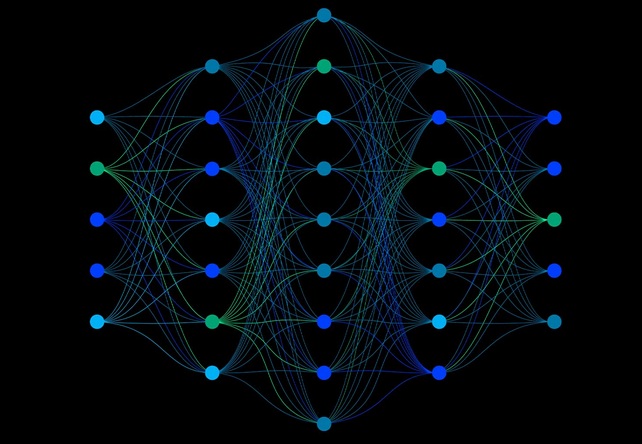
Norma y producto punto

Invalid Date

## Objetivos:







* Definir qué es una norma en álgebra lineal.
* Explicar su importancia en la teoría de vectores.
* Mostrar las propiedades principales de las normas.
* Aplicar normas en ejemplos gráficos.

## Normas inducidas por productos internos

Dado un espacio vectorial con un producto interno $ , $

tomando la norma inducida por el producto interno es:

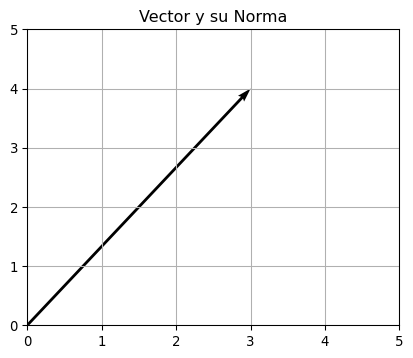
## Importancia de la Norma en Álgebra Lineal

La norma permite cuantificar el tamaño de un vector.

Es esencial en la normalización de vectores (vectores unitarios).

Se usa en diversas aplicaciones, desde análisis de datos hasta mecánica clásica.

import matplotlib.pyplot as plt  
import numpy as np  
  
vector = np.array([3, 4])  
origin = np.array([0, 0])  
  
plt.quiver(\*origin, \*vector, scale=1, scale\_units='xy', angles='xy')  
plt.xlim(0, 5)  
plt.ylim(0, 5)  
plt.axhline(0, color='black',linewidth=0.5)  
plt.axvline(0, color='black',linewidth=0.5)  
plt.title("Vector y su Norma")  
plt.grid(True)  
plt.show()



## Propiedades de las Normas (I)

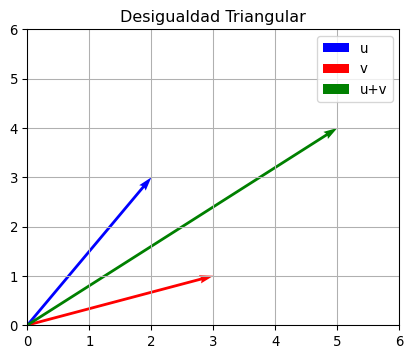
**Propiedad 1: No negatividad**

**Propiedad 2: Homogeneidad**

## Propiedades de las Normas (II)

**Propiedad 3: Desigualdad triangular**

import matplotlib.pyplot as plt  
import numpy as np  
  
vector\_u = np.array([2, 3])  
vector\_v = np.array([3, 1])  
origin = np.array([0, 0])  
  
plt.quiver(\*origin, \*vector\_u, color='b', scale=1, scale\_units='xy', angles='xy', label='u')  
plt.quiver(\*origin, \*vector\_v, color='r', scale=1, scale\_units='xy', angles='xy', label='v')  
plt.quiver(\*origin, \*(vector\_u + vector\_v), color='g', scale=1, scale\_units='xy', angles='xy', label='u+v')  
  
plt.xlim(0, 6) # Ajusta los límites de los ejes para que todos los vectores estén dentro de la vista  
plt.ylim(0, 6)  
plt.axhline(0, color='black', linewidth=0.5)  
plt.axvline(0, color='black', linewidth=0.5)  
plt.title("Desigualdad Triangular")  
plt.grid(True)  
plt.legend()  
plt.show()



