Engineering Applications of Artificial Intelligence 68 (2018) 10–17

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Supervised kernel approach for automated learning using General Stochastic Networks

D. Cárdenas-Peña [a,*], D. Collazos-Huertas [a], A. Álvarez-Meza [b], G. Castellanos-Dominguez [a]

[a] Signal Processing and Recognition Group, Universidad Nacional de Colombia, Manizales, Colombia
[b] Automatics Research Group, Universidad Tecnologica de Pereira, Pereira, Colombia

ARTICLE INFO

Keywords:
Generative Stochastic Network
Kernel-based learning
Supervised learning

ABSTRACT

Generative Stochastic Networks (GSN) for supervised tasks generalize the denoising autoencoders by fixing the deepest layer to the output variables (e.g. class) and estimate the input–output joint distribution as the stationary transition operator of a Markov chain. Because of multi-layer network architectures with stochastic neurons, GSN performance depends on the selected architecture and network training. Aiming to improve such a performance, we introduce a supervised kernel-based learning within a GSN framework. Firstly, the considered network model induces a temporal model working as a data filtering that extracts refined data representations. Then, we use the conventional exhaustive search strategy to fix the hidden layer size. Lastly, we propose a novel supervised layer-wise pre-training that initializes the fine tuning stage of the GSN with more discriminative projection matrices favoring the optimization of the non-convex cost function. Initial matrices are computed by maximizing the centered-kernel alignment (CKA) metric, measuring the affinity between projected samples and labels. We evaluate the proposal performance in comparison with Random, AutoEncoders, and Principal Component Analysis approaches. As a result, CKA-based pre-training approach captures the complex dependencies between parameters, increases the convergence speed in the learning stage, and unravels the data distribution to favor the class discrimination for five widely image collections used in classification tasks of image object recognition.

1. Introduction

Generative Stochastic Networks (GSN) are a deep neural network architecture based on learning the transition distribution of a Markov chain for estimating the data generating distribution, i.e., generative learning. The combination of noise, a multi-layer, feed-forward neural network, and walk back training makes GSN simplify the learning problem, be less like density estimation, and resemble more a supervised function approximation, with gradients that can be obtained by back-probable stochastic units at each layer (Alain et al., 2016). This training principle for generative probabilistic models has gained increasing interest in signal processing, pattern recognition, and machine learning fields, confirming their representational power (Huang, 2015; Bengio et al., 2013). However, to bring accurate results, deep neural networks require correct data pre-processing, architecture selection (Bergstra and Bengio, 2012), and network training (Panchal et al., 2011). The first aspect deals with learning data representations extracting more useful information, for which the generative stochastic model sequentially filters input samples. Each transition provides additional refined data that also feed the estimation of the generating distribution. The second one influences the performance of used training algorithm through different factors (Sheela and Deepa, 2013): (i) input and output dimensions, (ii) number of training samples, and (iii) noise injection method. Aiming to provide a wider field of new applications for GSN, however, some network training issues must be deeper investigated. Particularly, a relevant aspect is how to manage the non-convexity of the training criterion for the parameter space searching.

For deep architectures the training criterion is non-convex and involves many local minima, getting worse for architectures with more than two or three levels (Erhan et al., 2009). Although a straightforward procedure to cope with this issue is the use of multiple random initializations, it highly increases the time consumption (Bengio, 2012). A more efficient random initialization is to establish the distribution and range of network parameters either empirically or to assume the characteristics of hidden units. In the latter case, the Glorot-style normalized initialization ensures that each neuron operates in the active region of its saturating function so that the forward and backward

* Corresponding author.
E-mail address: dcardenasp@unal.edu.co (D. Cárdenas-Peña).

https://doi.org/10.1016/j.engappai.2017.10.003
Received 28 April 2016; Received in revised form 5 July 2017; Accepted 9 October 2017
0952-1976/

propagated variances are layer-wise fixed (Glorot and Bengio, 2010). Yet, this approach promotes the fast parameter saturation during training of complex problems, decreasing the system performance (Glorot and Bengio, 2010). With the purpose of avoiding the saturation of dense initializations, the sparse initialization technique (SI) randomly connects neurons of consecutive layers, draws the weights from a unit Gaussian, and sets the biases to zero (Martens, 2010). Despite allowing the use of second order optimizations, SI is highly sensitive to the established activation function and scale constant, slowing down the learning speed (Sutskever et al., 2013).

