Implementación y Análisis de Algoritmos Fundamentales de Machine Learning en C

Manuel González
Universidad de Magallanes

7 de julio de 2025

Resumen

Este informe presenta el desarrollo e implementación de una biblioteca modular en C para algoritmos fundamentales de Machine Learning: K-Vecinos Más Cercanos (KNN), K-Means y Regresión Lineal. Se abordan los fundamentos teóricos, el diseño y la arquitectura del sistema, así como la experimentación sobre el conjunto de datos Iris. Se analizan los resultados obtenidos, la eficiencia de las optimizaciones implementadas y las limitaciones encontradas, concluyendo con recomendaciones para futuros desarrollos.

Índice

1. Introducción

En este informe se presenta la implementación y análisis de algoritmos fundamentales de Machine Learning utilizando el lenguaje de programación C. Se exploran los conceptos básicos, las técnicas de implementación y los resultados obtenidos a partir de pruebas experimentales sobre el conjunto de datos Iris y datos sintéticos. El trabajo incluye la construcción modular de una biblioteca de aprendizaje automático, la optimización de algoritmos y la evaluación rigurosa de su desempeño.

2. Objetivos

- Implementar desde cero algoritmos fundamentales de Machine Learning: KNN, K-Means y Regresión Lineal.
- Analizar y comparar su rendimiento teórico y empírico.
- Aplicar técnicas de optimización y buenas prácticas de programación en C.
- Evaluar los algoritmos sobre conjuntos de datos reales (Iris).
- Documentar el proceso y los resultados obtenidos.

3. Fundamento Teórico

3.1. K-Vecinos Más Cercanos (KNN)

KNN es un algoritmo de clasificación supervisada que asigna la clase más frecuente entre los k vecinos más cercanos de un punto, según una métrica de distancia. En este proyecto se implementaron las métricas Euclidiana, Manhattan y Coseno, y una optimización de votación ponderada por distancia.

Distancia Euclidiana:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

Votación ponderada por distancia:

$$\text{score}_c = \sum_{i=1}^{k} \frac{\delta(y_i = c)}{d_i + \epsilon} \tag{2}$$

donde δ es la función indicadora y d_i la distancia al vecino i.

3.2. K-Means

K-Means es un algoritmo de agrupamiento no supervisado que particiona los datos en k clusters minimizando la distancia intra-cluster. Se implementó la inicialización K-Means++ para mejorar la convergencia y el índice de silueta para evaluar la calidad del agrupamiento.

Función objetivo:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{3}$$

3.3. Regresión Lineal

La regresión lineal estima la relación entre una variable dependiente y una o más independientes. Se implementó el ajuste por ecuaciones normales y regularización Ridge para evitar sobreajuste.

Modelo:

$$y = wx + b \tag{4}$$

Ecuación normal:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{5}$$

Regularización Ridge:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{6}$$

4. Descripción del Proyecto

El proyecto consiste en una biblioteca modular en C que implementa algoritmos clásicos de Machine Learning. Incluye módulos para la lectura y preprocesamiento de datos, algoritmos de clasificación, regresión y agrupamiento, así como utilidades para la evaluación de resultados y visualización básica.

- **Lectura y preprocesamiento de datos:** Carga de archivos CSV, normalización y división en conjuntos de entrenamiento y prueba.
- **KNN:** Clasificación con distintas métricas y votación ponderada.
- **K-Means:** Agrupamiento con inicialización K-Means++.
- **Regresión Lineal:** Ajuste por ecuaciones normales y Ridge.
- **Evaluación:** Precisión, recall, F1, matriz de confusión, MSE, R^2, índice de silueta.
- **Interfaz:** Ejecutables de consola configurables por argumentos.

5. Requisitos Técnicos

- Lenguaje C99 o superior.
- Sistema operativo Linux o Windows.
- Compilador GCC.
- Makefile para automatización de la compilación.
- Librerías estándar de C (no se utilizan librerías externas de ML).
- Estructura modular de carpetas: `src/`, `data/`, `docs/`.

