CZECH TECHNICAL UNIVERSITY IN PRAGUE

MULTIDIMENSIONAL FUNCTION APPROXIMATION FOR LOWER CONVEX ENVELOPE COMPUTATION

A study of neural networks as a mechanism for computing the lower convex envelope f(x) of a set of points Υ, where x is in d dimensions

Presented by José Hilario
Supervised by Mgr. Branislav Bošanský, Ph.D.

Summer 2018/2019

INTRODUCTION

Considering the Heuristic Search Value Iteration (HSVI) [1] algorithm for Partially Observable Markov Decision Processes (POMDPs), and, more recently, its adaptation for One-Sided Partially Observable Stochastic Games (OS-POSGs) [2], we investigate enhancements that would make such algorithms a viable option in larger applications.

Particularly, in HSVI for OS-POSGs [2] the upper bound of the value function $\overline{v}$ is represented by a set of points Υ. Each new update adds a new point to Υ, where each point in Υ corresponds to the value backup operator $[H\overline{v}](b)$ (defined in [2]) at belief b, which is the value of the so-called stage game, where players choose their Nash Equilibrium (NE) strategies for one stage of the game. Then, a point-based projection to the lower convex envelope (LCE) [3] of Υ is obtained, which represents the upper bound function $\overline{v}$. This step is computationally costly, and presents an overhead which makes the algorithm impractical for larger scenarios.

The idea herein explored is to replace with a neural network the explicit computation of this projection. The goal is to have an adaptive system usable in two modes: 1) query; and, 2) update. In query mode the system will take as input a belief b and will output upper bound values corresponding to $\overline{v}(\mathbf{b})$ which is given by the LCE of Υ. In update mode the system will take as input a belief b, and output the predicted upper bound value $\overline{v}(\mathbf{b})$ corresponding to it (exactly the same as in query mode); then, it will compare its predicted value with the value of the stage game $[H\overline{v}](\mathbf{b})$, which is taken here as a ground truth that will drive an update of the parameters of the network.

We focus in this introductory investigation in assessing the capacity of standard fully connected neural networks, to approximate the function $\overline{v}$, first, as a standard regression task; and second, through the computation of the LCE, using the points Υ.

RELATED WORK

Solving linear equations has been shown to be possible using neural networks [4]. Also, the approximation of a value function in POMDPs can be obtained directly [5, 6]. The first option was not considered for our experiments due to the fact that it possibly would not convey an improvement in efficiency of computation. The second option was discarded because it would get us outside of the spectrum of "improving" HSVI, which is our main interest.

Papers were investigated on the topic of multidimensional function approximation [7, 9], and of convex functions in particular [8]. Notions about the proper activation to use, and the tradeoff between width and depth in a fixed width network, were considered through the development of the experiments. We also found among these papers, network design guidelines based on width and depth relationship, depending on the dimensionality of the input, specifically suited for convex function approximation [8]. This is considered in the last part of the experiments.

Research of literature was carried out on existing methods to compute the convex hull of a set of points in high dimensions [11, 12, 13], in particular, using neural networks [10]. Among these options is included an adaptive convex hull computation using neural networks, which corresponds to the HSVI use case. The main limitation found in these options was in terms of the time complexity, which was exponential in the input dimensions. Since we are interested in dealing with an input with dimensionality of up to $d = 10000$, these options had to be discarded.

Finally, research was also carried out on adaptive, online, computations of the convex hull [14] in particular, using neural networks [15] that could be implemented in the latter stages of our investigations, to completely replace the upper bound LCE computation in HSVI-based algorithms. The investigations for the online learning scenario started being carried out for the general supervised learning setting [16].

PROPOSED METHOD

At this initial stage of research we will focus on examining the accuracy of the function approximation considering the standard offline supervised learning setting.

We consider as input a belief:

$$\mathbf{b} = (b_1, \cdots, b_n) \triangleq (b(s_1), \cdots, b(s_n))$$

such that:

$$\sum_{s \in \mathcal{S}} b(s) = 1$$

We are interested in evaluating for a state space of up to $|\mathcal{S}| = 10000$. Therefore we will assess the performance while incrementing the state space, paying attention to possible limitations encountered, given by the high dimensionality of our problem.

