Neurocomputing 10 (1996) 375–384

A dynamic neural net to compute convex hull

Amitava Datta [a], Swapan K. Parui [b,*]

[a] *Computer and Statistical Service Centre, Indian Statistical Institute, 203 B.T. Road, Calcutta 700 035, India*

[b] *Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700 035, India*

Abstract

Computing the convex hull of a planar point set is a well-known problem and several conventional algorithms are available to solve it. We have formulated this problem into a self-organizing neural net model. Self-organization is a learning phenomenon that works in the human vision system. In this paper, an artificial self-organizing neural net model is suggested. The net is dynamic in the sense that it can automatically grow itself without supervision and hence no prior knowledge on the number of processors is required. It has been established that such a net can compute the convex hull of a planar point set. Experimental results have been provided. Such a formulation helps parallelism and the biological modeling of the problem.

Keywords: Dynamic neural net; Self-organization; Planar convex hull; Biological modeling; Parallel computing

1. Introduction

In the present paper we deal with a specific computational geometry problem. In computational geometry, the problem to compute the convex hull of a finite number of points in 2D is quite well-known. The computation of the convex hull of a finite set of points, particularly in the plane, has been studied extensively and has wide applications in pattern recognition, image processing, statistics and several other fields [1,2]. Several conventional algorithms are available for this problem. What we aim at here is to formulate the convex hull problem as an artificial neural net problem. We discuss how an initial net can iteratively update itself on the basis of the input signals (points) and grow accordingly to finally arrive at the convex hull.

Artificial neural network models have been of great interest for some years in various fields like optimization, pattern recognition, computer vision and image processing. Artificial neural network models or simply 'neural nets' are massively parallel interconnections of computational elements (processors) that work as a collective system. They are computer-based simulations of living neural systems, which work quite differently from conventional computing. Instead of performing operations sequentially, neural network models are capable of doing the same simultaneously using nets composed of many processors connected by links, and thus provide high computational rates for real-time processing. Neural nets provide a new form of parallel computing, inspired by biological models. With the help of a number of processors, serving as the neurons in biological systems, connected by links, the neural network model can artificially work in a similar way as the brain.

In the recent past, neural net technology has been showing a great deal of promise even in areas of conventional computing. Several neural network models have been proposed so far [3–7]. Neural network models are specified by their net topology, node (processor) characteristics and training rules. Starting from an initial set of weights (usually random) these rules indicate how to adapt or adjust the initial weights to improve performance. The adaptation is a major focus of neural network models. Thus the potentiality to achieve parallelism and to mimic biological neural systems works as the major motivation behind neural net research.

Our focus is on a particular model of neural nets, namely, the self-organizing neural net. The term 'self-organization' refers to the ability to learn from the input without having any prior supervising information and arrange the processors accordingly. It is argued that self-organization works as a basic principle of sensory paths of the human visual system. In this case, self-organization means a meaningful ordering of the neurons (processors) of the brain, where the 'ordering' does not mean moving of neurons physically. The basic self-organizing system is a 1- or 2-dimensional array of processors forming a net and characterized by short-range lateral feedback between neighboring units. A self-organizing neural network model has been suggested by Kohonen [4]. In Kohonen's feature mapping algorithm, a net of processors, having some initial weights, is considered and the input signals are fed to it so that the net can adjust itself. The process is repeated for all the inputs and several iterations are performed. The process stops when it converges. The learning (adjustments) is unsupervised. After convergence, the net organizes itself in an orderly fashion.

A dynamic version of Kohonen's model has very recently been proposed by Sabourin et al. [8]. It is called selective multi-resolution approach. Such a dynamic model can overcome the following shortcomings of Kohonen's model, namely, (a) it is necessary to determine a priori the number of processors; (b) it can produce data collisions (overloaded nodes or nodes with no entry at all). Another dynamic version of Kohonen's model has been suggested by Fritzke [9] to model the probability distribution in the plane.

The present neural net also is dynamic and behaves, as discussed in the next section, like a self-organizing net. It is dynamic since we do not presume the number of processors beforehand. We start with a small number of processors and dynamically go on growing the number. The weights of the initial processors are not random. They start with some ordering, and when they grow in number, the ordering is maintained throughout the process. The inputs are not fed sequentially as in Kohonen's model. Instead, each processor works simultaneously on all the inputs and the feedback is passed on to itself. The process is unsupervised. Some new processors can be created and some old processors can be killed during the iterative process depending on the feedback at each iteration.

