A Modular Approach to Implementation of the Self-Organising Map

N.Lightowler, C.T.Spracklen, A.R.Allen

Dept. of Engineering, University of Aberdeen, Scotland. email: neil@erg.abdn.ac.uk

Abstract

We present an overview of our design for a fully digital hardware implementation of the Self Organising Map (SOM) [1]. Our approach has resulted in a modular system (Modular Maps) which utilises fine grain parallelism with each neuron being a separate entity implemented as a small RISC processor. The essence of the SOM has been maintained by this design although minor modifications have been made to the original algorithm to facilitate implementation. Modules can be used as either stand alone systems or combined to enable large networks to be created and large input vectors to be catered for. A simulator system was developed to facilitate investigation into the high level behaviour of Modular Map systems, and as Modular Maps are computationally intensive and parallel in nature, it was implemented on a parallel computer system. A series of simulations was carried out using encoded images of human faces where it was found that the classification accuracy of a Modular Map system offered an improvement over that of the traditional SOM.

1 Introduction

When implementing Artificial Neural Networks (ANN) in hardware, difficulties are encountered as network size increases, not simply because of silicon area or pin out considerations, but because inter-neuron communication increases considerably with network size [2]. Our approach to the creation of scaleable ANNs has resulted in a modular system (Modular Maps) which provides us with a fully digital, and parallel, implementation of the Self Organising Map (SOM) [1], which utilises fine grain parallelism, i.e. each neuron is represented by an individual processing element with its weight vectors held in local memory to minimise communication overheads. Kohonen's original algorithm has been maintained, except that parameters have been quantised and the Euclidean distance metric has been replaced by the Manhattan distance. Each module contains a number of neurons which can be configured in a variety of ways. However, the Modular Map design is such that many modules can be used together to create a wide variety of configurations and network sizes while maintaining a highly parallel implementation of the self-organising map and avoiding the usual communications bottleneck between processors by minimising the amount of inter-module communication required.

2 The Modular Map

In our intended implementation of the SOM (following Kohonen's notation [3]) the multidimensional Euclidean input space $\Re^n$, where $\Re$ covers the range $(0, 255)$ and $(0 < n \leq 16)$, is mapped to a two dimensional output space $\Re^2$ (where $\Re$ is 8, i.e. each module contains 64 neurons) by way of a non-linear projection of the probability density function. Each neuron in the network has a reference vector $m_i = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{in}] \in \Re^n$ where μ_{ij} are scalar values, i is the neuron index and j the weight index. An input vector $x = [\xi_1, \xi_2, \ldots, \xi_n] \in \Re^n$, where ξ_i are 8 bit values, is presented to all neurons in the network where the closest matching reference vector (codebook vector) C is determined, i.e.

$$\sum_{j=0}^{n} |\xi_j - \mu_{cj}| = min\{\sum_{j=0}^{n} |\xi_j - \mu_{ij}|\}_{i=1}^{k}$$

where k = network size.

The neuron with minimum distance between its codebook vector and the current input, i.e. greatest similarity, becomes the active neuron. A variety of distance metrics can be used as a measure of similarity, the Euclidean distance being the most popular. However, the distance metric being used here is the Manhattan distance, known to many as the L_1 metric of the family of Minkowski metrics, i.e. the distance between two points a and b is $L_p = (|a-b|^p + |a-b|^p)^{1/p}$. Clearly Euclidean distance would be the L_2 metric under Minkowski's scheme. The Manhattan distance metric has been found to be both simple to implement and a reasonable alternative, in Euclidean space, to the Euclidean distance metric which is rather expensive to implement in terms of hardware due to the need to calculate the squares of distances involved.

After the active neuron has been identified reference vectors are updated to bring them closer to the current input vector. The amount by which codebook vectors are changed is determined by their distance from the input, and the current gain factor $\alpha(t)$, the gain factor being restricted to negative powers of two. If neurons are within the neighbourhood of the active neuron then their reference vectors are updated, otherwise no changes are made.

$$m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_i(t)] \text{ if i } \in N_c(t)$$

and

$$m_i(t+1) = m_i(t) \text{ if i } \notin N_c(t)$$

where $N_c(t)$ is the current neighbourhood and t = 0, 1, 2 ...

It was found that a square, step function neighbourhood, one of several approaches suggested by Kohonen, could be defined by the Manhattan distance metric. This approach to defining the neighbourhood has the effect of rotating the square through 45 degrees but has been found to work well and offers a solution that can be used by individual neurons to determine if they are in the current neighbourhood when given the index of the active neuron. When all these parameters are combined to form the Modular Map it has been found to have the same characteristics as the SOM and gives comparable results when evaluated [4].

