Two-step Modified SOM for Parallel Calculation*

Petr Gajdoš and Pavel Moravec

Department of Computer Science, FEECS, VŠB - Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic {petr.gajdos, pavel.moravec}@vsb.cz

Abstract. This paper presents a simple modification of classic Kohonen network (SOM), which allows parallel processing of input data vectors or partitioning the problem in case of insufficient memory for all vectors from the training set. The algorithm pre-selects potential centroids of data clusters and uses them as weight vectors in the final SOM network. We have demonstrated the usage of this algorithm on images as well as on two well-known datasets representing hand-written digits.

Keywords: SOM, Kohonen Network, parallel calculation, handwritten digits

1 Introduction

With the massive boom of GPU-based calculations, massive parallelism, memory considerations, simplicity of algorithms and CPU-GPU interaction have yet again to play an important role. In this paper, we present a simple modification of classic Kohonen's self-organizing maps (*SOM*), which allows us to dynamically scale the computation to fully utilize the GPU-based approach.

There were some attempts to introduce parallelism in Kohonen networks [4,6,5,7,8], however we needed an approach which is simple and easy to implement. Moreover, it should work both with and without the bulk-loading algorithm [2].

In this paper, we present such approach, which divides the training set into several subsets and calculates the weights in multi-step approach. Calculated weights with nonzero number of hits serve as input vectors of SOM network in the following step. Presently, we use a two-step approach, however more steps could be used if necessary.

The paper is organized as follows: in second chapter we mention classic SOM networks and describe the basic variant we have used. In third chapter we describe our approach and provide the calculation algorithm. The fourth chapter introduces experimental data we have used and presents the results of comparison of results provided by our method with classic SOM calculation.

2 Kohonen self-organizing neural network

In following paragraphs, we will shortly describe the Kohonen self-organizing neural networks (self-organizing maps - *SOM*). The first self-organizing networks were pro-

^{*} This work was partially supported by SGS, VŠB-Technical University of Ostrava, No. SP/2010196 grant – Machine Intelligence



Fig. 1. Kohonen network structure

posed in the beginning of 70's by Malsburg and his successor Willshaw. SOM was proposed by Teuvo Kohonen in the early 1980s and has been improved by his team since. The summary of this method can be found in [3].

The self-organizing map is one of the common approaches on how to represent and visualize data and how to map the original dimensionality and structure of the input space onto another – usually lower-dimensional – structure in the output space.

The basic idea of SOM is based on the human brain, which uses internal 2D or 3D representation of information. We can imagine the input data to be transformed to vectors, which are recorded in neural network. Most neurons in cortex are organized in 2D. Only the adjacent neurons are interconnected.

Besides of the input layer is in SOM only the output (competitive) layer. The number of inputs is equal to the dimension of input space. Every input is connected with each neuron in the grid, which is also an output (each neuron in grid is a component in output vector). With growing number of output neurons, the quality coverage of input space grows, but so does computation time.

SOM can be used as a classification or clustering tool that can find clusters of input data which are more closer to each other.

All experiments and examples in this paper respect following specification of the SOM (see also the Figure 1):

- The SOM is initialized as a network of fixed topology. The variables dimX and dimY are dimensions of such 2-dimensional topology.
- V^m represents an m-dimensional input vector.
- W^m represents an m-dimensional weight vector.
- The number of neurons is defined as N = dimX * dimY and every neuron $n \in < 0, N-1 >$ has its weight vector W_n^m
- The neighborhood radius r is initialized to the value min(dimX, dimY)/2 and will be systematically reduced to a unit distance.
- All weights vectors are updated after particular input vector is processed.

- The number of epochs e is know at the beginning.

The Kohonen algorithm is defined as follows:

1. Network initialization

All weights are preset to a random or pre-calculated value. The learning factor η , $0 < \eta < 1$, which determines the speed of weight adaptation is set to a value slightly less than 1 and monotonically decreases to zero during learning process. So the weight adaptation is fastest in the beginning, being quite slow in the end.

