Extracting Rules from Neural Networks with Partial Interpretations

Cosimo Persia, Ana Ozaki

Department of Informatics
University of Bergen, Norway

Abstract

We investigate the problem of extracting rules, expressed in Horn logic, from neural network models. Our work is based on the exact learning model, in which a learner interacts with a teacher (the neural network model) via queries in order to learn an abstract target concept, which in our case is a set of Horn rules. We consider *partial interpretations* to formulate the queries. These can be understood as a representation of the world where part of the knowledge regarding the truthness of propositions is unknown. We employ Angluin's algorithm for learning Horn rules via queries and evaluate our strategy empirically.

1 Introduction

Neural networks have been used to achieve important milestones in artificial intelligence [4, 10, 12, 8], but it is difficult to understand how predictions of the models are made, and this limits their usability. In this work, we propose an approach for extracting rules from black-box machine learning models, such as neural networks. It is often the case that not all values in a dataset are known or trustable. For this reason, our approach assumes settings in which the dataset used to train the neural network contains missing values.

We first binarize a given dataset and we train a neural network with it. Then, we run the LRN algorithm [9]. This algorithm poses queries to the neural network, seen as a teacher, in order to extract rules encoded in it. Rules are represented using Horn logic, for example, they can be of the form $((\mathsf{horse} \wedge \mathsf{wings}) \rightarrow \mathsf{pegasus})$. With Horn rules, we can carry automated reasoning in polynomial time, and it is feasible to check the quality of the model.

We perform an empirical study using the hepatocellular carcinoma (HCC) dataset [16], which describes survivability of patients diagnosed with hepatocellular carcinoma according to clinical information. HCC contains many missing values of attributes of patients. We compare the hypothesis built with our approach with the hypothesis built by a state-of-the-art implementation of the incremental decision tree algorithm [7]. Our rule extraction procedure correctly extracts meaningful rules and it is two times faster than the decision tree algorithm.

Related Work. A similar work [20] extracts probabilistic automata from neural networks by asking queries, and a recent work [15] focuses on how to better simulate queries asked to black-box models. We can also find methods that verify binarized neural networks by extracting a binary decision diagram [17] through queries. The interpretability field is large and there are many approaches to interpret neural networks models [21]. Our technique belongs to the global and active approach that explains the already trained model as a whole, as opposed to changing the network architecture for interpretability (passive), or explaining through feature studies or correlation (local).

2 Preliminaries

We provide basic definitions for (propositional) logic and the exact learning model.

https://doi.org/10.7557/18.6301

2.1 Logic and Neural Networks

Let V be a finite set of *boolean variables*. *A literal* over V is either a *variable* $\mathsf{v} \in \mathsf{V}$ or its negation $\neg\mathsf{v}$. A literal is *positive*, if it is a variable, and *negative* otherwise. *A clause* over V is a disjunction ($\vee$) of literals over V. It is *Horn* if at most one literal is positive. *A (propositional) formula* over V is a conjunction of clauses over V (in conjunctive normal form). It is *Horn* if its clauses are Horn.

An *interpretation* is a function that maps all variables V to either 0 (false) or 1 (true). It also maps the constant symbol $\top$ to 1 and $\bot$ to 0. We write $v \in \mathcal{I}$ if $\mathcal{I}(v) = 1$. We may omit 'over V' in formulas, clauses, literals, and interpretations. A variable $\mathsf{v} \in \mathsf{V}$ is *satisfied* by $\mathcal{I}$ if $\mathsf{v} \in \mathcal{I}$, otherwise it is *falsified*. A negative literal $\neg\mathsf{v}$ is satisfied by $\mathcal{I}$ iff v is falsified by $\mathcal{I}$. A clause c is satisfied by $\mathcal{I}$ iff at least one literal in c is satisfied by $\mathcal{I}$ and a formula t is satisfied by $\mathcal{I}$ iff every clause in t is satisfied by $\mathcal{I}$. A *partial interpretation* extends the notion of an interpretation by allowing some values to be "missing" or "unknown", denoted '?'. In detail, it is a function which maps V to $\{0, 1, ?\}$. A partial interpretation $\mathcal{I}$ satisfies a formula t if there is a way to replace each ? in its image by either 0 or 1 and the resulting function satisfies t.

