Universidad César Vallejo

APLICACIÓN DE VECTORES EN DIFERENTES DIMENSIONES

OPERACIONES DE MACHINE LEARNING

INTEGRANTES:

Huapaya Cespedes Mark

Ruiz Delgado Joseph Aribel

Mendoza Cruz Alex

Castillo Villanueva Orlando Stephen

Instructor: ODON ARESTEGUI SIERRA

Título del proyecto:

Aplicación de Vectores en diferentes dimensiones en operaciones de Machine Learning.

Objetivo general:

Explorar y analizar el papel de las operaciones vectoriales en una, dos y tres dimensiones como base para algoritmos de aprendizaje automático, demostrando su importancia en la representación y manipulación de datos.

Objetivos específicos:

- Describir los conceptos fundamentales de vectores en 1D, 2D y 3D.
- Analizar las operaciones vectoriales básicas y avanzadas que se utilizan en Machine Learning.
- Demostrar mediante ejemplos prácticos cómo estas operaciones se aplican en algoritmos de aprendizaje automático como regresión lineal, redes neuronales y clustering.
- Implementar un experimento sencillo que evidencie la importancia de las operaciones vectoriales en un modelo de Machine Learning.

Resumen/Justificación:

El aprendizaje automático se fundamenta en operaciones sobre conjuntos de datos que, a nivel matemático, son representaciones vectoriales. Comprender a profundidad las operaciones sobre vectores en distintas dimensiones permite optimizar modelos, entender su funcionamiento interno y mejorar la interpretación de los resultados. Este proyecto busca reforzar esa conexión entre matemáticas básicas y tecnologías avanzadas.

Marco Teórico:

- **Introducción**

  Los vectores son elementos matemáticos fundamentales en Machine Learning (ML), caracterizados por su magnitud y dirección. Permiten representar datos en espacios multidimensionales y son esenciales para operaciones algebraicas que sustentan algoritmos de ML. Su versatilidad abarca desde modelos básicos hasta arquitecturas complejas de deep learning.

- **Representación de Datos**

  En ML, los datos se codifican como vectores numéricos donde cada dimensión representa una característica. Por ejemplo:

  • Imágenes se transforman en vectores al aplanar sus píxeles
  • Textos se representan mediante vectores de frecuencia o embeddings
  • Datos tabulares se organizan como vectores de características

  Los modelos también utilizan vectores internamente, como los pesos en regresión lineal o las activaciones en redes neuronales, que se ajustan durante el entrenamiento.

- **Operaciones Clave**

  Las operaciones vectoriales más importantes en ML incluyen:

  El producto escalar, que calcula la similitud entre vectores y es fundamental para algoritmos como regresión lineal y sistemas de recomendación. Mide qué tan alineados están dos vectores en el espacio.

  La norma euclídea determina la magnitud de un vector y se usa en clustering (k-means) para agrupar datos similares basados en su proximidad geométrica. También es crucial para normalizar datos.

  El producto matricial permite transformaciones lineales entre capas en redes neuronales. Cada capa aplica una combinación lineal de los vectores de entrada seguida de funciones de activación no lineales.

- **Aplicaciones en Algoritmos**

  En regresión lineal, los vectores de pesos se optimizan para minimizar el error en las

  predicciones. Geométricamente, esto equivale a encontrar el hiperplano que mejor ajusta los datos.

  Los algoritmos de clustering como k-means utilizan distancias entre vectores para agrupar datos similares. Esta técnica se aplica en segmentación de clientes o análisis de imágenes.

  Las redes neuronales procesan información mediante transformaciones vectoriales sucesivas, permitiendo el reconocimiento de patrones complejos en datos como imágenes, audio o texto.

- **Ventajas Computacionales**

  Las operaciones vectoriales ofrecen tres ventajas principales:

  - Eficiencia al procesarse en paralelo en GPUs/TPUs
  - Escalabilidad para manejar grandes volúmenes de datos
  - Capacidad de visualización en 2D/3D para interpretar resultados

- **Casos de Estudio**

  NVIDIA DLSS: Utiliza tensores (vectores multidimensionales) y núcleos especializados (Tensor Cores) en sus GPUs para aplicar redes neuronales profundas. Analiza múltiples fotogramas anteriores mediante operaciones vectoriales masivas, reconstruyendo imágenes nítidas a mayor resolución. Esto permite mejorar el rendimiento manteniendo calidad visual en juegos.

  AMD FSR: Al no contar con núcleos tensoriales dedicados, emplea algoritmos de escalado espacial convencionales optimizados con operaciones vectoriales matriciales. Usa análisis de bordes y patrones geométricos mediante vectores 2D/3D para mejorar la resolución, ofreciendo una solución compatible con múltiples hardware.

