Self-aware AI Systems

4 year fully funded PhD position at the ADA research group, LIACS, Leiden University, supervised by Holger Hoos and Jan N. van Rijn.

Background

For AI systems to be trustworthy, they need to signal clearly when they “get out of their depth”, i.e., when their output (information, advice, actions) should be treated with caution or becomes entirely unreliable. For example, a configurator can configure a SAT solver to handle a ‘stream’ of SAT instances. However, once the new instances do no longer resemble the instances under which the solver was configured, performance might degrade and ideally the solver should be reconfigured.

Overall Goal

In this project, we will develop methods capable of detecting whether the trained system is no longer up to date, signaling users when this is the case. There is an obvious trade-off between the additional cost of running an outdated system on the new instances, vs the cost to reconfigure the system to better perform on the new instances. Moreover, we aim to identify locations in the meta-feature space where Empirical Performance Models (more general: AutoML systems) work poorly (e.g., due to lack of training instances, ambiguous training instances) and produce instance generators that can fill these gaps.

Work package 1

Empirical Performance Models (EPMs) can be used to assess the expected performance of a given AI algorithm or system on given instances. When the expected performance on new instances decreases, this means that the system is less suitable for these instances, and reconfiguration might yield better results. Techniques from data-stream literature [1, 2] can be generalized to any problem type (TSP, SAT, etc).

In this context, we will look at two types of concept drift:

* Gradually the distribution of the problem instances change in such a way that the fitted model is no longer adequate (concept drift)

* A single instances that does not adhere to the distribution on which the model was trained (outlier detection)

There is a vast body of literature on the topic of concept drift for Machine Learning. This is usually combined with ensembling. The contributions of this work package will lie in the generalization to other problem domains and the inclusion of automated machine learning techniques to update the outdated models.

Work package 2

EPMs are known for their limited ability to generalize to data deviating from the distribution on which they have been trained.

Applied to meta-learning, given an instance, an EPM can be trained on various sub-problems, from moderate to hard, the binary-class problem (which of a subset of two solvers will work best), the multi-class classification problem (which solver will work best) or the regression problem (how well a given solver will work).

It has well been recognized that this line of research has hit a dead-end, and that traditional meta-features do not offer the predictive power to go beyond simple patterns.

Techniques

* Generative Adversarial Networks and boosting techniques (e.g., [5]). By training an auto-encoder to generate specifically instances that the EPM has problems classifying, presumably instances can be generated that (when properly trained) the EPM will improve upon.

* Dataset Generators (see: [3]). By filling the problem space with with yet unseen problems that occupy in places of the search space that are not covered by conventional instances, we can measure when the EPM works and when it doesn't work. Furthermore, we can improve the performance of the EPM by training on such instances. (In case that this technique is used, we should also quantify and overcome a potential limitation that instances with similar features can have different algorithm behavior.)

* Learning Curve prediction. By running a solver on an increasing amount of budget, a learning curve can be generated. Early stages in the learning curve can be used to predict where it is going (see: [6]).

Evaluation / Baselines

* Comparison against vanilla meta-learning systems, that often train a random forest on a set of meta-features.

Work Package 3

Model-agnostic meta-learning (MAML) [4] is (despite the ambiguous name) a form of transfer learning, that aims to train a neural network (or any other gradient based method) with an initial set of weights in such a way that it can be optimized with a low number of gradient steps to a set of train datasets. The assumption is that as long as newly classified datasets resemble the train datasets, the weights can also be adjusted to this dataset within a similar number of gradient steps (effectively with less data).

It is important to be able to detect for a given dataset how close it is to earlier seen datasets. Current references in literature have applied this technique to commonly known related datasets (e.g., CIFAR-10 vs CIFAR-100, various problem domains from the same datasets, etc). However, having access to a larger amount of convolutional and tabular datasets, allows us to test this on a set of potentially more disconnected datasets.

Goal

Detect whether a dataset is closely related to earlier seen datasets, and hence whether a system from the MAML-family can efficiently transfer knowledge from earlier seen datasets.

Techniques

* The techniques developed in “Work package 2” determine whether an EPM is effective in a certain part of the search space, based on instances in the training set. This technique can be generalized towards situations, where a different subset of the training set should be used as the basis for MAML.

Evaluation

* The effectiveness will be compared against a vanilla MAML system, using appropriate statistical methods.

References

[1] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with Drift Detection, 2004.

[2] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, 2007.

[3] M. A. Munoz, L. Villanova, D. Baatar, K. Smith-Miles, Instance Spaces for Machine Learning classification, 2017.

[4] C. Finn, P. Abbeel, S. Levine, Model-Agnostic Meta-Learning for fast adaptation of deep neural networks, 2017.

[5] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, Y. Shoham, Boosting as a metaphor for algorithm design, 2003.

[6] Y. Pushak and H. Hoos, Algorithm Configuration Landscapes: More Benign Than Expected? 2018.