontsize=12, color='black')
ax.set_title('Tiempo de Respuesta en Función de la Tasa de Procesamiento', fontsize=14, color='black')
ax.grid(True, linestyle='-.', linewidth=0.7)


ax.set_xlim(5, 20)
ax.set_ylim(0, max(tiempo_y) + 10)
ax.legend()

st.pyplot(fig)
```

Figura 13: Código python

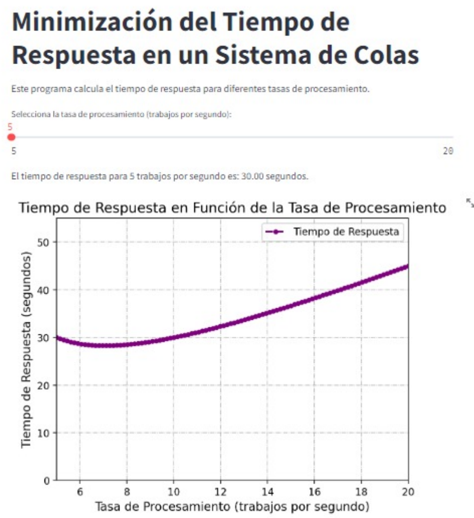

Figura 14: Gráfico

Ejercicio 08

El entrenamiento de un modelo de deep learning en una GPU consume x unidades de energía por lote. El objetivo es maximizar el tamaño del lote x, pero el consumo de energía total no puede exceder las 200 unidades por segundo, y cada lote adicional más allá del 10 reduce el rendimiento en un 10

Planteamiento del problema

Queremos maximizar el tamaño del lote x durante el entrenamiento de un modelo de *deep learning* bajo las siguientes condiciones:

- El consumo de energía total no puede exceder las 200 unidades por segundo.
- Cada lote adicional más allá de un tamaño de lote de 10 reduce el rendimiento en un 10 %.

El consumo de energía por lote se puede modelar de la siguiente forma:

$$E = 10x$$

donde x es el tamaño del lote y 10 es el consumo por lote. La restricción de energía es:

$$E \leq 200 \quad \Rightarrow \quad 10x \leq 200$$

Despejando x:

$$x \leq \frac{200}{10} = 20$$

Por lo tanto, $1 \leq x \leq 20$.

El rendimiento en función del tamaño del lote x está dado por:

$$f(x) = \begin{cases} x & \text{si } x \leq 10 \\ x - 0{,}1(x - 10) & \text{si } x > 10 \end{cases}$$

Para $x > 10$:

$$f(x) = 0{,}9x + 1$$

Nuestro objetivo es maximizar $f(x)$ bajo la restricción $1 \leq x \leq 20$.
Evaluamos los puntos clave:

- Para $x = 10$, $f(10) = 10$.
- Para $x = 20$, $f(20) = 0{,}9(20) + 1 = 19$.

El rendimiento es creciente en ambos casos, por lo que el tamaño del lote óptimo es $x = 20$, lo que maximiza el rendimiento a $f(20) = 19$.

Conclusión

El tamaño del lote óptimo es $x = 20$, con un rendimiento máximo de 19 unidades. Este tamaño de lote consume exactamente 200 unidades de energía, cumpliendo con la restricción del problema.


Ejercicio08.py > ...

```python
import streamlit as st
import matplotlib.pyplot as plt

def calcular_rendimiento(x):
    if x <= 10:
        return x
    else:
        return x - 0.1 * (x - 10)
def calcular_energia(x):
    return x * 10

st.title('Maximización del Tamaño del Lote en Entrenamiento de Deep Learning')
st.write('Este programa calcula el tamaño de lote óptimo para maximizar el rendimiento bajo una restricción de consumo de energía.')

x = st.slider('Seleccione el tamaño del lote', min_value=1, max_value=20, value=10)

energia_consumida = calcular_energia(x)
rendimiento = calcular_rendimiento(x)

if energia_consumida > 200:
    st.write(f'Consumo de energía excedido: {energia_consumida} unidades. El tamaño del lote debe reducirse.')
else:
    st.write(f'Rendimiento para un lote de tamaño {x}: {rendimiento:.2f} unidades.')
    st.write(f'Consumo de energía: {energia_consumida} unidades.')

# Gráfico
fig, ax = plt.subplots()

lotes_x = list(range(1, 21))
rendimiento_y = [calcular_rendimiento(i) for i in lotes_x]

ax.plot(lotes_x, rendimiento_y, marker='o', linestyle='--', color='purple', linewidth=2, markersize=8, label='Rendimiento')

for i, txt in enumerate(rendimiento_y):
    ax.annotate(f'{txt:.2f}', (lotes_x[i], rendimiento_y[i]), textcoords="offset points", xytext=(0,10), ha='center', fontsize=10, color='black')

ax.set_xlabel('Tamaño del Lote', fontsize=12, color='black')
ax.set_ylabel('Rendimiento', fontsize=12, color='black')
ax.set_title('Rendimiento en función del tamaño del lote', fontsize=14, color='black')
ax.grid(True, linestyle='-.', linewidth=0.7)

