ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Reinforcement learning for constructing low density sign representations of Boolean functions

Oytun Yapar[1] and Erhan Oztop[1,2]

[1]Computer Science Department, Ozyegin University, Istanbul, Turkey
[2]SISReC, OTRI, Osaka University, Osaka, Japan

Abstract. Boolean functions (BFs) can be uniquely represented with polynomial functions by representing True and False with ± 1. With the 'sign-representation' framework, i.e., when the sign of the polynomials is used instead of the exact ± 1, the representation is not unique anymore, and several measures of sign-representation become the target of research. One such measure is the polynomial threshold function density (PTF density), i.e., the minimum number of monomials that suffices to sign-represent a given BF. Several algorithms can find sign-representations with a low number of monomials; however, to find a representation with the minimum number of monomials possible is a combinatorial search problem. The recent success of reinforcement learning (RL) algorithms in solving combinatorial search problems poses the question of whether RL can perform well in finding sign-representations with a low number of monomials. To address this question, we focused on Deep Q-Networks (DQN) and explored its applicability to the sign-representation problem. To be concrete, we present our work on modeling RL agents for solving the sign-representation problem and give our results on the application of DQN to BFs with a low number of variables ($n = 4$). Our results indicate that the trained DQN agent generalizes well and exploits intrinsic structure of BFs, such as their equivalence in terms of certain equivalence relations.

1 Introduction

1.1 The sign representation of Boolean functions

An n-variable Boolean function (BF) can be represented by a unique multilinear vector function with at most 2^n terms i.e. monomial when ± 1 is used to represent True and False. The coefficients of the monomials can be easily found by considering the BF and the coefficients as 2^n-vectors of $\mathbf{f}$ and $\mathbf{a}$ respectively with $\mathbf{a} = \mathbf{D}^{-1}\mathbf{f}$ where $\mathbf{D}$ is a $2^n \times 2^n$ Sylvester-type Hadamard matrix [7]. $\mathbf{D}$ has a recursive form and almost orthogonal with the property $\mathbf{D}^{-1} = 2^{-n}\mathbf{D}$. The coefficients of multilinear function representing the BF f is called the *spectrum of* f. Consider the function $g : \{1, -1\}^n \to \mathbb{R}$ $(g \neq 0)$, a BF f can be represented also by $f = sign(g)$. The coefficients of such functions that satisfy this relation can still be found with the help the Sylvester-type Hadamard matrix by again using the vector notation as follows:

$$\mathbf{a}^T_{g(\mathbf{k})} = \mathbf{k}^T \mathbf{diag}(\mathbf{f})\mathbf{D} : \forall\, \mathbf{k} > \mathbf{0}, \mathbf{k} \in \mathbb{R}^{2^n}, \tag{1}$$

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

where $\mathbf{a}^T_{g(\mathbf{k})}$ indicates the coefficients of g induced by the selection of $\mathbf{k}$. It is often useful to define $\mathbf{Q}_f = \mathbf{diag}(\mathbf{f})\mathbf{D}$ which is simply the Sylvester-type Hadamard matrix whose rows are scaled by the function output [5]. It is easy to see that given a $\mathbf{k} > \mathbf{0}$, the coefficient of a monomial corresponding to column i, can be found by $\mathbf{a}^T_i = \mathbf{k}^T\mathbf{Q}_{*,i}$ $(1 \leq i \leq 2^n)$ where the f subscript is dropped for clarity. In particular, the row-sum of $\mathbf{Q}_f$ (i.e., $\mathbf{1^T Q}$) is called the *Walsh spectrum of* f, denoted by $\mathbf{w}$, which is 2^n times the *spectrum* of f.

1.2 The minimum-term sign representing polynomial problem

The *polynomial threshold function density (PTF density)* of f is the minimum number of monomials that can represent a given n-variable BF f. It is possible to find the PTF density of f with an exhaustive search procedure. Let $\mathbf{R}$ be an r-column subset of $\mathbf{Q}_f$, where $\mathbf{Q}_f = \mathbf{diag}(\mathbf{f})\mathbf{D}$ $(1 < r < 2^n)$. Then the condition

$$\mathbf{k}^T\mathbf{R} = \mathbf{0} : \forall\, \mathbf{k} > \mathbf{0}, \mathbf{k} \in \mathbb{R}^{2^n} \tag{2}$$

guarantees that the PTF density of f is at most $2^n - r$ since $\mathbf{k}^T\mathbf{Q}_f$ is a sign-representation for f (due to Eq.1). Thus given an $\mathbf{R}$ we can ask a *linear programming* solver to check whether a $\mathbf{k} > \mathbf{0}$ satisfying 2 exists. The minimum term sign representing polynomial then can be found by iterating over all possible r-column $(1 < r < 2^n)$ subsets of $\mathbf{Q}$ and returning the maximum of r that is satisfiable[8]. However, this algorithm is super-exponential in n, thus heuristics can be used reduce the r-iteration and still find sufficiently short (i.e. with low number of terms) sign-representation [6]).

