Informed Machine Learning Through Functional Composition

C. Bauckhage, C. Ojeda, J. Schücker, R. Sifa, and S. Wrobel

Fraunhofer Center for Machine Learning, Sankt Augustin, Germany
Fraunhofer IAIS, Sankt Augustin, Germany
B-IT, University of Bonn, Bonn, Germany

Abstract. Addressing general problems with applied machine learning, we sketch an approach towards informed learning. The general idea is to treat data driven learning not as a parameter estimation problem but as a problem of sequencing predefined operations. We show by means of an example that this allows for incorporating expert knowledge and leads to traceable or explainable decision making systems.

1 Introduction

Recent progress in artificial intelligence has been largely driven by deep neural networks [1] which are trained using vast amounts of data and high performance hardware. This way, deep networks can now solve demanding tasks in computer vision [2], speech recognition [3], text understanding [4], medical diagnosis [5], or game AI [6]. Nevertheless, data science practitioners outside the IT industry are increasingly dissatisfied with deep learning, mainly because of:

1) Lack of training data. Vapnik-Chervonenkis theory [7] establishes that it takes substantial training data for machine learning to work well. Since details depend on a system's VC dimension which is hard to pinpoint, Widrow's rule of thumb (see [8]) suggests to train with at least ten times more data than there are system parameters. Modern deep networks with millions of parameters therefore need tens if not hundreds of millions of representative examples to learn robustly.

At first sight, this does not seem problematic, after all this is the age of *big data*. However, reliable supervised machine learning actually requires *thick data*, i.e. large amounts of annotated data, which are rarely available where business models do not revolve around digital services. Even in contexts such as Internet of things or industry 4.0 where data accumulate en masse, we still frequently face *thin data* scenarios where appropriate training data for machine learning are inaccessible.

2) Lack of traceability. Trained connectionist architectures are black boxes whose inner computations (non-linear activations of synaptic summations) are abstracted away from conceptual information processing. As a consequence, their decision making processes are often unaccountable so that data scientists in, say, finance or industrial system control are wary of deep learning because regulatory guidelines demand automated decision making to be comprehensible.

Indeed, a growing body of literature shows that silly mistakes made by deep networks could be avoided if they had "common sense" or, even more alarmingly, that silly mistakes can be provoked using adversarial inputs [9, 10]. Hence, while the need for explainable AI has been recognized early on [11, 12] corresponding research has been reinvigorated recently [13–16].

In this paper, we address both these issues and propose an approach towards *informed machine learning* that copes with thin data and leads to traceable systems. Our basic observation is that domain experts in industry know a lot about the data they are dealing with and that this knowledge is often procedural, i.e. there is experience as to what to do with data in order to achieve a certain goal. Assuming training data and a database of procedures to be given, our main idea is thus to use Monte Carlo tree search [17] or reinforcement learning [18] to determine sequences of operations that map input to desired outputs.

2 Deep Learning as Functional Composition

To motivate our idea, we point out two facts about neural networks: First, a trained, feed-forward neural network of L layers that maps inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$ computes a composite function

$$\boldsymbol{y}(\boldsymbol{x}) = f_L\Big(\ldots f_2\big(f_1(\boldsymbol{x})\big)\Big) = f_L \circ \ldots f_2 \circ f_1(\boldsymbol{x}) \tag{1}$$

where the functions f_l represent the computations of layer l of the network. To be more precise, letting $\boldsymbol{z}_l$ denote the output of layer l, we typically have

$$\boldsymbol{z}_l = f_l\big(\boldsymbol{z}_{l-1}\big) = \sigma\big(\boldsymbol{W}_l\, \boldsymbol{z}_{l-1}\big) \tag{2}$$

where $\boldsymbol{W}_l$ are the input weights of layer l and $\sigma(\cdot)$ is an activation function.