ax.set_xlim(1, 20)
ax.set_ylim(0, max(rendimiento_y) + 5)
ax.legend()

st.pyplot(fig)
```

Figura 15: Código python

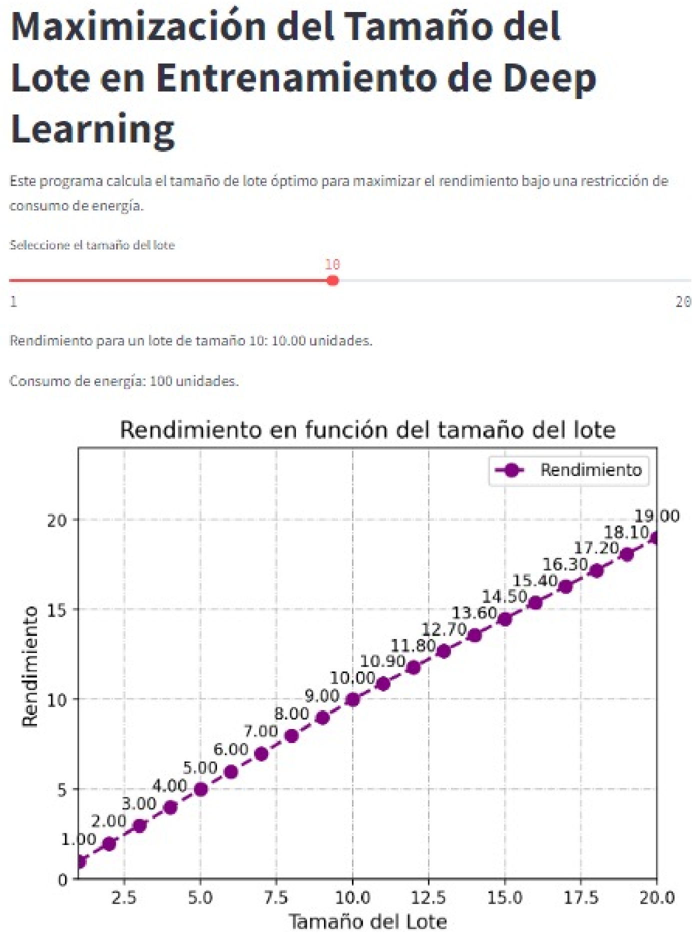

Figura 16: Gráfico

Ejercicio 09

Una empresa almacena datos en la nube. El costo de almacenamiento por TB es de $C(x) = 50 + 5x$ dólares, donde x es la cantidad de TB de almacenamiento utilizado. La empresa tiene un presupuesto de 500 dólares. Maximice la cantidad de datos almacenados sin exceder el presupuesto.

Solución

La función de costo total está dada por:

$$C(x) = 50 + 5x$$

donde x es la cantidad de terabytes (TB) almacenados y $C(x)$ es el costo total en dólares.

La restricción es que el costo total no puede exceder el presupuesto de 500 dólares:

$$C(x) \leq 500$$

Despejamos x en términos de $C(x)$:

$$50 + 5x \leq 500$$

Restamos 50 en ambos lados:

$$5x \leq 450$$

Dividimos entre 5:

$$x \leq \frac{450}{5} = 90$$

Conclusión

Por lo tanto, la cantidad máxima de datos que la empresa puede almacenar sin exceder el presupuesto es de **90** TB.

```
Ejercicio09.py > ...
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

def calcular_costo(x):
    return 50 + 5 * x

presupuesto = 500

st.title('Maximización del Almacenamiento de Datos en la Nube')

st.write('Este programa maximiza la cantidad de datos almacenados (en TB) sin exceder el presupuesto de 500 dólares.')

x = st.slider('Seleccione la cantidad de almacenamiento en TB (1 ≤ x ≤ 100)', min_value=1, max_value=100, value=10)

costo = calcular_costo(x)

st.write(f'Almacenamiento seleccionado: {x} TB')
st.write(f'Costo total: {costo} dólares')

if costo > presupuesto:
    st.warning(f'Advertencia: El costo excede el presupuesto de {presupuesto} dólares.')

almacenamiento_x = np.arange(1, 101)
costo_y = [calcular_costo(i) for i in almacenamiento_x]

fig, ax = plt.subplots()

ax.plot(almacenamiento_x, costo_y, marker='o', linestyle='-', color='purple', label='Costo de Almacenamiento', linewidth=2)
ax.set_xlabel('Almacenamiento (TB)')
ax.set_ylabel('Costo (dólares)')
ax.axhline(presupuesto, color='red', linestyle='--', linewidth=1, label=f'Presupuesto: {presupuesto} dólares')

ax.grid(True)

ax.set_ylim(0, max(costo_y) + 50)
ax.set_xlim(1, 100)

for i, txt in enumerate(costo_y):
    if i % 10 == 0:
        ax.text(almacenamiento_x[i], costo_y[i] + 10, f'{txt}', fontsize=8, ha='center')

ax.legend()

