Neurocomputing 74 (2011) 1840–1847

Contents lists available at ScienceDirect

Neurocomputing

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

Dynamic self-organising map

Nicolas Rougier [a,*], Yann Boniface [b]

[a] LORIA/INRIA Nancy – Grand Est Research Centre, 54600 Villers-lès-Nancy, France
[b] LORIA/Université Nancy 2, 54015 Nancy Cedex, France

ARTICLE INFO

Available online 1 March 2011

Keywords:
Self-organisation
On-line
Cortical plasticity
Dynamic

ABSTRACT

We present in this paper a variation of the self-organising map algorithm where the original time-dependent (learning rate and neighbourhood) learning function is replaced by a time-invariant one. This allows for on-line and continuous learning on both static and dynamic data distributions. One of the property of the newly proposed algorithm is that it does not fit the magnification law and the achieved vector density is not directly proportional to the density of the distribution as found in most vector quantisation algorithms. From a biological point of view, this algorithm sheds light on cortical plasticity seen as a dynamic and tight coupling between the environment and the model.

1. Introduction

Vector quantisation (VQ) refers to the modelling of a probability density function into a discrete set of prototype vectors (sometimes called the codebook) such that any point drawn from the associated distribution can be associated to a prototype vector. Most VQ algorithms try to match the density through the density of their codebook: high density regions of the distribution tend to have more associated prototypes than low density region. This generally allows to minimise the loss of information (or distortion) as measured by the mean quadratic error. For a complete picture, it is to be noted that there also exists some cases where only a partition of the space occupied by the data (regardless of their density) is necessary. In this case, one wants to achieve a regular quantification *a priori* of the probability density function. For example, in some classification problems, one wants to achieve a discrimination of data in terms of classes and thus needs only to draw frontiers between data regardless of their respective density.

Vector quantisation can be achieved using several methods such as variations of the *k*-means method [1], Linde–Buzo–Gray (LBG) algorithm [2] or neural network models such as the self-organising map (SOM) [3], neural gas (NG) [4] and growing neural gas (GNG) [5]. Among all these methods, the SOM algorithm is certainly the most famous in the field of computational neurosciences since it can give a biologically and plausible account on the organisation of receptive fields in sensory areas where adjacent neurons share similar representations. The stability and quality of such self-organisation depend heavily on a decreasing learning rate as well as on a decreasing neighbourhood function. This is quite congruent with the idea of a critical period in the early years of development where most sensory or motor properties are acquired and stabilised [6–8]. However, this fails to explain cortical plasticity since we know that the cortex has the capacity to re-organise itself in face of lesions or deficits [9–11]. The question is then to know to what extent it is possible to have both stable and dynamic representations?

Quite obviously, this cannot be achieved using SOM-like algorithms that depend on a time decreasing learning rate and/or neighbourhood function (SOM,NG,GNG) and, despite the huge amount of literature [12,13] around self-organising maps and Kohonen-typed networks (more than 7000 works listed in [14]), there is surprisingly and comparatively very little work dealing with online learning (also referred as incremental or lifelong learning). Furthermore, most of these works are based on incremental models, that is, networks that create and/or delete nodes as necessary. For example, the modified GNG model [15] is able to follow non-stationary distributions by creating nodes like in a regular GNG and deleting them when they have a too small *utility* parameter. Similarly, the evolving self-organising map (ESOM) [16,17] is based on an incremental network quite similar to GNG that creates dynamically based on the measure of the distance of the winner to the data (but the new node is created at exact data point instead of the mid-point as in GNG). Self-organising incremental neural network (SOINN) [18] and its enhanced version (ESOINN) [19] are also based on an incremental structure where the first version is using a two layers network while the enhanced version proposed a single layer network. One noticeable result is the model proposed by [20] which does not rely on an incremental structure but is based on the Butterworth decay scheme that does not decay parameters to zero. The model works in two phases, an initial phase (approximately 10 epochs)

* Corresponding author.
E-mail address: Nicolas.Rougier@loria.fr (N. Rougier).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.06.034

is used to establish a rough global topology thanks to a very large neighbourhood and the second phase uses a small neighbourhood phase to train the network. Unfortunately, the size of the neighbourhood in the second phase has to be adapted to the expected density of the data.

Without judging performances of these models, we do not think they give a satisfactory answer to our initial question and we propose instead to answer by considering a tight coupling between the environment and representations. If the environment is stable, representations should remain stable and if the environment suddenly changes, representations must dynamically adapt themselves and stabilise again onto the new environment. We thus modified the original SOM algorithm in order to make its learning rule and neighbourhood independent of time. This results in a tight coupling between the environment and the model that ensure both stability and plasticity. In the next section, we formally describe the dynamic self-organising map in the context of vector quantisation and both neural gas and self-organising map are formally described in order to underline differences between the three algorithms. The next section re-introduces the model from a more behavioural point of view and main experimental results are introduced using either low or high dimensional data and offers side-to-side comparison with other algorithms. Results concerning dynamic distributions are also introduced in the case of dynamic self-organising map in order to illustrate the coupling between the distribution and the model. Finally, we discuss the relevancy of such a model in the context of computational neurosciences and embodied cognition.