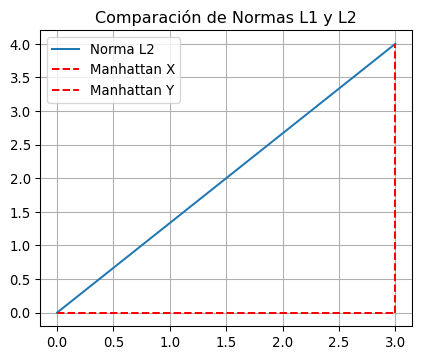
## Tipos de Normas (I)

Norma Euclidiana ( L\_2 ):

## Tipos de Normas (II)

Norma ( L\_1 ) (Manhattan):

vector = np.array([3, 4])  
  
plt.plot([0, 3], [0, 4], label="Norma L2")  
plt.plot([0, 3], [0, 0], 'r--', label="Manhattan X")  
plt.plot([3, 3], [0, 4], 'r--', label="Manhattan Y")  
plt.title("Comparación de Normas L1 y L2")  
plt.grid(True)  
plt.legend()  
plt.show()



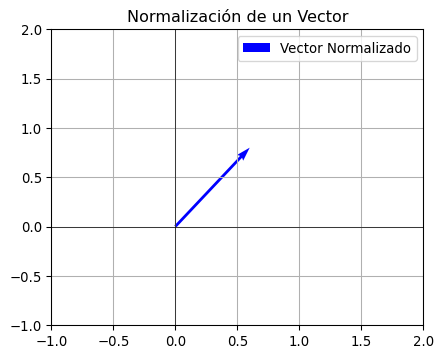
## Norma (Norma Máxima)

## Aplicaciones de las Normas

Aplicaciones comunes:

* Normalización de vectores para algoritmos de aprendizaje automático.
* Cálculo de distancias entre puntos en análisis de datos.
* Aplicaciones en física, como la mecánica y la óptica.

import matplotlib.pyplot as plt  
import numpy as np  
  
vector = np.array([3, 4])  
origin = np.array([0, 0])  
  
# Normalización del vector  
vector\_normalizado = vector / np.linalg.norm(vector)  
  
# Dibujamos el vector normalizado  
plt.quiver(\*origin, \*vector\_normalizado, color='b', scale=1, scale\_units='xy', angles='xy', label='Vector Normalizado')  
  
# Ajustamos los límites de los ejes para que todo el vector esté en la vista  
plt.xlim(-1, 2) # Los valores pueden ajustarse según sea necesario  
plt.ylim(-1, 2)  
plt.axhline(0, color='black', linewidth=0.5)  
plt.axvline(0, color='black', linewidth=0.5)  
plt.title("Normalización de un Vector")  
plt.grid(True)  
plt.legend()  
plt.show()



## Calculo de normas

### Norma (Manhattan) en Logística

**Contexto:** Un repartidor debe viajar entre dos puntos de una ciudad que sigue una disposición en cuadrícula. La ciudad solo permite moverse en dirección horizontal o vertical.

Dado un vector de desplazamiento , calcula la norma y proporciona una interpretación en términos de distancia recorrida.

**Interpretación:** La norma en este contexto es la distancia total recorrida por el repartidor, sumando los movimientos horizontales y verticales. Representa la “distancia de taxi” o la distancia efectiva en una ciudad con una cuadrícula.

## Norma (Euclidiana) en Física

**Contexto:** En un sistema físico, la norma Euclidiana se utiliza para calcular la magnitud de la velocidad de un objeto que se mueve en el espacio tridimensional.

Dado el vector de velocidad , calcula la norma y proporciona una interpretación en términos de la magnitud de la velocidad.