Second, recall that number and sizes of layers as well as activation functions are usually chosen by hand so that neural network training is to estimate those weight parameters $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L$ that minimize a loss function such as

$$E = \sum_{i=1}^{n} \big\|\boldsymbol{y}_i - \boldsymbol{y}(\boldsymbol{x}_i)\big\|^2 \tag{3}$$

where $\big\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\big\}_{i=1}^{n}$ is a sample of labeled training data.

Given these observations regarding the design and training of deep architectures, it then seems natural to ask: **Q1:** When building a deep system, do the functions f_l it consists of have to be of the form in (2) or are there other basic building blocks? **Q2:** When training a deep system, does it necessarily have to be a parameter estimation problem or are there other ways of adapting it to a task at hand?

Especially the soft computing community has answered both these question affirmatively before. For instance, recent work in [19] present a deep fuzzy logic system assembled via genetic programming that shows superhuman capabilities

Table 1: Functional building blocks for a classifier for the problem in Fig. 1(a)

$\mathbb{R}^2 \to \mathbb{R}^2$	$\mathbb{R}^2 \to \mathbb{R}$	$\mathbb{R} \to \mathbb{R}$
$f_0(\boldsymbol{x}) = \mathrm{id}(\boldsymbol{x})$	$f_6(\boldsymbol{x}) = \sum_i x_i$	$f_{11}(x) = 2\,x$
$f_1(\boldsymbol{x}) = \boldsymbol{R}\boldsymbol{x}$	$f_7(\boldsymbol{x}) = \prod_i x_i$	$f_{12}(x) = -x$
$f_2(\boldsymbol{x}) = \big(\boldsymbol{w}_a^T \boldsymbol{x}\big) \cdot \boldsymbol{w}_a$	$f_8(\boldsymbol{x}) = \sum_i x_i^2$	$f_{13}(x) = x - 1$
$f_2(\boldsymbol{x}) = \big(\boldsymbol{w}_b^T \boldsymbol{x}\big) \cdot \boldsymbol{w}_b$	$f_9(\boldsymbol{x}) = \boldsymbol{w}_a^T \boldsymbol{x}$	$f_{14}(x) = x + 1$
$f_4(\boldsymbol{x}) = \big(\boldsymbol{w}_a \boldsymbol{w}_a^T - \boldsymbol{I}\big)\boldsymbol{x}$	$f_{10}(\boldsymbol{x}) = \boldsymbol{w}_b^T \boldsymbol{x}$	$f_{15}(x) = \lvert x \rvert$
$f_5(\boldsymbol{x}) = \big(\boldsymbol{w}_b \boldsymbol{w}_b^T - \boldsymbol{I}\big)\boldsymbol{x}$		$f_{16}(x) = \mathrm{sign}(x)$
		$f_{17}(x) = \tanh(x)$
		$f_{18}(x) = e^{-\frac{1}{2}x^2}$

$$\text{where } \boldsymbol{R} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \boldsymbol{w}_a = \begin{bmatrix} \sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix} \qquad \boldsymbol{w}_b = \begin{bmatrix} -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} \end{bmatrix}$$

in an air defense scenario. Here, however, we are interested in answers based on statistical machine learning and illustrate our idea by means of a simple example.

Figure 1(a) shows labeled training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where the data $\boldsymbol{x}_i \in \mathbb{R}^2$ come from two classes and the labels $y_i \in \{+1, -1\}$ indicate class membership. Given this XOR-type data, the goal is to learn a binary classifier $y(\boldsymbol{x})$ such that

$$y(\boldsymbol{x}) = \begin{cases} -1 & \text{if } x_1 \approx x_2 \\ +1 & \text{otherwiese.} \end{cases} \tag{4}$$

