DISTRIBUTED FRAMEWORK FOR TESTING MACHINE LEARNING METHODS

Jan Klusáček
Doctoral Degree Programme (4) , FEEC BUT
E-mail: xklusa00@stud.feec.vutbr.cz

Supervised by: Václav Jirsík
E-mail: jirsik@feec.vutbr.cz

Abstract: When designing new Machine learning (ML) methods, solid testing is very important part of process. This article describes a framework that was created for automated testing and comparison of different ML methods. This framework allows to automate most of tedious and recurrent tasks related to comparison and testing of new methods.

It consists of two parts. First part is intended for work with results of ML methods. It allows to compare results of different methods with different settings. It allows to create tables and graph from these results and it also perform statistical tests on these results.

It is often necessary to perform considerable number of test runs on different datasets and with different settings for purpose of comparison and statistical test. For these reasons it is advisable to automate these tasks as much as possible. Automation of these tasks is purpose of second part of this framework. It allows to divide, plan and execute tasks on remote machines.

Whole framework is written using Python and Django framework allowing to easily extend customize it for particular task.

Keywords: Machine learning, Python, Parallelization, Web interface

1 INTRODUCTION

Regular approach when developing Machine learning (ML) methods[1] is to compare them to the existing method. It is also possible to search for best parameters of single method by comparing its results to results gained with different settings. In both cases, approach is pretty similar. It consists from creation of model using particular settings and training datasets. This model is than used to predict values of the testing dataset. Resulting predictions are then compared with correct labels and performance of this model is evaluated.

1.1 COMPARING RESULTS OF MACHINE LEARNING METHODS

Generally it is not sufficient to perform just one test on one dataset to compare performance of different models[1]. Improvement in one test can be easily coincidence. For this reasons, methods are usually compared on multiple different datasets. One of problems when evaluating performance of model is division of the dataset between training and testing subsets[2]. Using bigger training dataset improve quality of model, but resulting smaller testing set means lower reliability of performance estimation and vice versa. Cross validation can be used to improve reliability of these tests further.

[1]Methods considered in this work are belongs to group of supervised learning. Described framework can work with unsupervised methods but evaluation of quality of these methods is different from supervised ones

[2]It is desirable to use two disjoint datasets for training and testing model. If one dataset is used for both training and testing, resulting performance estimates are optimistic and underestimate error. This is a result of ignoring overfitting error (lack of a generalization).

The main principle behind cross-validation is division of original dataset to multiple disjoint subsets[2]. Different combinations of these subsets are subsequently used to create models and evaluate their performance. Performances acquired this way are subsequently used to estimate the performance of model created using the whole dataset. This approach allows to use all data for creation of a model and for testing (not in same time, but in many different models models). The result is a better a model with a good estimation of its performance. Cross validation also allows to estimate a stability of a method (its sensitivity to changes in input data) by comparing changes of performance on slightly different datasets.

1.2 TASK PARALLELIZATION

From the previous description, it's obvious that a simple comparison of two methods require considerable number computations. Depending on character of used models and datasets, these computations are often very time consuming and can take many hours or days when executed on a single processor of consumer computer, for this reason parallelization of testing is desired. One of possible solutions of this problem is to use high performance machine. This approach clearly mean additional costs, but today it is not necessary to buy own hardware. It is possible to rent desired hardware for hourly fee[3]. Another solution is usage of ordinary computers when they are not used. This approach doesn't require additional investments to hardware, but it is limited by available time (computers can be used only when they are not required for their primary function) and it is not suitable for some tasks[3].

1.3 PARALLELIZATION TYPES

As mentioned previously, parallel tasks can be divided according to their requirements for exchanging intermediate results (inter-process communication). Most demanding tasks require frequent communication and are very sensitive on the communication latency. These tasks are called fine grained.

Fine grained tasks can be run on multicore microprocessors or on multiprocessor machines, that allows very fast (high bandwidth and short latency) communication between individual cores. Depending on type of task it's possible to run these tasks on computers interconnected with high performance network such as Infiniband with various performance penalties.

Coarse grained tasks also needs to exchange intermediate results, but not so often as fine grained tasks therefore requirement on the communication between machines are much lower and Ethernet is usually sufficient.

Some of the tasks require almost no communication between individual tasks. These tasks are called embarrassing parallel tasks.

Tasks used for testing of ML methods are mostly in class of embarrassingly parallel tasks because they are independent of each other. Each task creates one model with specific settings and a dataset and evaluate its performance. Only necessary communication is concentration of all results required for comparison and statistical tests.

