On Merging the Fields of Neural Networks and Adaptive Data Structures to Yield New Pattern Recognition Methodologies

B. John Oommen

School of Computer Science, Carleton University, Ottawa, Canada*

Abstract. The aim of this talk is to explain a pioneering *exploratory* research endeavour that attempts to merge two completely different fields in Computer Science so as to yield very fascinating results. These are the well-established fields of Neural Networks (NNs) and Adaptive Data Structures (ADS) respectively. The field of NNs deals with the training and learning capabilities of a large number of neurons, each possessing minimal computational properties. On the other hand, the field of ADS concerns designing, implementing and analyzing data structures which adaptively change with time so as to optimize some access criteria. In this talk, we shall demonstrate how these fields can be merged, so that the neural elements are themselves linked together using a data structure. This structure can be a singly-linked or doubly-linked list, or even a Binary Search Tree (BST). While the results themselves are quite generic, in particular, we shall, as a *prima facie* case, present the results in which a Self-Organizing Map (SOM) with an underlying BST structure can be adaptively re-structured using conditional rotations. These rotations on the nodes of the tree are local and are performed in constant time, guaranteeing a decrease in the Weighted Path Length of the entire tree. As a result, the algorithm, referred to as the Tree-based Topology-Oriented SOM with Conditional Rotations (TTO-CONROT), converges in such a manner that the neurons are ultimately placed in the input space so as to represent its stochastic distribution. Besides, the neighborhood properties of the neurons suit the best BST that represents the data.

Summary of the Research Contributions

Consider a set $A = \{A_1, A_2, \ldots, A_N\}$ of records, where each record A_i is identified by a unique key, k_i. The records are accessed with respective probabilities $S = [s_1, s_2, \ldots, s_N]$, which are assumed unknown. In the field of Adaptive Data Structures (ADS), we try to maintain A in a data structure which is constantly changing so as to optimize the average or amortized access times.

* *Chancellor's Professor*; *Fellow : IEEE* and *Fellow : IAPR*. The Author also holds an *Adjunct Professorship* with the Dept. of ICT, University of Agder, Norway. The author is grateful for the partial support provided by NSERC, the Natural Sciences and Engineering Research Council of Canada. Although the research associated with this paper was done together with my students including Rob Cheetham, David Ng and Cesar Astudillo, the future research proposed is truly of an *exploratory* nature, and in one sense, could be "wishful thinking".

If the data is maintained in a list, adaptation is obtained by invoking a Self-Organizing List (SLL), which is a linear list that rearranges itself each time an element is accessed. The goal is that the elements are eventually reorganized in terms of the descending order of the access probabilities. Many memoryless update rules have been developed to achieve this reorganization, [5,8,13,15,16,17]. Foremost among these are the well-studied Move-To-Front (MTF), Transposition, the POS(k) and the Move-k-Ahead rules. Schemes involving the use of extra memory have also been developed [16,17]. The most obvious of these, uses counters to achieve the estimation of the access probabilities. Another is a stochastic Move-to-*Rear* rule due to Oommen and Hansen [15], which moves the accessed element to the rear with a probability which decreases each time the element is accessed. Stochastic MTF [15] and various stochastic and deterministic Move-to-*Rear* schemes [16,17] due to Oommen *et. al* have also been reported. All of these rules can also be used for Doubly-Linked Lists (DLLs), where accesses can be made from either end of the list.

A Binary Search Tree (BST) may also be used to store the records where the keys are members of an *ordered* set, A. Each record A_i is identified by a unique key, and the records are stored in such a way that a symmetric-order traversal of the tree (with respect to the identifying key) will yield the records in an ascending order. The problem of constructing an optimal BST given A and S requires $O(N^2)$ time and space [11]. Generally speaking, all the BST heuristics use the primitive **Rotation** operation [1] to restructure the tree. Memoryless BST schemes also employ the Move-To-Root [4] and Simple Exchange [4] rules which are analogous to the MTF and transposition rules for SLLs. Sleator and Tarjan [18] introduced a scheme, which moves the accessed record up to the root of the tree using the *splaying* operation – a multi-level generalization of rotation. Schemes requiring extra memory such as the Monotonic Tree scheme and Melhorn's D-Tree etc. have also been proposed [14]. In spite of the fact that SLLs and BSTs could have conflicting reorganization criteria, there is a close mapping between *certain* SLL heuristics and the corresponding BST heuristics as reported by Lai and Wood [13]. With regard to *Adaptive* BSTs, the most effective solution is due to Cheetham *et al.* which uses the concept of *Conditional* Rotations [6]. The latter paper proposed a solution where an accessed element is rotated towards the root if and only if the overall Weighted Path Length of the resulting BST decreases.

The field of NNs [7,9] deals with the training and learning capabilities of a large number of computing elements (i.e., the neurons), each possessing minimal computational properties. There are scores of families of NNs described in the literature, including the Backpropagation, the Hopfield network, the Neocognitron, the SOM etc. [12]. However, unlike the traditional concepts useful in developing families of NNs, we propose to "link" the neurons together using a data structure which can be a SLL, a DLL or even a BST. As far as we know, such an attempt to merge the fields of NNs and ADS is both novel and pioneering.

The advantage of using an ADS is that during the training phase, we can modify the configuration of the data structure by moving a neuron closer to its head (root), and thus explicitly recording the relevant role of the particular node with respect to its nearby neurons. This leads us to the concept of **Neural Promotion**, which is the process by which a neuron is relocated in a more privileged position[1] in the network with respect to the other neurons in the neural network. Thus, while "all neurons are born equal", their importance in the society of neurons is determined by what they represent. This is achieved, by an explicit advancement of its rank or position.

