Learn-O-Matic: A Fully Automated Machine Learning Suite for Profile Retrieval Applications

Frank Sehnke, Martin D. Felder and Anton K. Kaifel
Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg (ZSW), Industriestr. 6, 70565 Stuttgart, Germany

Abstract

This paper describes the Machine Learning suite Learn-O-Matic. Its key features are that it provides a completely automated framework for supervised learning with an easy-to-use web frontend which executes the complete learning process on NVIDIA based graphic cards. Meta parameters like the network architecture and the regularization term involved are optimised via state-of-the-art Reinforcement Learning techniques. The performance of the function approximator on the test set serves as the reward for the Reinforcement Learner.

We show on data for ozone profile retrieval applications how to use Learn-O-Matic and provide results of the resulting retrieval system and of a wind power forecast system.

1 INTRODUCTION

With the upcoming hyperspectral sounder instruments on geostationary satellites like MTG-IRS, the data rates will soon be too high for classical retrievals based on optimal estimation to be performed in real time. For MTG-IRS the targeted data rate is 2500 spectra/s with about 1740 spectral channels each. Especially for NWP applications near real time processing is paramount. This means that for the necessary retrieval tasks only very fast Machine Learning (ML) based algorithms will be able to perform this challenging job.

But before the retrieval can take place, the ML based systems have to be trained and optimized for channel selection in order to reduce the necessary data rates by means of an automated feature selection [Ng, 2004]. For training in the ML context standard methods like Resilient Propagation (RProp, [Riedmiller and Braun, 1993]) are used, while the meta parameters like size and structure of the model or the amount of regularization is estimated by means of Reinforcement Learning (RL), using recently developed optimisation schemes like Policy Gradients with Parameter Exploration (PGPE) [Sehnke et al., 2010]. This RL approach replaces the expert knowledge by which these values are usually defined.

All these algorithms incur high computational costs however. Graphics Processing Units (GPUs), developed primarily to cope with the high demand of matrix operations needed in rendering 3D sceneries, mostly for computer games, are beginning to see more and more use in scientific tasks [Raina et al., 2009, Ciresan et al., 2011a, Ciresan et al., 2011b, Owens et al., 2008]. This is because of the often similar requirements in both fields: Many simple matrix operations that can be computed in parallel. One of the applications that fits this description is the calculation and training of standard Feed Forward Neural Networks (FFN). We regularly observe computation time reductions by two orders of magnitude. The computational challenges that we see for this problem field can therefore be overcome with GPUs.

This tremendous speed gain for a comparatively low price makes it possible to automatically explore the space of meta parameters by Reinforcement Learning (RL). Here the performance of a converged supervised learner on the test set provides the reward for the RL algorithm. The supervised learning
problem is therefore solved many times with different meta parameters, and the experience thus gathered is harnessed by the RL algorithm to converge on an optimal meta parameter set, and hence on an optimal solution to the supervised task.

Combined with an automated data normalisation and restriction procedure this makes supervised learning tasks accessible to non-experts. Such a system is the Learn-O-Matic described here.

In this paper we present the Learn-O-Matic web application and library for complete automation of machine learning tasks [Sehnke et al., 2012], including spectral channel selection for satellite data retrieval. Besides full automation of ML tasks, key features of Learn-O-Matic are an easy-to-use web frontend for data handling and numerous visualisation modes for all important processing steps executed by the backend. The backend consists of drivers for one or more GPUs which allow processing of different tasks in parallel. The multi-tier approach of Learn-O-Matic imposes no practical limitations on the number of GPUs supported.

Learn-O-Matic supports different neural network architectures like standard multi-layer perceptrons (MLP, [Bishop et al., 2006]), also configured as Deep Neural Networks (DNN) which in turn can be preconditioned with Restricted Boltzmann Machines (RBM, [Hinton et al., 2006]) and trained with different learning algorithms. Gaussian Processes (GP) and Support Vector Regression (SVR) can also be chosen as function approximators.

Besides a technical discussion regarding the application of Learn-O-Matic for satellite data processing, we will compare the performance of the meta-trained MLP with the published results of two real-life problems, combined ozone profile retrieval using GOME-2 and IASI data and wind power prediction using WMO-Stations and GFS data. We highlight the advantages of DNNs over conventional MLPs and of using GPUs over standard computer hardware.

The methods used and our problem dependent adaptations as well as the RL framework generated for meta learning are presented in Section 2 by giving a step by step description how one uses the Learn-O-Matic framework with its own data. In Section 3 we give some examples for the successful use of the Learn-O-Matic. Conclusions and an outlook on future work and the limits of this approach are presented in Section 4.

