From: C. von der Malsburg, J.C. Vorbrüggen, W. von Seelen, and B. Sendhoff (Eds.): *Artificial Neural Networks – ICANN 96*, Lecture Notes in Computer Science, vol.1112, Springer Verlag, 1996, pp. 145–150

How fast can neuronal algorithms match patterns?

Rolf P. Würtz [*1], Wolfgang Konen[2], and Kay-Ole Behrmann

[1] Computing Science, University of Groningen, The Netherlands
[2] Zentrum für Neuroinformatik GmbH, Bochum, Germany

Abstract. We investigate the convergence speed of the Self Organizing Map (SOM) and Dynamic Link Matching (DLM) on a benchmark problem for the solution of which both algorithms are good candidates. We show that the SOM needs a large number of simple update steps and DLM a small number of complicated ones. A comparison of the actual number of floating point operations hints at an exponential vs. polynomial scaling behavior with increased pattern size. DLM turned out to be much less sensitive to parameter changes than the SOM.

1 Introduction

For visual perception in a biological or artificial system the *visual correspondence problem* is of central importance: "Given two images of the same physical object decide which point pairs belong to the same point on the object." A generic solution to this problem will at least greatly alleviate many of the difficulties encountered by computer vision research. Invariant object recognition, e.g., becomes easy if a correspondence map of sufficient density and reliability can be constructed between objects and stored prototypes [1]. The study [2] is another illustration of the power of such an algorithm. It describes a network based on Dynamic Link Matching (DLM) that learns to evaluate input patterns for the presence of one out of three mirror symmetries. That approach has been compared with a standard backpropagation network that needed some 10^4 examples [3]. The basic idea of DLM dates back to [4].

For a solution of the correspondence problem an ordered mapping from one plane to another has to be established with a combination of two constraints: Matching points must carry *similar features*, and *neighborhood relations* must be preserved. A good candidate to solve that problem is a self-organizing algorithm that develops from an unordered initial state to a clean one-to-one mapping. As self-organization is a notoriously slow process the *convergence speed* is an important detail. Short of sound analytical results we have used the problem from [2] as a benchmark to evaluate the relative performance of DLM [5] and the Self Organizing Map (SOM) algorithm [6, 7].

[*] Please address correspondence to rolf@cs.rug.nl. Funding from the HCM network "Parallel modeling of neural operators for pattern recognition" by the European Community is gratefully acknowledged.

2 Definition of a benchmark problem

A fair comparison of algorithms that were developed to solve different problems and whose full range of applicability is still subject of intensive research is not easy. The least one can do is to define problem and simulations very explicitly and leave it to the reader to judge if justice has been done to both algorithms, which are specified in sections 3.1 and 4.1, respectively.

If a square lattice is mapped onto a continuous input square (a typical problem for the SOM), the correct solution is not obvious. For a fair comparison, however, the quality must be assessed by objective means. The decision will be influenced considerably by boundary effects. Furthermore, there may be multiple solutions of identical intuitive quality (e.g., mirror reflections, rotations by 90°).

To avoid these problems we have chosen the above-mentioned mirror-problem as a benchmark. The setup consists of two square layers X and Y of $N \times N$ neurons that in addition carry features $f \in \{1, \ldots, F\}$. The feature distribution in X is chosen at random, but patterns with multiple symmetries are discarded. The distribution in Y is identical to the one in X except for either a mirror reflection or a rotation by 90°. Now the feature distributions induce a unique neighborhood-preserving mapping from X to Y. The benchmark task for the self-organizing algorithms is to find these mappings given only the feature distributions. *Similarity* of features of neurons $x \in X$ and $y \in Y$ is defined as all-or-nothing for this benchmark (for practical applications smooth similarity functions are usually more suitable):

$$T(x,y) = \delta\left(f_x, f_y\right) = \begin{cases} 1 & \text{if } f_y = f_x \\ 0 & \text{otherwise}. \end{cases} \tag{1}$$

A nearest-neighbor topology with wrap-around borders is imposed on both X and Y (this makes them 2D-tori rather than squares). The mirror axis may thus be any line and the center of rotation any point on the layers.