2. Definitions

Let us consider a probability density function $f(x)$ on a compact manifold $\Omega \in \mathbb{R}^d$. A vector quantisation (VQ) is a function Φ from Ω to a finite subset of n code words $\{\mathbf{w}_i \in \mathbb{R}^d\}_{1 \le i \le n}$ that form the codebook. A cluster is defined as $C_i \stackrel{\text{def}}{=} \{x \in \Omega | \Phi(x) = \mathbf{w}_i\}$, which forms a partition of Ω and the distortion of the VQ is measured by the mean quadratic error

$$\xi = \sum_{i=1}^{n} \int_{C_i} \|x - \mathbf{w}_i\|^2 f(x)\, dx. \tag{2.1}$$

If the function f is unknown and a finite set $\{x_i\}$ of p non-biased observations is available, the distortion error may be empirically estimated by

$$\hat{\xi} = \frac{1}{p} \sum_{i=1}^{n} \sum_{x_j \in C_i} \|x_j - \mathbf{w}_i\|^2. \tag{2.2}$$

Neural maps define a special type of vector quantifiers whose most common approaches are the self-organising map (SOM) [3], elastic net (EN) [21], neural gas (NG) [4] and growing neural gas (GNG) [22]. In the following, we will use definitions and notations introduced by [23] where a neural map is defined as the projection from a manifold $\Omega \subset \mathbb{R}^d$ onto a set $\mathcal{N}$ of n *neurons* which is formally written as $\Phi : \Omega \to \mathcal{N}$. Each neuron i is associated with a code word $\mathbf{w}_i \in \mathbb{R}^d$, all of which established the set $\{\mathbf{w}_i\}_{i \in \mathcal{N}}$ that is referred as the codebook. The mapping from Ω to $\mathcal{N}$ is a closest-neighbour winner-take-all rule such that any vector $\mathbf{v} \in \Omega$ is mapped to a neuron i with the code $\mathbf{w}_\mathbf{v}$ being closest to the actual presented stimulus vector $\mathbf{v}$,

$$\Phi : \mathbf{v} \mapsto \arg\min_{i \in \mathcal{N}} (\|\mathbf{v} - \mathbf{w}_i\|). \tag{2.3}$$

The neuron $\mathbf{w}_\mathbf{v}$ is called the *winning element* and the set $C_i = \{x \in \Omega | \Phi(x) = \mathbf{w}_i\}$ is called the *receptive field* of the neuron i. The geometry corresponds to a Vorono diagram of the space with $\mathbf{w}_i$ as the centre.

2.1. Self-organising maps (SOM)

SOM is a neural map equipped with a structure (usually a hypercube or hexagonal lattice) and each element i is assigned a fixed position $\mathbf{p}_i$ in $\mathbb{R}^q$ where q is the dimension of the lattice (usually 1 or 2). The learning process is an iterative process between time $t=0$ and time $t = t_f \in \mathbb{N}^+$ where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$ at time t, a winner $s \in \mathcal{N}$ is determined according to Eq. (2.3). All codes $\mathbf{w}_i$ from the codebook are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon(t) h_\sigma(t,i,s)(\mathbf{v} - \mathbf{w}_i) \tag{2.4}$$

with $h_\sigma(t,i,j)$ being a neighbourhood function of the form

$$h_\sigma(t,i,j) = e^{-\|\mathbf{p}_i - \mathbf{p}_j\|^2 / 2\sigma(t)^2} \tag{2.5}$$

where $\varepsilon(t) \in \mathbb{R}$ is the learning rate and $\sigma(t) \in \mathbb{R}$ is the width of the neighbourhood defined as

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{t/t_f} \quad \text{with } \varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/t_f}, \tag{2.6}$$

while σ_i and σ_f are respectively the initial and final neighbourhood width and ε_i and ε_f are respectively the initial and final learning rate. We usually have $\sigma_f \ll \sigma_i$ and $\varepsilon_f \ll \varepsilon_i$.

2.2. Neural gas (NG)

In the case of NG, the learning process is an iterative process between time $t=0$ and time $t = t_f \in \mathbb{N}^+$ where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$ at time t, neurons are ordered according to their respective distance to $\mathbf{v}$ (closest distances map to lower ranks) and assigned a rank $k_i(\mathbf{v})$. All codes $\mathbf{w}_i$ from the codebook are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon(t) h_\lambda(t,i,\mathbf{v})(\mathbf{v} - \mathbf{w}_i) \tag{2.7}$$

with $h_\lambda(t,i,\mathbf{v})$ being a neighbourhood function of the form:

$$h_\lambda(t,i,\mathbf{v}) = e^{-k_i(\mathbf{v})/\lambda(t)} \tag{2.8}$$

where $\varepsilon(t) \in \mathbb{R}$ is the learning rate and $\lambda(t) \in \mathbb{R}$ is the width of the neighbourhood defined as

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{t/t_f} \quad \text{with } \varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/t_f}, \tag{2.9}$$

while λ_i and λ_f are respectively the initial and final neighbourhood and ε_i and ε_f are respectively the initial and final learning rate. We usually have $\lambda_f \ll \lambda_i$ and $\varepsilon_f \ll \varepsilon_i$.

2.3. Dynamic self-organising map (DSOM)

DSOM is a neural map equipped with a structure (a hypercube or hexagonal lattice) and each neuron i is assigned a fixed position $\mathbf{p}_i$ in $\mathbb{R}^q$ where q is the dimension of the lattice (usually 1 or 2). The learning process is an iterative process where vectors $\mathbf{v} \in \Omega$ are sequentially presented to the map with respect to the probability density function f. For each presented vector $\mathbf{v}$, a winner $s \in \mathcal{N}$ is determined according to Eq. (2.3). All codes $\mathbf{w}_i$ from the codebook $\mathbf{W}$ are shifted towards $\mathbf{v}$ according to

$$\Delta \mathbf{w}_i = \varepsilon \|\mathbf{v} - \mathbf{w}_i\|_\Omega h_\eta(i,s,\mathbf{v})(\mathbf{v} - \mathbf{w}_i) \tag{2.10}$$