2. The model

In the present model we consider a set of n 2-dimensional input vectors (representing the signals)

$$S = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$
$$= \{P_1, P_2, \dots, P_n\}$$

Denote a circular list of m processors by

$$\{\pi_1, \pi_2, \dots, \pi_m\}$$

where each processor stores a 2-dimensional weight vector. The weight vector here represents a position in the plane where a processor can be thought to be located. In our discussion, placing a processor at a location does not mean that it is moved physically to a location. Instead, it means assigning a new weight vector to the processor.

Let

α_j = nearest P_i from π_j and

β_j = second nearest P_i from π_j

$i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.

We start initially with $m \geq 2$ (say, $m = 4$). The processors are ordered in a sequence $(\pi_1, \pi_2, \pi_3, \pi_4)$ and they are assumed to be placed on the circumference of a circle enclosing all the input points. Suppose they are situated on the four extreme points of two orthogonal diameters. Assume processor $\pi_{m+1} = \pi_1$.

Definition 1. For a processor π_j, its previous neighbour is processor π_{j-1} and next neighbour is processor π_{j+1}.

Definition 2. Two processors π_j and π_{j+1} are said to be equivalent (Fig. 1) if $\{\alpha_j, \beta_j\} = \{\alpha_{j+1}, \beta_{j+1}\}$.

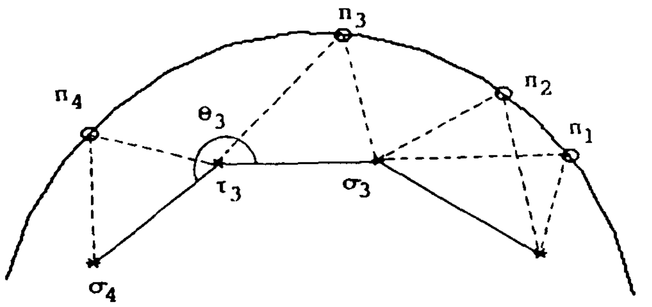

Fig. 1. Circles represent processors and *'s represent input points. Each processor is connected with its α_j and β_j by dashed lines. Processors π_1 and π_2 are equivalent; and π_3 and π_4 are shaking hands. θ_3 = angle $(\sigma_3, \tau_3, \sigma_4)$.

Definition 3. Two processors π_j and π_{j+1} are said to shake hands (Fig. 1) if they are not equivalent and

$$\{\alpha_j, \beta_j\} \cap \{\alpha_{j+1}, \beta_{j+1}\} \neq \phi \text{ (say, } \{\tau_j\})$$

Symbolically, $\pi_j \leftrightarrow \pi_{j+1}$. π_j and π_{j+1} are also called partners. π_j is previous partner of π_{j+1} and π_{j+1} is next partner of π_j.
Denote

$$\sigma_j = \{\alpha_j, \beta_j\} - \{\tau_j\}$$
$$\sigma_{j+1} = \{\alpha_{j+1}, \beta_{j+1}\} - \{\tau_j\}$$
$$\Theta_j = \text{angle}(\sigma_j, \tau_j, \sigma_{j+1}) \quad \text{(Fig. 1)}.$$

Definition 4. Two partner processors π_j and π_{j+1} are called convex partners if $\Theta_j > 180°$. Otherwise they are concave partners (Fig. 2).

Definition 5. For processors π_{j-1}, π_j and π_{j+1}; π_j is called an X-type processor if both the pairs (π_{j-1}, π_j) and (π_j, π_{j+1}) are concave partners. π_j is a Y-type processor if both the pairs (π_{j-1}, π_j) and (π_j, π_{j+1}) are convex partners. Otherwise, π_j is a Z-type processor (Fig. 2).

Definition 6. The convex hull of S is the smallest convex set containing S. The convex hull here is in fact a convex polygon. Each edge of the polygon is a hull edge and each of its vertices is a hull vertex.

For a Z-type processor π_j denote (Fig. 2)

$$\mu_j = \begin{cases} \tau_{j-1} & \text{if } \Theta_{j-1} > 180° \\ \tau_j & \text{if } \Theta_j > 180° \end{cases}$$

At any iteration if two consecutive processors π_j and π_{j+1} are equivalent, the processor created earlier is killed (deleted from the list). On the other hand, if π_j