Due to the discrete lattices in both layers neighborhood preservation is clearly defined. Thus, the optimal solution is known beforehand, which gives a straightforward error measure that can be monitored through the whole process. Let $\mathbf{w}_y(t)$ be the position where neuron y points in layer X at time t, and $\mathbf{w}_y^{opt}$ the optimal mapping. Then the error will be

$$E(t) = \sum_y \left(\mathbf{w}_y^{opt} - \mathbf{w}_y(t)\right)^2 . \tag{2}$$

For the $N \times N$-size benchmark problem a particular solution is said to have *converged* if the average position error $E(t)/N^2$ is below the threshold $\varepsilon = 1/640$. The number of features F is a useful parameter to control the difficulty of the problem. Here we have used $F = 10$ equally distributed features.

3 Solution with the SOM

3.1 Method description

In order to apply the SOM to our benchmark we have identified the discrete neuron layer with our layer Y and the input space with X. Furthermore, we have included the feature similarity $T(x,y)$ into the learning rule of an otherwise unmodified SOM-algorithm [6, 7]:

$$\Delta \mathbf{w}_i = \lambda \, \exp\left(-|i-i_\mu|^2/2\sigma^2\right)(\mathbf{v}^\mu - \mathbf{w}_i)\, T(\mu, i_\mu)\,. \tag{3}$$

Here, $T(\mu, i_\mu) \in [0,1]$ is the similarity between the features of neurons $\mu \in X$ and its best matching counterpart $i_\mu \in Y$, respectively, i.e. the adaptation rate is weighted by the feature similarity. In our benchmark, T can only take the binary values 0 and 1, so that iteration steps for neurons with unequal features are without effect. Therefore, we have optimized the algorithm by skipping all such iteration steps and choosing i_μ directly among the neurons with the same features as μ. Only those "effective" iteration steps are counted in our results.

3.2 Parameter tuning

Many applications of the SOM require considerable care in adjusting the parameters. Generally, a decrease of the learning rate λ and the width σ is necessary to assure convergence. Unfortunately, there is no problem-independent rule on how to find these parameters. We have chosen a linear decreasing scheme. Different schemes (e.g. exponential ones) are in use, but are known [7] to yield the same general behavior of the algorithm. Extensive experiments have shown that the performance could not be improved by decreasing λ. We have thus kept it constant at $\lambda = 1$, but the width parameter σ and its decrease schedule (start/stop-value, decrease rate) had to be chosen carefully. Both were optimized individually for each problem size ($N = 4, \ldots, 20$) by scanning the reasonable parameter range (40 values) with 10 executions of SOM each. Only the 8 best results are shown in figure 1. The results are consistent with an exponential scaling of the number of update steps with the problem size (see figure 1).

4 Solution with DLM

4.1 Method description

In the DLM scheme layer X is fully connected with Y by a matrix $J(x,y)$ of dynamical links. Their development is governed by a Hebbian rule with competition and influence of feature similarity. In other words, links between pairs of neurons which have *similar features* and are *active at the same time* will be strengthened, others decay. Neighborhood preservation in DLM is achieved by ensuring that in each layer only *one connected subregion* of a given form and size, which we will call a *blob*, can be active at one time. This is a way to code

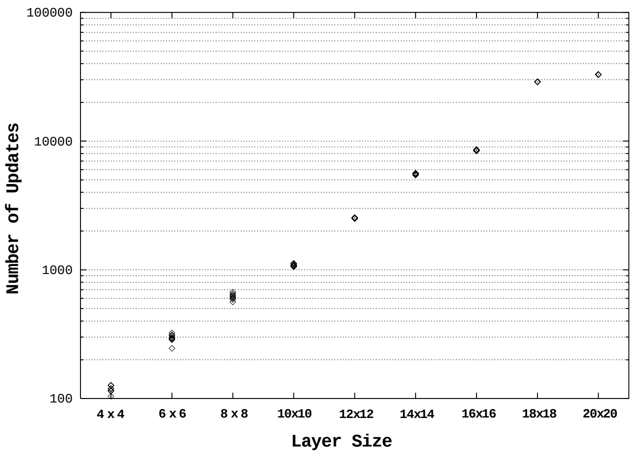

Fig. 1. Scaling behavior of the SOM algorithm on the mirror benchmark. Only runs with the optimal parameter set for each problem size are shown. The vertical spread of data points is due to different random seeds. A straight line fits the data well, hinting at exponential scaling with problem size.

