Ph.D. thesis proposal in computer science
2014-2017

An empirical approach to machine learning: algorithm selection, hyperparameter optimization, and automatic principle design

Acronym	EmpML
Title	An empirical approach to machine learning: algorithm selection hyperparameter optimization, and automatic principle design
Co-directors	Michéle Sebag (DR/CNRS), Laboratoire de la Recherche en Informatique Balázs Kégl (DR/CNRS), Laboratoires de l'Accélérateur Linéaire et de la Recherche en Informatique
Research groups	Apprentissage et Optimisation (TAO/LRI)
Mail	michele.sebag@lri.fr, kegl@lri.fr
ED	427 (Informatique)
Name of the head of the ED	Nicole Bidoit
Phone number	01 69 15 63 19
Mail	ed-info@lri.fr

1 Introduction (Summary)

The main paradigm of developing general-purpose machine learning algorithms is trial and error. Of course, the process is *not random*, and the richness of the different motivating principles that guide the process (learning-theoretical, neuroscientific, statistical, algorithmic/computational, just to mention the most important principles) makes machine learning one of the most exciting research areas. Nevertheless, at the end of the day, whether a family of algorithms, a given algorithm, or an algorithmic trick survives is entirely decided by whether it works in practice, where "works in practice" may be defined by "there is a well-defined subset of practical problems on which it is (one of) the best approach(es)". This general definition leaves the playing field wide open for radically different solutions for the same problem as long as there is a wide variety of applications calling for different techniques. For example, depending the number of features and the number of instances in supervised classification, one could choose between quadratic-time kernel methods, linear-time AdaBoost, or L_1-penalized logistic regression. There is no reason to exclude any of the families represented by these algorithms from either the practitioner's toolbox or from research, as long as they answer to practical problems of a well-defined niche.

Measuring whether an algorithm works on a well-defined subset of practical applications should *not* be a problem. There are several reasons why it *is*. First, families of applications have fuzzy borders. Benchmark repositories are usually ad hoc, and there is no generally accepted ontology of problem types (for example, what "large scale" means depends on whether you are in a small research lab or you work for Google or Microsoft Research). What benchmark sets are used to assess an algorithm is essentially part of the research freedom of the researcher who is in charge to "prove" that his/her algorithm works, so selection bias is usually present even in the most carefully designed experimental setups. Second, there is no widely accepted standard methodology to compare methods. In most of the classical designs (e.g., 10-fold cross-validation), significances are usually largely overestimated (Dietterich, 1998; Bengio and Grandvalet, 2004). Publication pressure (it is one of the easiest ways to reject a paper to say that it is not "significantly" better than other approaches) also compels authors to overstate differences. Third, proper hyperparameter optimization is computationally *very* costly and so it is usually done in an ad hoc, non-reproducible way. Even when the proposed algorithm has an open-source turn-key implementation (which is a small minority of the research papers), the experimental design of hyperparameter optimization is rarely described in detail or made available as part of the software package.

From a scientific point of view this mess is not necessarily detrimental: it favors exploration. At the same time, the lack of a rigorous *empirical approach* makes it rather hard for a novice practitioner to find useful advice on what technique he or she should try on a given practical problem. To a certain extent it also *distorts* research: useful algorithms and tricks pop up rather randomly, the exploration of techniques is not guided by the scientific method (that proved to be immensely successful in natural sciences), and there are a lot of "folklore" believes floating around that are not necessarily backed by rigorous experimental results. In this thesis project we propose to apply the scientific method to machine learning. We will explore two lines of research. In the first (Section 2.1) we will build on recent work applying modern *experimental design* for algorithm selection and hyperparameter tuning (Hutter, 2009; Hutter *et al.*, 2009; Bergstra *et al.*, 2011; Bergstra and Bengio, 2012; Snoek *et al.*, 2012; Thornton *et al.*, 2012). The main thrust of this sub-project is the multi-problem approach of Brendel and Schoenauer (2011), Lacoste *et al.* (2012), and Bardenet *et al.* (2013): we will explore the interaction between methods (and hyperparameters) and data sets to find out whether and to what extent experience can be generalized across data sets. The output of this project is a toolbox for practitioners and a stockpile of knowledge on what algorithm works on what (kind of) data sets. This second output will feed into the second line of research: we will ask the question of *why* certain methods work on certain data sets (Section 2.2). There are several principles, mostly derived from theoretical results, that explain the success of methods and guide algorithmic development; our goal is to verify these principles and to discover new ones based on an *empirical* approach. We will study algorithms as natural phenomena, form hypotheses, design and evaluate experiments, and carry out *measurements* that could validate or refute our hypotheses. Whereas the first sub-project is a rather low-risk technical/engineering topic, this second line is open-ended: the student will have a lot of freedom in asking questions and designing computer experiments. The ideal outcome of this line of research is the discovery of new learning principles that can guide algorithmic design.

