Cognitive Automation of Data Science

Horst Samulowitz SAMULOWITZ@US.IBM.COM
Chandra Reddy CREDDY@US.IBM.COM
IBM TJ Watson Research Center, Yorktown Heights, NY, USA

Ashish Sabharwal AshishS@ALLENAI.ORG
Allen Institute for Artificial Intelligence (AI2), Seattle, WA, USA

Abstract

This paper explores how an automated procedure may leverage domain knowledge and reasoning to further automate Machine Learning (ML) and Data Science in a manner that may be thought of as cognitive. To this end, we first describe key features that we believe a cognitive automation system for data science must possess. The goal of a system embodying this concept would be to extend existing data-driven approaches by incorporating knowledge from experts as well as unstructured data, and performing inference on the knowledge. It would include basic concepts such as reasoning based on realizations (such as overfitting) during the configuration process that results in the system performing corrective actions driven by knowledge of the underlying analytics tool. Furthermore, the system would directly incorporate end-user constraints (e.g., the wish for explainable decisions) in order to guide the learning process. While knowledge can be directly contributed by experts (e.g., known best practices in data science), the system would also extract relevant knowledge from unstructured data by employing DeepQA systems (e.g., querying Wikipedia pages) and through interactions with the user in order to support recent developments in data science and active user guidance. Finally, in the spirit of IBM Blue Chef, the system would bring the notion of creativity to automated ML by composing novel variations of existing ML techniques. The present paper discusses the main features of and challenges in building such a system.

Keywords: Meta-Learning, Cognitive Systems, Unstructured Data, Knowledge Representation, User-Interaction

1. Introduction

The expression "Big Data" is commonly used to describe not only the fact that an immense amount of data is being collected from various sources (e.g., internet and social media users, mobile phones, surveillance sensors), but also the challenging task of actually analyzing it. To this end, a wide variety of approaches have been proposed in the active research areas of Machine Learning and Statistics, ranging from algorithms for classification to time-series analysis. For instance, one recent celebrated approach for both unsupervised and supervised learning called Deep Learning is able to automatically extract complex patterns from Terabytes of data in application areas such as Natural Language Processing (NLP) and image analysis. However, while these advances improve our ability to make sense of large scale datasets, no single approach is applicable to or successful in most of the scenarios occurring commonly in data analytics. For instance, the data requirements imposed by deep learning stem from the availability of large amounts of data in specific application areas but can be too restrictive in other domains (e.g., when data is highly non-linear). Besides different "shapes" of data, the underlying objective of the data analytics process drives the usability of as well as preference for different approaches. A common example is that end-users of a data analytics tool may want to understand, at least to some extent, the reasoning process of the tool and may thus favor traditional approaches such as low-depth Decision Trees over deep Neural Networks, Support Vector Machines (SVMs), or even Random Forests.

A range of tools are deployed in the processing pipeline needed to analyze big data. Roughly speaking, the tool chain starts with algorithms for cleaning and processing data, and concludes with presenting the predictions in a way that aligns with the end-users' objective(s) and preferences.

In recent years various data-drive meta-approaches, under the umbrella term *algorithm portfolios*, have been introduced to automatically select (at runtime) from a collection of base algorithms the expected best match for a given problem instance. For example, in the context of solving problems posed in propositional logic (SAT) and constraint reasoning (CSP), a range of ML methods such as Linear Regression, Decision Trees, k-Nearest Neighbor, and Collaborative Filtering have been used to automatically select the most promising solver for a given problem instance at hand (cf. Hutter et al., 2009; Stern et al., 2010; Xu et al., 2008; Malitsky et al., 2013). Similarly, some approaches use ML methods in order to change algorithms on-the-fly in an instance-specific way, such as by changing heuristic settings online or adapting to data flow when evaluating a complex database query. Furthermore, within ML, approaches such as Auto-Weka (Thornton et al., 2013) attempt to exploit the variation in performance across different clustering or classification approaches or across different parameter settings of an algorithm (e.g., number of units in various layers of a deep neural network).