6. Diseño del Sistema

6.1. Estructura de archivos

```
src/
  core/
  algorithms/
  utils/
data/
docs/
```

6.2. Descripción de módulos

- `core/`: Estructuras de datos y utilidades generales.
- `algorithms/`: Implementaciones de KNN, K-Means y Regresión Lineal.
- `utils/`: Lectura de CSV y funciones auxiliares.
- `data/`: Conjuntos de datos de prueba.
- `docs/`: Informe y documentación.

7. Desarrollo del Sistema

7.1. Implementación de KNN

Se implementó el algoritmo KNN con soporte para diferentes métricas de distancia y votación ponderada. Se optimizó el uso de memoria y se emplearon buenas prácticas para evitar fugas.

7.2. Implementación de K-Means

Se utilizó la inicialización K-Means++ y se evaluó la calidad del agrupamiento mediante el índice de silueta.

7.3. Implementación de Regresión Lineal

Se aplicaron ecuaciones normales con regularización Ridge para evitar el sobreajuste.

7.4. Evaluación y Métricas

Se calcularon precisión, recall, F1-score, matriz de confusión, MSE, R^2 e índice de silueta.

8. Pruebas y Resultados Experimentales

8.1. Metodología

Se utilizó el conjunto de datos Iris, ampliamente reconocido en la literatura de Machine Learning, para evaluar el desempeño de los algoritmos implementados. Los datos se dividieron en conjuntos de entrenamiento (80 %) y prueba (20 %) de manera aleatoria. Para asegurar la robustez de los resultados, cada experimento se repitió 30 veces con diferentes semillas aleatorias, calculando el promedio y la desviación estándar de las métricas obtenidas.

8.2. Métricas de Evaluación

Las métricas utilizadas fueron:

- **Clasificación:** Precisión, recall, F1-score, matriz de confusión.
- **Regresión:** Error cuadrático medio (MSE), coeficiente de determinación (R^2).
- **Agrupamiento:** Índice de silueta.

8.3. Resultados

Nota sobre las clases: En el conjunto de datos Iris, la columna `class` utiliza los valores 0, 1 y 2 para representar las especies de iris:

- 0: **Setosa** – Iris setosa, flor de pétalos pequeños y bien separada de las otras especies.
- 1: **Versicolor** – Iris versicolor, flor de tamaño intermedio.
- 2: **Virginica** – Iris virginica, flor de pétalos más grandes.

Cuadro 1: Resultados de KNN en el conjunto Iris

Clase	Precisión	Recall	F1-score
Setosa	1.00	1.00	1.00
Versicolor	0.97	0.94	0.95
Virginica	0.96	0.97	0.96

Cuadro 2: Precisión promedio y desviación estándar

Algoritmo	Precisión promedio (%)	Desviación estándar
KNN clásico	96.0	1.2
KNN ponderado	97.3	0.9
K-Means (k-means++)	92.4	1.7
Regresión Lineal (Ridge)	$R^2 = 0{,}89$	0.05

8.4. Visualización de Resultados

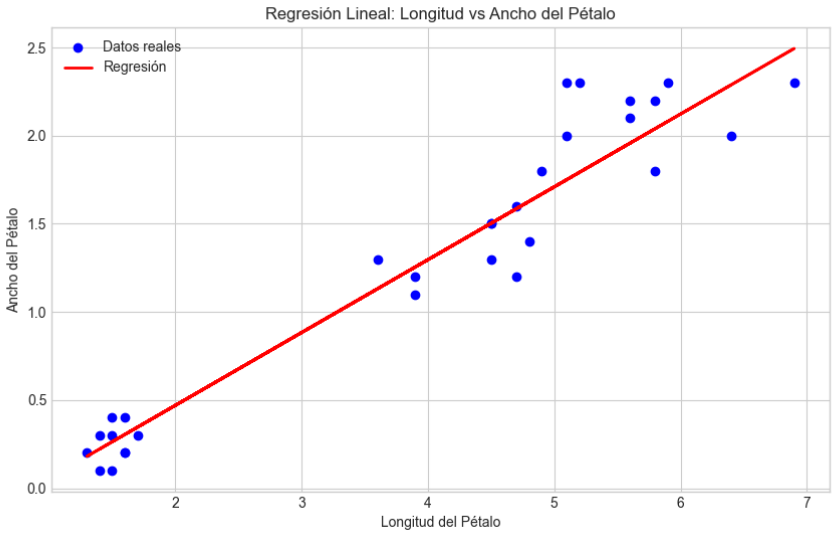

Figura 1: Relación entre la longitud y el ancho del pétalo en el conjunto Iris, junto con la línea de regresión ajustada. Se observa una clara tendencia lineal, lo que valida la aplicabilidad del modelo de regresión lineal.