In the first scenario we will have a training set $\mathcal{T} = (\mathbf{b}^1, \overline{v}(\mathbf{b})^1), \cdots, (\mathbf{b}^m, \overline{v}(\mathbf{b})^m))$. The idea is to learn the function directly, without considering the LCE.

In the second scenario we will have a training set $\mathcal{T} = (\mathbf{b}{v_\Upsilon}^1, \overline{v}(\mathbf{b})^1), \cdots, (\mathbf{b}{v_\Upsilon}^m, \overline{v}(\mathbf{b})^m))$, where the input now contains values from Υ which need to be projected by the network into the LCE. Here we want to use the values in Υ as part of the input. This scenario should convey some idea of the capacity of the neural network to approximate the LCE function. Intuitively, the values in Υ already provide an approximation to the LCE, which means that the network should converge faster than in the first scenario. Let us remark that we will be considering an offline training phase with arbitrarily set v_Υ (actually, randomly generated uniformly under some allowed margin). As the algorithm [1, 2] progresses, however, the value backup obtains values which are closer to the optimal upper bound $\overline{v}$. Thus, we conjecture that the randomness component in the artificial scenario will result in worse accuracy of the network, than the real scenario.

EXPERIMENTS

We will approximate a function $\tilde{v}$ where each of its components $\tilde{v}_i$ is convex, and, as such, it follows that $\tilde{v}$ is also convex:

$$\tilde{\mathbf{v}} = \tilde{v}_1(b_1) + \cdots + \tilde{v}_n(b_n)$$

The range $\mathcal{R}(\mathbf{v})$ should be considered when assessing the resulting average L1 losses below. The density range for the generated beliefs was sampled uniformly from , i.e., we are considering examples at the input that span the whole range of sparsity levels.

$$\tilde{v}_i = \frac{1}{10} \cdot i^2 \cdot b_i^2 - \frac{1}{5} \cdot i \cdot b_i + 10 \cdot [\![i = 1]\!], \; \forall i = \{1, \cdots, n\}$$
$$\forall \tilde{\mathbf{v}} \in \Upsilon : \tilde{\mathbf{v}} \in [1, 1.5] \cdot \mathbf{v}$$
$$\text{density of beliefs} = [0.1, 0.99]$$

	$\mathcal{R}(\mathbf{v})$	**Input dimension**	**Output dimension**	**Width**	**Layers**	**LCE**
1	[-50, 500]	1000	1	400	10	False
2	[-50, 500]	1000	1	400	10	True
3	[-50, 500]	1000	1	500	14	False
4	[-50, 500]	1000	1	500	14	True
	Optimizer	**Training examples**	**Epochs**	**Batch size**	**Testing examples**	**Avg Test L1 loss**
1	Adam	10000	300	100	10000	17.5591
2	Adam	10000	300	100	10000	6.4570
3	Adam	60000	300	2000	10000	16.7806
4	Adam	60000	300	2000	10000	6.3647