Horn clauses c can be written as *rules* of the form $\mathsf{ant}(\mathsf{c}) \rightarrow \mathsf{con}(\mathsf{c})$, where $\mathsf{ant}(\mathsf{c})$ (*antecedent*) is the set of variables that occur negated in c or the constant symbol $\top$ if none is negated; and $\mathsf{con}(\mathsf{c})$ (*consequent*) is the positive literal in c or $\bot$ if none is positive.

If an interpretation $\mathcal{I}$ satisfies a literal, a clause or theory x, we write $\mathcal{I} \models x$, otherwise, $\mathcal{I} \not\models x$. Let t be a theory and let c be a clause. If, for every $\mathcal{I}$, we have that $\mathcal{I} \models t$ implies $\mathcal{I} \models \mathsf{c}$, then we write $t \models \mathsf{c}$ and we say that t *entails* c. If t entails every clause in a theory t', then we also write $t \models t'$. If $t' \models t$ also holds, then t and t' are logically equivalent and we write $t \equiv t'$. We say that a formula ϕ is *satisfiable* if there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \phi$ and *falsifiable* if its negation $\neg\phi$ is satisfiable.

Neural network models in this work can be understood as an alternative way of representing a formula in propositional logic. A neural network model N is a function that receives a vector in the $|\mathsf{V}|$ dimensional space, with values in $\{0, 1, ?\}$ (with '?' standing for an 'unknown value'), and outputs a classification of this input. The mapping from interpretations to vectors is defined as follows. Given an interpretation $\mathcal{I}$ over V, we assume a total order on the elements of V and denote by $\mathsf{vector}(\mathcal{I})$ the vector in the $|\mathsf{V}|$ dimensional space where the element at position i is 1 if $\mathsf{v}_i \in \mathcal{I}$ and 0 otherwise. In this work, a *dataset* is a set of elements of the form $(\mathsf{vector}(\mathcal{I}), l)$, where l is either 0 or 1 and $\mathcal{I}$ is a partial interpretation. For every neural network N trained on a given dataset, there is a propositional formula t_N such that $N(\mathsf{vector}(\mathcal{I})) = 1$ iff $\mathcal{I} \models t_N$. In this sense, N can be seen as an alternative representation of t_N.

2.2 Learning via Queries

To formally define the problem setting, we use the notion of a *learning framework* $\mathfrak{F}$ as pair $(\mathcal{E}, \mathcal{H})$, where $\mathcal{H}$ is the set of all formulas in propositional logic and $\mathcal{E}$ is the set of all partial interpretations (over variables in V). We say that $\mathfrak{F}$ is *Horn* if $\mathcal{H}$ is restricted to the set of all Horn formulas. For any $h \in \mathcal{H}$, $\mathcal{I} \in \mathcal{E}$ is a *positive example for* h, if $\mathcal{I} \models h$, and, a *negative example for* h if $\mathcal{I} \not\models h$. For any $h, t \in \mathcal{H}$, a *counterexample* for t and h is an example $\mathcal{I} \in \mathcal{E}$ such that either $\mathcal{I} \models t$ and $\mathcal{I} \not\models h$ (a *positive counterexample*), or $\mathcal{I} \models h$ and $\mathcal{I} \not\models t$ (a *negative counterexample*).

We study the problem of identifying an unknown target $t \in \mathcal{H}$ by posing queries to two kinds of oracles [9] (implementation in Section 3.1). A *membership oracle* $\mathsf{MQ}_{\mathfrak{F},t}$ is a function that takes as input $\mathcal{I} \in \mathcal{E}$ and it outputs 'yes' if $\mathcal{I} \models t$, 'no' otherwise. An *equivalence oracle* $\mathsf{EQ}_{\mathfrak{F},t}$ takes as input a hypothesis $h \in \mathcal{H}$ and it outputs 'yes' if $h \equiv t$, otherwise, it outputs a counterexample for t and h. A *membership query* is a call to $\mathsf{MQ}_{\mathfrak{F},t}$ and an *equivalence query* is a call to $\mathsf{EQ}_{\mathfrak{F},t}$.

