LSE AI subject 2018

Contents

1 Introduction

This project is the opportunity to evaluate your *parallel programming skills* and your interest in machine learning methods. We will check if you are able to work on IA projects with LSE IA team. The project is composed of three steps, the last one is optional. You will approach some notions around vectorization, parallel programming and deep neural network methods.

Your project has to classify binary programs into three classes :

1. malicious,
2. benign,
3. or *unknown*.

For this task, you will use the EMBER(*Endgame Malware BEnchmark for Research*) dataset[1].

Remember that all your code will be evaluated (and read) manually, but we can't guess how to use it. On the other side, your text outputs don't need to match some kind of specification, common sense should dictate what you write. You can do anything you like besides what is written on this subject, just remember to add it in your README.

A description of your work and conclusions should be provided in your "report.pdf"

Have fun and impress us!

[1]You will find a short description at:
https://www.endgame.com/blog/technical-blog/introducing-ember-open-source-classifier-and-dataset
and a detailed description at : https://arxiv.org/abs/1804.04637

2 Basic Rules

2.1 Build system

- Your build system must be clean and well built.
- You can choose to use either:
  - autotools
  - cmake
  - make
  - meson.
- The binary kmeans program for the first step should be produced at the root directory.
- The src directory should contain three directories step1, step2 and step3 corresponding to each part of the project.

2.2 Authorized Libraries

Your code will be tested with the matching archlinux packages.

- glibc
- libmath
- openmp
- tensorflow
- keras
- numpy.

Ask for permission if you need anything else.

We provide you a training and testing labeled data. The XTrain.dat file contains the training feature vectors. YTrain.dat gives the label*(-1, 0 and*

1) for each feature vector. This part of the data can be used for the training part of neural network, and in the evaluation of the computing performance of the K-means program *(step1)*. It is composed of 900K examples. The test dataset (Xtest.dat and Ytest.dat) will be used in the evaluation part of the neural method, it contains 200K vectors.

We provide also a simple implementation of the k-means algorithm in the file kmeans.c and a python script "eval.py" to evaluate the performance of your machine learning algorithms.

All these files should be downloaded from `http://ia.dehak.org/`

2.3 Coding Style

We don't check your coding style for this project, but since your code will be read by humans, it must be *understandable and clean.*

Some rules though, we hate 2 indented spaces. Please use 4 spaces, or 1 tab (with a size of 8). Also, we have small terminals, and like lines that have at most 80 columns.

3 Objectives:

The goal of this project is to create the fastest k-means program that classify feature vectors into K clusters. We will use this program to classify malware programs into three classes: *Malware, Benign and Unknown.* The program will be used to classify the feature vectors of EMBER dataset.

3.1 DataSet

The EMBER dataset is a set of 900K training programs feature vectors, each one is a real vector with 2351 components. We provide the database in C Binary files format.

3.2 K-means

In this project, you will produce a fast k-means program in C ”kmeans.c”.

K-means is an unsupervised learning method, used when you have unlabeled data (which is the case if you don't use the labels). The goal is to make up K clusters of similar data, with K being a chosen variable. You can find the pseudo code here. In this work, we have chosen K = 3 and the distance is the classical Euclidean distance.

4 First Step: Parallel programming

For this part, you must propose a fasted version of the K-Means algorithm using multi-thread and vectorization programming approaches. For vectorization, your Program will use Advanced Vector Extensions 2 (AVX2) which is supported by most recent laptops and intel CPUs.

4.1 Thread level

The k-means algorithm can be easily parallelized at a thread level, you may speed up your algorithm using OpenMp[2].

4.2 Instruction level SIMD

The algorithm can be made faster by performing SIMD (Single instruction multiple data) vectorization, the same operation multiple times, all in one instruction. OpenMP is a feature of your compiler, which enables (among other things) giving hints about which loops are worth trying to vectorize. The compiler still may not recognize anything it can emit vector instructions for, in which case you may have to write these yourself using low level bindings (see immintrin.h and https://software.intel.com/sites/landingpage/IntrinsicsGuide)

4.3 Work

1. Propose a fasted version of the provided K-Means program.

[2]https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf

5 Second Step: Deep Neural Network

This part of the project will evaluate your research skills. Your submission must be python3 code, you must use tensorflow[3] or keras [4]. You should explain what you have done in your report.

5.1 Metric Learning : Siamese networks

Siamese network are a good class neural network to find relationship between two inputs on the same class. Siamese network will compare the output of two different inputs and then measure their distance. The distance should be small if two inputs are in the same class.

In these part, we will improve the representation of your feature vector for your k-means. The siamese network should find a new feature vector representation that will improve the k-Means algorithm performance.

You must propose a solution using the triplet network to enhance the representation. You can found more details about triplet network in the paper of Hoffer et Ailon "DEEP METRIC LEARNING USING TRIPLET NETWORK"[5]. Triplet network is a new method inspired from siamese networks.

5.1.1 Work

1. Propose a topology of your deep neural network to enhance your representation feature vector.

2. Train your triplet network on training data

3. Test the performance of your kmeans with (K=2) on test data after projection using the proposed triplet network.

4. compare the result with kmeans (K=2) on ember features?

5.2 Malware Classification using Deep Neural Network

Rather than using the K-means algorithm to classify your data, we can use deep neural network which provide directly the class of your example. In

[3]https://www.tensorflow.org/tutorials
[4]https://keras.io/
[5]https://arxiv.org/abs/1412.6622

this part you should propose a deep neural network for malware classification. You can found a lot of documentation and tutorial on how to built a classification deep neural network on the web.

5.2.1 Work

1. Propose a topology of your deep neural network.
2. Train your deep neural network on training data.
3. Test the performance of trained network on the test data.

6 Third Step: Optional

In this step, you are free to use whatever you want, the goal is to obtain the best performance on test data.