ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Functional Gradient Descent for n-Tuple Regression

Rafael F. Katopodis[1], Priscila V.M. Lima[1,2] and Felipe M.G. França[1] *

1- COPPE, 2- NCE, Universidade Federal do Rio de Janeiro - Brazil

Abstract. n-tuple neural networks have been in the past applied to a wide range of learning domains. However, for the particular area of regression, existing systems have displayed two shortcomings: little flexibility in the objective function being optimized and an inability to handle nonstationarity in an online learning setting. A novel n-tuple system is proposed to address these issues. The new architecture leverages the idea of functional gradient descent, drawing inspiration from its use in kernel methods. Furthermore, its capabilities are showcased in two reinforcement learning tasks, which involves both nonstationary online learning and task-specific objective functions.

1 Introduction

Weightless, or n-tuple, neural networks are general learning models that have been successfully employed in a variety of areas of learning [1]. Based on the idea of memory elements as fundamental building blocks, these networks were originally proposed for pattern recognition tasks [2], but later extended into other domains, such as regression, in the form of the n-tuple regression network (NTRN) [3] and derived models [4].

The existing NTRN-based architectures have shown promise in multiple complex tasks but face two shortcomings that limit their application. The first is that they are incapable of tracking nonstationary targets in an online learning setting. The second is that they are fundamentally connected to a particular error metric and unable to optimize different objective functions.

The structure of the present text is as follows: Section 2 points out the root cause of the shortcomings in the NTRN and presents concepts from kernel methods that can be leveraged to surpass the limitations listed. Section 3 presents a new n-tuple model that makes use of these concepts. Section 4 showcases the model in tasks of policy search for reinforcement learning (RL), a domain that gives rise to both nonstationary online learning and task-dependent objective functions, and the paper is concluded in Section 5.

2 Fundamental concepts

2.1 n-Tuple regression networks

The n-tuple regression network is a model suitable for the approximation of real-valued functions [3]. Unlike other RAM-based systems that precede it, such

*The authors thank CAPES and CNPq for financial support for this work.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

as the *WiSARD*, for pattern recognition, or the SLLUP [5], for regression, the NTRN is set apart by its storage of two values, a weight and a counter in each memory position, and by its connection to kernel-based models.

For a set of examples $\{(\mathbf{x}_j, y_j)|j = 1, \ldots, T\}$ sampled from the joint distribution $p(X, Y)$, the NTRN forms its estimate, f, by implicitly implementing a Nadaraya-Watson kernel regression which, in turn, is an approximation of the conditional expectation of Y (1), where k is a kernel function. This expectation is the estimator that minimizes the mean squared error [6].

$$f(\mathbf{x}) = \frac{\sum_j y_j k(\mathbf{x}_j, \mathbf{x})}{\sum_j k(\mathbf{x}_j, \mathbf{x})} \approx \mathbb{E}[Y|X = \mathbf{x}] \tag{1}$$

This form of nonparametric regression is capable of representing highly nonlinear targets. The use of the NTRN allows for training and predicting in constant time instead of proportional to the number of samples, as would be the case if an explicit kernel representation were adopted.

The approximation of the conditional estimator, however, brings upon the network two limitations. First, is the assumption that the metric of interest to be optimized is the mean squared error. This can be contrasted with the flexibility afforded by learning models that make use of gradient descent.

The second limitation concerns online learning. Although the NTRN is naturally trained in a sample by sample manner, it cannot properly handle target function drift. The root cause being that training samples are all equally taken into account in the minimization of the mean squared error.

2.2 Functional gradient descent in kernel methods

Kernel methods, particularly for real-valued function approximation, are learning models of the form (2), where $\mathbf{x}_j$ are points from training data, α_j are weights learned from it and k is a kernel, denoting a similarity relationship between two inputs. These methods are deeply rooted in statistical learning theory and have been used in a variety of domains [7]. Albeit generally used in a batch learning context, where all training data is known *a priori* and the target function is assumed stationary, kernel machines can also fit the online learning paradigm.

$$f(\mathbf{x}) = \sum_j \alpha_j k(\mathbf{x}_j, \mathbf{x}) \tag{2}$$

One way this can be done is by gradient descent in the space of functions that can be represented as (2). Let $\mathcal{R}[f, \mathbf{x}_t, y_t] = c(\mathbf{x}_t, y_t, f(\mathbf{x}_t))$ be a risk functional defined in terms of the loss of the model f for a sample $(\mathbf{x}_t, y_t)$. The method consists of iteratively updating the model according to the functional gradient $\nabla_f \mathcal{R}$ (3). This gradient is defined as the linear term in an approximation of the change in the value of $\mathcal{R}$ due to a perturbation in the model (4).