To improve the initialization random approaches, the results obtained in Erhan et al. (2010) suggest that unsupervised pretraining guides the learning towards basins of attraction of minima that support better generalization from the training dataset. The layer-wise pre-training is the most used approach for finding a suitable initialization, aiming to extract a useful higher-level description of the output of the preceding layer of representation. To this end, a greedy layer-wise stage of unsupervised learning is firstly carried out, followed by a fine, supervised tuning (Hinton et al., 2006). Particularly, this pre-training has been used for parameter optimization in Bengio et al. (2007), using the restricted Boltzmann machines as building blocks (Bengio et al., 2007), and in Vincent et al. (2008) for estimating the parameters of stacked denoising autoencoders (Vincent et al., 2008). As a result, either supervised learning machine regularizes the fine tuning, depending on the architecture depth and layer size (Erhan et al., 2010). Nonetheless, the greedy principle underachieves if the conditional output distribution is not accurately associated with the input structure (Bengio et al., 2007). On the other hand, some approaches have been discussed to boost the learning speed. Thus, the salient features can be extracted by independent component analysis (ICA) to initialize the first multi-layer perceptron (MLP) layer though it yields to local minimum solutions (Chen and Lu, 2013). In contrast to the layer-wise approaches, the parameters learned simultaneously by a multilayer generative network can also be employed to initialize the training of a supervised feed-forward network, increasing the classification accuracy as discussed in Mohamed et al. (2011). Thus, the contractive regularization for pre-training two-layered auto-encoders forces the system to have small derivatives on the inputs, outperforming greedy methods in data generalization and classification accuracy (Schulz et al., 2015). In general terms, the above discussed unsupervised pre-training approaches generate more useful hidden representations than the input space, but many of the resulting features may be irrelevant for discrimination tasks (Weston et al., 2012).

Here, we introduce a supervised kernel-based learning within a framework of General Stochastic Networks to cope with the network topology issues above-described. Firstly, an exhaustive heuristic search is conducted to select the optimal architecture based on the network performance measure. Then, we propose a pre-training approach that reduces the influence of the non-convexity by finding the parameters increasing the class separability at each layer. To this end, we make use of the Centered Kernel Alignment (CKA) criterion that assesses the similarity of two distributions from their characterizing kernels. The first kernel is built layer-wise from the latent samples and the second kernel corresponds to the target distribution from the supervised information. Hence, maximizing the CKA with respect to each projection matrix results in a pre-trained network that sequentially highlights the discriminative information from the input samples to the output labels. The validation is carried out on datasets for object classification and shows that CKA-based pre-training improves both the learning speed and the classification accuracy.

The agenda of this paper is organized as follows: Initially, we describe the mathematical framework and our proposed approach in Section 2. Then, Section 3 illustrates the results of carried out evaluations on six well-known datasets. Lastly, we discuss all obtained results and provide conclusions in Sections 4 and 5, respectively.

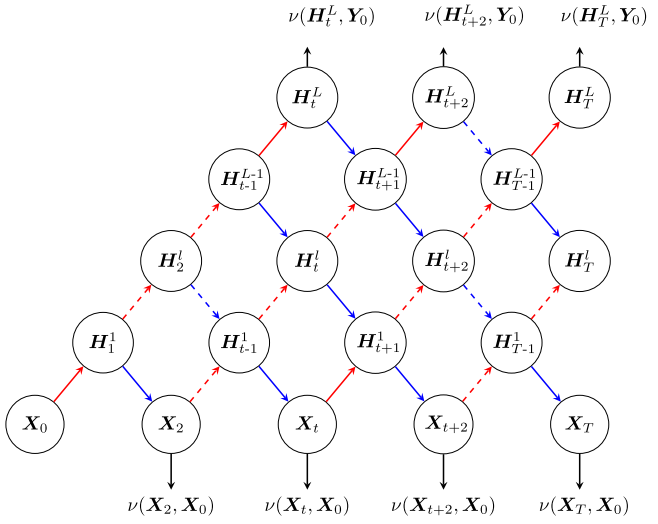

Fig. 1. Schematic representation of a GSN Markov chain with backprop-able stochastic units. – Upward step. – Downward step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2. Materials and methods