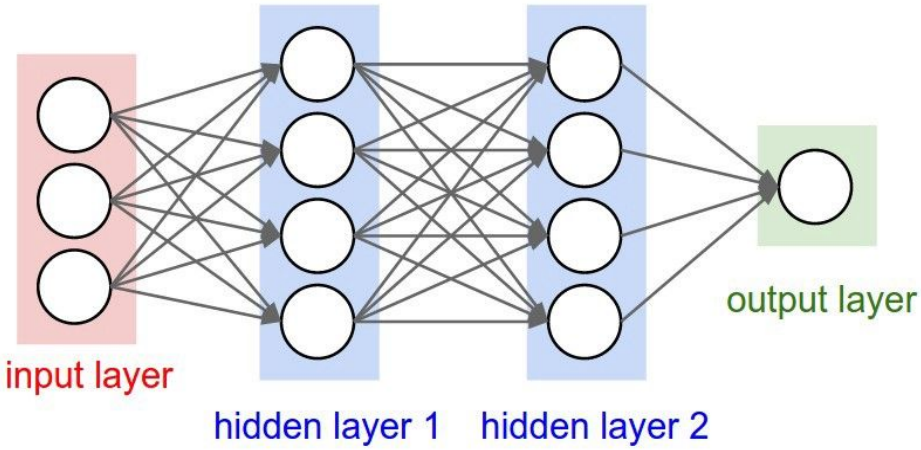

Figure 1: Basic neural network architecture. The amount of hidden layers can be increased through the configuration. The network is of fixed width, and uses PReLU activations. As optimizer, Adaptive Moment Estimation (Adam) performed better than Stochastic Gradient Descent (SGD) in high dimensions, and in the task of LCE. When using SGD, we do so with a learning rate of 0.001 and a momentum of 0.9.

$$\tilde{v}_i = \frac{1}{100} \cdot i^2 \cdot b_i^2 - \frac{1}{50} \cdot i \cdot b_i + 10 \cdot [\![i = 1]\!],\ \forall i = \{1, \cdots, n\}$$

$$\forall \tilde{\mathbf{v}} \in \Upsilon : \tilde{\mathbf{v}} \in [1, 1.5] \cdot \mathbf{v}$$

$$\text{density of beliefs} = [0.01, 0.2]$$

	$\mathcal{R}(\mathbf{v})$	**Input dimension**	**Output dimension**	**Width**	**Layers**	**LCE**
5	[60, 3000]	5000	1	400	10	False
6	[60, 3000]	5000	1	400	10	True
	Optimizer	**Training examples**	**Epochs**	**Batch size**	**Testing examples**	**Avg Test L1 loss**
5	Adam	40000	500	200	10000	113.0728
6	Adam	40000	500	200	10000	32.6046

We now show the results obtained by using the bounds established in [8] as guidelines for the design of the network architecture. We start from lower dimensions, and increment it consecutively, in order to see the effect it produces when we use these bounds as guidelines.

$$\tilde{v}_i = \frac{40}{1+i} \cdot b_i^2 - \frac{k}{1+i} \cdot b_i + \frac{20}{1+i^2},\ \forall i = \{0, \cdots, n-1\}$$
$$\forall \tilde{\mathbf{v}} \in \Upsilon : \tilde{\mathbf{v}} \in [1, 1.5] \cdot \mathbf{v}$$
$$\text{density of beliefs} = [1, 1]$$

	k	$\mathcal{R}(\mathbf{v})$	**Dim in**	**Dim out**	**Width**	**Layers**	**LCE**
7	**120**	[-30, 40]	3	1	4	0	False
8	**120**	[-30, 40]	3	1	4	0	True
9	**120**	[60, 100]	10	1	11	0	False
10	**120**	[60, 100]	10	1	11	0	True
11	**1200**	[-100, 200]	100	1	101	0	False
12	**8000**	[-120, 230]	500	1	501	0	False
	Optimizer		**Training**	**Epochs**	**Batch**	**Testing**	**Val L1 loss**
7	SGD		10	1500	1	10000	1.0212
8	Adam		10	1500	1	10000	2.0452
9	SGD		100	1500	1	10000	0.4890
10	Adam		100	1500	1	10000	6.3545
11	SGD		1000	1500	10	10000	1.5564
12	SGD		3000	1500	10	10000	4.5910
12	SGD		2000	1500	10	10000	14.7985

CONCLUSION

The results obtained in this investigation suggest that the upper bound of the value function used for HSVI can be approximated using a neural network. When learning a multivariate (high dimensional) convex function in the offline scenario we achieved a good accuracy, which could be improved upon by fine tuning: trying out some changes in the architecture of the network, and introducing some practically verified methods that improve the overall performance of neural networks in regression tasks.

Also, when learning the LCE from beliefs and points from the upper bound cloud Υ, in high dimensions, the function was approximated with more accuracy, and faster convergence.

