CEID
COMPUTER ENGINEERING & INFORMATICS DEPARTMENT

Parallel Processing

Spring semester 2022-23

P. Hadjidoukas, S. Gallopoulos

Function approximation with k-Nearest Neighbors

Issued: April 26, 2023

1 Introduction

The k-nearest neighbors algorithm (k-NN) is a non-parametric supervised learning method used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The output depends on whether k-NN is used for classification or regression. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. If k = 1, then the output is simply assigned to the value of that single nearest neighbor.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

2 k-NN regression

The training examples are vectors in a multidimensional feature space, each with a class label or value. The training phase of the algorithm consists only of storing the feature vectors and class labels or values of the training samples.

In k-NN regression, the k-NN algorithm is used for estimating continuous variables, where k is a user-defined constant. The value of an unlabeled vector (a query or test point) is predicted by using a weighted average of the values of the k training samples that are nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance.

This algorithm works as follows:

- Compute the Euclidean distance from the query example to the labeled examples.
- Order the labeled examples by increasing distance.
- Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

3 Surrogate modeling

Building prediction methods can be classified into two categories: physics-based methods and data-driven methods. Physics-based methods build physical models and compute their energy performance using simulation tools. The data-driven method does not require such detailed information about target buildings, and instead, it applies statistical theory to learn the patterns from recorded data for prediction. Moreover, the advances in machine learning algorithms also promote the development of data-driven models. As one of the top ten algorithms in data mining, K-Nearest-Neighbors (KNN) algorithm is commonly used and often performs better than other methods, such as Artificial Neural Networks and Support Vector Machines, for multi-class classification tasks and regression tasks with large sample sizes.

As a data-driven approximation of the original physics-based model, on the one hand, surrogate modeling does not require that much detailed building information and significantly reduces the computation burdens for physics-based methods. On the other hand, with sufficient surrogate models being pre-simulated, the required amount of training data can be provided for data-driven methods.

4 Sequential code

The goal of the project is the parallel implementation of a k-nearest neighbor regressor that follows the brute force search approach. A dataset generation code and a sequential, not optimized, C implementation of the k-nn algorithm is provided. More specifically, the dataset generator program generates training and query datasets of size $TRAINELEMS$ and $QYERYELEMS$, respectively. The dimension of the points is defined as $PROBDIM$ and the k number of nearest neighbors as $NNBS$. The multi-dimensional points are constructed by randomly selecting each coordinate within $[LB, UB)$. Finally, each point is associated with the floating point value computed at that point by a specified multidimensional function. The query dataset contains the set of points for which the function values must be approximated using the k-NN regressor. The actual function value is included for every dataset entry and can be used to evaluate the regressor.

For each query point, the regressor finds the $NNBS$ nearest neighbors and then approximates the function value by taking the average function values of the neighboring points without considering their distance from the query point.

The code computes the average percentage error (APE) and the mean squared error (MSE) of the predictions, and the quantity $R^2 = 1 - \frac{MSE}{VAR(y)}$, where $VAR(y)$ is the variance of the observed (actual) values. Moreover, it reports the total time for the predictions and the average time per request.

The data generator produces points for the 16-D "circle" function, which is defined as $f(x) = \sum n = 0^{n=D} x[i] * x[i]$. Figure 1 depicts the scatter plot of the 2-D "circle" function with the center at (0,0) in the domain [-6,6]. We have chosen a smooth function for simplicity reasons, but any N-dimensional function can be used and, in real-world cases, the function to be approximated correspond to computationally expensive simulations.

Besides surrogate modeling and function approximation, finding the k-nearest neighbors of a query point in a large dataset have several additional applications, such as text mining, image and video recognition, and recommendation systems.

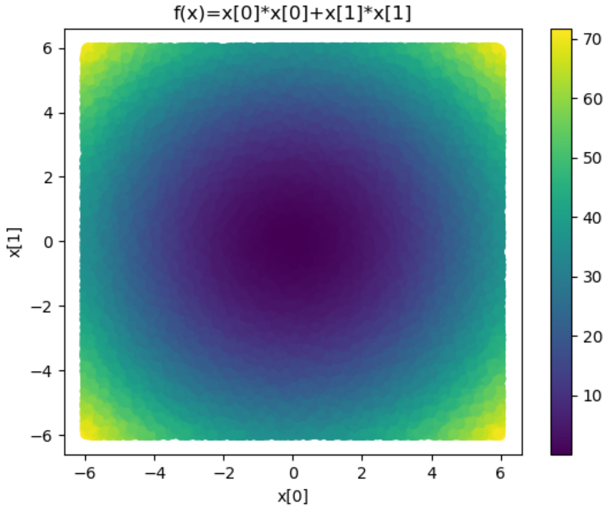

Figure 1: The "circle" function in 2D.

The default experiment, as defined in the provided Makefile, uses the following configuration options:

- problem dimension = 16
- nearest neighbors = 32
- training points = 1048576
- query points = 1024
- lower bound= 0
- higher bound = 2

Typical usage of the provided sequential code is depicted below:

```
$ make
$ ./gendata x.txt q.txt
1048576 data points written to x.txt!
1024 data points written to q.txt!
$ ./myknn x.txt q.txt
Results for 1024 query points
APE = 4.86 %
MSE = 1.803033
R2 = 1 - (MSE/Var) = 0.920909
Total time = 52634.605885 ms
Time for 1st query = 46.855211 ms
Time for 2..N queries = 52587.750673 ms
Average time/query = 51.405426 ms
```

5 Tasks

You must optimize the sequential implementation of the k-NN regressor that is based on brute-force search.

1. Implement the inverse distance weighted average and provide accuracy and performance results.
2. Parallelize the code using OpenMP: exploit multi-core architectures.
3. Parallelize the code using the OpenMP tasking model.
4. Parallelize the code using MPI: exploit multi-core architectures and clusters of multi-core systems.

Remember that your code must be

- correct: it provides similar predictions (same) to those of the initial sequential version.
- efficient: the overall time for processing the query points and the average time per query is minimized.

You are allowed to reorganize the code and the adopted data management approach. You can modify the data generator and the regressor accordingly to work with data stored in binary format. The default implementation includes the overhead for reading the query data from the text file. However, the dataset can be loaded beforehand, and the time measurements will not include the specific I/O overhead. You can adjust the problem size (number of train and query elements) to have a total execution time compared to the time in the example.

In your report, you must

- Describe your design and implementation decisions
- Explain any optimization techniques that you applied
- Give an overview of the hardware platforms that you used
- Provide performance measurements and plots of your programs
- Analyze your results

You will use the course e-class to submit a zip file with your report (pdf) and source code. Do not include the datasets and any executable files. Please ensure you include information about any special steps that might be needed during the compilation and execution of your programs.