K-NEAREST NEIGHBOUR (KNN) SERIAL VS PARALLEL
ASSIGNMENT 1

HIGH PERFORMANCE COMPUTING

SCHOOL OF COMPUTER SCIENCE AND APPLIED MATHEMATICS
UNIVERSITY OF THE WITWATERSRAND

KYLE WEIHER
1116087

MARCH 7, 2018

Introduction

Given a dataset, with m rows of datapoints, each point in $\mathbb{R}^d$. Given n query points that fall within the domain of the dataset, the k Nearest Neighbours (kNN) search problem involves finding the k nearest neighbours in a dataset to each query point, with regards to a specific distance metric. This problem appears in a number of areas, such as kNN classification and regression in machine learning. ******CITATION******

In this report an algorithm will be described, namely the bruteforce solution to the problem, which has the following basic structure:

1. For a specific query point, compute the distance from said point to every point in the dataset.
2. Sort the distances in ascending order.
3. Take the first k points with the shortest distances.
4. Repeat for each point in the query set.

The main focus of this experiment is to compare the runtimes of the algorithm, differing search algorithms and distance metrics, with regards to serial or parallel execution.

Methodology

A number of different distance metrics are available, and as such two will be tested in this experiment, namely Euclidean distance, and Manhattan distance. Furthermore, different search algorithms will be compared, namely quick-sort, merge-sort, and bubble-sort.

The experiment will cover a variety of variations to the above distance metrics and search algorithms, as well as testing between parallel and serial versions. There will be no mixing of parallel and serial, however, as holistically the kNN algorithm will be entirely parallel or entirely serial, and not a mix of both. Furthermore, both task and section constructs will be tested.

A number of considerations will be made when testing the different set-ups that the kNN algorithm can use:

- As the distance calculation and the sorting are not reliant on each other in terms of which algorithm is picked for both parallel and serial tests, when testing the different distance algorithms, the sorting algorithm used will remain the same.
- The fact that the amount of query points used, n, is somewhat of a global multiplier on run time, one test focussing on how n influences runtime will be performed, and then for the rest of the tests n will remain constant.
- As said above, the algorithm will be tested as either entirely parallel, or entirely serial, the will be no mixing of this parameter.
- **Perhaps add more with regards to relations**

Experimental Setup

The data used in these tests is randomly generated at run time. Before any computations take place, both the reference dataset and the query dataset are created, and then tests begin. The data is of type double, the code is written in C, and the parallel framework used is OpenMP.

The specifications of the machine that ran the tests are as follows:

- **OS**: Ubuntu 17.10 64bit
- **Kernel**: 4.15.7-041507-generic
- **Compiler**: gcc 7.2.0
- **CPU**: Intel i7-4720HQ (8) @ 3.600GHz
- **Memory**: 16GiB

Results