The future steps to take include: 1) generate data from the actual games in order to better assess the quality of the approximation network; 2) to create a scenario that will appropriately simulate the process of the HSVI algorithm wherein the upper bound cloud of points Υ will sequentially be updated with new values to which we have to adapt the output of the network, that will correspond to the LCE, since new points added to Υ push down the LCE. A scenario of interest would be to sort the training set for the LCE approximation task, such that the examples with highest difference between $\tilde{\mathbf{v}}$ and $\overline{\mathbf{v}}$ are processed first. Then we would examine the result given by the network for just one epoch, processing the images one by one, which appropriately simulates aspects of the real scenario.

Another option is to implement the query mode of the network so that while it is processing examples one by one, updating the parameters of the network, we can at each step ask the network to randomly select some belief, such that we compare the real value of $\overline{v}$ with the output given by the network at that moment.

REFERENCES

[1] Trey Smith and Reid Simmons. 2004. Heuristic search value iteration for POMDPs. In Proceedings of the 20th conference on Uncertainty in artificial intelligence (UAI '04). AUAI Press, Arlington, Virginia, United States, 520-527.

[2] Horák, K., Bošanský, B., & Pěchouček, M. (2017). Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. AAAI Conference on Artificial Intelligence, North America, feb. 2017.

[3] Karel Horák and Branislav Bošanský. 2016. A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In 7th International Conference on Decision and Game Theory for Security - Volume 9996 (GameSec 2016), Quanyan Zhu, Tansu Alpcan, Emmanouil Panaousis, Milind Tambe, and William Casey (Eds.), Vol. 9996. Springer-Verlag New York, Inc., New York, NY, USA, 435-454.

[4] E. K. P. Chong, S. Hui and S. H. Zak, "An analysis of a class of neural networks for solving linear programming problems," in IEEE Transactions on Automatic Control, vol. 44, no. 11, pp. 1995-2006, Nov. 1999.

[5] Peter Karkus, David Hsu, and Wee Sun Lee. 2017. QMDP-Net: deep learning for planning under partial observability. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Ulrike von Luxburg, Isabelle Guyon, Samy Bengio, Hanna Wallach, and Rob Fergus (Eds.). Curran Associates Inc., USA, 4697-4707.

[6] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2017. Value iteration networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), Carles Sierra (Ed.). AAAI Press 4949-4953.

[7] Liang, S., & Srikant, R. (2017). Why Deep Neural Networks for Function Approximation? ICLR.

[8] Hanin, B. 2017. Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations. arXiv:1708.02691.

[9] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," in IEEE Transactions on Neural Networks, vol. 16, no. 1, pp. 24-38, Jan. 2005.
doi: 10.1109/TNN.2004.836233

[10] Leung, Yee & Zhang, Jiang-She & xu, Zongben. (1997). Neural networks for convex hull computation. Neural Networks, IEEE Transactions on. 8. 601 - 611. 10.1109/72.572099.

[11] H. R. Khosravani, A. E. Ruano and P. M. Ferreira, "A simple algorithm for convex hull determination in high dimensions," 2013 IEEE 8th International Symposium on Intelligent Signal Processing, Funchal, 2013, pp. 109-114.

[12] Antonio Ruano, Hamid Reza Khosravani, Pedro M. Ferreira. A Randomized Approximation Convex Hull Algorithm for High Dimensions. IFAC-PapersOnLine, Volume 48, Issue 10, 2015, Pages 123-128, ISSN 2405-8963.

[13] C. Bradford Barber, David P. Dobkin, and David P. Dobkin, Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. 22, 4 (December 1996), 469-483.

[14] Avraham A. Melkman. 1987. On-line construction of the convex hull of a simple polyline. Inf. Process. Lett. 25, 1 (April 1987), 11-12.

[15] Pal, S & Datta, A & Pal, Nikhil. (2001). A multilayer self-organizing model for convex-hull computation. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 12. 1341-7. 10.1109/72.963770.

[16] Lakhmi C. Jain, Manjeevan Seera, Chee Peng Lim, and P. Balasubramaniam. 2014. A review of online learning in supervised neural networks. Neural Comput. Appl. 25, 3-4 (September 2014), 491-509.