Definition 1 (Exact Learning)**.** A learning framework $\mathfrak{F}$ $(\mathcal{E}, \mathcal{H})$ is *exactly learnable* if there is a deterministic algorithm A that takes as input the set of variables V used to formulate the target $t \in \mathcal{H}$, asks membership and equivalence queries, and outputs a hypothesis $h \in \mathcal{H}$ equivalent to t. We say that $\mathfrak{F}$ is *exactly learnable in polynomial time* if the number of steps used by A is bounded by a polynomial on $|t|$ and the largest counterexample seen so far. Each query counts as one step of computation.

3 Extracting Horn Rules with Partial Interpretations

The goal of our work is to find rules hidden in a black box machine learning model such as a trained neural network model. We present an adaptation of the LRN algorithm [9] that learns from partial interpretations instead of entailments, as originally proposed by the authors of the mentioned paper. This algorithm is able to exactly identify any unknown target Horn theory by posing queries to oracles that can answer membership and equivalence queries. The algorithm is guaranteed to terminate in polynomial time with respect to the number of variables into consideration.

3.1 The LRN* Algorithm

We adapted LRN so that it is able to learn rules from partial interpretations. Membership queries take as input partial interpretations and counterexamples to equivalence queries are also partial interpretations. Algorithm 2 shows the main steps of the modified algorithm.

Algorithm 1 LRN*

Input: It is assumed that the learner knows $\mathfrak{F}$ (that is, it knows that the hypothesis should be a Horn theory) but not the target t.
Output: h such that $h \equiv t$.
Let S be the empty sequence.
Denote with $\mathcal{I}_i$ the i-th element of S.
Let h be the empty hypothesis.
while $\mathsf{EQ}_{\mathfrak{F},t}(h)$ returns a counterexample $\mathcal{I}$ **do**
  if there is $\mathcal{I}_i \in S$ such that $\mathcal{I}_i \cap \mathcal{I} \subset \mathcal{I}_i$ and $\mathsf{MQ}_{\mathfrak{F},t}(\mathcal{I}_i \cap \mathcal{I}) =$ 'no' **then**
    replace the first such $\mathcal{I}_i$ with $\mathcal{I}_i \cap \mathcal{I}$ in S
  else
    append $\mathcal{I}$ to S
  end if
  $h := \bigcup_{\mathcal{I} \in S} \{(\bigwedge_{\mathsf{v} \in \mathcal{I}} \mathsf{v}) \rightarrow \mathsf{u} \mid \mathsf{u} \in \mathsf{RHS}(\mathsf{V}, \bigwedge_{\mathsf{v} \in \mathcal{I}} \mathsf{v})\}$
end while
return h

LRN poses equivalence queries until it receives 'yes' as an answer. It keeps track of important partial interpretations that falsify the target. Each such partial interpretation corresponds to a rule

Algorithm 2 RHS

Input: V: variables. $\alpha \subseteq \mathsf{V}$
Output: A subset of $\mathsf{V} \cup \{\bot\}$
return $\{v \mid v \in \mathsf{V} \cup \{\bot\} \setminus \alpha,\ \mathsf{MQ}_{\mathfrak{F},t}(\bigwedge_{u \in \alpha} u \rightarrow v) =$ 'yes'$\}$

entailed by the target [6]. Upon receiving a negative counterexample, the algorithm asks membership queries to find more specific antecedents of rules entailed by the target. After that, it adds to the hypothesis rules entailed by the target by asking membership queries and the process repeats. Correctness and termination of Algorithm 2 can be proven simi'larly as with the LRN algorithm [9]. This is possible because we can simulate membership and equivalence queries from the learning from entailments setting to the learning from partial interpretations setting (and vice-versa) [6, Theorem 16].