2 TESTING FRAMEWORK

To address problems introduced above simple framework was created. This framework is written in Python. Its goal is not to create a complete application that can be easily deployed, requiring complex configuration depending on a particular usage. It is rather framework that takes care of tedious tasks and let user focus on a real problem. Instead of using complex configurations that try to address all possible scenarios, this framework implements a functionality required by all tasks and a customization is achieved by inheriting from prepared classes. This way it is easy to add a custom behavior, without need to prepare support for high configurability. A drawback of this approach is

[3]Some tasks require fast communication between threads, that is not achievable using a network. Another problem is if task require big amounts of memory

that user needs at least basic programing skills, but it should be no problem for people designing ML methods.

Whole testing framework is written in Python 2.7. using Django framework for access to database and visualization. Python was chosen because it allows a fast development and it is very widespread in machine learning community. It also allows to write a multiplatform code[4]. Basic diagram depicting frameworks function is in Fig. 1

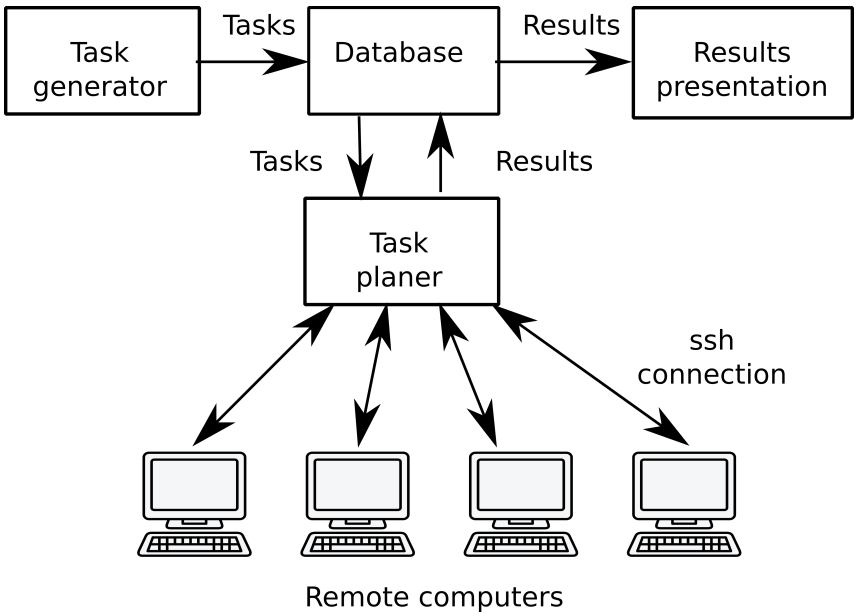

Figure 1: Framework diagram

All test configurations and their results are stored in database[5]. It allows to easily access configuration using web interface and at the same time it simplify presentation of results.

2.1 TASK CREATION

First part of process is a creation of tasks. All necessary functionality for one task are implemented in class `Task`. This class is derived from Django `Model` class, this mean that it is possible to easily load and save this class to database. Most important property of this class is `task_data`. This property contains a pickled[6] object that will be transferred and executed on a remote machine. User of the framework have to prepare an object with method `work` and save it to this property. It will be automatically executed on an available remote machine and a results will be automatically transferred back. `Task` class also allows to define some additional properties. One task can for example require another task finished before it is executed (this is useful if one task require some results from a previous task). Every task can also add requirements on remote machine (minimal available RAM).

2.2 TASK PLANER

Another part of the framework is a task planner. This python module can run as a thread in a Django server, or it can be run as a standalone daemon. It has two main tasks. It maintain list of available re-

[4]Described testing framework is currently able to run on both Windows and Linux without any modifications

[5]SQLite is currently used as a database engine but thanks to Django framework, it is possible to easily switch to a more powerful database engine if higher performance is required

[6]cPickle is a python library that allows to serialize a generic python object to a binary string and later recreate it using this string. It has some limitations (for example it is not possible to serialize file handles, sockets and similar resources), but it is very comfortable way of transferring an object to a different process, possibly on a different machines

mote machines and their state (e.g. ready, waiting, offline) and it assigns `Task` loaded form database to the available remote machines.

Access to remote machines is provided by ssh with usage of private keys. A list of all unfinished tasks is loaded every time one of previous tasks is finished and suitable task is selected from list for the available machine. Pickled data of task are then transferred to the remote machine, they are unpickled and executed. When the computation is finished its results are saved to the database and a next task is selected and executed.

Several possibilities for communication with remote machine were considered before selecting ssh. It would be possible to send task data to remote machine using http or https protocols, but it would require installation http server on each remote machine. Another possibility is usage of telnet or custom tcp/ip protocol. These possibilities were ruled out because they would require unnecessary amount of work and would take long time to debug, especially if they should be secure to use over internet. For these reasons ssh was selected. It not only allows to securely send data over untrusted networks but also allows to control remote machines. It is also heavily used and tested, so there should be minimal number of bugs. Its drawback is that its encryption can slow down transfer of large amounts of data. This can be overcome by disabling encryption (which would require recompiling ssh client and server), or selecting faster cipher (Blowfish, ARC4).