st.pyplot(fig)
```

Figura 17: Código python

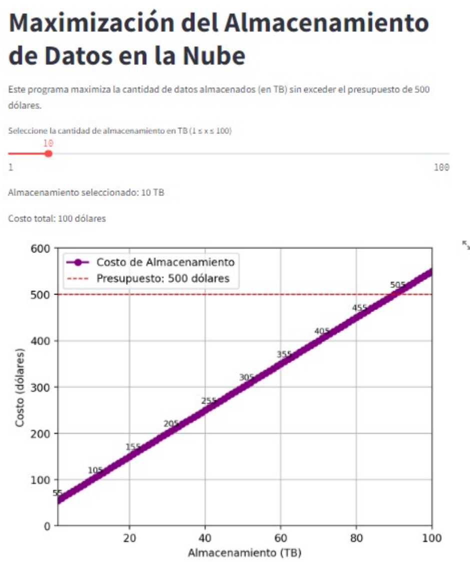

Figura 18: Gráfico

Ejercicio 10

Un sistema de mensajería tiene una latencia $L(x) = 100 - 2x$, donde x es el número de mensajes por segundo. La latencia no puede ser inferior a 20 ms debido a restricciones del protocolo. Maximice el número de mensajes enviados sin que la latencia caiga por debajo de este límite.

Solución

La función de latencia está dada por:

$$L(x) = 100 - 2x$$

donde x es el número de mensajes por segundo y $L(x)$ es la latencia en milisegundos.

La restricción es que la latencia no puede ser menor que 20 ms:

$$L(x) \geq 20$$

Despejamos x a partir de la ecuación de latencia:

$$100 - 2x \geq 20$$

Restamos 100 en ambos lados:

$$-2x \geq -80$$

Dividimos ambos lados por -2 (invirtiendo el signo de la desigualdad):

$$x \leq 40$$

Conclusión

Por lo tanto, el número máximo de mensajes que se pueden enviar sin que la latencia caiga por debajo de 20 ms es **40** mensajes por segundo.

```python
Ejercicio10.py > ...
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

def calcular_latencia(x):
    return 100 - 2 * x

limite_latencia = 20

st.title('Maximización del Número de Mensajes en un Sistema de Mensajería')

st.write('Este programa maximiza el número de mensajes enviados sin que la latencia caiga por debajo de 20 ms.')

x = st.slider('Seleccione el número de mensajes por segundo (1 ≤ x ≤ 50)', min_value=1, max_value=50, value=10)

latencia = calcular_latencia(x)

st.write(f'Número de mensajes por segundo: {x}')
st.write(f'Latencia: {latencia} ms')

if latencia < limite_latencia:
    st.warning(f'Advertencia: La latencia ha caído por debajo del límite de {limite_latencia} ms.')

mensajes_x = np.arange(1, 51)
latencia_y = [calcular_latencia(i) for i in mensajes_x]

fig, ax = plt.subplots()

ax.plot(mensajes_x, latencia_y, marker='o', linestyle='-', color='purple', label='Latencia (ms)', linewidth=2)
ax.set_xlabel('Número de Mensajes por Segundo')
ax.set_ylabel('Latencia (ms)')
ax.axhline(limite_latencia, color='red', linestyle='--', linewidth=1, label=f'Límite de {limite_latencia} ms')

ax.set_ylim(0, max(latencia_y) + 10)
ax.set_xlim(1, 50)
ax.grid(True)

for i, txt in enumerate(latencia_y):
    if i % 5 == 0:
        ax.text(mensajes_x[i], latencia_y[i] + 1, f'{txt}', fontsize=8, ha='center')

ax.legend()

st.pyplot(fig)
```

Figura 19: Código python

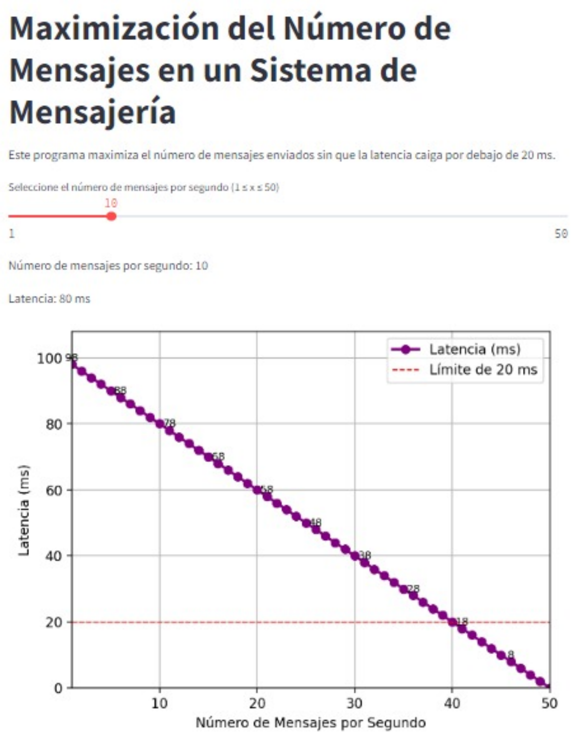

Figura 20: Gráfico