The last equality in (3) makes use of the chain rule, the evaluation functional $E_{\mathbf{x}}[f] = f(\mathbf{x})$ and its gradient $\nabla_f E_{\mathbf{x}}[f] = k(\mathbf{x}, \cdot)$ to relate the gradient of the risk functional to the partial derivative of the loss (5). With this last result, the

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

method may be understood as the iterative addition to the kernel expansion of the model of kernels centered over newly arrived samples, also known as basis functions, weighted by the gradient of the loss function.

$$\begin{aligned} f_{t+1}(\mathbf{x}) &= f_t(\mathbf{x}) + \eta_t \nabla_f \mathcal{R}[f, \mathbf{x}_t, y_t] \\ &= f_t(\mathbf{x}) + \eta_t \frac{\partial}{\partial z} c(\mathbf{x}_t, y_t, z)|_{f(\mathbf{x}_t)} k(\mathbf{x}_t, \mathbf{x}) \end{aligned} \tag{3}$$

$$\mathcal{R}[f + \epsilon g, \mathbf{x}_j, y_j] = \mathcal{R}[f, \mathbf{x}_j, y_j] + \epsilon \langle \nabla_f \mathcal{R}[f, \mathbf{x}_j, y_j], g \rangle + \mathcal{O}(\epsilon^2) \tag{4}$$

$$\begin{aligned} \nabla_f \mathcal{R}[f, \mathbf{x}_t, y_t] &= \frac{\partial}{\partial z} c(\mathbf{x}_t, y_t, z)|_{f(\mathbf{x}_t)} \nabla_f E_{\mathbf{x}_t}[f] \\ &= \frac{\partial}{\partial z} c(\mathbf{x}_t, y_t, z)|_{f(\mathbf{x}_t)} k(\mathbf{x}_t, \cdot) \end{aligned} \tag{5}$$

One shortcoming of this approach is the increase in time and space complexity as more basis functions are added to the model. The NORMA algorithm [8] handles this issue by truncation. Besides a loss, the risk functional also contains a regularization term which, during an update step, results in the shrinking of the weights of previously added basis functions. Based on these adaptive weights, the algorithm only retains a fixed number of the most significant basis.

3 The functional gradient-inspired n-tuple network

The proposed model's architecture follows along the lines of the RAM discriminator, used by WiSARD-based systems. It consists of a single layer of N RAM neurons with vector-valued memory positions and takes as input an $n \times N$-long binary vector, called a *retina*. Each neuron is connected to n positions of the retina, with such connections being established randomly and having no overlap.

We denote by $\tau^{[i]}(\mathbf{x})$ the tuple formed by input $\mathbf{x}$ for neuron i and by $v_t^{[i]}(\mathbf{x})$ the value addressed by the tuple formed by $\mathbf{x}$ for neuron i after t training samples. $v_0^{[i]}(\mathbf{x}) = 0, \forall \mathbf{x} \in \mathcal{X}, i = 1, \ldots, N$, implying that all memory positions are initialized with zeros. For a sample $(\mathbf{x}_{t+1}, y_{t+1})$, a differentiable loss function $c : \mathcal{X} \times \mathbb{R}^2 \mapsto \mathbb{R}$ and a learning rate η_t, the model is updated according to the rule (6) while its output after t training samples is computed as (7).

$$v_{t+1}^{[i]}(\mathbf{x}_{t+1}) = v_t^{[i]}(\mathbf{x}_{t+1}) + \eta_t \delta_t, \qquad i = 1, \ldots, N \tag{6}$$

$$\text{where } \delta_t = \frac{\partial}{\partial z} c(\mathbf{x}_{t+1}, y_{t+1}, z)\big|_{f_t(\mathbf{x}_{t+1})}$$

$$f_t(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} v_t^{[i]}(x) \tag{7}$$

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

It is straightforward to show that the model proposed implements a kernel machine. The demonstration closely mirrors the one given in [3]. First, the content of a memory position at any point in time may be stated in terms of all samples up to that time (8). $M_t^{[i]}$, here, serves as an indicator variable denoting whether $\mathbf{x}$ addresses the same memory position as $\mathbf{x}_t$ in neuron i.