**Interpretación:** La norma en este contexto te dará la magnitud de la velocidad, lo que indica la rapidez con la que el objeto se está moviendo en el espacio. Esta magnitud es la velocidad total en metros por segundo.

### Norma (Máxima) en Control de Calidad

**Contexto:** En control de calidad, la norma se utiliza para detectar la desviación máxima de un conjunto de datos.

Dado el vector de errores de medición , calcula la norma y proporciona una interpretación en términos de la desviación máxima.

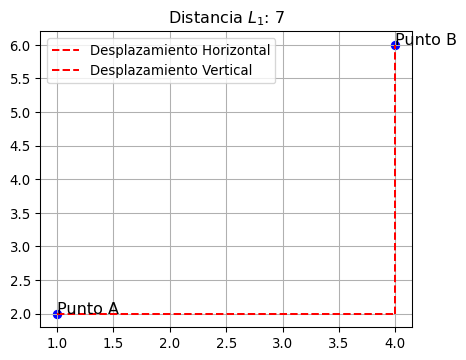
**Interpretación:** La norma te da la desviación más grande en valor absoluto, lo que indica cuál es el mayor error de medición en el conjunto de datos. Este valor es importante para evaluar el peor caso en términos de precisión.

## Distancia (Manhattan)

La distancia entre dos puntos y es:

Esta distancia sigue caminos horizontales y verticales, como en una cuadrícula.

import numpy as np  
import matplotlib.pyplot as plt  
  
# Definir puntos  
x = np.array([1, 2])  
y = np.array([4, 6])  
  
# Calcular distancia L1  
L1\_distance = np.sum(np.abs(x - y))  
  
# Gráfico del camino L1  
plt.plot([x[0], y[0]], [x[1], x[1]], 'r--', label="Desplazamiento Horizontal")  
plt.plot([y[0], y[0]], [x[1], y[1]], 'r--', label="Desplazamiento Vertical")  
plt.scatter([x[0], y[0]], [x[1], y[1]], c='b')  
plt.text(x[0], x[1], 'Punto A', fontsize=12)  
plt.text(y[0], y[1], 'Punto B', fontsize=12)  
plt.title(f"Distancia $L\_1$: {L1\_distance}")  
plt.grid(True)  
plt.legend()  
plt.show()

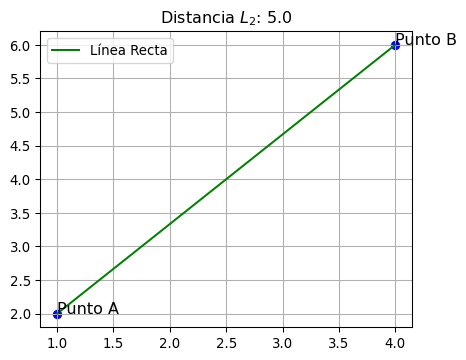


## Distancia (Euclidiana)

La distancia entre dos puntos y es:

Esta distancia representa la línea recta más corta entre los dos puntos.

# Calcular distancia L2  
L2\_distance = np.sqrt(np.sum((x - y) \*\* 2))  
  
# Gráfico del camino L2  
plt.plot([x[0], y[0]], [x[1], y[1]], 'g-', label="Línea Recta")  
plt.scatter([x[0], y[0]], [x[1], y[1]], c='b')  
plt.text(x[0], x[1], 'Punto A', fontsize=12)  
plt.text(y[0], y[1], 'Punto B', fontsize=12)  
plt.title(f"Distancia $L\_2$: {L2\_distance}")  
plt.grid(True)  
plt.legend()  
plt.show()



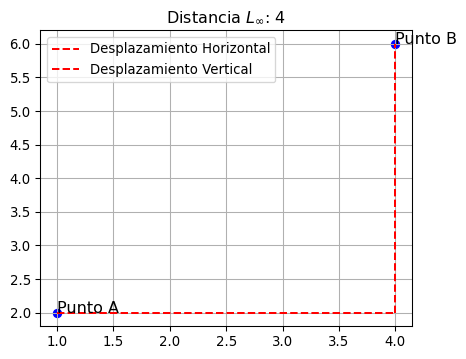
## Distancia (Máxima)

La distancia entre dos puntos y es:

Esta distancia se enfoca en el mayor desplazamiento en cualquier coordenada.

