Project Summary

Prepared by: Carlos Benavides Viquez
SP-2136 Programación Avanzada
Sistema de Estudios de Posgrado
Universidad de Costa Rica

December 1, 2024

Contents

Chapter 1

Introduction

The objective of this project is to accelerate the process of predicting the risk of bone fracture using DXA (Dual-energy X-ray Absorptiometry) and Senior Fitness Test (SFT) data through the use of machine learning models, specifically the Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) algorithms. This analysis involves handling potentially large and dynamic volumes of medical data related to bone mineral density (BMD) and physical test results that correlate with fracture risk, such as muscle strength and balance.

A particular problem we are addressing is the normalization of the Senior Fitness Test data before it is utilized to create the SVM and KNN models. Normalization is crucial to ensure that each feature contributes equally to the distance calculations in the KNN algorithm and to the decision boundary in the SVM algorithm. This preprocessing step is computationally expensive, especially when dealing with large datasets, and requires efficient processing to obtain results in reasonable times.

Chapter 2

Justification for Parallelism

The implementation of parallelism is highly convenient due to the time-intensive nature of the calculations involved in data preprocessing (scaling and feature selection) and model training. Using parallelism will allow dividing computational tasks (such as data normalization, cross-validation, and feature selection) among multiple processing cores, significantly speeding up the analysis process. This is crucial for handling large datasets and performing iterative hyperparameter testing, which is typical in machine learning projects.

For the KNN algorithm, parallelism is particularly beneficial because the algorithm requires calculating distances between the test instance and all training instances every time a prediction is made. Unlike SVM, KNN does not create a model during training, which means that the computational load is shifted to the prediction phase. This makes parallel processing essential to efficiently handle large datasets and reduce prediction times.

Chapter 3

Functionality to Parallelize

The first part of the data processing necessary for fracture risk analysis is the normalization/standardization of DXA and SFT data. This involves scaling each feature (BMD, strength, agility, etc.) simultaneously across multiple processing cores. For example, for each patient, data such as Hip BMD, Spine BMD, Femur BMD, Lean Mass, Fat Mass, Bone Mass, Chair Stand Test, 8-Foot Up-and-Go, Arm Curl Test, 6-Minute Walk Test, Chair Sit-and-Reach, and Back Scratch Test are normalized. This normalization/standardization process must be performed for n patients (potentially thousands) and may need to be repeated multiple times with new data and different types/methods of data standardization.

Normalization is particularly important when working with SVM and KNN algorithms because it ensures that all features contribute equally to the model's predictions. In SVM, normalization helps in defining a more accurate decision boundary by preventing features with larger ranges from dominating the model. For KNN, normalization is crucial because the algorithm relies on distance calculations; without normalization, features with larger scales could disproportionately affect the distance metric, leading to biased predictions. Therefore, normalization is a critical preprocessing step to ensure the effectiveness and accuracy of these machine learning models.

Chapter 4

Implementation Details

The project is implemented as a web application designed to facilitate the collection and processing of user data for predicting bone fracture risk. The key components and functionalities of the implementation are as follows:

4.1 Web Application for Data Collection

The web application allows users to upload CSV files containing patient data. This data includes various metrics such as age, height, weight, and results from the Senior Fitness Test (SFT). The application is built using Next.js and React, providing a user-friendly interface for data management.

4.2 Data Storage and Management

Once uploaded, the patient records and their associated data are stored securely within the application. This ensures that all patient information is readily accessible for further processing and analysis.

4.3 Integration with Kabre Infrastructure

A critical feature of the application is its ability to establish a connection with the Kabre infrastructure. This integration allows the application to leverage the computational power of a supercomputer for processing large datasets. The connection is facilitated using the 'node-ssh' library, enabling secure and efficient data transfer and remote command execution.

In summary, the achieved integration with the Kabre infrastructure is as follows:

- Uploading Dataset to Kabre so that Machine Learning Models can be trained.
- Requesting normalization and prediction SLURM scripts from Kabre to process the data.
- Uploading and updating the SLURM scripts on Kabre to process the data.

4.4 Parallel Data Processing

The data processing tasks, including normalization and standardization, are implemented in C using the OpenMP library. This approach takes advantage of parallel processing capabilities to efficiently handle large volumes of data, significantly reducing computation times.

4.5 Code References

- The web application setup and data handling are managed in the Next.js project, as indicated in the ‘package.json‘ and ‘README.md‘ files. - The connection to the Kabre infrastructure is established using the ‘node-ssh‘ library, as specified in the ‘package.json‘ dependencies. - The data normalization and standardization processes are implemented in C, as seen in the ‘normalize-parallel.c‘ file: “‘c:docs/avance/normalize-parallel.c startLine: 1 endLine: 201 “‘

The provided C code implements a K-Nearest Neighbors (KNN) classification algorithm, which is designed to predict the class of a new data point based on its proximity to existing data points. The code utilizes OpenMP, a parallel programming model, to enhance performance by executing certain operations concurrently. Specifically, there are three main opportunities for parallelization within the code. First, the calculation of Euclidean distances between the new data point and all existing data points is performed in parallel using the ‘#pragma omp parallel for‘ directive, allowing multiple threads to compute distances simultaneously. Second, the sorting of distances and corresponding labels is also parallelized, enabling efficient organization of the nearest neighbors. Lastly, the counting of votes for each class among the K nearest neighbors is executed in parallel, with a critical section ensuring safe updates to shared variables. By leveraging these parallelization opportunities, the code significantly reduces computation time, making it suitable for handling larger datasets efficiently.

This comprehensive setup ensures that the application can efficiently collect, store, and process patient data, leveraging advanced computational resources for accurate and timely predictions.

Chapter 5

References

- OpenMP Architecture Review Board. (n.d.). *OpenMP documentation.* Retrieved from `https://www.openmp.org`
- Vercel Inc. (n.d.). *Next.js documentation.* Retrieved from `https://nextjs.org/docs`
- Meta Platforms, Inc. (n.d.). *React documentation.* Retrieved from `https://react.dev/`