

Paper:

# Self-Organizing Map with Generating and Moving Neurons in Visible Space

Kanta Tachibana and Takeshi Furuhashi

Dept. of Computational Science and Engineering, Graduate School of Engineering, Nagoya University

Furou-cho, Chikusa, Nagoya 464-8603, Japan

E-mail: kanta@fcs.coe.nagoya-u.ac.jp

[Received February 5, 2007; accepted March 20, 2007]

Kohonen's Self-Organizing feature Map (SOM) is used to obtain topology-preserving mapping from high-dimensional feature space to visible space of two or fewer dimensions. The SOM algorithm uses a fixed structure of neurons in visible space and learns a dataset by updating reference points in feature space. The mapping result depends on mapping parameters fixed, which are the number and visible positions of neurons, and parameters of learning, which are the learning rate, total iteration, and the setting of neighboring radii. To obtain a satisfactory result, the user usually must try many combinations of parameters. It is wasteful, however, to set up every possible combination of parameters and to repeatedly run the algorithm from the beginning because the computation cost for learning is large, especially for a large-scale dataset. These problems arise due to the fixing of two types of mapping parameters, i.e., the number and visible positions of neurons. The high computation cost is mainly in the calculation of distances from each sample to all reference points. At the beginning of learning, reference points should be adjusted globally to preserve the topology well because they are initially set far from optimal positions in feature space, e.g. randomly. Such many reference points subdivides feature space into unnecessarily fine Voronoi regions. To avoid this computational waste, it is natural to start learning with a small number of neurons and increase the number of neurons during learning. We propose a new SOM method that varies the number and visible positions of neurons, and thus is applicable also to visible torus and sphere spaces. We apply our proposal to spherical visible space. We use central Voronoi tessellation to move visible positions for two reasons: to tessellate visible space evenly for easy visualization and to level the number of neighboring neurons and better preserve topology. We demonstrate the effect of generating neurons to reduce computation cost and of moving visible positions in visualization and topology preservation.

**Keywords:** self-organizing map, extension of mapping parameter, spherical visible space

## 1. Introduction

An effective way to understand high-dimensional data distribution is to visualize similarities within a dataset sampled from distribution in two-dimensional space. Kohonen proposed the self-organizing map (SOM) algorithm [11, 12, 14] to do this. The SOM algorithm obtains a map from high-dimensional norm space, which we call feature space, to a two-dimensional space, called visible space. The algorithm uses a fixed number of fixed positions in visible space, called visible positions, and relates each visible position to a variable point in feature space, called a reference point. A pair of visible position and reference point is called a "neuron". In Kohonen's algorithm, the number of neurons and their visible positions determined by the user before learning starts remain fixed throughout the learning process. Reference points are initially set randomly or linearly. Mapping from feature space to visible space is self-organized by repeatedly feeding data sampled from distribution and updating reference points corresponding to neighbors in visible space of the neuron that has the reference point nearest to the fed datum. After the algorithm is processed, similar features are visualized at positions near to each other in visible space.

The SOM algorithm raises two major issues – (1) the need to set up two mapping parameters, the number and visible positions of neurons, before learning starts. The mapping result depends on these two fixed parameters and on learning parameters, such as the learning rate, total iteration, and neighboring radii. A user trying another set of parameters must thus start again from the beginning of learning. (2) Computation cost for a large-scale dataset is high. Advances in hardware and observation and simulation methodology make large-scale datasets more available more easily than before and increase the need to understand large-scale datasets better. If a large-scale dataset is not extremely redundant, the required number of neurons must be large for good quantization. In Kohonen's algorithm, the distance from each sample to all reference points must be calculated. The batch learning algorithm [15], which omits a learning parameter to be searched for, i.e., the learning rate, reduces the computation cost of updating reference points, but still requires that much distance be calculated.

Given these issues, it is natural to start learning with a small number of neurons and increase their number during learning. Fritzke [5–7] proposed a Growing Cell Structure (GCS) that starts learning with a small number of neurons. The output graph grows when nodes are added at new sampling points and edges are added between best-fit and second-best-fit nodes. This algorithm targets the obtaining node connections, however, rather than visualizing the data distribution in two-dimensional space. The graph obtained is not guaranteed to be planar, either, so another technique is required for visualizing the graph in two-dimensional space.