# Calcular distancia Linf  
Linf\_distance = np.max(np.abs(x - y))  
  
# Gráfico del camino Linf  
plt.plot([x[0], y[0]], [x[1], x[1]], 'r--', label="Desplazamiento Horizontal")  
plt.plot([y[0], y[0]], [x[1], y[1]], 'r--', label="Desplazamiento Vertical")  
plt.scatter([x[0], y[0]], [x[1], y[1]], c='b')  
plt.text(x[0], x[1], 'Punto A', fontsize=12)  
plt.text(y[0], y[1], 'Punto B', fontsize=12)  
plt.title(f"Distancia $L\_\infty$: {Linf\_distance}")  
plt.grid(True)  
plt.legend()  
plt.show()

<>:10: SyntaxWarning: invalid escape sequence '\i'  
<>:10: SyntaxWarning: invalid escape sequence '\i'  
/tmp/ipykernel\_40759/1327578443.py:10: SyntaxWarning: invalid escape sequence '\i'  
 plt.title(f"Distancia $L\_\infty$: {Linf\_distance}")



## Comparación de las distancias

Podemos comparar las tres distancias calculadas entre los puntos y :

* Distancia :
* Distancia :
* Distancia :

Cada norma mide la distancia de una forma diferente. La distancia suma las diferencias en las coordenadas, la distancia mide la línea recta más corta, y la distancia se enfoca en el mayor desplazamiento.

## Como te imaginas que seria una circunferencia en las diferentes distancias?

**Definicio de circunferencia** Una circunferencia es el lugar geométrico de los puntos del plano cuya distancia a otro punto fijo, llamado centro, es constante.

plt.figure(figsize=(6, 6))  
  
# Rango de ángulos para dibujar las "circunferencias"  
theta = np.linspace(0, 2 \* np.pi, 100)  
  
# Función para dibujar circunferencia bajo norma L1  
x\_L1 = np.abs(np.cos(theta)) + np.abs(np.sin(theta)) # L1 suma de valores absolutos  
x1 = np.cos(theta) / x\_L1  
y1 = np.sin(theta) / x\_L1  
  
# Función para dibujar circunferencia bajo norma L2  
x2 = np.cos(theta)  
y2 = np.sin(theta)  
  
# Dibujar un cuadrado para la norma L\_inf (Máxima)  
x3 = np.array([-1, 1, 1, -1, -1])  
y3 = np.array([-1, -1, 1, 1, -1])  
  
# Dibujar las tres circunferencias en el mismo plano cartesiano  
plt.plot(x1, y1, label='$L\_1$', color='r')  
plt.plot(x2, y2, label='$L\_2$', color='g')  
plt.plot(x3, y3, label='$L\_\infty$', color='b')  
  
# Configuraciones del gráfico  
plt.gca().set\_aspect('equal')  
plt.title('Comparación de normas $L\_1$, $L\_2$ y $L\_\infty$')  
plt.grid(True)  
plt.legend()  
plt.xlim([-1.5, 1.5])  
plt.ylim([-1.5, 1.5])  
  
# Mostrar la figura  
plt.show()

<>:22: SyntaxWarning: invalid escape sequence '\i'  
<>:26: SyntaxWarning: invalid escape sequence '\i'  
<>:22: SyntaxWarning: invalid escape sequence '\i'  
<>:26: SyntaxWarning: invalid escape sequence '\i'  
/tmp/ipykernel\_40759/1189076720.py:22: SyntaxWarning: invalid escape sequence '\i'  
 plt.plot(x3, y3, label='$L\_\infty$', color='b')  
/tmp/ipykernel\_40759/1189076720.py:26: SyntaxWarning: invalid escape sequence '\i'  
 plt.title('Comparación de normas $L\_1$, $L\_2$ y $L\_\infty$')