$$v_t^{[i]}(\mathbf{x}) = \sum_{j=1}^{t} \eta_j \delta_j M_j^{[i]}(\mathbf{x}) \qquad M_j^{[i]}(\mathbf{x}) = \begin{cases} 1 & \text{if } \tau^{[i]}(\mathbf{x}) = \tau^{[i]}(\mathbf{x}_j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\sum_{i=1}^{N} M_t^{[i]}(\mathbf{x}) = N - \rho(\mathbf{x}, \mathbf{x}_t) = N\left(1 - \frac{\rho(\mathbf{x}, \mathbf{x}_t)}{N}\right) \tag{9}$$

Furthermore, a relationship may be established between the indicator and a measure of similarity between inputs (9), where $\rho(a, b) = \sum_{i=1}^{N} [\![\tau^{[i]}(a) \neq \tau^{[i]}(b)]\!]$ is the tuple distance between two inputs (and $[\![\text{predicate}]\!]$ denotes the Iverson bracket). Now, (6), (8), and (9) may be substituted into (7) to derive an equation that relates the model's output to a kernel:

$$\begin{aligned} f_t(\mathbf{x}) &= \frac{1}{N} \sum_{i=1}^{N} v_t^{[i]}(\mathbf{x}) \\ &= \frac{1}{N} \sum_{i=1}^{N} \left[v_{t-1}^{[i]}(\mathbf{x}) + \eta_{t-1} \delta_{t-1} M_{t-1}^{[i]}(\mathbf{x}) \right] \\ &= f_{t-1}(\mathbf{x}) + \frac{\eta_{t-1}\delta_{t-1}}{N} \sum_{i=1}^{N} M_{t-1}^{[i]}(\mathbf{x}) \\ &= f_{t-1}(\mathbf{x}) + \eta_{t-1}\delta_{t-1} k(\mathbf{x}_{t-1}, \mathbf{x}), \quad k(\mathbf{x}, \mathbf{x}_t) = \left(1 - {}^{\rho(\mathbf{x}, \mathbf{x}_t)}/_{N}\right) \end{aligned} \tag{10}$$

Equation (10) highlights how the network's output is a summation of basis functions, of the form (2). Moreover, the output at time t relates to the output at $t-1$ in the same way as (3). The update rule, therefore, implicitly performs functional gradient descent.

4 A reinforcement learning application

One application domain that involves both nonstationary online learning and the optimization of particular objective functions is policy search in online RL. That is the search for a probability distribution of actions conditioned on a state that maximizes the long-term cumulative reward attained by an agent.

The proposed n-tuple system may be employed in this context by parameterizing a suitable probability distribution for the task at hand. For tasks with binary action spaces (actions are either -1 or $+1$), one possible representation makes use of the sigmoid function (11). For continuous action spaces, a natural

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

choice is the gaussian distribution (12), where the model determines the mean vector. $\mathbf{\Sigma}$ is a predefined covariance matrix and $1/Z$ is a normalizing constant.

$$\pi_f^{\text{binary}}(a|s) = \frac{1}{1 + e^{\text{sign}(a) f(s)}} \tag{11}$$

$$\pi_f^{\text{continuous}}(a|s) = \frac{1}{Z} \exp\left\{ -\frac{1}{2}(f(s) - a)^\intercal \mathbf{\Sigma}^{-1}(f(s) - a) \right\} \tag{12}$$

The goal of searching for good policies can be framed as that of adjusting the learning model that parametrizes the policy to maximize the value of the initial state. This maximization can be performed numerically by gradient ascent and the algorithms that do so are called policy gradient methods. One such method is the REINFORCE algorithm [9], previously adapted to the functional gradient setting in kernel machines [10] yielding the update rule (13). G_t is the return from an episodic interaction of the agent with its environment.

$$f_{t+1} = f_t + \eta_t G_t \nabla_f \ln \pi_f(a|s) \tag{13}$$

As is shown in the previous section, the proposed n-tuple network implicitly performs functional gradient descent. Therefore, it can realize the update rule (13) by adopting $\mathcal{R}[f, a_t, s_t] = -G_t \ln \pi_f(a_t|s_t)$ as the empirical risk objective to be minimized. For the binary and continuous action spaces policies, the functional gradient of the log-probabilities are given by (14) and (15).

$$\nabla_f \ln \pi_f^{\text{binary}}(a_t|s_t) = -\text{sign}(a_t)(1 - \pi_f(a_t|s_t))k(s_t, \cdot) \tag{14}$$