To simulate the membership oracle $\mathsf{MQ}_{\mathfrak{F},t_N}$, we directly use the classifier N. Whenever the algorithm calls $\mathsf{MQ}_{\mathfrak{F},t_N}$ with input a partial interpretation $\mathcal{I}$, we check if $N(\mathsf{vector}(\mathcal{I})) = 1$, which means that $\mathcal{I} \models t_N$. If so, we return the answer 'yes' to the algorithm, 'no' otherwise. Simulating an equivalence query oracle $\mathsf{EQ}_{\mathfrak{F},t_N}$ is not as straightforward as we are checking if the hypothesis constructed is equivalent to t_N.

We simulate $\mathsf{EQ}_{\mathfrak{F},t_N}$ by generating a set of examples randomly and classifying the examples using membership queries. Then, we can search for examples in this set that the hypothesis constructed by LRN misclassifies. Depending on the size of the set of examples randomly generated [2, Section 2.4], if the hypothesis does not misclassify any example then one can ensure that with high probability the total number interpretations misclassified (considering the entire space of partial interpretations) is low. More precisely, if the size of the set of examples generated randomly is at least $\frac{1}{\epsilon} log_2(\frac{|\mathcal{H}|}{\delta})$ [19], then one can ensure that the hypothesis constructed is probably approximately correct [18]. The parameter $\epsilon \in (0, 1)$ indicates the probability that the hypothesis misclassifies an interpretation w.r.t. the target and $\delta \in (0, 1)$ is the probability that the learned hypothesis errs more than ϵ.

If $\mathcal{H}$ corresponds to the class of formulas only

expressible with Horn logic and variables V, then the number of logically different hypothesis in $\mathcal{H}$ is close to [1, 3]:

$$2^{\binom{|\mathsf{V}|}{\lfloor |\mathsf{V}|/2 \rfloor}}. \tag{1}$$

This number follows from the fact that Horn logic is closed under intersection: if $\mathcal{I}$ and $\mathcal{I}'$ satisfy a Horn theory then $\mathcal{I} \cap \mathcal{I}'$ also does [11].

3.2 Representing constraints

We explain how we can express constraints that are going to be extracted in the experimental section. Horn rules r are of the form $\mathsf{ant}(r) \rightarrow \mathsf{con}(r)$: $(\mathsf{sunny} \wedge \mathsf{happy}) \rightarrow \mathsf{jogging}$ where all the variables both in the antecedent and in the consequent are not negated. This means that with Horn logic we cannot express rules of the form:

$$(\neg\mathsf{sunny} \wedge \mathsf{happy}) \rightarrow \mathsf{boardgame\_night}.$$
$$(\mathsf{empty\_fridge} \wedge \mathsf{hungry}) \rightarrow \neg\mathsf{happy}.$$

To express a 'weak' form of negation, we duplicate all the variables in V and treat every new variable as the negation of a variable in V. For example, let $\hat{\mathsf{v}}_i$ be the duplicated variable of any $\mathsf{v}_i \in \mathsf{V}$. We can express the rule

$$(\mathsf{sun\hat{n}ny}_1 \wedge \mathsf{happy}) \rightarrow \mathsf{boardgame\_night}.$$

Usually, when duplicating variables in this way, we would like to avoid that both paired variables are true in a partial interpretation (since they represent each other's negation). For this reason, we assume that Horn rules of the form

$$(\mathsf{v} \wedge \hat{\mathsf{v}}) \rightarrow \bot \tag{2}$$

always hold, for every $\mathsf{v} \in \mathsf{V}$.

4 Experiments

In this section we show experimental results using the approach presented in the previous section where a trained neural network is treated as an oracle for the LRN algorithm. We implemented the algorithm in a Python 3.9 script and we used the SymPy library [13] to express rules and check for satisfiability of formulas. For the neural networks, we used the Keras library [5]. Our LRN implementation can start with an empty hypothesis or with a set of Horn formulas as background knowledge (assumed to be true properties of the domain at hand). The background knowledge can also be used to check if the neural network model respects some desirable properties. We conduct the experiments on an Ubuntu 18.04.5 LTS with i9-7900X CPU at 3.30GHz with 32 logical cores, 32GB RAM.

