Proyecto final Análisis y Diseño de Algoritmos 1

Título

- Entrenamiento y optimización de un perceptrón multicapa (MLP) “desde cero” con análisis algorítmico y estructuras de datos

Problema a resolver

- Clasificar imágenes/tablas simples (p. ej., MNIST/Fashion-MNIST o un dataset tabular UCI) implementando y analizando el algoritmo de entrenamiento de una red neuronal feed-forward sin usar frameworks de alto nivel, enfocándose en complejidad temporal/espacial y en el uso eficiente de estructuras de datos.

Objetivo general

- Diseñar, implementar y analizar el entrenamiento de una red neuronal (forward/backprop/SGD) y módulos auxiliares, aplicando notación asintótica, técnicas clásicas (divide y vencerás, recurrencias), y estructuras de datos para construir una solución eficiente.

Metodología y fases sugeridas

Fase 1 (RA1)

- Definición del problema y dataset.
- MLP mínimo (1 capa oculta) y derivación de complejidad temporal/espacial.
- Experimentos de escalado con B, h, E. Entregable: informe RA1.

Fase 2 (RA2)

- Módulos de selección/ordenamiento: mediana (Quickselect), top-k (heap vs. sort), hard mining.
- Análisis con recursiones y Método Maestro; validación empírica. Entregable: informe RA2.

Fase 3 (RA3)

- Integración de estructuras: cola de lotes, hash de pérdidas, heap/bst para poda.
- Evaluación de impacto en tiempo total de época y memoria. Entregable: informe RA3.

Fase 4

- Baseline k-NN y/o Heapsort/Quicksort para tareas auxiliares; comparación final.
- Documentación y presentación.
- Criterios de evaluación y métricas
- Correctitud: precisión en test (p. ej. >85% MNIST reducido) y verificación de gradientes (check numérico).
- Complejidad: derivaciones Big-O correctas y consistentes con mediciones.
- Estructuras de datos: justificación y mejora observable (p. ej., top-k con heap reduce tiempo vs sort en n grande).
- Análisis de recurrences: aplicación adecuada de Método Maestro y comparación teórico/empírico.
- Calidad experimental: gráficas tiempo vs n, B, h; memoria vs. parámetros; trade-offs de poda.

Restricciones y lineamientos

- Implementar el núcleo sin frameworks de alto nivel (permitido NumPy o matrices propias; si es Java, usar arreglos y librerías estándar).
- Código claro, modular, con pruebas unitarias y generación de datos sintéticos para stress tests.

Entregables finales

- Código fuente y scripts de experimentos.
- Informe técnico con:
- Derivaciones RA1–RA3, recurrences y Big-O.

- Comparativas de estructuras (heap/hash/árbol).
- Decisiones de diseño justificadas por datos.