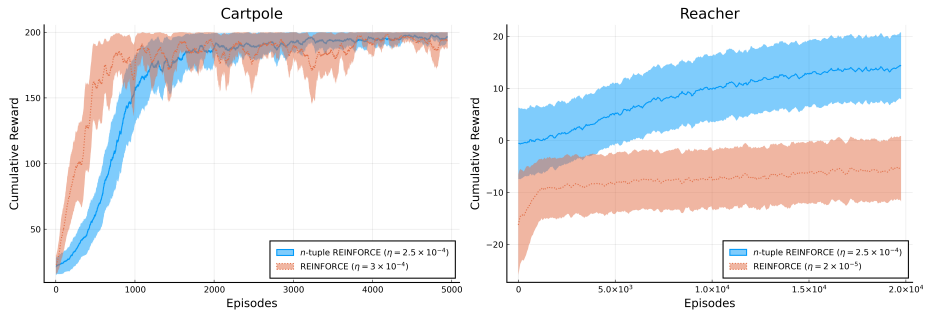
$$\nabla_f \ln \pi_f^{\text{continuous}}(a_t|s_t) = \mathbf{\Sigma}^{-1}(a_t - f(s_t))k(s_t, \cdot) \tag{15}$$

Two benchmark tasks from the Bullet physics engine [11] were used to showcase the forms of policies parameterized by an n-tuple network: *Cartpole*, with binary scalar actions, and *reacher*, with continuous vector actions. The cumulative rewards attained by this approach throughout training, together with baseline curves, can be seen in Figure 1. The baselines were obtained using standard REINFORCE and multilayer perceptrons with two hidden layers, 32 ReLU units each. Recordings of the trained agents can be found in [12].

5 Conclusion

A new and more flexible n-tuple network architecture for regression has been proposed. By leveraging the idea of functional gradient descent, this model can be applied to nonstationary online learning problems with differential objective functions, such as policy search in RL. Building upon the foundation of the NTRN, this new architecture inherits the property of inferring values in constant time. Explicit kernel methods, on the other hand, would have to resort to techniques such as truncation to avoid an inference time-complexity linear in the number of training samples. Finally, the network was showcased with two

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.



(a) (b)

Fig. 1: Learning curves for the cartpole and reacher tasks, averaged across 10 runs with distinct random initializations. Curves are smoothed with a simple moving average, with a 50-point window in (1a) and 200-point in (1b). Shaded areas denote the 95% confidence interval. Learning rates provided in the legend.

policy search tasks. Ongoing work is being done in comparing the proposed approach for RL to established ones, along dimensions such as sample efficiency and stability. Future research could explore the use of adaptive learning rates and curvature information to accelerate convergence.

References

[1] I. Aleksander, M. De Gregorio, F. M. G. França, P. M. V. Lima, and H. Morton. A brief introduction to weightless neural systems. In *ESANN*, 2009.

[2] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Papers presented at the 1959 eastern joint IRE-AIEE-ACM computer conference, IRE-AIEE-ACM 1959 (Eastern), Boston, Massachusetts, USA, December 1-3, 1959*.

[3] A. Kolcz and N. M. Allinson. N-tuple regression network. *Neural Networks*, 9(5), 1996.

[4] L. L. Filho, L. F. R. Oliveira, A. L. Filho, G. P. Guarisa, L. M. Felix, P. M. V. Lima, and F. M. G. França. Extending the weightless wisard classifier for regression. *Neurocomputing*, 416:280–291, 2020.

[5] G. D. Tattersall, S. Foster, and R. D. Johnston. Single-layer lookup perceptrons. In *IEE Proceedings F (Radar and Signal Processing)*, volume 138, pages 46–54. IET, 1991.

[6] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to probability, 2nd Edition*. Athena Scientific Belmont, MA, 2008.

[7] A. J Smola and B. Schölkopf. *Learning with kernels*. MIT Press, 2002.

[8] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

[9] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[10] J. A. Bagnell and J. Schneider. Policy search in reproducing kernel hilbert space. Technical Report CMU-RI-TR-03-45, Carnegie Mellon University, Pittsburgh, PA, November 2003.

[11] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016–2019. http://pybullet.org.

[12] Recordings of the trained agents on the cartpole and reacher environments. https://youtube.com/playlist?list=PL0k075xYpLenHAv9E6riQgnZX1nX6ONv8.