We experiment our approach of extracting Horn theories from partial interpretations on a dataset in the medical domain [16]. This dataset contains missing values for attributes. We can consider each instance as a partial interpretation that sets some variables (attributes of that instance) to true, some to false, and other variables to "unknown".

4.1 HCC Dataset

Hepatocellular carcinoma (HCC) causes liver cancer, and it is a serious concern for global health. The HCC dataset [16] consists of 165 instances of many risk factors and features of real patients diagnosed with this illness.

There are 49 features selected according to the EASL-EORTC (European Association for the Study of the Liver - European Organisation for Research and Treatment of Cancer). From these features, 26 are quantitative variables, and 23 are qualitative variables. Missing values represent 10.22% of the whole dataset and only 8 patients have complete information in all fields (4.85%). The target class of each patient is binary. Each patient is classified positively if they survive after 1 year of having been diagnosed with HCC, and negatively otherwise. 63 cases are labelled negatively (the patient dies) and 102 positively (the patient survives). Quantitative variables describe, for example, the amount of oxygen saturation in the human body, the concentration of iron in the blood, or number of cigarettes packages consumed per year. The range of the values that each variable can assume varies, but it is specified. Qualitative variables can only have two different values in this dataset (either 0 or 1). Usually they describe categorical information such as if the patient comes from an endemic country, or if it is obese, etc.

The LRN^* algorithm expects to receive counterexamples in the form of a partial interpretation that specifies the truth values of boolean variables. For this reason, we encode quantitative variables in a binary representation format. The interval of

values of each quantitative variable is partitioned into three sub-intervals. These intervals divide the values of the quantitative variable into "low", "middle", and "high" values. For example, the interval of values of the variable that describes the number of cigarettes packages consumed by the patient per year is $[0, 510]$ can be partitioned into $[0, 50], (50, 200], (200, 510]$.

The binarised dataset has in total $26*3+23+1 = 102$ variables and it can be considered a set of partial interpretations. A missing value in the new dataset is denoted with '?' similarly as in the original one, otherwise the value is 1 (0) if the variable is set to true (false). Each partial interpretation I matches a rule (not necessarily Horn) of the form

$$(l_1 \wedge \cdots \wedge l_{n-1}) \rightarrow l_n \tag{3}$$

where each l_i is a positive literal if the variable i is set to true in I and false otherwise. The literal l_k is not present in the rule if l_k has a missing value.

As explained in the previous section, by duplicating the number of variables and pairing them such that one represents the negation of another variable, we can express the previous rule with a Horn formula. For this reason, we further modify the dataset by duplicating variables. Each new variable semantically represents the negated concept of its paired variable. So, we form a dataset D of partial interpretations with 204 variables.

We can express each example in D with Horn rules like in Formula 3. We denote by T the set of such rules that can be formed by looking at all partial interpretations in the extended dataset. To express disjointness constraints between paired variables, we assume T to also have the additional Horn rules of the form $(\mathsf{v}_i \wedge \hat{\mathsf{v}}_i) \rightarrow \bot$ (Formula 2).

Finally, the dataset used for training the neural network is formed by randomly generating partial interpretations (with 204 variables) whose classification label is 0 if they do not satisfy a rule in T, 1 otherwise.

4.2 Model selection

By only randomly generating partial interpretations (with 204 variables), we can create a very unbalanced dataset with most partial interpretations classified as positive by the target Horn theory T (note: T is defined in the previous subsection). We solve this problem by oversampling interpretations with negative label that are created by violating rules that match interpretations in the binarised dataset. In total, there are 200 negative examples and 200 positive examples in the training dataset. 80% of the (balanced) binarised dataset was used for training and validation. We used 3-fold. As T is a Horn theory, there is no noisy data generated in this process.