These meta-approaches, although derived independently for a variety of application domains, are all based on: a) characterizing problem instances by features and b) selecting the most promising approach for a problem instance at hand based solely on observed outcomes on training data.

While these approaches have achieved outstanding results such as winning international competitions (cf. Järvisalo et al., 2011) and making complex algorithmic tools more accessible to non-experts, existing methods do not explicitly leverage semantic algorithmic or model information, external expert knowledge, or end-user constraints and objectives when driving the automated analytics process. In fact, most existing approaches are purely data-driven and traverse the configuration search space in a way that completely ignores higher-level knowledge and insights that a human expert might have. In the most basic case, this means that one simply tries various values of numerical parameters (e.g., learning rate) or categorical parameters (e.g., use pre-processing or not), hoping to do better than brute force enumeration of the entire search space. While successful, this approach operates at the lowest possible semantic level, i.e., at the level of the base algorithm IDs or their core parameters, treating the algorithms themselves as complete *black-boxes*. For instance, recently proposed systems like Auto-Weka for auto-selection of ML methods work in the space of about 700 categorical and numeric parameters, trying to select their best configuration by mainly considering achieved output value and past evaluation results.

2. Cognitive Automation

We begin by defining what we mean by *cognitive* in the context of an ML automation framework. While we find the proposed properties to be critical to such a system, one could arguably relax some of these or expand the notion to include additional desirables.

Definition 1 *An algorithmic framework will be called* ***cognitive*** *if it has the following properties:*

1. *it integrates knowledge from (a) various structured or unstructured sources, (b) past experience, and (c) current state, in order to reason with this knowledge as well as to adapt over time;*
2. *it interacts with the user (e.g., by natural language or visualization) and reasons based on such interactions; and*
3. *it can generate novel hypotheses and capabilities, and test their effectiveness.*

While the first part of this characterization is embodied to some extent by existing frameworks such as Auto-Weka, the next two aspects are not.

2.1. Incorporating Domain Knowledge

In our context, we would like a cognitive automated data analytics system to "understand" and exploit knowledge of the following nature:

- *Basic properties and techniques* underlying the base analytics algorithms it can use and compose, such as input requirements, theoretical guarantees, key concepts pertinent to that algorithm (e.g., distance function in k-NN), properties of the output (e.g., with or without correctness confidence), computational properties such as runtime, applicability such as online vs. offline learning, support for missing data, etc.
- *Domain knowledge* from experts (e.g., common practices), structured and unstructured data (e.g., corpus of WEKA manual, ML books and papers), interaction with the user, basic statistical knowledge (e.g., overfitting detection), and insights from data-driven evaluations and past experience.
- The overall *objective of the end-user*, *constraints* from the end-user or the problem domain such as interpretability of results, fast and/or adaptive decision making, streaming vs. batch data, etc.
- For various base algorithms, *how to react* to observations or realizations made during the automation process, such as high classification error and overfitting.

2.2. Exploiting Domain Knowledge

The following examples illustrate how the system could successfully leverage such knowledge in order to improve the decision making process not only for what base analytics algorithms to deploy and in which configuration, but also to automatically create novel methods.

