ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Ensembles of Extreme Learning Machine Networks for Value Prediction

Pablo Escandell-Montero, José M. Martínez-Martínez,
Emilio Soria-Olivas, Joan Vila-Francés, José D. Martín-Guerrero *

IDAL, Intelligent Data Analysis Laboratory, University of Valencia
Av. de la Universidad, s/n, 46100, Burjassot, Valencia - Spain

Abstract. Value prediction is an important subproblem of several reinforcement learning (RL) algorithms. In a previous work, it has been shown that the combination of least-squares temporal-difference learning with ELM (extreme learning machine) networks is a powerful method for value prediction in continuous-state problems. This work proposes the use of ensembles to improve the approximation capabilities of ELM networks in the context of RL.

1 Introduction

Situated in between supervised learning and unsupervised learning, the paradigm of reinforcement learning (RL) deals with solving sequential decision-making problems in which there is limited feedback [1]. The RL problem is usually formulated as a Markov decision process (MDP). A discounted MDP is defined by the tuple $\{S, A, P, \rho, \gamma\}$ where S is the state space of the process, A is the action space, P the transition probability function $P : S \times A \times S \to [0, 1)$, $\rho : S \times A \times S \to \mathbb{R}$ the reward function, and γ is the discount factor. Value prediction is an important subproblem of several RL algorithms that consists of learning the value function V^π of a given policy π [1]. When the state space is discrete and small enough, value functions can be stored in tables with one entry per state; however, in the general case, the state space is continuous and the value function must be represented approximately.

A popular method for value prediction in continuous-state problems is least-squares temporal-difference (LSTD) learning [2]. In LSTD, value functions are assumed to be represented with linear architectures, which limits the kind of approximators that can be used. Given a state s, its value $V^\pi(s)$ is approximated by first mapping s to a feature vector $\phi(s) \in \mathbb{R}^k$, and then computing a linear combination of those features: $\phi(s)^\top \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is the weight vector (to be learned). The accuracy of the approximated value function depends in part on the features employed; thus, selecting a suitable feature space is a crucial stage of LSTD [3]. Generally, there is no prior knowledge about the shape of the value function and it is not possible to define an ad-hoc feature space. The most common procedure consists of partitioning the state space in a regular set of features employing, for example, state aggregation methods or radial basis function (RBF) networks with fixed bases. These techniques are local approximators, which are less powerful than global ones (e.g. artificial neural networks (ANNs)

*This work has been partially funded by SMARTPIF project (FP7-SME-2012, 312573)

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

or support vectors machines) [4]. In [5] we proposed the LSTD-ELM algorithm, a method based on the extreme learning machine (ELM) [6] that allowed the use of single-hidden layer feed-forward networks (SLFNs) to approximate value functions in LSTD learning.

LSTD-ELM has shown to provide better accuracy and scalability properties for high dimensional problems than other LSTD methods based on local approximators [5]. These benefits are due to the global nature of SLFNs. However, similar to other techniques based on ELM, the quality of the results may vary in different trials due to the randomness introduced by the ELM algorithm. Several authors have applied ensemble methods in ELM to alleviate this problem and enhance the approximation accuracy and stability [7, 8, 9]. This paper explores the use of ensembles in LSTD-ELM for value prediction problems.

2 Extreme Learning Machine

Extreme learning machine (ELM) is an algorithm for training SLFNs where the weights of the hidden layer can be initialized randomly, thus being only necessary the optimization of the output layer weights by means of standard least-square methods [6].

Let us consider a set of N patterns, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{o}_i); i = 1 \dots N\}$, where $\{\mathbf{x}_i\} \in \mathbb{R}^{d_1}$ and $\{\mathbf{o}_i\} \in \mathbb{R}^{d_2}$, so that the goal is to find a relationship between $\mathbf{x}_i$ and $\mathbf{o}_i$. If there are M nodes in the hidden layer, the SLFN's output for the j-th pattern is:

$$\mathbf{y}_j = \sum_{k=1}^{M} h_k \cdot f(\mathbf{w}_k, \mathbf{x}_j) \tag{1}$$

where $1 \leq j \leq N$, $\mathbf{w}_k$ stands for the parameters (weights and biases) of the k-th element of the hidden layer, h_k is the weight that connects the k-th hidden element with the output layer and f is the function that gives the output of the hidden layer; in the case of MLP, f is an activation function applied to the scalar product of the input vector and the hidden weights. Equation (1) can be expressed in matrix notation as $\mathbf{y} = \mathbf{G} \cdot \mathbf{h}$, where $\mathbf{h}$ is the vector of weights of the output layer, $\mathbf{y}$ is the output vector and $\mathbf{G}$ is given by:

$$\mathbf{G} = \begin{pmatrix} f(\mathbf{w}_1, \mathbf{x}_1) & \dots & f(\mathbf{w}_M, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1, \mathbf{x}_N) & \cdots & f(\mathbf{w}_M, \mathbf{x}_N) \end{pmatrix} \tag{2}$$

