Lab 5: Herència en C++

Alexis Tarifa Pérez U232817 i Arnau Carbonell Carrasco U214562

1 Introducció

L'objectiu principal d'aquest laboratori és implementar algorismes de **Regressió Lineal** utilitzant el paradigma de la Programació Orientada a Objectes (POO) en C++. Se'ns ha proporcionat una estructura base amb classes per gestionar dades (**Dataset**, **Record**) i operacions matemàtiques (**Vector**, **Model**).

El problema consisteix a dissenyar i implementar el çervell"de l'aprenentatge: els algorismes que ajustaran el model matemàtic per minimitzar l'error en les prediccions.

Per fer-ho, hem definit i implementat les següents classes:

- **Algorithm (Classe Abstracta):** Defineix l'estructura comuna per a qualsevol algorisme d'aprenentatge, contenint principalment la taxa d'aprenentatge (learning rate).
- **GradientDescent (Descens del Gradient):** Classe concreta que hereta d'**Algorithm**. Implementa el mètode d'aprenentatge iteratiu utilitzant totes les dades del dataset i un criteri d'aturada (stoppingCriterion).
- **StochasticGradientDescent (SGD):** Classe concreta que hereta d'**Algorithm**. Implementa una versió estocàstica que utilitza mostres aleatòries (batch) i un nombre fix d'iteracions.
- **SupervisedLearner:** La classe gestora que connecta un **Dataset** amb un **Algorithm** per entrenar un **Model**.

2 Descripció de la Solució i Decisions de Disseny

Durant el desenvolupament, ens hem trobat amb situacions on el disseny proposat en el diagrama UML entrava en conflicte amb la lògica de programació o amb el codi base proporcionat. A continuació, es descriuen les decisions preses:

2.1 Gestió del Constructor a la Classe Algorithm

Es van discutir dues alternatives per implementar el constructor de la classe pare:

- **Alternativa A (Estricta al diagrama):** El diagrama suggeria un constructor **Algorithm(lr, sc)**, obligant a passar un Çriteri d'Aturada"(sc) a la classe pare.
- **Alternativa B (Lògica):** Eliminar **sc** del pare, ja que l'algorisme **StochasticGradientDescent** no utilitza un criteri d'aturada (utilitza un nombre fix d'iteracions).

Solució Escollida: Vam optar per l'**Alternativa B**. Vam modificar **Algorithm** perquè el seu constructor només accepti **learningRate**. Vam deixar el **stoppingCriterion** com un atribut exclusiu de la classe **GradientDescent**. Això respecta millor el principi d'encapsulament i evita tenir variables sense ús a la classe pare.

2.2 Accés als Paràmetres del Model

El codi base proporcionat de la classe **Model** no incloïa el mètode **getParams()**, tot i aparèixer referenciat al diagrama UML.

Solució: Vam implementar el mètode **getParams()** a **Model.h**. Això era necessari perquè els algorismes poguessin conèixer la dimensió dels vectors de pesos i crear els gradients inicials correctament.

2.3 Conceptes de POO Aplicats

- **Herència:** Las classes **GradientDescent** i **StochasticGradientDescent** hereten atributs i comportaments base de la classe **Algorithm**, evitant la duplicació de codi.
- **Polimorfisme:** La classe **SupervisedLearner** utilitza un punter de tipus **Algorithm***. Això permet que el sistema pugui funcionar indistintament amb qualsevol dels dos algorismes sense necessitat de modificar el codi del gestor.
- **Abstracció:** La classe **Algorithm** es va definir com a abstracta mitjançant el mètode virtual pur **virtual Model solve(...) = 0**. Això impedeix crear instàncies d'un algorisme genèric incomplet.

3 Conclusió

La solució implementada ha funcionat correctament a la pràctica.

Per verificar-ho, vam actualitzar l'arxiu **TestLearner.cpp** creant un dataset sintètic simple que representa la funció lineal $y = 2x$.

- En entrenar amb **Gradient Descent**, l'error va disminuir fins a complir el criteri d'aturada, aconseguint prediccions precises (per a una entrada de 5.0, la predicció va ser molt propera a 10.0).
- En entrenar amb **Stochastic Gradient Descent**, el model també va convergir correctament utilitzant lots aleatoris (batches).

Dificultats trobades: Una dificultat tècnica va ser la gestió de les dimensions dels vectors. Inicialment, les operacions fallaven perquè el model inclou un terme independent (bias). Això es va solucionar comprenent i aplicant el mètode **augment()** als vectors d'entrada abans de realitzar prediccions o càlculs d'error.

Així mateix, es van solucionar problemes de compilació a l'entorn local assegurant la correcta ubicació dels fitxers font i capçaleres.