  NVIDIA RTX: usa núcleos especializados para acelerar cálculos vectoriales de trazado de rayos, simulando cómo la luz interactúa con objetos en 3D. Mediante vectores, calcula: 1) trayectorias de rayos, 2) reflejos, y 3) sombras en tiempo real, creando iluminación realista en juegos.

Metodología:

1. **Revisión bibliográfica** sobre vectores, álgebra lineal y Machine Learning.
2. **Desarrollo teórico**: descripción y ejemplificación de operaciones vectoriales en diferentes dimensiones.
3. **Implementación computacional** (Python, usando NumPy, Scikit-learn o PyTorch):
    - Visualización de vectores en 2D y 3D.
    - Aplicación de operaciones vectoriales en un modelo sencillo de regresión o clasificación.
4. **Análisis de resultados**: interpretación del impacto de operaciones vectoriales en el rendimiento y funcionamiento del modelo.
5. **Conclusiones y recomendaciones.**

Resultados esperados:

- Comprensión clara de la importancia de las operaciones vectoriales en Machine Learning.
- Demostración experimental que relacione operaciones vectoriales con el comportamiento de un modelo ML.
- Visualizaciones que refuercen la interpretación geométrica de operaciones en 2D y 3D.

Herramientas sugeridas:

- **Python (NumPy, Matplotlib, Scikit-learn, PyTorch)**
- **GeoGebra 3D** (para visualización geométrica)
- **Jupyter Notebook** (para documentación y experimentación reproducible)

Ejemplos:

1. Vectores en 1D, 2D y 3D (Parte Física):

- Explica qué es un vector: módulo, dirección y sentido.
- Representa vectores en 1D: por ejemplo, velocidad en línea recta.
- Vectores en 2D y 3D: posición, fuerza, aceleración.
- Usa Python para representar y sumar vectores:

python

```
import numpy as np

import matplotlib.pyplot as plt


# Ejemplo de vectores en 2D

A = np.array([3, 2])

B = np.array([1, 4])

C = A + B

# Visualización

plt.quiver(0, 0, A[0], A[1], angles='xy', scale_units='xy', scale=1, color='r', label='A')

plt.quiver(0, 0, B[0], B[1], angles='xy', scale_units='xy', scale=1, color='b', label='B')

plt.quiver(0, 0, C[0], C[1], angles='xy', scale_units='xy', scale=1, color='g', label='A + B')

plt.xlim(0, 6)

plt.ylim(0, 6)

plt.grid()

plt.legend()

plt.title("Suma de vectores en 2D")

plt.show()
```

---

2. Aplicación en Machine Learning:

- Perceptrón: usa vectores para representar los pesos y entradas.
- Muestra cómo el algoritmo calcula un producto punto entre vectores:

python

```
from sklearn.linear_model import Perceptron

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt


# Generar datos 2D

X, y = make_classification(n_samples=100, n_features=2, n_redundant=0,
n_informative=2, n_clusters_per_class=1, n_classes=2, random_state=42)


# Entrenar perceptrón

model = Perceptron()

model.fit(X, y)


# Visualizar datos y vector de pesos

w = model.coef_[0]

b = model.intercept_[0]


plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm')

x_line = np.linspace(min(X[:,0]), max(X[:,0]), 100)

y_line = -(w[0]*x_line + b)/w[1]

plt.plot(x_line, y_line, 'k--', label='Frontera de decisión')

plt.legend()

plt.title('Separación de clases usando producto escalar (Perceptrón)')
```

```
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```

Aplicacion plus en 3D

Visualizar vectores en *3D* usando plotly o matplotlib:

```
python
from mpl_toolkits.mplot3d import Axes3D


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Vectores 3D
A = [1, 2, 3]
B = [2, 1, -1]
C = np.add(A, B)

ax.quiver(0, 0, 0, *A, color='r', label='A')
ax.quiver(0, 0, 0, *B, color='b', label='B')
ax.quiver(0, 0, 0, *C, color='g', label='A + B')
ax.set_xlim([0, 4])
ax.set_ylim([0, 4])
ax.set_zlim([0, 4])
ax.set_title("Vectores en 3D")
ax.legend()
plt.show()
```

Conclusión:

- Los vectores son esenciales tanto en física como en ML.
- El producto escalar entre vectores es la base del perceptrón y otros algoritmos.
- Comprender los conceptos físicos ayuda a entender la geometría y funcionamiento de muchos modelos.

Bibliografía:

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Strang, G. (2016). *Introduction to Linear Algebra*. Wellesley-Cambridge Press.
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.