Implementing   
the Kohonen Self-Organizing Map algorithm on Graphics Processing Unit

## Nicolas Said University of Technology of Belfort-Montbeliard 90010 Belfort cedex - +33 (0) 3 84 58 30 00

**Abstract:** *Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus adipiscing nulla ac enim. Aliquam sodales condimentum sapien. Donec pellentesque, mi quis dapibus egestas, dolor tortor tempor eros, vitae egestas lectus nunc vel risus. Suspendisse lacinia nibh at purus. Pellentesque porttitor libero eu felis. Maecenas iaculis erat. Nam euismod rutrum tortor. Nunc non dolor. Pellentesque laoreet posuere velit. Vivamus cursus tellus vitae enim imperdiet tristique.*

## Introduction

## Kohonen Self-Organizing Maps

### Brief presentation

Kohonen Self-Organizing Maps belong to a class of neural networks based on unsupervised learning. Also called SOM (for Self Organizing Maps), they were introduced by Teuvo Kohonen in 1975. Kohonen presents them as a “new, effective software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions”. Like most of the type of neural networks, the SOMs are directly inspired by the behavior of living beings. In that precise case, the algorithm is inspired by the cerebral cortex. It has been observed that the cortical structures are either linear or planar, however, the sensorial data that they must process are highly dimensional. The cortex seems to apply a sort of projection inside of its neural structures, from the high-dimension data space to the linear or planar space of cortical neural structures space. One of interesting property of this internal operation is that the so-called projection seems to preserve the topology of the original space. This property is very interesting and can be used to build powerful visualization or classification tools.

The SOM tries to imitate this behavior and use the properties of such non-linear projection.  
Well, the projection process appears to actually be vector quantization, in which the codebook is built automatically through the learning process.

### Principle

The SOM is composed of two neural network layers:  
-The output layer, to which we give a topology (most of the time, a rectangular mesh). Every vector of the output layer is associated to a referring vector, of the same dimension as the input data.   
-The input layer, composed of as much neuron as we have input data to classify, also associated to referring vectors (which are in fact the actual input data for classification)

Every input neuron is connected to each neuron of the output layer.

We then use a competitive learning algorithm which will modify the value of the referring vectors of the output layer. Those vectors will then constitute the codebook of a vector quantization (if we use a VQ analogy).

### Mathematical Formalization

Given A the grid of output neurons. Such a neuron map associate to each input vector , a neuron identified by its position in the grid , such as its referring vector is the closest to .

This association is given by this function:

Given this function, we can define the applications of the map :

*Vector quantization:*  
We approximate every point in the input space to the closest prototype in the map with :

*Classification:*  
Using the function , we can assign to each neuron in the map a label corresponding to a class, all the points of the input space which project to the same neuron in the map, belong to the same class.

To modify the output neurons referring vector and to make them converging toward the input space vectors, we use an iterative learning algorithm.

Learning Algorithm:  
**Initialization** of the output layer referring vectors to random values.

**(Loop) Until** the convergence conditions are not met:

**(Loop) For each** input neuron, chosen randomly,   
and given v its referring vector:

We search the output neuron having the referring vector closest to , called :

The winning neuron and its neighbors (defined by a neighboring function) move their referring vectors toward the input vector according to :

With

where is the learning ratio and is the neighboring function.

La fonction de voisinage décrit comment les neurones dans la proximité du vainqueur *s* sont entraînés dans le mouvement de correction. On utilise en général :

où σ s'appelle *coefficient de voisinage*. Son rôle est de déterminer un rayon de voisinage autour du neurone vainqueur.

Sigma est fonction de t, et est défini par :

avec N le nombre d'itérations prévues.

**(End of loop)**

**(End of loop)**

## General Purpose Computing on graphic processor

With the coming of programmable graphics hardware and their increase in processing power, memory and parallel processing capability, the use of graphics hardware for general purpose computation became a reality and practical solution for solving certain problem using relatively cheap hardware as supercomputer.

## Algorithm modification

## Implementation

## Complexity

First, we will consider that each GPU operation runs in constant time.

One iteration of the learning algorithm is composed of the two following steps :

Finding BMU:

Constant time (refer to experimental data)

This operation runs in O(

Update the output layer for each input value:

## Results