Then, the weights of the output layer can be computed as $\mathbf{h} = \mathbf{G}^{-1} \cdot \mathbf{o}$ using the Penrose-Moore pseudoinverse to invert $\mathbf{G}$ robustly.

3 LSTD-ELM algorithm

Despite the powerful approximation capabilities of ANNs, they can not be used to approximate value functions due to technical restrictions imposed by LSTD. ELM theory introduces a training algorithm for SLFNs that differs from traditional methods based on iterative procedures (such as gradient-descent or global

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

search). On the contrary, ELM transforms the training process into a system of linear equations that can be solved analytically. LSTD-ELM exploits this property to allow the use of SLFNs in a LSTD learning scheme.

The LSTD-ELM algorithm starts by initializing the parameters $\mathbf{w}$ of the hidden nodes. Then, these nodes are used to define the feature space $\phi(s) \in \mathbb{R}^M$, where M is the size of the hidden layer and each hidden node is equivalent to a feature. After an observed trajectory of L states and rewards, $(s_0, r_0, \ldots, s_L, r_L)$, the following data structures are used to build from experience the matrix $\mathbf{A}$ (of dimension $M \times M$) and the vector $\mathbf{b}$ (of dimension M):

$$\mathbf{b} = \sum_{i=0}^{L} \boldsymbol{\phi}(s_i) r_i; \qquad \mathbf{A} = \sum_{i=0}^{L} \boldsymbol{\phi}(s_i)(\gamma \boldsymbol{\phi}(s_{i+1}) - \boldsymbol{\phi}(s_i))^\top \tag{3}$$

Then, the output layer weights can be computed as $\mathbf{h} = \mathbf{A}^{-1} \cdot \mathbf{b}$.

It has been shown that, after enough independent trajectories and under some technical conditions, LSTD-ELM achieves a good approximation of the value functon, $V^\pi(s) \approx \phi(s)^\top \mathbf{h}$ [5].

An issue with LSTD-ELM is that as some parameters are randomly assigned and remain unchanged during the training process, they can be non-optimum and the approximation performance may be degraded. This problem is intrinsic to any method based on the ELM algorithm. Some authors have proposed the use of ensemble techniques to reduce the possible negative effects of randomness in ELM. In the next section, an extension of LSTD-ELM based on ensembles is proposed.

4 LSTD based on ELM ensembles

An ensemble is a method that consists of taking a combination of several models to form a single new model. It is known that combining suboptimal models is an effective and simple strategy to improve the performance of each one of the combination members. There are different ways to combine the output of several models. In the ELM context, for example, there exist ensembles based on bagging [7], AdaBoost [8] or evolutionary algorithms [9]. These methods use the desired output to optimize the way in which individual models are combined. However, they cannot be applied in the RL paradigm due to the absence of an explicit desired signal.

Another option to form a single model from several individual models is by simply taking the average of their outputs. This technique, which was used here, does not require a desired signal. In particular, three different methods to compute the average were tested: mean, median and trimmed mean. The trimmed mean involves the calculation of the mean after discarding some samples located at the extremes of the distribution [10]. Although the three methods are similar, the last two are more robust to the presence of outliers.

The Algorithm 1 shows the pseudo-code of LSTD based on an ELM ensemble, denoted by LSTD-eELM. Given a policy π, the discount factor γ and the number c of ensemble members, LSTD-eELM starts assigning randomly the hidden layer

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Algorithm 1 LSTD algorithm based on ELM ensemble