2.1. General Stochastic Networks for supervised learning

Provided the input distribution $P(\mathcal{X})$ for which we only have empirical samples $\boldsymbol{X} \subset \mathcal{X}$, General Stochastic Networks (GSN) combine multilayer perceptrons with backprop-able stochastic neurons, noisy propagations, and walkback training in estimating the corresponding transition operator of a Markov chain as generative learning approach. Fig. 1 illustrates a GSN Markov chain for a deep network with L layers, where $\boldsymbol{X}_t \in \mathbb{R}^{N \times P}$ is the input matrix sampled at time instants $t \in [1, \ldots, T]$, $\boldsymbol{H}_t^l \in \mathbb{R}^{N \times m_l}$ is the lth hidden state matrix at a time step t and $m_l \in \mathbb{N}$ is the size of lth layer so that $l \in [1, \ldots, L]$ and $T > L$. Matrix $\boldsymbol{H}_t^l$ holds N vectors $\boldsymbol{h}_{t,i}^l \in \mathbb{R}^{m_l}$, mapping the input samples to the layer l.

Further, the dependency of available latent variables (hidden states) $\boldsymbol{H}_t^l$ is encoded into a GSN graph through the following set of upward/downward iterations.

$$\begin{cases} \boldsymbol{H}_t^l = \varsigma_{out} + \left(\phi^l(\boldsymbol{b}^l + \boldsymbol{H}_{t-1}^{l+1} (\boldsymbol{W}^l)^\top + \varsigma_{in}) + \phi^l(\boldsymbol{a}^l + \boldsymbol{H}_{t-1}^{l-1} (\boldsymbol{W}^l) + \varsigma_{in}) \right) \\ \boldsymbol{H}_t^0 = \boldsymbol{X}_t \end{cases} \tag{1}$$

where $\boldsymbol{b}^l \in \mathbb{R}^{m_l}$ and $\boldsymbol{a}^l \in \mathbb{R}^{m_{l-1}}$ are the offset vectors, $\boldsymbol{W}^l \in \mathbb{R}^{m_{l-1} \times m_l}$ is the lth linear projection, vectors $\varsigma_{out} \in \mathbb{R}^{N \times m_l}$ and $\varsigma_{in} \in \mathbb{R}^{N \times m_{l-1}}$ are independent noise sources, and the function $\phi^l(\cdot) \in \mathbb{R}$ applies saturating, non-linear, element-wise operations.

We will define a dGSN, i.e. discriminative–generative stochastic network that has L layers for classification through the following cost function (Zöhrer and Pernkopf, 2014):

$$\upsilon(\mathcal{X}, \mathcal{Y}) = \log(P(\mathcal{X})) + \log(P(\mathcal{Y}|\mathcal{X})). \tag{2}$$

Note that the unsupervised learning of $\log(P(\mathcal{X}))$ is usually used for semi-supervised learning if the labeled data is scarce. In this case, $P(\mathcal{Y}|\mathcal{X})$ is introduced as the conditional probability distribution between the input $\mathcal{X}$ and the output $\mathcal{Y}$ to make GSN suitable for a supervised learning task. In addition, both distributions are sampled to guarantee convergence, namely, $\boldsymbol{X} \subset \mathcal{X}$ with $\boldsymbol{X} \in \mathbb{R}^{N \times P}$, holds N input vectors $\boldsymbol{x}_i \in \mathbb{R}^P$ $(i \in N)$ and $\boldsymbol{Y} \subset \mathcal{Y}$ with $\boldsymbol{Y} \in [0, 1]^{N \times C}$ that contains N output vectors $\boldsymbol{y}_i \in [0, 1]^C$ representing C mutually exclusive classes. Hence, the last layer is fixed to the output dimension, i.e. $m_L = C$.

Due to the target $\boldsymbol{Y}$ cannot be forward propagated through the graph network, it is instead introduced in the cost function so that the Markov