BML Programming Assignment I

Naive Bayes

Deadline: 06.01.2025 23:59

1 Naive Bayes

Your task is to implement the distributed Bernoulli Naive Bayes classifier for the text classification. Then, you are supposed to perform experiments on the Google Cloud virtual machines and write a short report on the results.

A Naive Bayes classifier assigns a class that maximizes

$$p(C_k)\prod_i p(x_i|C_k),$$

where $p(C_k)$ is the a priori probability of class C_k and $p(x_i|C_k)$ is the probability that the value of feature x_i is produced by class C_k. Assume you want to train a Naive Bayes classifier on a very large corpus of text-like documents following a standard Bernoulli Naive Bayes method: the feature vector is binary and the i-th coordinate $x_i = 1$ if and only if the i-th word from the dictionary occurs in the document to be classified. The a priori class probability $p(C_k)$ is the fraction of documents classified to the class C_k, and the feature probability $p(x_i|C_k)$ is the fraction of the documents of class C_k containing the word x_i. This classifier then models

$$p(x_i|C_k) = P(x_i = 1|C_k)x_i + (1 - P(x_i = 1|C_k))(1 - x_i).$$

2 Two-stage solution format

We expect a two-stage solution. The first stage is training. During training, you want to compute $P(x_i = 1|C_k)$, i.e., the frequency of the word x_i appearing in the documents of class C_k. Write an efficient distributed algorithm computing this frequency, given that the corpus is large and distributed across many machines in a cluster. The input file will be a CSV file with no header and two columns, the first column containing a text to classify, and the second column containing the label: an integer from 0 to the number of the labels - 1. Each worker will receive its file, with a part of the whole dataset.

Finally, each worker should save the result to the CSV formatted file. The result file should contain no header, the first row should contain the list of words in the vocabulary listed in an alphabetic order. The following rows should contain vectors of the word probabilities for each class and the last column should contain probability of the class $P(C_k)$. The vocabulary should be created from continuous alphabetic characters in the whole dataset, case-insensitive. Then all the words occurring only once should be removed from the vocabulary. The vocabulary should be saved to the output file in lowercase and without any quotes. **The training stage should be submitted as a Python program named `train.py` that gets two command line parameters: the dataset path and the output file path.** The program will be executed by the `mpirun` command.

The second stage is the classifier, which should read the result file from the first stage, the input file with several lines, each line containing a document to classify. Then it should save the predicted classes in the result file: the number of each prediction in a separate line. **The testing stage should be submitted as a Python program named `classify.py` that gets 3 command line parameters: training results file path, input file path and output file path.** The classifier should be sequential.

3 Experiments description

Finally, you are supposed to perform experiments using the Google Cloud virtual machines. The dataset, to run the experiment on, is a part of the Amazon Reviews'23 `https://amazon-reviews-2023.github.io/` dataset and is available under

`http://mimuw.edu.pl/~tk385674/amazon_reviews_2M.csv`.

The text column is a concatenation of the title and the review, while the label denotes the product category. We ask you to measure both the strong and weak scaling of the training process. For the strong scaling, you should measure the time it takes to train the classifier on the whole dataset, using respectively 5, 10, 15 and 20 VMs. For the weak scaling, you should also run training on 5, 10, 15 and 20 VMs, but in this case, the amount of data per VM should be constant. It means, that you should sample smaller datasets from the original dataset, that have respectively 5/20, 10/20 and 15/20 fractions of records (the last run may be shared with the strong scaling to save credits). Note, that in all experiments, you are also supposed to shard the whole dataset into equal chunks and distribute them over all VMs before starting the training (to pretend that the dataset is stored in a distributed manner). All the experiments should be performed on the **'t2d-standard-1' Spot** VMs with single worker per VM. Note, that by using spot VMs there is a small risk that your experiments will be terminated, but it makes them much cheaper. In the case of termination, you should just retry the experiments. Provide the run times of your (successful) experiments in the `report.pdf` file. Comment on those results, and provide your hypothesis as to why the results came out the way they did. Finally try to predict how long would it take to train the whole dataset on 50 and 100 VMs. Describe your method of making such a prediction and argue why have you chosen this method.

4 Formal requirements

To submit your solution use the following Google Form: `https://forms.gle/aBdFPgHhhKm81QLb8`. You should submit a single ZIP archive containing a single folder named the same as your students login: `ab123456`, where `ab` are your initials and `123456` is your student number. Inside this folder, there should be 3 files: `train.py`, `classify.py` and `report.pdf`.

Your solution should run in Python 3.12 in the environment containing packages: torch, numpy, pandas, scipy, mpi4py. Therefore you **can't** use other Python packages. The training will be run using the `mpirun` command from OpenMPI, and will have the `MASTER_ADDR` and `MASTER_PORT` environment variables set respectively to the address of one of the workers, and a free port on this worker. The classifier will be run sequentially.

You can get a total of 30 points for the assignment. They are divided into several categories:

- Correctness of the training: 12 points
- Performance of the training: 8 points
- Correctness of the inference: 2 points
- Report: 8 points

Your solution needs to score at least 3 points for the training correctness to get a positive score for the performance and report.

You will lose 1 point for every started 6 hours of the submission delay.

5 Important remarks

Google Cloud imposes a quota of max 8 IP addresses per region. So to perform experiments on more than 8 VMs you need to rent them in different regions. The easiest way to do this using Terraform, is to add a 'zone' parameter to the resource block and copy the block for every region, changing the value of the 'zone' parameter appropriately. Some cheap regions you can use: us-central-1, us-east-1, us-east-5, us-west-1 (to create zone name just append '-b' to the region name).

IT IS IMPORTANT TO USE SPOT VMS! The regular VMs are 6 times more expensive and will eat through your credits quickly. The instruction how to create Spot VM using Terraform are for example here: `https://cloud.google.com/compute/docs/instances/create-use-spot#terraform`