Input: Policy π to be evaluated, discount factor γ, number of ensemble members c
Initialize randomly $\mathbf{w}_j; j = 1, \ldots, c$, the hidden layer parameters of the c ELM networks
Let $\phi_j(\mathbf{x}) : \mathbf{x} \rightarrow f(\mathbf{w}_j, \mathbf{x})$ denote the mapping from an input $\mathbf{x}$ to the output of the SLFN's hidden layer
Set the c matrices $\mathbf{A}$ and vectors $\mathbf{b}$ to 0; $t = 0$
repeat
  Select a start state s_t
  while $s_t \neq s_{\text{END}}$ **do**
    Apply policy π to the system, producing a reward r_t and next state s_{t+1}
    for $j = 1, \ldots, c$ **do**
      $\mathbf{A}_j = \mathbf{A}_j + \phi_j(s_t)(\gamma\phi_j(s_{t+1}) - \phi_j(s_t))^\top$
      $\mathbf{b}_j = \mathbf{b}_j + \phi_j(s_t)r_t$
    end for
    $t = t + 1$
  end while
until reaching the desired number of trayectories
$\mathbf{h}_j = \mathbf{A}_j^{-1}\mathbf{b}_j$
output $V^\pi(s) \approx combineMembers(\boldsymbol{\phi}, \mathbf{h})$

parameters of the c ELM networks, $\mathbf{w}_j; j = 1, \ldots, c$. Afterwards, it builds an individual matrix $\mathbf{A}$ and vector $\mathbf{b}$ for each network. Then, the output layer parameters can be computed as $\mathbf{h}_j = \mathbf{A}_j^{-1}\mathbf{b}_j$. The function *combineMembers()* computes and combines the output of the c networks using one of the averaging methods like the mean or the median.

5 Experimental study

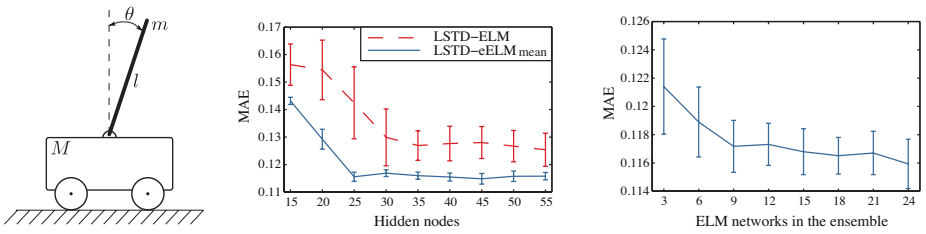
The performance of the proposed method was evaluated on the inverted pendulum problem, a classical benchmark for approximate RL. This problem consists of a rigid pole mounted on a mobile cart (see Fig. 1a). The cart is free to move in a one-dimensional track and the pole is free to move only in the vertical plane of the cart and track. The mass (m) and length (l) of the pendulum are unknown to the agent. The agent can apply three actions to the cart: left force (-50 N), right force (50 N), or no force (0 N). Each action is corrupted by uniformly distributed noise in the range $[-10, 10]$. The dynamics of the inverted pendulum is governed by the following nonlinear equation [3]:

$$\ddot{\theta} = \frac{g \cdot \sin(\theta) - \alpha \cdot m \cdot l \cdot (\dot{\theta})^2 \cdot \sin(2\theta)/2 - \alpha \cdot \cos(\theta) \cdot a}{4 \cdot l/3 - \alpha \cdot m \cdot l \cdot \cos^2(\theta)} \tag{4}$$

where g is the gravity constant ($g = 9.8$ m/s^2), m is the mass of the pendulum ($m = 2$ Kg), M is the mass of the cart ($M = 8$ Kg), l is the length of the pendulum ($l = 0.5$ m), a is the control action, and $\alpha = 1/(m + M)$. The state space of the problem consisted of the vertical angle θ and the angular velocity $\dot{\theta}$ of the pendulum, which was bounded to [-5,5] rad/s. An angle greater (in absolute value) than $\pi/2$ indicated the end of the trajectory.

Trajectories start with a state randomly selected from the subset defined by the ranges $\theta = [-\pi/2, \pi/2]$ and $\dot{\theta} = [-5, 5]$. The agent receives a positive reward equal to 1 as long as the pendulum is balanced, i.e., $|\theta| < \pi/2$. Otherwise, the reward received is 0. While the usual goal is to learn a policy that balances the

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.



(a) (b) (c)

Fig. 1: (a) Inverted pendulum problem. (b) Comparison between LSTD-eELM$_{\text{mean}}$ and the standard LSTD-ELM. (c) Performance of LSTD-eELM$_{\text{mean}}$ for different ensemble sizes.

pole, we want to estimate the value function for a given policy. Thus, it was fixed a random policy that selects the three actions with the same probability.