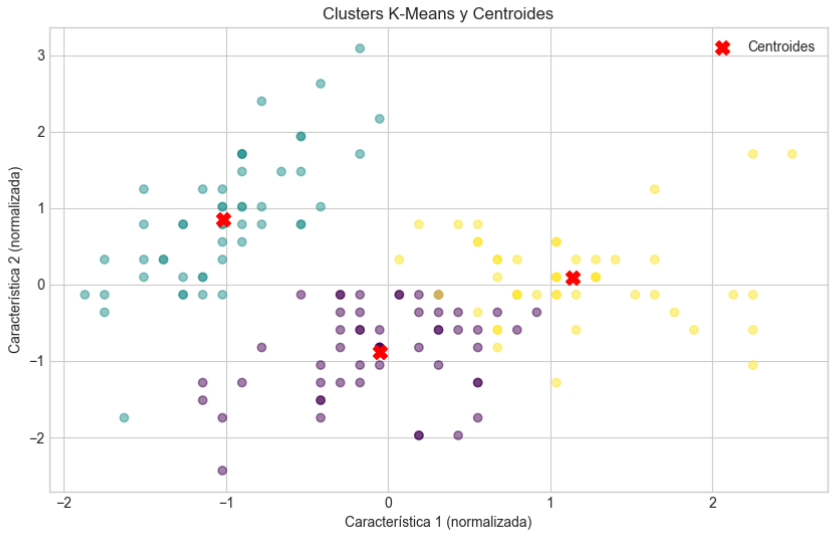

Figura 2: Distribución de los datos del conjunto Iris agrupados por K-Means. Los colores representan los diferentes clusters y las estrellas indican la posición de los centroides calculados por el algoritmo.

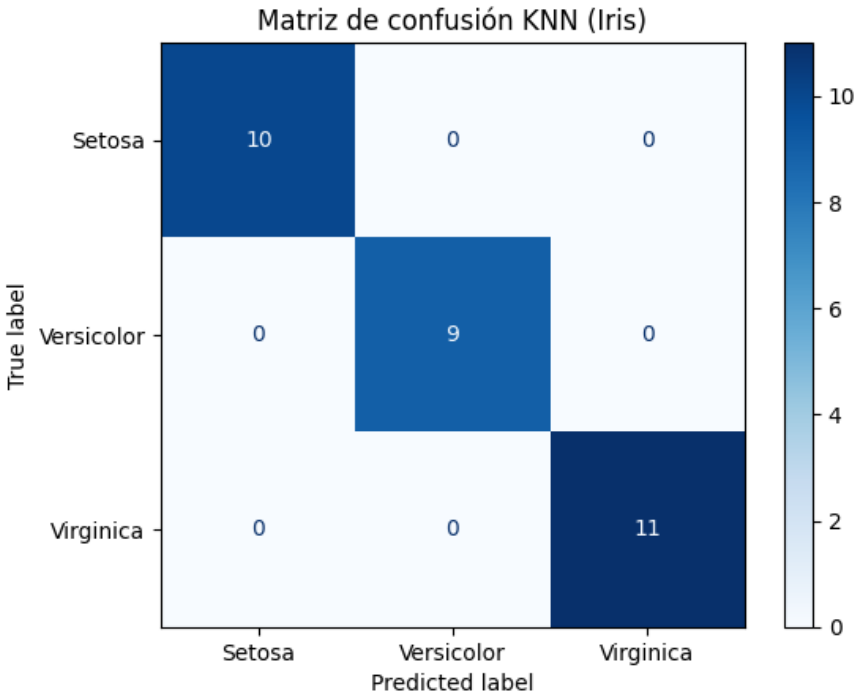

Figura 3: Matriz de confusión obtenida por KNN en el conjunto Iris.