Example 1, Reasoning (Overfitting): Consider a model based on k-nearest neighbor (k-NN). For a classification or regression model, one can automatically detect overfitting (Hastie et al., 2001). Based on this observation and a semantic understanding of how k-NN works, the system could try to systematically increase k rather than uniformly searching over a different value of k (as if k-NN was a black-box). Similarly, when configuring neural networks, it could try to reduce the number of hidden layers in response to overfitting.
Example 2, Reasoning (End-User Constraint): The end-user may prefer as succinct an explanation of a recommendation as possible. If the automated process is aware of this objective and understands that certain methods are more suitable for it than others, it could directly start employing methods such as decision trees or rules and then perform finer grained adjustments such as reducing the depth of the decision tree.
Example 3, Reasoning (Modeling): Multi-Class Classification can be achieved in different ways. When the number of classes is low, one can often successfully combine binary classifiers to perform the task. However, when the number of classes is high this approach can become less practical (e.g., extremely high training time). Simply being aware of this relationship could help the automated process make a better decision.
Example 4, Deep QA (Novel Knowledge): Suppose one wanted to use k-NN on a data set with discrete features. The system could search for "distance measure knn discrete data" and extract the knowledge stated on a Wikipedia page: "A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance)." It could thus "know" and exploit this knowledge to configure k-NN appropriately.
Example 5, Interactivity (Suggestion): In the above example, suppose only the Euclidean distance metric is provided by the user. The automated system should be able to exploit the Wikipedia knowledge and interactively suggest to the user that a Hamming distance metric may be more suitable.
Example 6, Creativity (Regularization): In deep learning one recently introduced technique called "DropOut" randomly selects subsets of node activations in each layer and sets them to zero (Hinton et al., 2012). Subsequently, a technique called "DropConnect" (Wan et al., 2013) was introduced that instead sets a randomly selected subset of edge weights (or connections) within the network to zero.[1] A cognitive automated system for ML could possibly automatically explore such novel extensions as DropConnect given knowledge about DropOut, not to mention a combination of the two approaches. Exploring certain simple combinations and variations of existing techniques in order to develop new creations would be in the spirit of IBM Blue Chef (Varshney et al., 2013).

Although the above examples focus mainly on modeling, the ideas could be extended to the entire tool chain used in data analytics (e.g., data cleaning, outlier detection, etc.).

3. Challenges in Cognitive Automation

In order to design a system that automatically guides and develops the decision process in ways exemplified above, one must figure out: (a) how to represent such knowledge; (b) how to acquire such knowledge and how to improve the internal representation as new knowledge

1. This is a very simplified presentation of these two techniques, meant to illustrate the idea of creativity.

arrives; and (c) how to perform inference on this knowledge in order to guide the automated process and create novel techniques.

Representation of Knowledge: While some part of the knowledge can be cast as a rule-based system (e.g., "if k-NN is being used and overfitting is observed, then increase k") in combination with a standard knowledge representation system (e.g., knowledge graph (Eder, 2012)), it will not be sufficient to capture and facilitate several key concepts. For instance, in k-NN one does not only want to "know" that the distance measure plays a critical role, is related to overfitting, and impacts accuracy, but also that it is based on geometric distances in the feature space and that one can potentially "look for" another distance measure. Each algorithm will need to be appropriately described using "meta-knowledge" in order to capture key properties in a way that provides the system sufficient "meaning" to enable it to add to or alter existing methods. Furthermore, most knowledge in this context will be of probabilistic nature. For example, while more pruning in a decision tree is very likely to reduce overfitting, there is no absolute guarantee that this will succeed. A major challenge is to create a representation that is a) machine-readable, b) captures key properties of algorithms, and c) facilitates inference over this knowledge.
Knowledge Acquisition: Here one can distinguish between multiple sources of knowledge. Manual: One can exploit expert knowledge by adding it to the system manually. This could be known "best practices" such as employing bootstrap sampling to ensure model accuracy. Automatic: The goal here is to automatically extract expert knowledge from unstructured data using NLP/DeepQA techniques. E.g., the phrase "main drawbacks of decision trees" in a search engine returns a paper that lists the following drawbacks: "reduced performance when the training set is small" and "rigid decision criteria." Clearly, it is not straightforward to make an algorithm understand these statements and act upon them. However, since the applicability of new knowledge is somewhat restricted by the existing specification of the algorithmic components (e.g., what a "measure" means for k-NN), it makes the understanding process limited but, at the same time, computationally more feasible. While manually adding best practices for each analytics tool may be simpler, the automatic way of acquiring knowledge is more amenable to incorporating emerging technologies. Even if it manages to do so only in an offline fashion (i.e., without integrating novel techniques automatically in an algorithmic fashion), it could provide advice to the data scientist, similar to the role of IBM's Watson applied to health care where it assists medical doctors with, for example, newest results from medical studies.
End-User Constraints: User constraints such as strict training/testing runtime requirements or interpretability of results should trigger rules that can potentially prune large chunks of the configuration space. For instance, having a tight training time budget might eliminate multi-class classification via binary classifiers or the training of deep neural networks.

