THREE OBSTACLES IN DATA SCIENCE AND ONE VISION

by Patrick Michl

20. March, 2019

For the current development- and exploration process in data science three obstacles in particular appear as outstanding hurdles, when it comes up to realize projects - and even more, when it comes up to venture collaborations.

Some years ago - in the early 2010s - when Google's TensorFlow still was only an idea and Geoffrey Hinton's daredevil Science Article still only received a bunch of citations, the undisputed technical issues in data science were the absence of computing power and the absence of a common play ground. Of course, during the last decade, NVIDIA and Google respectively stepped into the breach with CUDA and TensorFlow. So the question arises "What are today's foremost technical obstacles in data science?". In the following I present our personally experienced proposal to this question and our vision: The Liquid ML framework.

#1: The Plug Jumble

Data scientists require statistical samples to work with, but not data-backends. The integration of the data-backends within the analysis pipeline, often turns out as an unappreciated and frustrating job: For a comprehensible application this extra task may still be quite manageable, but for collaborations with different operational data landscapes, the additional efforts can occupy a lot of time and become a critical factor.

We want to solve this issue with Pandora, a lightweight unified data interface, that mediates between the data analysis application and the data-backends. On the analysis side, Pandora provides peace by offering data scientist their favorite formats: NumPy-Arrays and R-Tables. On the (dark) backend-side Pandora aims to tame any important SQL-Databases (IBM Db2, Oracle Database, SAP HANA, Microsoft SQL, MySQL, Postgresql, ...) as well as laboratory measurement devices and flat files, that appear in the wild.

#2: Paper Bottlenecks

The development- and exploration process in data science heavily depends on the ability to adapt current cutting-edge approaches. This ability, however, frequently is impaired by the detour experienced by publications in paper form: Due to the limited space, the provided pseudo code often loses valuable details over the original algorithm. But also if the original algorithm is provided online, it can take tremendous efforts to properly identify it's scope and adapt it to the underlying prerequisites - only to decide about it's suitability!

We want to automate this process with Motley. Motley is a smart algorithm repository server, that enforces unified data interfaces for different algorithm categories. This allows Motley not only to automatically evaluate and compare the hosted algorithms with respect to given metrics but thereupon also to determine, which algorithm of a given category and data domain is the currently best fitting (CBF) algorithm with respect to the required metric.

#3: Dead Horses

Due to the rapid scientific advances, data science and analytical applications like in no other domain suffer of short code lifespans. Of course:

> “ *When you're riding a dead horse, the best strategy is to get off!* ”

So the question arises, how to keep the code alive. This issue can only be addressed by letting the code be dynamic!

For this purpose we started to develop Nemoa, a templating machine-learning framework, that orchestrates TensorFlow. However, Nemoa does not simply provide a new interface, but abstracts the coding process by following our Cloud-Assisted Meta Programming (CAMP) paradigm. The fundamental observation behind Nemoa is, that it is almost never required to use a specific algorithm but only one that does the job - so why not simply use the best one, that's currently available? This is the point, where Motley and it's CBF algorithms join the game. And finally to easily integrate the application into any existing operational data landscapes also Pandora joins the team. So these three together constitute our Liquid ML framework.

The Vision: Liquid ML

If you are a data scientist imagine the following situation: The new postgraduate in your workgroup just released a gradient descent that outperforms the one you wrote some years ago by far. The bad news, however, is that nearly any single application in your lab uses your old algorithm. So the basic benefits of the Liquid ML framework in this situation should be quite clear: All your application automatically use the new algorithm. But now, let's get one step beyond and imagine that your workgroup is interconnected with the algorithm catalogs of many other workgroups … To be quite honest: Personally this picture gives me the creeps.

If you are part of an enterprise, you may know the following situation: Since the incorporation of customer and market information is getting more important, your enterprise extends its analytical tools in market research and decision support by business intelligence software. The Liquid ML framework provides you the best option, not only to minimize the TCO of this software, but also to maintain it state-of-the-art.