neighborhood in the layer as common activity in the same time slot. A blob in layer X excites layer Y by means of the dynamic links $J(x, y)$. Layer Y supports only blobs of the same form and size as X. The links only influence the *position* of the blob, which can be calculated. Now only the neurons in one blob in X and one blob in Y are active and can strengthen their links in the following update step according to their feature similarities. An activity blob is a unimodal non-negative function b of the layer neurons. In our simulations we have chosen it to be 1 inside a square of size $B \times B$ and 0 outside. Then, the concrete algorithm runs as follows:

1. All links are initialized to $1/N^2$.
2. A position $x_0 \in X$ is chosen at random, a blob is placed there, and the resulting blob position y_0 in Y is calculated such as to minimize the *potential*

$$V(y_0) = -\sum_y \sum_x J(x,y)\, T(x,y)\, b(x-x_0)\, b(y-y_0)\,. \tag{4}$$

3. The activities in X and Y are now blobs positioned at x_0 and y_0, respectively, and the links $J(x, y)$ are updated by the learning rule:

$$\Delta(x,y) := \lambda\ (J(x,y) + J_0)\ T(x,y)\, b(x-x_0)\, b(y-y_0)\,. \tag{5}$$

4. The updated links are first normalized by $\sum_x J(x,y)$ and then by $\sum_y J(x,y)$.

Steps 2 through 4 are iterated until convergence. For the pointer vector $\mathbf{w}_y$ that enters the error function (2) the *center of mass* for all links $J(x, y)$ projecting onto neuron y has been used:

$$\mathbf{w}_y = \frac{\sum_x \mathbf{p}(x) J(x,y)}{\sum_x J(x,y)}\,, \tag{6}$$

where $\mathbf{p}(x)$ is the vector in the unit square specifying the position of neuron x in layer X.

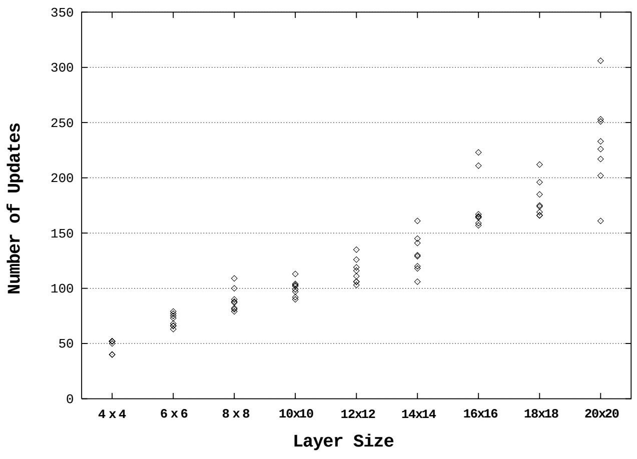

Fig. 2. Scaling behavior of the DLM algorithm on the mirror benchmark. The vertical spread of data points is due to different random seeds. A straight line is compatible with the data well, this time suggesting linear scaling with problem size.

4.2 Parameter tuning

The simulations of DLM on the mirror benchmark have shown that its parameters are fairly simple to adjust. There is no necessity to decrease the learning rate λ or change the blob size B during iterations in order to assure convergence. The form of the blob (circle, square, ...) does not influence convergence. J_0 serves only to prevent very small links from being suppressed for good, its value makes no difference. The only relevant parameter is the ***blob size*** B. Simulations in [8] have shown that convergence is fastest if a blob covers half the layer area. This blob size, which we used for all DLM simulations, can be shown to maximize the average information gain per iteration step (see [9] for details).