2. Learning of input vector Introduce k training input vectors V_1, V_2, \ldots, V_k , which are introduced in random order.

3. Distance calculation

An neighborhood is defined around each neuron whose weights are going to change, if the neuron is selected in competition. Size, shape and the degree of influence of the neighborhood are parameters of the network and the last two decrease during the learning algorithm.

4. Choice of closest neuron

We select the closest neuron for introduced input.

5. Weight adjustment

The weights of closest neuron and its neighborhood will be adapted as follows:

$$W_{ij}(t+1) = W_{ij}(t) + \eta(t)h(v,t)(V_i - W_{ij}(t)),$$

where i = 1, 2, ..., dimX a j = 1, 2, ..., dimY and the radius r of neuron's local neighborhood is determined by adaptation function h(v).

6. Go back to point 2 until the number of epochs e is reached.

To obtain the best organization of neurons to clusters, a big neighborhood and a big influence of introduced input are chosen in the beginning. Then the primary clusters arise and the neighborhood and learning factor are reduced. Also the $\eta \rightarrow 0$, so the changes become less significant with each iteration.

3 Proposed method

The main steps of SOM computation have been already described above. Following text is focused on description of proposed method, that in the end leads to results similar to the classic SOM (See also Figure 2 for illustration of our approach). We named the method Global-Merged SOM, which suggests, that the computation is divided into parts and then merged to obtain the expected result. Following steps describe the whole process of GM-SOM:

1. Input set split

The set of input vectors is divided into a given number of parts. The precision of proposed method increases with the number of parts, however, it has own disadvantages related to larger set of vectors in the final phase of process. Thus the number of parts will be usually determined from the number of input vectors. Generally,



Fig. 2. GM-SOM: An Illustrative schema of the proposed method. All input vectors are divided into ten parts in this case.

 $k \gg N * p$, where k is the number of input vector, N is the number of neurons and p is the number of parts. The mapping of input vectors into individual parts does not affect final result. This will be later demonstrated by the experiments, where all the input vectors were either split sequentially (images) or randomly (sets of handwritten digits).

2. In parts computation

Classic SOM method is applied on every part. For simplicity sake, an acronym *PSOM* will be used from now on to indicate SOM, which is computed in a given part. All PSOMs start with the same setting (the first distribution of weights vectors, number of neurons, etc.) Such division speeds up parallel computation of PSOMs on GPU. Moreover, the number of epochs can be lower in comparison with the number of epochs in case of processing of input set by one SOM. This is represented by a factor f, which is going to be equal to $\frac{1}{3}$ in our experiments.

3. Merging of parts

Weight vectors, that where computed in every part and correspond with neurons with at least one hit, represent input vectors in the final phase of GM-SOM. The unused neurons and their weight vectors have red color in Figure 2. Again, a new SOM with the same setting is computed and output weights vectors make the final result of proposed method.

The main difference between the proposed algorithm and well known batch SOM algorithms is, that individual parts are fully independent on each other and they update different PSOMs. Moreover, different SOM algorithms can be applied on PSOM of a given part, which makes proposed algorithm more variable. Next advantage can be seen in different settings of PSOMs. Thus more dense neuron network can be used in case of larger input set. The last advantage consists in a possibility of incremental updating

of GM-SOM. Any additional set of input vectors will be processed by a new PSOM in a separate part and the final SOM will be re-learnt.

4 **Experiments**



Fig. 3. Black-and-white symbols - (a) Original, (b) Per-partes, (c) Overlaid results

Our approach has been tested both on a generic set of black-and-white images as well as on two well-known datasets used for machine learning which were obtained from UCI repository [1].

In this section, we present examples of such results for a 10×10 hexagonal SOM with 300 iterations for both the original SOM network and the final calculation. Each partial SOM in our approach used 100 iterations ($f = \frac{1}{3}$). Moreover, where reasonable, a 20×20 hexagonal SOM will be used.