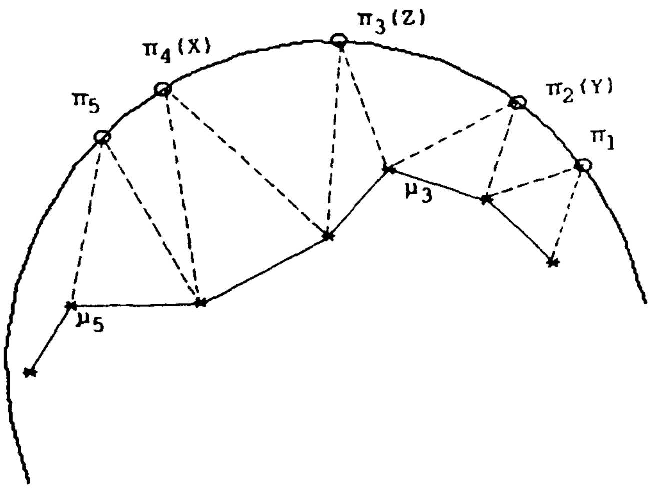

Fig. 2. Three types of processors X, Y and Z. Types are shown in parentheses. π_2 and π_3 are convex partners while π_3 and π_4 are concave partners.

and π_{j+1} are found not to shake hands then a new processor is created (inserted) in the middle of the circular arc joining π_j and π_{j+1}.

Theorem. *After a finite number of iterations the processors will be organized in such a way that for all $j = 1, 2, \ldots, m$ (m = current number of processors), $\pi_j \leftrightarrow \pi_{j+1}$. In such a situation the iterative process of insertion/deletion of processors stops. (Call it final handshaking.)*

Proof. Consider the Voronoi diagram of order two of S [1], which is a partition of the plane defined by the convex polygons $V(r, s)$, $1 \leq r, s \leq n$ where

$$V(r, s) = \{P\colon \max(d(P, P_r), d(P, P_s)) \leq d(P, P_t) \text{ for all } t\ (t \neq r, t \neq s)\}$$

where d stands for the Euclidean distance.
In general, $V(r, s)$ may be empty for some r, s.
It can be seen that

(a) If a processor π_j lies in the interior of $V(r, s)$, then $\{\alpha_j, \beta_j\} = \{P_r, P_s\}$. Two processors falling in the same $V(r, s)$ are equivalent.

(b) If two polygons $V(r, s)$ and $V(u, v)$ share a common edge, then the two sets $\{P_r, P_s\}$ and $\{P_u, P_v\}$ have exactly one point in common. For two processors π_j and π_k, if π_j lies in $V(r, s)$ and π_k lies in $V(u, v)$ then they are shaking hands.

Now the probability of a processor falling in the interior of some $V(r, s)$ is 1. Hence the theorem. □

Observation 1. Final handshaking will give rise to a polygon where each processor of the net corresponds to one edge and the τ_j's are the vertices of the polygon.

Observation 2. If all the processors of the net are Y-type then the generated polygon is the resulting convex hull where every processor corresponds to a hull edge and the τ_j's are the hull vertices. Otherwise, bridging is required to get the final convex hull from the polygon.

Before bridging, all the X-type processors are deleted from the circular list of processors.

Observation 3. After deleting the X-type processors, if two consecutive processors π_j and π_{j+1} are Z-type then there are three cases:
(a) π_j and π_{j+1} do not shake hands (due to deletion of X-type)
(b) π_j and π_{j+1} are concave partners
(c) π_j and π_{j+1} are convex partners.
Bridging is required for cases (a) and (b).

The formal algorithm can now be stated as follows:

```
Step 1 [Initialization].
Take $m = 4$. Place the ordered processors $(\pi_1, \pi_2, \pi_3, \pi_4)$ on the four extreme
points of two orthogonal diameters of a circle enclosing the points. Let $\pi_{m+1} = \pi_1$.
Step 2 [Handshaking].
(i)   For all new processors calculate $\alpha_j$, $\beta_j$.
(ii)  For $j = 1,\ldots,m$, if $\pi_j$ and $\pi_{j+1}$ are equivalent then kill the older one
      (processor created last is kept).
(iii) For $j = 1,\ldots,m$, if $\pi_j$ and $\pi_{j+1}$ do not shake hands and are not equivalent
      then create a new processor at the middle of the arc joining $\pi_j$ and $\pi_{j+1}$.
Repeat this step until for every processor, $\pi_j \leftrightarrow \pi_{j+1}$.
Step 3 [Bridging].
(i)   Label all the processors as X, Y and Z according to
      Definition 5.
(ii)  Kill all the X-type processors.
(iii) For $j = 1,\ldots,m$,
        if $\pi_j$ and $\pi_{j+1}$ are Z-type then
        begin
          if either of cases (a) or (b) in Observation 3
            is true then
          begin
            kill $\pi_j$ and $\pi_{j+1}$ and
            create a new processor $\pi_k$ (say) such that
            $\alpha_k = \mu_j$ and $\beta_k = \mu_{j+1}$
          end
        end.
Repeat this step until every processor is Y-type.
Step 4 [Termination]. Stop.
```