1.3 The equivalence classes of sign representing polynomials

Considering a sign-representing polynomial $p(x_1, x_2, \ldots, x_n)$, it is easy to see that the transformations of negation and permutation of input variables, as well as negating the polynomial itself would not change the number of terms in the new polynomial. Similarly, the multiplication of p or any of the input variables with an arbitrary monomial of $x_1, x_2, \ldots, x_n$ would not change the number of terms in the new polynomial formed. Thus, all the BFs that can be transformed to each other with any combination of these five operations can be considered equivalent in the sense that they must have the same PTF density. The set of all BFs that can be converted to each other in this way forms so-called an *equivalence class* [2, 6]. Therefore, in the quest for the PTF density of all BFs, we can focus only on one BF from each equivalence class. A convenient way to label BFs is to associate them with integers in the range of 0 to 2^{2^n} and use hexadecimal notation to refer to BFs. We adopt the labeling used in [6] that associates $\mathbf{f}$ with the binary number $(1 - \mathbf{f})/2$. According to this, for example, $\mathbf{f} = [-1, 1, -1, 1, -1, 1, -1, 1]$ is represented with **0xaa** in hexadecimal (**10101010** in binary). There are only 8 equivalent classes in the space of 4-variable BFs making $n = 4$ an ideal place to explore algorithms for the low number of monomial sign-representations.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

2 Reinforcement Learning for finding sign-representations

Reinforcement learning (RL) aims to equip an agent with optimal policy based on active interaction with an environment that provides a scalar feedback in terms of a reward signal. The agent learns by exploration and by making use of the reward function. In this study, we adopted DQN, an RL method that is based on Q-learning [4], where the state-action value table (i.e., the Q table) is approximated by a deep neural network. Successful applications of DQN on combinatorial search problems such as Minimum Vertex Cover, Maximum Cut, and Traveling Salesman problem ([3], [1]) have motivated us to adopt DQN.

2.1 The RL setup for finding short sign-representations

An RL agent operates on a state space S, with an action set A. In the deterministic case, the environment is modeled with a transition function $T(s,a) : S \times A \to S$ and a reward function $R(s,a) : S \times A \to \mathbb{R}$. Thus we need to define these entities within the sign-representation framework.

2.1.1 State, Action, Transition function and Reward definitions

State space (S): The state-space S contains the triplets of the Walsh spectrum of the BF under consideration (w), an indicator vector representing the monomials selected for elimination (c), and the number of monomials that can be eliminated with c (m) :

$$s = [w, c, m] : \forall\, c_i \in \{0,1\}, c_i \in c \tag{3}$$

Walsh spectrum is part of the state, as DQN agent must be able to operate on multiple BFs, and thus it must know which BF it is operating on. The indexes $i : c_i = 1$ determine the column subset $\mathbf{R}$ of $\mathbf{Q}$ to be tested for feasibility in the sense of Eq.2. For notational convenience we use s_w, s_c and s_m to indicate the components of the state s.

Action space (A): The action space is given by $A = \{a : a \in \mathbb{Z}, 1 \leq a \leq 2^n\}$. An action indicates which monomial to be added to the current elimination set (defined with c in the current state).

Transition function (T): After an action, a, is issued by the agent, the environment responds by creating a new elimination set by adding the indicated monomial to the set (i.e. $c' = c$, followed by $c'_a = 1$). Then, the feasibility of this elimination set is checked (Eq.2) with the *linear programming* solver. According to the response of the solver, the state of the agent is updated as $s' = [w, c', m']$, where m' indicates the number of monomials that can be zeroed (i.e. the number of 1s in the c, it will increase if the current elimination set can indeed be eliminated).

Reward function (R): The reward function awards the agent if it finds more monomials for elimination than the previous state:

$$R(s,a) = \begin{cases} (s'_m)^2 & s'_m > s_m \text{ where } s' = T(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

2.2 The scope of the experiments

We conducted our experiments on 4-variable BFs. Two types of experiments were designed: in the first, we compared the DQN agent with random subset selecting agents ($Rand_v$) by comparing the monomial elimination performances. $Rand_v$ agents are random action agents taking actions in the sense of the RL agent defined above. The v subscript indicates that the agent is allowed to run v times to increase its chance of finding low-monomial solutions. Each trial of the Rand$_v$ agent ends when it faces an unfeasible c (Eq. 3). The c which has the maximum number of 1s (i.e. the maximum number of monomials for the elimination) is used as the performance of the Rand$_v$ agent for the benchmark. In the second experiment type, we measured the equivalence class generalization performance of the DQN agent by designing the training set as not to include BFs from a selected equivalence class.