Introspection: *Knowledge Learning:* In every application of the automated process, specific pieces of knowledge (e.g., rules) are applied to guide it. While this provides explanations to the user who can potentially leverage this information further, the system itself can learn from the decisions made by it and the outcome achieved. For instance, it might be able to deduce new knowledge that reduces the number of user queries in the future.
Online Adaptation: Over time the configuration process might have seen various data sets and tried many different approaches. It can try to generalize this knowledge and exploit it

to improve future decision making in an online fashion. One could, for instance, start with the definition of features that characterize data sets (e.g., number of data points, feature types, variance in feature values, etc.) in order to leverage past experience. This can also include end-user preference learning (e.g., realization that end-user in a certain domain prefers compact and interpretable models and in another domain does not care so much).

Inference: *Reasoning:* Probabilistic logic inference (such as with probabilistic graphical models (Koller and Friedman, 2009), Markov Logic Networks (Richardson and Domingos, 2006), etc.) on the underlying knowledge base is likely to be the best match for the context, since the process is guided by rules of thumb as well as uncertain knowledge about what approaches are most likely to succeed on previously unseen data sets or objectives.
Creativity: Besides guiding the analytics process using provided knowledge, one also wants the system to infer novel concepts by combining various pieces of provided knowledge and extrapolating from them. As suggested in Example 6, it may be possible to represent such techniques for neural networks in a way that one could perform automated inference over them in order to explore new variations of these techniques to analyze data. One other option to creativity arises in the context of hybrid learning methods such as combining statistical methods with neural networks for image analysis. For instance, the system could try to use a decision tree to divide the training instances and then perform predictions using k-NN on these reduced sets of instances. When considering the entire tool chain in data science, there is large combinatorial space for generating novel combinations of tools. While perhaps somewhat far-fetched, this would, at least in spirit, be similar to the technology behind IBM's Blue Chef mentioned earlier, which combines various ingredients and end-user preferences to come up with a novel yet "edible" recipe.

4. Closing Remarks

While the proposed framework involves various sophisticated (and some currently not well-understood) techniques including reasoning and optimization over knowledge acquired from unstructured data and interaction with a user, it can be developed in stages. One can start off with a system that is relatively close to the purely data-driven "black-box" approach but has some basic semantic knowledge. For instance, with self-awareness of overfitting, one can represent rules that appropriately cause the process to trigger the right amount of pruning when learning a decision tree model or to adapt k in k-NN appropriately. After manually adding some basic rules, one needs to define ways to represent key properties of the algorithms so that the automated process is able to reason with them and eventually modify them based on its own experience. These first steps are in fact related to recent work that aims at "building an automatic statistician" (Lloyd et al., 2014). Subsequently one can try to query external sources of knowledge in a focused way using existing and emerging NLP technologies (e.g., extracting knowledge from Wikipedia pages), and integrate this knowledge via interaction with a user. The most ambitious concept of creativity can perhaps be dealt with last, but considering it already during the design of the system may have benefits.

References

J.S. Eder. Knowledge graph based search system, 2012. URL https://www.google.com/patents/US20120158633. US Patent App. 13/404,109.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stutzle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

Matti Järvisalo, Daniel Le Berre, and Olivier Roussel. SAT competition, 2011.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193, 9780262013192.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and Natural-Language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.

Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm portfolios based on cost-sensitive hierarchical clustering. In *The 23rd International Joint Conference on Artificial Intelligence, IJCAI*, pages 608–614, 2013.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006. ISSN 0885-6125.

D. Stern, H. Samulowitz, R. Herbrich, T. Graepel, L. Pulina, and A. Tacchella. Collaborative expert portfolio management. In *AAAI 2010, Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010*, pages 210–216, 2010.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD-2013*, pages 847–855, 2013.

Lav R. Varshney, Florian Pinel, Kush R. Varshney, Angela Schorgendorfer, and Yi-Min Chee. Cognition as a part of computational creativity. In *12th IEEE International Conference Cognitive Informatiatics and Cognitive Computing, ICCI*CC 2013*, page 3643, 2013.

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1058–1066. JMLR.org, 2013.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, June 2008.