Hidd. Layers	**L. rate**	**Accuracy**
32, 16, 8, 16, 32	0.1	0.9612
32, 32, 32, 8	0.1	0.9592
32, 32, 32	0.1	0.9382
32, 8, 32, 16	0.1	0.9217

Table 1: Architecture and learning rate of the top four neural networks in ascending order with respect accuracy. The model in the first row was the selected one.

#Equiv.	t_h	t_nn	h_nn	t_tree
100	9.2%	6.0%	5.8%	8.4%

Table 2: The outcome of the rule extraction process with the HCC dataset. The numbers are the percentages of interpretations classified differently between the target t (Section 4.1), neural network nn, LRN^* hypothesis h, and the tree.

We built a sequential neural network model, where the number of nodes in the input layer is 204, which is the number of variables in a partial interpretation. We used the library "Keras version 2.4.3" [5] and we empirically searched for the sequential architecture with the best performance varying the number of hidden layers, nodes in hidden layers and the learning rate.

We searched our model with the following hyperparameters: 2,3,4,5 numbers of hidden layers, 4, 8, 16, 32 nodes per layer, and 0.001, 0.01, 0.1 as the learning rate. The model with the best performance had 5 hidden layers, $32, 16, 8, 16, 32$ nodes per layer, and 0.1 learning rate. In total we tested (No.learning rates x No. node-layer combinations) $= 3 \cdot (4^2 + 4^3 + 4^4 + 4^5) = 4080$ configurations. This means that we carried in total $3 \cdot 4080 = 12240$ training and evaluation runs. The best performing architectures are showed in Table 1.

4.3 Test Setting

In our experiments, we run the LRN^* algorithm and we set a limit of 100 equivalence queries that the algorithm can ask before terminating with the built hypothesis as its output. To simulate an equivalence query, we randomly generate a sample of partial interpretations and we classify each interpretation using the neural network. Afterwards, we search for a counterexample to return to h as the answer of the query.

We compare the quality of the LRN^* hypothesis with the hypothesis formed by an incremental decision tree [7], an established white box machine learning model. We use "Hoefffding Decision Tree" implementation present in the "skmultiflow" framework [14]. It is possible to generate a set of propositional rules by visiting every branch of the tree from the root to leafs labelled negatively. The sampling idea for finding negative counterexamples for LRN^* is also used for extracting a decision tree from the neural network.

We generate partial interpretations randomly and they are classified by the neural network. We check if at least one of those classified partial interpretations is misclassified by the decision tree algorithm. If this is the case, we incrementally train the tree with the entire sample. This process is repeated until all classified interpretations in the sample are correctly classified by the tree.

Since the considered number of variables is 204, it is not feasible to have the size of the sample for simulating equivalence queries as dictated by Formula 1 (this number is of the order 2^{204}). Moreover, a size of the sample too small often fails in finding a counterexample. When it is the case, the LRN^* algorithm will terminate and output a hypothesis with few (if not zero) rules, and the tree will only be one node with label 1. This problem is especially noticeable in our current scenario as there are many variables but relatively few interpretations that are negatively labelled by the neural network. The selected size of the sample is therefore

$$s := \left\lceil \frac{1}{\epsilon} log_2(\frac{2^{|\mathsf{V}|^{2.1}}}{\delta}) \right\rceil$$

, both for training the decision tree and for answering queries asked by the LRN^* algorithm.

When the LRN^* hypothesis and the tree have been extracted, we compute a partial truth table of 204 variables of size $2s$. We classify these interpretations according to the target T, the neural network, the LRN^* hypothesis and the decision tree. We then compare the truth tables and count the number of times an interpretation is classified differently between the different models.

4.4 Results

Table 2 shows the outcome of our experiment. The columns *t_h*, *t_nn*, *h_nn*, *t_tree* are, respectively, the percentage of interpretations that are labelled differently between the target and the hypothesis, the target and the neural network, the hypothesis and the neural network, and the tree and the target. The running time of the LRN^* algorithm with at most 100 equivalence queries was around 60 hours. The time for extracting an incremental decision tree is twice, around 120 hours.