2 METHODS

2.1 Specifications

The specifications of the software discussed are:

<table>
<thead>
<tr>
<th>System Requirements</th>
<th>Windows XP/Vista/7, Mac OS X, Linux, CUDA enabled graphics card(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Dependencies</td>
<td>Python 2.6 or greater, Matplotlib, NumPy, Cudamat¹, Gnumpy², CUDA³</td>
</tr>
<tr>
<td>Data Preprocessing</td>
<td>AFS, filtering for non-finite data, normalisation, data set generation</td>
</tr>
<tr>
<td>Models and learning methods</td>
<td>Neural Nets, Gaussian Processes, Support Vector Machines</td>
</tr>
<tr>
<td>Additional features</td>
<td>RL meta parameter optimization, validation tools, web interface</td>
</tr>
</tbody>
</table>


2.2 Example Data

We use a synergistic ozone profile retrieval by using METOP data. As input data we use: IASI - IR spectra (8700 channels), GOME-2 - UV/VIS spectra (4096 channels), AVHRR - cloud fraction within GOME-2/IASI FOV and ECMWF - temperature profile data. As output data the NN produces an ozone profile with 61 layers (centered on 0.5 km to 60.5 km). The ozone profile training data or target is generated by ground and satellite based collocations from ozone sonde measurements (WOUDC and SHADOZ), AURA-MLS and ACE-FTS.
2.3 Data Ingestion

The first rule in ML is *Before you care about your algorithm take care about your data*. In this spirit the first part of using LearnOMatic is how to import data into the framework and how it is handled. The data has to be uploaded as a binary file. Also, a so called feature definition file that contains in ASCII format the structure and description of the data has to be provided. This file also specifies which of the available features are to be used as predictors. The data can be uploaded via an upload tab in the web frontend.

2.4 Data Handling and Normalisation

Fig. 1 shows a screenshot of the web frontend. The red tabs at the top of the web page resemble the GPU clients. The green tab marks the CPU client that is recommended for CPU- and memory intensive data preprocessing. The uploaded data is accessible on a specialized high memory CPU client by selecting the corresponding data type like shown in the enlargement. Subsequently the data appears in the source list and can be selected. For the same data package (binary file) several table definition files can be created that use different combinations of input features. By selecting a certain data package a selection for these different feature sets appears like shown in Fig. 2. By selecting an available set the problem definition is finished.

After selecting the data the user can define the settings for data preprocessing. An inexperienced user can just press start to begin data processing with the default settings. ML expert users may choose to divide the output vector into several overlapping parts, set a minimum block size for randomizing the data (needed if the data are e.g. a time series there adjacent data sets are not independent) and choose a
different fraction of the data as test set (or a number of examples as test set if a number > 1 is provided).

The data will then be preprocessed by removing examples with invalid numbers in the inputs, normalising the data between 0 and 1 by using the minimum and maximum value on every data dimension, shuffling the examples, extracting a test set and dividing the data into inputs and targets. The thusly processed data is saved in binary format and can than be selected on every GPU client for learning.

2.5 Setting up Experiments

Now that the data are accessible in their preprocessed form on all GPU clients, the data can be selected by the same procedure as described above.

After selecting the data the user can define the settings for learning. An inexperienced user can again just press the ‘start’ button to start the learning process with the default settings. A more skilled user might like to set some parameters by hand. The user can choose between FFN, GP and SVM as function approximator and for FFN between RProp–, RProp+, BackPercolation and BackPropagation as learning methods. The self adapt checkboxes indicate which meta parameters are to be adapted via RL. For FFN these are the number of hidden layers and the number of neurons per layer for defining the network structure. The strength of the weight decay can be defined that is realised as a L1 regularisation. For GP the meta parameters are the length scale, the signal variance and the noise variance. For SVM regression, because \( \nu \)-SVR is used, the free meta parameters are cost, \( \gamma \) and \( \nu \).

By default all meta parameters are learned by means of RL. The RL algorithm chosen for meta parameter optimization is PGPE ([Sehnke et al., 2010]). PGPE has shown to cope with the noise in the reward signal that is e.g. also present in the final quality of converged FFNs due to the variance and reliability of the initial random weights. This makes an optimization of the meta parameters by a grid search difficult. PGPE is also faster (needs less evaluations) than comparable evolutionary methods that would also be suitable for optimisation here. For GP and SVMs an alternative grid search can be used that is due to their deterministic learning behaviour sometimes faster in finding the optimal (or near optimal) meta parameters. One should keep in mind that learning all meta parameters with RL can become quite time consuming for large data sets especially if the output is generated with multiple structures (parts).