2 Research themes

2.1 Data-specific algorithm selection and hyperparameter optimization

Hyperparameter tuning is a crucial yet often overlooked step in machine learning practice. Recently, it was shown that the state of the art on image classification benchmarks can be improved by configuring existing techniques better rather than inventing new learning paradigms (Pinto *et al.* 2009, Coates *et al.* 2011, Bergstra *et al.* 2011, Snoek *et al.* 2012, Thornton *et al.* 2012). Hyperparameter tuning is often carried out by hand, progressively refining a grid over the hyperparameter space. Several automatic hyperparameter tuning methods are already available, including local-search based methods (ParamILS of Hutter *et al.* 2009), estimation of distribution methods (REVAC of Nannen and Eiben 2007), and surrogate-based methods Hutter (2009). Recently, Bergstra *et al.* (2011) successfully applied surrogate-based optimization methods to tuning numerous hyperparameters of deep belief networks. The method combined brute force computational power with building a model of the behavior of the cost function (validation error) in the hyperparameter space, and it could significantly improve on manual hyperparameter tuning. In a similar setup, Thornton *et al.* (2012) applied surrogate-based optimization for a large number of classifiers from the WEKA package, outperforming the state of the art on a large set of problems.

What may still make experienced practitioners better at hyperparameter optimization is their ability to generalize *across* similar learning problems. For example, if somebody in the past successfully applied a classification algorithm $\mathcal{A}$ to the popular MNIST dataset with a given set of hyperparameters x, he or she would certainly use this set as a hint (or "prior") to choose the hyperparameters of $\mathcal{A}$ when tuning $\mathcal{A}$ on a slightly noisy or rotated version of MNIST. Bardenet *et al.* (2013) recently formalized this idea and showed that collaborative hyperparameter optimization can outperform single-task optimization.

In this sub-project we will tackle the following problems.

1. COMPUTATIONAL/METHODOLOGICAL ISSUES. Sequential model-based optimization (Jones, 2001; Lizotte, 2008) has proved to be a suitable framework for single-task continuous hyperparameter optimization since it allows a neat Bayesian treatment of prior knowledge and because it handles the exploration/exploitation dilemma efficiently. It has also been the method of choice of Bardenet *et al.* (2013) for the same reason: by embedding data sets into a continuous meta-parameter space and replacing the regression setup with a ranking-based approach, generalization *across data sets* can be handled in the same way as generalization on a single set across hyperparameters of the algorithm. However, there are two problems with this setup. First,

it is based on Gaussian process (GP) regression, and so it will not scale. Second, there are several interesting cases when the optimization has to be carried out in a non-continuous, sometimes hierarchically structured spaces (e.g., algorithm *selection* is discrete, or adding a new layer in a neural network adds a fresh set of new hyperparameters to be optimized, making the space of variable dimension). Exploring algorithmic solutions for this problem will be the first step of this sub-project.

Another interesting question is how to combine this experimental design setup with sub-sampling. It would be clearly advantageous to first rapidly explore the error surface using small samples than concentrate on only promising hyperparameters or algorithms. Phase transitions (abrupt changes of the optimum as the data size grows) make this idea difficult to implement, nevertheless, there are some interesting techniques to explore (Hoeffding races of Maron and Moore 1994 and Domingos and Hulten 2001; sequential testing of Krueger *et al.* 2012; scalable bootstrap of Kleiner *et al.* 2011).