A single Modular Map contains 64 neurons, each of which is effectively an individual processor. In addition, each module contains a controller which controls the input/output for the module and coordinates all processing elements making up the neural network. Each module could be viewed as a Single Instruction stream Multiple Data stream (SIMD) array of processors with each individual processor being a Reduced Instruction Set Computer (RISC) with very limited capabilities. The processors only perform three basic operations which are add, subtract and bit shift. From the addition and subtraction operations a neuron can calculate the Manhattan distance between input and weight vectors, determine if it is within the current neighbourhood, and update its weight vector. However, we require the bit shift operation to determine the extent of the update to be applied to a weight vector by performing a multiplication operation equivalent to the current gain factor value. It is the simplicity of implementing a bit shift operation, rather than a traditional multiplier, that has prompted us to restrict the gain factor to negative powers of two.

During the learning process defined for the SOM [3] several interesting factors emerge. The codebook vectors tend to approximate various distributions of input vectors with some sort of regularity and the resulting order always reflects properties of the probability density function $P(x)$, i.e. according to the formalism proposed by Luttrell [5] the point density of the reference vectors becomes proportional to $[P(x)]^{1/3}$. While the reference vectors are tending to describe the density function of inputs, local interactions between neurons tend to preserve continuity on the surface of the map. A combination of these opposing forces causes the vector distribution to approximate a smooth hyper-surface in the pattern space with optimal orientation and form that best imitates the overall structure of the input vector density. This formation of topological maps of the input space provides an important premise on which the Modular Map has been developed. It is because the activations across a network have meaningful order, as though a feature coordinate system were defined over the network, that the X, Y coordinates of the active neuron can be used to provide a meaningful output from a Modular Map. As the Modular Map is intended to allow for expansion, by combining many modules together to create large networks, it is convenient to restrict the output of a module to two eight bit values in order to minimise inter-module communication, however, the option to output distance values has been maintained should it be required.

3 Combining Modular Maps

As many difficult tasks require large numbers of neurons the Modular Map has been designed to enable the creation of networks with up to 65,536 neurons on a single plane by allowing lateral expansion. Each module consists of 64 neurons and consequently this is the building block size for the lateral expansion of networks. Each individual neuron can be configured to be at any position on a 2-dimensional array measuring up to 256^2 but networks should ideally be expanded in a regular manner so as to create rectangular arrays. As each neuron determines for itself if it is in the current neighbourhood when given the address of the active neuron, there are no real difficulties introduced by having several modules connected together. All that is required is some additional simple logic that will determine the minimum distance among modules and feed back the index of the neuron with this value to all modules forming the network.

All modules calculate the local 'winner' and are able to output the distance between the reference vector of this 'winner' and the current input. The external logic then takes as input this distance value and the index of the winning neuron from each module. The global winner is then determined by comparing the distances of all local winners and the index that accompanied the minimum distance is output in the same manner as a single module would do. During the learning phase the index is also fed back to a global bus that feeds all modules, who then transmit it to all their constituent neurons, where they can determine if they are in the current neighbourhood in exactly the same way as they do for stand alone systems. Once a network has been created in this way it acts as though it were a stand alone Modular Map and can be used in conjunction with other modules to create a wide range of network configurations. However, it should be noted that, while inter-module communications have been kept to a minimum, the communication overhead still increases with network size, and the number of training steps required is also proportional to network size, which suggests that, when expanding maps laterally, they are best kept to a moderate size.

The Modular Map system has been designed to allow expansion by connecting maps together in different ways to cater for changes in network size, as mentioned above, and input vector size, which will be discussed here, as well as providing the flexibility to enable the creation of novel neural network configurations. This modular approach offers a mechanism that maintains an even workload among processing elements as systems are scaled up, thereby providing an effective parallelism of the Self Organising Map. To facilitate expansion in order to cater for large input vectors, modules are arranged in a hierarchical manner which also appears plausible in terms of biological systems where, for example, layers of neurons are arranged in a hierarchical fashion in the primary visual system with layers forming increasingly complex representations the further up the hierarchy they are situated [6]

As each Modular Map is limited to a maximum of 16 inputs, it is necessary to provide a mechanism which will enable these maps to accept larger input vectors so they may be applied to a wide range of problem domains. Larger input vectors are accommodated by connecting together a number of Modular Maps in a hierarchical manner and partitioning the input data across modules at the base of the hierarchy. Each module in the hierarchy is able to accept up to 16 inputs, and outputs the X,Y coordinates of the active neuron for any given input; consequently there is a fan-in of eight modules to one which means that a single layer in such a hierarchy will accept vectors containing up to 128 inputs. By increasing the number of layers in the hierarchy we increase the number of inputs which can be catered for, i.e. maximum number of inputs $= 2 * 8^n$ (where n = number of layers in the hierarchy). From this simple equation it is easy to see that very large inputs can be catered for with very few layers in the hierarchy.