It can be seen that the above model works as a dynamic self-organizing neural net. Let us look at what basically happens in our algorithm. We start with an initial

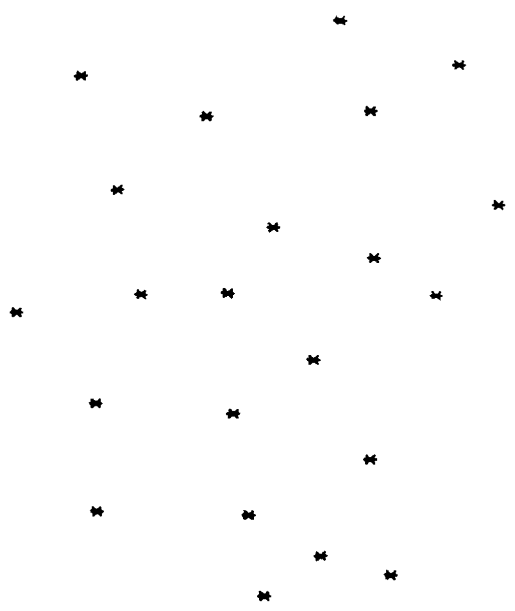
Fig. 3. The input points.

net having a given number ($m = 4$) of processors. The input signals are fed to the net and some feedbacks (α_j and β_j) are generated. Depending on the feedback the net grows in such a fashion that a certain ordering (of the processors) takes place after a number of iterations. Like other neural net models the net finally converges. Moreover, such an ordering happens without any supervision. In brief, from the inputs the net adapts itself in an unsupervised way and organizes itself according to the input. Similar things essentially happen in a self-organizing neural net.

3. Results and conclusions

Neural net based information processing or neurocomputing has been established to be a promising computational technique. One inherent advantage of neural nets is parallelism. Another advantage of neural net based models is the adaptation capability that enables biological modeling of a problem. This is why more and more researchers are interested in this field and thus its domain of application is increasing. As a result, there have been attempts to solve more and more problems, already having conventional solutions, using neurocomputing. This, in turn, is enriching this field. The proposed work is an attempt to demonstrate an application of neural net based models in a computational geometry problem, namely, convex hull computation for a planar point set [1,2]. A specific neural net model is suggested for this purpose. It is shown that a (dynamic)

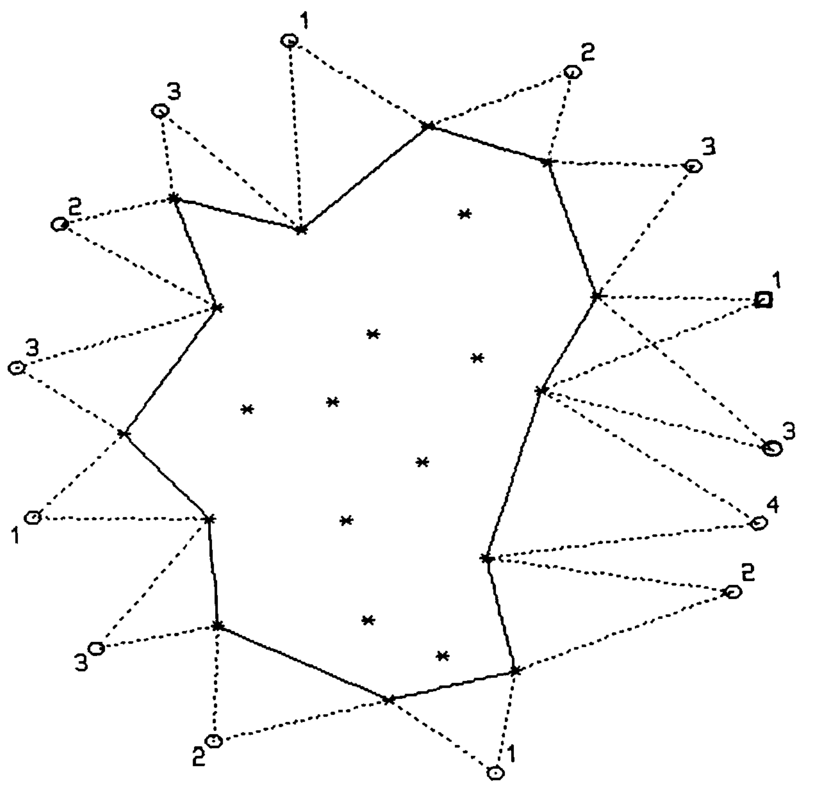

Fig. 4. The final handshaking is shown. A box represents a processor that was killed (deleted). Solid lines are the edges of the generated polygon.

self-organizing neural net is capable of learning from the input and can arrange itself from which the convex hull can be obtained.