2. MULTI-PROBLEM OPTIMIZATION. Bardenet *et al.* (2013) paved the road for collaborative hyperparameter optimization, but there remain several open problems. Perhaps the most important one is the issue of meta-features. Bardenet *et al.* (2013) used simple descriptors (number of instances, features, classes; intrinsic dimensionality) and found some light correlation between, e.g., complexity parameters (such as the number of leaves in trees) and the sample size/dimensionality ratio, which could be exploited for multi-problem inference. There is obviously plenty of space for designing easy-to-compute statistics which better predict optimal algorithms and hyperparameters. Finding these descriptors are closely related to *understanding* why certain algorithms work on certain kinds of problems, so this sub-project can be iterated with the theme described in Section 2.2.

   Collaborative algorithm selection and optimization closely resembles collaborative filtering (with "movies" replaced by algorithms or parameters, and "users" replaced by data sets), and so a second line of research will be to explore the possibility to adapt collaborative filtering techniques for this problem. Finally, exploring the obvious connections to transfer learning and multi-task learning (Caruana, 1997; Niculescu-Mizil and Caruana, 2005a) will also be part of the thesis.

2.2 An empirical approach to machine learning

In this theme we will ask the question of *why* certain algorithms work on certain kinds of data sets. The empirical basis of the study will be accumulated using the tool developed in Section 2.1. We will follow an empirical methodology by treating algorithms as natural phenomena, observing their actual behavior on real data, designing observables, carrying out quantitative measurements, and inferring general principles.

One generic sub-theme is connecting (training) margin distributions to classifier (generalization) quality. The large margin principle, at the root of several algorithms and justifications (Boser *et al.*, 1992; Bartlett, 1998; Schapire *et al.*, 1998) basically says that at the same complexity, larger training margins mean smaller test errors. While it is true (and widely accepted) in general, using the principle quantitatively in practice is rather difficult, essentially for two reasons. First, measuring the complexity of a classifier is rather difficult. Complexity measures, even effective/constructive measures, appearing in theoretical bounds are assigned to function sets rather than to individual functions. Defining a function set of which a given classifier comes from is inherently ambiguous. In the famous debate on why AdaBoost works (variance reduction vs. large margin principle), one of the most interesting results is by Reyzin and Schapire (2006): they show that Breiman (1999)'s arc-gv (an aggressive minimum-margin maximizer), while achieving a larger minimum margin, uses deeper trees than AdaBoost, even when the number of nodes in the trees (and of course, the number of iterations) are the same in the two algorithms. They then argue that deeper trees are more complex, and so arc-gv's overfitting can be explained by the larger (more complex) function class from which it effectively chooses its classifier. The second problem is that margin distributions (CDFs) cross each other. When decision stumps are used as base classifiers (so the number of iterations arguably determines the complexity), Reyzin and Schapire (2006) find that while arc-gv still achieves a larger *minimum* margin than AdaBoost, its *mean* margin is smaller. When margin distributions cross each other, the large margin principle, strictly speaking, does not postulate anything.

There is now a whole line of research (Shen and Li, 2010; Laviolette *et al.*, 2011; Shivaswamy and Jebara, 2011) which, instead of maximizing the mean margin or the minimum margin, or defining a convex margin-based loss and hope for the best, tries to control the margin distribution more in detail. The main idea is to maximize the mean margin while also minimizing (or controling) the *variance*. We also know that by calibrating the raw scores produced by AdaBoost (essentially "playing" with the margin distribution), we can significantly improve its performance under probabilistic measures (Niculescu-Mizil and Caruana, 2005b; Caruana and Niculescu-Mizil, 2006). All this research indicates that generalization is related to some intricate properties to the training margin distribution. The goal of this sub-project is to pinpoint these properties by searching through various statistics, and finding those that correlate with the generalization error.