By building hierarchical configurations of Modular Maps to cater for large input vectors, we are in effect parallelising the workload among many simple processing elements. This approach was preferred over the alternative of using more complex neurons which would be able to accept larger input vectors. There are many reasons for this, not least the problems associated with implementation which, in the main, dictate that hardware requirements for individual neurons will increase with increasing input vector sizes catered for. Furthermore, as the input vector size increases, so too does the workload on individual neurons, which leads to considerable increases in propagation delay through the network. By building the hierarchical configurations being discussed here, the workload on individual neurons is kept almost constant with an increasing workload being met by an increase in neurons used to do the work. Clearly, there is still an increase in propagation time, with every layer added to the hierarchy, but this was regarded as a small price to pay for the benefits gained. When modules are connected together in a hierarchical configuration they operate asynchronously, requiring only that communication between the sending and receiving modules be synchronised. There is no communication between modules on the same level in the hierarchy and modules on different levels communicate via a bus. Only one module may place its output on the bus at any one time and the receiving module uses a simple handshake mechanism to control access. As a module only outputs two eight bit values (X,Y coordinates of its active neuron) and communication between modules is point to point, obviating routing overheads, the inter-module communication overhead is kept to a minimum and does not form a limiting factor as hierarchies are scaled up [4].

A hierarchical approach to Modular Map configuration offers us a mechanism to fuse together many inputs in a way that preserves the essence of the features represented by those inputs with regard to the metric being used. Our own simulations on the Modular Map simulator (MAPSIM) have shown this to be the case and have given further insight into the capabilities of this type of hierarchical configuration. There have been investigations into other hierarchical configurations utilising the SOM by other authors [7, 8], but in general these approaches have concentrated on extracting additional functionality such as the ability to track trajectories [9] and have required a considerable communication overhead compared with this implementation. When modules are connected together in a hierarchical manner there is still the opportunity to partition input data in various ways. The most obvious approach is to simply split the original high dimensional input data into vectors of 16 inputs or less, i.e. given the original feature space $\Re^n$, n is partitioned into groups of 16 or less. When data are partitioned in this way, each module forms a map of its respective input domain, there is no overlap of maps, and a module has no interaction with other modules on its level in the hierarchy. However, it is also realistic to consider an approach where inputs to the system would span more than one module, thereby enabling some sort of data overlap between modules. An approach of this nature may assist modules in their classification by providing them with some sort of context for the inputs, or perhaps it is a mechanism which allows the feature space to be viewed from a range of perspectives with the similarity between views being determined by the extent of the data overlap. Our own experiments with face data have also found that an overlap of inputs, i.e. feeding some inputs to two or more separate modules, can lead to an improved mapping and classification.

The design of the Modular Map is of such flexibility that inputs could be partitioned across the network system in some interesting ways, e.g. inputs could be taken directly to any level in the hierarchy, similarly, outputs can also be taken from any module in the hierarchy, which may be useful for merging or extracting different information types. There is no compulsion to maintain symmetry within a hierarchy which may lead to some novel configurations, and consequently separate configurations could be used for specific functionality and combined with other modules and inputs to form systems with increasing complexity of functionality. It is also possible to introduce feedback into Modular Map systems which is expected to enable the creation of some interesting modular architectures and expand possible functionality. Research by other authors suggests that these features are found in nature and that a modular architecture is also likely to exist in biological neural systems [10, 11]; our own research suggests that a modular approach towards the creation of ANNs is also viable [4].

4 Simulations

A simulator system was required to enable an investigation into the high level behaviour of Modular Map systems. It was important that our simulator offered great flexibility and gave a true representation of the behaviour of our intended Modular Map implementation, which has been achieved by good design coupled with rigorous test procedures. As Modular Maps are computationally intensive and parallel in nature, consideration was given to using a parallel computing system as the platform for our simulator. Of the parallel systems available to us, the Parsytec PowerXplorer, which contains 12 Motorola PowerPC 601 processors and can be configured as a Multiple Instruction stream Multiple Data stream (MIMD) system, was the most suitable and consequently the Modular Map Simulator (MAPSIM) was designed and implemented on the PowerXplorer. MAPSIM is a versatile tool for investigating the behaviour of Modular Map systems, its flexibility allows the user to evaluate the effects of varying network parameters and Modular Map configurations. While MAPSIM is a complex system, due in part to its being implemented on the PowerXplorer and supporting extensive flexibility, most of this complexity is hidden from the user who can only access the system via a GUI. The GUI protects the user from this complexity while enabling them to create and run simulations utilising arbitrary topologies of Modular Map systems with the minimum of effort, by utilising a hierarchical template system. Rigorous test procedures and defensive programming were used throughout the development process of MAPSIM in order to ensure that the results gained from simulations would be a true reflection of Modular Map system behaviour.