Several conventional algorithms are available for convex hull computation [1,2]. The worst-case complexity of this problem, using a single processor, is O(n log n) where n is the input size. The divide-and-conquer method also has the same complexity. This complexity is known to be optimal. In our algorithm, in the handshaking step, at each iteration, all the processors calculate α_j and β_j in parallel. Since there are n input points, this step takes O(n) time. It can be seen that the handshaking step decides the complexity of the algorithm in the worst-case sense. Thus the worst-case complexity of the algorithm, with O(n) processors, is linear at each iteration.

The neural net model discussed above has been implemented and tested on several point sets. One such set consisting of 22 points is shown in Fig. 3. An intermediate result obtained after handshaking step is shown in Fig. 4. The numeral label (in Fig. 4) for each processor indicates the iteration number at which

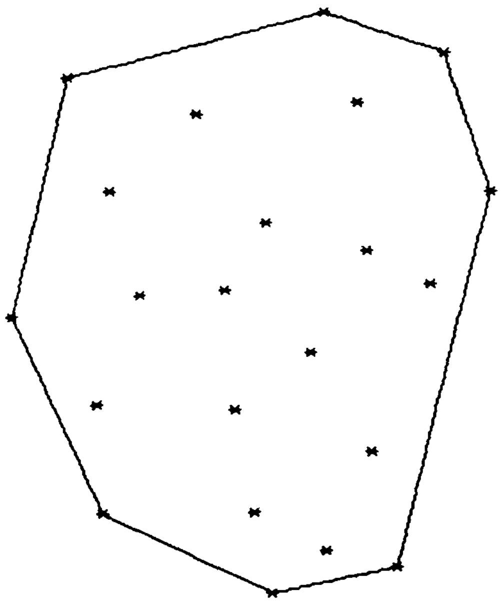

Fig. 5. The convex hull found after bridging.

the processor was created. It can be seen that totally four iterations were required for final handshaking. Fig. 5 shows the final result (i.e. the convex hull) obtained after the bridging step.

References

[1] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction* (Springer-Verlag, New York, 1985).

[2] G.T. Toussaint, ed., *Computational Geometry* (North-Holland, New York, 1985).

[3] D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing, Vol. 1* (MIT Press, Cambridge, 1986).

[4] T. Kohonen, *Self-Organization and Associative Memory* (Springer-Verlag, 1988).

[5] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks* (Addison-Wesley, 1989).

[6] G.A. Carpenter and S. Grossberg, eds., *Neural Networks for Vision and Image Processing* (MIT Press, 1992).

[7] R.P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Mag.* (Apr. 1987) 4–22.

[8] M. Sabourin and A. Mitiche, Modeling and classification of shape using a Kohonen associative memory with selective multiresolution, *Neural Networks* 6 (1993) 275–283.

[9] B. Fritzke, Let it grow - self-organizing feature maps with problem dependent cell structure, in: T. Kohonen et al., eds., *Artificial Neural Networks, Vol. 1* (North-Holland, 1991) 403–408.

Amitava Datta received his Master degree from the Indian Statistical Institute, Calcutta, India and subsequently did post-graduate course in Computer Science at the same institute. After working for a few years in reputed computer industries Mr. Datta joined the Computer Centre of the Indian Statistical Institute as a System Analyst in 1988. Since then he has been working in image processing and pattern recognition. From 1991 to 1992 he visited GSF, Munich as a Guest Scientist and worked on query-based decision support systems. He is currently working on neural-net based image processing and computer vision.

S. K. Parui received the Master of Statistics degree from the Indian Statistical Institute, Calcutta in 1975. After a brief career in commercial software development, he was back to the academics and received his Ph.D. degree in image processing from the Indian Statistical Institute in 1986. From 1985 to 1987 he worked as a Research Assistant on an automatic industrial inspection project in Leicester Polytechnic, England. Since 1987 he has been a Lecturer in the Indian Statistical Institute. From 1988 to 1989 he was a Visiting Scientist in GSF, Munich to work on a pattern recognition problem in biomedicine. His current research interests include shape analysis, remote sensing, statistical pattern recognition and neural networks.