CS2023 - Programación III

Proyecto: **Neural Network - Aplication**

Rubén Rivas Medina

Setiembre 2025

Índice

1 Descripción general

Desarrollar de una red neuronal y Aplicacion a un caso práctico usando C++.

2 Epic 1: Biblioteca Genérica de Álgebra

Objetivo: Implementar **Tensor<T, Rank>** en C++20 para arrays multidimensionales estilo NumPy con un único método de acceso variádico.

3 Epic 2: Red Neuronal Full

Objetivo: Crear un framework en C++20 que implemente una red neuronal end-to-end usando únicamente para todos los cálculos de **forward** y **backward**, integración de funciones de pérdida y optimización por gradiente descendente.

4 Epic 3: Aplicación de la Red Neuronal y Documentación

Objetivo. Aplicar el modelo de red neuronal desarrollado en el proyecto dentro de entornos prácticos, de manera que se valide su utilidad y se explore su capacidad de generalización. El trabajo práctico se dividirá en tres ejes:

1. Integrar la red neuronal entrenada en un entorno de prueba simplificado (EnvGym) para recibir entradas y producir salidas de decisión.
2. Desarrollar un pipeline experimental de entrenamiento y ajuste fino (p. ej., hill-climbing, búsqueda aleatoria, o gradiente estocástico simple).
3. Serializar, cargar y documentar el modelo, asegurando su portabilidad para posteriores pruebas y comparaciones.

4.1 Ejemplos de Aplicación:

- **Clasificación de patrones:** usar la red para reconocer formas básicas (círculos, cuadrados, triángulos) en entradas vectoriales simuladas.
- **Predicción de series:** entrenar la red con datos sintéticos de una secuencia numérica y evaluar su capacidad de predecir el siguiente valor.
- **Control simplificado:** recibir como entrada un estado reducido de un sistema (posición, velocidad) y decidir una acción apropiada (mover izquierda/derecha, acelerar/detener).
- **Generalización y robustez:** probar el modelo serializado en condiciones distintas a las de entrenamiento, observando cómo responde a perturbaciones o entradas ruidosas.

5 Organización y Estructura del proyecto

```
pong_ai/
+-- include/              # cabeceras principales
|   +-- utec/
|       +-- algebra/      # álgebra: implementa Tensor<T,Rank>
|       |   +-- Tensor.h
|       +-- nn/           # red neuronal: define capas y NeuralNetwork
|       |   +-- nn_interfaces.h
|       |   +-- nn_dense.h
|       |   +-- nn_activation.h
|       |   +-- nn_loss.h
|       |   +-- nn_optimizer.h
|       |   +-- neural_network.h
|       +-- apps/         # aplicaciones prácticas de la red
|           +-- PatternClassifier.h
|           +-- SequencePredictor.h
|           +-- ControllerDemo.h
+-- src/                  # archivos fuentes
|   +-- utec/
|       +-- apps/
|           +-- PatternClassifier.cpp
|           +-- SequencePredictor.cpp
|           +-- ControllerDemo.cpp
+-- tests/                # casos de prueba automatizados
|   +-- test_tensor.cpp
|   +-- test_neural_network.cpp
|   +-- test_applications.cpp
+-- docs/                 # documentación y bibliografía
    +-- README.md
    +-- BIBLIOGRAFIA.md
```

6 Reglas de formación de grupos

- Cada grupo de 4 a 5 alumnos (mínimo 4).
- Todos los alumnos individual y obligatoriamente deben desarrollar los Epics #1 y #2.
- El Epic #3 se debe realizar de forma grupal; el grupo seleccionara la mejor implementación y la integra con los mejores Epics #1/#2 del grupo.

7 Evaluacion

- 8 ptos por el promedio del Epic #1 y Epic #2.
- 12 ptos por el Epic #3.