2.3 RESULTS VISUALIZATION

Last part of the framework is visualization of the results. Similarly to previous parts the framework does not provide any complete solution for visualization as is, but it is very easy to use existing libraries to create tables and graphs as necessary. Used Django framework provides easy way to create web interface.

All results are saved in the database, so it is quite easy to access them and make simple statistics directly using SQL functions. Whenever more complex processing is necessary, it is possible to use existing python modules such as ScyPy. Tables can be easily created using library tables2 (provided as part of Django framework). MatPlotLib can be used for creating various graphs.

3 EXAMPLE OF USAGE

The described testing framework is currently used for development of the new method of transformation of feature space transformation. This task is little different from testing of classification methods, but it is still possible to use this framework. In this case, the task form chain of requirements longer than usual. The first task uses a whole dataset to find transformation of feature space. Subsequent task divides original dataset to ten parts for purpose of a cross-validation and apply the transformation found in previous task. Following tasks prepare different models with different settings on datasets from previous task and evaluate their performances. All these tasks are repeated for five different original datasets. Example output is in Fig. 2.

4 CONCLUSION

The new framework introduced in this article enables easy parallelization when testing and comparing ML methods. It has also prepared methods that allows easily process results, apply statistical tests and present results in form of tables and graphs using a web interface. Access through web interface allows to access from remote computer so it is possible to have one continuously running computer and check progress and results of testing from different machines in different places. Web interface is also most cross-platform way of creating GUI.

Designed framework is currently used for developing new ML method as described in previous chapter. Remote machines used for computations are currently heterogeneous set of different available

Sumary table for best result			
-		Method	
Model	Dataset Name	IWFD	Gram. Evol. IWFD
K1	numerals	0.984(18)	0.973(20)
	sensvehicle	0.709(16)	0.689(15)
	spectro	0.882(16)	0.859(20)
	isolet	0.738(20)	0.847(20)
	optdig	0.965(20)	0.937(20)
	scene	0.522(8)	0.639(20)
	satelite	0.893(20)	0.853(17)
AMLP	numerals	0.971(15)	0.966(19)
	sensvehicle	0.711(5)	0.590(3)
	spectro	0.853(10)	0.878(18)
	isolet	0.759(20)	0.827(14)
	optdig	0.895(20)	0.867(20)
	scene	0.582(8)	0.673(20)
	satelite	0.797(3)	0.830(2)
	numerals	0.971(15)	0.966(19)
	sensvehicle	0.711(5)	0.590(3)
	spectro	0.853(10)	0.878(18)

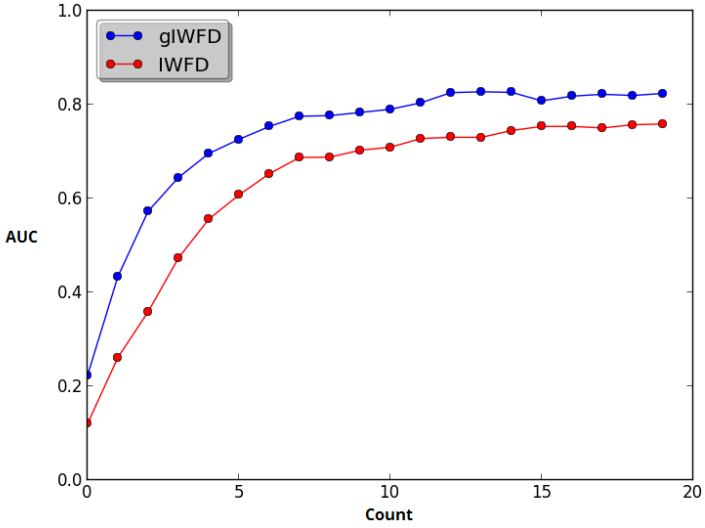

Figure 2: Example of web interface output comparing accuracy of two methods using table and graph.

computers. This state is far from ideal for further testing it will be necessary to replace these computers with higher number of (ideally) same computers. Therefore it is desirable to automate preparation of system for these computers. Ideal solution will be booting these computers from a network because it allows to easily prepare one central image with a preconfigured operating system for all these computers. Second advantage of this approach is that it does not require any changes to settings of current OS, meaning that these computers can be easily used for their original purpose just by rebooting from their hard drive.

Acknowledgement: The completion of this paper was made possible by the grant No. FEKT-S-14-2429 - „The research of new control methods, measurement procedures and intelligent instruments in automation" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[2] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[3] EC2 Instances Pricing. `https://aws.amazon.com/ec2/pricing/`. Accessed: 2016-03-15.