2.3 The results of the experiments

The PTF density of all the 4-variable BF is known [6], thus the maximum number of monomials that can be eliminated when representing a BF f is known. Denoting this maximum for a BF f as z^f_{max}, and the number of eliminated monomials by an agent is z^f_{agent}, then the performance metric for an n-variable BF f, p_f, can be defined with $p_f = z^f_{agent}/z^f_{max}$ ($2^n > z^f_{max} \geq z^f_{agent} \geq 0$). For testing the performance of the agents, we separated the set of all 4-variable BFs into test and training sets. We calculated the performances of the agents for each BF. Then, we computed the average performance for both sets.

2.3.1 Benchmark of the DQN agent against Rand$_v$ agents

The 75% of all functions in 4 dimension (49152 out of 65536) are uniformly randomly selected and set as the training set; the rest is set as the test set. We used the same training and test sets for the DQN agent and Rand$_v$ agents. To account for the stochasticity in Rand$_v$ agents the experiments are repeated 10 times and the average is reported.

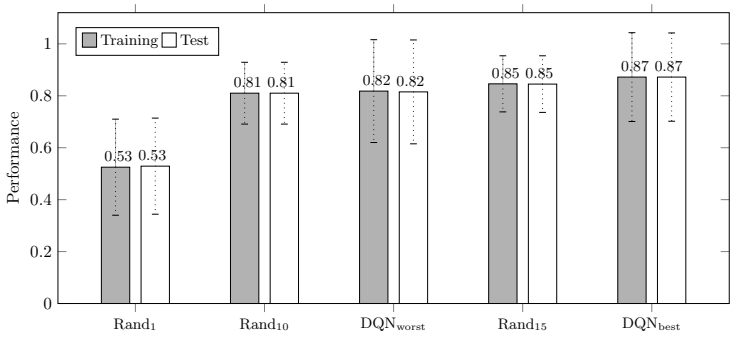

Fig. 1: The monomial elimination performances of the DQN and Rand$_v$ agents

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

For the DQN results, we conducted 6 training sessions. The worst DQN agent obtained is found to be better than the Rand_1 and Rand_{10}. The best DQN agent outperforms the Rand_1, Rand_{10}, and Rand_{15} agents (Fig. 1). If v is large enough, Rand_v agent will outperform the DQN agents, obviously. In addition to this general performance comparison, we deepened the analysis by investigating how the agents perform with respect to individual equivalence classes. For this, we grouped the BFs in the training and test sets by their equivalence classes and calculated their performance averages as shown in Table 1.

Equiv. Class	# Functions	DQN-train	DQN-test	Rand_1-test	Rand_1-train	Rand_{15}-train	Rand_{15}-test
0xaa55	32	0.69 ± 0.14	0.72 ± 0.06	0.47 ± 0.3	0.47 ± 0.28	0.93 ± 0.06	0.93 ± 0.06
0xab55	512	0.96 ± 0.09	0.96 ± 0.06	0.62 ± 0.32	0.61 ± 0.33	0.99 ± 0.02	0.99 ± 0.02
0xbb55	3840	0.82 ± 0.15	0.82 ± 0.15	0.51 ± 0.24	0.51 ± 0.24	0.89 ± 0.08	0.89 ± 0.08
0xaba5	17920	0.92 ± 0.14	0.92 ± 0.14	0.54 ± 0.21	0.54 ± 0.21	0.91 ± 0.09	0.91 ± 0.09
0xaaff	1120	0.82 ± 0.09	0.82 ± 0.1	0.44 ± 0.22	0.45 ± 0.22	0.84 ± 0.1	0.84 ± 0.1
0xaba4	26880	0.88 ± 0.16	0.87 ± 0.16	0.52 ± 0.17	0.52 ± 0.17	0.83 ± 0.1	0.83 ± 0.1
0xab12	14336	0.83 ± 0.21	0.82 ± 0.21	0.52 ± 0.13	0.51 ± 0.13	0.76 ± 0.09	0.76 ± 0.09
0xac90	896	0.81 ± 0.14	0.82 ± 0.14	0.78 ± 0.14	0.78 ± 0.15	0.99 ± 0.04	0.99 ± 0.04

