Lab 4: Stochastic Gradient Descent y Datasets Estandarizados

1. Introducción

The objective of this laboratory was to extend the linear-regression framework from previous labs by introducing standardized datasets and implementing Stochastic Gradient Descent (SGD). To support these features, the dataset hierarchy was redesigned using an abstract class Dataset, from which two concrete classes inherit: RawDataset, representing unmodified data, and StandardizedDataset, which stores mean and standard deviation values and applies the corresponding transformations.

The algorithm hierarchy was also updated. An abstract Algorithm class now represents the common structure of all optimization methods. Two algorithms were implemented: GradientDescent, which computes the full gradient, and StochasticGradientDescent, which computes approximate gradients using random minibatches. Finally, the class SupervisedLearner was adapted so its predict method works correctly with transformed data by applying both forward and reverse transformations.

2. Descripción de la Solución

A significant part of the solution involved refactoring the dataset design. In previous labs, StandardizedDataset inherited directly from a concrete dataset class, which forced it to include methods it did not need, such as mean and standard deviation computation. In this lab, an abstract class Dataset was introduced to provide a cleaner and more modular hierarchy. This ensures that both raw and standardized datasets only implement the functionality relevant to their purpose, following principles of abstraction, encapsulation, and the Liskov substitution principle.

The SupervisedLearner class required changes to support this new structure. Before making any prediction, the learner now transforms the input using the dataset's transform method, then passes the transformed vector to the model, and finally applies the dataset's output method to translate the prediction back to the original space. This approach preserves consistency between training and prediction and clearly separates responsibilities.

Regarding the optimization algorithm, the implementation of Stochastic Gradient Descent was not a design choice but a mandatory part of the assignment. The focus was therefore on integrating SGD correctly into the existing framework. The algorithm computes an approximate gradient by selecting a random minibatch of records in each iteration. To do this efficiently, the Java Random API was used to generate distinct indices without creating additional data structures. Since the value of a stochastic gradient fluctuates due to

randomness, the lab instructions specify that the algorithm must use a fixed number of iterations instead of a convergence-based stopping criterion. This behavior was implemented faithfully.

By relying on the abstract Algorithm class, SGD could be integrated smoothly into the object-oriented structure without duplicating logic. Both full gradient descent and stochastic gradient descent now fit naturally within the same framework, although the focus of this laboratory was specifically on the implementation and understanding of SGD.

3. Conclusion

The final implementation worked correctly and produced coherent results for both raw and standardized datasets. Gradient Descent and Stochastic Gradient Descent yielded similar model parameters when appropriately configured. Transformations were consistently applied during prediction, confirming the correct behavior of SupervisedLearner.

Some challenges included handling vector dimensions, preventing errors caused by zero standard deviation during standardization, and tuning SGD hyperparameters to ensure stable updates. Overall, the lab successfully demonstrated how to integrate abstract classes, dataset transformations, and stochastic optimization into a clean and modular object-oriented design.