To obtain a planar graph in visible space of a SOM, Blackmore and Miikkulainen [2, 3] have proposed Incremental Grid Growing (IGG) and Alahakoon and Halgamuge [1] a similar method called the Dynamic SOM. The IGG grows nodes and connections on grid points laid on a plane. The Dynamic SOM introduces a parameter to control IGG growth. Together, they guarantee that a planar graph will be obtained but the shape of the grids obtained cannot be predefined and these methods cannot be applied to visible space of a topology other than planar, such as a torus or a sphere [17, 21], with which the so-called edge problem is solved. Rodrigues and Almeida [19, 20] proposed repeatedly halving grid intervals to increase the number of neurons, but this has not been applied to visible space topology other than a plane, perhaps due to the complexity of setting grids for toroidal and spherical visible space.

The new SOM method we propose generates neurons to significantly reduce computation cost. It also moves visible positions, which

- 1) enables application to visible space other than grids on a plane,
- 2) tessellates visible space evenly for easy visualization, and
- 3) levels the number of neighboring neurons to preserve topology well.

This paper is organized as follows: Section 2 presents new notation for the SOM algorithm to emphasize the extension of mapping parameters. Section 3 examines the effect of our proposal in reducing computation cost and discusses the effect of moving visible positions on topological preservation. Section 4 gives conclusions and presents perspectives.

## 2. Notation and Algorithm

In new notation of the SOM emphasizing our extension of mapping parameters, feature space  $X$  must have distance measure  $d: X^2 \rightarrow \mathbb{R}_+$ , which yields a non negative value for any two elements of  $X$ . For any non empty subset of  $X$ , the averaging of the subset exists, i.e.,  $\forall X' \subset X, \exists \bar{X}' \in X$ . Averaging is required for Kohonen's batch learning algorithm and our new algorithm. Visible space  $Y$  is usually two-dimensional space, such as a plane,

torus, or sphere surface. In a wider sense,  $d$  and  $Y$  are regarded as mapping parameters, but, we assume that  $d$  and  $Y$  are given.

Section 2.1 gives parameters  $\theta$  of mapping  $f_\theta: X \rightarrow Y$  in Kohonen's and our SOM algorithms, Section 2.2 Kohonen's batch learning algorithm, and Section 2.3 our algorithm.

### 2.1. Mapping Parameters

The SOM uses the following parameters

$$\theta = (N, m, \psi) \dots \dots \dots (1)$$

to represent mapping

$$f_\theta: X \rightarrow Y. \dots \dots \dots (2)$$

Here,  $N$  is a set of neurons,  $m: N \rightarrow X$  is the correspondence of neurons to their reference points, and  $\psi: N \rightarrow Y$  is the correspondence of neurons to their visible positions. A reference point is provided for a neuron  $i \in N$  by

$$m: i \mapsto m_i \in X \dots \dots \dots (3)$$

and its visible position by

$$\psi: i \mapsto \psi_i \in Y. \dots \dots \dots (4)$$

Using these mapping parameters, we first subdivide  $X$  into  $|N|$  Voronoi regions, such as

$$X_i = \left\{ x \in X \mid i = \arg \min_{j \in N} d(x, m_j) \right\} \dots \dots \dots (5)$$

Then, we map any feature point in  $i$ -th Voronoi region  $X_i \subset X$  to  $\psi_i$ , in mapping described as

$$f_\theta: (x \in X_i) \mapsto \psi_i. \dots \dots \dots (6)$$

In Kohonen's algorithm,  $m$  is the only variable parameter, and the user must repeatedly empirically set  $N$  and  $\psi$  as well as learning parameters. Our algorithm makes  $N$  and  $\psi$  variable, so, after mapping with  $|N| = n_1$  is obtained by learning, the user may increase  $N$  further to obtain mapping with  $n_2 (> n_1)$  neurons, i.e., with finer quantization. During learning,  $\psi$  is updated to tessellate  $Y$  evenly and to level the number of neighboring neurons.

### 2.2. Kohonen's Algorithm

In our notation we redefine Kohonen's batch learning SOM algorithm [15], which uses a set of samples  $S \subset X$ . Kohonen's algorithm requires that the number of neurons  $|N|$  and visible positions  $\psi$  be defined before learning starts. Total iteration  $T \in \mathbb{Z}_+$  and neighboring radii  $\rho: ([1, T]_{\mathbb{Z}}) \rightarrow \mathbb{R}_+$  must also be decided beforehand. The neighboring radius controls neighboring neurons for each neuron in each iteration.

The batch learning algorithm is as follows:

1. Fix  $N$  and  $\psi$ . Initialize  $m$ .
2. Subdivide  $S$ .
3. Update  $m$ .