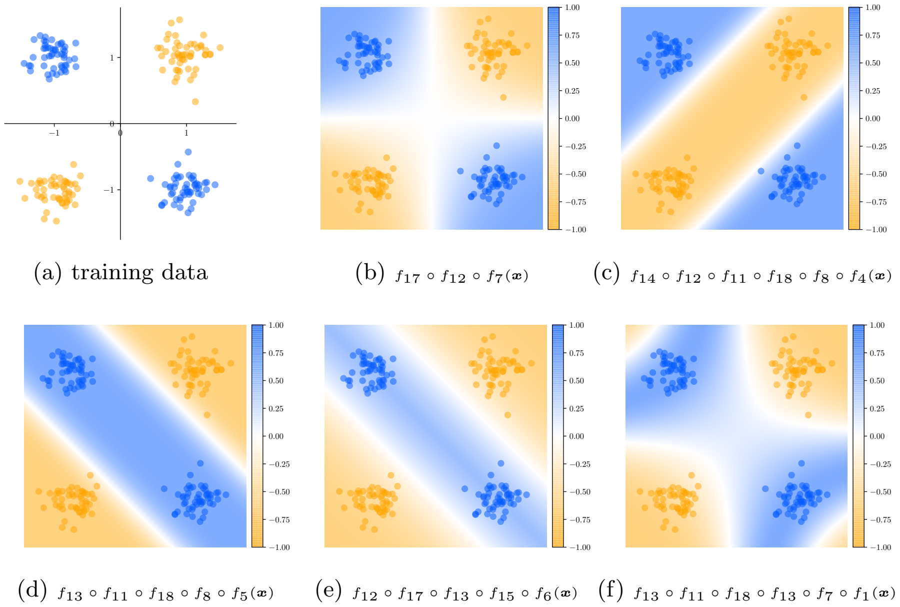
Table 1 shows a set of functions $F = \{f_1, f_2, \ldots, f_n\}$ compiled by a domain expert who —based on long term experience— supposes that there might be a sequence of operations $y(\boldsymbol{x}) = f_{i_T} \circ \ldots \circ f_{i_2} \circ f_{i_1}(\boldsymbol{x})$ that can solve the above problem.

In order to determine such a sequence automatically, we let $s_0 = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ be the initial state of an abstract state space S and search for a sequence of states $s' = f(s) = [f(s_1), \ldots, f(s_n)]$ that eventually leads (close) to the target state $s_T = [y_1, \ldots, y_n]$ where closeness to the target can, for instance, be measured in terms of 0-1 loss or accuracy.

Figures 1(b)–(f) show different examples of sequences that come close to the target state and could be considered solutions to our problem.

In the language of game AI, our problem is to determine an optimal sequence of moves $\mu^* : S \times F \to S$ and, in the language of reinforcement learning, it is to find an optimal policy $\pi^* : S \to F$. Both problems can be solved using Monte Carlo tree search or Q-learning, respectively, and in both cases we may consider the following reward function

$$r\big(f(s)\big) = \begin{cases} \mathrm{accuracy}\big(f(s)\big) & \text{if } f \in \mathbb{R} \to \mathbb{R} \\ -0.1 & \text{otherwise.} \end{cases} \tag{5}$$



(a) training data

(b) $f_{17} \circ f_{12} \circ f_7(\boldsymbol{x})$

(c) $f_{14} \circ f_{12} \circ f_{11} \circ f_{18} \circ f_8 \circ f_4(\boldsymbol{x})$

(d) $f_{13} \circ f_{11} \circ f_{18} \circ f_8 \circ f_5(\boldsymbol{x})$

(e) $f_{12} \circ f_{17} \circ f_{13} \circ f_{15} \circ f_6(\boldsymbol{x})$

(f) $f_{13} \circ f_{11} \circ f_{18} \circ f_{13} \circ f_7 \circ f_1(\boldsymbol{x})$

Fig. 1: XOR problem and classifiers composed of the building blocks in Tab. 1.

Figure 2 shows a solution found through Monte Carlo tree search. It achieves perfect accuracy on the given training data but also generalizes well.