The experimental conditions have been identical to those for the SOM. The scaling behavior, however, was different. On variation of the problem sizes N the iteration steps needed for convergence increase only linearly with N (figure 2), in sharp contrast with the exponential behavior of the SOM (figure 1).

5 Results and conclusions

During our experiments we have encountered fewer difficulties in adjusting the parameters for DLM than for the SOM. Nevertheless, we have invested considerable effort to tailor both algorithms for the benchmark problem. Figures 1 and 2 show the number of iterations required to solve the mirror problem. The comparison of these two figures is not completely fair, because the single update steps for DLM are much more complicated ($O(N^4)$) than the ones for the SOM ($O(N^2)$). In order to show the actual execution time for concrete layer sizes we have plotted the number of floating point operations required to reach convergence in figure 3. This figure indicates that DLM converges faster once the layer size exceeds 16×16. We conclude that the convergence time for DLM will scale as N^4 and the one for the SOM as $\exp(N)$. Experiments that check this trend for higher values of N are currently performed and will be reported in [9].

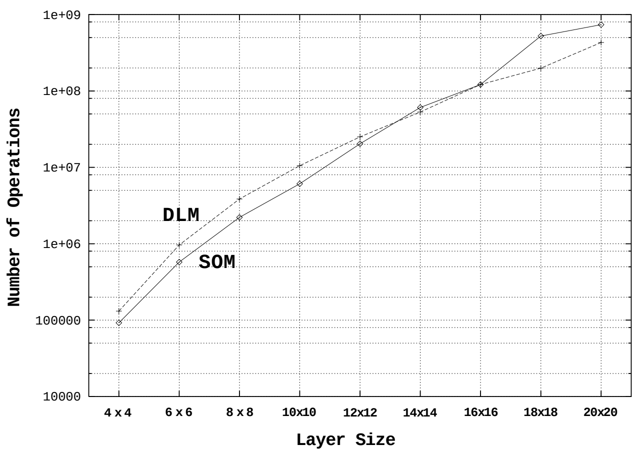

Fig. 3. Floating point operations required for the SOM and DLM on the mirror benchmark. The data indicate that DLM is faster for layers larger than 16 × 16.

A question that can not be ignored in the comparison of neuronal algorithms as models for perception is the time required on parallel machines. Given arbitrarily many parallel processors the single update steps can be executed in constant time for both algorithms. The update steps, however, can not be parallelized completely. To what extent partial parallelization (epoch learning) can be applied is unclear at this point. We therefore expect that on massively parallel machines DLM will scale linearly with the layer size, whereas the SOM will probably retain its exponential behavior.

References

1. R.P. Würtz. *Multilayer Dynamic Link Networks for Establishing Image Point Correspondences and Visual Object Recognition*, volume 41 of *Reihe Physik*. Verlag Harri Deutsch, Thun, Frankfurt am Main, 1995.
2. W. Konen and C.v.d. Malsburg. Learning to generalize from single examples in the dynamic link architecture. *Neural Computation*, 5:719–735, 1993.
3. T.J. Sejnowski, P.K. Kienker, and G.E. Hinton. Learning symmetry groups with hidden units: Beyond the perceptron. *Physica D*, 22:260–275, 1986.
4. D.J. Willshaw and C. v.d. Malsburg. How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society, London B*, 194:431–445, 1976.
5. W. Konen, T. Maurer, and C. v.d. Malsburg. A fast dynamic link matching algorithm for invariant pattern recognition. *Neural Networks*, 7(6/7):1019–1030, 1994.
6. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
7. T. Kohonen. The self-organizing map. *Proc. IEEE*, 78:1464–1480, 1990.
8. K.-O. Behrmann. Leistungsuntersuchungen des Dynamischen-Link-Matchings und Vergleich mit dem Kohonen-Algorithmus. Technical Report IR-INI 93-05, Ruhr-Universität Bochum, 1993.
9. R.P. Würtz, W. Konen, and K.-O. Behrmann. On the performance of neuronal matching algorithms. Manuscript in preparation.