While the results themselves are quite generic and can potentially lead to many new avenues for further research, in particular, we shall, as a *prima facie* case, present the results [2,3] in which the NN is the Self-Organizing Map (SOM) [12]. Even though numerous researchers have focused on deriving variants of the original SOM strategy, few of the reported results possess the ability of modifying the underlying topology, leading to a dynamic modification of the structure of the network by adding and/or deleting nodes and their inter-connections. Moreover, only a small set of strategies use a tree as their underlying data structure. From our perspective, we believe that it is also possible to gain a better understanding of the unknown data distribution by performing *structural* tree-based modifications on the tree, by rotating the nodes within the BST that holds the whole structure of neurons. Thus, we attempt to use rotations, tree-based neighbors *and* the feature space as an effort to enhance the capabilities of the SOM by representing the underlying data distribution and its structure more accurately. Furthermore, as a long term ambition, this might be useful for the design of faster methods for locating the SOM's Best Matching Unit.

The *prima facie* strategy for which we have obtained encouraging results is the Tree-based Topology-Oriented SOM with Conditional Rotations (TTO-CONROT). TTO-CONROT has a set of neurons, which, like all SOM-based methods, represents the data space in a condensed manner. Secondly, it possesses a connection between the neurons, where the neighbors are based on a learned tree-based nearness measure. Similar to the reported families of SOMs, a subset of neurons closest to the BMU are moved towards the sample point using a vector quantization rule. But, unlike many of the reported SOM families, the identity of the neurons moved is based on the tree-based proximity (and not on the feature-space proximity). CONROT-BST achieves neural promotion by performing a *local* movement of the node, where only its direct parent and children are aware of the neuron promotion. Finally, the TTO-CONROT incorporates tree-based mutations, namely the above-mentioned conditional rotations.

Our proposed strategy is adaptive, with regard to the migration of the points *and* with regard to the identity of the neurons moved. Additionally, the distribution of the neurons in the feature space mimics the distribution of the sample points. Lastly, by virtue of the conditional rotations, it turns out that the entire tree of neurons is optimized with regard to the overall accesses, which is a unique phenomenon – when compared to the reported family of SOMs.

The potential to extend these results for other NN families and ADSs is open.

[1] As far as we know, we are not aware of any research which deals with the issue of Neural Promotion. Thus, we believe that this concept, itself, is pioneering.

References

1. Adel'son-Velski'i, G.M., Landis, E.M.: An algorithm for the organization of information. Sov. Math. Dokl. 3, 1259–1262 (1962)
2. Astudillo, C.A., Oommen, J.B.: A novel self organizing map which utilizes imposed tree-based topologies. In: Kurzynski, M., Wozniak, M. (eds.) Computer Recognition Systems 3. Computer Recognition, vol. 57, pp. 169–178. Springer, Heidelberg (2009)
3. Astudillo, C.A., Oommen, B.J.: On using adaptive binary search trees to enhance self organizing maps. In: Nicholson, A., Li, X. (eds.) AI 2009. LNCS, vol. 5866, pp. 199–209. Springer, Heidelberg (2009)
4. Allen, B., Munro, I.: Self-organizing binary search trees. Journal of the ACM 25, 526–535 (1978)
5. Arnow, D.M., Tenenbaum, A.M.: An investigation of the move-ahead-k rules. In: Proceedings of Congressus Numerantium, Proceedings of the Thirteenth Southeastern Conference on Combinatorics, Graph Theory and Computing, Florida, pp. 47–65 (1982)
6. Cheetham, R.P., Oommen, B.J., Ng, D.T.H.: Adaptive structuring of binary search trees using conditional rotations. IEEE Transactions on Knowledge and Data Engineering 5, 695–704 (1993)
7. Duda, R., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, Hoboken (2000)
8. Gonnet, G.H., Munro, J.I., Suwanda, H.: Exegesis of self-organizing linear search. SIAM Journal of Comput. 10, 613–637 (1981)
9. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice-Hall, Englewood Cliffs (2008)
10. Hester, H.J., Herberger, D.S.: Self-organizing linear search. In: ACM Computing Surveys, pp. 295–311 (1976)
11. Knuth, D.E.: The Art of Computer Programming, vol. 3. Addison-Wesley, Reading (1973)
12. Kohonen, T.: Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1995)
13. Lai, T.W., Wood, D.: A relationship between self organizing lists and binary search trees. In: Proceedings of the 1991 Int. Conf. Computing and Information, May 1991, pp. 111–116 (1991)
14. Mehlhorn, K.: Data Structures and Algorithms 1: Sorting and Searching. Springer, Berlin (1984)
15. Oommen, B.J., Hansen, E.R.: List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations. SIAM Journal of Computing 16, 705–716 (1987)
16. Oommen, B.J., Hansen, E.R., Munro, J.I.: Deterministic optimal and expedient move-to-rear list organizing strategies. Theoretical Computer Science 74, 183–197 (1990)
17. Oommen, B.J., Ng, D.T.H.: An optimal absorbing list organization strategy with constant memory requirements. Theoretical Computer Science 119, 355–361 (1993)
18. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. Journal of the ACM 32, 652–686 (1985)
19. Walker, W.A., Gotlieb, C.C.: A top-down algorithm for constructing nearly optimal lexicographical trees. In: Graph Theory and Computing (1972)