There are two parameters in LSTD-eELM that should be tuned: the number of hidden nodes of the ensemble members, M, and the number of members, c. The effects of varying both parameters were studied in two experiments. In the first experiment, c was fixed to 18 and the number of hidden nodes was varied from 15 to 55 in steps of 5. This process was repeated with the three versions of LSTD-eELM, each one using a different method (mean, median and trimmed mean) to combine the ensemble members. For comparison purposes, the same procedure was performed with the original LSTD-ELM. In the second experiment, M was fixed to 35 and the number of members was varied from 3 to 24 in steps of 3.

In both experiments, γ was fixed to 0.9 and the performance was measured in terms of the mean absolute error (MAE). Similar to [2, 5], MAE was measured against a "gold standard" value function, V_{MC}, built using Monte Carlo simulation [1] on a representative set of discrete states.

5.1 Results

The results of all experiments were computed as the mean values and standard deviations from 30 independent trials. Table 1 summarizes the results of the first experiment. For each size of the hidden layer, the minimum MAE is highlighted in bold. As it can be observed, the proposed method provided the best results in all cases. Among the three versions, LSTD-eELM$_{\text{mean}}$ obtained the best approximation in 7 of the 9 sizes tested. However, in general, the three LSTD-eELM versions achieved similar results. Fig. 1b compares graphically the performance of LSTD-eELM$_{\text{mean}}$ and the standard LSTD-ELM. For this case, the proposed method reduced (in average) the MAE by 11.09% and the standard deviation by 77.22%.

The results of the second experiment are shown in Fig. 1c. As expected, the MAE and standard deviation exhibit a decreasing tendency, which suggests that the quality of the approximation can be improved by adding more members to the ensemble. On the other hand, it should be noted that the computational

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

#Nodes	LSTD-ELM	LSTD-eELM		
		Mean	Median	Trimmed mean
15	0.1563 (0.007)	**0.1432** (0.001)	0.1442 (0.002)	0.1436 (0.001)
20	0.1544 (0.011)	0.1292 (0.004)	0.1314 (0.004)	**0.1290** (0.004)
25	0.1424 (0.013)	0.1156 (0.002)	0.1165 (0.002)	**0.1154** (0.002)
30	0.1299 (0.010)	**0.1169** (0.001)	0.1174 (0.001)	0.1171 (0.001)
35	0.1269 (0.005)	**0.1159** (0.001)	0.1167 (0.001)	0.1163 (0.001)
40	0.1276 (0.006)	**0.1155** (0.001)	0.1164 (0.002)	0.1158 (0.001)
45	0.1280 (0.006)	**0.1148** (0.002)	0.1156 (0.002)	0.1151 (0.002)
50	0.1267 (0.006)	**0.1157** (0.002)	0.1165 (0.002)	0.1160 (0.002)
55	0.1254 (0.006)	**0.1158** (0.001)	0.1162 (0.001)	0.1159 (0.001)

Table 1: Performance of LSTD-ELM and the three versions of LSTD-eELM for different sizes of the hidden layer. Performance is measured in terms of MAE; standard deviation is provided within parentheses. The best result for each architecture is highlighted in bold.

complexity of the algorithm grows linearly with the number of ensemble members.

6 Conclusions

In this paper, we have proposed to use ensembles of ELM networks for value prediction in continuous-state problems. Although ELM networks have demonstrated to be a powerful function approximator in RL, their accuracy and stability can be further improved by combining several networks. The resulting algorithm, called LSTD-eELM, has been empirically evaluated in the well-known inverted pendulum benchmark problem. Our experimental results suggest that the proposed method outperforms LSTD-ELM.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[2] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, November 2002.

[3] Michail G Lagoudakis, Ronald Parr, and L. Bartlett. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:2003, 2003.

[4] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2007.

[5] Pablo Escandell, José M. Martínez, José D. Martín, Emilio Soria, and Juan Gómez. Least-squares temporal difference learning based on extreme learning machine. In *21 ESANN*, pages 233—238, Belgium, 2013.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, December 2006.

[7] Huixin Tian and Bo Meng. A new modeling method based on bagging ELM for day-ahead electricity price prediction. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 1076–1079, 2010.

[8] Hui-Xin Tian and Zhi-zhong Mao. An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace. *IEEE Transactions on Automation Science and Engineering*, 7(1):73–80, 2010.

[9] Dianhui Wang and Monther Alhamdoosh. Evolutionary extreme learning machine ensembles with size control. *Neurocomputing*, 102:98–110, February 2013.

[10] Bradley Efron and Robert Tibshirani. Statistical data analysis in the computer age. *Science (New York, N.Y.)*, 253(5018):390–395, July 1991.