The type of rules that the LRN^* algorithm extracted are of the form:

$$\{\mathsf{medium\_hemoglobin} \wedge \cdots \wedge \hat{\mathsf{obese}} \rightarrow \mathsf{survives}\}$$

with around 40 different variables in the antecedent. With 100 equivalence queries, the hypothesis extracted has 20 rules of this type that are also present in the target T. Other rules that are entailed by T can be found in the hypothesis. Examples labelled negatively with many missing values contain more information about the dependency between variables that must be respected. Indeed, we noticed an increase of the accuracy of the neural network trained on more missing values ensuring ensured balanced classes. As a consequence, also the quality of the extracted rules improves.

5 Conclusion

In this work we presented an approach for extracting Horn rules from neural network models using partial interpretations. It is often the case that not all values in a dataset are known or trustable. Our method based on partial interpretations covers such scenarios and generalizes the case with (full) interpretations. We test our approach empirically using a real world dataset in the medical domain.

Acknowledgements

Ozaki is supported by the Research Council of Norway, project number 316022.

References

[1] V. Alekseev. On the number of intersection semilattices [in russian]. *DiskretnayaMat.1*, page 129–136, 1989.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988. ISSN 0885-6125. doi: 10.1023/A:1022821128753.

[3] G. Burosch, J. Demetrovics, G. Katona, D. Kleitman, and A. Sapozhenko. On the number of closure operations. pages 91–105. János Bolyai Mathematical Society, Budapest, 1993.

[4] M. Campbell, A. J. H. Jr., and F. Hsu. Deep blue. *Artif. Intell.*, 134(1-2):57–83, 2002. doi: 10.1016/S0004-3702(01)00129-1.

[5] F. Chollet et al. Keras, 2015. URL `https://github.com/fchollet/keras`.

[6] L. De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201, 1997. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00041-6.

[7] P. M. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, pages 71–80. ACM, 2000. doi: 10.1145/347090.347107.

[8] D. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56:1:1–1:15, 05 2012. doi: 10.1147/JRD.2012.2184356.

[9] M. Frazier and L. Pitt. Learning from entailment: An application to propositional horn sentences. In *ICML*, 1993. doi: 10.1007/3-540-49730-7_11.

[10] S. Hölldobler, S. Möhle, and A. Tigunova. Lessons learned from alphago. In *YSIP2*, pages 92–101. CEUR-WS.org, 2017.

[11] A. Horn. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951. ISSN 00224812. doi: 10.2307/2268661.

[12] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*. icml.cc / Omnipress, 2012. doi: 10.48550/arXiv.1112.6209.

[13] A. Meurer et al. Sympy: symbolic computing in python. *PeerJ Comput. Sci.*, 3:e103, 2017. doi: 10.7717/peerj-cs.103.

[14] J. Montiel et al. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018. doi: 10.48550/arXiv.1807.04662.

[15] T. Okudono et al. Weighted automata extraction from recurrent neural networks via regression on state spaces. In *AAAI*, pages 5306–5314. AAAI Press, 2020. doi: 0.1609/aaai.v34i04.5977.

[16] M. S. Santos et al. A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. *Journal of Biomedical Informatics*, 58:49–59, 2015. ISSN 1532-0464. doi: 10.1016/j.jbi.2015.09.012.

[17] A. Shih, A. Darwiche, and A. Choi. Verifying binarized neural networks by angluin-style learning. In *SAT*, 2019. doi: 10.1007/978-3-030-24258-9_25.

[18] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782. doi: 10.1145/1968.1972.

[19] V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. 1971. doi: 10.1007/978-3-319-21852-6_3.

[20] G. Weiss, Y. Goldberg, and E. Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In *ICML*, volume 80, pages 5244–5253. PMLR, 2018. doi: 10.48550/arXiv.1711.09576.

[21] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021. doi: 10.1109/TETCI.2021.3100641.