Besides this concrete problem, the theme is open for the student to ask and test other empirically testable questions. The non-exhaustive list may include "Why isn't AdaBoost overfitting (as the number of iterations goes to infinity)?", "why huge self-tuning ensembles outperform tuned but single models (Niculescu-Mizil, A. et al., 2009; Busa-Fekete *et al.*, 2011, 2013)?", or "why the seemingly "simple" covertype data set prefers to boost huge trees while the (tuned) choice of other UCI sets is rarely more than 20 leaves?".

References

Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. (2013). Collaborative hyperparameter tuning. In *International Conference on Machine Learning (ICML)*.

Bartlett, P. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, **44**(2), 525–536.

Bengio, Y. and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, **5**, 1089–1105.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*.

Bergstra, J., Bardenet, R., Kégl, B., and Bengio, Y. (2011). Algorithms for hyperparameter optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24. The MIT Press.

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

Breiman, L. (1999). Prediction games and arcing classifiers. *Neural Computation*, **11**, 1493–1518.

Brendel, M. and Schoenauer, M. (2011). Instance-based parameter tuning for evolutionary AI planning. In *Proceedings of the 20th Genetic and Evolutionary Computation Conference*.

Busa-Fekete, R., Kégl, B., Éltető, T., and Szarvas, G. (2011). Ranking by calibrated AdaBoost. In *(JMLR W&CP)*, volume 14, pages 37–48.

Busa-Fekete, R., Kégl, B., Éltető, T., and Szarvas, G. (2013). Tune and mix: learning to rank using ensembles of calibrated multi-class classifiers. *Machine Learning Journal*, **93**(2), 261–292.

Caruana, R. (1997). Multitask learning. In *Machine Learning*, pages 41–75.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168.

Coates, A., Lee, H., and Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, **10**(7), 1895–1923.

Domingos, P. and Hulten, G. (2001). A general method for scaling up machine learning algorithms and its application to clustering. In *International Conference on Machine Learning*, pages 106—113.

Hutter, F. (2009). *Automated Configuration of Algorithms for Solving Hard Computational Problems*. Ph.D. thesis, University of British Columbia.

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. (2009). ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, **36**, 267–306.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, **21**, 345–383.

Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. (2011). A scalable bootstrap for massive data. Technical report, `http://arxiv.org/abs/1112.5016`.

Krueger, T., Panknin, D., and Braun, M. (2012). Fast cross-validation via sequential testing. Technical report, `http://arxiv.org/abs/1206.2248`.

Lacoste, A., Laviolette, F., and Marchand, M. (2012). Bayesian comparison of machine learning algorithms on single and multiple datasets. In *15th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Laviolette, F., Marchand, M., and Roy, J.-F. (2011). From PAC-Bayes bounds to quadratic programs for majority votes. In *International Conference on Machine Learning*.

Lizotte, D. (2008). *Practical Bayesian Optimization*. Ph.D. thesis, University of Alberta.

Maron, O. and Moore, A. W. (1994). Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6*, pages 59–66.

Nannen, V. and Eiben, A. E. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 975–980.

Niculescu-Mizil, A. and Caruana, R. (2005a). Learning the structure of related tasks. In *In Proceedings of NIPS-2005 Workshop on Inductive Transfer: 10 Years Later*.

Niculescu-Mizil, A. and Caruana, R. (2005b). Obtaining calibrated probabilities from boosting. In *Proceedings of the 21st International Conference on Uncertainty in Artificial Intelligence*, pages 413–420.

Niculescu-Mizil, A. et al. (2009). Winning the KDD Cup Orange Challenge with ensemble selection. In *KDDCup 2009 (JMLR W&CP)*, pages 1–16.

Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Computational Biology*, **5**(11).

Reyzin, L. and Schapire, R. E. (2006). How boosting the margin can also boost classifier complexity. In *International Conference on Machine Learning*, pages 753–760.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, **26**(5), 1651–1686.

Shen, C. and Li, H. (2010). Boosting through optimization of margin distributions. *IEEE Transactions on Neural Networks*, **21**(7).

Shivaswamy, P. and Jebara, T. (2011). Variance penalizing AdaBoost. In *Neural Information Processing Systems*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Auto-WEKA: Automated selection and hyper-parameter optimization of classification algorithms. Technical report, `http://arxiv.org/abs/1208.3719`.