Table 1: DQN agent equivalence class based statistics

The results indicate that Rand_{15} is better than the DQN agent for some of the equivalence classes. However, the DQN agent's overall performance is better. In particular, the DQN agent performs better for the equivalence classes denoted by 0xaba5, 0xaba4, and 0xab12. These classes dominate the function space as they makeup of 90% of all the 4-variable BFs. So, this means DQN experience more of these types of functions, and thus gains better knowledge about them. Consequently, we may expect it to perform better on these functions as exploration is key for RL. As expected DQN agent is superior to the Rand_1 which has only one shot at getting a right subset. However, still, interestingly the Rand_1 agent performs closely on one equivalence class (0xac90). Functions in this class are *bent functions*. They seem to have the maximum *PTF density* among all other BFs [6]. In fact, the Rand_{15} agent could find the maximum possible for almost all bent functions. These findings indicate that a large number of subsets can be eliminated when bent functions are considered, as evidenced by the fact that random agents can find such subsets in relatively few trials.

2.3.2 Equivalence class generalization of the DQN agent

An interesting question is how well the DQN agent performs on the BFs from an equivalence class it has never experienced during its learning. To answer this question we design the training set so as to include all functions from an equivalence class except the one equivalence class that we wish to check for generalization performance. Thus the test set is composed of only functions from one equivalence class function that are not included in the training set. Consequently, we conducted 8 training and test sessions in total as there are 8 equivalence classes in the space of 4-variable BFs. Although there are variations in the generalization performance depending on the selected equivalence class, overall, the results show the DQN agent can build a policy that allows it to

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

select monomials for minimization for unseen BFs from an unseen equivalence class (Table 2).

Equiv. Class	Walsh spec. summary	DQN performance on Train	DQN performance on Test
0xaa55	1x16, 15x0	0.66 ± 0.169	0.638 ± 0.194
0xab55	1x14, 15x2	0.915 ± 0.15	0.92 ± 0.172
0xbb55	1x12, 7x4, 8x0	0.809 ± 0.178	0.823 ± 0.152
0xaba5	1x10, 3x6, 12x2	0.899 ± 0.154	0.889 ± 0.186
0xaaff	4x8, 12x0	0.812 ± 0.152	0.688 ± 0.228
0xaba4	2x8, 8x4, 6x0	0.842 ± 0.179	0.732 ± 0.187
0xab12	6x6, 10x2	0.8 ± 0.207	0.652 ± 0.169
0xac90	16x4	0.782 ± 0.146	0.758 ± 0.136

Table 2: Test performance indicate performance for equivalence classes in Column 1.

3 Conclusion

The current report focuses on the question: can reinforcement learning help us solve and better understand the minimum term sign representing polynomial problem. The results found are promising. Our experiments show that the DQN agents do not overfit and generalize the solutions for the BFs that are not present in the training phase. Moreover, the other performance criterion addressed, i.e., the equivalence class generalization performance shows that a DQN agent can extract the minimization pattern of an equivalence class from the minimization pattern information of the others. This is a surprisingly good generalization performance. The ongoing work is underway addressing 5 and 6 variable BFs. The future work will address the utilization of the knowledge discovered by RL agents (in terms of neural network weights) for a better understanding of the nature of BF sign-representation.

References

[1] Barrett et al, Exploratory Combinatorial Optimization with Reinforcement Learning, *CoRR*, abs/1909.04063, 2019

[2] C. R. Edwards, The Application of the Rademacher-Walsh Transform to Boolean Function Classification and Threshold Logic Synthesis, in *IEEE Transactions on Computers*, vol. C-24, no. 1, pp. 48-62, Jan. 1975

[3] Dai et al, Learning Combinatorial Optimization Algorithms over Graphs, *CoRR*, abs/1704.01665, 2017

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra D. and Riedmiller M. A., Playing Atari with Deep Reinforcement Learning, *CoRR*, abs/1312.5602, 2013

[5] Oztop, E., Sign-representation of Boolean functions using a small number of monomials. *Neural Networks*, 22(7):938-948, Elsevier, 2009

[6] Sezener, CE, Oztop. E. Minimal Sign Representation of Boolean Functions: Algorithms and Exact Results for Low Dimensions, *Neural Computation* 27(8): 1796-1823, 2015.

[7] Siu, K. Y., V. Roychowdhury and T. Kailath. *Discrete Neural Computation*. Englewood Cliffs, NJ, Prentice Hall, 1995.

[8] Yapar, O., Oztop, E., On the Co-absence of Input Terms in Higher Order Neuron Representation of Boolean Functions. In Cong, F., Leung, A., Wei, Q., editors, proceedings of the 14^{th} *International Symposium on Neural Networks* (ISNN 2017). Lecture Notes in Computer Science 10262, pages 362-370, Springer, 2017.