In the first set of experiments, we have generated a collection of black-and-white images, based on simple geometric shapes and symbols. Coordinates of black pixels were considered the input vectors for this experiment. Given a sufficient number of iterations, the weight vectors of Kohonen self-organizing neural network are known to spread through the image and cover the black areas.

We have compared the original method with our approach on a wide selection of shapes, ranging from simple convex shapes to complicated symbols consisting of several parts. Each symbol was tested several times to reduce the impact of random initiation of the network weights. Our approach typically provided results which were marginally better than the classic SOM. Some of the results are shown in Figure 3.



Fig. 4. OCR of handwritten digits: (a) SOM hits in case of classic approach, (b) SOM hits of proposed method. Black cells represent neurons without any hits.

For the second experiment, we have used *Optical Recognition of Handwritten Digits Data Set* from UCI repository, containing 43 sets of hand-written digits from different people. 30 sets of digits contributed to the training set and different 13 to the test set. The digits were scanned as 32×32 bitmaps, which were then divided into 4×4 blocks. The number of on pixels of each block has been recorded (to reduce the dimension and reduce the impact of small distortions) resulting in 64 features with values ranging from 0 to 16. The results for all existing classes in original and our approach are shown in Figure 6. Figure 4 shows the combined result based on SOM hits for the test set in both 10×10 and 20×20 networks. The results are again comparable for both methods.



Fig. 5. Pen digits: (a) SOM hits in case of classic approach, (b) SOM hits of proposed method. Black cells represent neurons without any hits.

The third experiment was conducted on *Pen-Based Recognition of Handwritten Digits Data Set.* The data set consists of 250 samples from 44 writers with 30 writers were used for training and the remaining digits written by the other 14 writers were used for testing. The recorded points were re-sampled to 8 x, y coordinates. The combined result based on SOM hits for the test set in both 10×10 and 20×20 networks for original and our approach is shown in Figure 5.

5 Conclusion

The need of parallel computation of SOM drove us to a new method, that has been presented in this paper. Although it has some common features with well known SOM batch or hierarchical algorithms, it is not one of them, as it has its unique properties.



Fig. 6. OCR of handwritten digits: Particular classes of neurons, that indicate digits 0–9 are illustrated separately.

Firstly, the proposed algorithm can utilize the power of batch processing in all inner parts (PSOMs). Moreover, all PSOMs can have different number of neurons in their networks, which could be found in hierarchical algorithms. Lastly, our method excludes neurons, which do not cover any input vectors in the intermediate phase of GM-SOM.

All experiments suggest, that the results are very close to results provided by classic SOM algorithm. We would like to test the proposed algorithm on huge data collections in the near future.

References

- 1. A. Asuncion and D. Newman. UCI machine learning repository, 2007.
- J. Fort, P. Letremy, and M. Cottrel. Advantages and drawbacks of the batch kohonen algorithm. In 10th-European-Symposium on Artificial Neural Networks, Esann'2002 Proceedings., pages 223–230, 2002.
- 3. T. Kohonen. Self-Organizing Maps. Springer Verlag, Berlin, second (extended) edition, 1997.
- R. Mann and S. Haykin. A parallel implementation of Kohonen's feature maps on the warp systolic computer. In *Proc. IJCNN-90-WASH-DC, Int. Joint Conf. on Neural Networks*, volume II, pages 84–87, Hillsdale, NJ, 1990. Lawrence Erlbaum.
- 5. T. Nordström. Designing parallel computers for self organizing maps. In Forth Swedish Workshop on Computer System Architecture, 1992.
- S. Openshaw and I. Turton. A parallel kohonen algorithm for the classification of large spatial datasets. *Computers & Geosciences*, 22(9):1019–1026, November 1996.
- I. Valova, D. Szer, N. Gueorguieva, and A. Buer. A parallel growing architecture for selforganizing maps with unsupervised learning. *Neurocomputing*, 68:177 – 195, 2005.
- L. Wei-gang. A Study of Parallel Self-Organizing Map. In Proceedings of the International Joint Conference on Neural Networks, 1999.