C3: Large Scale Machine Learning

Data Engineering II 7.5HP, 2023

In this assignment you will explore some of the challenges and possible solutions arising when scaling up machine learning tasks and pipelines to a distributed setting.

Part 1: Distributed machine learning

This part constitutes the mandatory part of the assignment, and its successful completion awards 1 point.

1.1: Hyperparameter tuning
A common task in all machine learning pipelines is to tune so-called *hyperparameters* of the ML algorithms. These are parameters involved in the training process that are not learned from the data itself, but typically related to the optimization routine used. Examples include the learning rate for SGD, penalty terms, batch sizes, and many other things. A good introduction to the problem with examples for sklearn using cross-validation is found here:
https://scikit-learn.org/stable/modules/grid_search.html

The problem is that since the ML model training is often expensive, hyperparameter tuning becomes a highly computationally demanding and hence time-consuming process. Fortunately, it is possible to treat it as a task-parallel problem, where the ML-model training is systematically repeated for different combinations of hyperparameters, in the search for optimally performing combinations in the global space of hyperparameters. Strategies range from simple ones like exhaustive Random search and Grid Search, to more sophisticated alternatives like Bayesian Optimization. There are several established frameworks for distributed parameter tuning such as RayTune, HyperOpt and DaskML. Spark, which you encountered in Data Engineering I, also provides functionality for models made in Spark MLLib.

Your task now is to use scikit-learn to train a RandomForestClassifier https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html for predicting the forest cover type. The dataset can be downloaded directly via the sklearn API: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_covtype.html#sklearn.datasets.fetch_covtype and there are several notebooks on Kaggle where you can learn about the data and background to the dataset: https://www.kaggle.com/c/forest-cover-type-prediction.

Do the following:

1. Set up the problem using sklearn. Explore the parameters used in the Random Forest implementation. Hint: use the get_param() method. Train the model using the default parameters.
2. Implement a distributed hyperparameter tuning pipeline for tuning the parameters ‘max_depth’, ‘n_estimators’ and ‘ccp_alpha’ using Ray Tune and an exhaustive Grid Search strategy. Configure, deploy and run it on up to 3 VMs of “small” flavor.

Please report on:

1. The hyperparameters found and the associated cross-validation score. How does it compare to the score for the default parameters?
2. The time taken to complete the tuning, when using 1, 2 and 3 VMs of “small” flavor.

1.2 Distributed training of deep neural networks

Another challenge that calls for large scale computing is when training very large models, and/or when the training dataset is very large. Again, distributed computing provides a solution to speed up the training process, but now it is less straight-forward how to achieve parallelism.

In this task, which will be mostly theoretical in nature (unfortunately we are not able to provide access to GPU clusters), you will explore available approaches to scale training by leveraging multiple GPUs/TPUs.

Since you are familiar with the PyTorch library, start by reading up on this background:

- Background on data parallelism: https://en.wikipedia.org/wiki/Data_parallelism
- Implementation of DataParallel training in PyTorch: http://www.vldb.org/pvldb/vol13/p3005-li.pdf
- Documentation for PyTorch Distributed: https://pytorch.org/tutorials/beginner/dist_overview.html

Then study and do a tutorial using another broadly used ML stack - Keras/Tensorflow. While the APIs and implementations are different, you will recognize the common themes for parallelism:

1. Distributed Tensorflow (on CPU)
    a. https://www.tensorflow.org/guide/distributed_training
        i. Do this tutorial in Google Colab: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/guide/distributed_training.ipynb

Equipped with this background knowledge, explain (in max 0.5 page 11pt Arial) the main difference between data parallel training strategies and model parallel training strategies. In which situations are the respective strategies most appropriate to use? Assuming you also need to do hyperparameter tuning, in which situations would you consider distributing the training of individual neural networks rather than distributing test cases for the tuning pipeline?

Part 2: Federated Learning

Awards up to 2 extra credits:

Federated machine learning is an emerging class of distributed and/or decentralized schemes that sets out to solve privacy problems in machine learning. In many situations, multiple organizations, each having private datasets, might want to collaborate to jointly train a ML model, *without disclosing their private data to other parties.*

1. Read this introduction to Federated Learning: https://en.wikipedia.org/wiki/Federated_learning
2. Read this paper about Federated Averaging: https://arxiv.org/pdf/1602.05629.pdf
3. Read our preprint about FEDn: https://arxiv.org/abs/2103.00148

FEDn, https://github.com/scaleoutsystems/fedn, is a software developed jointly by the ISCL lab and the spin-off company Scaleout Systems. It can be used to set up production systems for horizontal federated learning in a ML-framework agnostic manner.

Task: Use FEDn to set up and run fully distributed federated training for either the Keras or the PyTorch MNIST example (bundled in the repository) on three virtual machines. Use one for the base services (MongoDB and Minio)+reducer, one for one combiner and one to run 2 clients. Include a snapshot of the training result in the Dashboard. Then answer the following questions:

1. Write a short (max 0.5 page Arial 11pt) reflection on how federated learning differs from distributed machine learning. Focus on statistical and system heterogeneity.
2. Write a short (max 0.5 page Arial 11pt) discussion on how federated learning relates to the *Parameter Server* strategy for distributed machine learning.