Moreover, the classifier found this way it traceable in the sense that its internal computations are transparent to domain experts. Here, for instance, the 2D data are first rotated clockwise by 90°, then the two components of each data point are multiplied, and finally the sign of this product is determined and returned as the output of this "deep" system.

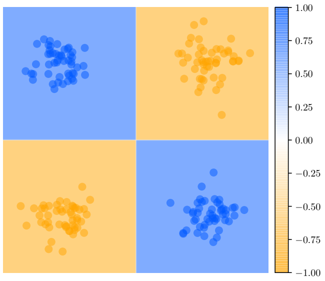


Fig. 2: Classifier found via MCTS: $f_{16} \circ f_7 \circ f_1(\boldsymbol{x}) = \operatorname{sign}\Big(\prod_i \big(\boldsymbol{R}\boldsymbol{x}\big)_i\Big)$

3 Research Directions

We sketched a deep learning technique that can incorporate world knowledge and work in thin data scenarios. Yet, while there are first efforts on properties of functional composition for machine learning [20], many questions regarding our idea are still open. Two pressing ones are: How to systematically formalize expert knowledge to fit into this framework? Are there universal functional building blocks from which to compose solutions for arbitrary problems?

References

1. Bengio, Y.: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning **2**(1) (2009)
2. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. In: Proc. NIPS. (2012)
3. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N.: Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Processing Magazine **29**(6) (2012)
4. Conneau, A., Schwenk, H., Barrault, L., LeCun, Y.: Very Deep Convolutional Networks for Text Classification. arXiv:1606.01781 [cs.CL] (2016)
5. Cireşan, D., Giusti, A., Gambardella, L., Schmidhuber, J.: Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks. In: Proc. MICCAI. (2013)
6. Silver, D., et al.: Mastering the Game of GO with Deep Neural Networks and Tree Search. Nature **529**(7587) (2016)
7. Vapnik, V., Chernovenkis, A.: On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. Theory of Probabilities and its Applications **16**(2) (1971)
8. Morgan, N., Bourland, H.: Generalization and Parameter Estimation in Feedforward Nets: Some experiments. In: Proc. NIPS. (1990)
9. Ribeiro, M., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv:1602.04938 [cs.LG] (2016)
10. Brown, T., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial Patch. arXiv:1712.09665 [cs.CV] (2017)
11. Tickle, A., Andrews, R., Golea, M., Diederich, J.: The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural networks. IEEE Trans. Neural Networks **9**(6) (1998)
12. van Lent, M., Fisher, W., Mancuso, M.: An Explainable Artificial Intelligence System for Small-unit Tactical Behavior. In: Proc. IAAI, AAAI (2004)
13. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLoS ONE **10**(7) (2015)
14. Wahabzada, M., Mahlein, A.K., Bauckhage, C., Steiner, U., Oerke, E.C., Kersting, K.: Metro Maps of Plant Disease Dynamics – Automated Mining of Differences Using Hyperspectral Images. PLoS ONE **10**(1) (2015)
15. Belle, V.: Logic Meets Probability: Towards Explainable AI Systems for Uncertain Worlds. In: Proc. IJCAI. (2017)
16. Sifa, R., Ojeda, C., Cvejoski, K., Bauckhage, C.: Interpretable Matrix Factorization with Stochasticity Constrained Nonnegative DEDICOM. In: Proc. KDML. (2017)
17. Browne, C., et al.: A Survey of Monte Carlo Tree Search Methods. IEEE Trans. Computational Intelligence and AI in Games **4**(1) (2012)
18. Sutton, R., Barto, A.: Reinforcement Learning an Introduction. MIT Press (1998)
19. Ernest, N., Carroll, D., Schumacher, C., Clark, M., Cohen, K., Lee, G.: Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions. Journal of Defense Management **6**(1) (2016)
20. Bohn, B., Griebel, M., Rieger, C.: A Representer Theorem for Deep Kernel Learning. arXiv:1709.10441 [cs.LG] (2017)