Lab Course: distributed data analytics
Exercise Sheet 2

Nghia Duong-Trung, Mohsan Jameel
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: Friday 23:59PM (on LearnWeb, course code: 3117)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit a zip or a tar file containing two things a) python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.
4. If you are M.Sc. Data Analytics summer 2017 intake student, you should submit to "First term students" link on LearnWeb.
5. And if you are M.Sc. Data Analytics winter 2016 intake student, you should submit to "Second term students" link on LearnWeb.
6. If you are not M.Sc. Data Analytics student, you can submit to anyone of the links above.

Case study: k-Nearest Neighbor (k-NN)

In this exercise sheet, you are going to use threading and multiprocessing APIs to solve a widely used classification technique called K-nearest neighbors (k-NN). In machine learning, the k-NN is a non-parametric method used for classification and regression [1,2]. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

This exercise sheet focuses on the implementation of threading and multiprocessing APIs. Thus, you are going to implement a light version of k-NN classification, $k = 1$, $X \in \mathbb{R}^{n \times m}$ be the training set, $x_0 \in \mathbb{R}^m$ be a testing observation, where:

1. The training and test observation's labels are not considered.
2. The test set only contains a testing observation.
3. The output is the index of the training observation which is the nearest to the testing observation.
4. The distance between a pair of observations is calculated using the Cosine similarity [3].

Throughout this exercise sheet, the k-NN problem refers to the characteristics described in this section.

The lecture and lab course slides provide the basic introduction to the APIs. You have gained some parallel programming experiences through the first exercise sheet. In the Annex section there are few useful resources that will help you further understand k-NN and provide help in solving following exercises.

Exercise 1: Formalize the task of k-NN (5 points)

In this exercise, you are going to formalize the task of k-NN described in the above section. More specifically, you are going to formalize (1) how the training and testing sets look like, (2) the distance measurement, and (3) the pseudo-code of the task. Note that the values of n and m might be set to (10^3, 10^4, 10^5, 10^6).

Exercise 2: Solve the k-NN sequentially (5 points)

In this exercise, you are going to implement the k-NN without using the idea of parallelization. This implementation will help you compare the running time between sequentiality and parallelization.

The output of the program should report some basic information such as:

1. The maximum cosine similarity.
2. Which training observation that we get the maximum cosine similarity.
3. Time of calculation for method that does not implement threads/processes.

Exercise 3: Solve the k-NN in parallel (10 points)

In this exercise, you are going to improve the exercise 1 using the idea of parallelization. First, let's describe which part(s) of the k-NN that you can implement in parallel. Briefly explain why you want to parallelize the part(s).

The output of the program should report some basic information such as:

1. The maximum cosine similarity.
2. Which training observation that we get the maximum cosine similarity.
3. Time of calculation for method that implements threads/processes.
4. For the method that implements threads/processes, you should compare the time of calculation within a range of different number of threads/processes, e.g. the number of threads/processes might be $1, 2, 3, 4, \ldots$.

Annex

1. Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185.
2. K-nearest neighbors algorithm: `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`
3. Cosine similarity measurement: `https://en.wikipedia.org/wiki/Cosine_similarity`