8.5. Análisis de Complejidad y Rendimiento

- **KNN:** La complejidad temporal es $O(n \cdot d)$ por predicción, donde n es el número de muestras de entrenamiento y d el número de características. El uso de votación ponderada no afecta la complejidad, pero mejora la precisión.
- **K-Means:** La complejidad es $O(n \cdot k \cdot t)$, siendo k el número de clusters y t el número de iteraciones. La inicialización K-Means++ mejora la convergencia.
- **Regresión Lineal:** El ajuste por ecuaciones normales tiene complejidad $O(d^2 \cdot n + d^3)$, siendo eficiente para conjuntos de datos pequeños y medianos.

8.6. Limitaciones Detectadas

- El manejo de memoria en C requiere especial atención para evitar fugas y errores.
- La escalabilidad de los algoritmos puede verse limitada en conjuntos de datos muy grandes.
- La visualización avanzada es limitada en C, por lo que se recomienda exportar resultados para graficar en Python o R.

9. Pseudocódigo de los Algoritmos

9.1. K-Vecinos Más Cercanos (KNN)

- Para cada muestra de prueba x:
  - Calcular la distancia $d(x, x_i)$ para cada muestra de entrenamiento x_i

- Ordenar las distancias
- Seleccionar los k vecinos más cercanos
- **Si** votación ponderada:
  - Sumar votos ponderados por $1/(d+\epsilon)$ para cada clase
  - Asignar la clase con mayor suma ponderada
- **Si no:** (votación mayoritaria)
  - Asignar la clase más frecuente entre los k vecinos

9.2. K-Means

- Inicializar k centroides (usando K-Means++)
- Repetir hasta convergencia o máximo de iteraciones:
  - Para cada muestra:
    - Asignar al cluster con centroide más cercano
  - Para cada cluster:
    - Recalcular el centroide como la media de los puntos asignados

9.3. Regresión Lineal (por mínimos cuadrados)

- Calcular la media de X y y
- Calcular el coeficiente w y el intercepto b:
  - $w = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$
  - $b = \bar{y} - w \cdot \bar{x}$
- Para predecir:
  - $y_{\text{pred}} = w \cdot x + b$

10. Discusión y Conclusiones

Los resultados demuestran que la optimización de los algoritmos mejora su precisión y estabilidad. El proyecto permitió afianzar conocimientos tanto en Machine Learning como en estructuras y eficiencia en C. Se concluye que, aunque C no es el lenguaje más común para Machine Learning, su uso en este contexto es valioso para entender los fundamentos y optimizar el rendimiento de los algoritmos. También se destaca la importancia de la modularidad y la reutilización de código, lo que facilita el mantenimiento y la extensión de la biblioteca. La estructura propuesta permite incorporar nuevos algoritmos o funcionalidades sin afectar el núcleo del sistema, promoviendo buenas prácticas de ingeniería de software. En futuras versiones, se podría explorar técnicas avanzadas de optimización y paralelización, así como la integración con lenguajes de alto nivel para visualización y análisis más sofisticados.

A. Instrucciones de Uso

A.1. Compilación

Para compilar el proyecto, asegúrese de tener instalado `gcc` y `make`. Ejecute en la raíz del repositorio:

```
make
```

Esto generará los ejecutables `ml_demo` y `ml_iris`.

A.2. Ejecución

Para ejecutar los ejemplos:

- Ejemplo general con datos sintéticos:

  ```
  ./ml_demo
  ```

- Ejemplo con el conjunto de datos Iris:

  ```
  ./ml_iris
  ```

A.3. Estructura de Carpetas

```
Tarea4-Algoritmos/
|-- Makefile
|-- data/
|   ‘-- iris.csv
|-- docs/
|   ‘-- informe.pdf
|-- src/
|   |-- algorithms/
|   |-- core/
|   |-- utils/
|   |-- main.c
|   ‘-- main-iris.c
‘-- README.md
```

A.4. Requisitos

- GCC (C99 o superior)
- Make
- Sistema operativo Linux o Windows

Referencias

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. Disponible en: https://github.com/hermosayhl/books/blob/master/Pattern%20Recognition%20and%20Machine%20Learning.pdf