A series of simulations were carried out to assess hierarchical Modular Map systems using a range of data, including encoded pictures of human faces. As human face recognition is a well known and difficult problem, it was decided that it would provide a good means of evaluating the Modular Map. A database containing a range of encoded faces was obtained. The faces had been photographed by a colour CCD camera under differing conditions including different lighting and orientation. In addition, the pictures were taken over a period of weeks with different intervals between shots. The background to the photographs was eliminated, but the hair which is not invariant over time was left in the picture. Using Principal Component Analysis each picture was transformed into an eigenvector (which we call eigenfaces because of the application)[12].

During our simulations a variety of Modular Map configurations, utilising different data partitioning and pre-processing, was used to help determine how best to configure Modular Map systems and how they would perform. From these experiments we have found that the classification accuracy of a Modular Map system was good, achieving 96% recognition, which was better than that of a traditional SOM which achieved 94% recognition. The mappings formed were also found to be similar to those on SOMs, which suggests that the essential features of the inputs have been maintained. The Modular Map requires more neurons than the SOM when large input vectors are used, due to the limited vector size catered for by our RISC neurons, but hierarchical configurations of Modular Map systems address this problem in a way that leads to much shorter training times. These reduced training times are achieved by using smaller network sizes and multiple modules, therefore spreading the workload among more neurons than the traditional SOM. Using the eigenface data it was found that partitioning of the input data should provide some overlap between modules at the base of the hierarchy. For these particular data sets it was found that an overlap of four inputs (i.e. the last four inputs to the first module were also input to the second module etc.) gave the best results. In addition, the clustering and classification accuracy of the module at the top of the hierarchy was better than that of any module at a lower level.

5 Conclusion

The Modular Map design is intended to be implemented in a fully digital manner utilising fine grain parallelism with each neuron being a separate entity implemented as a small RISC processor. The essence of the SOM has been maintained by this design allowing modules to be used as either stand alone systems or connected together with other modules to form more complex systems. To further facilitate expansion inter-module communication is kept to a minimum and, while a slight increase in overheads is found when expanding networks laterally, only a fixed overhead is incorporated when connecting modules in hierarchies, which avoids the usual communication bottleneck normally encountered as network size increases. Modules can be combined in a variety of ways to enable large networks to be created and large input vectors to be catered for, as well as providing potential for the creation of novel network configurations which could utilise feedback and connections from anywhere in the system. A simulator was built to run on a parallel computing platform and a series of simulations were carried out to investigate the potential for Modular Map systems. From these experiments it was found that Modular Map systems gave good classification accuracy, and offered an improvement over the traditional SOM. The mappings formed were also found to be similar to those on SOMs, which suggests that the essential features of the inputs have been maintained. The Modular Map requires a larger number of neurons than the SOM, when large input vectors are used, due to the limited vector size catered for by our RISC neurons, but hierarchical configurations of Modular Map systems address this problem in a way that leads to much shorter training times-as the workload is spread among many simple neurons. It was also found that classification and clustering improved moving up the hierarchy of a Modular Map system such that the results from any layer in the hierarchy were better than those from any lower layers.

References

[1] Tuevo Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics*, 44(2):135–140, 1982.

[2] Derek B. I. Feltham Wojciech Maly. Limitations to the size of single-chip electronic neural networks. In *1991 Int. Conf. on Wafer Scale Integration*, pages 61–67. IEEE, IEEE, 1991.

[3] Tuevo Kohonen. *Self-Organizing Maps.* Springer series in information sciences. Springer-Verlag, Germany, 1995.

[4] Neil Lightowler. PhD thesis, University of Aberdeen, 1997.

[5] S. P. Luttrell. A bayesian analysis of self-organising maps. *Neural Computation*, 6(5):767–94, 1994.

[6] S.W. Kuffler et al. *From Neuron to Brain.* Sinauer and Associates, 2 edition, 1984.

[7] Helge Ritter. Combining self-organizing maps. In *Int. Joint Conference on Neural Networks*, pages II–499 – 502. IEEE, 1989.

[8] E. Oja J. Lampinen. Clustering properties of hierarchical self-organising maps. *Journal of Mathematical Imaging and Vision*, 2:165–176, 1992.

[9] S. P. Luttrell. Hierarchical vector quantisation. *IEE Proceedings*, 136(6):405–413, December 1989.

[10] Martin Ebdon. Toward a general theory of cerebral neocortex. *Journal of Mind and Language*, 8:368–403, 1993.

[11] Murre J. M. J Happel B. L. M. The design and evolution of modular neural network architectures. *Neural Networks*, 7:985–1004, 1994.

[12] Ian Craw. Machine coding of human faces. Technical report, Dept. of Mathematical Sciences, University of Aberdeen, March 1996.