There are also some advanced training algorithm settings that are beyond of the scope of this paper and usually do not have to be modified.
2.6 Evaluation

With evaluation we mean here the process of observing the learning and the possibility to evaluate how good the learning is processing. To this end, the user can observe several statistics. The leftmost plot in Figure 4 tracks the best performance on the test set over the RL – it is always the RMSE normalised by the minimum and maximum data target and stated as % values. The middle plot in the same figure shows the best, the last and some past learning curves (for FFN only) to give some impression of the learning behaviour and the variance between different learning runs. The rightmost table gives some numbers about the convergence in quadratically growing epoch steps like the quality on the training set, the quality on the test set the best quality on the test set in this run, the best quality up to now in all runs and the average quality on the test set at the moment. The values in the last plot are normalised by the normalisation value given by the user. Note that a value of 0 is default indicating the min-max normalisation and that a value of 1 will yield the errors in the original units of the data.

2.7 Validation

After learning has finished, one can validate the learning results on the tab ‘Results’. One can select the original data that is not preprocessed. A validation statistic is then calculated on the complete data. Also the preprocessed data can be chosen, then the training or the test data can be validated. The resulting plot graphically displays the root mean square (RMS) of the weights for every input. This gives the user a hint which inputs are crucial for solving the problem and which inputs might be of less significance for the learning problem. The color code is on a logarithmic scale normalised to the ‘biggest’ RMS-input. In Figure 5 the leftmost plot shows an example. The most dark blue squares are the inputs
Figure 4: While learning is in progress the user can observe certain statistics. The leftmost plot tracks the best performance on the test set over the RL iterations - it is always RMSE% normalised by the minimum and maximum target value. The centre plot shows the best, the last and some past learning curves of RProp to give some impression of the learning behaviour and its variance. The rightmost table provides some numbers about the convergence in quadratically growing epoch steps like the quality on the training set, the quality on the test set the best quality on the test set in this run, the best quality up to now in all runs and the average quality on the test set at the moment. The values in the last plot are normalised by the normalisation value given by the user. Note that a value of 1 will yield errors in the original unit of the data.

that there removed by the feature selection. While yellow and red squares can be seen in the lower GOME-2 bands at the bottom of the plot, in the upper IASI channels at the upper part of the plot and especially for the temperature profiles at the very top of the plot, indicating the importance of this inputs.

The middle plot in Figure 5 shows the performance of the learned structures on the test set (dark green) and the training set (light green).

Figure 5: After learning has finished, one can validate the learning on the tab ‘Results’. By selecting the same data as for the learning 2 to 3 figures are generated showing the relevant results.
We show the results that could be achieved by using the Learn-O-Matic on an ozone retrieval problem and on a wind power prediction problem. In both cases the complete data were several gigabytes in size.

For the ozone data we used ∼200000 collocations with over 12000 inputs and up to 61 targets for each collocation. More information on this problem can be found in [Felder et al., this issue]. The best structure found for this task was a DNN with 3 hidden layers, 384 neurons each and a very small regularization term of $10^{-5}$. The feature selection picked 1664 inputs as relevant. The resulting structure therefore consists of 957312 weights and hence one forward computation (for one collocation) consists of $10^6$ multiplications. This computational load is well within today’s real-time capabilities, even without the use of a GPU. Figure 6 shows the results on the test set. This results are very well competitive to DNN retrievals generated by human experts.

For the wind power problem we used 38000 patterns with over 1000 inputs and 721 targets for each pattern, namely the 15 min averaged wind power for the next 160 hours. The best found structure for this task were 40 DNNs with 2 hidden layers, 256 neurons each and a regularization term of 0.0024 in average. The feature selection selected 600 to 900 inputs as relevant. The resulting nets constitute an overlapping ensemble the average of which gives rise to the final output vector. Figure 7 shows the results on the test set compared to the results obtained by calculating the on-site wind field from the NWP system using atmospheric physics. The final system is competitive with current wind power prediction systems using a much larger array of input data. The planned addition of several other NWPs will surely improve our prediction further.

4 DISCUSSION

In the fields of remote sensing and meteorology we see a variety of applications that involve gigabytes to terabytes of data containing thousands of example patterns that could be used for model learning via supervised learning. These models optimized on these huge amounts of data promise to be fast and reliable alternatives to the common locally optimising retrieval techniques. We have shown with two
examples from this fields that the Learn-O-Matic toolset is an easy to use and generic way to generate these models. We hope we can provide our assistance in this manner also, and more so, in the future.

References


